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)

Дослідження методів налаштування архітектури та алгоритмів навчання динамічно масштабованої нейромережі

(тема)

Виконав:
студент ІІ курсу, групи КІТм-22-1
Вячеслав ГОЛУБНИЧИЙ
(власне ім'я, прізвище)

Спеціальність 123 Комп’ютерна інженерія
(код і повна назва спеціальності)

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

Освітня програма Комп’ютерні інтелектуальні технології
(повна назва освітньої програми)

Керівник проф. каф. КІТС Олег Руденко
(посада, власне ім'я, прізвище)

Допускається до захисту

Зав. кафедри Олег РУДЕНКО

(підпис) (власне ім'я, прізвище)

2023 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)
«_____»_____202_ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Голубничому В. А.
(прізвище, ініціали)

1. Тема роботи (проекту) дослідження методів налаштування архітектури та алгоритмів динамічно масштабованої нейромережі

затверджена наказом університету від « 03 » листопада 2023 р. № 1290Ст

2. Термін подання студентом роботи до екзаменаційної комісії 11.01.2024

3. Вхідні дані до роботи (проекту)
Програму розроблено на персональному комп'ютері з процесором на базі архітектури x86-64 та операційною системою Windows 10. Для написання програми використана мова C++, бібліотеки STL та cstdlib (стандартна бібліотека C). Код програми написаний, налагоджений, а також отримані тестові результати роботи програми за допомогою середовища розробки Microsoft Visual Studio.

4. Перелік питань, що потрібно опрацювати в роботі
загальні принципи роботи нейронних мереж, аналіз задачі і проблеми, пошук існуючих досліджень та рішень, планування та розробка системи, налагодження та тестування, опис розробленої системи, результати експериментальних досліджень та їх аналіз, висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій схема структури башатошарового персептрону, схема зв'язків двох нейронів із результатом їх обчислення, схема зв'язку між нейронами двох сусідніх шарів, граф-схема алгоритму роботи штучної імунної системи, граф-схема алгоритму генерації ваг, схема алгоритму пошуку найкращого рішення, схеми масштабування нейромереж, UML-блок класу NNetwork, код конструктору класу NNetwork, код класу шару Layer, код обчислення значень власних нейронів шару, UML-діаграма зв'язків між класами матриці, шару та нейромережі, графік функції активації, код обчислення нейромережею вхідних даних, код обчислення гіперболічного тангенсу, графік функції для тестування

6. Консультанти розділів роботи (п.6 включається до завдання за наявністю консультантів згідно до наказу, зазначеному 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	Отримання завдання	06.11.2023	виконано
2	Пошук та ознайомлення з джерелами	07.11.2023 - 10.11.2023	виконано
3	Аналіз знань та пошук рішень	10.11.2023 - 17.11.2023	виконано
4	Вибір методів вирішення задачі	17.11.2023 - 18.11.2023	виконано
5	Планування структури системи	18.11.2023 - 25.11.2023	виконано
6	Проектування та налагодження системи	26.11.2023 - 26.12.2023	виконано
7	Оформлення пояснювальної записки	26.12.2023 - 23.01.2024	виконано
8	Захист роботи	25.01.2024	виконано

Дата видачі завдання 06 листопада 2023 р.

Студент _____
(підпис)

Керівник роботи _____ професор Олег Руденко
(підпис) (посада, ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 57 с., 16 рис., 3 табл., 1 дод., 8 джерел.

НЕЙРОННА МЕРЕЖА, МАСШТАБ, РОЗМІРНІСТЬ, ДИНАМІЧНА, АДАПТАЦІЯ, КІЛЬКІСТЬ ШАРІВ, ОПТИМІЗАЦІЯ, ЕКСПЕРИМЕНТ, ШТУЧНА ІМУННА СИСТЕМА

Метою кваліфікаційної роботи є розробка моделі та програмної реалізації динамічно масштабованої нейронної мережі, тестове тренування та перевірка його якості за допомогою не тренувальних даних.

У роботі пропонується принцип організації нейронної мережі, а також програмна реалізація алгоритмів функціонування згідно пропонованим принципам, що дозволяє гнучко налаштовувати розміри (кількість шарів та кількість навчальних ітерацій) та проводити дослідження пов'язані з нейромережами різних розмірів. У ході виконання роботи досліджені наявні знання та досягнення у побудові сучасних нейромереж, програмно реалізовані алгоритми їх побудови та розроблені методи ініціалізації нейромережі у залежності від параметрів її розмірів. Проведене тестування отриманої програмної реалізації шляхом тренування мережі та перевірка якості навчання на даних, які не були використані при тренуванні.

Проект складається зі схем взаємодії функціональних блоків програмного забезпечення (UML-діаграм), граф-схем алгоритмів, які запрограмовані у принципи функціонування програмного проекту, та власне сам програмний проект (програмне забезпечення). Програмне забезпечення складається з функціональних блоків (класів об'єктів), що реалізують окремий функціонал, і співпрацюють за програмованим сценарієм з метою отримання результатів дослідження, що проводиться у цій роботі.

Для виконання роботи використаний програмно-експериментальний підхід, завдяки чому вдається ефективно перевірити гіпотези, що виникають при створенні

моделі, шляхом їх програмування та тестування. Даний підхід є економічно вигідним, оскільки теоретична модель формалізується математично для можливості програмування, а комп'ютерне моделювання дозволяє зекономити час на тестуванні цих теоретичних гіпотез.

Об’єктом дослідження є явище залежності якості навчання нейромережі від її розмірів, через те, що нейромережі різних розмірностей здатні реєструвати закономірності різних рівнів комплексності. Предметом дослідження є якість навчання нейромережі та її залежність від розмірів нейромережі. Якість навчання у даній роботі виражається як відповідність результатів роботи нейромережі після обчислення тестових вхідних даних до правильних відповідей до цих вхідних даних. У результаті виконання роботи розроблено програмне забезпечення, що реалізує функціонал нейромережі та імунної системи для її навчання та тестування, а також отримані статистичні дані щодо впливу кількості шарів та кількості навчальних ітерацій на якість навчання нейромережі. Дані проаналізовані та зроблені відповідні висновки щодо досліджуваних впливів та закономірностей.

Отримані результати дослідження можуть бути використані у майбутніх дослідженнях, що пов'язані з нейронними мережами, а також у прикладному програмному забезпеченні з їх використанням, для встановлення оптимальних значень розмірності нейронної мережі. Отримані дані дозволяють ефективніше обирати початкову розмірність нейромережі, що сприяє отриманню більш якісних даних та економить час, ефективно спрощуючи та здешевшуючи проведення досліджень у технологіях нейромереж.

ABSTRACT

Explanatory note of qualification work: 57 pages, 16 figures, 3 tables, 1 appendix, 8 sources.

NEURAL NETWORK, SCALE, DIMENSION, DYNAMIC, ADAPTATION, NUMBER OF LAYERS, OPTIMIZATION, EXPERIMENT, ARTIFICIAL IMMUNE SYSTEM

The major goal of this work is the development of model and software implementation of a dynamically scaled neural network, test training and check its quality using non-training data.

The paper proposes the principle of neural network organization, as well as the software implementation of functioning algorithms according to the proposed principles, which allows flexible adjustment of dimensions (the number of layers and the number of train iterations) and conducting research related to neural networks of different sizes. During research and implementation, available knowledge and achievements in the construction of modern neural networks were investigated, algorithms for their construction were implemented in software, and methods of neural network initialization were developed depending on the parameters of its size. The received software implementation was tested by training the network and the quality of training was checked on data that was not used during training.

The project consists of UML-diagrams showing interaction of functional software blocks, graph-diagrams of algorithms that are programmed into the principles of functioning of the software project, and the actual software project itself (software). The software consists of functional blocks (classes of objects) that implement a separate functionality and cooperate according to the programmed scenario in order to obtain the results of the research conducted in this work.

A program-experimental approach was used to perform the work, thanks to which it

is possible to effectively test the hypotheses that arise during the creation of the model by programming and testing them. This approach is economically beneficial, since the theoretical model is formalized mathematically for the possibility of programming, and computer modeling allows you to save time on testing these theoretical hypotheses.

The object of research is the phenomenon of the dependence of the quality of neural network learning on its size, due to the fact that neural networks of different dimensions are able to register patterns of different levels of complexity. The subject of research is the quality of neural network training and its dependence on the size of the neural network. The quality of training in this work is considered as the correspondence of the results calculated by neural network using the test input data to the benchmark answers. As a result of the work, software was developed that implements the functionality of the neural network and the immune system for its training and testing, as well as obtained statistical data on the influence of the number of layers and the number of training iterations on the quality of neural network learning. The data was analyzed and appropriate conclusions were drawn regarding the studied influences and patterns.

The obtained research results can be used in future research related to neural networks, as well as in application software using them, to establish optimal values of the dimension of the neural network. The obtained data make it possible to more effectively choose the initial dimension of the neural network, which contributes to obtaining higher quality data and saves time, effectively simplifying and reducing the cost of conducting research in neural network technologies.

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

Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження методів налаштування архітектури та алгоритмів навчання динамічно масштабованої нейромережі

(тема)

Виконав:
студент ІІ курсу, групи КІТм-22-1
Вячеслав ГОЛУБНИЧИЙ
(власне ім'я, прізвище)

Спеціальність 123 Комп’ютерна інженерія
(код і повна назва спеціальності)

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

Освітня програма Комп’ютерні інтелектуальні технології
(повна назва освітньої програми)

Керівник проф. каф. КІТС Олег Руденко
(посада, власне ім'я, прізвище)

2023 р.

АНОТАЦІЯ

Голубничий В.А. Дослідження методів налаштування архітектури та алгоритмів навчання динамічно масштабованої мережі. – Магістерська кваліфікаційна робота.

У магістерській кваліфікаційній роботі вирішено актуальну проблему алгоритмізації масштабування розмірності нейронної мережі за кількістю шарів та кількістю нейронів у них.

Метою даної роботи є створення програмної системи, а саме нейромережі та допоміжних програмних модулів для проведеня дослідження щодо впливу розмірності нейронної мережі на якість її навчання. Отримати дані та проаналізувати закономірність впливу розмірності нейромережі на якість її навчання, що виражена близькістю відповідей нейромережі на тренувальні вхідні дані до правильних відповідей.

Об’єктом дослідження є явище залежності якості навчання нейромережі від її розмірів, через те, що нейромережі різних розмірностей здатні реєструвати закономірності різних рівнів комплексності. Підвищення якості навчання нейромережі для виконання комплексних завдань потребує підвищення її розмірності, а, у той же час, підвищення розмірності нейромережі призводить до підвищення обчислювальних та/або часових витрат на навчання (тренування) нейромережі. Тому, через вищевказані причини існує проблема вибору оптимальної розмірності мережі зі збереженням високого рівня якості навчання і, водночас, мінімізацією обчислювальних та часових витрат.

