

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 проекту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ
НЕЙРОМЕРЕЖЕВИХ ЗАСОБІВ ДІАГНОСТУВАННЯ
ХВОРОБИ АЛЬЦГЕЙМЕРА ЗА МРТ-ЗНІМКАМИ
RESEARCH AND SOFTWARE IMPLEMENTATION OF
NEURAL NETWORK TOOLS FOR MRI-BASED
DIAGNOSING OF ALZHEIMER'S DISEASE

Виконав(ла): студент(ка) 4 курсу, групи КНТ-131

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ГОРПІНІЧ І.О.

(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»
 (повне найменування закладу вищої освіти)

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
 “ ” 2025 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ГОРПІНІЧ Ірини Олексіївни
 (ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація нейромережових засобів діагностування хвороби Альцгеймера за МРТ-знімками. Research and Software Implementation of Neural Network Tools for MRI-Based Diagnosing of Alzheimer's Disease

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 “28” квітня 2025 року № 209

2. Строк подання студентом проєкту (роботи) 02 червня 2025 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	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	СУББОТІН С.О., зав.кафедри		
Нормоконтроль	КАЛІНІНА М.В., асистент		

7. Дата видачі завдання «14» квітня 2025 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 архітектури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділ 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)

(підпис) Ірина ГОРПІНІЧ
(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Сергій СУББОТІН
(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 64 с., 4 табл., 22 рис., 3 дод., 35 джерел.

PYTHON, PYCHARM, НЕЙРОМЕРЕЖА, МОВА ПРОГРАМУВАННЯ, ДІАГНОСТУВАННЯ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ДІАГНОСТУВАННЯ ХВОРОБИ АЛЬЦГЕЙМЕРА.

Об'єкт дослідження – процес обчислювального діагностування деменції, керований даними на основі магнітно-резонансної томографії (МРТ) головного мозку пацієнтів.

Предмет дослідження – методи та засоби автоматизованого діагностування стадій хвороби Альцгеймера за МРТ-знімками на основі глибоких нейронних мереж.

Мета роботи – розробка та оцінка ефективності нейромережевої моделі для ранньої діагностики хвороби Альцгеймера з метою підвищення точності класифікації до 90% на тестовій вибірці.

Матеріали, методи та технічні засоби: мова програмування Python, середовище розробки PyCharm.

Результати – створено модель на основі ResNet50 із механізмом самопосиленої уваги, яка продемонструвала високу точність класифікації стадій деменції; реалізовано механізми динамічного балансування класів та навчання на важких зразках.

Практична цінність – розроблена модель може бути інтегрована у медичні діагностичні системи для автоматичного аналізу МРТ-знімків, допомагаючи лікарям у виявленні та моніторингу прогресування хвороби Альцгеймера.

Висновки – використання попередньо натренованих моделей із механізмами самопосиленої уваги та адаптивного балансування класів

дозволяє підвищити точність діагностики деменції; подальші дослідження можуть включати розширення вибірки та застосування інших архітектур нейромереж.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 64 pages, 4 tables, 22 figures, 3 appendixes, 35 sources.

PYTHON, PYCHARM, NEURAL NETWORK, PROGRAMMING LANGUAGE, DIAGNOSIS, SOFTWARE, DIAGNOSIS OF ALZHEIMER'S DISEASE.

The object of research is the process of computational diagnosis of dementia, driven by data based on magnetic resonance imaging (MRI) of the brain patients.

The subject of the research is methods and tools for automated diagnosis of the stages of Alzheimer's disease from MRI images based on deep neural networks.

The purpose of the work is the develop and evaluate the effectiveness of a neural network model for early diagnosis of Alzheimer's disease in order to increase the classification accuracy to 90% on the test sample.

Materials, methods and technical means: Python programming language, PyCharm development environment.

Results - a model based on ResNet50 with self-reinforced attention mechanism was created, which demonstrated high accuracy in classifying dementia stages; mechanisms for dynamic class balancing and learning on difficult samples (Hard Simple Mining) were implemented.

Practical value – the developed model can be integrated into medical diagnostic systems for automatic analysis of MRI images, helping doctors detect and monitor the progression of Alzheimer's disease.

Conclusions – the use of pre-trained models with mechanisms of self-reinforced attention and adaptive class balancing allows to increase the accuracy of

dementia diagnosis; further studies may include expanding the sample and applying other neural network architectures.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	10
Вступ.....	11
1 Аналіз предметної області	12
1.1 Нейромережа для діагностування хвороби Альцгеймера за МРТ-знімками	12
1.2 Аналіз нейромереж для діагностування хвороби Альцгеймера	14
1.3 Висновки за розділом 1	18
2 Матеріали і методи	19
2.1 Вибір мови програмування.....	19
2.2 Вибір середовища розробки для нейромережі діагностування Альцгеймера.....	20
2.3 Висновки за розділом 2	22
3 Опис програми	23
3.1 Алгоритм функціонування розробленої програми для діагностування хвороби Альцгеймера за МРТ-знімками	23
3.2 Опис обраної вибірки для тренування нейронної мережі	25
3.3 Опис використаних алгоритмів для покращення точності побудованої моделі.....	27
3.4 Опис використаного методу для візуалізації прогресу навчання нейромережі	29
3.5 Висновки за розділом 3	31
4 Експлуатація, тестування та експериментальне дослідження програми	32
4.1 Призначення й умови застосування програми	32
4.2 Характеристики нейронної мережі для виявлення хвороби Альцгеймера.....	33
4.3 Інструкція по експлуатації програми	34
4.4 Тестування нейронної мережі для виявлення хвороби Альцгеймера ...	35
4.5 Висновки за розділом 4	36

Висновки.....	37
Перелік джерел посилання	39
Додаток А Технічне завдання.....	45
Додаток Б Тексти програм.....	49
Додаток В Слайди презентації	58

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

ADNI – Alzheimer’s Disease Neuroimaging Initiative;

CNN – Convolutional Neural Networks;

GAN – Generative Adversarial Networks;

LSTM – Long Short-Term Memory;

MCI – Mild Cognitive Impairment;

OASIS – Open Access Series of Imaging Studies;

RBF – Radial Basis Function Networks;

RNN – Recurrent Neural Networks;

SBA – Self-Boosted Attention.

ВСТУП

Хвороба Альцгеймера є однією з найбільш поширених і небезпечних нейродегенеративних хвороб, що поступово призводить до втрати пам'яті, когнітивних здібностей та інших порушень мозкової діяльності. За даними Всесвітньої організації охорони здоров'я, кількість випадків деменції, спричиненої цією хворобою, неухильно зростає, що створює значне навантаження на систему охорони здоров'я. Одним із ключових аспектів боротьби з хворобою Альцгеймера є її рання діагностика, що дозволяє своєчасно розпочати терапію та уповільнити прогресування патологічних змін у мозку [1].

Для підвищення ефективності діагностики необхідно розробити методи автоматизованого аналізу медичних зображень, зокрема магнітно-резонансної томографії (МРТ), які зменшать суб'єктивність оцінки та пришвидшать процес діагностування. Особливу увагу слід приділити застосуванню глибоких нейронних мереж, зокрема згорткових архітектур [2]-[3], здатних виявляти патологічні зміни на МРТ-знімках із високою точністю. Важливим є дослідження можливості покращення ефективності таких моделей за рахунок оптимізації архітектури, використання механізмів уваги [4] та методів балансування даних [5].

Також необхідно оцінити, як використання методів збільшення даних (data augmentation) та навчання з передачею знань (transfer learning) впливає на точність класифікації [3]. Важливим аспектом є перевірка ефективності моделі на реальних вибірках пацієнтів із різними стадіями деменції, а також потенційна можливість її інтеграції в медичні інформаційні системи для підтримки лікарських рішень.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

У цьому розділі досліджено основні підходи до використання нейронних мереж для діагностування хвороби Альцгеймера за допомогою медичних зображень та інших біомедичних даних. Проаналізовано архітектури нейромереж, що застосовуються для класифікації МРТ-знімків, оцінки функціональних змін у мозку та створення синтетичних даних. Визначаються їх переваги, обмеження та доцільність використання для розв'язання поставленої задачі. Це дослідження є основою вибору оптимального методу, який забезпечить високу точність і надійність діагностики.

1.1 Нейромережа для діагностування хвороби Альцгеймера за МРТ-знімками

Хвороба Альцгеймера є однією з найпоширеніших форм деменції, що вражає переважно людей похилого віку [1]. Рання діагностика цієї хвороби є критично важливою для ефективного лікування та уповільнення її прогресування. Одним із сучасних підходів до діагностування є використання методів глибокого навчання та нейронних мереж для аналізу магнітно-резонансних томографічних (МРТ) знімків мозку.

Глибокі нейронні мережі, зокрема згорткові нейромережі (Convolutional Neural Networks, CNN [2]-[3]), добре підходять для обробки медичних зображень завдяки здатності ефективно виявляти патерни та аномалії, які можуть бути непомітними для лікаря. У випадку діагностики хвороби Альцгеймера використовується архітектура CNN для автоматичного аналізу МРТ-знімків, що дозволяє класифікувати пацієнтів на різні групи (наприклад, здорові пацієнти з mild cognitive impairment (MCI) та хворі на Альцгеймер).

Для побудови моделі зазвичай використовується глибока згорткова нейромережа, наприклад, ResNet50 [6], VGG16 [7] або InceptionV3 [8]. Ці моделі вже навчені на великих наборах даних (наприклад, ImageNet [3]) і можуть бути адаптовані до задач класифікації МРТ-знімків за допомогою техніки transfer learning [3].

Типова архітектура такої нейромережі включає:

- вхідний шар: отримує МРТ-зображення у вигляді тривимірного тензора;
- згорткові шари: виконують фільтрацію та виділення ознак, таких як зміни в структурі мозку;
- шари підвибірки (Pooling Layers [3]): зменшують розмірність вхідних даних, зберігаючи основні характеристики;
- повнозв'язні шари (Fully Connected Layers [3]): обробляють отримані ознаки для прийняття рішень;
- вихідний шар: зазвичай використовує функцію Softmax для класифікації знімків на категорії «здоровий», «МСІ» або «Альцгеймер».

Перед подачею МРТ-знімків у нейромережу необхідно виконати їх попередню обробку, а саме:

- нормалізація та масштабування: вирівнювання значень інтенсивності пікселів;
- вирівнювання зображень: приведення всіх знімків до єдиного просторового стандарту;
- сегментація: виділення сірої речовини, білої речовини та цереброспінальної рідини для більш точного аналізу;
- аугментація: створення додаткових варіантів знімків (зміна кута повороту, яскравості тощо) для покращення генералізації моделі.

Навчання моделі відбувається на основі великої кількості МРТ-знімків, наприклад, з відкритих наборів даних OASIS [9] (Open Access Series of Imaging Studies) або ADNI [10] (Alzheimer's Disease Neuroimaging Initiative). Для підвищення якості класифікації використовується підхід

transfer learning [3], який дозволяє адаптувати вже попередньо навчені моделі для цієї задачі.

Метрики оцінки ефективності нейромережі включають [11]:

- Accuracy(точність) – частка правильних передбачень;
- Precision (Прецизійність) – частка коректних передбачень класу «Альцгеймер»;
- Recall (повнота) – частка правильно розпізнаних випадків хвороби;
- F1-score – гармонійне середнє між precision і recall.

