

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ
ДЛЯ СИНТЕЗУ НЕЙРОМОДЕЛЕЙ
SOFTWARE FOR NEURAL
MODEL SYNTHESIS

Виконав(ла): студент(ка) 4 курсу, групи КНТ-122

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ПОШУКАЙ А.А.

(ПРИЗВИЩЕ та ініціали)

Керівник СУББОТІН С.О.

(ПРИЗВИЩЕ та ініціали)

Рецензент КОРОТКИЙ О.В.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2026 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ПОШУКАЯ Артема Андрійовича

(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення для синтезу нейромоделей.
Software for Neural Model Synthesis

керівник проєкту (роботи) д.т.н., професор, СУББОТІН Сергій Олександрович,
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 07 ” квітня 2026 року № 139

2. Строк подання студентом проєкту (роботи) 10 червня 2026 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне
завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів) _____

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	СУББОТІН С.О., завідувач кафедри		
Нормоконтроль	КАЛІНІНА М.В., асистент		

7. Дата видачі завдання “20” квітня 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка структури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділи 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

_____ Артем ПОШУКАЙ
(підпис) (Імя ПРІЗВИЩЕ)

Керівник проєкту (роботи)

_____ Сергій СУББОТІН
(підпис) (Імя ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
89 с., 3 табл., 26 рис., 3 дод., 18 джерел.

C#, VISUAL STUDIO, ЗАСТОСУНОК, МОВА ПРОГРАМУВАННЯ,
МОДЕЛЬ, НЕЙРОННА МЕРЕЖА, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ.

Об'єкт дослідження – процес розробки програмного забезпечення для синтезу нейромоделей.

Предмет дослідження – програмні засоби для синтезу нейромоделей.

Мета роботи – розробка програмного забезпечення для синтезу нейромоделей, що забезпечує збільшення точності та надійності розпізнавальних моделей.

Матеріали, методи та технічні засоби: мова програмування C#, середовище розробки Visual Studio.

Результати. Розроблено проєктні рішення для створення програмного забезпечення для синтезу нейромоделей. Створено програмне забезпечення для синтезу нейромоделей. Проведено тестування розробленої програмної системи для синтезу нейромоделей.

Висновки. Мету роботи досягнуто. Розроблено програмне забезпечення для синтезу нейромоделей за допомогою мови програмування C# та середовища розробки Visual Studio.

Галузь використання – розпізнавання образів, діагностування.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 89 pages, 3 tables, 26 figures, 3 appendixes, 18 sources.

C#, VISUAL STUDIO, APPLICATION, PROGRAMMING LANGUAGE, MODEL, NEURAL NETWORK, SOFTWARE.

The object of research is the process of developing software for synthesizing neuromodels.

The subject of the research is software for synthesizing neuromodels.

The purpose of this work is to develop software for synthesizing neuromodels, which ensures an increase in the accuracy and reliability of recognition models.

Materials, methods and technical tools: C# programming language, Visual Studio development environment.

Results. Project solutions for the creation of software for synthesizing neuromodels has been designed. The software to support the process for synthesizing neuromodels has been developed. The developed software system for synthesizing neuromodels has been tested.

Conclusions. The goal of the work has been achieved. The software tools for synthesizing neuromodels using the C# programming language and the Visual Studio development environment have been developed.

Scope of use – pattern recognition, diagnostics.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ.....	9
1 Аналіз предметної області.....	11
1.1 Програмне забезпечення для синтезу нейромоделей	11
1.2 Аналіз програмного забезпечення для синтезу нейромоделей.....	18
1.3 Висновки за розділом 1	30
2 Матеріали і методи	32
2.1 Вибір мови програмування	32
2.2 Вибір середовища розробки для створення програмного забезпечення для синтезу нейромоделей	36
2.3 Висновки за розділом 2	40
3 Опис програми	41
3.1 Структура програмного забезпечення для синтезу нейромоделей	41
3.2 Функціонування програмного забезпечення для синтезу нейромоделей	44
3.3 Архітектура програмного забезпечення для синтезу нейромоделей	46
3.4 Проєктування інтерфейсу програмного забезпечення для синтезу нейромоделей	49
3.5 Висновки за розділом 3	51
4 Експлуатація, тестування та експериментальне дослідження програми.....	53
4.1 Призначення й умови застосування програми	53
4.2 Характеристики програми для синтезу нейромоделей.....	54
4.3 Інструкція по експлуатації програми.....	56
4.3.1 Звернення до програми	56
4.3.2 Вхідні й вихідні дані.....	56
4.3.3 Повідомлення	56
4.4 Виконання програмного забезпечення для синтезу нейромоделей	57

4.5 Тестування програмного забезпечення для синтезу нейромоделей.....	60
4.6 Висновки за розділом 4	60
Висновки	61
Перелік джерел посилання	64
Додаток А Технічне завдання	66
Додаток Б Фрагмент тексту програми	70
Додаток В Слайди презентації	82

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

GUI – Graphical User Interface;

НМ – нейронна мережа;

ПЗ – програмне забезпечення;

ШІ – штучний інтелект.

ВСТУП

Актуальність програмного забезпечення для синтезу нейромоделей визначається стрімким зростанням обсягів даних, ускладненням завдань обробки інформації та потребою в автоматизації інтелектуальної діяльності. У сучасних умовах дедалі більшого значення набуває здатність ефективно створювати, адаптувати та оптимізувати нейронні моделі без значних витрат часу та ресурсів, що обумовлює підвищений інтерес до спеціалізованих програмних рішень у цій сфері [1]-[2].

Спостерігається тенденція до ускладнення архітектур нейронних мереж, що потребує використання інструментів, здатних автоматизувати процес їх побудови. Застосування програмного забезпечення для синтезу нейромоделей дозволяє формалізувати підходи до проєктування, забезпечити відтворюваність результатів та мінімізувати вплив людського фактора. У таких умовах важливим стає впровадження інтелектуальних механізмів, які здатні здійснювати підбір структури моделі, налаштування гіперпараметрів і вибір оптимальних стратегій навчання [3]-[5].

Отже, актуальність дослідження та використання програмного забезпечення для синтезу нейромоделей зумовлюється необхідністю підвищення ефективності розроблення, автоматизації складних процесів проєктування, оптимізації використання ресурсів і забезпечення гнучкості при розв'язанні широкого спектра прикладних задач [4]-[7].

Проте деякі програми для синтезу нейромоделей потребують глибокого розуміння принципів функціонування нейронних мереж, методів оптимізації та специфіки обробки даних. У результаті виникає ситуація, коли заявлена універсальність такого програмного забезпечення обмежується практичними навичками розробника. Крім того, програмні засоби синтезу нейронних мереж часто генерують складні моделі, структура яких є непрозорою для користувача. Це ускладнює аналіз причин прийняття рішень нейромоделлю, знижує рівень довіри до результатів і створює труднощі при їх впровадженні в

критично важливі системи. Додатковим недоліком є ризик отримання надмірно складних нейромоделей, оскільки автоматизовані алгоритми можуть формувати архітектури, які добре працюють на навчальних даних, але демонструють низьку здатність до узагальнення. Це потребує додаткового контролю та валідації результатів, що частково нівелює переваги автоматизації процесу синтезу нейромоделей. Не можна ігнорувати й проблему обмеженої гнучкості окремих програмних рішень. Деякі системи орієнтовані на конкретні типи задач або архітектур, що ускладнює їх адаптацію до нестандартних умов. У таких випадках виникає необхідність модифікації або доповнення інструментів, що підвищує складність розроблення [1]-[6].

Тому актуальною є розробка програмного забезпечення для синтезу нейромоделей. У дипломній кваліфікаційній роботі бакалавра розв'язується актуальне завдання розробки програмного забезпечення для синтезу нейромоделей, що забезпечує збільшення точності та надійності розпізнавальних моделей.

Для досягнення поставленої мети у кваліфікаційній роботі бакалавра необхідно розв'язати такі задачі:

- виконати аналіз предметної області та програмних засобів для синтезу нейромоделей;
- здійснити проєктування програмного забезпечення для синтезу нейромоделей;
- створити програмне забезпечення для синтезу нейромоделей;
- виконати тестування розробленого програмного забезпечення для синтезу нейромоделей.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Програмне забезпечення для синтезу нейромоделей

Програмне забезпечення для синтезу нейромоделей є спеціалізованим класом програмних систем, призначених для створення, налаштування, навчання та використання штучних нейронних мереж з метою розв'язання задач аналізу даних, прогнозування та виявлення прихованих закономірностей. У таких програмних системах поєднуються засоби формування структури моделі, інструменти керування параметрами та алгоритми навчання, що забезпечує можливість переходу від вихідних даних до побудови інтелектуальної моделі без необхідності реалізації всіх математичних механізмів вручну [1]-[2].

Функціонування подібного програмного забезпечення базується на узагальненні принципів роботи нейронних мереж у вигляді програмних компонентів, які дозволяють описувати архітектуру мережі, визначати зв'язки між її елементами та організовувати процес адаптації вагових коефіцієнтів у відповідь на вхідні дані. У межах таких систем реалізується підтримка різних підходів до навчання, що дає змогу адаптувати модель до специфіки поставленої задачі. Завдяки цьому досягається можливість ефективного використання нейромереж у різних прикладних галузях без необхідності глибокого занурення у математичні деталі [1]-[2].

Характерною рисою програмних засобів цього типу є їх орієнтація на спрощення процесу розробки інтелектуальних систем. Це досягається шляхом використання готових компонентів, автоматизованих процедур налаштування та візуальних інтерфейсів, які забезпечують більш інтуїтивне керування процесом створення моделей. У результаті підвищується доступність технологій нейронних мереж навіть для користувачів, які не мають значного досвіду програмування або глибоких знань у сфері машинного навчання та штучного інтелекту [1]-[2].

Важливою особливістю є можливість багаторазового використання створених моделей та їх адаптації до нових умов, що дозволяє зменшити витрати часу на розробку та експериментування. Також забезпечується підтримка аналізу результатів роботи моделей, що сприяє оцінюванню їх ефективності та подальшому вдосконаленню. У сукупності це формує основу для створення гнучких та масштабованих рішень, здатних обробляти складні дані та формувати обґрунтовані висновки [1]-[2].

Проаналізуємо штучні нейронні мережі (НМ). НМ розглядаються як обчислювальні моделі, створені з метою відтворення принципів обробки інформації, характерних для людського мозку. У їх основі лежить ідея використання умовних елементів, що виконують роль нейронів, які взаємодіють між собою та забезпечують аналіз вхідних даних. Подібно до біологічної нервової системи, такі моделі здатні виявляти приховані залежності, узагальнювати отриману інформацію та формувати прогнози на її основі. У процесі функціонування штучних нейронних мереж відбувається накопичення досвіду, що дозволяє підвищувати точність результатів при подальшому використанні [1]-[2].

Структурна організація нейронних мереж передбачає наявність кількох рівнів обробки інформації, які взаємодіють між собою. На початковому етапі здійснюється прийом даних, що можуть бути представлені у різних формах, включаючи зображення, текст або числові значення. Надалі ці дані послідовно проходять через проміжні рівні, де відбувається їх перетворення та узагальнення. Кожен наступний етап обробки формує більш складне представлення інформації, що дозволяє виявляти закономірності, які не є очевидними на початковому рівні. Завершальний етап забезпечує формування підсумкового результату, який може бути представлений у вигляді класифікації, прогнозу або іншого рішення [3]-[4].

Механізм роботи таких систем ґрунтується на процесі навчання, під час якого модель поступово вдосконалює власні параметри. Це досягається шляхом зіставлення отриманих результатів із відомими правильними

значеннями, що дозволяє коригувати внутрішні зв'язки між елементами мережі. У ході багаторазового повторення цього процесу відбувається адаптація системи до характеру вхідних даних, що сприяє підвищенню точності її функціонування [3]-[4].

Після завершення етапу навчання здійснюється перевірка здатності моделі працювати з новою інформацією, яка не використовувалася раніше. Такий підхід дозволяє оцінити рівень узагальнення знань і визначити, наскільки ефективно система здатна застосовувати отриманий досвід у нових умовах. У разі достатнього рівня підготовки мережа демонструє здатність коректно інтерпретувати невідомі дані та формувати обґрунтовані результати, що підтверджує її придатність для практичного використання у різноманітних задачах аналізу та прогнозування [3]-[4].

Процес навчання штучних нейронних мереж ґрунтується на послідовному опрацюванні великого обсягу даних, у ході якого система поступово формує уявлення про характерні ознаки об'єктів. Під час цього процесу моделі подається набір прикладів, що містять необхідну інформацію для виявлення закономірностей. У результаті багаторазового проходження даних через внутрішню структуру мережі відбувається налаштування параметрів, завдяки чому формується здатність розрізняти об'єкти або явища на основі їхніх властивостей [3]-[4].

Після етапу навчання здійснюється перевірка ефективності моделі шляхом подачі нових даних, які раніше не використовувалися. Отримані результати зіставляються з відомими правильними значеннями, що дозволяє оцінити точність функціонування. У випадку невідповідності між очікуваним і фактичним результатом відбувається корекція внутрішніх параметрів мережі. Така корекція реалізується через механізм, що передбачає передачу інформації про помилку у зворотному напрямку, внаслідок чого змінюються вагові коефіцієнти зв'язків між елементами системи. Повторення цього циклу забезпечує поступове зменшення похибки та підвищення якості роботи моделі [3]-[4].

Завдяки багаторазовому виконанню операцій навчання та корекції система набуває здатності ефективно узагальнювати інформацію та застосовувати отримані знання до нових ситуацій. Таким чином, формується адаптивна модель, яка може не лише відтворювати відомі залежності, але й виявляти нові закономірності у вхідних даних [3]-[4].

Важливу роль у функціонуванні нейронних мереж відіграють механізми перетворення сигналів усередині нейронів, що забезпечують нелінійний характер обробки інформації. Саме завдяки таким перетворенням стає можливим відображення складних залежностей, які не можуть бути описані простими лінійними моделями. У різних випадках використовуються різні типи функціональних перетворень, кожен із яких має власні особливості впливу на результати обчислень. Одні з них обмежують вихідні значення певним інтервалом, що є корисним для задач, пов'язаних із визначенням належності до певної категорії. Інші забезпечують просту та ефективну обробку сигналів у внутрішніх рівнях мережі, сприяючи стабільності навчання та зменшенню проблем, пов'язаних із передачею інформації між шарами. Існують також варіанти, які дозволяють враховувати як додатні, так і від'ємні значення, розширюючи можливості моделі щодо представлення складних залежностей. Окремі підходи спрямовані на перетворення результатів у форму, зручну для інтерпретації, зокрема у вигляді розподілу ймовірностей для задач багатокласової класифікації [3]-[4].

Ефективність навчання нейронних мереж досягається завдяки поєднанню механізмів адаптації параметрів і використанню спеціальних функцій перетворення, які разом забезпечують здатність моделі до складного аналізу даних і формування точних прогнозів [3]-[4].

Штучні нейронні мережі можуть реалізовуватися у різних формах, що відрізняються за принципами обробки інформації та характером внутрішніх зв'язків. Одним із базових варіантів є модель, у якій передача сигналу відбувається послідовно від вхідних даних до кінцевого результату без повернення на попередні рівні. Така організація забезпечує простоту

структури та передбачуваність обчислень, що робить її придатною для задач, де не потрібне врахування залежностей між попередніми станами [4]-[5].

Інший підхід орієнтований на обробку даних, що мають просторову впорядкованість. У цьому випадку використовуються спеціальні механізми, здатні виділяти локальні особливості та структури, що повторюються у даних. Завдяки цьому забезпечується ефективне розпізнавання складних образів, оскільки система здатна поступово формувати уявлення про характерні ознаки об'єктів на різних рівнях абстракції. Такий тип організації широко застосовується там, де необхідно працювати з візуальною або подібною за структурою інформацією [4]-[5].

Окрему групу становлять моделі, що ґрунтуються на використанні функцій, чутливих до відстані між значеннями. У таких системах акцент робиться на локальному впливі вхідних даних, що дозволяє ефективно відтворювати залежності, які мають характер згладжених або концентричних змін. Це робить подібні підходи корисними у випадках, коли необхідно апроксимувати складні функції або працювати з даними, що мають чітко виражені центри групування [4]-[5].

Існують також моделі, здатні враховувати послідовність подій або залежність між елементами, розташованими у часі. У таких системах передбачено механізми збереження інформації про попередні стани, що дозволяє використовувати накопичений контекст під час обробки нових даних. Завдяки цьому досягається можливість аналізу часових рядів, текстової інформації та інших структур, де порядок розташування елементів має суттєве значення [4]-[5].

Під час навчання будь-якого з описаних типів мереж ключову роль відіграють процедури коригування внутрішніх параметрів. Ці процедури спрямовані на зменшення розбіжності між отриманими результатами та очікуваними значеннями. У процесі оптимізації використовуються різні підходи, які відрізняються способом обчислення змін та швидкістю їх застосування. Деякі методи базуються на поступовому переміщенні

параметрів у напрямку зменшення похибки, інші враховують попередні зміни для стабілізації процесу навчання або адаптують швидкість оновлення залежно від особливостей даних [4]-[5].