Предметом дослідження є якість навчання нейромережі та характер її зміни при змінах у розмірності нейромережі, а саме кількості її шарів та/або кількості нейронів у них, а також у залежності від різних алгоритмів її функціонування та навчання.

У ході проведення дослідження використаний метод порівняння значень якості навчання нейромережі при різних значеннях параметрів, що відрізняються від

одного експерименту до іншого. Це параметри витрат на навчання (кількість навчальних ітерацій), а також кількість шарів нейромережі. Умова досягнення поставленої мети – отримання даних, що відрізняються, про якість навчання нейромережі за різних заданих значень параметрів кількості шарів та різних витратах на навчання.

Новизна роботи полягає у вивченні, програмній імплементації та дослідженні алгоритмів масштабування нейронної мережі. Дані алгоритми розроблені та досліджені з метою підвищення економічної ефективності проведення майбутніх досліджень пов'язаних із нейромережами за допомогою вибору більш ефективних параметрів розмірності нейромережі від самого початку досліджень.

У першому розділі розглянуто актуальні рішення та методи організації нейронних мереж. Розглянуті наявні знання з архітектур нейромереж, а згорткових та рекурентних нейромереж, а також багатошарових персептронів. З особливою увагою до деталей розглянуто архітектуру багатошарового персептрону. Проаналізовані знання, що стосуються призначення кожного з видів шарів персептрону. Проаналізовані та формалізовані знання з принципів роботи даного виду нейромереж, а саме алгоритми організації шарів та нейронів у них, обробки вхідних даних, принципи взаємодії між нейронами двох сусідніх шарів. Дані принципи візуалізовано за допомогою формул та схем, що сприяє спрощеному сприйняттю викладених знань. Розглянуті призначення та різні підходи до реалізації функцій активації персептрону. Узагальнені сучасні знання, що стосуються організації, принципів роботи нейромереж, а також алгоритмів та методів їх навчання. Розглянуті наявні знання з теорії штучних імунних систем, які допомагають у реалізації проекту роботи. У скороченому вигляді викладені наявні знання з теорії штучних імунних систем, що використовуються при реалізації програмної частини проекту. Проаналізований загальний алгоритм та принцип функціонування штучної нейронної мережі. Описані кожний з етапів алгоритму функціонування штучної імунної системи, а також різноманітні підходи та алгоритми, що використовуються на різних етапах його роботи. Наприклад, різні підходи до етапу відбору рішень задачі (антитіл) для схрещування, як от вибір

найкращих N (елітарний), випадковий або турнірний відбір. Описаний алгоритм мутації антитіл, що надалі використаний у програмній частині проекту. Визначена мета роботи програмного продукту і, відповідно, мета роботи нейромережевої та штучної імунної систем.

У другому розділі зпроектовано план розробки системи: формалізовані принципи взаємодії шарів нейромережі та обчислення значень відповідних нейронів, побудовані граф-схеми алгоритмів, що будуть використані у програмному проекті, описані принципи роботи алгоритмів, що становлять новизну проекту. Визначено, що система матиме три типи шарів: вхідний, декілька внутрішніх та вихідний шар. Нейрони окремих шарів організовані в матриці, що спрощує формалізацію принципів взаємодії, алгоритмів обчислення та посилання на відповідні нейрони. З метою оптимізації проектування та обчислень, допущено, що при постійних параметрах конфігурації нейромережі кожний шар матиме однакову кількість нейронів, тобто матимуть матриці нейронів однакових розмірностей. Формалізовано рішення задачі нейромережі, а саме найкращі значення ваг між її шарами. Матриця найкращих значень ваг – є склад рішення ШІС. Також, формалізовано принцип роботи ШІС за допомогою граф-схеми алгоритму. ШІС функціонуватиме, роблячи встановлену кількість ітерацій пошуку найкращого рішення (матриць ваг для нейромережі). Одна ітерація складатиметься з генерації випадкових рішень, їх оцінки. відбору кращих, видалення найгірших та схрещування найкращих. Наданий детальний опис кожного з етапів роботи ШІС. Формалізовано алгоритм генерації початкових рішень методом створення випадкових значень вагових коефіцієнтів (випадкові рішення). Також, детально описаний алгоритм тренування нейромережі штучною імунною системою, а також формалізований алгоритм тренування у вигляді граф-схеми алгоритму. Є наявними тренувальні вхідні дані та правльні вихідні дані відповідно до них. Система подає вхідні дані до нейромережі та отримує відповідь від неї. Обчислює відстань вихідних даних нейромережі до еталонних вихідних даних. Дія повторюється для всіх записів вхідних даних і сума значень, обернених до цих відстаней, є якістю цього рішення. Після встановленої кількості ітерацій, до система знаходить

найкраще поточне рішення (не обов'язково ідеальне) і встановлює ці значення ваг до нейромережі. Також, описані методи зміни розмірності нейромережі: за кількістю шарів. за кількістю нейронів у шарах та комбінований метод. Розглянуті різні аспекти впливу масштабованості нейромережі на її показники якості навчання, а також на проектування та простоту або складність її налаштування. Встановлюється мета виконання роботи по програмній реалізації системи.

У третьому розділі наданий опис системи як вцілому, так і з розглядом окремих його складових. Описано стрктурну організацію програмного проекту у вигляді розбиття на об'єкти - функціональні модулі, а також розбиття кожного з модулів на два файли .h та .cpp з метою оптимізації проектування коду шляхом інкапсуляції внутрішнього механізму роботи у окремих модулях. Розглянуто загальну структуру даних та методів класу нейронної мережі NNetwork у вигляді блоку UML - діаграми. Наведено характеристику та принцип функціонування модулю, а також його роль у правильному функціонуванні системи. Розглянуто програмний код конструктору об'єкту, що надає розуміння про структурну організацію об'єкту шару Layer. Об'єкт класу Layer зберігає показчик на матрицю нейронів та показчик на матрицю ваг цього шару, а також флаг, що визначає чи є він вхідних шаром. У випадку, якщо шар є вхідним, значення матриці ваг відповідає значенням одиничної матриці. Завдяки цим значенням матриці ваг, вхідний шар приймає вхідні дані без перетворень. У випадку, якщо шар є прихованим (внутрішнім) – значення елементів матриці ваг генеруються випадково у діпапзоні [0; 1]. Далі розглянуто код обчислення значень нейронів окремого шару з використанням функції гіперболічного тангенсу. Зпочатку, метод перевіряє чи є вхідні значення нейронів попереднього шару матрицею-стовпцем, отже даними коректного формату. Результат обчислюється за допомогою перевантаженого оператору множення матриць, що відповідає матричному добутку. Кінцевий результат – обчислення гіперболічного тангенсу над кожним елементом матриці добутку матриці ваг на значення попереднього шару. У результаті програмування системи, згенеровано UML-діаграму зв'язків між об'єктами описаних класів. Описано реалізацію функції гіперболічного тангенсу за допомогою формули,

представлено графік функції у діапазоні аргументів [-1; 1], а також програмний код реалізації. Також, представлено технічні вимоги для запуску програми та інструкція користувача до розробленої системи.

Четвертий розділ присвячено звіту про результати експериментальних досліджень щодо впливу розмірності нейромережі, що встановлюється у залежності від параметрів numberOfLayers та кількості тренувальних ітерацій. Для тестування нейромережі було обрано перевірити якість тренування, засновану на дискретних значеннях неперервної функції у визначеному обмеженому діапазоні. Заради прикладу, взято довільну функцію, що не має постійного значення в діапазоні аргументів [0; 1] для більш комфортного тестування. У розділі вказана формула функції, що досліджується та її графік у вищевказаному діапазоні аргументів. Приведені еталонні (тренувальні) та майбутні тестові значення аргументів функції та відповідні значення функції від цих аргументів. У наступній таблиці приведені результати тренувань нейромережі при різних значеннях кількості шарів та кількості тренувальних ітерацій у різних їх комбінаціях. Приведені розрахунки статистичних даних щодо помилки (обернене значенню якості) тренування та тестування для різної кількості шарів. Результати експериментів, показники та статистичні дані проаналізовані, зроблені відповідні висновки. Отже, у результаті експериментів, найкращою комбінацією кількості шарів нейромережі та кількості ітерацій є кількість шарів = 2 та кількість ітерацій навчання = 4000, оскільки при даних значеннях нейромережа має найменші середні значення помилки тренування та тестування. Виокремлено також і залежність якості навчання нейромережі від кількості шарів, а саме, що при збільшенні кількості ітерацій, звісно, збільшується якість навчання (помилка зменшується). Тим не менш, помітно, що при збільшенні кількості шарів до 3, помилка тестування більшується, а після – стає спадати. Отже, можна зробити висновок, що найбільш оптимальним є значення кількості шарів = 6, оскільки зменшення кількості шарів призведе до збільшення помилки тестування, а збільшення кількості шарів призведе до зменшення помилки ціною стрімкого підвищення витрат на тренування.

НЕЙРОННА МЕРЕЖА, МАСШТАБ, РОЗМІРНІСТЬ, ДИНАМІЧНА, АДАПТАЦІЯ, КІЛЬКІСТЬ ШАРІВ, ОПТИМІЗАЦІЯ, ЕКСПЕРИМЕНТ, ШТУЧНА ІМУННА СИСТЕМА

Використані публікації керівника та співробітників кафедри, що становлять теоретичну базу роботи:

1. Руденко О., Бодянський Є. Штучні нейронні мережі : навч. посіб. Харків : Компанія СМІТ, 2006. 404 с.

2. Корабльов Н. М., Сорокина И. В. Адаптивные нечеткие модели идентификации нелинейных объектов на основе искусственных иммунных систем. Бионика Интеллекта. 2008. Т. 2, № 69. С. 125–131.

3. Руденко О., Безсонов О., Романюк О. Нейромережеве прогнозування часових рядів на основі багатошарового персептрона. Development Management. 2019. Том 5, № 1. С. 12. URL: https://doi.org/10.21511/dm.5(1).2019.03 (дата звернення: 09.11.2023).

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

НМ – нейронна мережа

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

GRU – рекурентний блок із вентилем (англ., Gated Reccurent Unit)

ШІС – штучна імунна система

ІНМС – імунна нейромережева система

ВСТУП

Нейронні мережі (НМ) – сучасний спосіб вирішення проблем, для яких не знайдено формального алгоритму рішення або вирішення класичними детермаінованими алгоритмами займає занадто багато часу через велике обчислювальне навантаження. Результат роботи нейронних мереж має випадковий характер через генерацію псевдовипадкових чисел, закладену в алгоритм їх функціонування. Однак, нейромережі – це можливість дізнатися як машина сприймає та які закономірності бачить у добре відомих нам матеріальних речах або абстрактних поняттях. Штучний інтелект і нейроні мережі створені та функціонують за моделлю людського мозку: нейронна мережа представляється як послідовність шарів, в кожному з яких знаходяться нейрони [1]. На основі цієї моделі створено багато різновидів мереж, із різними принципами функціонування, призначенням, внутрішньою організацією, масштабом, різними додатковими функціями тощо.