Розроблена нейромережа може бути впроваджена в систему підтримки прийняття рішень у лікарнях, де вона аналізуватиме МРТ-знімки та допомагатиме лікарям у постановці діагнозу. Також можливе використання моделі у віддалених медичних системах для швидкого попереднього скринінгу пацієнтів.

Використання нейромереж у діагностуванні хвороби Альцгеймера за МРТ-знімками є перспективним напрямом досліджень, який може значно покращити точність і швидкість діагностики. Поєднання передових методів глибокого навчання, правильного підбору архітектури нейромережі та якісної попередньої обробки даних дозволяє створити ефективні та надійні системи для раннього виявлення цієї нейродегенеративної хвороби.

1.2 Аналіз нейромереж для діагностування хвороби Альцгеймера

Хвороба Альцгеймера є серйозною проблемою сучасної медицини, і її рання діагностика є критично важливою для ефективного лікування. В останні роки широке застосування отримали нейромережі, які дозволяють аналізувати медичні зображення, електрофізіологічні сигнали та інші біомедичні дані. У цьому розділі розглянемо основні архітектури нейромереж, що використовуються для діагностики хвороби Альцгеймера, та їхні особливості.

Згорткові нейромережі (Convolutional Neural Networks, CNN [2]-[3]) є основним інструментом для аналізу зображень, включаючи магнітно-резонансні томографічні (МРТ) та позитронно-емісійні томографічні (ПЕТ) знімки мозку. Вони добре підходять для виділення патернів, що можуть вказувати на дегенеративні зміни в мозку.

Моделі, які найчастіше використовуються:

- ResNet50: використовується для аналізу МРТ-знімків, забезпечуючи глибоку обробку характеристик знімків завдяки залишковим зв'язкам (skip connections);
- VGG16/VGG19 [12]: дозволяють отримувати високоточні результати, хоча вимагають значних обчислювальних ресурсів;
- InceptionV3: ефективні для аналізу багаторівневих особливостей у знімках мозку.

Перевагами моделей CNN є:

- висока точність при аналізі МРТ/ПЕТ знімків;
- хороша генералізація при використанні трансферного навчання.

Недоліками CNN моделей є:

- потребують великої кількості даних для навчання;
- високі вимоги до обчислювальних ресурсів.

Рекурентні нейромережі (Recurrent Neural Networks, RNN [13]-[14]) та їх удосконалення, такі як довготривала короткочасна пам'ять (Long Short-Term Memory, LSTM [15]), застосовуються для аналізу часових рядів і динамічних змін у мозковій активності, наприклад електроенцефалограми (ЕЕГ) та функціональної МРТ (fMRI).

Моделі, які найчастіше використовуються:

- Bi-LSTM [16]: двонаправлена модель, що аналізує зміни у функціональній активності мозку;
- GRU(Gated Recurrent Unit) [17]: покращена версія LSTM, що має менше параметрів та швидше навчається.

Перевагами моделей RNN\LSTM є:

- добре підходять для аналізу часових залежностей у нейрофізіологічних даних;
- ефективні для прогнозування прогресування хвороби.

Недоліками моделей RNN\LSTM є:

- складні в навчанні, особливо на довгих часових рядах;
- можуть потребувати великої кількості етапів передобробки.

Радіально-базисні функції (Radial Basis Function Networks, RBF [18]) використовуються для класифікації медичних зображень та біомаркерів на основі нелінійного поділу простору ознак.

Підходами, які найчастіше використовуються для даних моделей є:

- гібридні моделі RBF-CNN [19]: використовуються для аналізу МРТ-знімків, де RBF-мережі застосовуються на вихідних шарах CNN для уточнення класифікації.

Перевагами нейромереж RBF є:

- висока точність класифікації при невеликій вибірці;
- добре працюють у комбінації з іншими методами.

Недоліками нейромереж RBF є:

- чутливі до вибору центрів базисних функцій;
- вимагають складної оптимізації.

Автоенкодери (Autoencoders) та генеративно-змагальні нейромережі (Generative Adversarial Networks, GAN [20]) використовуються для виявлення аномалій у МРТ-знімках та створення синтетичних зразків для покращення навчання моделей.

Підходи, які найчастіше використовуються для моделей даного типу є:

- Variational Autoencoder (VAE [21]): дозволяє виявляти приховані ознаки хвороби в МРТ;
- CycleGAN [22]: використовується для перетворення зображень між різними медичними модальностями (наприклад, з ПЕТ у МРТ).

Перевагами моделей Autoencoders\GAN є:

- можуть створювати синтетичні дані для покращення навчання;
- ефективні для аномального виявлення.

Недоліками моделей Autoencoders\GAN є:

- потребують великої обчислювальної потужності;
- інколи складно інтерпретувати отримані результати.

Детальне порівняння характеристик зазначених нейромереж наведено в табл. 1.1.

Таблиця 1.1 – Порівняльна таблиця нейромереж

Нейромережа	Аналіз зображень	Аналіз часових рядів	Синтетичні дані	Вимоги до ресурсів
CNN	+	–	–	Високі
RNN/LSTM	–	+	–	Високі
RBF	+	–	–	Середні
Autoencoders/GAN	+	–	+	Дуже високі

Різні нейромережеві архітектури можуть бути використані для діагностики хвороби Альцгеймера, залежно від доступних даних та мети дослідження:

- CNN є найкращими для аналізу медичних зображень;
- RNN/LSTM підходять для аналізу часових даних, таких як ЕЕГ та fMRI;
- RBF добре працюють у комбінації з іншими методами класифікації;
- автоенкодер/GAN можуть покращити виявлення аномалій та збагатити навчальну вибірку.

Вибір конкретної архітектури залежить від доступних ресурсів, обсягу вибірки та необхідної точності класифікації.

1.3 Висновки за розділом 1

У першому розділі було проведено аналіз предметної області, зокрема, досліджено різні архітектури нейронних мереж, які застосовуються для діагностування хвороби Альцгеймера. Описано особливості CNN, RNN, RBF, автоенкодерів і GAN, їхню ефективність у роботі з медичними зображеннями, часовими рядами та синтетичними даними. Наведено порівняльний аналіз цих методів, що дозволяє обґрунтовано підійти до вибору найкращої архітектури для подальшого моделювання.

2 МАТЕРІАЛИ І МЕТОДИ

У даному розділі розглядається вибір мови програмування та середовища розробки для створення нейромережі, призначеної для діагностики хвороби Альцгеймера. Вибір мови програмування здійснюється на основі таких критеріїв, як простота синтаксису, продуктивність, наявність бібліотек для машинного навчання, підтримка багатопотоковості, кросплатформність та популярність у наукових дослідженнях. Аналогічно, для вибору середовища розробки враховуються підтримка обраної мови, інтеграція з бібліотеками штучного інтелекту, зручність налагодження, робота з віртуальними середовищами та підтримка інструментів для машинного навчання. На основі порівняльного аналізу буде визначено найоптимальніше рішення для реалізації даного проєкту.

2.1 Вибір мови програмування

Мову програмування для розробки нейромережі було обрано на основі кількох ключових критеріїв, серед яких простота синтаксису, продуктивність, наявність бібліотек для штучного інтелекту, гнучкість, підтримка багатопотоковості, кросплатформність та поширеність у наукових дослідженнях. Python [23] був обраний, оскільки він поєднує зручність використання, читабельність коду, широкий набір бібліотек для машинного навчання (TensorFlow [24], PyTorch [25], Keras [26], scikit-learn [27]), а також активну підтримку спільноти. Java, C++ та C# розглядалися як альтернативи, проте вони мають певні обмеження. Java демонструє високу продуктивність, проте її синтаксис є складнішим, а кількість бібліотек для глибокого навчання – меншою. C++ відзначається максимальною продуктивністю, але складний у розробці, що ускладнює швидке прототипування моделей. C# використовується переважно у корпоративному середовищі та розробці ігор,

а його бібліотеки для штучного інтелекту значно поступаються Python. Всі ці аспекти були проаналізовані, а результати порівняння наведені в табл. 2.1.

Таблиця 2.1 – Порівняння мов програмування

Критерій	Python	Java	C++	C#
Простота синтаксису	++	–	–	–
Продуктивність	–	+	++	+
Бібліотеки для AI	++	+	+	–
Гнучкість	++	+	–	–
Підтримка багатопотоковості	+	++	++	+
Кросплатформність	++	++	++	–
Використання у науці	++	+	+	–

Таким чином, Python виявився найбільш збалансованим вибором для розробки нейромережі, забезпечуючи оптимальне поєднання простоти, продуктивності та доступності бібліотек. Завдяки своїй популярності у сфері науки та штучного інтелекту, а також широкій екосистемі інструментів, він дозволяє ефективно реалізовувати та тестувати моделі машинного навчання, що робить його найкращим варіантом для аналізу медичних даних та автоматизованої діагностики хвороби Альцгеймера.

2.2 Вибір середовища розробки для нейромережі діагностування Альцгеймера

Середовище розробки було обрано на основі таких критеріїв, як підтримка мови програмування Python [23], інтеграція з бібліотеками машинного навчання, зручність налагодження, підтримка Jupyter Notebook [28] та можливості керування залежностями. Враховуючи ці аспекти, було проведено порівняння середовищ PyCharm [29], Eclipse, Xcode та Apache NetBeans (табл. 2.2). PyCharm забезпечує вбудовану підтримку Python, зручні

інструменти для роботи з бібліотеками TensorFlow [24], PyTorch [25], Keras [26], а також розширені можливості налагодження та інтеграцію з Jupyter Notebook. Eclipse, хоч і є потужним середовищем для Java, має обмежену підтримку Python через додаткові плагіни. Xcode, основне середовище для macOS та iOS, не орієнтоване на машинне навчання та не підтримує Python на належному рівні. Apache NetBeans добре підходить для Java та C++, але не має необхідних інструментів для ефективної розробки нейромережі.

Результати порівняння середовищ розробки наведені в табл. 2.2. PyCharm отримало найвищі оцінки за всіма критеріями, тоді як альтернативні середовища мають суттєві обмеження щодо підтримки Python та інтеграції з бібліотеками машинного навчання.

Таблиця 2.2 – Порівняння середовищ розробки

Середовище	Підтримка Python	Інтеграція з ML-бібліотеками	Зручність налагодження	Підтримка Jupyter Notebook	Управління залежностями
PyCharm	++	++	++	+	++
Eclipse	+—	—	+	—	+
Xcode	—	—	+	—	—
Apache NetBeans	—	—	+—	—	+—

Таким чином, PyCharm є найбільш підходящим для розробки реалізації нейромережі для діагностики хвороби Альцгеймера. Воно забезпечує всі необхідні інструменти для написання, тестування та налагодження коду, підтримує роботу з науковими бібліотеками та Jupyter Notebook, що є важливим для аналізу медичних зображень та експериментів із конфігураціями нейромереж.

2.3 Висновки за розділом 2

У другому розділі було проведено аналіз вибору мови програмування та середовища розробки для створення нейромережі, що діагностує хворобу Альцгеймера. Розглянуто основні мови програмування, зокрема Python, Java, C++ та C#, та оцінено їх відповідність вимогам машинного навчання та аналізу медичних зображень. На основі порівняльного аналізу Python [23] було обрано як найбільш оптимальний варіант завдяки його простоті та розвинутій екосистемі бібліотек. Середовище розробки обиралося на основі критеріїв інтеграції з Python, підтримки машинного навчання та зручності роботи з даними, і за результатами аналізу найкращим рішенням виявився PyCharm [29], який забезпечує всі необхідні інструменти для ефективної розробки та тестування нейромережі.