Застосування вдосконалених варіантів алгоритмів дозволяє досягати більш стійкої та швидкої збіжності, що є важливим для складних моделей. У деяких випадках оновлення параметрів здійснюється на основі окремих прикладів, що прискорює процес, але водночас може призводити до коливань у результатах. Інші підходи використовують накопичену інформацію про попередні кроки, що сприяє більш плавному та контрольованому налаштуванню моделі [4]-[5].

Різноманітність типів нейронних мереж і методів їх оптимізації забезпечує гнучкість у вирішенні широкого спектра задач, дозволяючи обирати найбільш відповідну модель залежно від структури даних та вимог до точності й швидкодії [4]-[5].

Сфери використання штучних нейронних мереж охоплюють широкий спектр практичних завдань, у межах яких здійснюється аналіз великих масивів інформації та формування інтелектуальних рішень. У цифрових платформах, пов'язаних із взаємодією користувачів, такі моделі застосовуються для виявлення інтересів і поведінкових характеристик, що дозволяє формувати персоналізовані рекомендації та підбирати інформаційний вміст відповідно до індивідуальних вподобань. Завдяки цьому забезпечується більш релевантна взаємодія між користувачем і сервісом, а також підвищується ефективність інформаційного впливу [4]-[5].

У сфері комерційної діяльності подібні технології використовуються для аналізу поведінки споживачів, що дає змогу передбачати їхні потреби та формувати індивідуалізовані пропозиції. На основі накопичених даних визначаються закономірності купівельної активності, що сприяє оптимізації стратегій просування товарів і послуг. У результаті покращується взаємодія з клієнтами та підвищується ефективність маркетингових рішень [4]-[5].

У медичній практиці нейромережеві моделі застосовуються для обробки складних діагностичних даних, що дозволяє виявляти патологічні зміни з високим рівнем точності. Вони сприяють аналізу медичних зображень, прогнозуванню розвитку захворювань та формуванню індивідуальних підходів до лікування. Використання таких систем відкриває можливості для підвищення якості діагностики та підтримки прийняття рішень у складних клінічних випадках [4]-[5].

У засобах взаємодії людини з комп'ютером нейронні мережі забезпечують розпізнавання мовлення та обробку природної мови, що дозволяє реалізувати інтелектуальні системи керування. Це сприяє автоматизації повсякденних дій, пов'язаних із плануванням, пошуком інформації та виконанням команд, що значно підвищує зручність використання цифрових пристроїв [4]-[5].

У галузі обслуговування клієнтів такі моделі застосовуються для автоматизації обробки звернень, що дозволяє швидко реагувати на запити та надавати користувачам необхідну інформацію. Завдяки здатності аналізувати текстові та інші дані забезпечується підвищення якості комунікації та скорочення часу обслуговування [4]-[5].

У фінансовій сфері нейромережеві підходи використовуються для аналізу транзакційної активності, що дозволяє виявляти нетипові операції та оцінювати рівень ризику. Це сприяє підвищенню безпеки фінансових операцій, а також забезпечує більш точне прогнозування економічних показників [4]-[5].

Попри значні переваги, застосування нейронних мереж супроводжується рядом обмежень. Однією з ключових проблем є залежність від якості та обсягу вхідних даних, оскільки недостатня або некоректна інформація може негативно вплинути на результати роботи. Також суттєвим фактором є потреба у значних обчислювальних ресурсах, що ускладнює використання складних моделей у середовищах із обмеженими технічними можливостями [4]-[5].

Крім того, під час навчання може виникати ситуація, коли модель надмірно пристосовується до навчальних даних, що знижує її здатність коректно працювати з новими прикладами. Важливим аспектом є також складність інтерпретації отриманих результатів, оскільки внутрішні механізми прийняття рішень залишаються недостатньо прозорими. Окрему проблему становить тривалість процесу навчання, яка може значно впливати на швидкість впровадження розроблених рішень [4]-[5].

У цілому, незважаючи на наявні труднощі, розвиток технологій у цій сфері сприяє розширенню можливостей застосування нейронних мереж, що поступово змінює підходи до обробки інформації та вирішення складних прикладних задач, роблячи інтелектуальні системи невід'ємною частиною сучасного цифрового середовища [4]-[5]. Отже, програмне забезпечення для синтезу нейромоделей являє собою інструментальне середовище, яке інтегрує методи штучного інтелекту з сучасними підходами програмної інженерії, забезпечуючи ефективну реалізацію, дослідження та впровадження неймережевих рішень у різних сферах діяльності [1]-[2].

Визначено, що програмне забезпечення для синтезу нейромоделей є спеціалізованим класом програмних систем, призначених для створення, налаштування, навчання та використання штучних нейронних мереж з метою розв'язання задач аналізу даних, прогнозування та виявлення прихованих закономірностей. У таких програмних системах поєднуються засоби формування структури моделі, інструменти керування параметрами та алгоритми навчання, що забезпечує можливість переходу від вихідних даних до побудови інтелектуальної моделі без необхідності реалізації всіх математичних механізмів вручну.

1.2 Аналіз програмного забезпечення для синтезу нейромоделей

Проаналізуємо програмне забезпечення для синтезу неймережевих моделей [6]-[14].

Програмне забезпечення Neuroph (рис. 1.1) являє собою середовище для побудови та дослідження нейромереж, реалізоване на основі мови програмування Java. У його основі лежить бібліотека з відкритим кодом, структура якої побудована навколо базових понять нейронних мереж, що забезпечує зрозуміле відображення їхньої внутрішньої логіки. Архітектура програмного забезпечення Neuroph орієнтована на спрощення процесу створення моделей, завдяки чому забезпечується можливість швидкого переходу від теоретичного розуміння до практичної реалізації нейромережових рішень [7], [8], [12].

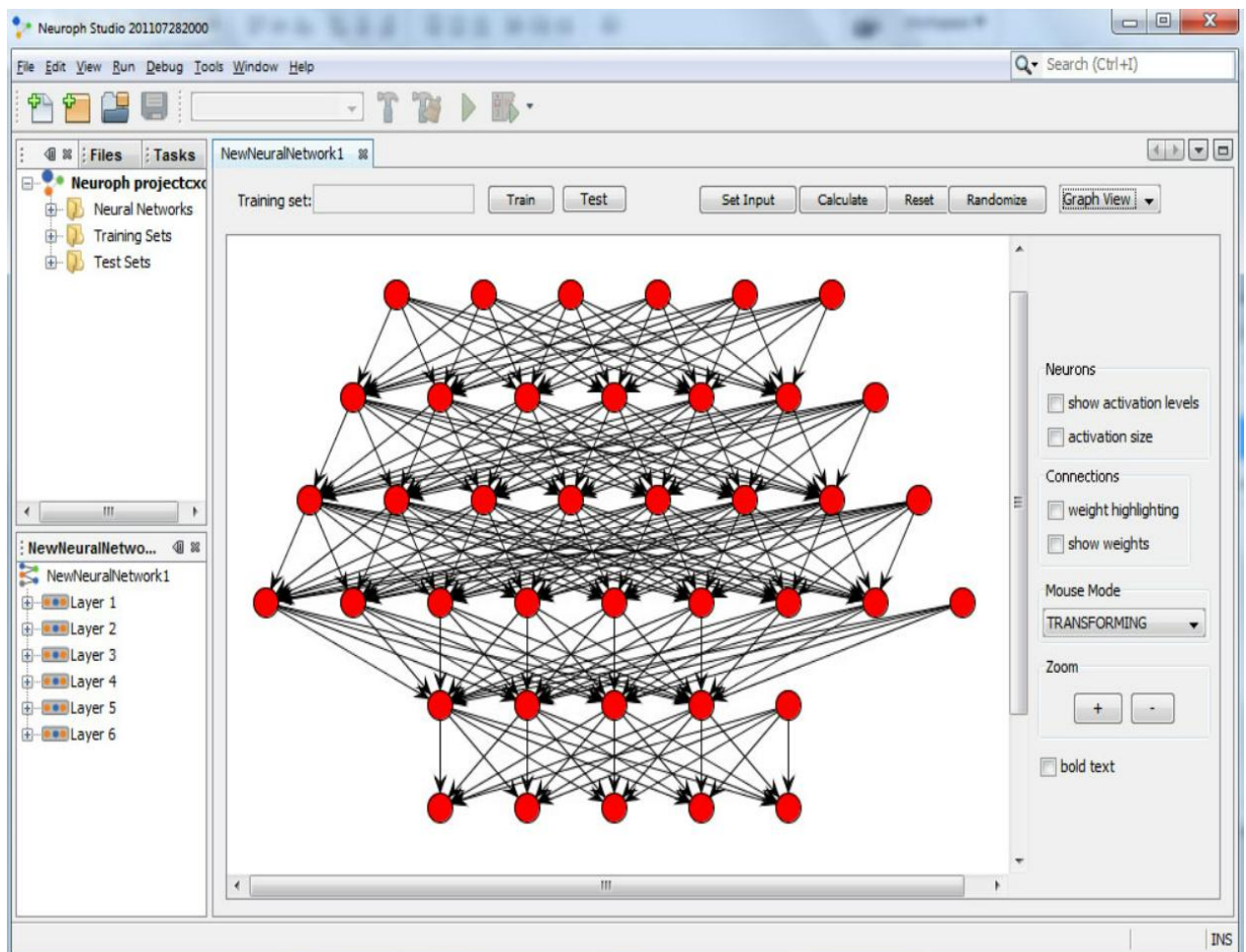


Рисунок 1.1 – Програмне забезпечення Neuroph [12]

Функціональні можливості Neuroph включають підтримку процесів проєктування, навчання та подальшого використання нейронних мереж. У межах середовища реалізовано засоби, що дозволяють формувати структуру

мережі, налаштовувати її параметри та виконувати навчання на основі підготовлених даних. Особливістю є наявність графічного інтерфейсу, який значно спрощує взаємодію з системою та дозволяє створювати моделі без необхідності глибокого занурення у програмний код. Такий підхід сприяє підвищенню доступності інструменту та робить його придатним для широкого кола користувачів [7], [8], [12].

Загальна концепція побудови Neuroph передбачає мінімалізм у реалізації основних компонентів, що позитивно впливає на зрозумілість системи. Внутрішні модулі організовано таким чином, щоб забезпечити чітке відображення структури нейромережі та її елементів, включаючи нейрони, зв'язки та механізми навчання. Завдяки цьому досягається баланс між функціональністю та простотою використання, що особливо важливо під час освоєння принципів роботи нейронних мереж [7], [8], [12].

Застосування відкритої ліцензії сприяє вільному використанню та модифікації програмного забезпечення, що відкриває можливості для його адаптації під конкретні задачі. Це дозволяє використовувати Neuroph як навчальний інструмент, а також як основу для розробки власних рішень у сфері інтелектуального аналізу даних. Простота інтеграції та зрозуміла структура роблять його зручним середовищем для експериментів із різними архітектурами нейромереж [7], [8], [12].

Характерною рисою програмного забезпечення Neuroph є орієнтація на доступність та легкість освоєння, що досягається за рахунок інтуїтивного інтерфейсу та логічної організації компонентів. Разом із тим, окремі аспекти, пов'язані з візуальним оформленням або зручністю використання зовнішніх ресурсів, можуть потребувати подальшого вдосконалення. Незважаючи на це, Neuroph залишається ефективним інструментом для ознайомлення з принципами побудови нейромереж і реалізації базових моделей у практичних задачах [7], [8], [12].

Основними характеристиками програмного забезпечення Neuroph є такі [7], [8], [12]:

– легковаговість системи: програмне забезпечення Neuroph характеризується компактною структурою та невеликим обсягом ресурсів, необхідних для його функціонування, що забезпечує швидкий запуск і ефективну роботу навіть на менш продуктивних системах [7], [8], [12];

– відкритий вихідний код: наявність відкритої кодової бази дозволяє модифікувати програмне забезпечення відповідно до потреб користувача та інтегрувати його у власні розробки [7], [8], [12];

– орієнтація на Java-платформу: реалізація на мові програмування Java забезпечує кросплатформеність та можливість використання в різних операційних системах [7], [8], [12];

– наявність графічного інтерфейсу: інтегрований візуальний редактор програмного забезпечення Neuroph дозволяє створювати нейронні мережі без глибокого програмування та спрощує процес їх налаштування [7], [8], [12];

– підтримка базових архітектур: у програмному забезпеченні Neuroph реалізовано можливість створення стандартних типів нейромереж, що дозволяє вирішувати типові задачі класифікації та прогнозування [7], [8], [12];

– простота освоєння: структура програмного забезпечення Neuroph є зрозумілою та логічною, що полегшує навчання та використання для користувачів з різним рівнем підготовки [7], [8], [12];

– модульна організація: програмне забезпечення Neuroph побудовано за принципом розділення компонентів, що забезпечує можливість розширення та вдосконалення окремих частин без зміни всієї програми [7], [8], [12];

– інтеграція процесів навчання: у програмному забезпеченні Neuroph передбачено засоби для реалізації навчання нейромереж без використання сторонніх інструментів [7], [8], [12];

– підтримка збереження моделей: у програмному забезпеченні Neuroph забезпечується можливість збереження створених нейромереж для подальшого використання або аналізу [7], [8], [12];

– орієнтація на навчальні цілі: програмне забезпечення Neurorh широко використовується для ознайомлення з принципами роботи нейронних мереж та проведення експериментів [7], [8], [12].

Серед недоліків програмного забезпечення Neurorh можна відзначити обмеженість функціональних можливостей порівняно з більш сучасними середовищами, а також певні особливості реалізації інтерфейсу, які можуть впливати на зручність роботи користувача [7], [8], [12].

Програмне забезпечення Caffe (рис. 1.2) представляє собою спеціалізоване середовище для побудови та дослідження глибоких нейронних мереж, орієнтоване на забезпечення високої продуктивності та гнучкості у процесі розробки моделей. Його архітектура побудована таким чином, щоб надати можливість формувати різноманітні конфігурації нейромереж шляхом комбінування окремих компонентів, що дозволяє адаптувати систему до широкого спектра задач у сфері інтелектуального аналізу даних. Особливу увагу приділено ефективності виконання обчислень, що робить це програмне забезпечення придатним для роботи з ресурсомісткими алгоритмами глибокого навчання [9], [13].

Caffe Performance on Multi-GPUs Per Node

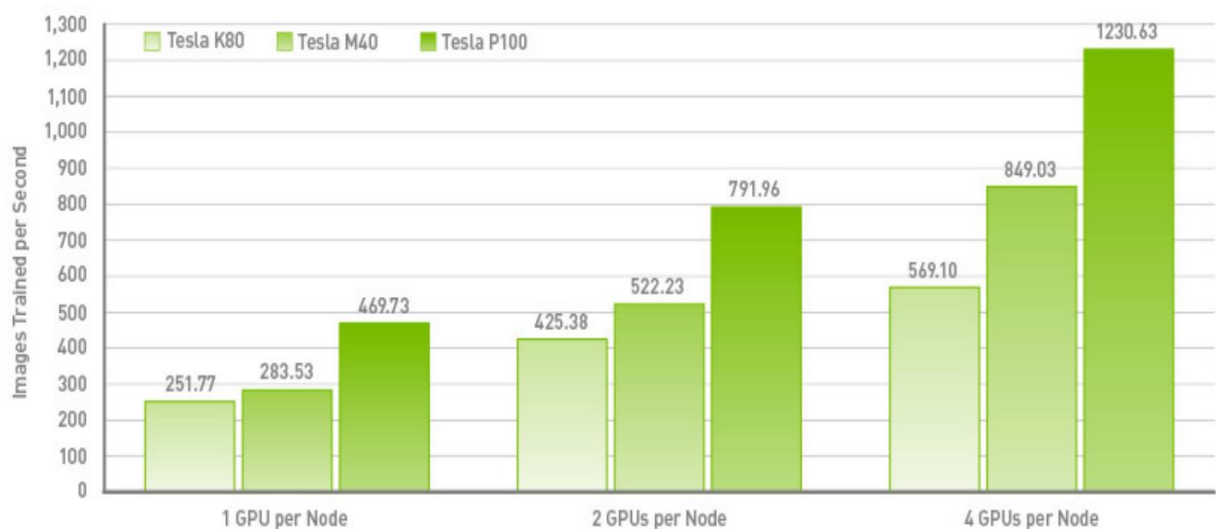


Рисунок 1.2 – Програмне забезпечення Caffe [13]

Однією з ключових характеристик програмного забезпечення Caffe є його модульна організація, яка дозволяє користувачу описувати структуру нейромережі через спеціалізовані конфігураційні механізми, не заглиблюючись безпосередньо у програмний код. Такий підхід забезпечує високу гнучкість при проєктуванні моделей і дає змогу експериментувати з різними архітектурними рішеннями без значних витрат часу на програмування. Завдяки цьому процес створення, налаштування та модифікації нейромереж набуває більш формалізованого та структурованого характеру [9], [13].

Важливою особливістю програмного забезпечення Caffe є орієнтація на використання апаратних ресурсів, зокрема графічних процесорів, що дозволяє значно прискорити виконання обчислень. Система забезпечує сумісність із різними обчислювальними платформами та надає можливість інтеграції з іншими інструментами, зокрема середовищами програмування, що розширює спектр її застосування. Така інтеграційна здатність сприяє використанню Caffe у складних програмних рішеннях, де необхідна взаємодія між різними компонентами програмної екосистеми [9], [13].

