МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інтелектуальних інформаційних систем

_____Юрій КОНДРАТЕНКО

«____»_____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ МАГІСТРА

ІНТЕЛЕКТУАЛЬНА СИСТЕМА ДІАГНОСТУВАННЯ НЕЙРОСУДИННИХ ХВОРОБ З ВИКОРИСТАННЯМ НЕЙРОННИХ МЕРЕЖ

Спеціальність 122 Комп'ютерні науки

Освітня програма «Інтелектуальні інформаційні системи»

Здобувачка _____Антоніна СЛОБОДЕНЮК

«___»_____ 2024 р.

Керівник канд. техн. наук, доцент _____ Євген СІДЕНКО

«___»_____ 2024 р.

Миколаїв – 2024

Чорноморський національний університет імені Петра Могили
(повне найменування закладу вищої освіти)

Факультет	Комп'ютерних наук
Кафедра	Інтелектуальних інформаційних систем
Рівень вищої освіти	Другий (магістерський)
Освітній ступень	Магістр
Спеціальність	122 Комп'ютерні науки
Освітня програма	Інтелектуальні інформаційні системи

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем

_____ Юрій КОНДРАТЕНКО

«____» _____ 2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу здобувача

Слободенюк Антоніни Ігорівни
((прізвище, ім'я, по батькові здобувачки)

1. Тема кваліфікаційної роботи: «Інтелектуальна система діагностування нейросудинних хвороб з використанням нейронних мереж».

Керівник роботи: Сіденко Євген Вікторович, доцент кафедри інтелектуальних інформаційних систем, канд. техн. наук, доцент.
Затверджена наказом ЧНУ ім. Петра Могили від «03» червня 2024 р. № 140/1.

2. Строк представлення кваліфікаційної роботи «17» грудня 2024 р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні: система, здатна класифікувати нейросудинні хвороби за допомогою КТ-зображення мозку, виконувати сегментацію уражених ділянок із розрахунком відсотка ураження, аналізувати симптоми пацієнта; початковими даними є медичні КТ-знімки.

4. Перелік питань, що підлягають розробці: аналіз сучасного стану задачі діагностування хвороб з використанням нейронних мереж; огляд існуючих методів

машинного навчання; вибір та налаштування відповідної архітектури нейронної мережі для сегментації та класифікації патологій мозку.

5. Перелік графічних матеріалів: презентація.

Керівник роботи _____ Євген СІДЕНКО
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Здобувачка _____ Антоніна СЛОБОДЕНЮК
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Дата видачі завдання «06» червня 2024 р.

КАЛЕНДАРНИЙ ПЛАН
кваліфікаційної роботи

Тема: Інтелектуальна система діагностування нейросудинних хвороб з використанням нейронних мереж.

№	Найменування роботи	Початок	Закінчення	Примітки
1	Отримання завдання на виконання КР	03.06.2024	07.06.2024	*Виконано*
2	Аналіз предметної області та постановка задачі	10.06.2024	20.06.2024	*Виконано*
3	Огляд літературних джерел за темою кваліфікаційної роботи включає аналіз публікацій та існуючих систем, присвячених діагностуванню нейросудинних хвороб	21.06.2024	01.07.2024	*Виконано*
4	Аналіз існуючих архітектур штучних нейронних мереж, що застосовуються для класифікації та сегментації медичних зображень.	01.09.2024	25.10.2024	*Виконано*
5	Впровадження обраних методів з подальшим аналізом отриманих результатів їх застосування.	26.10.2024	21.11.2024	*Виконано*
6	Перший попередній захист КР на засіданні комісії кафедри	22.11.2024	22.11.2024	*Виконано*
7	Корегування роботи за результатами попереднього захисту	23.11.2024	05.12.2024	*Виконано*
8	Другий попередній захист КР на засіданні комісії кафедри	06.12.2024	06.12.2024	*Виконано*
9	Доробка та остаточне оформлення КР	07.12.2024	10.12.2024	*Виконано*
10	Подання КР, її електронної копії та інших документів (відгуку, рецензії) до захисту	16.12.2024	17.12.2024	*Виконано*

Керівник роботи _____ Євген СІДЕНКО
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Здобувачка _____ Антоніна СЛОБОДЕНЮК
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Дата складання календарного плану
«21» червня 2024 р.

АНОТАЦІЯ

до кваліфікаційної роботи
здобувачки групи 601м ЧНУ ім. Петра Могили

Слободенюк Антоніни Ігорівни

на тему: **«ІНТЕЛЕКТУАЛЬНА СИСТЕМА ДІАГНОСТУВАННЯ НЕЙРОСУДИННИХ ХВОРОБ З ВИКОРИСТАННЯМ НЕЙРОННИХ МЕРЕЖ»**

Актуальність даного дослідження зумовлена зростаючою потребою у впровадженні інтелектуальних технологій для підтримки медичних рішень, зокрема в діагностиці нейросудинних захворювань. Сучасні статистичні дані свідчать про високий рівень смертності та інвалідності через інсульти та інші нейросудинні патології, що вимагає швидкої та точної діагностики. Використання нейронних мереж дозволяє автоматизувати аналіз великих обсягів медичних зображень, підвищуючи точність діагностування, скорочуючи час обробки даних і зменшуючи ризик людських помилок.

Розробка таких систем особливо важлива для медичних установ з обмеженими ресурсами, де немає доступу до висококваліфікованих спеціалістів. Інтелектуальна система допоможе лікарям визначати патології на ранніх стадіях, що сприятиме своєчасному лікуванню пацієнтів.

Об'єктом дослідження є процес діагностики нейросудинних хвороб за допомогою медичних зображень.

Предметом дослідження є методи розробки нейронних мереж для аналізу медичних зображень та автоматизації процесу діагностики нейросудинних захворювань.

Метою дослідження є діагностування нейросудинних хвороб за допомогою нейронних мереж.

В результаті виконання роботи було досліджено сучасні методи діагностики нейросудинних захворювань на основі медичних зображень, виконано огляд архітектур штучних нейронних мереж для автоматизації цього процесу, розроблено

та реалізовано модель для сегментації та класифікації патологій, а також протестовано її ефективність із використанням відповідних метрик.

Дана робота складається з чотирьох розділів. У першому досліджено нейросудинні хвороби, методи їх діагностики та сучасні інтелектуальні системи. Другий розділ описує математичні моделі, методи та інструменти, які використовувались у розробці. У третьому розділі описано проєктування системи та підготовку даних. Четвертий розділ присвячений створенню, тестуванню та впровадженню моделей. Загальний обсяг роботи – 94 сторінки. Кваліфікаційна робота містить 2 додатки, 50 рисунків, 7 таблиць і 45 джерел посилання.

Ключові слова: нейронні мережі, сегментація, кваліфікація, нейросудинні хвороби, U-Net архітектура, ResNet50 архітектура.

ABSTRACT

to the qualification work by the student of the group 601m of Petro Mohyla Black Sea National University

Slobodeniuk Antonina

«INTELLIGENT SYSTEM FOR DIAGNOSING NEUROVASCULAR DISEASES USING NEURAL NETWORKS»

A relevance of this study is driven by the growing need to implement intelligent technologies to support medical decisions, in particular in the diagnosis of neurovascular diseases. Modern statistics show a high level of mortality and disability due to strokes and other neurovascular pathologies, which requires fast and accurate diagnosis. The use of neural networks allows automating the analysis of large volumes of medical images, improving diagnostic accuracy, reducing data processing time, and reducing the risk of human error.

The development of such systems is especially important for medical institutions with limited resources, where there is no access to highly qualified specialists. The intelligent system will help doctors identify pathologies at early stages, which will facilitate timely treatment of patients. In addition, its implementation creates an opportunity to standardize diagnostic processes, which will improve the quality of medical services and reduce regional differences in health care.

An object of research is the process of diagnosing neurovascular diseases using medical images.

A subject of the study is the methods of developing neural networks for analyzing medical images and automating the process of diagnosing neurovascular diseases.

A purpose of the study is to diagnose neurovascular diseases using neural networks.

As a result of the work, we investigated modern methods for diagnosing neurovascular diseases based on medical images, reviewed artificial neural network architectures for automating this process, developed and implemented a model for

segmenting and classifying pathologies, and tested its effectiveness using appropriate metrics.

This paper consists of four sections. The first section examines neurovascular diseases, methods of their diagnosis, and modern intelligent systems. The second section describes the mathematical models, methods, and tools used in the development. The third section describes the system design and data preparation. The fourth section is devoted to the creation, testing, and implementation of models. The total volume of the work is 97 pages. The qualification work contains 2 appendixes, 50 figures, 7 tables, and 45 references.

Key words: neural networks, segmentation, qualification, neurovascular diseases, U-Net architecture, ResNet50 architecture.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

КТ	–	Комп’ютерна томографія
МРТ	–	Магнітно-резонансна томографія
ЦСА	–	Цифрова субтракційна ангіографія
УЗД	–	Ультразвукова доплерографія
ШІ	–	Штучний інтелект

SVM	–	Support vector machine
CNN	–	Convolutional Neural Networks
RNN	–	Recurrent neural networks
DNN	–	Deep neural networks
GAN	–	Generative adversarial networks
VGG	–	Visual Geometry Group
ReLU	–	Rectified Linear Unit
ASPP	–	Atrous Spatial Pyramid Pooling

ВСТУП

Сучасна медицина знаходиться на стадії активного розвитку завдяки впровадженню новітніх технологій, серед яких особливе місце займають штучний інтелект (ШІ) та методи машинного навчання. Однією з ключових проблем охорони здоров’я є своєчасна та точна діагностика складних захворювань, зокрема нейросудинних хвороб, які займають одне з перших місць за рівнем смертності та інвалідності у світі. Такі захворювання, як інсульт, аневризми та тромбози мозкових судин, вимагають швидкої та точної діагностики для збереження життя пацієнтів та уникнення важких наслідків.

Традиційні методи діагностики, такі як магнітно-резонансна томографія (МРТ), комп’ютерна томографія (КТ) та ангіографія, є ефективними, але водночас вимагають значного часу на аналіз результатів і часто залежать від кваліфікації лікаря. Недостатня кількість фахівців у сфері нейросудинних захворювань та можливість людських помилок є суттєвими факторами ризику. Саме тут на допомогу приходять інтелектуальні системи, здатні автоматизувати процес діагностики та підвищити її точність завдяки використанню нейронних мереж.

Об'єкт роботи – процес діагностики нейросудинних хвороб за допомогою медичних зображень.

Предмет роботи – методи розробки нейронних мереж для аналізу медичних зображень та автоматизації процесу діагностики нейросудинних захворювань.

1 АНАЛІЗ ПРЕДМЕТНОЇ СФЕРИ. ПОСТАНОВКА ЗАДАЧІ

1.1 Огляд нейросудинних захворювань

Нейросудинні захворювання [1] (цереброваскулярні захворювання) охоплюють широкий спектр патологій, пов'язаних з порушенням кровообігу в судинах головного мозку. Ці хвороби є однією з основних причин смертності та інвалідності у світі. Розвиток таких захворювань може призвести до серйозних уражень нервової тканини мозку, порушень функцій організму та навіть смерті.

1.1.1 Інсульт

Інсульт [2] – це небезпечний для життя стан, який виникає, коли частина мозку не отримує достатньої кількості крові. Найчастіше це відбувається через закупорку артерії або крововилив у мозок. Без постійного надходження крові клітини мозку в цій області починають відмирати через нестачу кисню. Якщо в певній області гине достатня кількість клітин мозку, пошкодження стає постійним, і є можливість втратити здібності, якими раніше керувала цю область. Однак відновлення кровотоку може запобігти такому пошкодженню або принаймні обмежити його тяжкість. Ось чому час має вирішальне значення для лікування інсульту. Існує два основних типи інсультів.