3 ОПИС ПРОГРАМИ

У даному розділі розглядаються принципи роботи розробленої програми для діагностики хвороби Альцгеймера за допомогою нейромережевого аналізу МРТ-знімків. Описуються етапи підготовки даних, алгоритм функціонування розробленої програми та алгоритми покращення її точності. Розглядаються методи обробки вибірки OASIS, використані підходи до оптимізації навчального процесу та засоби моніторингу прогресу навчання.

3.1 Алгоритм функціонування розробленої програми для діагностування хвороби Альцгеймера за МРТ-знімками

Алгоритм функціонування розробленої програми для діагностування хвороби Альцгеймера показано на рис.3.1.

Дана блок-схема описує процес підготовки, тренування та оцінки нейромережевої моделі на основі ResNet50 [6] із додатковим шаром SBA [4]. Процес починається з ініціалізації, імпорту необхідних бібліотек і налаштування логуювання, зокрема створення каталогу logs для збереження інформації про виконання моделі. Далі проводиться розподіл вхідних даних на три частини: Train (навчальна вибірка), Validation (вибірка для валідації) та Test (тестова вибірка). Якщо дані ще не розподілені, вони копіюються у відповідні папки.

Наступний етап включає створення генераторів зображень за допомогою ImageDataGenerator, що використовується для попередньої обробки даних. Потім визначається структура моделі ResNet50 із додатковим SBA шаром, після чого модель компілюється із заданим оптимізатором та функцією втрат. Навчання моделі відбувається протягом 50 епох із використанням callbacks для збереження найкращих параметрів або контролю за процесом.

Завершальний етап передбачає збереження натренованої моделі, її оцінку на тестовій вибірці та вивід отриманої точності. Така схема дозволяє автоматизувати процес навчання нейромережі та забезпечує чітку послідовність кроків від підготовки даних до отримання результату.

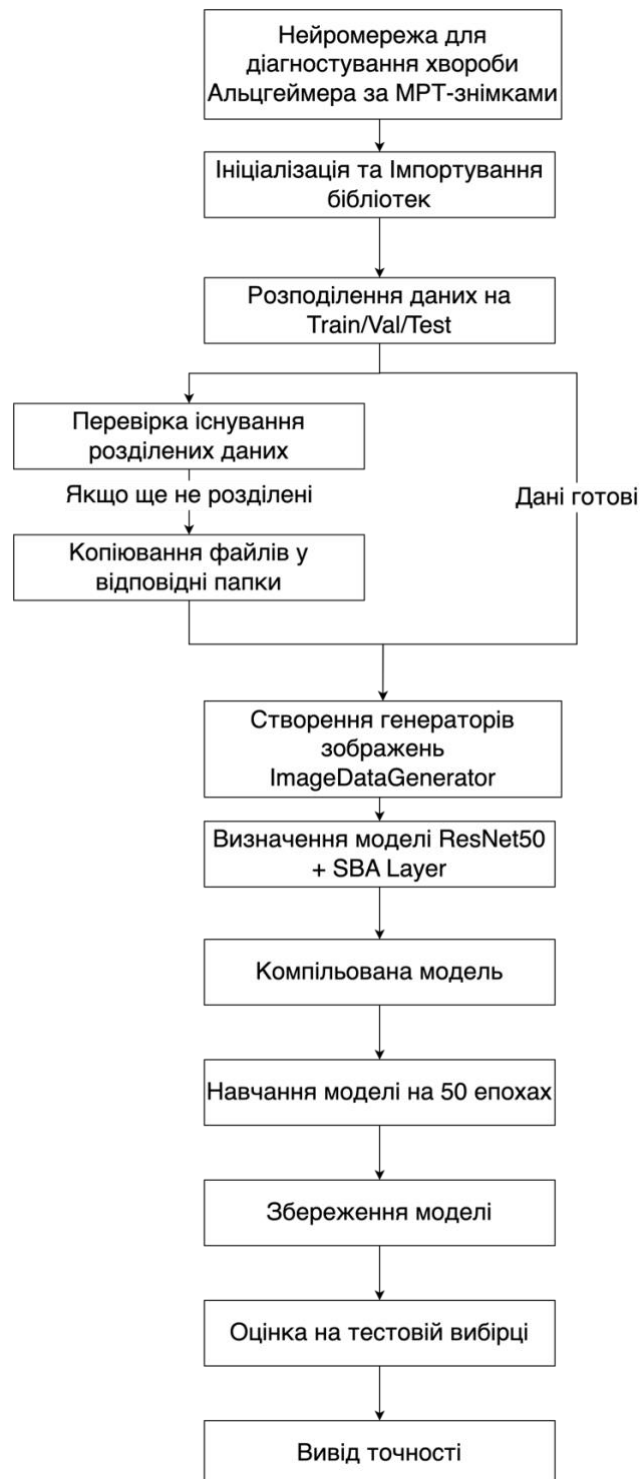


Рисунок 3.1 – Алгоритм функціонування розробленої програми

3.2 Опис обраної вибірки для тренування нейронної мережі

Вибірка для тренування нейронної мережі базується на наборі даних OASIS [9] (Open Access Series of Image Studies). Ця вибірка містить зображення сканів головного мозку, які використовуються для діагностики деменції на різних стадіях. Загальний обсяг датасету становить 86400 зображень, загальний розмір файлів – 1.36 ГБ. Вибірка структурована відповідно до рівнів вираженості когнітивних порушень, що дозволяє навчити нейромережу диференціювати різні стадії хвороби.

Основна папка датасету має назву Data та містить чотири підпапки, кожна з яких відповідає певному рівню хвороби:

- Mild Dementia – легка деменція;
- Moderate Dementia – помірна деменція;
- Non Demented – відсутність ознак деменції (контрольна група);
- Very Mild Dementia – дуже легка деменція.

Кожна з цих папок містить зображення, які представляють результати медичної візуалізації. Ці зображення є відсканованими томографічними зрізами головного мозку, отриманими за допомогою магнітно-резонансної томографії (МРТ) (рис.3.2 - 3.5).

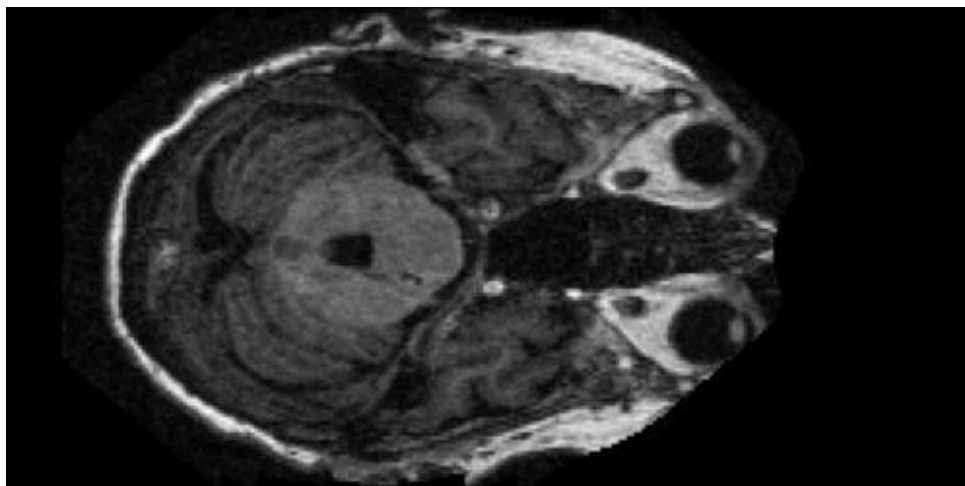


Рисунок 3.2 – Знімок МРТ з класу Mild Dementia [9]

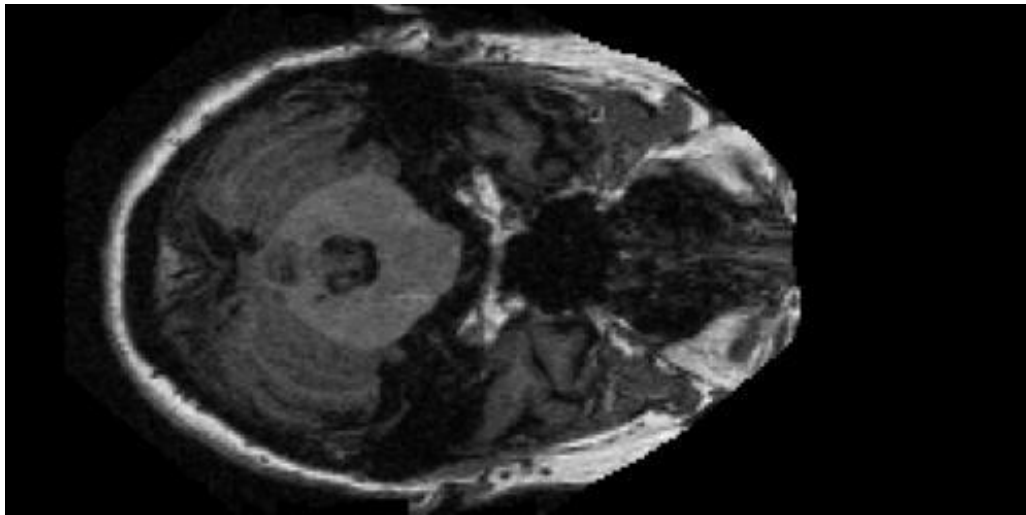


Рисунок 3.3 – Знімок МРТ з класу Moderate Dementia [9]

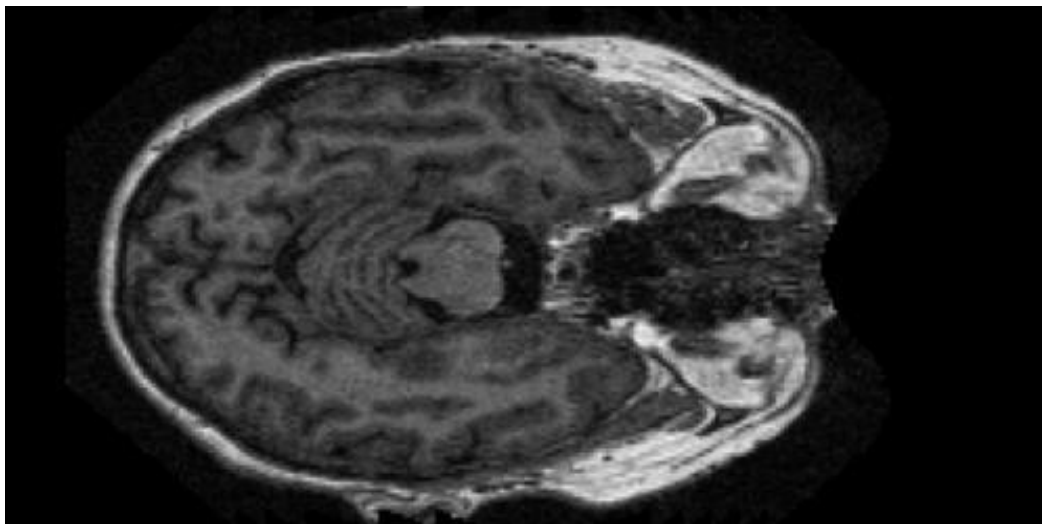


Рисунок 3.4 – Знімок МРТ з класу Non Demented [9]