Архітектура програмного забезпечення Caffe сприяє чіткому розмежуванню етапів роботи з нейромережею, включаючи визначення структури, оптимізацію параметрів і впровадження отриманих моделей у практичні задачі. Завдяки цьому забезпечується системність підходу до розробки, що дозволяє зосередитися на експериментуванні та вдосконаленні моделей без зайвих технічних ускладнень. У той же час, для досягнення максимальної ефективності роботи програмного забезпечення Caffe може знадобитися додаткове налаштування окремих параметрів, що вимагає певного рівня підготовки користувача [9], [13].

Разом із цим слід зазначити, що освоєння програмного забезпечення Caffe може бути пов'язане з певними труднощами для користувачів, які не мають достатнього досвіду роботи з подібними системами. Незважаючи на це, завдяки своїй продуктивності, гнучкості та можливості інтеграції з іншими

інструментами, дане програмне забезпечення залишається ефективним засобом для розробки та дослідження нейромережових моделей, особливо у випадках, коли пріоритетом є швидкість обчислень і можливість побудови складних архітектур [9], [13].

Програмне забезпечення Caffe має такі особливості [9], [13]:

- висока продуктивність обчислень: програмне забезпечення Caffe оптимізоване для швидкого виконання операцій глибокого навчання, що забезпечує ефективну обробку великих обсягів даних [9], [13];

- модульна архітектура: структура системи дозволяє комбінувати окремі компоненти для побудови різних конфігурацій нейромереж [9], [13];

- декларативний підхід до опису моделей: реалізовано можливість задавати структуру нейромережі через конфігураційні файли без необхідності безпосереднього програмування [9], [13];

- підтримка графічних процесорів: у програмному забезпеченні Caffe забезпечується використання апаратного прискорення для підвищення швидкодії навчання моделей [9], [13];

- інтеграція з іншими середовищами: у програмному забезпеченні Caffe передбачена взаємодія з популярними мовами програмування та інструментами для розширення функціональності [9], [13];

- орієнтація на глибоке навчання: програмне забезпечення Caffe спеціалізується на реалізації складних багатoshарових нейромереж [9], [13];

- масштабованість рішень: архітектура програмного забезпечення Caffe дозволяє працювати як з невеликими задачами, так і з великими наборами даних [9], [13];

- чітка структуризація моделей: процес побудови нейромережі у програмному забезпеченні Caffe організовано у вигляді окремих етапів, що полегшує керування моделлю [9], [13];

- можливість повторного використання компонентів: у програмному забезпеченні Caffe окремі частини нейромережі можуть застосовуватися у різних проєктах без суттєвих змін [9], [13];

– гнучкість налаштування параметрів: у програмному забезпеченні Caffe передбачено широкий набір засобів для оптимізації процесу навчання нейронних мереж [9], [13].

Серед недоліків програмного забезпечення Caffe можна відзначити складність освоєння для користувачів без достатнього досвіду роботи з глибоким навчанням, необхідність додаткового налаштування окремих компонентів для досягнення максимальної ефективності, а також дещо обмежену гнучкість у порівнянні з сучаснішими фреймворками, що може ускладнювати реалізацію нетипових або експериментальних архітектур нейромереж [9], [13].

Програмне забезпечення Neural Designer (рис. 1.3) є інструментом, орієнтованим на спрощення процесів створення, навчання та використання нейромережових моделей за рахунок інтуїтивно зрозумілого інтерфейсу. У його основі лежить підхід, який дозволяє виконувати більшість дій без необхідності написання програмного коду, що робить середовище доступним для користувачів з різним рівнем підготовки [10], [11], [14].

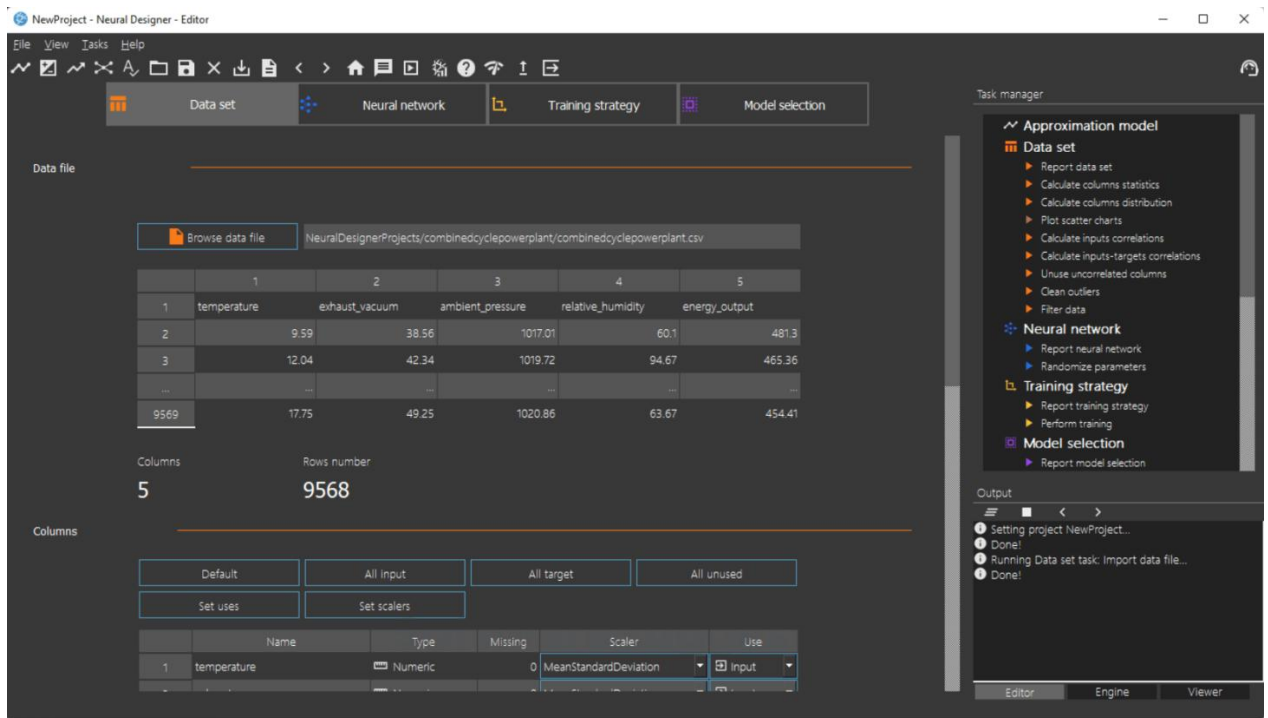


Рисунок 1.3 – Програмне забезпечення Neural Designer [14]

Функціональні можливості програмного продукту Neural Designer спрямовані на підтримку повного циклу роботи з нейромережами, включаючи їх проєктування, навчання та оцінювання. Значна увага приділяється автоматизації процесів, що традиційно вимагають глибоких знань у галузі машинного навчання. Зокрема, у програмі Neural Designer реалізовано механізми, які дозволяють підбирати параметри моделі без значного втручання користувача, що сприяє підвищенню ефективності роботи та скороченню часу на налаштування [10], [11], [14].

Особливістю програмного забезпечення Neural Designer є використання візуального підходу до побудови моделей, де основні елементи нейромережі представлені у вигляді готових компонентів. Це дає можливість формувати структуру мережі шляхом комбінування цих елементів, що значно полегшує експериментування та модифікацію моделей. Такий підхід особливо зручний у випадках, коли необхідно швидко перевірити різні варіанти архітектур або дослідити вплив параметрів на результати [10], [11], [14].

Neural Designer також орієнтований на вирішення прикладних задач, пов'язаних із прогнозуванням, аналізом даних та виявленням закономірностей. Його використання дозволяє створювати програмні рішення на основі штучного інтелекту без необхідності глибокого занурення у технічні деталі реалізації алгоритмів. Це робить систему привабливою для команд, які лише починають працювати з нейромережами або прагнуть швидко отримати практичні результати [10], [11], [14].

Разом із тим, певні обмеження можуть впливати на вибір цього інструменту у складніших сценаріях. Зокрема, орієнтація на спрощений інтерфейс і автоматизацію процесів може зменшувати гнучкість у налаштуванні моделей, що ускладнює реалізацію нестандартних підходів. Крім того, використання комерційної моделі розповсюдження може бути менш привабливим порівняно з відкритими рішеннями, які надають ширші можливості для модифікації та інтеграції [10], [11], [14].

Таким чином, Neural Designer є ефективним засобом для швидкого створення та дослідження нейромережових моделей, орієнтованим на зручність використання, автоматизацію процесів та доступність для користувачів, що не мають глибокого досвіду програмування, хоча в окремих випадках може поступатися більш гнучким інструментам, заснованим на програмному коді [10], [11], [14].

Програмне забезпечення Neural Designer має такі особливості [10], [11], [14]:

- інтуїтивний інтерфейс: взаємодія з програмним забезпеченням Neural Designer організована у зручній візуальній формі, що спрощує побудову та налаштування нейромереж без глибокого програмування [10], [11], [14];

- відсутність потреби у кодуванні: створення та навчання моделей у Neural Designer здійснюється без написання програмного коду, що робить систему доступною для широкого кола користувачів [10], [11], [14];

- автоматизація налаштувань: у Neural Designer реалізовано механізми автоматичного підбору параметрів моделі, що дозволяє підвищити ефективність навчання [10], [11], [14];

- підтримка повного циклу роботи: програмне забезпечення Neural Designer охоплює всі етапи створення, навчання та оцінювання нейромережових моделей [10], [11], [14];

- візуальне конструювання моделей: у Neural Designer структура нейромережі формується шляхом використання готових компонентів, що значно спрощує процес проєктування [10], [11], [14];

- орієнтація на аналіз даних: система Neural Designer забезпечує ефективне виявлення закономірностей, прогнозування та обробку складних залежностей у даних [10], [11], [14];

- прискорення робочих процесів: у Neural Designer використання готових рішень і автоматизованих функцій дозволяє зменшити час розробки моделей [10], [11], [14];

– орієнтація на вебплатформу: доступ до функціоналу програми Neural Designer реалізовано через вебсередовище, що визначає специфіку використання системи [10], [11], [14];

– використання готових компонентів: наявність шаблонних елементів дозволяє швидко створювати моделі без детального опрацювання їх внутрішньої структури [10], [11], [14].

Серед недоліків програмного забезпечення Neural Designer можна відзначити обмежену гнучкість у налаштуванні моделей у порівнянні з програмними рішеннями, що базуються на написанні коду, залежність від вбудованих інструментів і компонентів, що ускладнює реалізацію нестандартних підходів, а також менш вигідні умови використання у порівнянні з відкритими аналогами, що може стримувати його застосування у складних або масштабних проєктах [10], [11], [14].

Порівняльну характеристику програмного забезпечення для синтезу нейромоделей наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння програмного забезпечення для синтезу нейромоделей

Критерій порівняння	Neuroph [12]	Caffe [13]	Neural Designer [14]
Простота використання	+	+—	+
Необхідність програмування	+—	+	—
Гнучкість налаштування	+—	+	+—
Підтримка графічного інтерфейсу	+	—	+
Інтеграція з іншими інструментами	+—	+	+—
Відкритість коду	+	+	—

За результатами проведеного аналізу можна зробити висновок, що у наш час існує досить багато програмних засобів для синтезу нейромоделей. Проте деякі програми для синтезу нейромоделей потребують глибокого розуміння принципів функціонування нейронних мереж, методів оптимізації та специфіки обробки даних. У результаті виникає ситуація, коли заявлена універсальність такого програмного забезпечення обмежується практичними навичками розробника. Крім того, програмні засоби синтезу нейронних мереж часто генерують складні моделі, структура яких є непрозорою для користувача. Це ускладнює аналіз причин прийняття рішень нейромоделлю, знижує рівень довіри до результатів і створює труднощі при їх впровадженні в критично важливі системи. Додатковим недоліком є ризик отримання надмірно складних нейромоделей, оскільки автоматизовані алгоритми можуть формувати архітектури, які добре працюють на навчальних даних, але демонструють низьку здатність до узагальнення. Це потребує додаткового контролю та валідації результатів, що частково нівелює переваги автоматизації процесу синтезу нейромоделей. Не можна ігнорувати й проблему обмеженої гнучкості окремих програмних рішень. Деякі системи орієнтовані на конкретні типи задач або архітектур, що ускладнює їх адаптацію до нестандартних умов. У таких випадках виникає необхідність модифікації або доповнення інструментів, що підвищує складність розроблення. Тому актуальною є розробка програмного забезпечення для синтезу нейромоделей.

При розробці програмного забезпечення для синтезу нейромоделей необхідно забезпечити такі функціональні вимоги:

- підтримка можливості створення та налаштування багатошарових неймереж із вибором кількості шарів та нейронів у кожному шарі;
- підтримка можливості вибору функцій активації для нейронів мережі;
- можливість завантаження вхідних даних для навчання неймережі;

- можливість вибору характеристик процесу навчання нейромоделі із визначенням параметрів навчання, таких як швидкість навчання, контроль похибки, максимально допустима кількість ітерацій та час виконання;
- можливість оцінювання точності нейромоделі;
- підтримка інтерактивної роботи користувача з програмою, включаючи зміну параметрів моделі та вибір наборів даних для синтезу нейромоделі.

1.3 Висновки за розділом 1

Визначено, що програмне забезпечення для синтезу нейромоделей є спеціалізованим класом програмних систем, призначених для створення, налаштування, навчання та використання штучних нейронних мереж з метою розв'язання задач аналізу даних, прогнозування та виявлення прихованих закономірностей. У таких програмних системах поєднуються засоби формування структури моделі, інструменти керування параметрами та алгоритми навчання, що забезпечує можливість переходу від вихідних даних до побудови інтелектуальної моделі без необхідності реалізації всіх математичних механізмів вручну.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато програмних засобів для синтезу нейромоделей. Проте деякі програми для синтезу нейромоделей потребують глибокого розуміння принципів функціонування нейронних мереж, методів оптимізації та специфіки обробки даних. У результаті виникає ситуація, коли заявлена універсальність такого програмного забезпечення обмежується практичними навичками розробника. Крім того, програмні засоби синтезу нейронних мереж часто генерують складні моделі, структура яких є непрозорою для користувача. Це ускладнює аналіз причин прийняття рішень нейромоделлю, знижує рівень довіри до результатів і створює труднощі при їх впровадженні в критично важливі системи. Додатковим недоліком є ризик отримання

надмірно складних нейромоделей, оскільки автоматизовані алгоритми можуть формувати архітектури, які добре працюють на навчальних даних, але демонструють низьку здатність до узагальнення. Це потребує додаткового контролю та валідації результатів, що частково нівелює переваги автоматизації процесу синтезу нейромоделей. Не можна ігнорувати й проблему обмеженої гнучкості окремих програмних рішень. Деякі системи орієнтовані на конкретні типи задач або архітектур, що ускладнює їх адаптацію до нестандартних умов. У таких випадках виникає необхідність модифікації або доповнення інструментів, що підвищує складність розроблення. Тому актуальною є розробка програмного забезпечення для синтезу нейромоделей.

Сформульовано функціональні вимоги до програмного забезпечення для синтезу нейромоделей.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір мови програмування

При розробці програмного забезпечення для синтезу нейромоделей було використано мову програмування C# [15]-[16].

Мова програмування C# вирізняється синтаксичною чіткістю та сучасною архітектурою, що робить її універсальним інструментом для розробки програмних систем різного рівня складності. Вона поєднує принципи об'єктно-орієнтованого підходу з високим рівнем абстракції, дозволяючи реалізовувати складні алгоритми у зрозумілій і підтримуваний формі. Це створює передумови для швидкого розвитку програмних продуктів, адже код легко модифікувати, розширювати та адаптувати до нових вимог, не порушуючи загальної логіки системи [15]-[16].

Особливістю C# є тісна інтеграція з платформою .NET, що забезпечує доступ до великої кількості бібліотек для роботи з мережею, базами даних, обчисленнями, графікою та інтерфейсами користувача. Це дозволяє не витрачати час на реалізацію базових функцій, а зосередитися на специфічних завданнях програми. Підтримка асинхронного програмування і багатопотоковості дає змогу ефективно обробляти великі обсяги даних, що особливо важливо для сучасних застосунків, які потребують швидкого реагування та одночасного виконання численних операцій [15]-[16].

C# вирізняється безпекою коду, що забезпечується автоматичним управлінням пам'яттю через збирач сміття і механізмами обробки винятків. Це зменшує ймовірність критичних помилок і робить програмні продукти більш стабільними навіть у випадках високих навантажень або непередбачуваних умов виконання. Одночасно мова підтримує роботу з низькорівневими структурами даних, що дозволяє реалізовувати обчислювально інтенсивні алгоритми без значних втрат у швидкодії [15]-[16].

Важливою перевагою C# є кросплатформеність завдяки .NET Core і .NET 7+, що дозволяє запускати розроблені програми на Windows, Linux і

macOS без суттєвих змін у кодi. Це забезпечує гнучкість при виборі середовища виконання і спрощує інтеграцію програмних рішень у різноманітні системи. Крім того, C# активно використовується у розробці вебзастосунків, ігор, мобільних застосунків, систем машинного навчання та корпоративного програмного забезпечення, що підтверджує її універсальність [15]-[16].