Ішемічний інсульт – це найпоширеніший вид інсульту. Він трапляється, коли кровоносні судини головного мозку звужуються або закупорюються. Це викликає зниження кровотоку, відоме як ішемія. Заблоковані або звужені кровоносні судини можуть бути спричинені жировими відкладеннями, які накопичуються в кровоносних судинах. Або вони можуть бути викликані згустками крові, іншим сміттям, що переміщається через кровотік, найчастіше із серця (рис. 1.1). Це призводить до недостатнього постачання крові до мозку. Ішемічні інсульти складають близько 85% усіх випадків. це найпоширеніший вид інсульту, що становить близько 85% усіх випадків. Він виникає, коли кровоносні судини головного мозку звужуються або закупорюються, що призводить до

зниження кровотоку – стану, відомого як ішемія. Основною причиною звуження або закупорки судин є атеросклероз – процес накопичення жирових відкладень (бляшок) на стінках артерій. З часом ці бляшки можуть зрости до таких розмірів, що повністю блокують кровотік або значно його обмежують. Крім того, ішемічний інсульт може бути викликаний утворенням згустків крові (тромбів), які часто утворюються в серці або великих артеріях і переміщуються через кровоносну систему до мозку (рис. 1.1). Такі згустки можуть закупорити мозкову артерію, викликаючи раптове припинення постачання кисню та поживних речовин до тканин мозку.

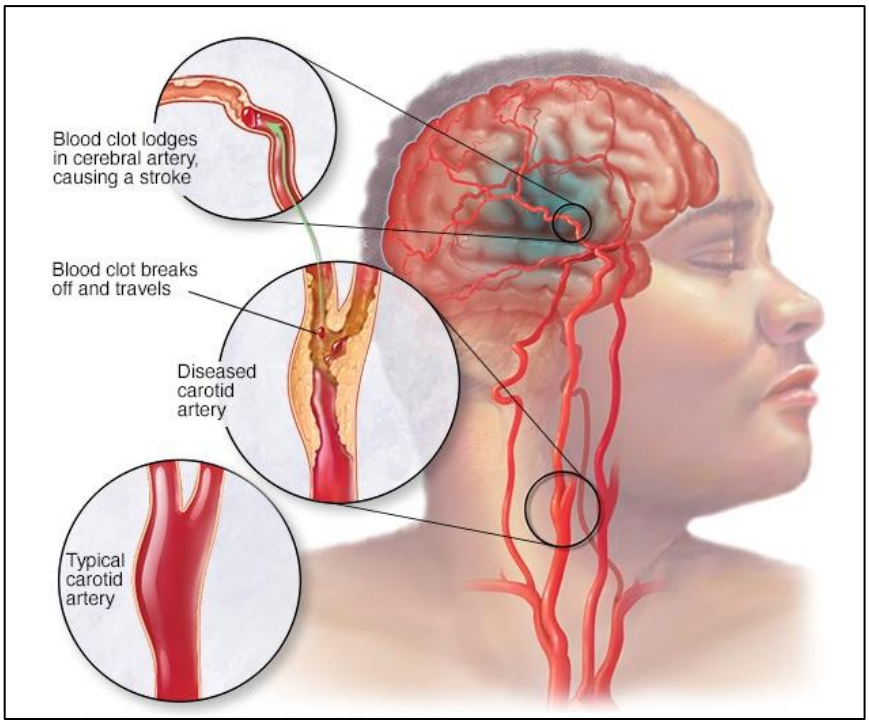

Рисунок 1.1 – Ішемічний інсульт

Коли мозок не отримує достатньої кількості крові, клітини нервової тканини починають пошкоджуватися або гинути протягом декількох хвилин, що призводить до втрати функцій, які контролює уражена ділянка мозку. Найчастіше ішемічні інсульти вражають моторну функцію, мовлення, пам'ять або інші важливі когнітивні здібності, що може призвести до серйозних наслідків, таких як параліч, афазія (порушення мовлення) або навіть смерть.

Ефективність лікування ішемічного інсульту залежить від швидкості надання медичної допомоги. Основним методом лікування є тромболітична терапія, яка розчиняє згустки крові і відновлює кровообіг у мозку. Проте цей метод є ефективним лише у перші години після виникнення симптомів, тому своєчасна діагностика має вирішальне значення.

Геморагічний інсульт виникає, коли кровоносна судина в мозку розривається або протікає, що призводить до крововиливу в тканини мозку (рис. 1.2). Це викликає сильний тиск на навколишні нейрони, пошкоджуючи їх і порушуючи нормальну роботу мозкових структур. Цей тип інсульту менш поширений, ніж ішемічний, проте він є більш небезпечним і має вищий рівень смертності та інвалідності.

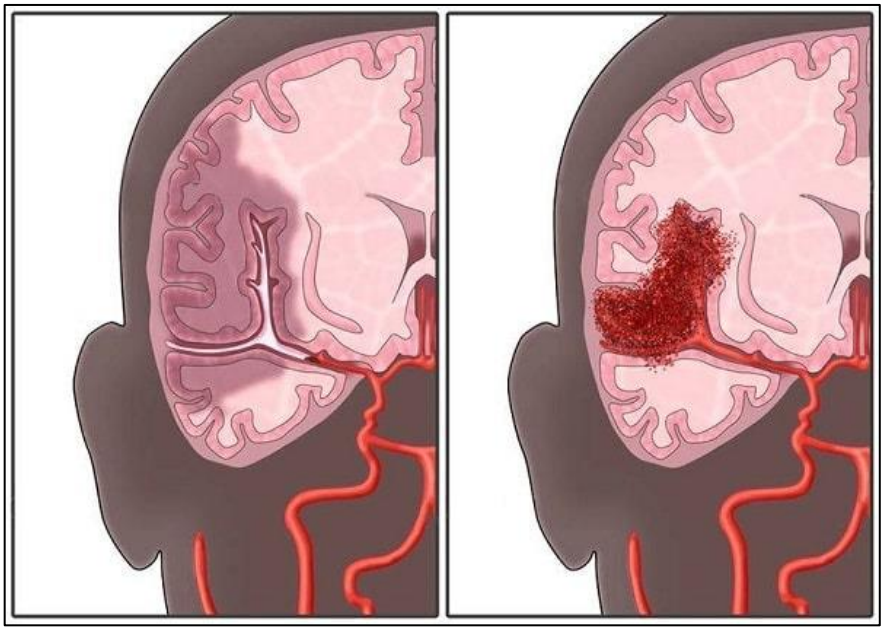

Рисунок 1.2 – Геморагічний інсульт

Геморагічні інсульти поділяються на два основних типи: внутрішньомозковий крововилив та субарахноїдальний крововилив.

1. Внутрішньомозковий крововилив відбувається, коли артерія всередині мозку розривається, що призводить до накопичення крові в оточуючих тканинах. Причиною цього найчастіше є хронічна гіпертонія (високий кров'яний тиск), яка поступово послаблює стінки судин, роблячи їх більш уразливими до розриву.

2. Субарахноїдальний крововилив виникає при розриві судини у просторі між мозком і його оболонками (субарахноїдальний простір). Найпоширенішою причиною є розрив аневризми. Субарахноїдальний крововилив часто супроводжується раптовим і сильним головним болем, який пацієнти описують як "удар блискавки". Він також може супроводжуватися нудотою, блюванням, втратою свідомості та іншими серйозними неврологічними симптомами.

Фактори, пов’язані з геморагічним інсультом, включають:

- високий кров'яний тиск, який не контролюється;
- надмірне лікування препаратами для розрідження крові;
- випуклості в слабких місцях стінок кровоносних судин, відомі як аневризми;
- травма голови, наприклад, внаслідок автомобільної аварії;
- відкладення білка в стінках кровоносних судин, що призводить до слабкості стінок судин. Це відоме як церебральна амілоїдна ангіопатія;
- ішемічний інсульт, що призводить до крововиливу в мозок.

Основні симптоми інсульту можна легко запам'ятати за допомогою фрази «BE FAST» (рис.1.2), яка є мнемонічним способом швидко визначити ознаки інсульту та негайно діяти. Кожна буква цієї фрази представляє один із ключових симптомів:

- **B (Balance)** – порушення рівноваги: раптове запаморочення, втрата координації або рівноваги, труднощі з ходьбою можуть бути ознаками інсульту. Людина може почати хитатися або впасти;
- **E (Eyes)** – проблеми з зором: раптове погіршення зору, зокрема втрата зору на одне око, двоїння в очах або повна втрата зору можуть свідчити про інсульт;
- **F (Face)** – обличчя: оніміння або слабкість на одній стороні обличчя. Найчастіше помітно, що одна сторона обличчя "падає", коли людина намагається посміхнутися;

– **A (Arms)** – руки: слабкість або оніміння в одній або обох руках. Якщо попросити людину підняти обидві руки, одна з них може опускатися або не підніматись взагалі;

– **S (Speech)** – мовлення: раптові труднощі з мовленням або його нерозбірливість. Людина не здатна говорити чітко, вимовляти слова або навіть втратити здатність розуміти мову;

– **T (Time)** – час є вирішальним фактором. Якщо є будь-який із цих симптомів, негайно треба викликати швидку допомогу.

Фраза "BE FAST" наголошує на необхідності швидкого реагування, оскільки при інсульті кожна хвилина має значення.

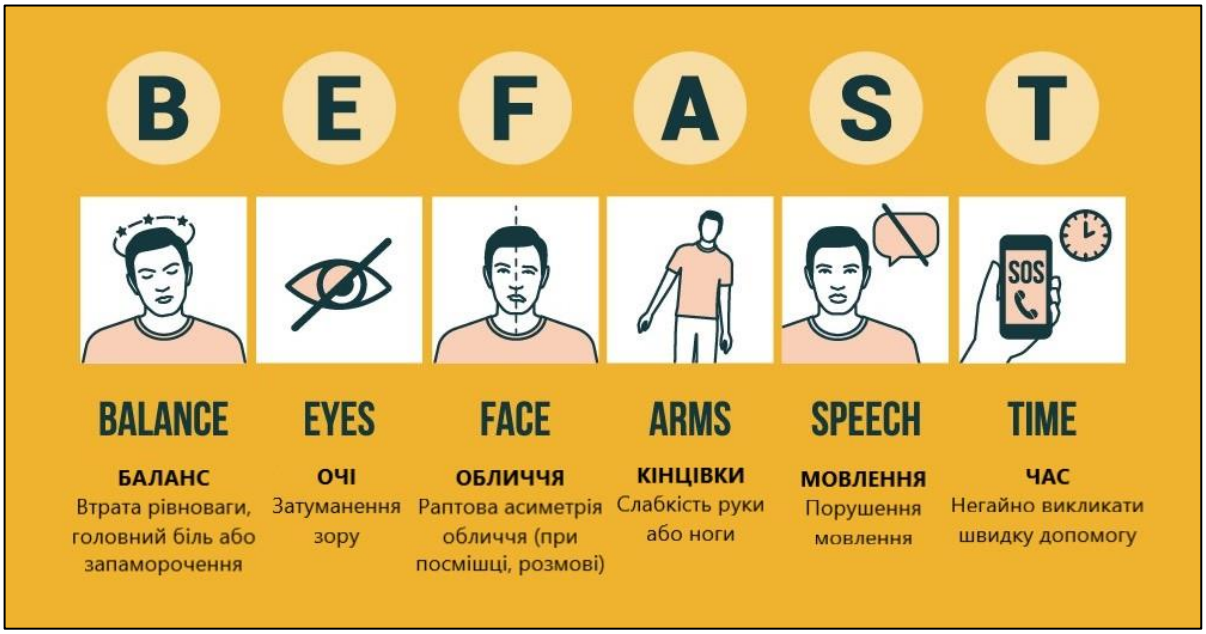

Рисунок 1.2 – Фраза «BE FAST»

До симптомів інсульту можна також віднести запаморочення, нудоту і блювання, ригідність шиї, сплутаність свідомості або збудження, судоми, втрату пам'яті (амнезія), головні болі (зазвичай раптові та сильні), втрату свідомості або непритомність.