Масштабованість і, безпосередньо, розмір нейромережі (кількість шарів та кількість нейронів у них) впливає на її показники, а отже впливає на якість навчання та вирішення призначеної їй задачі. Мережі великого розміру мають можливість фіксувати більш складні закономірності. Це правило працює і в зворотньому напрямі: чим більш комплексна проблема, тим більш комплексних закономірностей потребує її вирішення - тому необхідна мережа більшої розмірності. Мережі занадто малого розміру можуть запам'ятати вхідні дані і показувати високі результати при тренуванні, але погано виконувати реальні (тестові) задачі.

Часто вибір оптимального розміру мережі - питання пошуку правильного балансу між складністю моделі та можливостями узагальнення для оптимальної якості навчання. Оскільки нейронна мережа має шари та певну кількість нейронів у кожному з них, буде очевидним припустити, що існують такі схеми масштабування мережі: за кількістю шарів, за кількістю нейронів у шарах або комбінована схема.

Тому, як покращення наявної моделі вибору розміру нейронної мережі, в рамках даної роботи, пропонується нейромережа з можливістю зміни масштабу (кількості нейронів у шарах) з метою пошуку оптимального масштабу для збільшення ефективності мережі.

Для дослідження залежності якості навчання нейронної мережі від її масштабу, розробимо програмну реалізацію нейронної мережі з можливістю налаштування. Під налаштуванням розуміється гнучке встановлення параметрів нейронної мережі: кількості шарів та нейронів у цих шарах. Подібна гнучкість полегшить проведення досліджень щодо впливу розмірів нейронної мережі на якість навчання, на результати роботи тощо. Також, у майбутньому пропонується створити програмну модель нейромережі, що масштабується динамічно, у ході навчання. Однак, ця мета пропонується для майбутніх проектів на основі даної роботи, оскільки потребує додаткових часових витрат на тестування, налагодження та експериментів, що наразі має невигідне співвідношення користь / витрати. Отже, реалізуємо нейронну мережу у вигляді блоків, що надають гнучкий інтерфейс для перевірки вхідних даних, правильності проміжних обчислень, результатів роботи системи. а також гнучкого програмного налаштування масштабу нейронної мережі для окремих експериментів.

1 АНАЛІЗ ЗНАНЬ

Існує декілька видів нейронних мереж. Кожний має свої переваги та недоліки, область використання, особливості у підходах до проектування, навчання та тестування, і, звичайно, архітектуру. Існують згорткові нейронні мережі – мережі, розроблені для вирішення проблем комп'ютерного зору, таких як класифікація зображень, виявлення об'єктів на них, що є корисним для виконання різних прикладних завдань. Ця архітектура складається з трьох основних рівнів: згорткового рівня, рівня об'єднання та повністю підключеного рівня. Згортковий рівень є найважливішим і складається з кількох рівнів залежно від складності розв’язуваної задачі. Перший рівень шару згортки підключений до сприйнятливих функцій вхідного зображення. Перший рівень згорткового шару захоплює низькорівневі функції: кольори, лінії, краї тощо. Рівень об’єднання зменшує розмір зображення від згорткового шару для ефективності обчислень. Повністю зв’язаний рівень з’єднанує кожний нейрон одного шару з усіма нейронами іншого шару. Згорткові мережі застосовуються для виявлення об’єктів у самокерованих автомобілях, розпізнавання облич у соціальних мережах, аналізу зображень для медицини тощо.

У додаток до згорткових мереж, існують рекурентні нейронні мережі. Рекурентна нейронна мережа — це архітектура нейронної мережі, яка дуже добре працює при обробці послідовних даних, таких як обробка природної мови та дані часових рядів. Рекурентний рівень приймає вхідні дані, обчислює результат та надсилає вихідні дані самому собі. Рекурентна нейронна мережа має властивість зтикатися з проблемами нестабільного градієнта при обробці довгих послідовностей. Для вирішення цієї проблеми у рекурентних мережах замість нейронів використовують клітини довгої короткочасної пам'яті (Long Short-term Memory, LSTM) та рекурентні блоки з ветилем (Gated Recurrent Unit, GRU). Рекурентні нейронні мережі використовуються у чат-ботах, перекладі, аналізі емоцій, інтелектуальній обробці тексту тощо [2].

Також існують персептрони з різною кількістю шарів. Багатошаровий персептрон, також класичні нейронні мережі, — це архітектура нейронної мережі, призначена для оцінки будь-яких безперервних функцій і може вирішувати проблеми, які є нелінійними. Багатошарові персептрони можуть відрізнятися конфігурацією, мати різні принципи роботи та функції обчислення значень нейронів, функції активації тощо. Загалом, структура багатошарового персептрона майже така ж, як і рекурентної нейронної мережі, різниця в тому, що багатошаровий персептрон не має зворотніх зв'язків. Багатошаровий персептрон складається з 3 основних рівнів, а саме вхідного, прихованого та вихідного. Вхідний рівень отримає вхідний сигнал для обробки та перешле його на прихований рівень. Прихований рівень — це головний обчислювальний механізм персептрону, який отримує вхідні дані від вхідного рівня та пересилає їх на вихідний рівень або навпаки під час зворотного поширення. У той же час, вихідний рівень виконує передбачення або класифікацію на основі вхідних даних, які передаються з прихованих шарів. Багатошаровий персептрон має широкий діапазон сфер використання: у класифікації шаблонів, передбаченні, розпізнаванні, наближенні [3] тощо.

1.1 Опис знань, використаних при побудові нейронної мережі

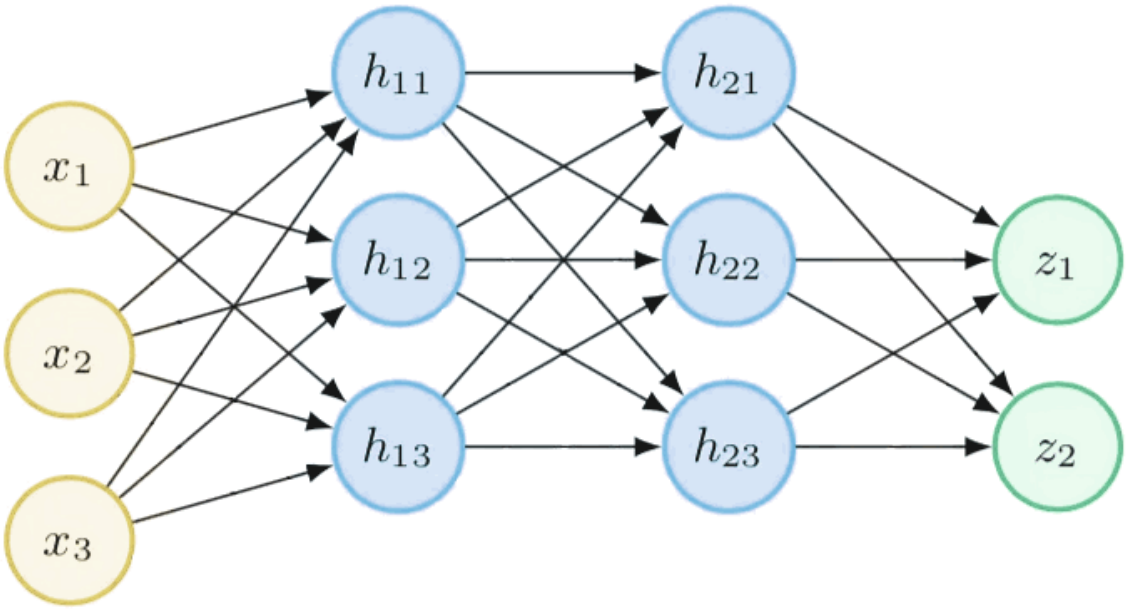


Рисунок 1.1 – Схема структури багатошарового персептрону

У нейронній мережі є 4 компоненти, а саме шар (група нейронів у рядку), нейрон (жовті кола на рисунку 1.1), вага (стрілка) і зміщення. Шари поділяються на 3 типи: вхідний шар, прихований шар(и) і вихідний шар. Нейрони є відповідно також 3-х типів: вхідні, приховані та вихідні. Стовпчик нейронів ліворуч є вхідним шаром, а праворуч — вихідним. Два стовпця нейронів між ними є частиною прихованого шару (на рисунку є 2 прихованих шару - прихованимі шари 1 і 2). Існує обчислювальний процес, який відбувається на всіх цих нейронах, крім вхідних нейронів.

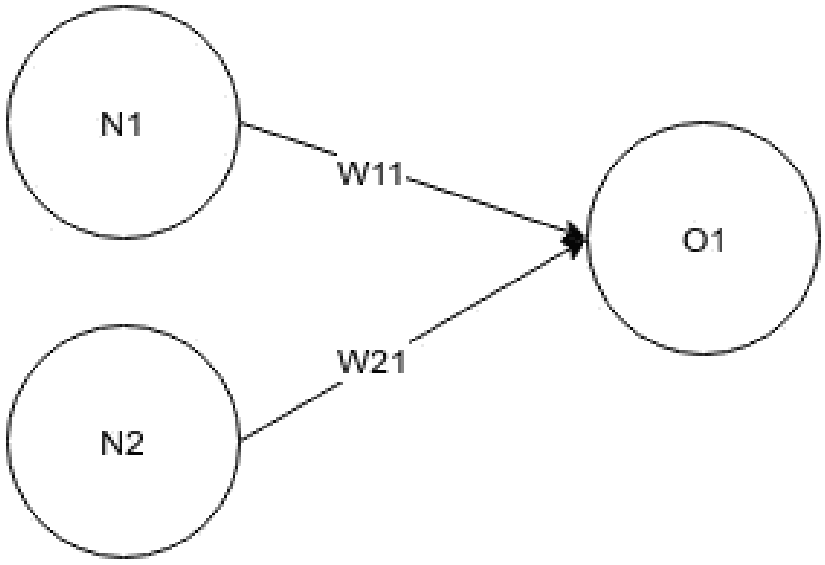


Рисунок 1.2 – Схема зв'язків двох нейронів N1 та N2 із результатом їх обчислення O

Значення набору предикторів передається на нейрон у напрямку стрілки (ваги) на зображенні. Значення N1 та N2 на рисунку 1.3 обчислюється за наступними формулами 1.1 та 1.2 відповідно

$$O_1 := N_1 * w_{11} + N_2 * w_{21} + b_1 \tag{1.1}$$

$$O_2 := N_1 * w_{12} + N_2 * w_{22} + b_2 \tag{1.2}$$

де O_i – значення і-го нейрону почного шару;

N_i – значення і-го нейрону попереднього шару;

w_{ij} – вага зв'язку і-го нейрону попереднього шару та j-того нейрону поточного шару;

b_i – значення зміщення і-го нейрону поточного шару.