Не менш важливою є підтримка сучасних парадигм програмування, включно з функціональними підходами, LINQ-запитами до даних, а також інтеграція з бібліотеками для роботи з штучним інтелектом і нейромережами. Це створює можливості для реалізації складних аналітичних і синтетичних процесів всередині програмного продукту. Розвинена екосистема інструментів розробки, таких як Visual Studio та інші IDE, забезпечує зручне налагодження, тестування і профілювання програм, що прискорює процес створення та оптимізації коду [15]-[16].

Таким чином, C# можна являє собою не просто мову програмування, а як комплексне середовище для створення сучасних програмних рішень. Вона поєднує зрозумілу синтаксичну структуру, багатий набір функціональних можливостей, безпеку та стабільність виконання, що робить її ефективним інструментом для розробки систем будь-якої складності, включаючи обчислювально інтенсивні та інтелектуальні застосунки [15]-[16].

Вибір мови програмування C# для розробки програмного забезпечення, призначеного для синтезу нейромоделей, обґрунтовується комплексом технічних та організаційних факторів, які підвищують ефективність розробки та зручність подальшого використання створених інструментів. Мова C# забезпечує високу продуктивність при роботі з об'єктно-орієнтованими структурами, що особливо актуально для побудови складних моделей нейронних мереж із багаторівневою архітектурою. Класова організація коду дозволяє чітко структурувати компоненти системи, забезпечуючи логічну ізоляцію різних модулів: генерацію архітектур, оптимізацію гіперпараметрів, навчання та оцінку моделей [15]-[16].

Мова програмування C# має потужну екосистему бібліотек і фреймворків, які спрощують реалізацію чисельних обчислень, обробки даних і паралельного виконання завдань. Наприклад, інтеграція з бібліотеками для лінійної алгебри, графічними інтерфейсами і сервісами хмарних обчислень дозволяє створювати програмні рішення, які легко масштабуються від локального тестування до виробничого впровадження. Така екосистема сприяє зменшенню часу розробки, оскільки багато базових компонентів вже готові до використання і не потребують реалізації з нуля [15]-[16].

Крім того, C# надає зручні засоби для створення графічних інтерфейсів користувача та візуалізації процесів синтезу нейромоделей. Це дає змогу інтегрувати механізми моніторингу, аналізу результатів і інтерпретації роботи моделей у зрозумілий і доступний спосіб. Візуальні компоненти полегшують контроль за процесом навчання, оцінку якості моделей і швидке виявлення помилок, що особливо важливо при розробці складних систем із великою кількістю параметрів [15]-[16].

Мова відзначається високим рівнем безпеки і стабільності, що забезпечується механізмами управління пам'яттю та системою обробки винятків. У контексті синтезу нейромоделей це дозволяє зменшити ризик помилок, пов'язаних із витоками ресурсів або некоректним виконанням обчислень, підвищуючи надійність і стійкість програмного забезпечення до навантажень. Безпечне управління ресурсами особливо важливе, якщо процес синтезу передбачає одночасне навчання кількох моделей або обробку великих обсягів даних [15]-[16].

Окремим фактором виступає сумісність C# із сучасними платформами та сервісами, включно з Windows, .NET Core і хмарними середовищами. Це створює гнучкість у виборі середовища виконання і дозволяє адаптувати програмне забезпечення під різні операційні системи і апаратні конфігурації. Крім того, підтримка багатопотоковості і асинхронних процесів робить можливим ефективно використання багатоядерних процесорів, що істотно

прискорює виконання ресурсомістких обчислень, притаманних синтезу нейромоделей [15]-[16].

Вибір C# доцільний також з точки зору командної розробки та підтримки проекту у довгостроковій перспективі. Мова має широке співтовариство розробників, доступ до навчальних матеріалів і професійної документації, що зменшує поріг входження і забезпечує ефективну комунікацію між учасниками проекту. Це особливо важливо, коли програмне забезпечення для синтезу нейромоделей розвивається і масштабуються функціональні можливості з часом, потребуючи постійного оновлення та підтримки [15]-[16].

Обґрунтування вибору мови програмування для розробки програмного забезпечення для синтезу нейромоделей наведено у таблиці 2.1.

Таблиця 2.1 – Обґрунтування вибору мови програмування для розробки програмного забезпечення для синтезу нейромоделей

Критерій порівняння мов програмування	Мова програмування		
	C#	Python	C++
Продуктивність виконання	+	+—	+
Бібліотеки для нейромереж	+	+	+—
Безпека коду та управління пам'яттю	+	+—	+—
Візуалізація та GUI	+	+—	—
Простота використання та документація	+	+	+—

Отже, для реалізації програмного забезпечення для синтезу нейромоделей обрано мову програмування C#, яка надає набір функціональних можливостей та забезпечує баланс між продуктивністю, структурованістю коду, безпекою і зручністю користування. Це створює

передумови для ефективної побудови, масштабування та підтримки складних систем, здатних автоматизувати процеси синтезу і навчання нейронних мереж у різноманітних прикладних сценаріях [15]-[16].

2.2 Вибір середовища розробки для створення програмного забезпечення для синтезу нейромоделей

В якості середовища розробки для створення програмного забезпечення для синтезу нейромоделей було обрано середовище Visual Studio [17]-[18].

Середовище розробки Visual Studio можна розглядати як комплексну платформу, що поєднує численні інструменти для створення, тестування, відлагодження та підтримки програмного забезпечення будь-якої складності. Його перевага полягає у високому рівні інтеграції різних компонентів розробки у єдиному робочому просторі, що дозволяє мінімізувати час на перемикання між окремими інструментами та підвищує ефективність процесу. Зручний інтерфейс із багаторівневою навігацією по проектах та файлами полегшує організацію великих проектів, де одночасно підтримуються численні модулі, бібліотеки та ресурси, що особливо важливо для розробки програмного забезпечення для синтезу нейромоделей [17]-[18].

Visual Studio вирізняється широкими можливостями для налагодження та аналізу коду. Крім стандартного покрокового виконання та відстеження змінних, середовище надає інструменти профілювання продуктивності, аналізу витрат пам'яті та виявлення вузьких місць у програмі. Це дозволяє оптимізувати обчислювальні процеси, які зазвичай є ресурсомісткими у сфері нейромережових обчислень. Додатково, система відстеження винятків та можливість налаштування складних сценаріїв налагодження підвищують стабільність і передбачуваність роботи програмного забезпечення [17]-[18].

Особливе значення має інтеграція Visual Studio з системами контролю версій, що дозволяє ефективно організовувати командну роботу над

проектами. Можливість безпосередньо працювати з Git, Azure DevOps та іншими системами у межах IDE спрощує управління версіями, відстеження змін та координацію між розробниками. Це робить середовище ідеальним для довгострокових і масштабованих проєктів, де необхідна постійна підтримка та оновлення програмних модулів [17]-[18].

Visual Studio також забезпечує інтеграцію з сучасними фреймворками та бібліотеками, включно з .NET, ML.NET, TensorFlow.NET та іншими, що робить можливим безшовну роботу з алгоритмами машинного навчання, синтезу нейромоделей і обробки даних. Можливість швидко підключати сторонні пакети і компоненти зменшує час розробки та дозволяє зосередитися на реалізації специфічних функціональних завдань [17]-[18].

Не менш важливою є підтримка розробки графічних інтерфейсів та візуалізацій. Visual Studio надає інструменти для створення інтерактивних панелей, діаграм, графіків і вікон моніторингу процесів, що дозволяє користувачам спостерігати за ходом синтезу нейромоделей, аналізувати результати і здійснювати корекцію параметрів у реальному часі [17]-[18].

Також Visual Studio підтримує сучасні підходи до автоматизованого тестування, аналізу якості коду і рефакторингу. Інструменти статичного та динамічного аналізу допомагають виявляти потенційні помилки ще на етапі розробки, підвищуючи надійність і стабільність продукту. Це особливо важливо для програмного забезпечення, яке виконує складні математичні обчислення і взаємодіє з великими обсягами даних [17]-[18].

Вибір середовища розробки Visual Studio для створення програмного забезпечення, призначеного для синтезу нейромоделей, обґрунтовується поєднанням потужних функціональних можливостей, інтегрованих інструментів та зручності роботи, що значно підвищує ефективність розробки складних систем. Насамперед Visual Studio забезпечує комплексне середовище для написання, налагодження та тестування коду, дозволяючи поєднувати різні компоненти програмного продукту у єдиному робочому просторі. Такий підхід сприяє логічній структуризації проєкту, зменшує ймовірність помилок і

полегшує підтримку програмного забезпечення у довгостроковій перспективі [17]-[18].

Середовище надає потужні засоби налагодження та профілювання, які дозволяють відстежувати виконання коду покроково, аналізувати витрати пам'яті та продуктивність окремих модулів. Для програмного забезпечення синтезу нейромоделей це особливо важливо, оскільки процеси навчання та оптимізації моделей є ресурсомісткими і потребують ретельного контролю. Можливість візуалізувати структуру програмних компонентів, виявляти вузькі місця і оптимізувати обчислювальні потоки дозволяє підвищити швидкодію та стабільність системи [17]-[18].

Visual Studio забезпечує інтеграцію з численними бібліотеками та фреймворками, що використовуються для роботи з нейромережами, машинним навчанням і обробкою даних. Це спрощує підключення готових рішень для лінійної алгебри, паралельних обчислень та візуалізації результатів, дозволяючи розробнику зосередитися на реалізації специфічних функцій програми, а не на створенні базових компонентів з нуля. Крім того, підтримка багатоплатформених розробок через .NET Core або .NET 7+ дозволяє створювати застосунки, які можуть працювати на Windows, Linux або macOS без суттєвих змін у коді [17]-[18].

Особливу увагу слід приділити інтеграції Visual Studio з системами контролю версій, зокрема Git, що дозволяє організовувати спільну роботу команди над великими проєктами. Це забезпечує зручне відстеження змін, керування версіями і швидке виявлення проблем, що виникають під час розробки. Для складних програмних рішень із синтезу нейромоделей така функціональність критично важлива, оскільки моделі часто оновлюються, а код розширюється, і потрібен надійний механізм контролю змін [17]-[18].

Також Visual Studio надає можливості для побудови графічних інтерфейсів користувача та візуалізації даних. У випадку програмного забезпечення для синтезу нейромоделей це дозволяє створювати інтерактивні панелі для моніторингу процесу навчання, оцінки якості моделей і аналізу

результатів. Інтерактивні графіки, діаграми та панелі керування підвищують зручність використання програмного продукту, роблять процес роботи більш інтуїтивним і зрозумілим навіть для користувачів із обмеженим технічним досвідом [17]-[18].

Не можна не відзначити і підтримку автоматичного тестування, аналізу коду та системи підказок і автозавершення, які зменшують ймовірність синтаксичних та логічних помилок. Для складних систем, що використовують інтелектуальні алгоритми і синтез нейромоделей, це дозволяє забезпечити стабільність і передбачуваність роботи програмного забезпечення [17]-[18].

Обґрунтування вибору середовища розробки для створення програмного забезпечення для синтезу нейромоделей наведено у таблиці 2.2.

Таблиця 2.2 – Обґрунтування вибору середовища розробки для створення програмного забезпечення для синтезу нейромоделей

Критерій порівняння середовищ розробки	Середовища розробки		
	Visual Studio	Rider	MonoDevelop
Інтеграція з бібліотеками для машинного навчання	+	+—	—
Потужність налагодження та профілювання	+	+—	+—
Графічний інтерфейс і візуалізація	+	+—	—
Інструменти для командної роботи	+	+	+—
Автоматизоване тестування та аналіз коду	+	+—	—

Використання Visual Studio для розробки програмного забезпечення для синтезу нейромоделей доцільне завдяки потужним засобам налагодження,

інтеграції з бібліотеками, підтримці кросплатформених рішень, можливостям для візуалізації даних і ефективній організації командної роботи. Це середовище забезпечує баланс між гнучкістю, продуктивністю та зручністю, що робить процес розробки складних систем більш ефективним і контрольованим [17]-[18].

Таким чином, для створення програмного забезпечення для синтезу нейромоделей обрано середовище розробки Visual Studio, що поєднує розробку, відлагодження, аналіз, тестування і підтримку програмних систем. Завдяки цьому середовище Visual Studio підвищує продуктивність розробників, забезпечує стабільність і надійність програмного забезпечення, спрощує інтеграцію з бібліотеками і сервісами, а також полегшує візуалізацію і моніторинг складних процесів, таких як синтез нейромоделей.

2.3 Висновки за розділом 2

Для реалізації програмного забезпечення для синтезу нейромоделей обрано мову програмування C#, яка надає набір функціональних можливостей та забезпечує баланс між продуктивністю, структурованістю коду, безпекою і зручністю користування. Це створює передумови для ефективної побудови, масштабування та підтримки складних систем, здатних автоматизувати процеси синтезу і навчання нейронних мереж у різноманітних прикладних сценаріях.

Для створення програмного забезпечення для синтезу нейромоделей обрано середовище розробки Visual Studio, що поєднує розробку, відлагодження, аналіз, тестування і підтримку програмних систем. Завдяки цьому середовище Visual Studio підвищує продуктивність розробників, забезпечує стабільність і надійність програмного забезпечення, спрощує інтеграцію з бібліотеками і сервісами, а також полегшує візуалізацію і моніторинг складних процесів, таких як синтез нейромоделей.

3 ОПИС ПРОГРАМИ

3.1 Структура программного обеспечения для синтеза нейромоделей

Структуру программного обеспечения для синтеза нейромоделей наведено на рис. 3.1.

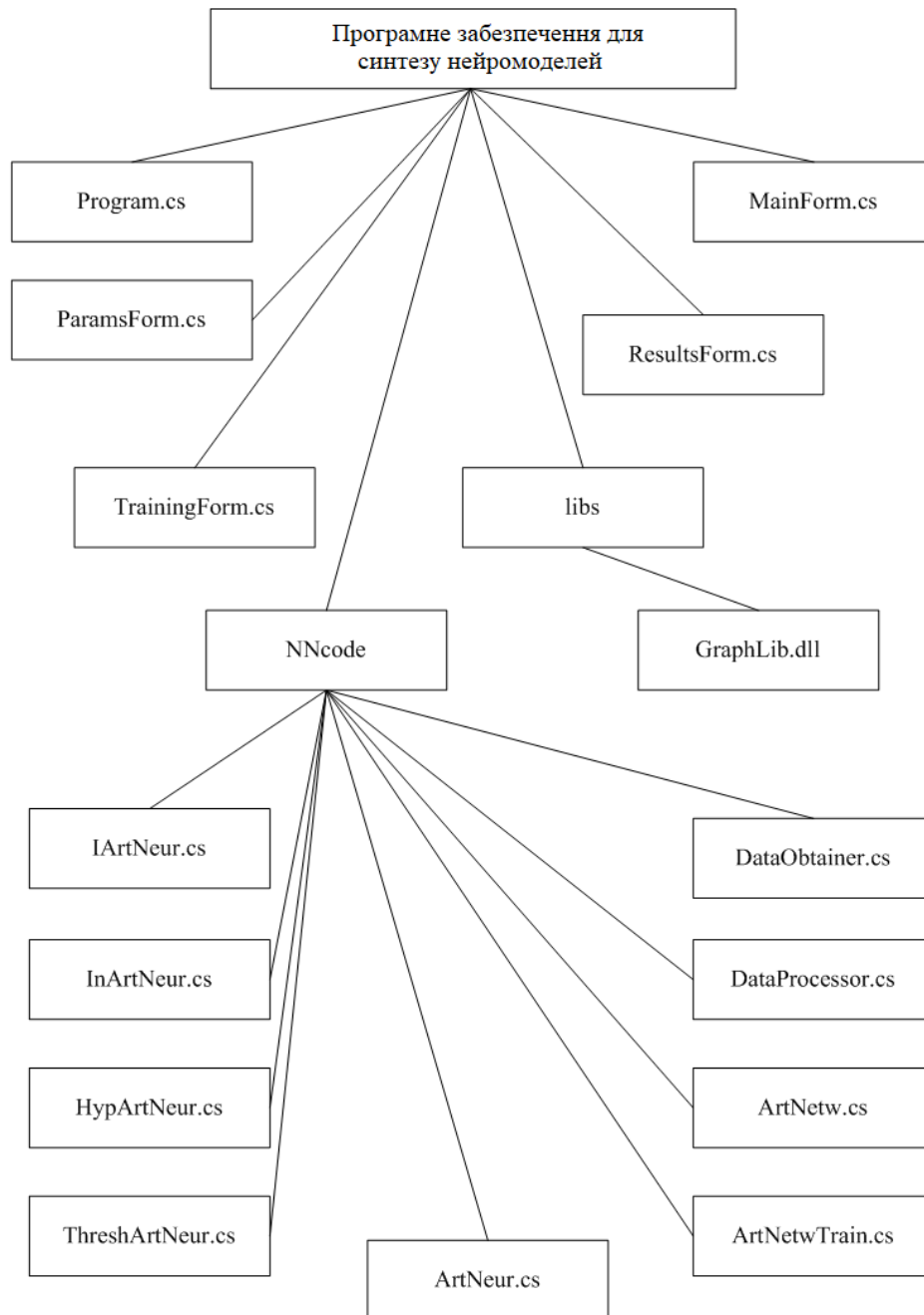


Рисунок 3.1 – Структура программного обеспечения для синтеза нейромоделей

Структура програмного забезпечення для синтезу нейромоделей побудована на засадах модульності, що забезпечує як зручність розробки, так і подальшу підтримку системи. У межах загального проєкту реалізовано підхід, за якого логіка взаємодії з користувачем відокремлена від обчислювального ядра, відповідального за створення та функціонування багатошарових нейромереж. Така організація дозволяє уникнути надмірного ускладнення коду та забезпечує можливість незалежного розвитку окремих підсистем.