1.1.2 Аневризми судин мозку

Аневризма головного мозку [3] – це опуклість або здуття кровоносної судини головного мозку. Аневризма часто виглядає як ягода, що висить на ніжці (рис. 1.3).

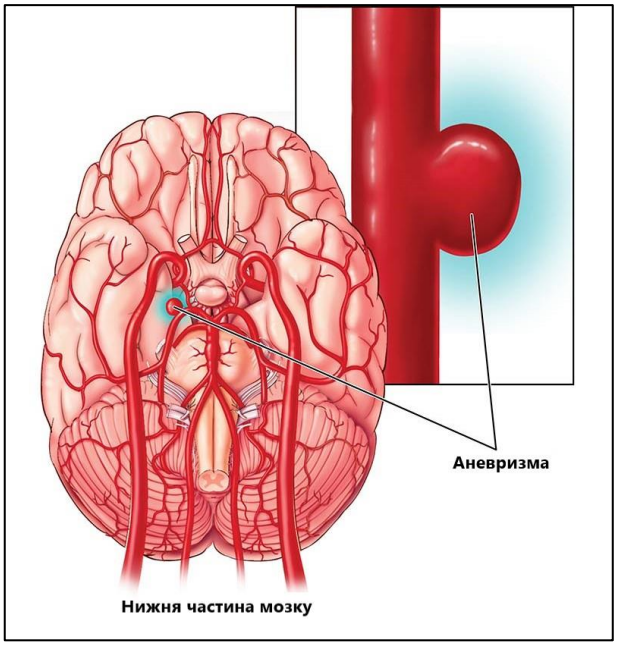

Рисунок 1.3 – Аневризма

Експерти вважають, що аневризми головного мозку утворюються і ростуть через те, що кров, яка тече через кровоносну судину, тисне на слабку ділянку стінки судини. Це може збільшити розмір аневризми головного мозку. Якщо аневризма головного мозку протікає або розривається, це викликає крововилив у мозок, тобто геморагічний інсульт.

Наявність аневризми головного мозку може бути невідомо, поки вона не розірветься. Більшість аневризм головного мозку не мають симптомів і мають невеликий розмір (менше 10 міліметрів). Менші аневризми можуть мати менший ризик розриву.

Однак іноді можуть виникати симптоми, які виникають перед розривом через невелику кількість крові, яка може витікати. Це називається «дозорним крововиливом» у мозок. Деякі аневризми є симптоматичними, оскільки вони

тиснуть на сусідні структури, такі як нерви ока. Вони можуть спричинити втрату зору або зменшення рухів очей, навіть якщо аневризма не розірвалася.

Симптоми нерозривної аневризми головного мозку включають наступне:

- головні болі (рідкісні, якщо не розриваються);
- біль в очах;
- зміни зору;
- зменшення руху очей;

Лікування нерозривної аневризми головного мозку може запобігти розриву в майбутньому.

1.2 Методи діагностики нейросудинних хвороб

Діагностика нейросудинних хвороб є надзвичайно важливою для своєчасного виявлення та лікування таких патологій, як інсульт, аневризма та інші порушення кровообігу в головному мозку. Враховуючи високу поширеність і серйозність цих захворювань, сучасні методи діагностики спрямовані на швидке та точне виявлення змін у структурі та функціях мозку.

Комп'ютерна томографія (КТ) – це неінвазивний метод рентгенівського дослідження [4], який дозволяє отримати пошарове зображення мозку. КТ широко використовується для швидкої діагностики гострих станів, таких як інсульт, особливо геморагічний.

КТ є ефективним для виявлення внутрішньочерепних крововиливів (геморагічний інсульт, субарахноїдальний крововилив). Дає змогу виявляти набряк мозку та його уражені ділянки. Швидкість проведення процедури робить КТ основним методом для екстреної діагностики.

Але існують і обмеження:

- менш ефективна для виявлення дрібних ішемічних уражень на ранніх стадіях;
- порівняно низька точність візуалізації судинних структур без використання контрастних речовин.

Магнітно-резонансна томографія (МРТ) [5] – це високоточний метод візуалізації, що використовує магнітне поле і радіохвилі для отримання детальних зображень мозкової тканини. МРТ є більш чутливим у порівнянні з КТ для виявлення ішемічних уражень та аномалій судин. Дозволяє отримати високу роздільну здатність зображень і чітко виявляти навіть дрібні патології. МРТ детально візуалізує судини головного мозку для діагностики аневризм, стенозів і тромбозів.

Обмеження:

- висока вартість і тривалість дослідження;
- протипоказання для пацієнтів із металевими імплантатами або кардіостимуляторами.

Комп'ютерна томографічна ангіографія (КТ-ангіографія) – це спеціальний метод діагностики, який використовує комп'ютерну томографію з введенням контрастної речовини для візуалізації кровоносних судин мозку [6]. Цей метод дозволяє отримати тривимірні зображення судин і виявляти їхню анатомію. Ефективно виявляє аневризми, тромби та стенози артерій. Тривимірна реконструкція судин дає можливість точно оцінити їхній стан.

Обмеження:

- вимагає введення контрастної речовини, що може бути небезпечним для пацієнтів із порушеннями функції нирок або алергією на контраст;
- радіаційне навантаження є вищим, ніж при звичайній КТ.

Магнітно-резонансна ангіографія (МР-ангіографія) [7] – це метод, який використовує магнітно-резонансну томографію для вивчення стану судин без використання іонізуючого випромінювання. Не потребує використання контрастної речовини в більшості випадків. Підходить для довготривалого спостереження за станом судин без радіаційного навантаження.

Обмеження:

- менш ефективна для дослідження дрібних судин у порівнянні з КТ-ангіографією;

− довший час сканування, що може бути проблемою для пацієнтів з гострими станами.

Цифрова субтракційна ангіографія (ЦСА) [8] є інвазивним методом діагностики, який використовується для детального вивчення судин мозку. У процесі дослідження вводиться контрастна речовина, а рентгенівські зображення дозволяють виявити аномалії судин. Має високу точність візуалізації навіть найдрібніших судин. Використовується для передопераційної оцінки стану судин або для планування хірургічного втручання.

Обмеження:

− вимагає інвазивного втручання, що несе ризики для пацієнта;

− радіаційне навантаження та можливі алергічні реакції на контраст.

Ультразвукова доплерографія (УЗД) судин голови та шиї – це неінвазивний метод дослідження, який використовує ультразвук для оцінки швидкості та напрямку кровотоку в основних судинах голови та шиї [9]. Швидкий і безпечний метод для первинної діагностики стенозу артерій або виявлення тромбів.
Не потребує введення контрастних речовин чи опромінення.

Обмеження:

− менш точний для дослідження дрібних судин мозку;

− не може замінити методи візуалізації для виявлення структурних патологій мозку.

1.3 Аналіз сучасних інтелектуальних систем у медицині

Останні роки ознаменувалися значним розвитком інтелектуальних систем у медицині, зокрема завдяки використанню методів штучного інтелекту (ШІ) і машинного навчання. Інтелектуальні системи здатні автоматизувати багато процесів, що раніше вимагали значної участі медичних фахівців, від діагностики до вибору лікування. Однією з найважливіших галузей застосування ШІ є діагностика на основі аналізу медичних зображень, таких як рентген, МРТ і КТ. Це

дозволяє значно підвищити точність і швидкість діагностичних процесів, знижуючи людський фактор і можливість помилок.

Сучасні інтелектуальні системи, зокрема на основі нейронних мереж, активно використовуються для аналізу зображень у радіології. Наприклад, Google Health [10] розробила систему для виявлення ознак раку молочної залози на мамографічних зображеннях. Її точність діагностики часто перевищує рівень деяких досвідчених лікарів-рентгенологів. Дослідження, опубліковане в Nature [11], описало навчання штучного інтелекту на наборах даних мамографії понад 28000 жінок у США та Великобританії. ШІ використовував три моделі глибокого навчання, щоб отримати оцінку від нуля до одиниці, що вказує на ймовірність раку. Аналіз ШІ призвів до значно меншої кількості хибнопозитивних і негативних результатів, ніж аналіз рентгенологів. Але були також випадки, коли радіологи правильно виявляли рак, який штучний інтелект пропускав.

Система DeepMind [12] від Google використовується для аналізу зображень сітківки ока з метою раннього виявлення захворювань, що можуть призводити до втрати зору, таких як діабетична ретинопатія. Це ускладнення діабету виникає через пошкодження судин сітківки і може призвести до сліпоти, якщо не виявити його вчасно. DeepMind аналізує зображення сітківки, зокрема отримані за допомогою оптичної когерентної томографії, і використовує алгоритми глибокого навчання для точного виявлення ранніх ознак хвороби. Система здатна ідентифікувати навіть незначні зміни в структурах судин і тканин, що дозволяє швидко поставити діагноз та розпочати лікування. Це допомагає запобігти прогресуванню хвороби та зберегти зір пацієнта.

Aidoc [13] – це компанія, яка спеціалізується на розробці штучного інтелекту для медичної візуалізації. Її технології допомагають лікарям швидше і точніше аналізувати медичні зображення, такі як комп'ютерна томографія (КТ) та магнітно-резонансна томографія (МРТ). Алгоритми Aidoc використовуються для автоматичного виявлення критичних патологій, зокрема інсультів, внутрішніх кровотеч, легеневих емболій та інших серйозних станів. Це дозволяє прискорити

діагностику та лікування пацієнтів, зменшуючи час очікування і підвищуючи точність медичних висновків.

Aidoc інтегрує свої рішення в робочі процеси радіологів, допомагаючи їм в реальному часі обробляти великі обсяги даних і забезпечувати кращий догляд за пацієнтами. Компанія також співпрацює з багатьма медичними установами по всьому світу, підтримуючи інновації в галузі охорони здоров'я.

Viz.ai [14] – це компанія, яка розробляє штучний інтелект для автоматизації та покращення діагностики гострих медичних станів, таких як інсульт. Їх платформа використовує ШІ для аналізу медичних зображень (наприклад, КТ або МРТ) у реальному часі, виявляючи серйозні патології та миттєво повідомляючи про це лікарів. Основна мета Viz.ai – прискорити процес виявлення гострих станів, зокрема ішемічного інсульту, і поліпшити комунікацію між медичними фахівцями, щоб пацієнти могли швидше отримати необхідне лікування.

Система Viz.ai використовує алгоритми штучного інтелекту для автоматичного виявлення блокувань судин у мозку, що дозволяє лікарям швидше реагувати і приймати рішення щодо лікування. Платформа також підтримує координацію між різними медичними установами та лікарями, полегшуючи передачу інформації та пришвидшуючи доступ до критично важливої допомоги. Viz.ai активно використовують у багатьох лікарнях по всьому світу для покращення результатів лікування пацієнтів з інсультами та іншими невідкладними станами.

Окрім існуючих систем також проводять різні дослідження щодо використання ШІ у медицині. Так наприклад у дослідженні [15] було розроблено два набори даних для покращення управління пацієнтами з гострим ішемічним інсультом. Дослід було зосереджено на пацієнтах, у яких були ознаки інсульту за системою FAST і які потребували тромболітичного лікування. Для забезпечення ефективного спостереження за лікуванням персоналу потрібні були доступні дані пацієнтів та безперебійна комунікація, але існуюча система не повністю задовольняла ці потреби через розрив між паперовими та електронними системами.