Вхідний рівень — це набір числових значень. Із вхідного рівня ці значення передаються на наступний рівень (прихований шар), який їх обробляє. Значення передачі обчислюється за допомогою операції лінійної регресії. Тому маємо вагу (w) і упередженість (bias, b). Після отримання результатів ці значення обробляються

цими нейронами за допомогою рівняння, яке називається функцією активації. Зазвичай, для функцій активації використовуються сигмоїда (формула 1.3) або гіперболічний тангенс (формула 1.4).

$$sigm(x) = \frac{1}{1 + e^{-x}} \tag{1.3}$$

$$tahn(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{1.4}$$

Для отримання значень результату на вихідному рівні, останній прихований шар передає своє значення на вихідний рівень за допомогою операції лінійної регресії, і тоді це кінцевий результат роботи нейронної мережі.

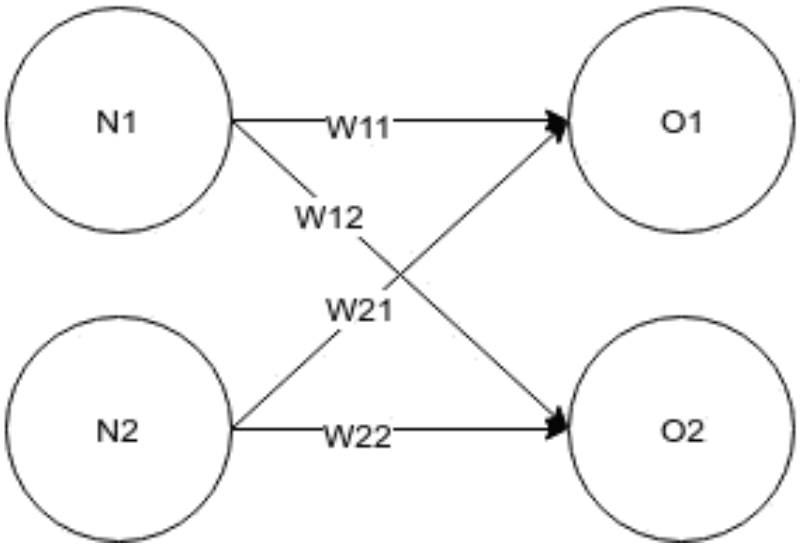


Рисунок 1.3 – Схема зв'язків між нейронами двох сусідніх шарів (у цьому прикладі кожний із шарів має по два нейрони)

Обчислення значень нейронів поточного шару O_j у зв'язку з нейронами попереднього шару N_i за допомогою вагових коефіцієнтів їх зв'язку W_{ij} може бути зображено у матричному виді

$$\begin{bmatrix} O_1 \\ O_2 \end{bmatrix} := \begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix} \begin{bmatrix} N_1 \\ N_2 \end{bmatrix} \quad (1.5)$$

де O_j – значення і-го нейрону почного шару;

N_i – значення і-го нейрону попереднього шару;

w_{ij} – вага зв'язку і-го нейрону попереднього шару та j-того нейрону поточного шару.

Отже, таким чином відбувається обробка вхідних даних у нейронній мережі та обчислення результату на виході. Даний алгоритм використовується як при тренуванні, так і тестуванні нейронної мережі.

При тренуванні нейронної мережі, на вхід подаються дані, для яких заздалегідь є обчислені правильні результати. Такі дані, на яких відбувається навчання нейронної мережі, називають навчальними. При обчисленні помилки передбачення нейронної мережі обчислюється різниця між заздалегідь обчисленим правильним значенням та отриманим результатом від нейромережі. При тестуванні, на вхід нейромережі подаються дані, яких не було в навчальному наборі, отримують результат від нейромережі, за допомогою алгоритму описаного вище, та перевіряють на результат на правильність.

1.2 Опис штучної імунної системи

Штучна імунна система (ШІС) – інтелектуальна адаптивна система, що за своїм алгоритмом функціонування імітує принципи роботи людської імунної системи. Вона складається із задачі, яку необхідно розв'язати (антиген), згенерованих можливих рішень задачі (антитіл), які формують собою популяцію та якість рішення (афінність). Якість рішення обчислюється за заздалегідь заданою формулою, що задана розробником системи та є математичною функцією розрахунку відповідності результату згенерованого рішення теоретичному ідеальному результату. Штучна імунна система може генерувати нові рішення

випадковим чином або методом перекрещування двох існуючих рішень з додаванням мутації та видаляти рішення з покоління.

Зпочатку, система випадковим чином генерує деяку кількість можливих рішень задачі та обчислює їх якість. Рішення, якість яких відрізняється не більше ніж на якесь порогове значення, видаляються. Це механізм клональної супресії або видаленням клонів. Далі, відсортувавши рішення за найвищою якістю, система обирає N найкращих рішень для схрещування (елітарний відбір). Також, існують різні алгоритми відбору, як от випадковий вибір або турнірний відбір. При турнірному відборі з популяції відбирається Q антитіл, які беруть участь у турнірі. Обирається антитіло з найкращою якістю серед цих Q обраних рішень. Обрані рішення можуть схрещуватися за різними алгоритмами. Наприклад, наступним чином: результуюче рішення складається на половину із першого та на половину з другого рішення. До результату додається мутація. Під мутацією розуміється вірогіднісна заміна одного або декількох складових рішення випадковими значеннями. Мутації є корисними для підтримки різноманітності популяції рішень. Для нових рішень обчислюється якість та присвоюється порядковий номер у сортуванні. Найгірші рішення видаляються. У подальшому схрещенні приймають участь найякісніші рішення задачі. Рішення, отримані у результаті чергової ітерації алгоритму називаються поколінням.

Вищеописаний алгоритм дії штучної імунної системи виконується декілька тисяч або сотень тисяч разів. Це процес еволюції рішень задачі. У результаті цього процесу система має найкраще (за оцінкою згідно функції обчислення якості) рішення задачі.

Метою тренування нейронної мережі є пошук найкращих значень ваг для для зв'язків між нейронами. Тоді, за умови вдало обраної функції обчислення якості, алгоритм штучної імунної системи можна дуже ефективно використати для пошуку найкращих значень цих ваг нейромережі.

2 ПРИНЦИПИ РОБОТИ СИСТЕМИ

У даному розділі описуються принципи функціонування імунної нейромережевої системи (ІНМС), що складається з нейромережевої частини, а саме нейромережі, а також імунної частини, що представлена імунною системою для знаходження значень матриці ваг нейромережевої частини. Нейромережева частина представлена нейромережею, що здатна отримувати матрицю-стовпець вхідних даних заздалегідь визначеної розміності. Результатом її роботи є матриця-стопець відповідного розміру, що представляє вихідні дані, тобто результати роботи нейромережі у залежності від вхідних даних. Імунна система представлена "вчителем" нейромережі – системою, що генерує рішення задачі нейромережі, а саме задачі знаходження значень матриці ваг, що сприятимуть правильності роботи нейромережі. Імунна система генерує популяцію за популяцією рішень задачі нейромережі шляхом першочергової генерації випадкових рішень і подальшому схрещенню найкращих рішень один з одним за алгоритмами, що описані далі. Також, описані математичні моделі та функції, використані для реалізації імунною системою поставленої задачі знаходження рішення (матриці ваг нейромережі) з метою оримання вагових коефіцієнтів найкращої якості. Якість рішення обчислюється шляхом тестуванню роботи нейромережі з ваговими коефіцієнтами поточного знайденого імунною системою рішення та перевіркою відповідності відповідей, наданих навченою нейромережею до еталонних відповідей, наданих імунній системі людиною у тренувальному наборі даних.

2.1 Використання отриманих знань у проектованій нейромережі

Для створення базової нейронної мережі використовується класична структура, що складається з декілької шарів: вхідного (x), внутрішніх прихованих (h) та вихідного шарів (z) із нейронами. Кожен шар має певну кількість нейронів. Для підвищення простоти та гнучкості програмування системи, а також збільшення

потенціалу для оптимізації, дані шарів (значення нейронів та ваг) зберігаються та обробляються у вигляді матриць. Обчислення значень нейронів кожного наступного прихованого шару виконуються за допомогою, відповідно, матричних операцій. Матриця ваг за допомогою індексів рядка та стовпця задає зв'язки між кожним нейроном попереднього шару та кожним нейроном поточного шару. Нейронна мережа отримує вхідні дані, у вигляді матриці-стовпця, на вхідний шар. Розрахунок значень нейронів кожного наступного шару, як і описано у принципі функціонування нейронної мережі, відбувається на основі значень нейронів із попереднього шару нейронної мережі та ваг, що з'єднують нейрони попереднього шару з даним. Фактично, розрахунок значень нейронів кожного чергового (із прихованих шарів) розраховуватиметься за наступною формулою:

$$\begin{bmatrix} N_{[l]\,0} \\ \vdots \\ N_{[l]\,i} \end{bmatrix} := \begin{bmatrix} w_{00} & \cdots & w_{0i} \\ \vdots & \ddots & \vdots \\ w_{i0} & \dots & w_{ii} \end{bmatrix} \begin{bmatrix} N_{[l-1]\,0} \\ \vdots \\ N_{[l-1]\,i} \end{bmatrix} \tag{2.1}$$

де $N_{[l]\,i}$ – значення і-го нейрону почного шару (l);

$N_{[l-1]\,i}$ – значення і-го нейрону попереднього шару $(l-1)$;

w_{ij} – вага зв'язку і-го нейрону попереднього шару та j-того нейрону поточного шару.

Рисунок 2.1 – Граф-схема алгоритму роботи ШІС

За описаними правилами функціонуватиме нейромережева частина системи (ІНМС). Імунна частина функціонуватиме згідно принципів, описаних вище. Отже, загальний алгоритм функціонування системи складатиметься з двох фаз: тренування та тестування. Фаза тренування, поперше, включає генерацію популяції рішень (рисунок 2.1). Кожне рішення – матриця вагових коефіцієнтів нейромережі. Ці початкові рішення згенеровані випадковим чином (рисунок 2.2).