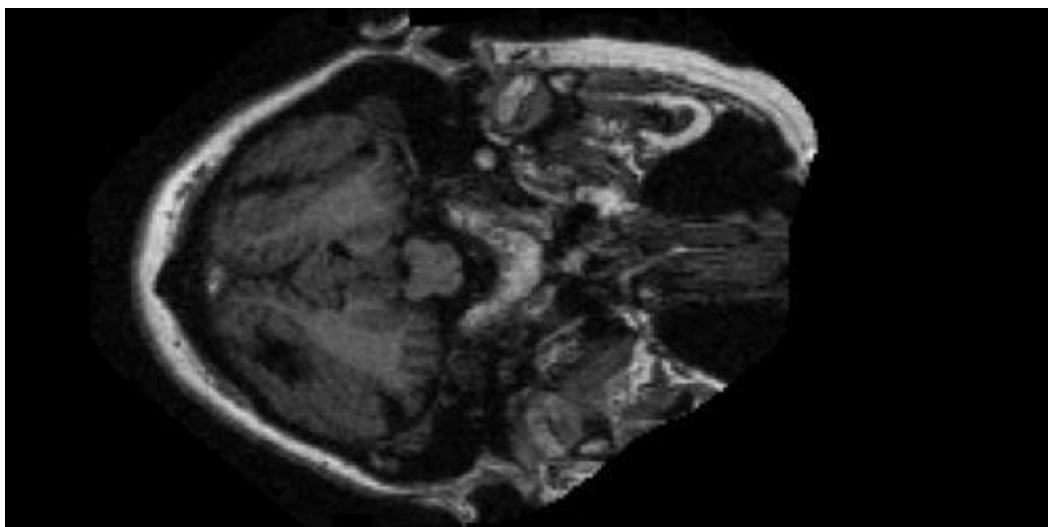


Рисунок 3.5 – Знімок МРТ з класу Very mild Dementia [9]

Перед тренуванням нейронної мережі необхідно виконати декілька кроків попередньої обробки:

а) поділ на підвибірки:

- 1) Train (тренувальна вибірка) – 60%;
- 2) Validation (валідаційна вибірка) – 20%;
- 3) Test (тестова вибірка) – 20%;

б) нормалізація зображень: масштабування інтенсивності пікселів у діапазон $[0, 1]$ за допомогою $\text{rescale}=1./255$.

Датасет використовується для навчання згорткової нейронної мережі (CNN [2]-[3]), побудованої на базі ResNet50 [6] із додаванням спеціалізованого шару Self-Boosted Attention (SBA [4]).

Модель тренується протягом 50 епох з використанням:

- оптимізатора: Adam [30];
- функція втрат: CustomLoss, яка враховує ваги класів для корекції дисбалансу у вибірці.

Валідаційна вибірка використовується для контролю процесу навчання, а тестова – для остаточної оцінки моделі.

Вибірка OASIS [9] забезпечує багатий набір даних для тренування глибокої нейронної мережі, що дозволяє автоматизувати діагностику хвороби на різних стадіях. Завдяки чіткій структурі, великій кількості зображень та можливості попередньої обробки цей датасет є ефективним інструментом для побудови високоточних моделей класифікації.

3.3 Опис використаних алгоритмів для покращення точності побудованої моделі

3.3.1 Self-Boosted Attention

Цей алгоритм покращує здатність нейромережі фокусуватися на важливих особливостях зображень, підсилюючи значущі області. Він працює шляхом використання згорткового шару з глибоким розділенням (Depthwise

Conv2D) для створення карти уваги, яка підсилює значущі області частини вхідного зображення. Це допомагає нейромережі краще виділяти паталогічні зміни, що є критично важливим у задачах медичної діагностики, таких як виявлення Альцгеймера (рис.3.6).

```
class SelfBoostedAttention(layers.Layer):
    def __init__(self, filters):
        super(SelfBoostedAttention, self).__init__()
        self.attention = layers.DepthwiseConv2D((3, 3), activation='sigmoid', padding='same')

    def call(self, inputs):
        attn_map = self.attention(inputs)
        return inputs * attn_map + inputs
```

Рисунок 3.6 – Текст програми, що реалізує алгоритм Self-Boosted Attention

3.3.2 Dynamic class Weight

Ця методика динамічно змінює ваги класів залежно від їх представленості у вибірці. Вона запобігає ситуаціям, коли модель переважно навчається на більш поширених класах, ігноруючи рідкісні. Після кожної епохи обчислюється кількість зразків кожного класу, і ваги коригуються так, щоб менш представлені класи отримували більшу значущість у функції втрат. Це покращує збалансованість навчання та допомагає уникнути упередженості моделі (рис.3.7).

```
class DynamicClassWeights(tf.keras.callbacks.Callback):
    def __init__(self, train_generator):
        super().__init__()
        self.train_generator = train_generator

    def on_epoch_end(self, epoch, logs=None):
        min_weight = 0.1
        max_weight = 10
        class_counts = np.bincount(self.train_generator.classes)
        total = np.sum(class_counts)
        new_weights = {i: np.clip(total / (len(class_counts) * count), min_weight, max_weight) for i, count in enumerate(class_counts) if count > 0}
        self.model.loss.class_weights = tf.convert_to_tensor([new_weights[i] for i in sorted(new_weights.keys())], dtype=tf.float32)
```

Рисунок 3.7 – Текст програми, що реалізує алгоритм Dynamic Class Weights

3.3.3 Hard Simple Mining

Цей метод спрямований на підсилення навчання моделі на важких для класифікації зразках. Після кожної епохи аналізуються прогнозовані та

реальні мітки, і зразки, де модель помилилася, позначаються як «важкі». Вони отримують підвищену увагу в наступних ітераціях навчання, що змушує модель більше уваги приділяти складним випадкам. Це покращує її здатність працювати з неоднозначними або складними зображеннями, що є важливим у діагностиці захворювань (рис.3.8).

```
class HardSampleMining(tf.keras.callbacks.Callback):
    def __init__(self, train_generator, factor=1.2):
        super().__init__()
        self.train_generator = train_generator
        self.factor = factor

    def on_epoch_end(self, epoch, logs=None):
        predictions = self.model.predict(self.train_generator)
        true_labels = self.train_generator.classes
        predicted_labels = np.argmax(predictions, axis=1)

        errors = true_labels != predicted_labels
        hard_samples = np.array(self.train_generator_filenames)[errors]

        print(f"Кількість важливих зразків після {epoch + 1} епохи: {len(hard_samples)}")
```

Рисунок 3.8 – Текст програми, що реалізує алгоритм Hard Simple Mining

3.4 Опис використаного методу для візуалізації прогресу навчання нейромережі

Для відстеження прогресу навчання моделі було використано TensorBoard [31], для цього було реалізовано клас TensorBoardBatchCallback.

Клас TensorBoardBatchCallback – це спеціальний зворотний виклик (callback) для Keras [25], який дозволяє записувати метрики навчання нейромережі в реальному часі з більшою гнучкістю, ніж стандартний TensorBoard. Він відрізняється тим, що оновлює значення метрики після певної кількості оброблених батчів, а не тільки в кінці епохи. Це корисно для моніторингу навчання в реальному часі та отримання більш детальної інформації про зміни показників якості (наприклад, loss та accuracy) протягом епохи (рис.3.9 – 3.10).

Використання цього підходу виправдане тим, що при тренуванні моделей на великих наборах даних оновлення метрик лише після завершення

епохи може бути недостатньо інформативним. Даний метод дозволяє оцінювати динаміку навчання на рівні батчів, що допомагає швидше виявляти проблеми, такі як перенавчання або занадто повільна зміна показників. Крім того, вибір частоти оновлення (`update_frequency`) у вигляді відносної величини дозволяє адаптивно підлаштовувати запис метрик під різну кількість батчів у епосі.

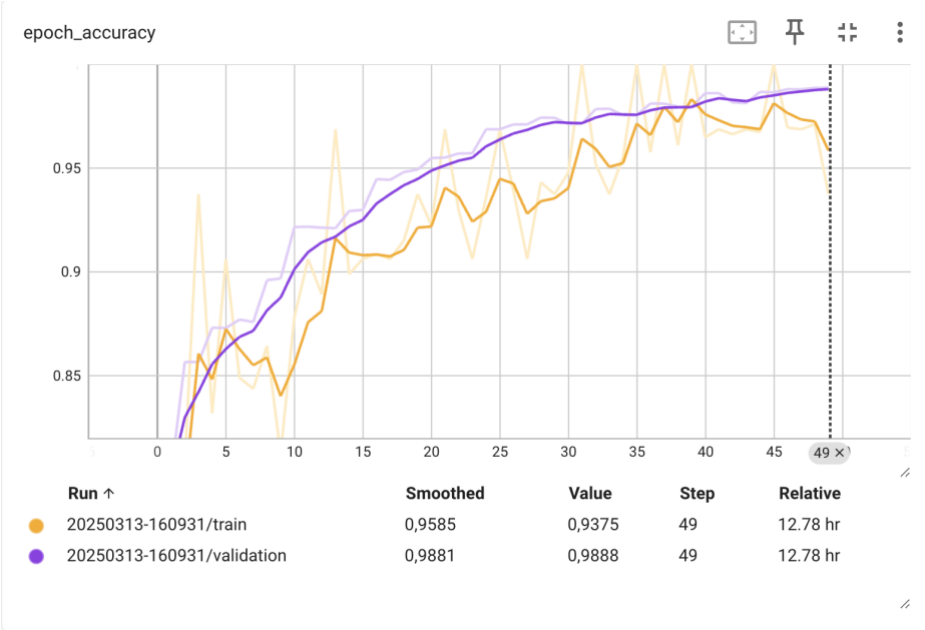


Рисунок 3.9 - Графік зміни точності в TensorBoard

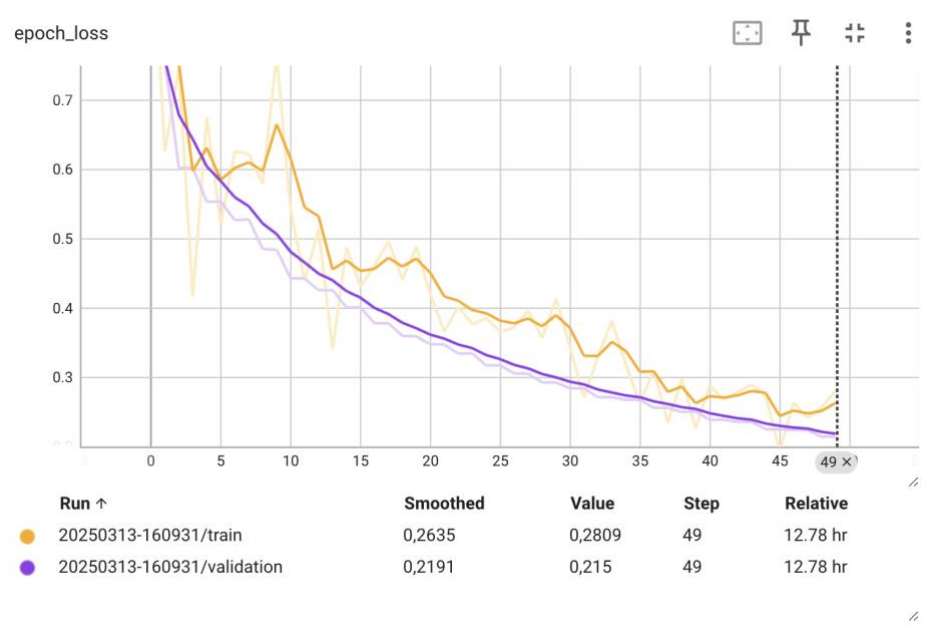


Рисунок 3.10 - Графік зміни функції втрат в TensorBoard

3.5 Висновки за розділом 3

У розділі 3 представлено алгоритм функціонування програми, її архітектуру та особливості роботи з вибіркою OASIS [9]. Детально описано методи підвищення точності моделі, зокрема Self-Boosted Attention [4], Dynamic Class Weights [5] і Hard Simple Mining [32]. Також було розглянуто спосіб візуалізації процесу навчання за допомогою TensorBoard [31], що забезпечує детальний контроль за змінами метрик у реальному часі.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

У цьому розділі розглядаються питання експлуатації, тестування та експериментального дослідження розробленої програми для виявлення хвороби Альцгеймера на основі нейронної мережі. Описується її призначення, умови застосування, архітектурні особливості, а також процес навчання та оцінки точності моделі. Тестування дозволяє визначити ефективність роботи нейромережі, оцінити її здатність до розпізнавання патологічних змін та знайти можливі напрями для покращення результатів.