В рамках дослідження було проведено аналіз та інтерв'ю з усіма зацікавленими сторонами, які працюють із системою FAST track, та враховано всі протоколи лікування. На основі клінічних практичних рекомендацій було розроблено два набори даних: перший – спрощена модель для часткового покращення процесу в лікарнях, другий – для інтеграції з мобільними додатками та електронною передачею даних.

У статті [16] досліджено метод використання моторного уявлення на основі інтерфейсу мозок-комп'ютер для реабілітації пацієнтів після інсульту. Моторне уявлення дозволяє контролювати зовнішні пристрої та використовувати нейрофідбек для відновлення рухових функцій. Дослідження порівнювало продуктивність класифікації моторного уявлення пальців у пацієнтів з інсультом та здорових осіб. Використовувалися сигнали електроенцефалографії від одинадцяти здорових добровольців та одинадцяти пацієнтів з інсультом. Вони виконували завдання на постукування пальцями під час моторного виконання та уявлення руху, причому здорові учасники використовували домінантну руку, а пацієнти – уражену.

Дослідження показало, що середня точність класифікації моторного уявлення у пацієнтів з інсультом була значно вищою (66.00%) у порівнянні зі здоровими учасниками (46.94%). Ці результати свідчать про потенціал використання моторного уявлення для контролю зовнішніх пристроїв у реабілітації пацієнтів після інсульту, що може значно покращити їх відновлення та функціональність.

Автори статті [17] розглянули застосування методів глибокого навчання та штучного інтелекту для автоматизації виявлення інсульту, діагностики та роботизованої реабілітації пацієнтів. Автори досліджують ключові аспекти: збір даних про інсульт, методи попередньої обробки, а також ШІ-підходи для допомоги в реабілітації. Було проаналізовано 130 наукових робіт, що охоплюють ці напрямки. Окрім огляду, стаття виділяє три ключові виклики для подальших досліджень у галузі автоматизації виявлення інсульту та реабілітації пацієнтів.

Отже, існуючі системи переважно зосереджені на аналізі медичних зображень для виявлення конкретних патологій, таких як інсульт, крововиливи чи аневризми. Але кожен пацієнт має унікальну клінічну картину, яка може включати симптоми або фактори ризику, не завжди видимі на медичних зображеннях. Наприклад, пацієнти можуть мати головний біль, втрату свідомості, артеріальну гіпертензію або спадкові фактори, які значно впливають на ризики та хід нейросудинних захворювань. Автоматизовані системи, які враховують тільки зображення, можуть упустити ці важливі аспекти. Додавання симптомів дозволить створити більш точну та персоналізовану модель діагностики.

1.4 Постановка задачі

Загальна проблематика питання полягає в:

− необхідності підвищення точності діагностики нейросудинних захворювань, таких як ішемічний, геморагічний інсульт та аневризми головного мозку;

− розробці методів автоматизованого виявлення та оцінки ступеня пошкодження мозкової тканини за допомогою медичних зображень (КТ).

У випадку даної роботи основна увага приділяється розробці інтелектуальної системи діагностики нейросудинних захворювань, яка базується на використанні нейронних мереж. Проблема підходу до вирішення задачі полягає у створенні максимально точної моделі, яка не тільки ідентифікує наявність патологій, але й оцінює ступінь пошкоджень мозкової тканини, допомагаючи лікарям приймати рішення щодо подальшого лікування пацієнта.

Тобто, основна проблема поставленої задачі полягає в побудові архітектури системи, яка здатна:

− розпізнавати різні типи нейросудинних захворювань (ішемічні та геморагічні інсульти, аневризми);

− оцінювати ступінь пошкоджень мозкової тканини за допомогою сегментації.

Актуальність теми дослідження. Нейросудинні захворювання, зокрема інсульти, є однією з основних причин смертності та інвалідності у світі. Своєчасна діагностика і оцінка ступеня ураження мозкової тканини є критично важливими для успішного лікування. Традиційні методи діагностики, що базуються на візуальному огляді медичних зображень лікарем, можуть бути суб'єктивними і займати багато часу. Тому актуальним є розвиток автоматизованих систем на базі штучного інтелекту, які дозволяють значно підвищити точність та швидкість діагностики.

Об'єктом дослідження є процес діагностики нейросудинних хвороб за допомогою медичних зображень (КТ).

Предмет роботи є методи розробки нейронних мереж для аналізу медичних зображень та автоматизації процесу діагностики нейросудинних захворювань.

Мета роботи є діагностування нейросудинних хвороб за допомогою нейронних мереж.

Для досягнення поставленої мети необхідно виконати такі завдання:

− аналіз предметної області та сучасного стану задачі діагностики нейросудинних захворювань за допомогою медичних зображень;

− огляд існуючих методів та підходів до автоматизації діагностики на основі нейронних мереж;

− проектування архітектури нейронної мережі для виявлення інсультів та аневризм;

− розробка та тестування системи, здатної не тільки виявляти наявність патології, але й оцінювати ступінь пошкодження мозкової тканини.

Висновки до розділу 1

У цьому розділі було проведено огляд нейросудинних захворювань, їх типи та симптоми. Розглянуто основні методи діагностики нейросудинних хвороб, включаючи традиційні підходи та сучасні інструменти візуалізації, такі як КТ та МРТ. Також проведено аналіз сучасних інтелектуальних систем, які

застосовуються в медицині для автоматизації діагностичних процесів, зокрема, систем на основі нейронних мереж.

У рамках постановки задачі визначено основні вимоги до створення інтелектуальної системи діагностики нейросудинних захворювань з використанням нейронних мереж, що включає ідентифікацію патологій та оцінку ступеня пошкодження мозкової тканини.

2 МЕТОДИ ТА ІНСТРУМЕНТИ РОЗРОБКИ НЕЙРОННИХ МЕРЕЖ ДЛЯ ДІАГНОСТУВАННЯ НЕЙРОСУДИННИХ ХВОРОБ

2.1 Огляд методів машинного навчання

Методи машинного навчання в медицині дозволяють аналізувати та інтерпретувати великі обсяги даних для діагностики, прогнозування й моніторингу захворювань. Основні методи включають класифікацію (для виявлення типів захворювань), регресію (для прогнозування розвитку хвороби), глибоке навчання (для аналізу зображень, таких як КТ і МРТ), кластеризацію (для групування пацієнтів за схожими ознаками) і Байєсівські моделі (для оцінки ризиків). Кожен з цих підходів підвищує точність та швидкість медичних рішень, сприяючи персоналізованому підходу до лікування.

Метод **класифікації** [18] в машинному навчанні є одним із основних підходів для вирішення діагностичних завдань у медицині. Класифікація використовується для автоматичного визначення наявності чи відсутності захворювань, а також для їх категоризації за типами, що особливо корисно для виявлення конкретних патологій за медичними зображеннями, аналізом крові чи генетичними даними. В медичній діагностиці вона допомагає скоротити час постановки діагнозу, мінімізувати вплив людського фактора та підвищити точність діагностичних процесів, що в результаті сприяє своєчасному наданню необхідної медичної допомоги.

Регресія [19] – це метод машинного навчання, який передбачає знаходження залежності між вхідними змінними та цільовою змінною, щоб робити точні передбачення на основі нових даних. На відміну від класифікації, де результати поділяються на певні категорії, регресія використовується, коли цільова змінна є неперервною (наприклад, передбачення рівня кров'яного тиску, віку, ваги тощо).

Глибоке навчання (Deep Learning) стало важливим інструментом у сучасній медичній діагностиці завдяки своїй здатності виявляти приховані патерни в складних наборах даних, зокрема в медичних зображеннях. Основна особливість

глибокого навчання полягає у використанні багатошарових нейронних мереж, які дозволяють автоматично виокремлювати важливі особливості в зображеннях без потреби в ручній обробці [20]. Глибокі нейронні мережі, особливо згорткові нейронні мережі (CNN), здатні обробляти та аналізувати рентгенівські, МРТ та КТ-зображення, значно покращуючи точність та швидкість діагностики.

Переваги глибокого навчання у медичній діагностиці:

− глибокі нейронні мережі автоматично виокремлюють ознаки на різних рівнях, від базових (краї, форми) до більш складних патернів (структурні зміни та патології). Це знижує необхідність ручного налаштування параметрів, що було типовим для традиційних підходів;

− глибоке навчання забезпечує високу точність у виявленні патологій навіть при незначних змінах у зображеннях. Вони досягають високих показників чутливості та специфічності, що є особливо важливим у медицині, де помилковий діагноз може мати критичні наслідки для пацієнта;

− глибокі нейронні мережі легко масштабуються і можуть обробляти величезні обсяги даних, що особливо корисно для великих медичних закладів з великим потоком пацієнтів. Системи, засновані на глибоких нейронних мережах, можуть швидко аналізувати сотні зображень, що значно прискорює процес діагностики та знижує навантаження на медичний персонал;

− однією з найважливіших переваг глибокого навчання є здатність нейронних мереж до високоточної сегментації та класифікації. Це особливо актуально для завдань, пов'язаних із виділенням уражених ділянок, таких як пухлини, аневризми або ішемічні зони, що в подальшому допомагає лікарям визначити ступінь ураження;

− можуть бути навчені на великій кількості історичних даних для прогнозування розвитку захворювань та оцінки ризику на основі стану пацієнта. Це надає змогу створювати системи, які не лише діагностують захворювання, але й підтримують лікарів у прийнятті клінічних рішень;

– моделі глибокого навчання можуть адаптуватися до нових даних і покращувати свої результати з часом. Це дає можливість системам самонавчатися на основі нових зображень, покращуючи точність та актуальність їх прогнозів і діагнозів.

Нейронні мережі – це моделі машинного навчання, які імітують роботу людського мозку, обробляючи дані за допомогою штучних нейронів. Вони складаються з кількох шарів: вхідного, прихованих і вихідного [20].

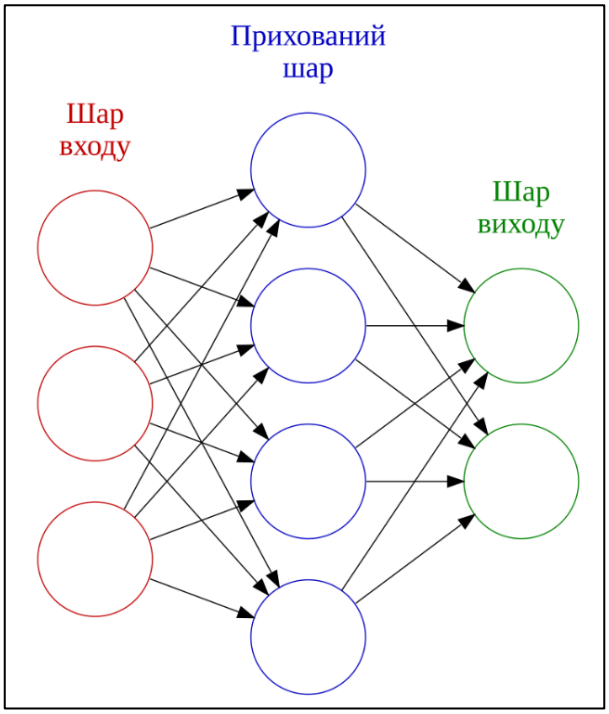

Рисунок 2.1 – Схема штучної нейронної мережі

Вхідні шари: це шар, на якому ми вводимо дані для моделі. Кількість нейронів у цьому шарі дорівнює загальній кількості ознак у наших даних (кількості пікселів у випадку зображення).

Прихований шар: вхідні дані з вхідного шару потім подаються в прихований шар. Може бути багато прихованих шарів залежно від моделі та розміру даних. Кожен прихований шар може мати різну кількість нейронів, яка зазвичай перевищує кількість функцій. Вихідні дані кожного шару обчислюються шляхом множення матриці вихідних даних попереднього шару на вагові коефіцієнти цього

шару, які можна дізнатися, а потім шляхом додавання зміщень, які можна дізнатися, з наступною функцією активації, яка робить мережу нелінійною.