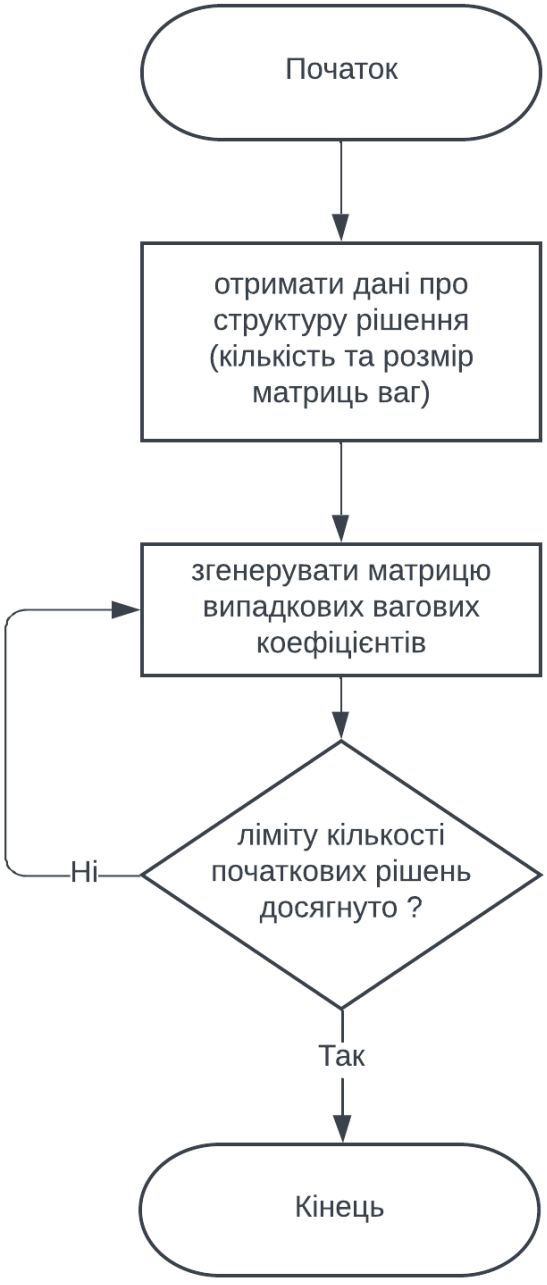


Рисунок 2.2 – Граф-схема алгоритму генерації випадкових значень ваг

Імунна нейромережева система (ІНМС) оцінює якість кожного рішення насупним чином: присвоює нейромережі згенеровані вагові коефіцієнти, подає тренувальні дані на нейромережу, отримує її відповідь та обчислює відстань відповіді нейромережі до правильної відповіді з тренувального набору даних. Тренувальний набір даних – набір даних, в якому задана відповідність вхідних даних очікуваним правильним результатам (відповідям). Чим більше відстань від правильної відповіді – тим менше значення якості рішення присвоюється цьому рішенню у ранжуванні. Система повторює цей алгоритм ітеративно, заздалегідь задану кількість разів, як показано на рисунку 2.3. Після цих ітерацій, система володіє поточним поколінням найкращих рішень, найвище серед яких має найкращу якість, тобто найменшу відстань до правильних результатів, і виводить на екран інформацію про нього.

Після виконання цього навчального алгоритму, нейромережа готова до тестування та використання. На нейромережу можна подавати тестові дані – набір даних, для яких немає заздалегідь заданих правильних відповідей. Навчена на тренувальних даних нейромережа дає свої припущення стосовно тестових (раніше незнайомих) даних, які розробник чи користувач може перевірити на правильність.

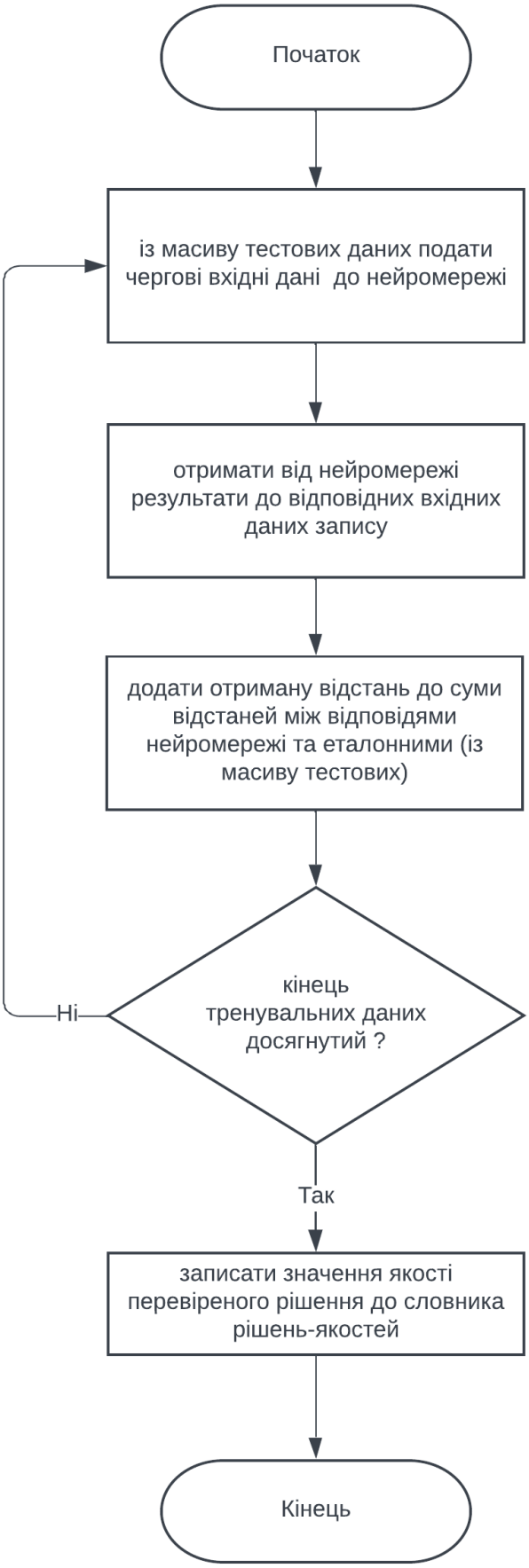
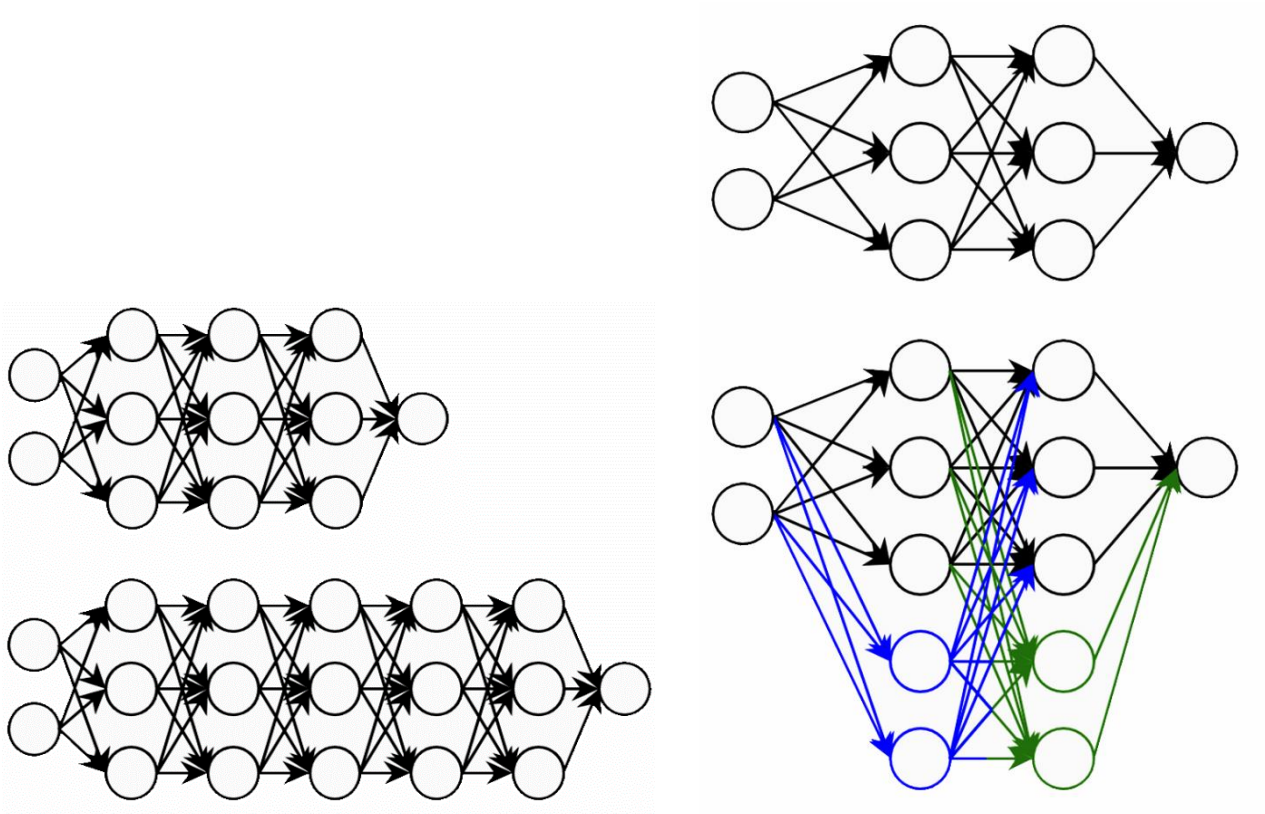


Рисунок 2.3 – Граф-схема алгоритму пошуку найкращого рішення

Часто вибір оптимального розміру мережі - питання пошуку правильного балансу між складністю моделі та можливостями узагальнення для оптимальної якості навчання. Тому, як покращення наявної моделі вибору розміру нейронної мережі, в рамках даної роботи, пропонується нейромережа з можливістю зміни масштабу (кількості нейронів у шарах) з метою пошуку оптимального масштабу для збільшення ефективності мережі.

Схеми масштабування нейронних мереж показано на рисунку 2.4.



(а) (б)

Рисунок 2.4 – Схеми масштабування нейронних мереж:
(а) – за кількістю шарів, (б) – за кількістю нейронів у шарах.

Оскільки робота фокусується на масштабованості нейромереж, розглянуто те, як розмір мережі, а саме, кількість шарів та кількість нейронів у кожному шарі впливає на її показники:

- більший розмір мережі може дати їй можливість фіксувати більш складні закономірності;

- складність проблеми: чим більш комплексна проблема, що вирішує нейромережа, тим більша розмірність є необхідною.

Модель нейронної мережі може легко змінювати кількість шарів з нейронами та кількість нейронів у шарах. Це підвищує гнучкість системи. Простота налаштування системи дозволяє стверджувати про:

- ієрархічна структура: допомагає системі знаходити ієрархічні моделі в досліджуваних або тестових даних;

- нелінійне перетворення: нелінійні функції активації та велика кількість прихованих шарів дає системі змогу знаходити нелінійні закономірності в даних;

- трансферне навчання: мережі глибокого навчання часто використовують трансферне навчання (transfer learning). Трансферне навчання – використання попередніх рівнів та тонке налаштування наступних рівнів мережі для оптимізації до конкретного набору даних чи завдання.

До того ж, гнучкість у масштабуванні нейронної мережі дозволяє підвищити швидкість навчання та досягнення оптимальної якості порівняно з перенавчанням іншої моделі. Додатково, масштабований підхід економить приблизно 50 % обчислювальної роботи [4].