Центральним елементом архітектури виступає внутрішня бібліотека NNCode, яка інкапсулює всі механізми, пов'язані з побудовою, обробкою та навчанням нейромоделей. У її межах реалізовано базову абстракцію штучного нейрону, яка визначає фундаментальні властивості та поведінку елементарних компонентів мережі. Взаємодія з такими об'єктами організована через відповідний інтерфейс, що забезпечує уніфікований доступ до їх функціоналу та сприяє гнучкому розширенню системи. Завдяки цьому досягається можливість створення різних типів нейронів, які можуть відрізнятися функціями активації або роллю в загальній структурі мережі.

Побудова багатошарової нейромережі реалізується як окремий рівень абстракції, що відповідає за об'єднання нейронів у єдину систему та керування їх взаємозв'язками. Такий підхід дозволяє формувати мережі різної складності без необхідності змінювати базові компоненти. Додатково передбачено механізми підготовки даних, які включають їх отримання з зовнішніх джерел та попередню обробку. Зокрема, реалізовано засоби формування вибірок і нормалізації значень, що є необхідною умовою ефективного навчання нейромережевих моделей. Ці етапи інтегровані в єдиний процес, що забезпечує узгодженість даних і підвищує якість результатів.

Окрему роль у структурі займає підсистема навчання, яка реалізує алгоритми налаштування параметрів нейромережі. Вона взаємодіє з базовими компонентами мережі та забезпечує ітеративне вдосконалення її поведінки відповідно до заданих критеріїв. При цьому враховується можливість використання різних типів нейронів, включаючи вхідні елементи, нейрони

зміщення та нейрони з нелінійними функціями активації. Така гнучкість дозволяє моделювати широкий спектр задач і адаптувати систему до різних умов використання.

Точка входу до програмного забезпечення реалізована у вигляді окремого модуля, який ініціалізує роботу системи та забезпечує зв'язок між користувацьким інтерфейсом і внутрішньою логікою. Взаємодія з користувачем організована через набір графічних форм, кожна з яких виконує чітко визначену функцію. Головна форма виступає центром керування, забезпечуючи доступ до основних параметрів і відображення результатів. Інші форми відповідають за налаштування структури мережі, параметрів навчання та способів представлення отриманих даних, що дозволяє розділити процес роботи на логічні етапи та зробити його більш зрозумілим.

Візуалізація результатів реалізується шляхом інтеграції зовнішньої бібліотеки, яка підключається у вигляді динамічного компонента. Це рішення дозволяє використовувати готові інструменти для побудови графіків і діаграм без необхідності їх розробки з нуля. Розміщення таких компонентів у окремій директорії підкреслює модульний характер системи та спрощує їх оновлення або заміну в майбутньому. У результаті досягається підвищення гнучкості програмного забезпечення та можливість його адаптації до нових вимог.

Таким чином, структура програмного забезпечення для синтезу нейромоделей базується на принципах модульності та абстракції, що дозволяє ефективно реалізувати процес синтезу нейромоделей. Вона забезпечує зручність розробки, можливість масштабування та інтеграцію додаткових компонентів без суттєвого впливу на вже існуючу функціональність. У підсумку така структура створює надійну основу для побудови гнучких і продуктивних систем, здатних вирішувати складні задачі у сфері штучного інтелекту.

3.2 Функціонування програмного забезпечення для синтезу нейромоделей

Функціонування програмного забезпечення для синтезу нейромоделей подамо за допомогою схеми, зображеної на рис. 3.2.

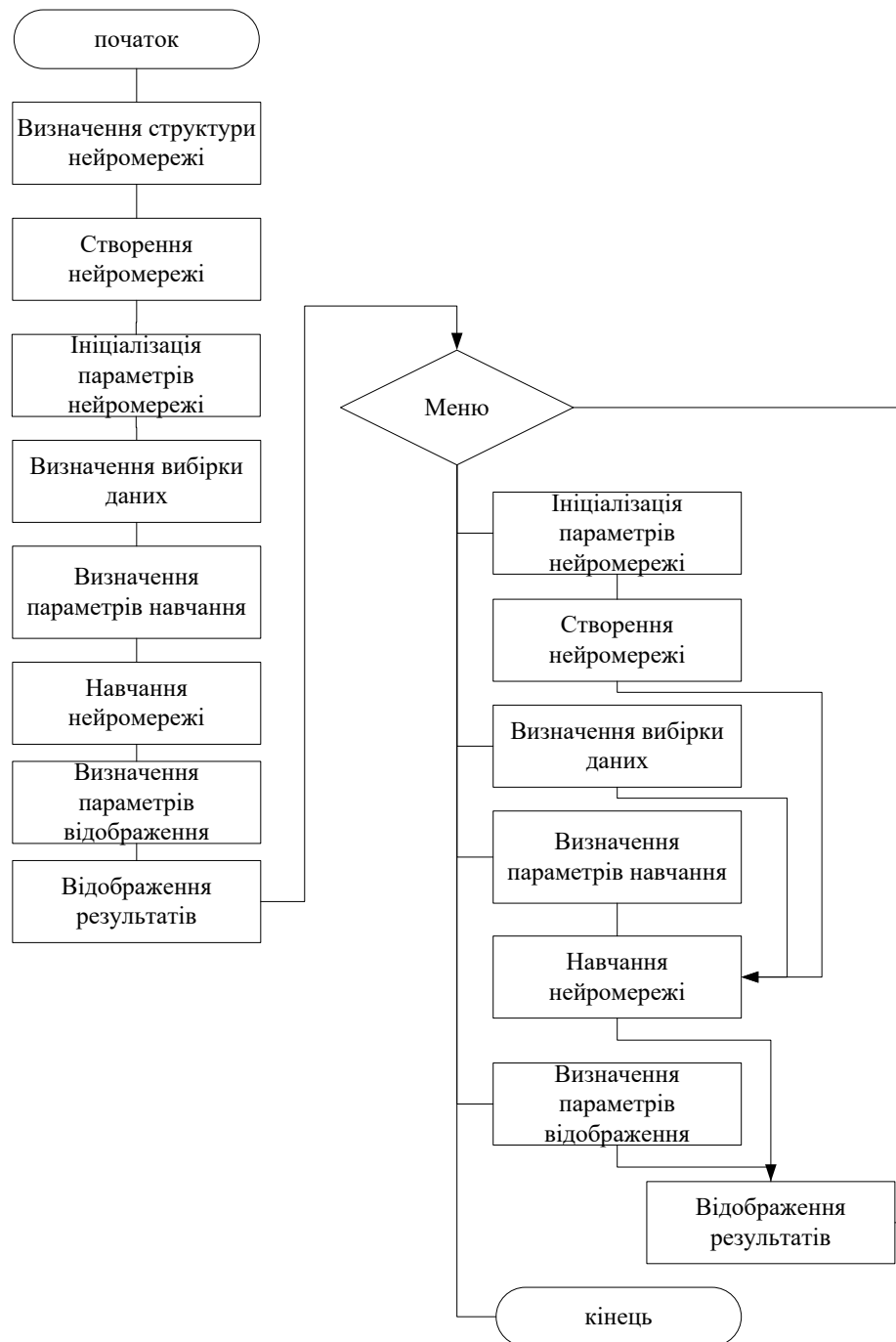


Рисунок 3.2 – Схема функціонування програмного забезпечення для синтезу нейромоделей

Функціонування програмного забезпечення для синтезу нейромоделей організовано як цілісний процес, у якому поєднуються автоматизовані етапи побудови нейромережі та інтерактивна взаємодія користувача з системою. Загальна логіка роботи передбачає початкове проходження всіх необхідних стадій у визначеній послідовності, після чого відкривається можливість гнучкого керування окремими параметрами з метою уточнення або оптимізації отриманих результатів. Такий підхід дозволяє спочатку сформулювати базову модель, а надалі виконувати її адаптацію без необхідності повторного проходження всього процесу з початку.

На початковому етапі здійснюється формування структури нейромережі, що передбачає визначення кількості шарів, типів нейронів і характеру їх взаємозв'язків. Після цього відбувається створення відповідної моделі, яка виступає основою для подальших обчислень. Наступним кроком є ініціалізація параметрів мережі, де задаються початкові значення вагових коефіцієнтів та інших характеристик, що впливають на процес навчання. У цей момент формується початковий стан системи, який визначає подальшу динаміку її роботи.

Подальше функціонування передбачає роботу з даними, які використовуються для навчання нейромережі. Здійснюється вибір або завантаження вибірки, після чого виконується її обробка, що забезпечує узгодженість і придатність даних для використання в алгоритмах навчання. Важливим аспектом є визначення параметрів навчання, які включають налаштування швидкості навчання, кількості ітерацій та інших характеристик, що впливають на ефективність і стабільність процесу. Саме на цьому етапі закладаються умови, за яких нейромережа буде адаптуватися до вхідних даних.

Після підготовчих дій відбувається безпосередній процес навчання, у ході якого нейромережа змінює свої параметри відповідно до заданого алгоритму. Результатом цього етапу є сформована модель, здатна виконувати поставлені завдання на основі отриманого досвіду. Паралельно з цим

передбачено налаштування параметрів відображення, які визначають спосіб представлення результатів користувачу. Завершальним кроком стає візуалізація отриманих даних, що дозволяє оцінити ефективність моделі та зробити висновки щодо її якості.

Особливістю функціонування програмного забезпечення для синтезу нейромоделей є поділ процесу на дві логічні частини. Перший етап має послідовний характер і забезпечує повну ініціалізацію системи, включаючи всі необхідні параметри. Другий етап орієнтований на інтерактивну взаємодію, де користувач отримує можливість змінювати окремі параметри через меню без порушення вже сформованої базової конфігурації. Це дозволяє оперативно впливати на результати, змінюючи, наприклад, параметри навчання або дані вибірки, і одразу спостерігати наслідки таких змін.

Такий підхід до організації функціонування забезпечує поєднання стабільності та гнучкості. З одного боку, початкове послідовне виконання всіх етапів гарантує коректність формування моделі, а з іншого — можливість часткової зміни параметрів дозволяє ефективно проводити експерименти та оптимізацію без значних витрат часу. Така організація дозволяє не лише створювати ефективні моделі, але й забезпечує можливість їх швидкої адаптації до нових умов, що робить систему зручною для досліджень і практичного застосування у сфері нейромережових технологій. Це особливо важливо у задачах синтезу нейромоделей, де результат значною мірою залежить від налаштувань і характеристик вхідних даних.

3.3 Архітектура програмного забезпечення для синтезу нейромоделей

Архітектура програмного забезпечення для синтезу нейромоделей побудована за принципом чіткого розмежування рівнів відповідальності, що забезпечує відокремлення обчислювального ядра від засобів взаємодії з користувачем. У межах такого підходу центральне місце займає модуль, який

реалізує логіку створення, налаштування та навчання нейромереж, тоді як інтерфейс користувача виконує функції керування процесом і відображення результатів. Це дозволяє досягти незалежності між внутрішніми алгоритмами та способом їх представлення, що значно спрощує як розвиток системи, так і її супровід.

Схему архітектури програмного забезпечення для синтезу нейромоделей наведено на рис. 3.3.

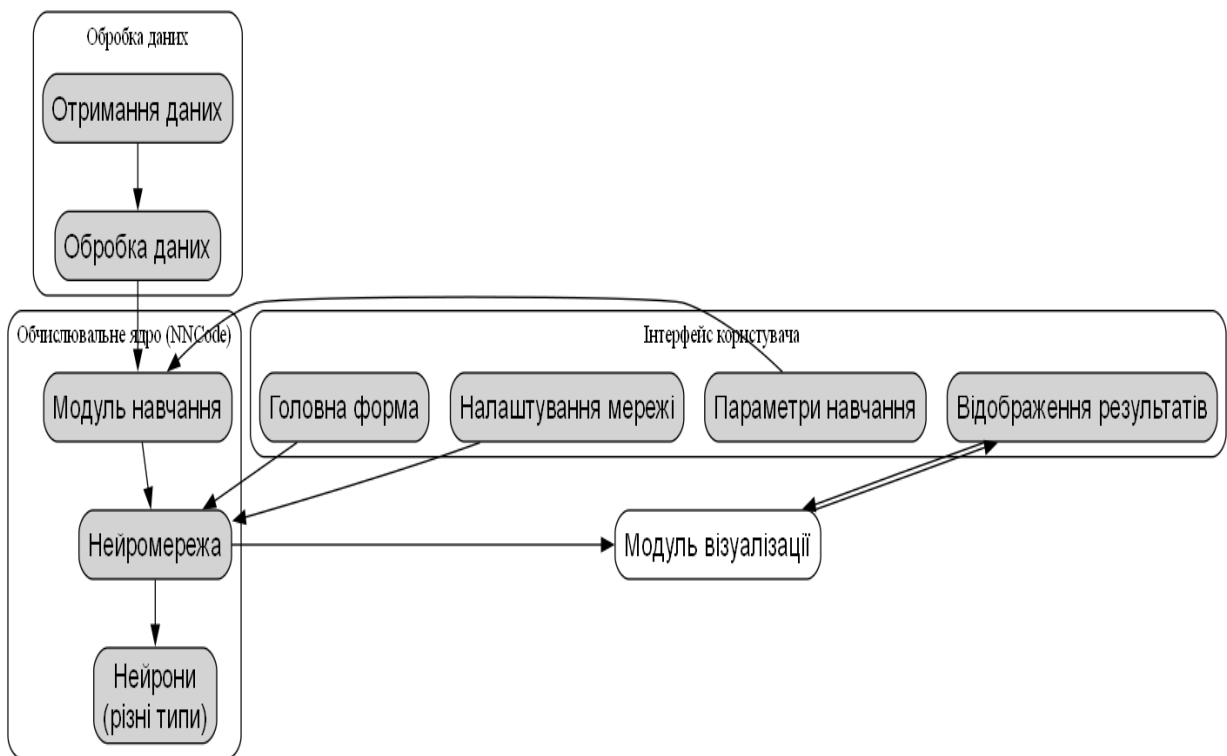


Рисунок 3.3 – Архітектура ПЗ для синтезу нейромоделей

Взаємодія між графічним інтерфейсом і обчислювальним ядром організована через механізм викликів методів і передачі параметрів, які визначають конфігурацію нейромережі та режими її роботи. Користувач, працюючи з формами програми, фактично задає початкові умови, що передаються до відповідних компонентів NNCode. У відповідь система виконує необхідні обчислення і повертає результати, які надалі візуалізуються у зручному вигляді. Така модель взаємодії забезпечує інкапсуляцію складної

логіки та дозволяє уникнути прямого втручання інтерфейсного рівня у внутрішню структуру нейромережі.

Потік даних у системі має чітко визначений напрям і проходить кілька взаємопов'язаних стадій. Спочатку відбувається надходження вхідної інформації, яка формується на основі зовнішніх джерел і передається до підсистеми обробки. Після цього дані проходять етап підготовки, де виконуються необхідні перетворення, зокрема нормалізація та приведення до формату, придатного для використання в нейромережі. Далі підготовлені значення надходять до модулів, відповідальних за побудову і навчання моделі, де відбувається їх безпосередня обробка. Результати обчислень повертаються у вигляді структурованих даних, які передаються назад до інтерфейсного рівня для подальшого відображення і аналізу.

Залежності між модулями організовані таким чином, щоб забезпечити мінімальне зчеплення і максимальну узгодженість компонентів. Основні елементи, що відповідають за реалізацію нейронів і мережі, функціонують незалежно від механізмів введення і виведення інформації. При цьому інтерфейс взаємодії визначається через узгоджені контракти, що дозволяє замінювати або розширювати окремі частини системи без порушення її загальної структури. Наприклад, модулі, які відповідають за підготовку даних, можуть бути модифіковані або доповнені новими алгоритмами без необхідності змін у компонентах, що реалізують саму нейромережу.

Архітектура також враховує можливість розширення функціональності, що досягається за рахунок використання абстракцій і виділення окремих рівнів логіки. Додавання нових типів нейронів, алгоритмів навчання або способів візуалізації не потребує кардинальної перебудови системи, оскільки кожен із цих аспектів реалізується у межах відповідних модулів. Така організація сприяє масштабованості програмного забезпечення і дозволяє адаптувати його до нових задач у сфері синтезу нейромоделей.

Важливою характеристикою архітектури є також її орієнтація на повторне використання компонентів. Внутрішня бібліотека NNCode може

бути використана незалежно від конкретного інтерфейсу користувача, що відкриває можливість інтеграції з іншими системами або створення альтернативних інтерфейсів. Це підвищує універсальність програмного забезпечення і розширює сферу його застосування.

Отже, архітектура програмного забезпечення для синтезу нейромоделей характеризується модульністю, чітким розподілом відповідальності та організованим потоком даних, що забезпечує ефективну взаємодію між компонентами системи. Такий підхід дозволяє створити гнучке, масштабоване і зручне у використанні програмне рішення, здатне адаптуватися до змін вимог і забезпечувати стабільну роботу у складних обчислювальних задачах.