Рівень виводу: вихідні дані з прихованого шару потім подаються в логістичну функцію, наприклад sigmoid або softmax, яка перетворює вихідні дані кожного класу в оцінку ймовірності кожного класу.

Згорткові нейронні мережі (Convolutional Neural Networks, CNN) – це тип нейронних мереж, спеціально розроблений для обробки зображень і просторових даних. Вони використовують принципи лінійної алгебри, зокрема операції згортання, щоб витягувати ознаки та ідентифікувати шаблони в зображеннях. Хоча CNN переважно використовуються для обробки зображень, їх також можна адаптувати для роботи з аудіо та іншими сигнальними даними [22].

CNN зазвичай складається з кількох рівнів, які можна розділити на три групи: згорткові шари, шари об’єднання та повністю зв’язані шари. Коли дані проходять через ці рівні, складність CNN зростає, що дозволяє CNN послідовно ідентифікувати більші частини зображення та більш абстрактні елементи.

Згортковий рівень (рис. 2.2) є основним будівельним блоком CNN, на якому виконується більшість обчислень. Цей рівень використовує фільтр або ядро – невелику матрицю вагових коефіцієнтів – для переміщення по сприйнятливому полю вхідного зображення для виявлення наявності певних особливостей.

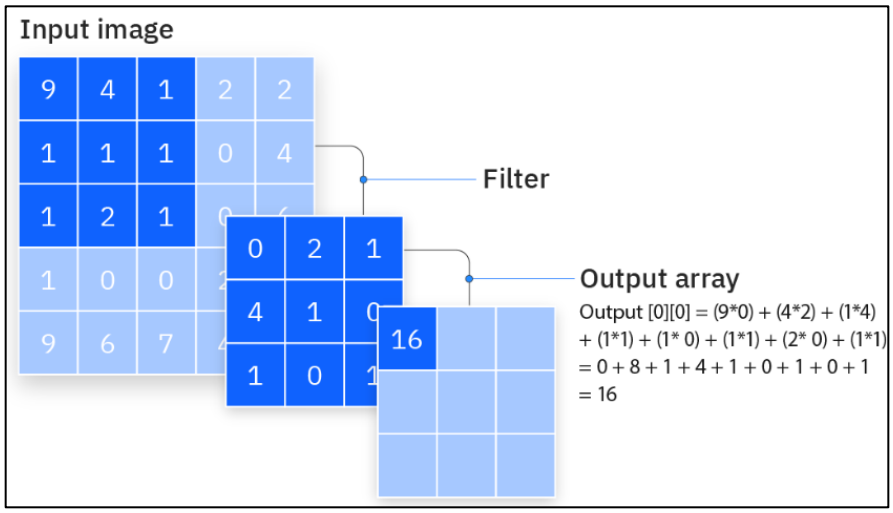

Рисунок 2.2 – Згортковий шар

Процес починається з переміщення ядра по ширині та висоті зображення, зрештою проходячи по всьому зображенню протягом кількох ітерацій. У кожній позиції обчислюється скалярний добуток між вагами ядра та значеннями пікселів зображення під ядром. Це перетворює вхідне зображення на набір карт функцій або згорнутих функцій, кожна з яких представляє присутність та інтенсивність певної функції в різних точках зображення.

CNN часто включають кілька складених згорткових шарів. Завдяки цій багатошаровій архітектурі CNN поступово інтерпретує візуальну інформацію, що міститься в необроблених даних зображення. На попередніх рівнях CNN визначає основні функції, такі як краї, текстури або кольори. Глибші шари отримують вхідні дані від карт властивостей попередніх шарів, що дозволяє їм виявляти складніші візерунки, об’єкти та сцени.

Шар об'єднання CNN є критичним компонентом, який слідує за згортковим рівнем. Рівень об’єднання має на меті зменшити розмірність вхідних даних, зберігаючи при цьому важливу інформацію, таким чином покращуючи загальну ефективність мережі. Зазвичай це досягається за допомогою зменшення дискретизації, тобто зниження кількості точок даних.

Для CNN це зазвичай означає зменшення кількості пікселів, які використовуються для представлення зображення. Найбільш поширеною формою є максимальне об’єднання (рис. 2.3), яке зберігає максимальне значення в певному вікні (тобто розмір ядра), відкидаючи інші значення.

Інший варіант – об’єднання за середнім, яке бере середнє значення замість максимального. Менш складні моделі з функціями вищого рівня, як правило, менш схильні до перенавчання, яке виникає, коли модель вивчає шум і надто конкретні деталі у своїх навчальних даних, що негативно впливає на її здатність бачити нову, невидиму інформацію. Незважаючи на можливу втрату деякої інформації, збереження головних характеристик зазвичай є достатнім для таких завдань, як розпізнавання об’єктів і класифікація.

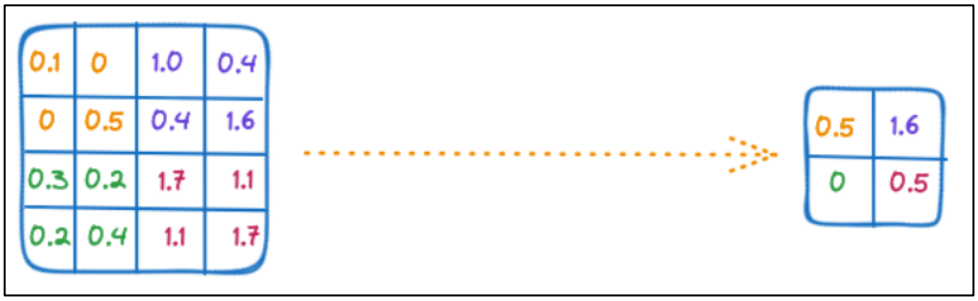

Рисунок 2.3 – Застосування максимального об’єднання з кроком 2 за допомогою фільтра 2x2

Повністю зв'язаний шар відіграє вирішальну роль на завершальних етапах CNN, адже він відповідає за класифікацію зображень на основі ознак, виділених на попередніх рівнях. Термін «повне підключення» означає, що кожен нейрон одного шару з’єднаний з кожним нейроном наступного шару.

Повністю зв’язаний рівень об’єднує різні функції, витягнуті на попередніх згорткових рівнях і рівнях об’єднання, і відображає їх у конкретних класах або результатах. Кожен вхід із попереднього рівня підключається до кожного блоку активації на повністю підключеному рівні, дозволяючи CNN одночасно враховувати всі функції під час прийняття остаточного рішення щодо класифікації.

Не всі рівні в CNN повністю пов’язані. Оскільки повністю зв’язані рівні мають багато параметрів, застосування цього підходу до всієї мережі створить непотрібну щільність, збільшить ризик перенавчання та зробить мережу дуже дорогою для навчання з точки зору пам’яті та обчислень. Обмеження кількості повністю пов’язаних шарів збалансовує ефективність обчислення та здатність до узагальнення з можливістю вивчення складних шаблонів.

На рисунку 2.4 зображено роботу згорткових нейронних мереж.

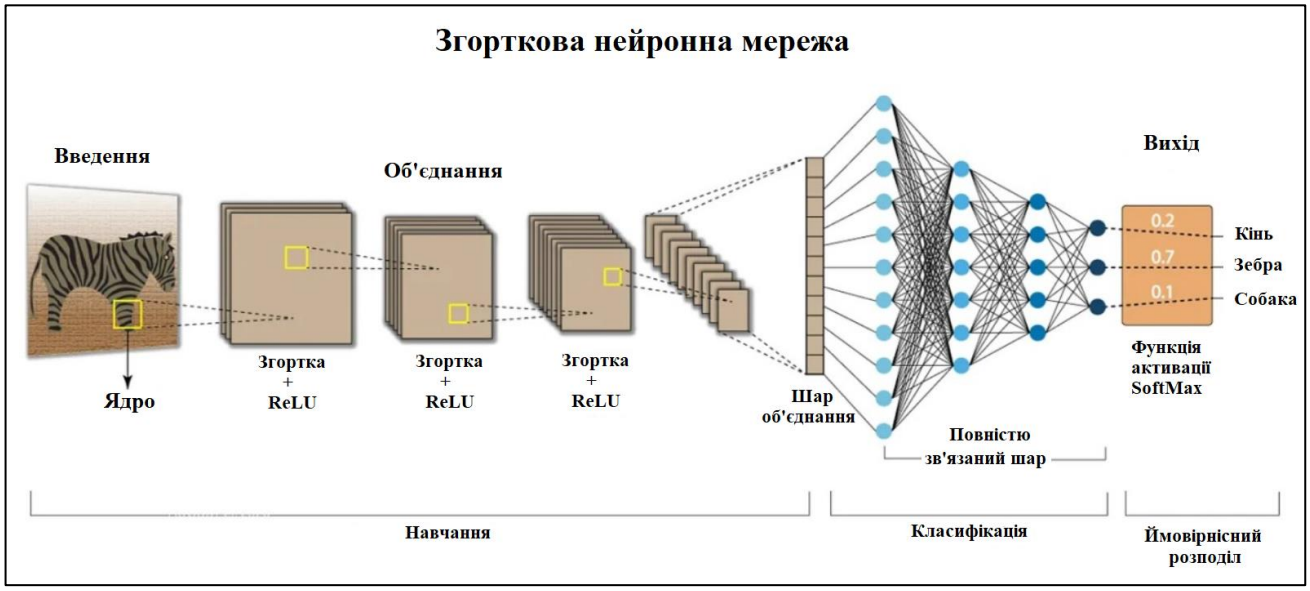

Рисунок 2.4 – Згорткова нейронна мережа

2.2 Технологічні засоби реалізації системи

2.2.1 Вибір мови програмування

Для розробки системи була використана мова програмування Python, а особливо бібліотеки Keras і TensorFlow, які є потужними інструментами для розробки та навчання нейронних мереж.

TensorFlow – це популярна бібліотека з відкритим кодом для машинного навчання та глибокого навчання, розроблена компанією Google. Вона надає потужні можливості для побудови, навчання та розгортання нейронних мереж, підтримуючи як прості, так і складні моделі.

TensorFlow [23] використовує оптимізовані алгоритми, що дозволяють швидко обробляти великі обсяги даних, що є критичним при обробці медичних зображень. Підтримує багато типів нейронних мереж, зокрема згорткові (CNN), рекурентні (RNN) та глибокі нейронні мережі (DNN). Можна легко адаптувати ці моделі під свої завдання, наприклад, для виявлення інсультів на КТ або МРТ зображеннях.

Keras [24] – це високорівнева бібліотека, яка працює поверх TensorFlow і надає зручний інтерфейс для розробки та тренування нейронних мереж. Вона

спрощує створення моделей і дозволяє швидко експериментувати з різними архітектурами. Keras має простий та інтуїтивно зрозумілий API, що дозволяє легко визначати нейронні мережі, а також обирати оптимальні методи для навчання та оцінки моделей. Завдяки своїй простоті, вона дозволяє швидко створювати прототипи моделей, що важливо на етапі розробки, коли потрібно тестувати кілька варіантів архітектур, таких як U-Net або VGG.

Також Keras надає доступ до великої кількості попередньо навчених моделей, що дозволяє заощадити час на навчання та підвищити точність системи, наприклад, за рахунок використання попередньо навчених моделей для класифікації зображень.

2.2.2 Аналіз архітектур нейронних мереж

Архітектури згорткових нейронних мереж – це структури, що визначають компонування та взаємозв'язки шарів у CNN, щоб вирішувати завдання комп'ютерного бачення. Існують різні типи архітектур CNN, такі як LeNet, AlexNet, VGG, ResNet, Inception та інші. Кожна з них має унікальні особливості.