4.1 Призначення й умови застосування програми

Програма написана в середовищі PyCharm [29] з використанням бібліотек TensorFlow [24] та Keras [25] для розробки моделі нейронної мережі, що призначена для виявлення Альцгеймера за допомогою класифікації медичних зображень. Модель базується на архітектурі ResNet50 [6] з додатковими шарами для покращення точності, такими як самокоригувальний механізм уваги [4], динамічне налаштування ваг класів [5] та навчання на важких зразках [32]. Програма також включає етапи передобробки даних, розбиття набору даних на тренувальні, валідаційні та тестові вибірки, а також використання TensorBoard [31] для візуалізації процесу навчання.

Функціональними можливостями програми є:

- завантаження та попередня обробка даних;
- розбиття даних на тренувальні, валідаційні та тестові набори;
- навчання моделі з використанням технік, таких як адаптивне коригування ваг класів, навчання на важких зразках та регуляризація;
- оцінка точності моделі на тестових даних;
- збереження та виведення результатів навчання в TensorBoard.

Потрібне апаратне забезпечення:

- оперативна пам'ять мінімум 16 ГБ для нормальної роботи з даними і тренуванням нейронної мережі;
- об'єм жорсткого диска достатньо 50 ГБ для зберігання даних, моделі та логів;
- процесор Intel i7 або еквівалентний для ефективного виконання обчислень;
- графічний процесор (GPU) для прискорення тренування бажано мати GPU з мінімум 8 ГБ пам'яті (наприклад, NVIDIA GTX 1060 або вище).

Потрібне програмне забезпечення:

- Python [23] 3.7 і вище;
- TensorFlow 2.x [24], Keras [25];
- бібліотеки для роботи з даними NumPy [33], OpenCV [34], Pillow;
- операційна система Linux або MacOS (Windows також підтримується).

4.2 Характеристики нейронної мережі для виявлення хвороби Альцгеймера

Нейронна мережа розроблена для класифікації зображень для виявлення різних стадій хвороби. Мережа використовує попередньо натреновану модель ResNet50 [6] в якості базової, що дозволяє досягти високої продуктивності на обмежених наборах даних. Для покращення результатів було додано спеціалізовані шари, зокрема механізм уваги SelfBoostedAttention [4], який допомагає мережі зосереджуватися на важливих частинах зображення, а також застосовано техніки регуляризації, такі як Dropout [35] та L2-регуляризація [35], щоб запобігти перенавчанню.

Для підвищення точності та стабільності навчання, мережа використовує додаткові стратегії, такі як динамічне налаштування ваг класів, що допомагає врахувати дисбаланс у кількості зразків для кожної категорії.

Також реалізовано стратегію вибору важких зразків, що дозволяє зосереджувати увагу на помилково класифікованих зображеннях, тим самим покращуючи точність моделі. Використання функції втрат CustomLoss забезпечує адаптивну вагу класів, що сприяє кращій стабільності навчання та підвищенню загальної ефективності моделі.

Модель є гнучкою, ефективною та спеціалізованою для медичних застосунків, таких як раннє виявлення хвороби на основі зображень мозку, і показує високу точність, використовуючи невеликий обсяг даних для тренування.

4.3 Інструкція по експлуатації програми

4.3.1 Підготовка даних

Спочатку необхідно підготувати набір зображень для навчання, валідації та тестування. Дані мають бути структуровані за категоріями. Потрібно використовувати функцію автоматичного поділу даних на тренувальний, валідаційний та тестовий набори, що реалізовано у коді. Вона створює відповідні директорії та копіює зображення до них, зберігаючи пропорції для кожного набору (60% для навчання, 20% для валідації та 20% для тестування).

4.3.2 Навчання моделі

Модель використовує архітектуру ResNet50 [6] як базову модель з додаванням уваги (SelfBoosted Attention [4]) для покращення результатів. Навчання проводиться за допомогою класу TensorBoardCallback, що дозволяє моніторити метрики за допомогою TensorBoard [31], а також за допомогою класу DynamicClassWeights [5], який коригує ваги класів на основі їх частоти у вибірці. Для навчання використовується кастомна функція втрат, що комбінує крос-ентропію та ваги класів.

4.3.3 Оцінка та збереження моделі

Після навчання модель оцінюється на тестових даних, і точність результату виводиться на екран. Модель також зберігається у файлі `alzheimer_detection_model.h5` для подальшого використання.

4.4 Тестування нейронної мережі для виявлення хвороби Альцгеймера

Для тестування розробленої нейромережі було розроблено окрему програму в якій розраховується точність класифікації для кожного з класів, а також кількість правильно віднесених зображень та загальна кількість зображень яка міститься в класах, для розрахунку результатів було обрано тестову вибірку. Для оцінки ефективності моделі було обрано саме метрику точності (accuracy [11]), оскільки вона відображає відсоток правильно класифікованих зображень від загальної кількості в кожному класі, що дозволяє об'єктивно порівнювати результати моделі для різних категорій деменції та визначати її здатність до патологічних змін. Результати представлені у табл. 4.1.

Таблиця 4.1 – Результати тестування нейромережі

Клас	Кількість правильно класифікованих зображень	Загальна кількість зображень	Точність (%)
Mild Dementia	989	1001	98.80
Moderate Dementia	92	98	93.88
Non Demented	13409	13445	99.73
Very mild Dementia	2574	2745	93.77

Нейромережева модель для виявлення хвороби Альцгеймера показала високу точність класифікації у всіх чотирьох категоріях. Найкращий результат досягнуто для класу Non Demented – 99.73%, що свідчить про високу здатність моделі розрізняти здорових пацієнтів. Класи Mild Dementia (98.80%) і Moderate Dementia (93.88%) також мають високу точність, що говорить про ефективність моделі у виявленні ранніх стадій деменції. Найнижча точність (93.77%) спостерігається для Very mild Dementia, що може свідчити про складність розмежування цього класу від здорових пацієнтів.

Загалом, отримані результати демонструють гарну узагальнювальну здатність моделі, але можливі покращення, зокрема для категорії Very mild Dementia. Це може включати додаткове балансування вибірки, удосконалення моделі або використання більш чутливих характеристик для аналізу.

4.5 Висновки за розділом 4

У цьому розділі було детально описано функціональні можливості розробленої програми, характеристики нейронної мережі та проведене тестування. Отримані результати свідчать про високу точність моделі у розпізнаванні різних стадій деменції, особливо для класу Non Demented. Водночас виявлено, що класифікація категорії Very mild Dementia може бути покращена за рахунок удосконалення моделі або методів балансування вибірки. Проведене тестування підтвердило ефективність запропонованого підходу та доцільність його використання у медичних дослідженнях.

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи бакалавра було досліджено та реалізовано нейромережеві методи для діагностування хвороби Альцгеймера на основі магнітно-резонансних томографічних (МРТ) знімків. Отримані результати вирішують практичну задачу автоматизованої класифікації стадій хвороби, що сприяє підвищенню точності діагностики та прискоренню аналізу медичних зображень. Проведене дослідження продемонструвало ефективність підходу, заснованого на використанні глибоких згорткових нейромереж, зокрема ResNet50 [6], із застосуванням додаткових покращень.

У роботі було проаналізовано ефективність різних нейромереж та реалізовано оптимальну архітектуру з механізмом самопосиленої уваги (Self-Boosted Attention [4]), адаптивним балансуванням класів та навчанням на важких зразках. Для підвищення стабільності моделі використано власну функцію втрат, що комбінує крос-ентропію з вагами класів, що дозволило досягти високої точності класифікації. Дані для навчання та тестування було взято з вибірки OASIS [9], яка містить МРТ-знімки пацієнтів з різними стадіями деменції. В результаті тестування модель досягла високих показників точності: найкращий результат отримано для класу Non Demented (99.73%), тоді як для Mild Dementia (98.80%) і Moderate Dementia (93.88%) також зафіксовано високу якість класифікації. Найнижча точність (93.77%) для Very mild Dementia, що може свідчити про складність розмежування цього класу від здорових пацієнтів.

Завдяки використанню бібліотек TensorFlow [24], Keras [25], NumPy [33] та використанню TensorBoard [31] було забезпечено ефективне навчання моделі, її моніторинг та аналіз динаміки тренування. Інтеграція TensorBoard дозволила в реальному часі відстежувати зміну функції втрат і точності, що сприяло кращому налаштуванню гіперпараметрів.

Подальше вдосконалення моделі може включати глибший аналіз особливостей зображень, використання більш складних механізмів уваги, додаткове балансування вибірки та оптимізацію архітектури мережі. Практична цінність роботи полягає у можливості застосування розробленої моделі у медичних діагностичних системах, що дозволить автоматизувати аналіз МРТ-знімків, зменшити навантаження на лікарів та підвищити якість ранньої діагностики деменції.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Dementia [Electronic resource]. – Access mode: <https://www.who.int/news-room/fact-sheets/detail/dementia>.
2. A CNN model: earlier diagnosis and classification of Alzheimer disease using MRI / [A. W. Salehi, P. Baglat, B. B. Sharma, G. Gupta, A. Upadhy] // 2020 International Conference on Smart Electronics and Communication (ICOSEC), Trichy, 10-12 September 2020: proceedings – India: IEEE, 2020. – P. 156-161. [Electronic resource]. – Access mode: <https://ieeexplore.ieee.org/abstract/document/9215402/>.
3. Ebrahimi A. Disease Neuroimaging Initiative A. Convolutional neural networks for Alzheimer's disease detection on MRI images / A. Ebrahimi, S. Luo // Journal of Medical Imaging. – 2021. – Vol. 8 (№ 2) – P. 024503-024503. [Electronic resource]. – Access mode: <https://www.spiedigitallibrary.org/journals/journal-of-medical-imaging/volume-8/issue-2/024503/Convolutional-neural-networks-for-Alzheimers-disease-detection-on-MRI-images/10.1117/1.JMI.8.2.024503.short>.
4. Improving fine-grained visual recognition in low data regimes via self-boosting attention mechanism / [Y. Shu, B. Yu, H. Xu, L. Liu] // European Conference on Computer Vision, Cham, 20 October 2022 : proceedings - Springer Nature Switzerland, 2022. – P. 449-465. [Electronic resource]. – Access mode: https://link.springer.com/chapter/10.1007/978-3-031-19806-9_26.
5. Feature selection using dynamic weights for classification / [X. Sun, Y. Liu, M. Xu, H. Chen, J. Han, K. Wang] // Knowledge-Based Systems. – 2013. – Vol. 37. – P. 541-549. [Electronic resource]. – Access mode: <https://www.sciencedirect.com/science/article/abs/pii/S0950705112002699>.
6. Classification of Alzheimer's disease with and without imagery using gradient boosted machines and ResNet-50 / [L. V. Fulton, D. Dolezel, J. Harrop, Yan Yan, C. P. Fulton] // Brain sciences. - 2019. – Vol. 9.(№ 9) – P. 212. [Electronic resource]. – Access mode: <https://www.mdpi.com/2076-3425/9/9/212>.