3.4 Проєктування інтерфейсу програмного забезпечення для синтезу нейромоделей

Проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для синтезу нейромоделей виконано з урахуванням вимог технічного завдання. При розробленні програмного забезпечення для синтезу нейромоделей враховані функціональні вимоги до програми:

- підтримка можливості створення та налаштування багатошарових нейромереж із вибором кількості шарів та нейронів у кожному шарі;
- підтримка можливості вибору функцій активації для нейронів мережі;
- можливість завантаження вхідних даних для навчання нейромережі;
- можливість вибору характеристик процесу навчання нейромоделі із визначенням параметрів навчання, таких як швидкість навчання, контроль похибки, максимально допустима кількість ітерацій та час виконання;
- можливість оцінювання точності нейромоделі;
- підтримка інтерактивної роботи користувача з програмою, включаючи зміну параметрів моделі та вибір наборів даних для синтезу нейромоделі.

Інтерфейс головної форми програмного забезпечення для синтезу нейромоделей наведено на рис. 3.4.

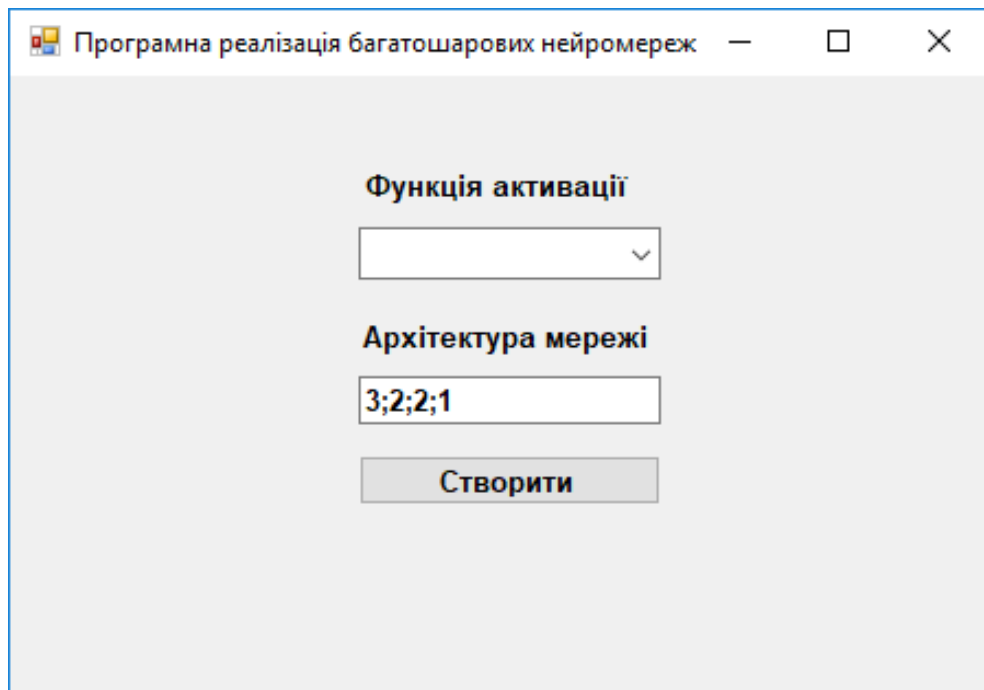


Рисунок 3.4 – Інтерфейс головної форми програмного забезпечення для синтезу нейромоделей

Інтерфейс прототипу форми вибору вхідних даних наведено на рис. 3.5.

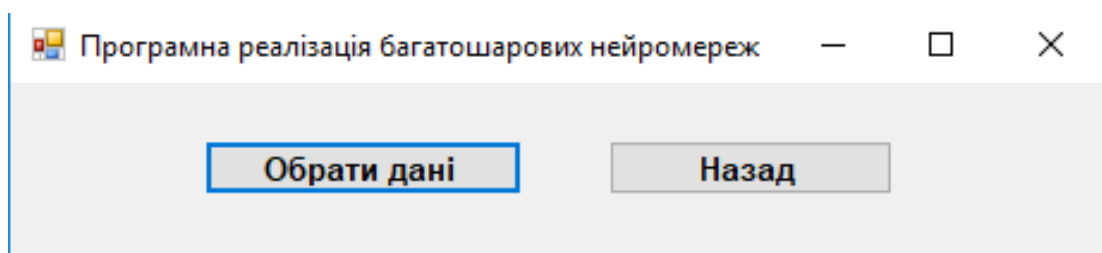


Рисунок 3.5 – Інтерфейс прототипу форми вибору вхідних даних

Інтерфейс форми встановлення параметрів та відображення результатів процесу синтезу нейронмережевої моделі наведено на рис. 3.6.

The screenshot shows a software window titled "Програмна реалізація багатошарових нейромереж". It contains several input fields for parameters and a section for training results.

Parameter	Value
Альфа	1
Лямбда	0.0001
Максимальна похибка	0.01
Прийнятна дельта	0.00001
Час завершення	10
Максимальна кількість ітерацій	990

Below the parameters, there are two buttons: "Відображення" (disabled) and "Навчання" (active). The "Навчання" button is highlighted with a blue border.

Result	Value
Ітерацій	589
Похибка на навчальній вибірці	0.00999
Похибка на тестовій вибірці	0.00377

At the bottom right, there is a button labeled "Назад" (Back).

Рисунок 3.6 – Форма встановлення параметрів та відображення результатів процесу синтезу нейромережевої моделі

3.5 Висновки за розділом 3

Розроблено програмне забезпечення для синтезу нейромоделей.

Запропоновано структуру програмного забезпечення для синтезу нейромоделей. Визначено, що структура програмного забезпечення для синтезу нейромоделей базується на принципах модульності та абстракції, що дозволяє ефективно реалізувати процес синтезу нейромоделей. Вона

забезпечує зручність розробки, можливість масштабування та інтеграцію додаткових компонентів без суттєвого впливу на вже існуючу функціональність. Як наслідок, така структура створює надійну основу для побудови гнучких і продуктивних систем, здатних вирішувати складні задачі у сфері штучного інтелекту.

Функціонування програмного забезпечення для синтезу нейромоделей організовано як цілісний процес, у якому поєднуються автоматизовані етапи побудови нейромережі та інтерактивна взаємодія користувача з системою. Загальна логіка роботи передбачає початкове проходження всіх необхідних стадій у визначеній послідовності, після чого відкривається можливість гнучкого керування окремими параметрами з метою уточнення або оптимізації отриманих результатів. Такий підхід дозволяє спочатку сформулювати базову модель, а надалі виконувати її адаптацію без необхідності повторного проходження всього процесу з початку.

Архітектура програмного забезпечення для синтезу нейромоделей побудована за принципом чіткого розмежування рівнів відповідальності, що забезпечує відокремлення обчислювального ядра від засобів взаємодії з користувачем. У межах такого підходу центральне місце займає модуль, який реалізує логіку створення, налаштування та навчання нейромереж, тоді як інтерфейс користувача виконує функції керування процесом і відображення результатів. Це дозволяє досягти незалежності між внутрішніми алгоритмами та способом їх представлення, що значно спрощує як розвиток системи, так і її супровід.

Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для синтезу нейромоделей.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення й умови застосування програми

Програма призначена для синтезу нейромоделей. Програма написана на мові C#, яка надає набір функціональних можливостей та забезпечує баланс між продуктивністю, структурованістю коду, безпекою і зручністю користування. В якості середовища розробки для створення програмного забезпечення для синтезу нейромоделей обрано Visual Studio, що поєднує розробку, відлагодження, аналіз, тестування і підтримку програмних систем.

Програмне забезпечення для синтезу нейромоделей забезпечує виконання таких функцій:

- підтримка можливості створення та налаштування багатошарових нейромереж із вибором кількості шарів та нейронів у кожному шарі;
- підтримка можливості вибору функцій активації для нейронів мережі;
- можливість завантаження вхідних даних для навчання нейромережі;
- можливість вибору характеристик процесу навчання нейромоделі із визначенням параметрів навчання, таких як швидкість навчання, контроль похибки, максимально допустима кількість ітерацій та час виконання;
- можливість оцінювання точності нейромоделі;
- підтримка інтерактивної роботи користувача з програмою, включаючи зміну параметрів моделі та вибір наборів даних для синтезу нейромоделі.

Для роботи програмного забезпечення для синтезу нейромоделей потрібні такі технічні та програмні ресурси: центральний процесор з тактовою частотою не нижче 2.5 ГГц, бажано з кількістю ядер не менше чотирьох, що дозволяє ефективно виконувати паралельні обчислення; обсяг оперативної пам'яті повинен становити не менше 8 ГБ, що забезпечує можливість роботи з вибірками середнього розміру та зберігання проміжних результатів обчислень,

для більш складних задач або великих наборів даних доцільним є використання 16 ГБ оперативної пам'яті або більше.

Накопичувач даних повинен мати достатній обсяг для збереження програмного забезпечення, бібліотек, вхідних даних і результатів обчислень; рекомендовано використовувати твердотільний накопичувач обсягом не менше 256 ГБ, що забезпечує високу швидкість доступу до даних.

З точки зору програмного забезпечення необхідною є наявність операційної системи, сумісної з платформою .NET, зокрема Windows 10 або новіших. Обов'язковою умовою є встановлення середовища виконання .NET відповідної версії, яке забезпечує коректну роботу програмних модулів. Для розробки, налагодження та модифікації програмного продукту доцільно використовувати інтегроване середовище розробки Visual Studio, яке надає повний набір інструментів для роботи з мовою C# та графічним інтерфейсом.

Таким чином, для ефективного функціонування програмного забезпечення необхідне поєднання достатньо продуктивної апаратної платформи та сучасного програмного середовища, що забезпечує стабільність, швидкодію та можливість масштабування при розв'язанні задач синтезу нейромоделей.

Функціональні характеристики розробленого програмного забезпечення для синтезу нейромоделей наведено у технічному завданні (додаток А). Фрагмент тексту програмного забезпечення для синтезу нейромоделей наведено у додатку Б.

4.2 Характеристики програми для синтезу нейромоделей

Програмне забезпечення для синтезу нейромоделей характеризується здатністю забезпечувати повний цикл роботи з нейромережами, починаючи від формування їх структури і завершуючи отриманням та інтерпретацією результатів. Його функціональність орієнтована на автоматизацію процесів побудови моделей, що дозволяє зменшити складність ручного налаштування

та підвищити ефективність роботи з даними. У межах такого програмного продукту реалізується можливість гнучкого формування архітектури нейромережі, що дає змогу адаптувати систему до різних типів задач і вхідних даних.

Важливою характеристикою виступає модульна організація, за якої окремі компоненти відповідають за різні аспекти функціонування системи. Такий підхід дозволяє розмежувати процеси обробки даних, навчання моделі та взаємодії з користувачем, забезпечуючи зручність розробки і підтримки програмного продукту. Завдяки цьому досягається можливість незалежного вдосконалення окремих частин системи без необхідності змінювати її загальну структуру.

Характерною особливістю є наявність засобів керування процесом навчання нейромоделей, які дозволяють змінювати параметри функціонування нейромережі та спостерігати вплив цих змін на результати. Це забезпечує інтерактивність роботи та дає змогу виконувати налаштування моделі у процесі її функціонування. При цьому зберігається цілісність вже отриманих результатів, що дозволяє уникнути повторного виконання всіх етапів.

Значну роль відіграє можливість візуалізації результатів, яка забезпечує наочне представлення процесів і підсумків роботи нейромережі. Це полегшує аналіз ефективності моделі та сприяє прийняттю обґрунтованих рішень щодо її вдосконалення. Інтерфейс користувача організовано таким чином, щоб забезпечити інтуїтивно зрозумілу взаємодію з системою та швидкий доступ до основних функцій.

Окремо варто відзначити орієнтацію програмного забезпечення на розширюваність і адаптивність. Його архітектура дозволяє інтегрувати нові алгоритми, змінювати способи обробки даних та розширювати функціональні можливості без істотного впливу на вже існуючі компоненти. Це робить систему придатною для використання в умовах, де вимоги до обробки інформації можуть змінюватися.

Таким чином, програмне забезпечення для синтезу нейромоделей характеризується гнучкістю, модульністю, здатністю до автоматизації складних процесів та зручністю взаємодії з користувачем, що забезпечує ефективне вирішення задач, пов'язаних із побудовою та використанням нейромережових моделей.

4.3 Інструкція по експлуатації програми

4.3.1 Звернення до програми

Звернення до програмного забезпечення для синтезу нейромоделей відбувається шляхом запуску основного файлу, після чого користувачу відкривається стартове вікно програми.

4.3.2 Вхідні й вихідні дані

Вхідними даними до програмного забезпечення для синтезу нейромоделей є параметри нейромережі, включно з кількістю нейронів у кожному шарі, типами функцій активації та параметрами навчання, а також навчальні набори даних.

Вихідними даними програмного забезпечення для синтезу нейромоделей є результати синтезу нейромережі, зокрема значення ваг нейронів і зміщень, показники похибок і якість узгодження моделі з вихідними даними.

4.3.3 Повідомлення

Користувач програмного забезпечення для синтезу нейромоделей може отримати такі повідомлення: повідомлення про успішне завантаження даних або ініціалізацію нейромережі, попередження про некоректний формат даних, а також повідомлення про завершення навчання нейромережової моделі.

4.4 Виконання програмного забезпечення для синтезу нейромоделей

Після запуску програмного забезпечення для синтезу нейромоделей користувачу відображається головна форма (рис. 4.1). Ця форма надає можливість обирати функції активації для нейронів мережі, задавати конфігурацію шарів та запускати процес створення нейромережі. Інтерфейс організовано таким чином, щоб користувач міг вибирати між доступними функціями активації, задавати кількість нейронів на кожному шарі через текстове поле та ініціювати формування структури кнопкою.

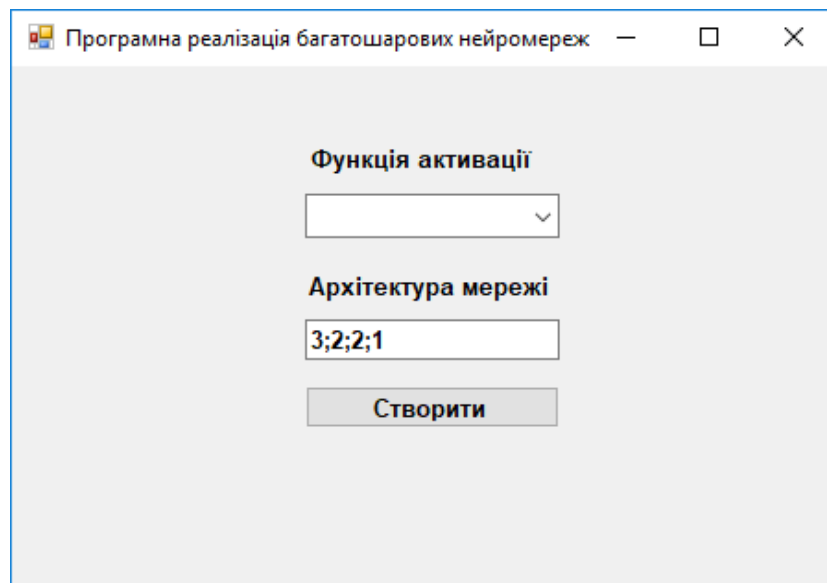


Рисунок 4.1 – Головна форма програмного забезпечення для синтезу нейромоделей

Після підтвердження параметрів програма створює (ініціалізує) нейромережу згідно з введеними налаштуваннями і переходить до наступного вікна (рис. 4.2). Ця форма дозволяє обирати файли, що містять вибірку для навчання мережі, за допомогою стандартного вікна вибору файлів. Водночас передбачена можливість повернення на попередній етап для повторного створення моделі у разі потреби.