Розглянемо більш детально архітектури для завдання класифікації паталогій.

AlexNet – це одна з найвідоміших архітектур згорткових нейронних мереж, яка була представлена в 2012 році командою на чолі з українцем Алексом Кріцевським на конкурсі ImageNet Large Scale Visual Recognition Challenge. Мережа стала революційною, оскільки значно перевершила результати попередніх моделей за точністю, вперше досягнувши значного прориву в обробці зображень.

AlexNet [25] складається з восьми шарів: п’яти згорткових і трьох повнозв’язних. Вона використовує функцію активації ReLU (Rectified Linear Unit), яка допомагає прискорити навчання порівняно з традиційними сигмоїдними функціями активації. Для запобігання перенавчанню в мережі застосовуються методи, такі як Dropout, що випадковим чином вимикає нейрони на етапах навчання, і також використовується нормалізація на вході до кожного шару. Важливою особливістю цієї мережі є використання техніки зменшення розміру

зображень, що сприяє зниженню обчислювальної складності та збереженню ключових ознак зображення (див. рис. 2.5).

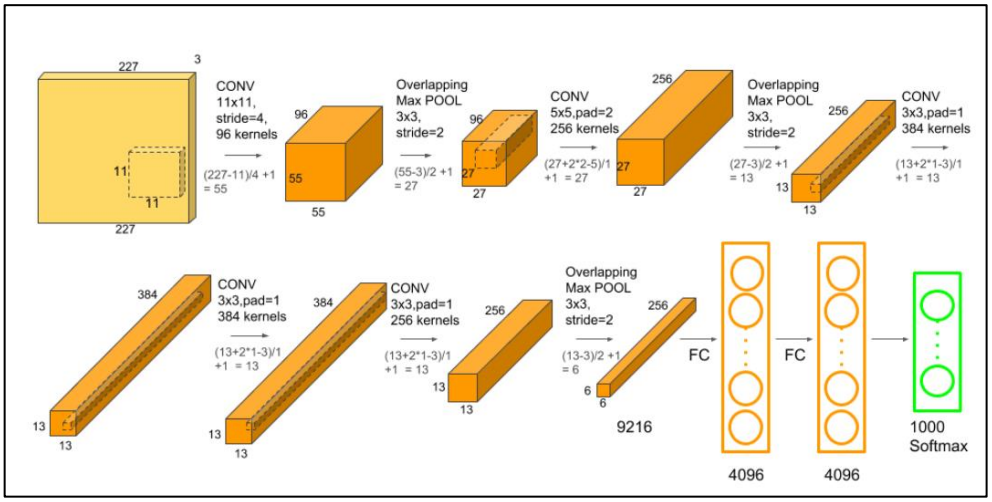

Рисунок 2.5 – Архітектура AlexNet

Переваги AlexNet:

– висока точність: завдяки глибинній архітектурі і вдосконаленим методам навчання, AlexNet продемонструвала значно кращі результати порівняно з попередніми моделями;

– швидке навчання завдяки використанню функції активації ReLU, яка значно прискорює процес навчання;

– використання Dropout: це допомогло знизити ризик перенавчання, що є важливим для досягнення високої узагальненої здатності моделі.

Недоліки AlexNet:

– висока потреба в обчислювальних ресурсах: для тренування AlexNet потрібні потужні апаратні засоби, що може бути обмеженням для деяких користувачів;

– мала гнучкість: хоча AlexNet показала хороші результати в обробці зображень, її архітектура не є найоптимальнішою для деяких специфічних завдань, що потребують більш гнучких та спеціалізованих мереж;

– обмеження у вирішенні: мережа використовує фіксований розмір вхідних зображень (227x227), що може бути проблемою при роботі з зображеннями різних розмірів без попереднього масштабування.

Отже, AlexNet була важливим кроком у розвитку згорткових нейронних мереж, однак для більш складних завдань та зображень сучасні моделі, такі як VGG, ResNet або Inception, зазвичай перевершують її за ефективністю та гнучкістю.

ResNet50 [26] – це одна з архітектур згорткових нейронних мереж, яка була представлена у 2015 році командою Microsoft Research на чолі з Кеєм Резайом. ResNet50 є частиною сімейства мереж ResNet (Residual Networks), яке стало революційним проривом в області глибоких нейронних мереж завдяки використанню концепції залишкових зв'язків (residual connections). Ці залишкові зв'язки дозволяють ефективно тренувати дуже глибокі мережі, дозволяючи уникнути проблеми згасаючого градієнта, що виникає при навчанні глибоких нейронних мереж (рис. 2.6).

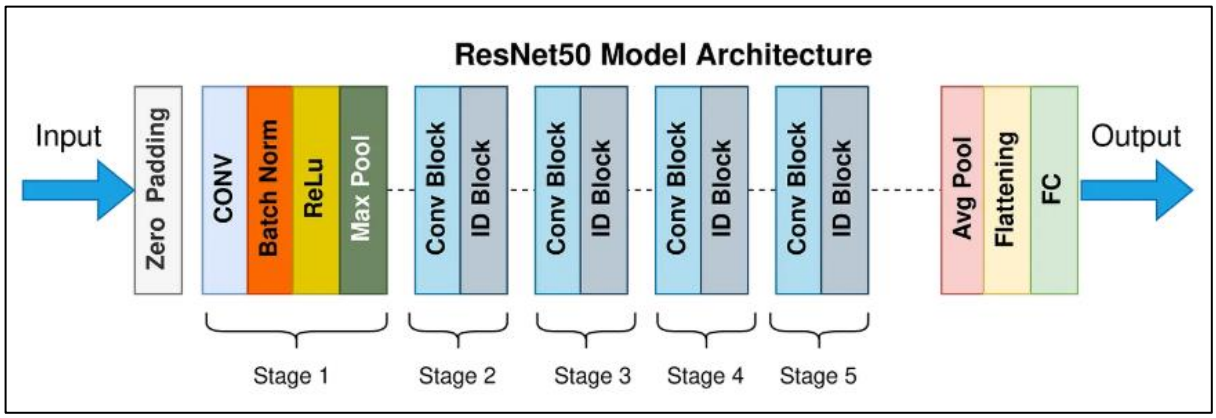

Рисунок 2.6 – Архітектура ResNet50

ResNet50 включає 50 шарів згорток, що дозволяє моделі вчити дуже складні патерни зображень. Це дозволяє моделі виявляти більш складні ознаки на різних рівнях абстракції. Заморожуючи інформацію на деяких етапах та додаючи її до наступних шарів, залишкові зв'язки дозволяють значно прискорити процес навчання, не стикаючись з проблемами згасаючого градієнта. Після основних

згорткових шарів використовується глобальний пулінг, що зменшує розміри карт ознак і готує їх до останнього шару класифікації. Вихід з глобального пулінгу проходить через один або кілька повнозв'язних шарів для визначення класу на основі ознак.

Переваги ResNet50:

− глибина архітектури: завдяки залишковим блокам можна створювати дуже глибокі мережі, що значно підвищує точність класифікації;

− ефективне навчання: залишкові зв'язки допомагають ефективно тренувати мережі навіть з великою кількістю шарів, що раніше було проблемою для традиційних мереж;

− висока точність: завдяки глибинній архітектурі і залишковим зв'язкам, ResNet50 досягає дуже високої точності на складних задачах комп'ютерного зору, таких як класифікація зображень на великих датасетах.

Недоліки ResNet50:

− висока вимогливість до ресурсів: через велику кількість шарів і складність моделі, ResNet50 потребує потужних апаратних засобів для навчання та прогнозування;

− час тренування: тренування такої глибокої мережі може бути дуже тривалим, навіть з використанням сучасних GPU;

− не завжди оптимальна для малих датасетів: як і багато інших глибоких нейронних мереж, ResNet50 може не працювати оптимально для малих та середніх за розміром датасетів без додаткового перенавчання або використання технік, таких як transfer learning.

Завдяки своїй глибині і зручності для перенавчання, ResNet50 є однією з найбільш використовуваних архітектур в області комп'ютерного зору на сьогодні.

VGG (Visual Geometry Group) була розроблена дослідницькою групою з Оксфордського університету і стала однією з найбільш впливових у галузі комп'ютерного зору. Модель VGG16, один із варіантів цієї архітектури, використовує 16 шарів із зваженими параметрами, а VGG19 – 19 шарів. Головною

особливістю VGG є послідовне застосування невеликих (3x3) згорткових ядер замість великих, що дозволяє краще виявляти особливості зображень та зберігати просторову інформацію. Мережа поступово зменшує розмір зображення через згорткові шари з наступним застосуванням об'єднання (зазвичай максимального). Після згорткових рівнів використовуються повністю зв’язані шари для остаточної класифікації зображень [27].

Переваги VGG:

− простота структури: використання 3x3 фільтрів робить VGG концептуально простою, що спрощує налаштування;

− стабільна продуктивність: вона забезпечує хорошу точність класифікації завдяки глибокій архітектурі;

− універсальність: модель можна використовувати для багатьох завдань комп'ютерного зору, таких як класифікація та сегментація зображень.

Недоліки VGG:

− високе споживання ресурсів: модель має значну кількість параметрів (до 140 млн), що робить її об’ємною і ресурсоємною;

− тривалий час навчання: через значну глибину і кількість параметрів, VGG потребує багато часу для навчання і оптимізації;

− обмежена ефективність для мобільних пристроїв: через великий обсяг обчислень модель складно розгорнути на малопотужних пристроях.

Загалом, VGG залишається популярною завдяки своїй точності, хоча сучасні моделі, такі як ResNet, демонструють подібні результати з меншою складністю.

Розглянемо архітектури для завдання сегментації паталогій.

SegNet [28] розроблена для завдань семантичної сегментації, особливо в комп’ютерному зорі для аналізу зображень. Основна особливість SegNet полягає у використанні шарів декодування, які відновлюють просторову роздільну здатність зображення, втрачену в процесі згортки та зменшення розмірності. Ця архітектура складається з пари шарів енкодера та декодера, що діють за принципом «вниз-вгору» та використовують значення з максимального пулінгу, отримані під

час етапу зниження розмірності, для точного відновлення зображення (рис.2.7). SegNet зберігає індекси з кожного шару пулінгу, що дозволяє відновити зображення в декодері без надмірного спотворення або втрати важливих деталей. Цей підхід допомагає уникнути надмірних обчислень і зберігає точність деталей, що робить SegNet ефективним для застосувань, де важлива точна сегментація, наприклад, в автомобільній галузі для розпізнавання дорожніх об’єктів або в медичній сфері.

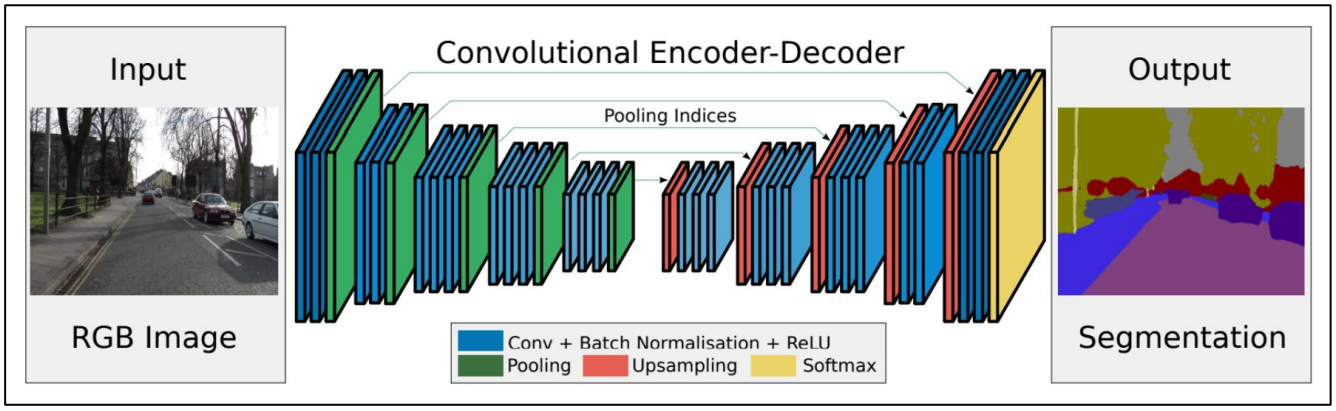

Рисунок 2.7 – Архітектура SegNet

Переваги SegNet:

– ефективне використання пам’яті: зберігання індексів пулінгу замість оригінальних даних зменшує вимоги до пам'яті;

– відновлення просторових деталей: завдяки зберіганню індексів з шарів пулінгу зменшується спотворення просторової структури об'єктів, що підвищує точність сегментації;

– швидше навчання: SegNet працює швидше на етапах тренування порівняно з іншими моделями сегментації, завдяки економії обчислень на декодувальному етапі.