Запропонований метод організації та роботи нейронної мережі дає можливість пристосовуватися до завдань різної складності, підвищити ступінь інтелектуальності системи шляхом регістрації більш складних взаємовідносин між даними, а також оптимізація часових і обчислювальних витрат. Дані факти дають можливість стверджувати про перспективність даного напрямку досліджень у покращенні нейронних мереж та інтелектуальних системах взагалі.

Отже, створене програмне забезпечення має тестувати нейромережеву систему при різних вхідних значеннях параметрів: кількості шарів з нейронами та кількості навчальних ітерацій, отримати та відобразити результати дослідження впливу цих значень на якість (обернене значення помилки) результатів тестування нейромережі. Виконати навчання мережі відтворенню функції, зібрати та відобразити зібрані дані дослідження.

3 ОПИС РОЗРОБЛЕНОЇ СИСТЕМИ

У результаті роботи над системою із запропонованим методом функціонування, мовою C++, розроблені класи для створення відповідних функціональних об'єктів (модулів) системи. Реалізація окремих модулів програми розміщується в різних директоріях проекту з відповідними назвами. Реалізація модулів розбита на заголовний файли *.h із об'явленнями макросів, типів та методів та файли *.cpp із реалізаціями необхідних для роботи системи функцій [5].

Розглянемо реалізацію класу нейронної мережі NNetwork. UML-блок класу NNetwork показано на рисунку 3.1. Клас дає можливість створювати об'єкт нейромережі, що автоматично створює необхідну кількість шарів з матрицями ваг та надає методи встановлення значень для цих матриць з вагами (для певного шару, зазначеного користувачем), виконує обчислення одного набору вхідних даних, що включає передачу вхідних даних на вхідний шар, обчислення прихованих шарів і вихідного шару з використанням значень попередніх шарів, а також відображення інформації про виконання обчислень. На даний момент, нейромережа надає розробнику можливість зручно змінювати параметри конфігурації нейронної мережі. Мережа підтримує зміну кількості шарів, відповідно до чого змінюються розміри матриць, кількість входів та виходів. Також, звісно, клас надає метод для виведення інформації про обчислення на екран.

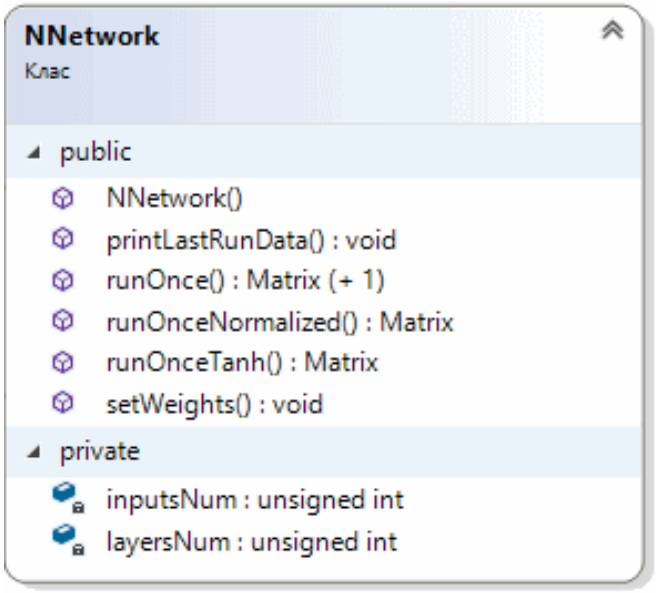


Рисунок 3.1 – UML-блок класу NNetwork

Отже, клас має декілька методів для створення та роботи з нейромережею. Перший серед них – NNetwork (конструктор), який показано на рисунку 3.2. Реалізований конструктор класу нейромережі, який дає можливість створювати нейромережі з різною кількістю входів та шарів. Шари нейромережі організовані у масив та обробляються згідно їх послідовному розташуванню у ньому. Виходячи з коду конструктора, очевидно, що кожний шар має кількість нейронів, що дорівнює значенню кількості входів нейромережі (inputsNum), параметру, що передається у конструкктор при його виклику. Логіка створення окремого шару та його вгутрішня організація регламентується внутрішньо, класом шару Layer. Конструктор нейромережі наявно демонструє структуру нейромережі: вона складаєтьсґя з вхідного шару (змінна inputLayer) та масиву внутрішніх шарів, останній з яких вважається вихідним шаром (змінна layers). При даній реалізації, спрощується керування вхідним шаром та прихованими шарами окремо.

```
18
19  class NNetwork
20  {
21      unsigned int inputsNum = 0;
22      Layer* inputLayer;
23      Layer* layers;
24      unsigned int layersNum = 0;
25
26      public:
27      /// ...
36      NNetwork(const unsigned int inputsNum, const unsigned int layersNum)
37          :
38          inputsNum(inputsNum),
39          layersNum(layersNum)
40          {
41          inputLayer = new Layer(inputsNum, true);
42          layers = new Layer[layersNum] {};
43          for (unsigned int i = 0; i < layersNum; i++)
44          {
45              layers[i] = Layer(inputsNum);
46          }
47      }
48
```

Рисунок 3.2 – Код конструктору класу нейромережі NNetwork

При створенні масиву шарів нейромережі, організація та алгоритм створення окремого шару імплементований у конструкторі класу Layer. Згідно з проектом системи, шар зберігає дві матриці: матрицю значень нейронів та матрицю вагових коефіцієнтів. Вагові коефіцієнти є 64-бітними числами з плаваючою точкою і можуть приймати значення в діапазоні [0; 1]. Також, оскільки нейромережа має вхідні та внутрішні шари, їх функції дещо відрізняються. Вхідний шар встановлює значення нейронів напряму з входів, проте внутрішній шар встановлює значення нейронів як результат матричного множення матриці ваг і матриці входів. Для їх розрізнення введена змінна-поле класу isInputLayer. Ця змінна відповідає за те, чи є цей шар вхідним. Якщо вона має значення True (1) – матриця ваг є одиничною та не впливає на перетворення вхідних значень. Натомість, якщо це значення False (0) – вагам присвоюються випадкові значення у відповідному діапазоні.

За встановлення значень вагових коефіцієнтів відповідає метод

randomiseMat01Floor (у випадку внутрішнього шару) або getIdentityMatrix (для вхідного шару). Метод randomiseMat01Floor встановлює значення матриці ваг внутрішнього шару у випадкові значення в діапазоні [0; 1]. Метод getIdentityMatrix повертає матрицю з одиничними коефіцієнтами на головній діагоналі, ефективно вимикаючи функцію матриці ваг на вхідному шарі нейромережі.

3.1 Організація об'єкту шару Layer

На рисунку 3.3 наведено код полів класу шару нейромережі (Layer) та його конструктору.

```
// ...
class Layer
{
    protected:
    Matrix* N = nullptr;    // values
    Matrix* w = nullptr;    // weights[thisLayerNeuronIndex][leftLayerNeuronIndex]
    bool isInputLayer = false;

    public:
    Layer() {}

    Layer(const unsigned int leftLayerNeuronsNum, bool isInputLayer = false)
        :isInputLayer(isInputLayer)
    {
        if (!isInputLayer)
        {
            N = new Matrix(leftLayerNeuronsNum, 1);
            w = new Matrix(leftLayerNeuronsNum, leftLayerNeuronsNum);
            randomiseMat01Floor(*w);
        }
        else
        {
            unsigned int inputsSize = leftLayerNeuronsNum;
            N = new Matrix(inputsSize, 1);
            w = new Matrix(Matrix::getIdentityMatrix(inputsSize, inputsSize));
        }

    }
```

Рисунок 3.3 – Код полів класу шару нейромережі (Layer) та його конструктору

На рисунку 3.4 показано код методу обчислення значень власних нейронів шару класу Layer.

```
/// ...
void calcMeTanh(const Matrix& leftLayerValues, double steepness)
{
    if (w->cols() != leftLayerValues.rows())
    {
        throw std::logic_error("leftLayerValues matrix is not [NeuronsNum x 1].
    }
    *N = (*w) * leftLayerValues;
    matop::tanhCustom(*N, steepness);
}
```

Рисунок 3.4 – Код методу обчислення значень власних нейронів шару класу Layer

Кожен шар складається з матриці значень нейронів і матриці ваг. Кожен прихований шар під час створення заповнюється випадковими вагами в діапазоні [0; 1]. Після виклику методу calcMeTanh для поточного шару нейромережі, перевіряється коректність розмірності матриці попереднього шару, що передається до функції параметром leftLayerValues. Далі обчислюється значення нейронів поточного шару як матричний добуток матриці ваг (w) на матрицю значень нейронів попереднього шару (шару зліва, leftLayerValues). Для матриці отриманих значень обчислюється гіперболічний тангенс tanh (кінцевий результат розрахунків всередині шару). Обчислення гіперболічного тангенсу відносно матриці ваг означає перезапис кожного елементу матриці значенням гіперболічного тангенсу відповідного значення матриці.

На рисунку 3.5 показано UML-діаграму класів матриці Matrix, шару Layer та нейромережі NNetwork.