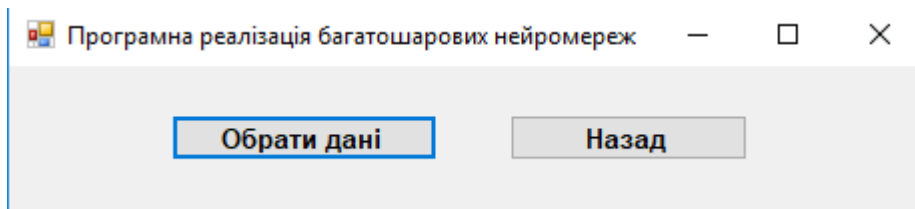


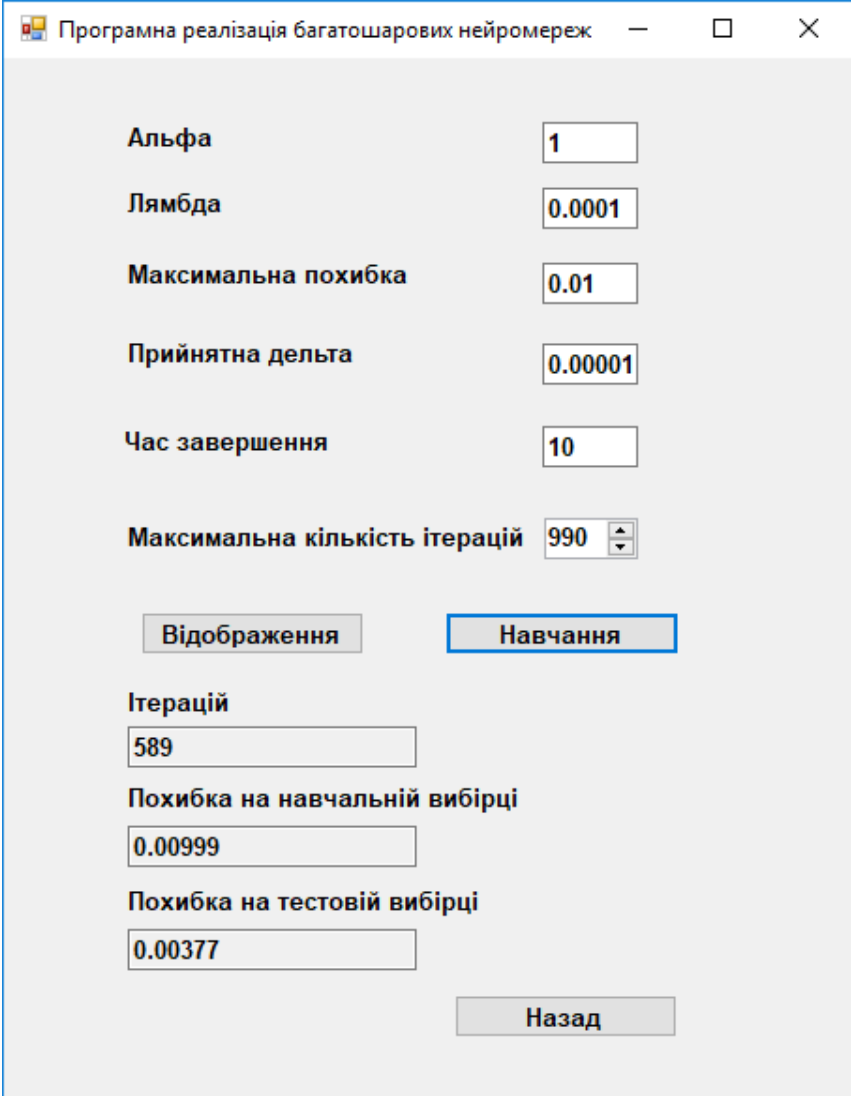
Рисунок 4.2 – Форма вибору вхідного набору даних для синтезу нейромережі

Після визначення вхідної інформації відкривається форма, призначена для встановлення параметрів навчання (рис. 4.3), де можна задавати основні величини, що впливають на процес оптимізації, контрольні показники завершення навчання та обмеження на кількість ітерацій.

Рисунок 4.3 – Форма встановлення параметрів навчання нейромоделей

Після введення значень активується кнопка запуску навчання, і програма починає обчислювальні операції синтезу нейромережі відповідно до встановлених параметрів.

По завершенні навчання користувач отримує доступ до результатів у вигляді текстових полів, де відображаються кількість виконаних ітерацій, розрахована похибка на навчальній та тестовій вибірках (рис. 4.4). Відповідність отриманих показників заданим параметрам дозволяє оцінити ефективність проведеного навчання. Інтерфейс забезпечує повернення на попередні кроки для корекції вибірки даних або налаштувань структури нейромоделі, якщо це необхідно.



The screenshot shows a software window titled "Програмна реалізація багатoshарових нейромереж". It contains a list of parameters for training a neural network, each with a corresponding input field. Below the parameters are two buttons: "Відображення" and "Навчання". The "Навчання" button is highlighted with a blue border. Below the buttons, there are three more input fields showing the results of the training: "Ітерацій" (589), "Похибка на навчальній вибірці" (0.00999), and "Похибка на тестовій вибірці" (0.00377). At the bottom right, there is a button labeled "Назад".

Parameter	Value
Альфа	1
Лямбда	0.0001
Максимальна похибка	0.01
Прийнятна дельта	0.00001
Час завершення	10
Максимальна кількість ітерацій	990
Ітерацій	589
Похибка на навчальній вибірці	0.00999
Похибка на тестовій вибірці	0.00377

Рисунок 4.4 – Результати синтезу нейромоделі

4.5 Тестування програмного забезпечення для синтезу нейромоделей

Виконано тестування програмного забезпечення для синтезу нейромоделей.

Виявлено, що програма для синтезу нейромоделей функціонує правильно та злагоджено. Вона реалізує всі функціональні вимоги та успішно виконує свою основну задачу навчання нейромережових моделей.

Розроблене програмне забезпечення дозволяє забезпечувати автоматизацію процесів, пов'язаних з підтримкою процесу синтезу моделей на основі нейронних мереж.

4.6 Висновки за розділом 4

Описано програмне забезпечення для синтезу нейромоделей. Виконано тестування розробленого програмного забезпечення для синтезу нейромоделей. Результати тестування програмного забезпечення показали, що розроблена програма дозволяє забезпечити автоматизацію процесів, пов'язаних з підтримкою процесу синтезу моделей на основі нейронних мереж.

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес розробки програмного забезпечення для синтезу нейромоделей.

Визначено, що програмне забезпечення для синтезу нейромоделей є спеціалізованим класом програмних систем, призначених для створення, налаштування, навчання та використання штучних нейронних мереж з метою розв'язання задач аналізу даних, прогнозування та виявлення прихованих закономірностей. У таких програмних системах поєднуються засоби формування структури моделі, інструменти керування параметрами та алгоритми навчання, що забезпечує можливість переходу від вихідних даних до побудови інтелектуальної моделі без необхідності реалізації всіх математичних механізмів вручну.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато програмних засобів для синтезу нейромоделей. Проте деякі програми для синтезу нейромоделей потребують глибокого розуміння принципів функціонування нейронних мереж, методів оптимізації та специфіки обробки даних. У результаті виникає ситуація, коли заявлена універсальність такого програмного забезпечення обмежується практичними навичками розробника. Крім того, програмні засоби синтезу нейронних мереж часто генерують складні моделі, структура яких є непрозорою для користувача. Це ускладнює аналіз причин прийняття рішень нейромоделлю, знижує рівень довіри до результатів і створює труднощі при їх впровадженні в критично важливі системи. Додатковим недоліком є ризик отримання надмірно складних нейромоделей, оскільки автоматизовані алгоритми можуть формувати архітектури, які добре працюють на навчальних даних, але демонструють низьку здатність до узагальнення. Це потребує додаткового контролю та валідації результатів, що частково нівелює переваги автоматизації процесу синтезу нейромоделей. Не можна ігнорувати й

проблему обмеженої гнучкості окремих програмних рішень. Деякі системи орієнтовані на конкретні типи задач або архітектур, що ускладнює їх адаптацію до нестандартних умов. У таких випадках виникає необхідність модифікації або доповнення інструментів, що підвищує складність розроблення. Тому актуальною є розробка програмного забезпечення для синтезу нейромоделей.

Сформульовано функціональні вимоги до програмного забезпечення для синтезу нейромоделей.

Для реалізації програмного забезпечення для синтезу нейромоделей обрано мову програмування C#, яка надає набір функціональних можливостей та забезпечує баланс між продуктивністю, структурованістю коду, безпекою і зручністю користування. Це створює передумови для ефективної побудови, масштабування та підтримки складних систем, здатних автоматизувати процеси синтезу і навчання нейронних мереж у різноманітних прикладних сценаріях.

Для створення програмного забезпечення для синтезу нейромоделей обрано середовище розробки Visual Studio, що поєднує розробку, відлагодження, аналіз, тестування і підтримку програмних систем. Завдяки цьому середовищу Visual Studio підвищує продуктивність розробників, забезпечує стабільність і надійність програмного забезпечення, спрощує інтеграцію з бібліотеками і сервісами, а також полегшує візуалізацію і моніторинг складних процесів, таких як синтез нейромоделей.

Розроблено програмне забезпечення для синтезу нейромоделей.

Запропоновано структуру програмного забезпечення для синтезу нейромоделей. Визначено, що структура програмного забезпечення для синтезу нейромоделей базується на принципах модульності та абстракції, що дозволяє ефективно реалізувати процес синтезу нейромоделей. Вона забезпечує зручність розробки, можливість масштабування та інтеграцію додаткових компонентів без суттєвого впливу на вже існуючу функціональність. Як наслідок, така структура створює надійну основу для

побудови гнучких і продуктивних систем, здатних вирішувати складні задачі у сфері штучного інтелекту.

Функціонування програмного забезпечення для синтезу нейромоделей організовано як цілісний процес, у якому поєднуються автоматизовані етапи побудови нейромережі та інтерактивна взаємодія користувача з системою. Загальна логіка роботи передбачає початкове проходження всіх необхідних стадій у визначеній послідовності, після чого відкривається можливість гнучкого керування окремими параметрами з метою уточнення або оптимізації отриманих результатів. Такий підхід дозволяє спочатку сформулювати базову модель, а надалі виконувати її адаптацію без необхідності повторного проходження всього процесу з початку.

Архітектура програмного забезпечення для синтезу нейромоделей побудована за принципом чіткого розмежування рівнів відповідальності, що забезпечує відокремлення обчислювального ядра від засобів взаємодії з користувачем. У межах такого підходу центральне місце займає модуль, який реалізує логіку створення, налаштування та навчання нейромереж, тоді як інтерфейс користувача виконує функції керування процесом і відображення результатів. Це дозволяє досягти незалежності між внутрішніми алгоритмами та способом їх представлення, що значно спрощує як розвиток системи, так і її супровід.

Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для синтезу нейромоделей.

Виконано тестування розробленого програмного забезпечення для синтезу нейромоделей. Результати тестування програмного забезпечення показали, що розроблена програма дозволяє забезпечити автоматизацію процесів, пов'язаних з підтримкою процесу синтезу моделей на основі нейронних мереж.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Artificial Neural Networks and its Applications. URL: <https://www.geeksforgeeks.org/deep-learning/artificial-neural-networks-and-its-applications/> (date of access: 06.05.2026).
2. Review: The Best Frameworks for Machine Learning and Deep Learning. URL: <https://www.computerworld.com/article/1674307/review-the-best-frameworks-for-machine-learning-and-deep-learning.html> (date of access: 06.05.2026).
3. Tools and Frameworks for Machine Learning and Deep Learning: A Review. URL: https://www.researchgate.net/publication/385116208_Tools_and_frameworks_for_machine_learning_and_deep_learning_A_review (date of access: 06.05.2026).
4. Neural Networks: An Overview of Early Research, Current Frameworks and New Challenges. URL: <https://www.sciencedirect.com/science/article/abs/pii/S0925231216305550> (date of access: 06.05.2026).
5. Neural Networks: How AI Mimics the Human Brain. URL: <https://focalx.ai/ai/neural-networks/> (date of access: 06.05.2026).
6. Best Neural Network Software Today. URL: <https://www.eweek.com/artificial-intelligence/best-neural-network-software/> (date of access: 06.05.2026).
7. Top Artificial Neural Network Software. URL: <https://decidesoftware.com/top-artificial-neural-network-software/> (date of access: 06.05.2026).
8. Best Neural Network Software. URL: <https://cofes.com/neural-network-software/> (date of access: 06.05.2026).
9. Charting the Best Neural Network Software. URL: <https://thectoclub.com/tools/best-neural-network-software/> (date of access: 06.05.2026).

10. The Top Neural Network Software. URL: <https://www.softwaretestinghelp.com/best-neural-network-software/> (date of access: 06.05.2026).

11. Best Neural Network Software. URL: <https://clickup.com/blog/neural-network-software/> (date of access: 06.05.2026).

12. Neuroph. URL: <https://neuroph.sourceforge.net/> (date of access: 06.05.2026).

13. Caffe. URL: <https://caffe.berkeleyvision.org/> (date of access: 06.05.2026).

14. Neural Designer. URL: <https://www.neuraldesigner.com/> (date of access: 06.05.2026).

15. C# Tutorial. URL: <https://www.tutorialspoint.com/csharp/index.htm> (date of access: 06.05.2026).

16. C# Documentation. URL: <https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/> (date of access: 06.05.2026).

17. What is Visual Studio. URL: <https://learn.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide> (date of access: 06.05.2026).

18. Introduction to Visual Studio. URL: <https://www.geeksforgeeks.org/csharp/introduction-to-visual-studio/> (date of access: 06.05.2026).

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення може використовуватися для синтезу нейромоделей.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Програмне забезпечення для синтезу нейромоделей», затверджене наказом Національного університету «Запорізька політехніка» № 139 від 07 квітня 2026 р.

A.2 Призначення розробки

Програмний продукт призначений для побудови моделей на основі нейронних мереж.

A.3 Основні вимоги до програми, що розробляється

A.3.1 Вимоги до функціональних характеристик

Програмне забезпечення для синтезу нейромоделей забезпечує виконання таких функцій:

- підтримка можливості створення та налаштування багатошарових нейромереж із вибором кількості шарів та нейронів у кожному шарі;
- підтримка можливості вибору функцій активації для нейронів мережі;
- можливість завантаження вхідних даних для навчання нейромережі;
- можливість вибору характеристик процесу навчання нейромоделі із визначенням параметрів навчання, таких як швидкість навчання, контроль похибки, максимально допустима кількість ітерацій та час виконання;
- можливість оцінювання точності нейромоделі;

– підтримка інтерактивної роботи користувача з програмою, включаючи зміну параметрів моделі та вибір наборів даних для синтезу нейромоделі.

А.3.2 Вимоги до інтерфейсу програми

Інтерфейс програмного забезпечення для синтезу нейромоделей повинен бути зручним для користувачів. Повинна бути забезпечена можливість роботи в візуальному режимі.

А.3.3 Вимоги до надійності

Програмне забезпечення для синтезу нейромоделей повинно забезпечити надійне функціонування.

А.3.4 Умови експлуатації

Для експлуатації програмного забезпечення для синтезу нейромоделей необхідна наявність персонального комп'ютера.

А.3.5 Вимоги до складу та параметрів технічний засобів

Для роботи програмного забезпечення для синтезу нейромоделей потрібні такі технічні та програмні ресурси: центральний процесор з тактовою частотою не нижче 2.5 ГГц; обсяг оперативної пам'яті повинен становити не менше 8 ГБ, що забезпечує можливість роботи з вибірками середнього розміру та зберігання проміжних результатів обчислень.

Накопичувач даних повинен мати достатній обсяг для збереження програмного забезпечення, бібліотек, вхідних даних і результатів обчислень; рекомендовано використовувати твердотільний накопичувач обсягом не менше 256 ГБ, що забезпечує високу швидкість доступу до даних.

З точки зору програмного забезпечення необхідною є наявність операційної системи, сумісної з платформою .NET, зокрема Windows 10 або новіших. Обов'язковою умовою є встановлення середовища виконання .NET відповідної версії, яке забезпечує коректну роботу програмних модулів. Для розробки, налагодження та модифікації програмного продукту доцільно використовувати інтегроване середовище розробки Visual Studio, яке надає повний набір інструментів для роботи з мовою C# та графічним інтерфейсом.

A.3.6 Вимоги до маркування і пакування

Програма для синтезу нейромоделей може бути записана на будь-якому носії інформації.

На пакуванні повинна бути назва програми – «Програмне забезпечення для синтезу нейромоделей».

A.4 Вхідні дані до роботи

Вхідними даними до програмного забезпечення для синтезу нейромоделей є параметри нейромережі, включно з кількістю нейронів у кожному шарі, типами функцій активації та параметрами навчання, а також навчальні набори даних.

Вихідними даними програмного забезпечення для синтезу нейромоделей є результати синтезу нейромережі, зокрема значення ваг нейронів і зміщень, показники похибок і якість узгодження моделі з вихідними даними.