Недоліки SegNet:

– менша точність, ніж в деяких інших моделях: для більш складних сегментаційних завдань SegNet може поступатися U-Net чи більш сучасним моделям;

− залежність від вхідного розміру: архітектура чутлива до розмірів вхідних зображень, що може вимагати попередньої обробки даних;

− обмежена здатність до генералізації: у порівнянні з більш глибокими мережами, такими як ResNet, SegNet іноді має обмежену здатність до генералізації на нових даних.

DeepLabV3 – це одна з сучасних архітектур для семантичної сегментації, розроблена для вирішення задач сегментації зображень із врахуванням контексту і деталізації. Вона базується на використанні атрозних згорток (dilated convolutions), які збільшують рецептивне поле моделі без втрати роздільної здатності. DeepLabV3 також використовує модуль ASPP (Atrous Spatial Pyramid Pooling), що дозволяє обробляти зображення на різних масштабах [29].

Переваги DeepLabV3:

− висока деталізація: використання атрозних згорток забезпечує збереження просторової роздільної здатності зображення, що важливо для точності сегментації;

− контекстна інформація: ASPP дозволяє моделі враховувати різні масштаби об'єктів, що підвищує точність сегментації, особливо на складних зображеннях;

− гнучкість: DeepLabV3 можна застосовувати як до великих зображень із високою роздільною здатністю, так і до малих зображень;

− сучасна архітектура: завдяки використанню моделі Xception як енкодера, DeepLabV3 демонструє високу продуктивність навіть на великих наборах даних.

Недоліки DeepLabV3:

− великі обчислювальні ресурси: архітектура потребує більше пам’яті та часу на обчислення через складні атрозні згортки та використання ASPP;

− проблеми на невеликих наборах даних: DeepLabV3 не так ефективна на малих наборах даних у порівнянні з U-Net;

− чутливість до гіперпараметрів: налаштування атрозних згорток і ASPP може бути складним і вимагає експериментів.

DeepLabV3 підходить для задач, де важлива обробка контекстної інформації та збереження деталізації. Вона часто використовується в задачах сегментації в автономному водінні, супутникових зображеннях та складних урбаністичних сценах.

U-Net [30] розроблена для задач сегментації зображень, особливо в медичній сфері. Вперше вона була представлена дослідниками з Університету Фрайбурга для сегментації клітинних структур на мікроскопічних зображеннях. Основна особливість U-Net – її симетрична структура у формі літери "U", де з одного боку проходить шлях "спуску" або згортування, а з іншого – шлях "підйому" або розгортання (рис. 2.8).

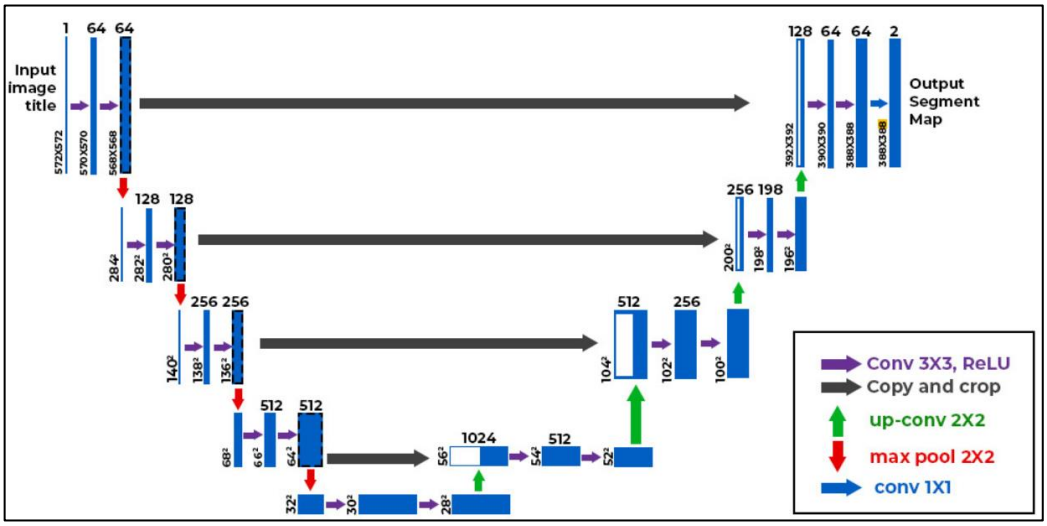

Рисунок 2.8 – Архітектура U-Net

Спочатку U-Net зменшує розмірність вхідного зображення, витягуючи ключові особливості через кілька шарів згорток і операцій зниження роздільної здатності. Потім відновлює зображення до початкового розміру, повертаючи деталі, втрачені під час згорток, за допомогою передачі інформації між рівнями на різних етапах обробки.

Це дозволяє U-Net отримувати як контекстну інформацію (на рівні великих особливостей), так і деталі (на рівні дрібних ознак) і точно відновлювати структуру об'єктів на зображенні. Завдяки цим характеристикам U-Net досягла значних

результатів у задачах медичної сегментації, де важливо виділяти навіть невеликі деталі.

Переваги U-Net:

− висока точність сегментації: завдяки своїй архітектурі U-Net ефективно виділяє об’єкти різного розміру та з високою точністю проводить сегментацію;

− ефективність для малих наборів даних: на відміну від багатьох інших мереж, U-Net показує хороші результати навіть на обмежених наборах даних, що є важливим для медичних досліджень;

− збереження дрібних деталей: використання пропусків між рівнями дозволяє уникати втрати деталей зображення, зберігаючи інформацію про текстуру та межі об'єктів.

Недоліки U-Net:

− високі вимоги до обчислювальних ресурсів: через велику кількість операцій згортування та розгортання U-Net може бути ресурсоємною, особливо для великих зображень;

− складність у налаштуванні: для специфічних задач необхідно коректно налаштувати гіперпараметри та можливі модифікації мережі, що може вимагати часу та експериментів;

− схильність до перенавчання: якщо не застосовувати регуляризацію та інші методи запобігання перенавчання, U-Net може запам'ятовувати тренувальні дані, що знижує її здатність узагальнювати.

U-Net є однією з найефективніших архітектур для сегментації зображень, особливо в медицині, де точність та можливість роботи з малими наборами даних є критичними факторами.

2.2.3 Вибір архітектур нейронних мереж

Для розробки системи необхідно обрати дві архітектури: для класифікації та для сегментації. Зробимо це за допомогою методу зважених оцінок [31]. Це простий багатокритеріальний метод прийняття рішень, який використовується для

порівняння альтернатив за кількома критеріями. Кожен критерій має вагу, яка визначає його важливість у порівнянні з іншими критеріями. Потім альтернативи оцінюються за кожним критерієм, і ці оцінки комбінуються, щоб отримати загальний рейтинг. Розрахунок відбувається за наступною формулою :

$$S_i = \sum_{j=1}^{n} w_j \times a_{ij}\,, \tag{2.1}$$

де S_i – загальна оцінка альтернативи i;

w_j – вага критерію j;

a_{ij} – оцінка альтернативи i за критерієм j.

Критерії класифікації представлені нижче.

1. Точність класифікації (Accuracy) – наскільки модель точна.
2. Обчислювальна ефективність (Efficiency) – швидкість роботи моделі.
3. Використання пам'яті (Memory Usage) – скільки пам'яті споживає модель.
4. Гнучкість (Flexibility) – наскільки модель підходить для адаптації до інших задач класифікації.

Ваги критеріїв наведені нижче.

1. Точність класифікації: 0.4
2. Обчислювальна ефективність: 0.3
3. Використання пам'яті: 0.2
4. Гнучкість: 0.1

Таблиця 2.1 – Оцінка архітектур

	Точність (1-10)	**Обчислювальна ефективність (1-10)**	**Використання пам'яті (1-10)**	**Гнучкість (1-10)**
AlexNet	6	9	8	7
ResNet50	10	7	6	9
VGG	8	6	4	8

Таблиця 2.2 – Результати

Архітектура	**Сума балів**
AlexNet	7.4
ResNet50	8.2
VGG	6.6

Критерії сегментації наведено нижче.

1. Точність сегментації (Accuracy) – точність сегментування цільові області.
2. Просторове відновлення (Spatial Recovery) – наскільки добре модель зберігає просторові деталі зображення.
3. Продуктивність (Computational Efficiency) – швидкість і потреби в обчислювальних ресурсах.
4. Здатність до генералізації (Generalization Ability) – наскільки добре модель працює на нових даних.

Ваги критеріїв.

1. Точність сегментації: 0.4.
2. Просторове відновлення: 0.25.
3. Продуктивність: 0.15.
4. Здатність до генералізації: 0.1.

Таблиця 2.3 – Оцінка архітектур

	Точність (1-10)	**Просторове відновлення (1-10)**	**Продуктивність (1-10)**	**Здатність до генералізації (1-10)**
SegNet	7	7	8	7
DeepLabV3	8	8	7	9
U-Net	9	9	8	8

Таблиця 2.4 – Результати

Архітектура	**Сума балів**
SegNet	7.2
DeepLabV3	8.1
U-Net	8.7

Висновки до розділу 2

Було розглянуто основні методи машинного навчання, зокрема нейронні мережі для виявлення патологій на медичних зображеннях. Найефективнішими є згорткові нейронні мережі, зокрема архітектури, що спеціалізуються на сегментації зображень, які дозволяють точно автоматизувати процес діагностики та оцінки ушкоджень мозкової тканини.

Проведено багатокритеріальний аналіз архітектур для сегментації медичних зображень, зокрема U-Net, DeepLabV3, та SegNet. За обраними критеріями було визначено, що найкращою архітектурою для сегментації КТ-знімків є U-Net, яка поєднує високу точність сегментації, здатність до відновлення просторових деталей та продуктивність.

Також проведено багатокритеріальний аналіз архітектур для класифікації зображень, зокрема AlexNet, VGG, та ResNet50. За результатами аналізу найкращою архітектурою для задачі класифікації зображень визначено ResNet50, завдяки високій точності та ефективності обчислень.

Для реалізації системи обрано TensorFlow та Keras, які є потужними інструментами для розробки нейронних мереж. TensorFlow підтримує масштабованість обчислень, а Keras полегшує створення та налаштування моделей, що робить їх оптимальним вибором для реалізації задач сегментації та класифікації медичних зображень.

3 МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДІАГНОСТУВАННЯ НЕЙРОСУДИННИХ ХВОРОБ

3.1 Опис структури системи

Інтелектуальна система діагностики нейросудинних захворювань базується на кількох ключових компонентах, які забезпечують її функціональність і продуктивність. Як і в будь-якій інформаційній системі, вона складається з таких основних елементів: апаратне та програмне забезпечення, мережеві комунікації, дані, люди та процеси. Всі ці компоненти взаємодіють, щоб створити єдину інтегровану систему, здатну виконувати завдання з автоматизованої діагностики.