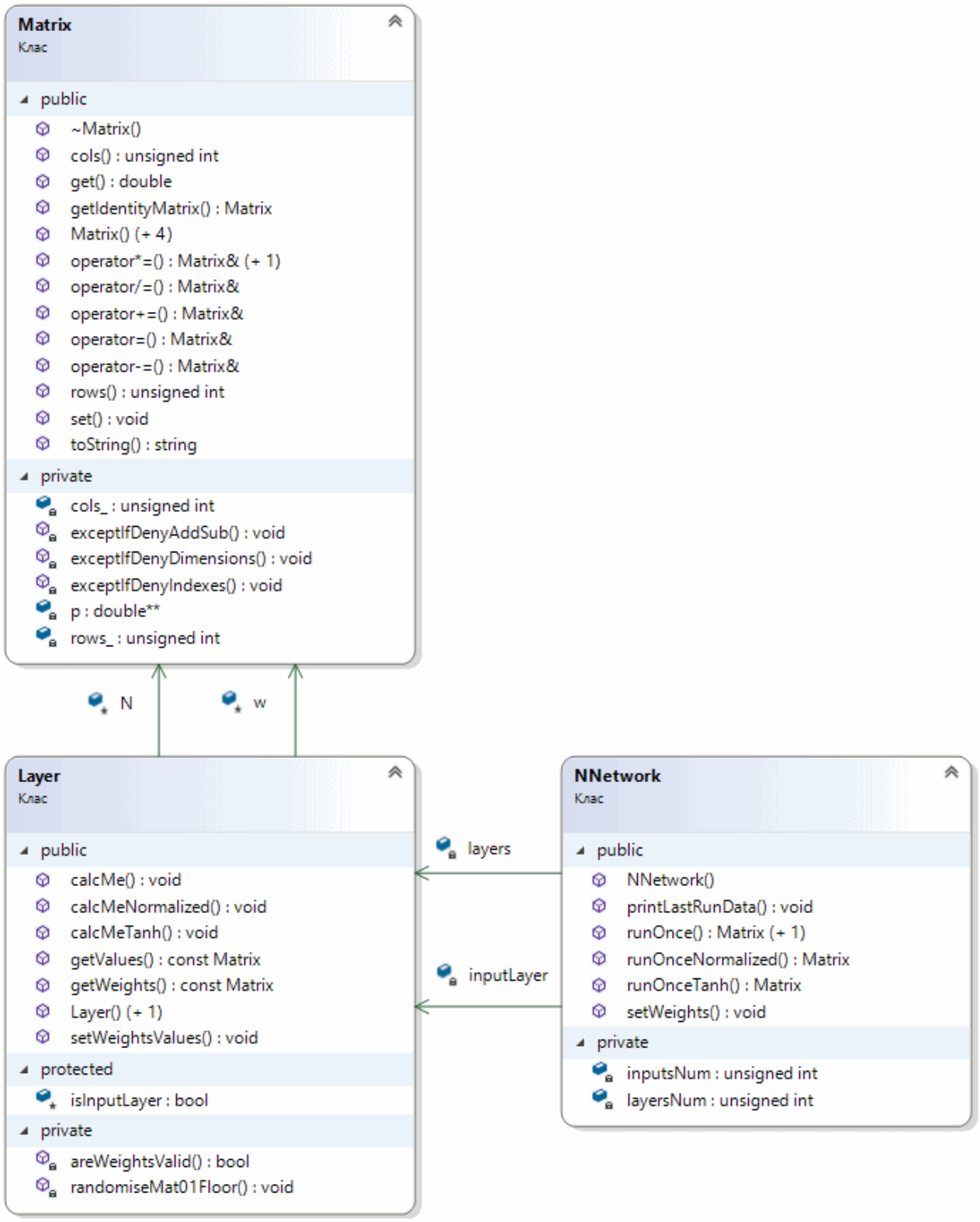


Рисунок 3.5 – UML-діаграма класів матриці Matrix, шару Layer та нейромережі NNetwork.

У коді функції обчислення нейронів поточного шару (функції активації), відбувається обчислення гіперболічного тангенсу, показаного на рисунку 3.6, за

допомогою виклику функції matop::tanhCustom() – функції простору імен matop (математичні операції – англ., math operations). Розроблена процедура обчислення має декілька вхідних параметрів: аргумент та параметр кривизни та обчислюється за формулою:

$$y(x) = \frac{e^{\pi * s * x} - e^{-\pi * s * x}}{e^{\pi * s * x} + e^{-\pi * s * x}} \quad , \tag{3.1}$$

де x – аргумент;

s – коефіцієнт крутості зростання функції.

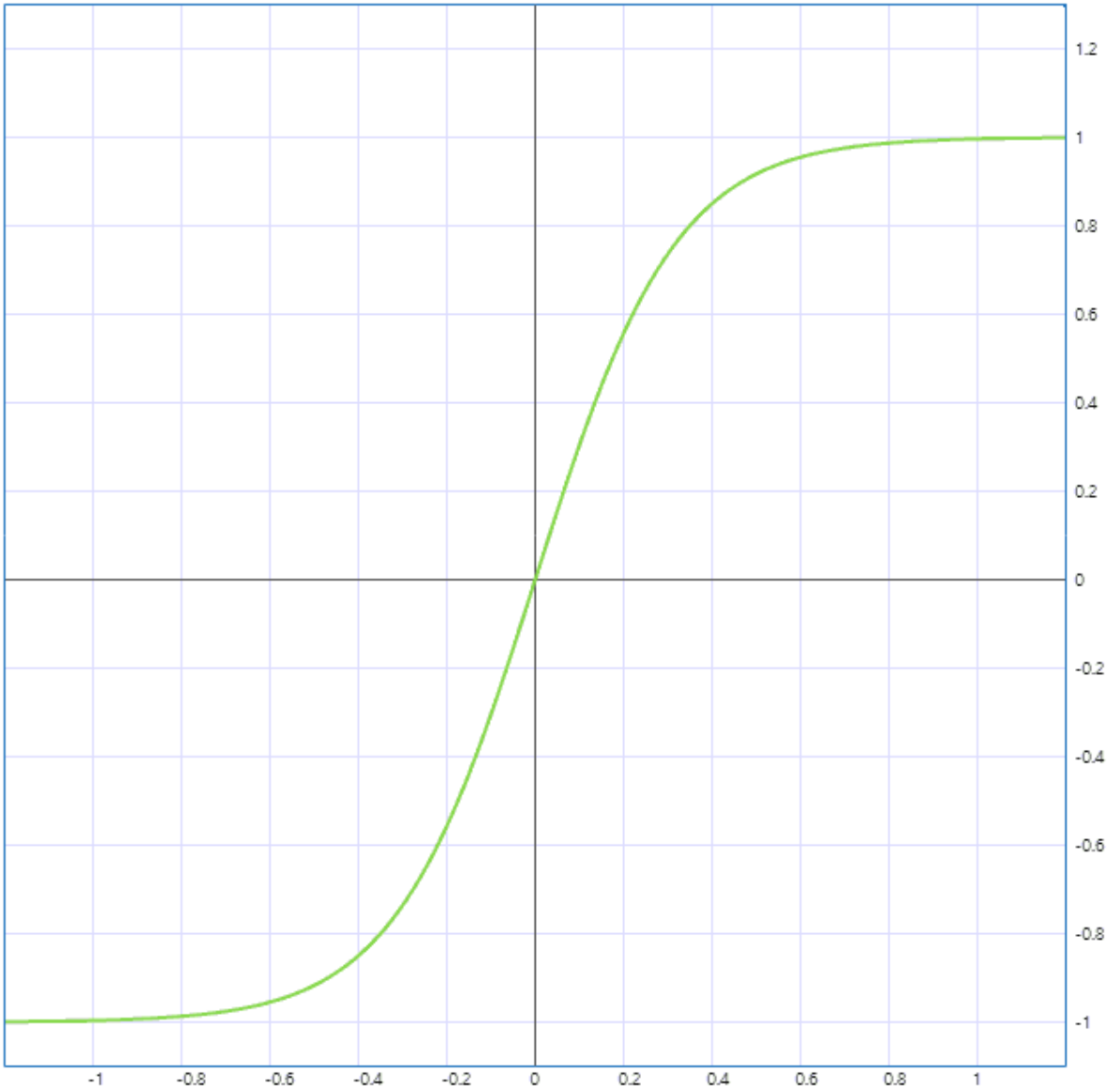


Рисунок 3.6 – Графік функції активації нейронів шару (формула 3.1).

На рисунку 3.7 показано код методу обчислення одиночного набору вхідних даних нейромережею.

```
Matrix runOnceTanh(const Matrix& xColumn, double steepness = 3.14159)
{
    if (inputsNum != xColumn.rows())
    {
        throw std::exception("Number of inputs (1-column rows) must be equal to inputsSize
    }
    if (!matop::isPositive(xColumn))
    {
        throw std::invalid_argument("Negative inputs are not yet supported!");
    }

    inputLayer->calcMe(xColumn);
    layers[0].calcMeTanh(inputLayer->getValues(), steepness);
    for (unsigned int i = 1; i < layersNum; i++)
    {
        layers[i].calcMeTanh(layers[i - 1].getValues(), steepness);
    }
    return layers[layersNum - 1].getValues();
}
```

Рисунок 3.7 – Код методу обчислення одиночного набору вхідних даних нейромережею

Метод runOnceTanh, поперше, виконує перевірку коректності вхідних даних, а саме перевіряє розміри матриці входів та її значення. Допускається лише матриця-стовпець із значеннями більше нуля. Далі функція обчислює значення нейронів у шарах мережі шляхом послідовного обчислення значень нейронів у кожному з шарів зліва направо (від вхідного до вихідного). Як можна побачити, метод runOnceTanh робить це шляхом виклику методу calcMeTanh для кожного з шарів, куди у якості аргументу преедаються значення нейронів попереднього шару (i-1). Метод calcMeTanh належить класу Layer (клас шару нейромережі).

Код методів обчислення гіперболічного тангенсу показано на рисунку 3.8.

```
95
96      /// ...
100     double tanhCustom(const double x, const double steepness)
101     {
102         double p = steepness * x;
103         return
104             (pow(e, p) - pow(e, -p)) /
105             (pow(e, p) + pow(e, -p));
106     }
107
108     /// ...
112     void tanhCustom(Matrix& mat, double steepness)
113     {
114         for (unsigned int i = 0; i < mat.rows(); i++)
115         {
116             for (unsigned int j = 0; j < mat.cols(); j++)
117             {
118                 mat.set(i, j, tanhCustom(mat.get(i, j), steepness));
119             }
120         }
121     }
122
```

Рисунок 3.8 – Код методів обчислення гіперболічного тангенсу

Отже, згідно коду методу обчислення гіперболічного тангенсу, реалізовано два методи. Один для обчислення одного значення функції від аргументу з параметром кривизни зростання функції, згідно формули (3.1). Другий метод, похідний від першого, - для виконання множини обчислень на матриці аргументів із записом результатів у матрицю-джерело. Ці обчислення виконуються поелементно, шляхом передачі чергового елементу матриці у функцію гіперболічного тангенсу, а результат записується до елементу матриці, що слугував джерелом. Дані методи дають можливість легко та гнучко налаштовувати алгоритм обчислень нейронної мережі, а також дають можливість повторного використання коду у даному проекті та за його межами.

3.2 Технічні вимоги та інструкція користувача

Мінімальні технічні вимоги:

- OS: Windows 7 або вище;
- CPU: Intel Pentium Dual-Core / AMD-еквівалент або краще;
- RAM: 1 ГіБ;
- жорсткий диск: 200 Мб вільного місця;
- пристрої введення: миша;
- пристрої виведення: екран.

Інструкція користувача.

Після запуску програми відкривається вікно, що відображає статус тренування та тестування нейронної мережі. Алгоритм починає виконуватися автоматично. Дочекайтеся результатів тренування та тестування системи. У випадку кліку по клієнтській області вікна, виведення текстової інформації буде припинено і дані не будуть виведені! Після завершення тренування, нейронна мережа автоматично завантажить найкращий варіант вагових коефіцієнтів та буде протестована.

Після тестування на екрані з'являться результати тренування і тестування: значення помилок при тренуванні і тестуванні та розширена інформація про процес обчислення нейронів та дані, що зберігаються у кожному з шарів нейромережі.

4 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

В експерименті щодо того, як масштаб (numberOfLayers) нейронної мережі та кількість тренувальних ітерацій впливає на якість навчання, виконується серія створення та запуску нейронної мережі. Навчання та тестування мережі здійснено на основі математичної функції:

$$y(x) = \sqrt[3]{x} - x \ , \tag{4.1}$$

графік функції якої наведено на рисунку 4.1.

Значення параметрів функції має своє призначення: завдяки їм, функція має вигляд синусоїди у діапазоні аргумента (x) та результату (y).