7. A deep learning based convolutional neural network model with VGG16 feature extractor for the detection of Alzheimer Disease using MRI scans / [S. Sharma, K. Guleria, S. Tiwari, S. Kumar] // Measurement: Sensors. - 2022. – Vol. 24. – P. 100506. [Electronic resource]. – Access mode: <https://www.sciencedirect.com/science/article/pii/S2665917422001404>.

8. Alzheimer's disease diagnosis using enhanced inception network based on brain magnetic resonance image / [Z. Cui, Z. Gao, J. Leng, T. Zhang, P. Quan, W. Zhao] // 2019 IEEE international conference on bioinformatics and biomedicine (BIBM), San Diego, 18-21 November 2019: proceedings – USA: IEEE, 2019. – P. 2324-2330. [Electronic resource]. – Access mode: <https://ieeexplore.ieee.org/abstract/document/8983046/>.

9. Link to source with sample [Electronic resource]. – Access mode: <https://www.kaggle.com/datasets/ninadaithal/imagesoasis>.

10. Automatic classification of patients with Alzheimer's disease from structural MRI: a comparison of ten methods using the ADNI database / [R. Cuingnet, E. Gerardin, J. Tessieras, G. Auzias, S. T. Lehericy, M. O. Habert, M. Chupin, H. Benali, O. Colliot] // Neuroimage. – 2011. – Vol. 56 (№ 2) – P. 766-781. [Electronic resource]. – Access mode: <https://www.sciencedirect.com/science/article/pii/S1053811910008578>.

11. Hossin M. A review on evaluation metrics for data classification evaluations / M. Hossin, M. N. Sulaiman // International journal of data mining & knowledge management process. – 2015. – Vol. 5(№ 2) – P. 1. [Electronic resource]. – Access mode: <https://www.academia.edu/download/37219940/5215ijdkp01.pdf>.

12. Pradhan A. Detection of Alzheimer's disease (AD) in MRI images using deep learning / A. Pradhan, J. Gige, M. Eliazar // Int. J. Eng. Res. Technol. – 2021. – Vol. 10(№ 3) – P. 580-585. [Electronic resource]. – Access mode: https://www.academia.edu/download/66199526/detection_of_alzheimers_disease_ad_in_mri_images_using_IJERTV10IS030310.pdf.

13. Ruoxuan C. RNN-based longitudinal analysis for diagnosis of Alzheimer's disease / C. Ruoxuan, L. Manhua // *Computerized Medical Imaging and Graphics*. – 2019. – Vol. 73. – P. 1-10. [Electronic resource]. – Access mode: <https://www.sciencedirect.com/science/article/pii/S0895611118303987>.

14. Cui R. Longitudinal analysis for Alzheimer's disease diagnosis using RNN / R. Cui, M. Liu, G. Li // 2018 IEEE 15th international symposium on biomedical imaging (ISBI 2018), Washington, 04-07 April 2018: proceedings – USA: IEEE, 2018. – P. 1398-1401. [Electronic resource]. – Access mode: <https://ieeexplore.ieee.org/abstract/document/8363833/>.

15. Predicting Alzheimer's disease using LSTM / [X. Hong, R. Lin, C. Yang, N. Zeng, C. Cai, J. Gou] // IEEE. – 2019. – Vol. 7. – P. 80893-80901. [Electronic resource]. – Access mode: <https://ieeexplore.ieee.org/abstract/document/8723315/>.

16. Matlani P. BiLSTM-ANN: early diagnosis of Alzheimer's disease using hybrid deep learning algorithms / P. Matlani // *Multimedia Tools and Applications*. – 2024. – Vol. 83 (№. 21) – P. 60761-60788. [Electronic resource]. – Access mode: <https://link.springer.com/article/10.1007/s11042-023-17867-5>.

17. Unlocking the potential of XAI for improved Alzheimer's disease detection and classification using a ViT-GRU model / [S. M. Mahim, Md. Shanin Ali, Md. Olid Hasan, A. Al Nomaan Nafi, A. Sadat, S. Al Hasan] // IEEE. – 2024. – Vol. 12. – P. 8390-8412. [Electronic resource]. – Access mode: <https://ieeexplore.ieee.org/abstract/document/10385046/>.

18. Classification of Moderate and Advanced Alzheimer's Patients Using Radial Basis Function Based Neural Networks Initialized with Fuzzy Logic / [C. R. Parra, A. P. Torres, J. M. Sotos, A. L. Borja] // *Irbm*. – 2023. – Vol. 44(№ 5) – P. 100795. [Electronic resource]. – Access mode: <https://www.sciencedirect.com/science/article/pii/S1959031823000441>.

19. Babu G. S. Alzheimer's disease detection using a Projection Based Learning Meta-cognitive RBF Network / G. S. Babu, S. Suresh, B. S. Mahanand // The 2012 International Joint Conference on Neural Networks (IJCNN), Brisbane,

10-15 June 2012: proceedings – Australia: IEEE, 2012. – P. 1-8. [Electronic resource] – Access mode: <https://ieeexplore.ieee.org/abstract/document/6252419/>.

20. A novel Alzheimer's disease detection approach using GAN-based brain slice image enhancement / [T. Bai, M. Du, L. Zhang, L. Ren, L. Ruan, Yuan Yang, G. Qian, Z. Meng, L. Zhao, M. J. Deen] // *Neurocomputing*. – 2022. – Vol. 492. – P. 353-369. [Electronic resource]. – Access mode: <https://www.sciencedirect.com/science/article/pii/S0925231222003782>.

21. Early prediction of alzheimer's disease progression using variational autoencoders / [S. Basu, K. Wagstyl, A. Zandifar, L. Collins, A. Romeo, D. Precup] // *Medical Image Computing and Computer Assisted Intervention–MICCAI 2019: 22nd International Conference, Shenzhen, 13–17 October, 2019: proceedings* – China: Springer International Publishing, 2019. – P. 205-213. [Electronic resource]. – Access mode: https://link.springer.com/chapter/10.1007/978-3-030-32251-9_23.

22. Parvatham N. K. Cycle GAN-Based MRI Augmentation and Hybrid Deep Learning Approach (GDP) for Binary Classification of Alzheimer's Disease: Differentiating Normal and Very Mild Dementia/ N. K. Parvatham, L. P. Maguluri // *International Journal of Intelligent Engineering & Systems*. – 2024. – Vol. 17 (№ 6) – P. 1281 [Electronic resource]. – Access mode: <https://search.ebscohost.com/login.aspx?direct=true&profile=ehost&scope=site&authtype=crawler&jrnl=2185310X&AN=180507194&h=XSibuZs%2FdJ50knmVostThp%2BVCYFVDAQ57nCA2fT8PtaMQcZ1ImPdTQ7y7ga%2BuDI9pj8XnlHnSQ2qsBdHsrgsxxg%3D%3D&crl=c>.

23. Van Rossum G. Python tutorial / G. Van Rossum, Jr F. L. Drake – Amsterdam, The Netherlands : Centrum voor Wiskunde en Informatica, 1995. – Vol. 620. [Electronic resource]. – Access mode: <https://zcmsdemo3.desss-portfolio.com/assets/images/zcms/pdf/testpdf1.pdf>.

24. Ertam F. Data classification with deep learning using Tensorflow / F. Ertam, G. Aydın // *2017 international conference on computer science and engineering (UBMK), Antalya, 05-08 October 2017 : proceedings* – Turkey: IEEE,

2017. – P. 755-758. [Electronic resource]. – Access mode: <https://ieeexplore.ieee.org/abstract/document/8093521/>.

25. Ramasubramanian K. Deep learning using keras and tensorflow / K. Ramasubramanian, A. Singh // Machine Learning Using R: With Time Series and Industry-Based Use Cases in R. – 2019. – P. 667-688. [Electronic resource]. – Access mode: https://link.springer.com/chapter/10.1007/978-1-4842-4215-5_11.

26. Imambi S. PyTorch / S. Imambi, K. B. Prakash, G. R. Kanagachidambaresan // Programming with TensorFlow: solution for edge computing applications. – 2021. – P. 87-104. [Electronic resource]. – Access mode: https://link.springer.com/chapter/10.1007/978-3-030-57077-4_10.

27. Scikit-learn: Machine learning in Python / [F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay] // the Journal of machine Learning research. – 2011. – Vol. 12. – P. 2825-2830. [Electronic resource]. – Access mode: https://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf?source=post_page.

28. Jupyter Notebooks—a publishing format for reproducible computational workflows / [T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, C. Willing] // Positioning and power in academic publishing: Players, agents and agendas. – IOS press, 2016. – P. 87-90. [Electronic resource]. – Access mode: <https://ebooks.iospress.nl/doi/10.3233/978-1-61499-649-1-87>.

29. Islam Q. N. Mastering PyCharm / Q. N. Islam // Packt Publishing Ltd. – 2015. [Electronic resource]. – Access mode: https://books.google.com/books?hl=ru&lr=&id=MPh_CwAAQBAJ&oi=fnd&pg=PP1&dq=PyCharm+Apple+ide&ots=GXYLY6hVEi0&sig=1OdTAzdzNGbcYOYJX3lMJWoCEAo.

30. Zhang Z. Improved adam optimizer for deep neural networks / Z. Zhang // 2018 IEEE/ACM: 26th international symposium on quality of service

(IWQoS), Banff, 04-06 June 2018: proceedings – Canada: IEEE, 2018. – P. 1-2. [Electronic resource]. – Access mode: <https://ieeexplore.ieee.org/abstract/document/8624183/>.

31. Vogelsang D. C. Magician's corner: 6. TensorFlow and TensorBoard / D. C. Vogelsang, B. J. Erickson // Radiology: Artificial Intelligence. – 2020. – Vol. 2 (№ 3) – P. e200012. [Electronic resource]. – Access mode: <https://pubs.rsna.org/doi/abs/10.1148/ryai.2020200012>.

32. Hard sample mining makes person re-identification more efficient and accurate / [K. Chen, Y. Chen, C. Han, N. Sang, C. Gao] // Neurocomputing. – 2020. – Vol. 382. – P. 259-267. [Electronic resource]. – Access mode: <https://www.sciencedirect.com/science/article/abs/pii/S0925231219316984>

33. Oliphant T. E. Guide to numpy. / T. E. Oliphant // USA : Trelgol Publishing. – 2006. – Vol. 1. – P. 85. [Electronic resource]. – Access mode: <https://ecs.wgtn.ac.nz/foswiki/pub/Support/ManualPagesAndDocumentation/numpybook.pdf>.