ДОДАТОК Б
Фрагмент тексту програми

```

namespace NrtwPrj
{
    public class Prgrm
    {
        [STAThread]
        public static void Main()
        {
            Application.EnableVisualStyles();
            Application.Run(new MnFrm());
        }
    }

    public class MnFrm : Form
    {
        private Button btnStrt;
        private Button btnTrn;
        private Button btnRsl;
        private TextBox txtLgs;
        private NrtwMdl nrtw;
        private DtsrcMdl dtsrc;

        public MnFrm()
        {
            this.Width = 800;
            this.Height = 600;

            btnStrt = new Button();
            btnStrt.Text = "Init";
            btnStrt.Location = new Point(20, 20);
            btnStrt.Click += BtnStrt_Click;

            btnTrn = new Button();
            btnTrn.Text = "Train";
            btnTrn.Location = new Point(120, 20);
            btnTrn.Click += BtnTrn_Click;

            btnRsl = new Button();
            btnRsl.Text = "Result";
            btnRsl.Location = new Point(220, 20);
            btnRsl.Click += BtnRsl_Click;

            txtLgs = new TextBox();
            txtLgs.Multiline = true;
            txtLgs.Width = 740;
            txtLgs.Height = 480;
            txtLgs.Location = new Point(20, 70);
        }
    }
}

```

```

        this.Controls.Add(btnStrt);
        this.Controls.Add(btnTrn);
        this.Controls.Add(btnRsl);
        this.Controls.Add(txtLgs);

        nrtw = new NrtwMdl();
        dtsrc = new DtsrcMdl();
    }

    private void BtnStrt_Click(object sender, EventArgs e)
    {
        dtsrc.gnrDts();
        nrtw.crtNrtw(3, 5, 1);
        nrtw.iniPrm();
        txtLgs.AppendText("Init done\r\n");
    }

    private void BtnTrn_Click(object sender, EventArgs e)
    {
        var dts = dtsrc.gtvDts();
        nrtw.trnNrtw(dts, 100);
        txtLgs.AppendText("Train done\r\n");
    }

    private void BtnRsl_Click(object sender, EventArgs e)
    {
        var dts = dtsrc.gtvDts();
        var rsl = nrtw.clcRsl(dts);
        foreach (var v in rsl)
        {
            txtLgs.AppendText(v.ToString("F4") + "\r\n");
        }
    }
}

public class DtsrcMdl
{
    private List<double[]> dts;

    public void gnrDts()
    {
        dts = new List<double[]>();
        Random rnd = new Random();

        for (int i = 0; i < 100; i++)
        {
            double[] row = new double[4];
            for (int j = 0; j < 3; j++)

```

```

        {
            row[j] = rnd.NextDouble();
        }
        row[3] = row[0] + row[1] * row[2];
        dts.Add(row);
    }
}

public List<double[]> gtvDts()
{
    return dts;
}
}

public class NrtwMdl
{
    private List<LyrMdl> lyrs;
    private Random rnd;

    public NrtwMdl()
    {
        rnd = new Random();
        lyrs = new List<LyrMdl>();
    }

    public void crtNrtw(int inp, int hdn, int otp)
    {
        lyrs.Clear();

        LyrMdl lyrInp = new LyrMdl(inp, 0);
        LyrMdl lyrHdn = new LyrMdl(hdn, inp);
        LyrMdl lyrOtp = new LyrMdl(otp, hdn);

        lyrs.Add(lyrInp);
        lyrs.Add(lyrHdn);
        lyrs.Add(lyrOtp);
    }

    public void iniPrm()
    {
        foreach (var lyr in lyrs)
        {
            lyr.iniWgt(rnd);
        }
    }

    public double[] fwdPrs(double[] inp)
    {

```

```

        double[] cur = inp;

        for (int i = 1; i < lyrs.Count; i++)
        {
            cur = lyrs[i].clc(cur);
        }

        return cur;
    }

    public void trnNrtw(List<double[]> dts, int epc)
    {
        for (int e = 0; e < epc; e++)
        {
            foreach (var row in dts)
            {
                double[] inp = row.Take(3).ToArray();
                double trg = row[3];

                double[] outp = fwdPrs(inp);
                double err = trg - outp[0];

                bckPrs(err);
            }
        }
    }

    private void bckPrs(double err)
    {
        double lrn = 0.01;

        for (int i = lyrs.Count - 1; i >= 1; i--)
        {
            lyrs[i].upd(err, lrn);
        }
    }

    public List<double> clcRsl(List<double[]> dts)
    {
        List<double> rsl = new List<double>();

        foreach (var row in dts)
        {
            double[] inp = row.Take(3).ToArray();
            var outp = fwdPrs(inp);
            rsl.Add(outp[0]);
        }
    }

```

```

        return rsl;
    }
}

public class LyrMdl
{
    private int sz;
    private int prv;
    private List<NrnMdl> nrns;

    public LyrMdl(int sz, int prv)
    {
        this.sz = sz;
        this.prv = prv;
        nrns = new List<NrnMdl>();

        for (int i = 0; i < sz; i++)
        {
            nrns.Add(new NrnMdl(prv));
        }
    }

    public void iniWgt(Random rnd)
    {
        foreach (var nrn in nrns)
        {
            nrn.iniWgt(rnd);
        }
    }

    public double[] clc(double[] inp)
    {
        double[] rsl = new double[sz];

        for (int i = 0; i < sz; i++)
        {
            rsl[i] = nrns[i].clc(inp);
        }

        return rsl;
    }

    public void upd(double err, double lrn)
    {
        foreach (var nrn in nrns)
        {
            nrn.upd(err, lrn);
        }
    }
}

```

```

    }
}

public class NrnMdl
{
    private double[] wgt;
    private double bst;

    public NrnMdl(int inp)
    {
        wgt = new double[inp];
        bst = 0;
    }

    public void iniWgt(Random rnd)
    {
        for (int i = 0; i < wgt.Length; i++)
        {
            wgt[i] = rnd.NextDouble() - 0.5;
        }
        bst = rnd.NextDouble() - 0.5;
    }

    public double clc(double[] inp)
    {
        double sum = 0;

        for (int i = 0; i < wgt.Length; i++)
        {
            sum += wgt[i] * inp[i];
        }

        sum += bst;
        return act(sum);
    }

    private double act(double x)
    {
        return Math.Tanh(x);
    }

    public void upd(double err, double lrn)
    {
        for (int i = 0; i < wgt.Length; i++)
        {
            double grd = err * (1 - wgt[i] * wgt[i]);
            wgt[i] += lrn * grd;
        }
    }
}

```

```

        bst += lrn * err;
    }
}

public void rstNrtw()
{
    if (lyrs == null || lyrs.Count == 0) return;

    Random rnd = new Random();
    foreach (var lyr in lyrs)
    {
        lyr.iniWgt(rnd);
    }
}

public double clcErr(List<double[]> dts)
{
    double ttl = 0;
    int cnt = 0;

    foreach (var row in dts)
    {
        double[] inp = row.Take(3).ToArray();
        double trg = row[3];

        double[] outp = fwdPrs(inp);
        double df = trg - outp[0];

        ttl += df * df;
        cnt++;
    }

    if (cnt == 0) return 0;
    return ttl / cnt;
}

public List<double[]> clcAll(List<double[]> dts)
{
    List<double[]> rsl = new List<double[]>();

    foreach (var row in dts)
    {
        double[] inp = row.Take(3).ToArray();
        double[] outp = fwdPrs(inp);

        double[] cmb = new double[inp.Length + outp.Length];
        Array.Copy(inp, cmb, inp.Length);
    }
}

```

```

        Array.Copy(outp, 0, cmb, inp.Length, outp.Length);

        rsl.Add(cmb);
    }

    return rsl;
}

public void nrmDts(List<double[]> dts)
{
    if (dts == null || dts.Count == 0) return;

    int ln = dts[0].Length;
    double[] mn = new double[ln];
    double[] mx = new double[ln];

    for (int i = 0; i < ln; i++)
    {
        mn[i] = double.MaxValue;
        mx[i] = double.MinValue;
    }

    foreach (var row in dts)
    {
        for (int i = 0; i < ln; i++)
        {
            if (row[i] < mn[i]) mn[i] = row[i];
            if (row[i] > mx[i]) mx[i] = row[i];
        }
    }

    foreach (var row in dts)
    {
        for (int i = 0; i < ln; i++)
        {
            double rng = mx[i] - mn[i];
            if (rng != 0)
            {
                row[i] = (row[i] - mn[i]) / rng;
            }
        }
    }
}

public void shflDts(List<double[]> dts)
{
    if (dts == null || dts.Count < 2) return;

    Random rnd = new Random();

```

```

    for (int i = dts.Count - 1; i > 0; i--)
    {
        int j = rnd.Next(i + 1);
        var tmp = dts[i];
        dts[i] = dts[j];
        dts[j] = tmp;
    }
}

public double clcAcc(List<double[]> dts)
{
    int cr = 0;
    int ttl = 0;

    foreach (var row in dts)
    {
        double[] inp = row.Take(3).ToArray();
        double trg = row[3];

        double[] outp = fwdPrs(inp);

        double df = Math.Abs(trg - outp[0]);
        if (df < 0.1) cr++;

        ttl++;
    }

    if (ttl == 0) return 0;
    return (double)cr / ttl;
}

public List<double[]> spltdts(List<double[]> dts, double prt)
{
    int cut = (int)(dts.Count * prt);
    List<double[]> sub = new List<double[]>();

    for (int i = 0; i < cut; i++)
    {
        sub.Add(dts[i]);
    }

    return sub;
}

public void savWgt(string pth)
{
    List<string> lns = new List<string>();

```

```

        foreach (var lyr in lyrs)
        {
            var lst = lyr.gtvWgt();
            foreach (var row in lst)
            {
                lns.Add(string.Join(",", row));
            }
        }

        System.IO.File.WriteAllLines(pth, lns);
    }

    public void lddWgt(string pth)
    {
        if (!System.IO.File.Exists(pth)) return;

        var lns = System.IO.File.ReadAllLines(pth);
        int idx = 0;

        foreach (var lyr in lyrs)
        {
            idx = lyr.stWgt(lns, idx);
        }
    }

    public List<double> clcErrHst(List<double[]> dts, int epc)
    {
        List<double> hst = new List<double>();

        for (int i = 0; i < epc; i++)
        {
            trnNrtw(dts, 1);
            double err = clcErr(dts);
            hst.Add(err);
        }

        return hst;
    }

    public void updLrnRt(double nwLrn)
    {
        foreach (var lyr in lyrs)
        {
            lyr.stLrn(nwLrn);
        }
    }

    public double clcVrnc(List<double[]> dts)
    {

```

```
List<double> vals = new List<double>();

foreach (var row in dts)
{
    double[] inp = row.Take(3).ToArray();
    var outp = fwdPrs(inp);
    vals.Add(outp[0]);
}

double avg = vals.Average();
double sm = 0;

foreach (var v in vals)
{
    double df = v - avg;
    sm += df * df;
}

return sm / vals.Count;
}
```

ДОДАТОК В
Слайди презентації

**Міністерство освіти і науки України
Національний університет «Запорізька політехніка»
Кафедра програмних засобів**

Програмне забезпечення для синтезу нейромоделей

Виконав: Артем ПОШУКАЙ

ст. групи КНТ-122

Керівник: Сергій Олександрович СУББОТІН

д.т.н., професор, завідувач кафедри ПЗ

Рисунок В.1 – Слайд 1

Об'єкт, предмет та мета роботи

Об'єкт дослідження – процес розробки програмного забезпечення для синтезу нейромоделей.

Предмет дослідження – програмні засоби для синтезу нейромоделей.

Метою роботи є розробка програмного забезпечення для синтезу нейромоделей, що забезпечує збільшення точності та надійності розпізнавальних моделей.

Рисунок В.2 – Слайд 2

Завдання роботи

- Відповідно до мети були поставлені такі завдання:
- виконати аналіз предметної області та програмних засобів для синтезу нейромоделей;
 - здійснити проєктування програмного забезпечення для синтезу нейромоделей;
 - створити програмне забезпечення для синтезу нейромоделей;
 - виконати тестування розробленого програмного забезпечення для синтезу нейромоделей.

3

Рисунок В.3 – Слайд 3

Порівняння програмного забезпечення для синтезу нейромоделей

Критерій порівняння	Neuroph	Caffe	Neural Designer
простота використання	+	+—	+
необхідність програмування	+—	+	—
гнучкість налаштування	+—	+	+—
підтримка графічного інтерфейсу	+	—	+
інтеграція з іншими інструментами	+—	+	+—
відкритість коду	+	+	—

4

Рисунок В.4 – Слайд 4

Вимоги до програмного забезпечення

При розробці програмного забезпечення для синтезу нейромоделей необхідно забезпечити такі функціональні вимоги:

- підтримка можливості створення та налаштування багат шарових нейромереж із вибором кількості шарів та нейронів у кожному шарі;
- підтримка можливості вибору функцій активації для нейронів мережі;
- можливість завантаження вхідних даних для навчання нейромережі;
- можливість вибору характеристик процесу навчання нейромоделі із визначенням параметрів навчання, таких як швидкість навчання, контроль похибки, максимально допустима кількість ітерацій та час виконання;
- можливість оцінювання точності нейромоделі;
- підтримка інтерактивної роботи користувача з програмою, включаючи зміну параметрів моделі та вибір наборів даних для синтезу нейромоделі.

5

Рисунок В.5 – Слайд 5

Порівняння мов програмування

Критерій порівняння мов програмування	Мова програмування		
	C#	Python	C++
Продуктивність виконання	+	+–	+
Бібліотеки для нейромереж	+	+	+–
Безпека коду та управління пам'яттю	+	+–	+–
Візуалізація та GUI	+	+–	–
Простота використання та документація	+	+	+–

6

Рисунок В.6 – Слайд 6

Порівняння середовищ розробки

Критерій порівняння середовищ розробки	Середовища розробки		
	Visual Studio	Rider	MonoDevelop
Інтеграція з бібліотеками для машинного навчання	+	+–	–
Потужність налагодження та профілювання	+	+–	+–
Графічний інтерфейс і візуалізація	+	+–	–
Інструменти для командної роботи	+	+	+–
Автоматизоване тестування та аналіз коду	+	+–	–

7

Рисунок В.7 – Слайд 7

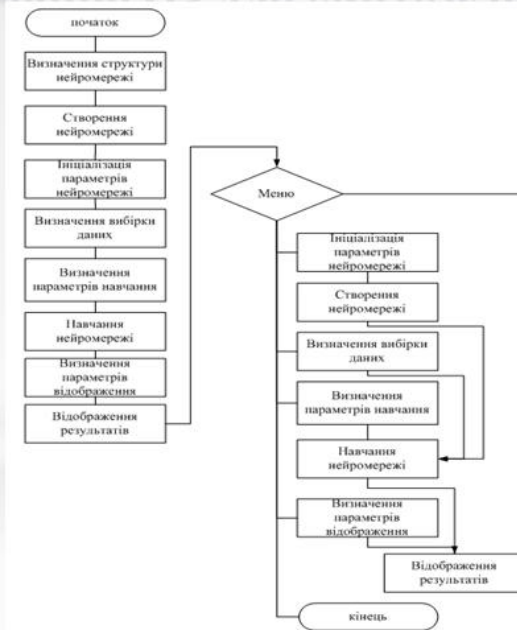
Структура ПЗ для синтезу нейромоделей



8

Рисунок В.8 – Слайд 8

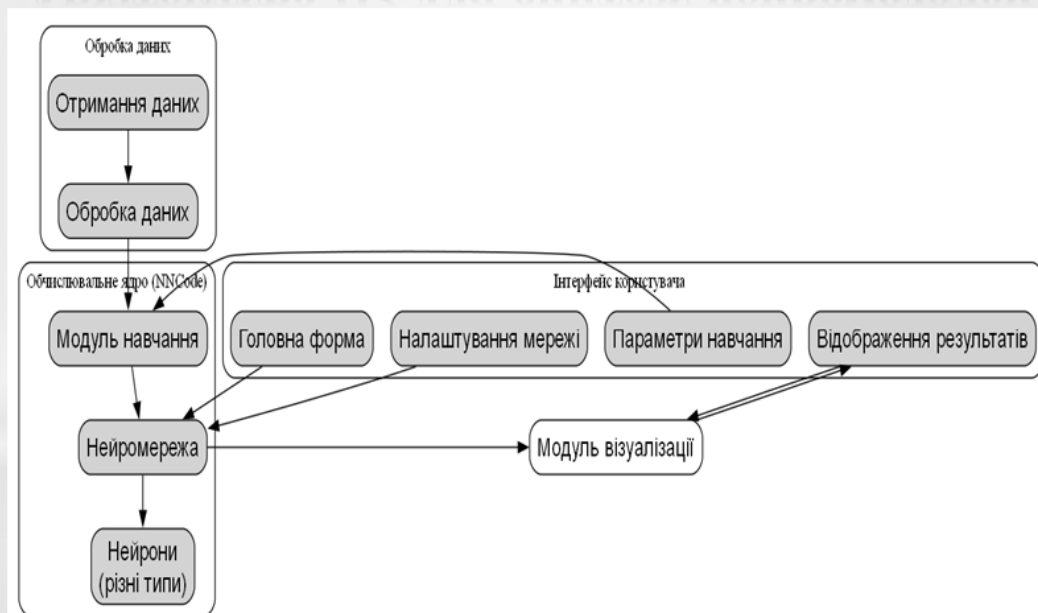
Функціонування ПЗ для синтезу нейромODEлей



9

Рисунок В.9 – Слайд 9

Архітектура ПЗ для синтезу нейромODEлей



10

Рисунок В.10 – Слайд 10

Робота з програмою. Введення вхідних даних

Програмна реалізація багатошарових нейромереж

Функція активації

Архітектура мережі

3;2;2;1

Створити

Обрати дані

Назад

11

Рисунок В.11 – Слайд 11

Робота з програмою. Встановлення параметрів та результати навчання нейромоделей

Програмна реалізація багатошарових нейромереж

Альфа

1

Лямбда

0.0001

Максимальна похибка

0.01

Прийнятна дельта

0.00001

Час завершення

10

Максимальна кількість ітерацій

990

Відображення

Навчання

Ітерацій

589

Похибка на навчальній вибірці

0.00999

Похибка на тестовій вибірці

0.00377

Назад

12

Рисунок В.12 – Слайд 12

Висновки

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес розробки програмного забезпечення для синтезу нейромоделей. Розроблено програмне забезпечення для синтезу нейромоделей. Для створення програмного забезпечення для синтезу нейромоделей обрано мову C# та середовище розробки Visual Studio.

Запропоновано структуру програмного забезпечення для синтезу нейромоделей. Визначено, що структура програми базується на принципах модульності та абстракції, що дозволяє ефективно реалізувати процес синтезу нейромоделей.

В результаті виконання завдання було отримано систему, що відповідає всім вимогам представленим в технічному завданні, зручна для використання та виконує всі необхідні функції.

Усі завдання випускної дипломної роботи бакалавра повністю виконано.

Рисунок В.13 – Слайд 13