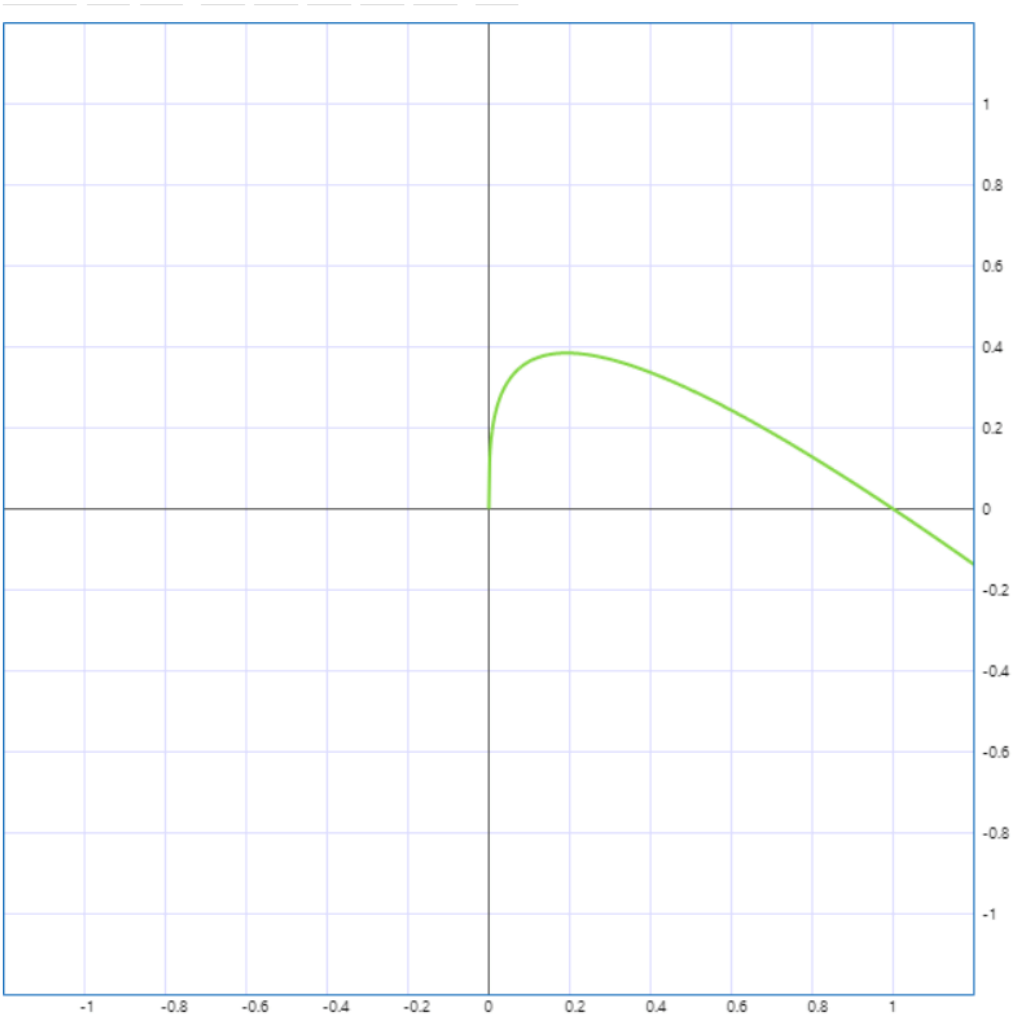


Рисунок 4.1 – Графік функції для тестування (формула 4.1).

Для обчислення якості та тренування нейронної мережі необхідні еталонні значення відповідей на тренувальний набір даних. Тренувальний набір даних створено із декількох дискретних аргументів функції, котрій навчається нейромережа. Для кожного значення аргументу обчислене еталонне значення функції. Нейромережі доступні тільки значення аргументів. Навчальній системі доступні аргументи та еталонні відповіді для перевірки якості навчання. Попередньо обчислені значення функції для тренувальних та тестових значень аргументів.

Таблиця 4.1 – Відповідність еталонних відповідей тренувальним та тестувальним вхідним даних (аргументам)

Тренувальні та тестові дані	x	y(x)
1. Тренувальні дані:	0.0	0
	0.1	0.36416
	0.2	0.3848
	0.3	0.36943
	0.4	0.33681
	0.5	0.2937
	0.6	0.24343
	0.7	0.1879
	0.8	0.12832
	0.9	0.06549
	1.0	0
2. Тестові дані:	0.05	0.3184
	0.15	0.38133

	0.47	0.30750
	0.87	0.08464

Таблиця 4.2 - Результати експериментів

Конфігурація нейромережі	Мінімальна помилка після тренування	Помилка тестування
1. кількість шарів = 2:		
- ітерації = 4 k	0.10216	0.09068
- ітерації = 20 k	0.10216	0.10790
- ітерації = 100 k	0.10216	0.10405
2. кількість шарів = 3:		
- ітерації = 4 k	0.10216	0.10518
- ітерації = 20 k	0.10216	0.10580
- ітерації = 100 k	0.10216	0.10676
3. кількість шарів = 4:		
- ітерації = 4 k	0.10216	0.09830
- ітерації = 20 k	0.10216	0.10818
- ітерації = 100 k	0.10216	0.10862
4. кількість шарів = 6:		
- ітерації = 4 k	0.10216	0.10688
- ітерації = 20 k	0.10216	0.09418
- ітерації = 100 k	0.10216	0.09886

Отже, проведене експериментальне дослідження щодо впливу масштабу (кількості шарів нейромережі) у поєднанні з кількістю тренувальних ітерацій на якість навчання (власне, якість є обернено-пропорційним значенням відносно помилки). Виконано статистичний аналіз отриманих даних.

Таблиця 4.3 – Результат статистичного аналізу даних.

Показник	Мінімальна помилка після тренування	Помилка тестування
1. Мінімум (шарів= 2, ітер= 4k)	0.10216	0.09068
2. Максимум (шарів= 4, ітер=100k)	0.10216	0.10862
3. Середнє значення:		
- кількість шарів = 2	0.10216	0.10088
- кількість шарів = 3	0.10216	0.12619
- кількість шарів = 4	0.10216	0.10503
- кількість шарів = 6	0.10216	0.09997
4. Загальне середнє	0.10216	0,10802

Отже, згідно отриманих результатів експериментів, можна зробити висновок, що найкращою комбінацією параметрів (серед розглянутих) є кількість шарів = 2 та кількість ітерацій = 4000. Тож, можна сказати, що у протестованому випадку помилка при тестуванні мережі не сильно змінюється. Тим не менш, за середніми показниками, очевидно, що конфігурація з більшою кількістю шарів є більш стабільною у вимірі різної кількості тренувальних ітерацій. Що не дивно, оскільки більша кількість шарів потребує більш довгого і цілеспрямованого процесу змін вагових коефіцієнтів та їх проникнення у глибші шари системи. І якість навчання пропорційна кількості ітерацій, тому що кожна ітерація додає шанс знаходження таких вагових коефіцієнтів, що дадуть у результаті менше значення помилки. Тим

не менш, помітно, що при збільшенні кількості шарів до 3, помилка тестування більшується, а після – стає спадати. Отже, можна зробити висновок, що найбільш оптимальним є значення кількості шарів = 6 або 8, оскільки зменшення кількості шарів призведе до збільшення помилки тестування, а збільшення кількості шарів призведе до зменшення помилки ціною стрімкого підвищення витрат на тренування.

ВИСНОВКИ

У результаті виконання завдання за даним курсовим проектом, було реалізовано програмне забезпечення для експериментального дослідження впливу масштабованості імунної нейромережевої системи (ІНМС) на якість її навчання. Програмне забезпечення реалізоване за допомогою мови програмування C++ та бібліотек STL а також, у незначній кількості, інших стандартних бібліотек. Проведене тестування програми.

Додаток реалізує адаптивну нейронну мережу з гнучким налаштуванням та системою навчання на базі штучної імунної системи. Має необхідний, згідно із дослідним завданням, функціонал:

- можливість гнучко змінювати кількість шарів у мережі;
- налаштовувати витрати на тренування нейромережі;
- отримання результатів дослідження у зручному зрозумілому вигляді;
- програма є крос-платформеною завдяки мові C++ та її бібліотекам.

Програмна частина реалізована в обсязі, достатньому для проведення досліджень об'єкту роботи, а саме впливу розмірності мережі на якість навчання. Отримані результати дослідження та зроблені відповідні наукові висновки щодо закономірності зв'язку між досліджуваними параметрами нейронної мережі. Результати досліджень мають практичну цінність, оскільки впливають на економічну ефективність та швидкість проведення майбутніх досліджень пов'язаних із нейромережами.

У якості продовження розвитку розробленого програмного забезпечення пропонується реалізація більш ефективного адаптивного методу навчання за алгоритмом зворотнього поширення помилки (backpropagation) [8]. Реалізація цього алгоритму навчання дасть змогу ще більше підвищити ефективність отримання результатів досліджень. Також перспективним напрямом є реалізація динамічної зміни параметрів у процесі навчання нейромережі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Руденко О., Бодянський Є. Штучні нейронні мережі : навч. посіб. Харків : Компанія СМІТ, 2006. 404 с.

2. Kurniawan D. Several types of artificial neural networks architecture that you should know. medium.com. URL: https://medium.com/mlearning-ai/several-types-of-artificial-neural-networks-architecture-that-you-should-know-c169a5e22ec7 (date of access: 08.11.2023).

3. Руденко О., Безсонов О., Романюк О. Нейромережеве прогнозування часових рядів на основі багатошарового персептрона. Development Management. 2019. Т. 5, № 1. С. 12. URL: https://doi.org/10.21511/dm.5(1).2019.03 (дата звернення: 09.11.2023).

4. Zewe A. Learning to grow machine-learning models / MIT News Office, 2023. URL: https://news.mit.edu/2023/new-technique-machine-learning-models-0322 (date of access: 16.11.2023).

5. Carlson D., Minerd I. Software design using C++ / Saint Vincent College, 2021. URL: cis.stvincent.edu/html/tutorials/swd/ (date of access: 19.11.2023)

6. Кораблев Н., Сорокина И., Русецкий А. Еволюційні гібридні системи обчислювального інтелекту зі змінною структурою для інтелектуального аналізу даних. Харків : ХНУРЕ, 2017. 196 с. URL: https://openarchive.nure.ua/server/api/core/bitstreams/b1048fcf-b395-42d9-9171-b29f96510244/content (дата звернення: 02.12.2023).

7. Корабльов Н. М., Сорокина И. В. Адаптивные нечеткие модели идентификации нелинейных объектов на основе искусственных иммунных систем. Бионика Интеллекта. 2008. Т. 2, № 69. С. 125–131.

8. Ryan M. How neural networks "learn". towardsdatascience.com. URL: https://towardsdatascience.com/how-neural-network-learn-3b56c175b5ca (date of access: 24.12.2023).