34. A brief introduction to OpenCV / [Ivan Culjak, David Abram, Tomislav Pribanic, Hrvoje Džapo, Mario Cifrek] // 2012 proceedings of the 35th international convention MIPRO, Opatija, 21-25 May 2012: proceedings. – Croatia: IEEE, 2012. – P. 1725-1730. [Electronic resource]. – Access mode: <https://ieeexplore.ieee.org/abstract/document/6240859/>.

35. Phaisangittisagul E. An analysis of the regularization between L2 and dropout in single hidden layer neural network / E. Phaisangittisagul // Modelling and Simulation (ISMS) : 7th International Conference on Intelligent Systems, Bangkok, 25-27 January 2016: proceedings – Thailand: IEEE, 2016. – P. 174-179. [Electronic resource]. – Access mode: <https://ieeexplore.ieee.org/abstract/document/7877209/>.

ДОДАТОК А
Технічне завдання

Вступ

Технічне завдання описує вимоги до розробки нейронної мережі для діагностування хвороби Альцгеймера. Метою проєкту є створення системи, здатної автоматично класифікувати медичні зображення мозку за категоріями (Non Demented, Very Mild Dementia, Moderate Dementia, Mild Dementia). Модель буде навчатися на основі вибірки OASIS та використовувати архітектуру ResNet50 з модифікаціями для підвищення точності.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Дослідження та програмна реалізація нейромережових засобів діагностування хвороби Альцгеймера за МРТ-знімками», затверджене наказом № 209 Національного університету «Запорізька політехніка» від 28 квітня 2025 р.

A.2 Призначення розробки

Програмний продукт призначений для раннього виявлення хвороби Альцгеймера.

A.3 Основні вимоги до програми, що розробляється

A.3.1 Вимоги до функціональних характеристик

Програма повинна виконувати передобробку зображень, їх класифікацію за допомогою нейромережі та формувати звіти з результатами прогнозування. Користувач повинен мати можливість завантажувати нові

зображення та отримувати прогноз про наявність хвороби з відповідною ймовірністю.

A.3.2 Вимоги до надійності

Система повинна мати точність класифікації не менше 85% на тестовій вибірці. Повинна бути реалізована перевірка коректності вхідних даних та захист від збоїв під час обробки. Логування роботи моделі та збереження вагових коефіцієнтів після кожного етапу навчання є обов'язковим.

A.3.3 Умови експлуатації

Програма розрахована на використання в дослідницьких медичних центрах або лікарнях, що займаються аналізом МРТ-знімків. Робота передбачається у середовищі Python із використанням TensorFlow та Keras.

A.3.4 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно запускатися на комп'ютерах із процесором не нижче Intel Core i7, мінімальним обсягом ОЗП 16 ГБ та графічним прискорювачем (наприклад, NVIDIA RTX 3060 або кращим). Дисковий простір повинен становити 100 ГБ.

A.3.5 Вимоги до пакування та маркування

Програмне забезпечення має бути упаковане у вигляді встановленого пакету або контейнера Docker, що включає усі необхідні залежності. Документація щодо встановлення та експлуатації повинна бути надана у форматі PDF або README-файлу.

A.4 Вхідні дані до роботи

Вхідні дані представлені у вигляді набору зображень МРТ, що мають стандартний формат JPG. Для навчання моделі використовується набір OASIS, який містить розмічені дані з категоріями Non Demented, Very Mild Dementia, Mild Dementia, Moderate Dementia. Дані будуть розділені на навчальну (60%), валідаційну (20%) та тестову (20%) вибірки.

ДОДАТОК Б
Тексти програм

Б.1 Текст програми з файлу main.py

```

import tensorflow as tf
from tensorflow.keras import layers, models, optimizers
from tensorflow.keras.preprocessing.image import
ImageDataGenerator
from tensorflow.keras.losses import Loss
from tensorflow.keras.applications import ResNet50
import numpy as np
import os
import shutil
import random
from datetime import datetime

log_dir = "logs/fit/" +
datetime.now().strftime("%Y%m%d-%H%M%S")
os.makedirs(log_dir, exist_ok=True)
tensorboard_callback =
tf.keras.callbacks.TensorBoard(log_dir=log_dir,
histogram_freq=1)

data_dir = "/Users/gorpinicirina/Documents/Data"
output_dir =
"/Users/gorpinicirina/Documents/Split_Data"

train_ratio = 0.6
val_ratio = 0.2
test_ratio = 0.2

def is_already_filled(split_dir):
    if not os.path.exists(split_dir):
        return False
    for category in os.listdir(split_dir):
        category_path = os.path.join(split_dir,
category)
        if os.path.isdir(category_path) and
os.listdir(category_path):
            return True
    return False

if all(is_already_filled(os.path.join(output_dir,
split)) for split in ["train", "val", "test"]):
    print("Дані вже розділені, пропускаємо
копіювання!")

```

```

else:
    for split in ["train", "val", "test"]:
        for category in os.listdir(data_dir):
            os.makedirs(os.path.join(output_dir, split,
category), exist_ok=True)
        for category in os.listdir(data_dir):
            category_path = os.path.join(data_dir,
category)
            if not os.path.isdir(category_path):
                continue
            images = os.listdir(category_path)
            random.shuffle(images)

            train_split = int(len(images) * train_ratio)
            val_split = int(len(images) * (train_ratio +
val_ratio))

            for i, img in enumerate(images):
                src_path = os.path.join(category_path, img)

                if i < train_split:
                    dst_path = os.path.join(output_dir,
"train", category, img)
                elif i < val_split:
                    dst_path = os.path.join(output_dir,
"val", category, img)
                else:
                    dst_path = os.path.join(output_dir,
"test", category, img)

                shutil.copy(src_path, dst_path)

            print("Дані успішно розділені на
train/val/test!")

class
TensorBoardBatchCallback(tf.keras.callbacks.Callback):
    def __init__(self, log_dir, update_frequency=0.25):
        super().__init__()
        self.log_dir = log_dir
        self.update_frequency = update_frequency
        self.writer =
tf.summary.create_file_writer(log_dir)
        self.step = 0

```

```

    def on_train_batch_end(self, batch, logs=None):
        logs = logs or {}
        total_batches = self.params['steps']
        update_interval = int(total_batches *
self.update_frequency)

        if batch % update_interval == 0 or batch ==
total_batches - 1:
            with self.writer.as_default():
                for metric, value in logs.items():

tf.summary.scalar(f'train/{metric}', value,
step=self.step)
                self.step += 1

    def on_test_batch_end(self, batch, logs=None):
        logs = logs or {}
        total_batches =
self.params.get('validation_steps', 0)
        if total_batches == 0:
            return
        update_interval = int(total_batches *
self.update_frequency)

        if batch % update_interval == 0 or batch ==
total_batches - 1:
            with self.writer.as_default():
                for metric, value in logs.items():
                    tf.summary.scalar(f'val/{metric}',
value, step=self.step)
            self.step += 1

class SelfBoostedAttention(layers.Layer):
    def __init__(self, filters):
        super(SelfBoostedAttention, self).__init__()
        self.attention = layers.DepthwiseConv2D((3,
3), activation='sigmoid', padding='same')

    def call(self, inputs):
        attn_map = self.attention(inputs)
        return inputs * attn_map + inputs

def build_model(input_shape=(96, 96, 3),
num_classes=4):

```

```

        base_model = ResNet50(weights="imagenet",
include_top=False, input_shape=input_shape)
        for layer in base_model.layers[:-5]:
            layer.trainable = False
        x = base_model.output
        x = SelfBoostedAttention(128)(x)
        x = layers.GlobalAveragePooling2D()(x)
        x = layers.BatchNormalization()(x)
        x = layers.Dropout(0.5)(x)
        x = layers.Dense(128, activation="relu",
kernel_regularizer=tf.keras.regularizers.l2(0.001))(x)
        outputs = layers.Dense(num_classes,
activation="softmax")(x)

    return models.Model(inputs=base_model.input,
outputs=outputs)

class DynamicClassWeights(tf.keras.callbacks.Callback):
    def __init__(self, train_generator):
        super().__init__()
        self.train_generator = train_generator

    def on_epoch_end(self, epoch, logs=None):
        min_weight = 0.1
        max_weight = 10
        class_counts =
np.bincount(self.train_generator.classes)
        total = np.sum(class_counts)
        new_weights = {i: np.clip(total /
(len(class_counts) * count), min_weight, max_weight)
for i, count in enumerate(class_counts) if count > 0}
        self.model.loss.class_weights =
tf.convert_to_tensor([new_weights[i] for i in
sorted(new_weights.keys())], dtype=tf.float32)

class CustomLoss(Loss):
    def __init__(self, class_weights):
        super(CustomLoss, self).__init__()
        weights_list = [class_weights[i] for i in
sorted(class_weights.keys())]
        self.class_weights =
tf.convert_to_tensor(weights_list, dtype=tf.float32)
    def call(self, y_true, y_pred):
        weights = tf.reduce_sum(self.class_weights *
y_true, axis=1)

```

```

        loss =
tf.keras.losses.categorical_crossentropy(y_true,
y_pred)
        return tf.reduce_mean(weights * loss)

datagen = ImageDataGenerator(
    rescale=1./255
)

train_generator = datagen.flow_from_directory(
    os.path.join(output_dir, "train"),
    target_size=(96, 96),
    batch_size=32,
    class_mode='categorical',
    subset='training',
    shuffle=True
)

val_generator = datagen.flow_from_directory(
    os.path.join(output_dir, "val"),
    target_size=(96, 96),
    batch_size=32,
    class_mode='categorical',
    shuffle=False
)

test_generator = datagen.flow_from_directory(
    os.path.join(output_dir, "test"),
    target_size=(96, 96),
    batch_size=32,
    class_mode='categorical',
    shuffle=False
)

class HardSampleMining(tf.keras.callbacks.Callback):
    def __init__(self, train_generator, factor=1.2):
        super().__init__()
        self.train_generator = train_generator
        self.factor = factor

    def on_epoch_end(self, epoch, logs=None):
        predictions =
self.model.predict(self.train_generator)
        true_labels = self.train_generator.classes

```

```

        predicted_labels = np.argmax(predictions,
axis=1)

        errors = true_labels != predicted_labels
        hard_samples =
np.array(self.train_generator_filenames)[errors]

        print(f"Кількість важливих зразків після {epoch
+ 1} епохи: {len(hard_samples)}")

class_weights = {i: 1.0 for i in range(4)}

model = build_model()

optimizer = optimizers.Adam(learning_rate=0.00001)

model.compile(
    optimizer=optimizer,
    loss=CustomLoss(class_weights),
    metrics=['accuracy']
)

tensorboard_batch_callback =
TensorBoardBatchCallback(log_dir=log_dir)
history = model.fit(
    train_generator,
    validation_data=val_generator,
    epochs=50,
    callbacks=[DynamicClassWeights(train_generator),
HardSampleMining(train_generator),
tensorboard_callback, tensorboard_batch_callback],
    steps_per_epoch=max(1, train_generator.samples //
train_generator.batch_size),
    validation_steps=max(1, val_generator.samples //
val_generator.batch_size)
)

model.save('alzheimer_detection_model.h5')

test_loss, test_accuracy =
model.evaluate(test_generator)
print(f"Test accuracy: {test_accuracy}")

```

Б.2 Текст програми з файлу `check_model.py`

```

import tensorflow as tf
import numpy as np
from tensorflow.keras.preprocessing import image
import os
import pandas as pd

class SelfBoostedAttention(tf.keras.layers.Layer):
    def __init__(self, filters, **kwargs):
        super(SelfBoostedAttention,
self).__init__(**kwargs)
        self.attention =
tf.keras.layers.DepthwiseConv2D((3, 3),
activation='sigmoid', padding='same')

    def call(self, inputs):
        attn_map = self.attention(inputs)
        return inputs * attn_map + inputs

    def get_config(self):
        config = super().get_config()
        config.update({"filters": 128})
        return config

model =
tf.keras.models.load_model('alzheimer_detection_model.h
5', custom_objects={'SelfBoostedAttention':
SelfBoostedAttention}, compile=False)

class_indices = {'Mild Dementia': 0, 'Moderate
Dementia': 1, 'Non Demented': 2, 'Very mild Dementia':
3}
class_names = {v: k for k, v in class_indices.items()}

image_dir = input("Введіть шлях до директорії із
зображеннями: ")

if not os.path.exists(image_dir):
    print("Вказана директорія не існує!")
    exit()

correct_predictions = {class_name: 0 for class_name in
class_indices.keys()}

```

```

total_images = {class_name: 0 for class_name in
class_indices.keys()}

for subdir, _, files in os.walk(image_dir):
    true_class_name = os.path.basename(subdir)
    if true_class_name not in class_indices:
        continue

    for file in files:
        img_path = os.path.join(subdir, file)

        img = image.load_img(img_path, target_size=(96,
96))
        img_array = image.img_to_array(img) / 255.0
        img_array = np.expand_dims(img_array, axis=0)

        predictions = model.predict(img_array)
        predicted_class_index =
np.argmax(predictions[0])
        predicted_class_name =
class_names[predicted_class_index]

        total_images[true_class_name] += 1
        if predicted_class_name == true_class_name:
            correct_predictions[true_class_name] += 1

accuracy_results = []
for class_name in class_indices.keys():
    total = total_images[class_name]
    correct = correct_predictions[class_name]
    accuracy = (correct / total * 100) if total > 0
else 0
    accuracy_results.append([class_name, correct,
total, round(accuracy, 2)])

df = pd.DataFrame(accuracy_results, columns=["Class",
"Correct predictions", "Total amount", "Accuracy (%)"])
print(df)

df.to_csv("classification_results.csv", index=False)
print("Результати збережені у
classification_results.csv")

```

ДОДАТОК В
Слайди презентації

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

Дослідження та програмна реалізація нейромережових засобів
діагностування хвороби Альцгеймера

Виконала ст. групи КНТ-131
Керівник зав. кафедри

Ірина ГОРПІНІЧ
Сергій СУББОТІН

Рисунок В.1 – Слайд 1

Об'єкт, предмет та мета роботи

Об'єкт дослідження – процес обчислювального діагностування деменції, керованого даними на основі магнітно-резонансної томографії (МРТ) головного мозку пацієнтів.

Предмет дослідження – методи та засоби автоматизованого діагностування стадій хвороби Альцгеймера за МРТ-знімками на основі глибоких нейронних мереж.

Мета роботи – розробка та оцінка ефективності нейромережевої моделі для ранньої діагностики хвороби Альцгеймера з метою підвищення точності класифікації до 90% на тестовій вибірці.

Рисунок В.2 – Слайд 2

Завдання до роботи

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

- виконати аналіз предметної області та нейромережових засобів для діагностування хвороби Альцгеймера;
- сформулювати вимоги до моделі нейронної мережі;
- розробити архітектуру нейромережі (ResNet50, Self-Boosted Attention тощо);
- навчання моделі з використанням оптимізаторів та функцій втрат;
- оптимізація параметрів;
- тестування нейромережі;
- аналіз отриманих результатів та порівняння з іншими методами.

3

Рисунок В.3 – Слайд 3

Порівняння існуючих аналогів

Нейромережа	Аналіз зображень	Аналіз часових рядів	Синтетичні дані	Вимоги до ресурсів
CNN	+	-	-	Високі
RNN/LSTM	-	+	-	Високі
RBF	+	-	-	Середні
Autoencoders/GAN	+	-	+	Дуже високі

4

Рисунок В.4 – Слайд 4

Обґрунтування вибору мови програмування

Критерій	Python	Java	C++	C#
Простота синтаксису	++	--	--	-
Продуктивність	-	+	++	+
Бібліотеки для AI	++	+	+	-
Гнучкість	++	+	-	-
Підтримка багатопотоковості	+	++	++	+
Кросплатформність	++	++	++	-
Використання у науці	++	+	+	-

5

Рисунок В.5 – Слайд 5

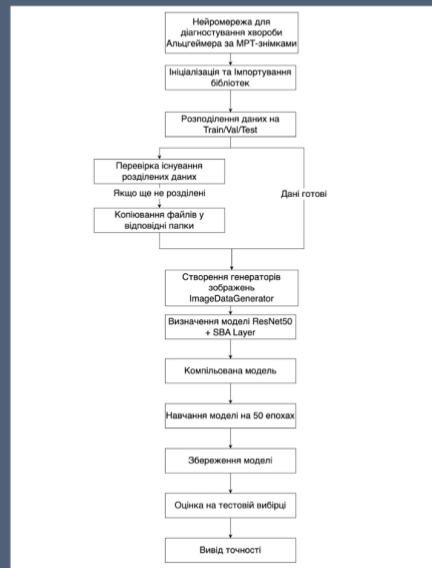
Вибір середовища розробки

Середовище	Підтримка Python	Інтеграція з ML-бібліотеками	Зручність налагодження	Підтримка Jupyter Notebook	Управління залежностями
PyCharm	++	++	++	+	++
Eclipse	+-	-	+	-	+
Xcode	-	-	+	-	-
Apache NetBeans	-	-	+-	-	+-

6

Рисунок В.6 – Слайд 6

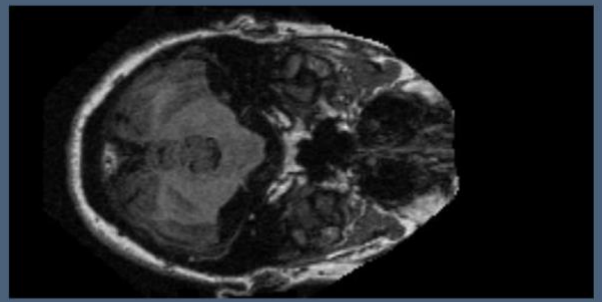
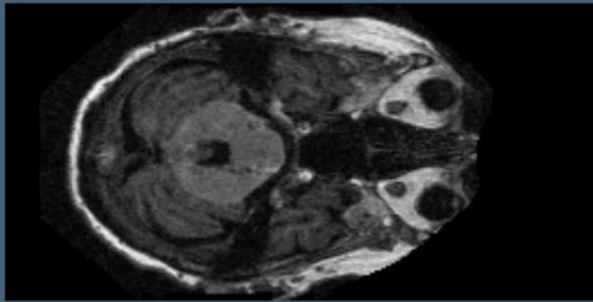
Алгоритм функціонування програми



7

Рисунок В.7 – Слайд 7

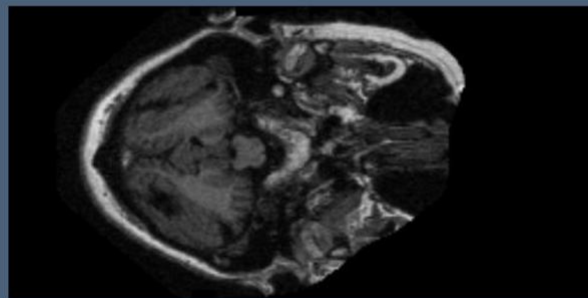
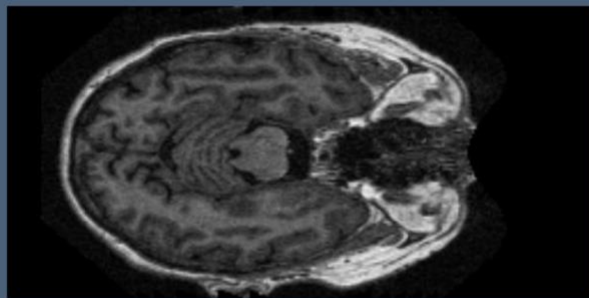
Приклади МРТ-знімків з класів вибірки Mild Dementia та Moderate Dementia



8

Рисунок В.8 – Слайд 8

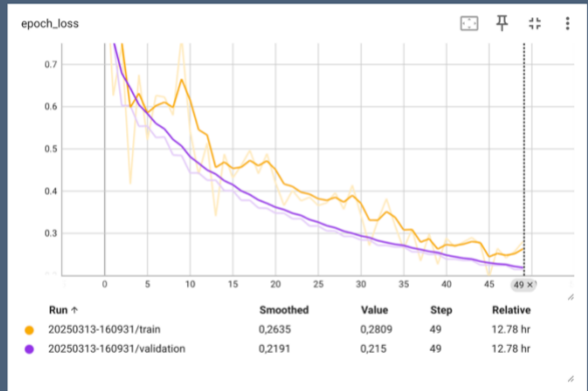
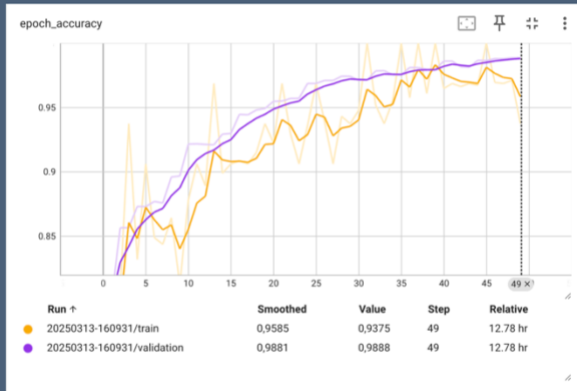
Приклади МРТ-знімків з класів вибірки Non Demented та Very mild Dementia



9

Рисунок В.9 – Слайд 9

Отримані графіки в ході роботи нейромережі



10

Рисунок В.10 – Слайд 10

Результати тестування нейронної мережі

Клас	Кількість правильно класифікованих	Загальна кількість зображень	Точність (%)
Mild Dementia	989	1001	98.80
Moderate Dementia	92	98	93.88
Non Demented	13409	13445	99.73
Very mild Dementia	2574	2745	93.77

11

Рисунок В.11 – Слайд 11

Висновки

- Досліджено та реалізовано нейромережеві методи для автоматизованої діагностики хвороби Альцгеймера за МРТ-знімками.
- Використано архітектуру ResNet50 із механізмом самопосиленої уваги (Self-Boosted Attention) та адаптивною функцією втрат, що дозволило покращити точність класифікації.
- Модель навчилася на вибірці OASIS, досягнувши високих результатів точності.
- Результати тестування:
 - Non Demented – 99.73%
 - Mild Dementia - 98.80%
 - Moderate Dementia - 93.88%
 - Very mild Dementia - 93.77%
- Розроблена система може бути інтегрована в медичні діагностичні процеси, що зменшить навантаження лікарів та підвищить якість аналізу.
- Дослідження підтвердило ефективність сучасних методів глибокого навчання для автоматизованої діагностики.

12

Рисунок В.12 – Слайд 12