Дані. Основу системи складають медичні дані, отримані зі знімків комп'ютерної томографії (КТ) головного мозку. Ці дані є центральним елементом, на основі якого здійснюється навчання моделі, діагностика та оцінка патологій. Знімки КТ включають зображення ішемічних і геморагічних інсультів, а також аневризм, які підлягають аналізу нейронною мережею.

Апаратне та програмне забезпечення. Апаратна частина системи включає потужні сервери для обробки великих обсягів даних, а також графічні процесори (GPU), які прискорюють навчання та виконання глибоких моделей нейронних мереж. Програмне забезпечення складається з бібліотек для побудови та тренування нейронних мереж, таких як TensorFlow або PyTorch, що дає змогу реалізувати згорткові нейронні мережі для аналізу зображень.

Мережеві комунікації. Для забезпечення безперервної роботи системи використовуються мережеві комунікації, які забезпечують обмін даними між різними компонентами системи, включаючи хмарні сервіси для зберігання та обробки інформації. Це також дозволяє інтегрувати систему з медичними установами для передачі даних пацієнтів та результатів діагностики.

Люди. Персонал, що працює із системою, включає лікарів, пацієнтів та інженерів. Лікарі використовують систему для отримання результатів автоматизованого аналізу знімків, а інженери налаштовують нейронні мережі та

підтримують технічну частину роботи системи. Пацієнти в свою чергу можуть використовувати систему для перевірки знімки КТ перед відвідуванням лікарні.

Процеси. Процеси у системі охоплюють збір, обробку та аналіз медичних даних, побудову та навчання нейронних мереж, тестування їхньої точності, а також виведення результатів діагностики. Ключовим процесом є побудова системи, яка здатна адаптуватися та навчатися на нових даних для покращення результатів діагностики.

Функціональні компоненти. Система включає кілька функціональних модулів, серед яких основним є модуль згорткових нейронних мереж, що аналізує медичні зображення для виявлення патологій. Окремі модулі відповідальні за передобробку зображень, нормалізацію даних, а також за виведення результатів у зручному для медичних працівників форматі.

3.1.1 Дерево функцій

Дерево рішень [33] системи для діагностики нейросудинних хвороб на основі КТ-зображень включає наступні компоненти:

а) збір та підготовка даних:

− збір КТ-зображень головного мозку з датасетів, що включають ішемічні та геморагічні інсульти, аневризми;

− попередня обробка даних, включаючи нормалізацію зображень, пригнічення шуму, зміну формату та розміру, розширення даних для збільшення різноманітності;

б) відображення та візуалізація зображень:

− відображення КТ-зображень для перегляду та аналізу лікарем;

− маркування зон патології для полегшення аналізу результатів системи, візуалізація меж уражених ділянок;

в) автоматична сегментація та класифікація:

− використання згорткових нейронних мереж (CNN), таких як U-Net, для автоматичної сегментації уражених зон (ішемія, крововиливи, аневризми);

– класифікація розпізнаних зон на основі моделей для діагностики типу хвороби;

г) оцінка ступеня пошкоджень:

– використання регресійних моделей для кількісної оцінки відсотка ураження мозкової тканини на основі сегментованих даних;

– оцінка обсягу уражених зон для клінічної інтерпретації;

д) інтеграція та розгортання системи:

– розробка користувацького інтерфейсу для зручного доступу до результатів сегментації та класифікації;

е) оцінка результатів та покращення:

– оцінка результатів системи на основі метрик точності, чутливості, специфічності для діагностики нейросудинних хвороб.

Схематично дерево функцій зображено на рисунку 3.1.

Рисунок 3.1 – Дерево функцій системи діагностування нейросудинних хвороб

3.1.2 IDEF0 та Use case

Система діагностування нейросудинних хвороб може бути описана за допомогою методу IDEF0 [32], що дозволяє представити її функціональні процеси в ієрархічній моделі (рис. 3.2). На першому рівні система отримує вхідні дані, такі як КТ-знімки, які підлягають попередній обробці. Далі зображення аналізуються

нейронними мережами для виявлення патологій, зокрема інсульту або аневризми. На останньому етапі система генерує діагностичні висновки для лікарів. Така модель дозволяє структурувати весь процес і забезпечити наочність взаємодії між компонентами системи.

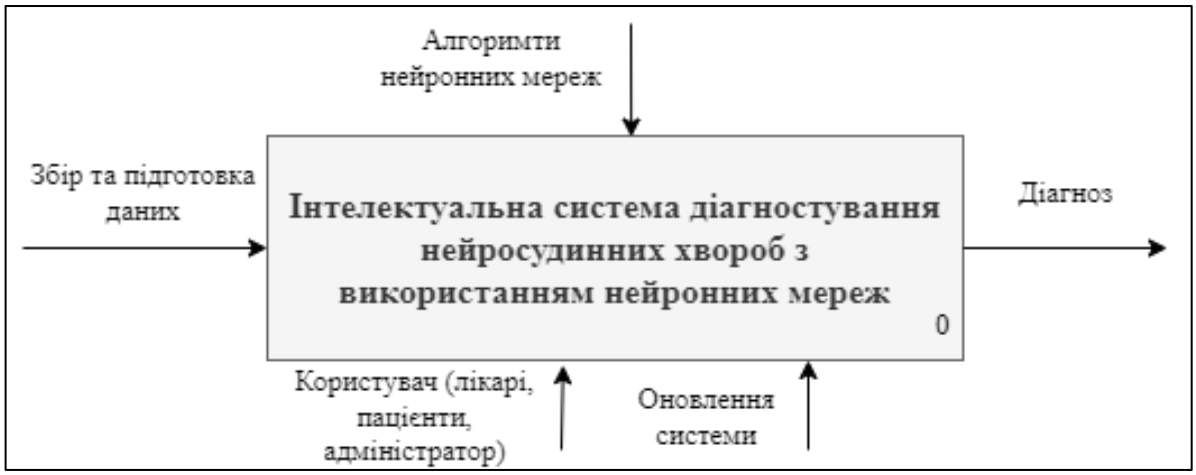

Рисунок 3.2 – Діаграма контексту блоку A0

Для більш детального опису створено діаграму IDEF0 для блоку A0 з підблоками A1, A2, A3 (рис. 3.3).

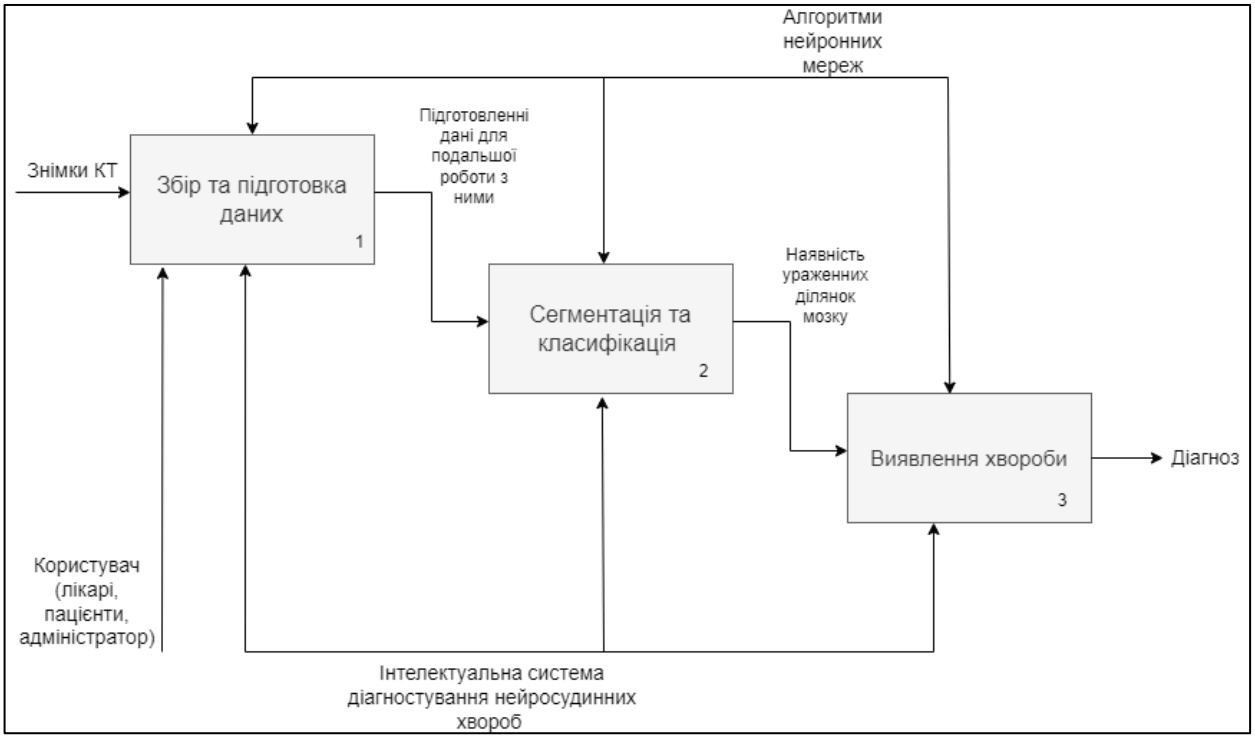

Рисунок 3.3 – Діаграма IDEF0 з підблоками

Таблиця 3.1 – Опис підблоків

Назва	Опис	Функції
A1: Збір та підготовка даних	Підблок відповідає за початкове збирання медичних КТ - зображень та їх підготовку до подальшої обробки. Першочергово користувач завантажує в систему знімки з медичних апаратів або електронних медичних записів. Потім виконується нормалізація зображень – приведення їх до єдиного формату та масштабу. Далі система проводить фільтрацію шумів і видаляє артефакти, щоб покращити якість зображення. Завершується підготовка корекцією контрасту та яскравості, щоб підвищити точність подальшого аналізу.	− завантаження зображень; − нормалізація та фільтрація шумів; − підготовка даних для подальшого аналізу (масштабування, вирівнювання).
A2: Сегментація та кваліфікація	Підблок виконує ключові функції аналізу зображень, починаючи з розподілу зображення на окремі сегменти, щоб виділити уражені ділянки судин. Сегментація дозволяє точно визначити межі різних областей мозку. Після цього система аналізує характеристики кожного сегмента та класифікує їх відповідно до заданих патологій, таких як ішемічний, геморагічний інсульт або аневризми.	− сегментація уражених ділянок; − класифікація патологій (інсульт, аневризма).
A3: Виявлення хвороби	Підблок відповідає за остаточне визначення наявності та характеру захворювання. На основі класифікації патологій система оцінює тяжкість стану, визначаючи відсоток уражених ділянок. Потім формується автоматичний діагноз, який включає тип нейросудинного захворювання, його стадію та потенційні наслідки для пацієнта. На основі аналізу система надає рекомендації щодо подальшого лікування.	− оцінка тяжкості ураження; − формування діагнозу на основі аналізу даних; − надання рекомендацій щодо лікування.

За проведеним функціональним аналізом можемо побудувати Use case діаграму (рис. 3.4). Вона моделює реальні сценарії використання системи, що допомагає створювати відповідні сценарії для перевірки працездатності системи та забезпечення її функціональності. Завдяки цьому можна виявити можливі проблеми та знайти шляхи оптимізації процесів.

Крім того, побудова Use case [34] сприяє кращому взаєморозумінню між всіма учасниками проєкту – як технічними фахівцями, так і медичними працівниками, пацієнтам, оскільки дозволяє всім сторонам узгодити очікування щодо роботи системи. Також підтримує масштабованість системи, даючи змогу виявити майбутні потреби, такі як додавання нових функцій або розширення системи на інші типи діагностик.

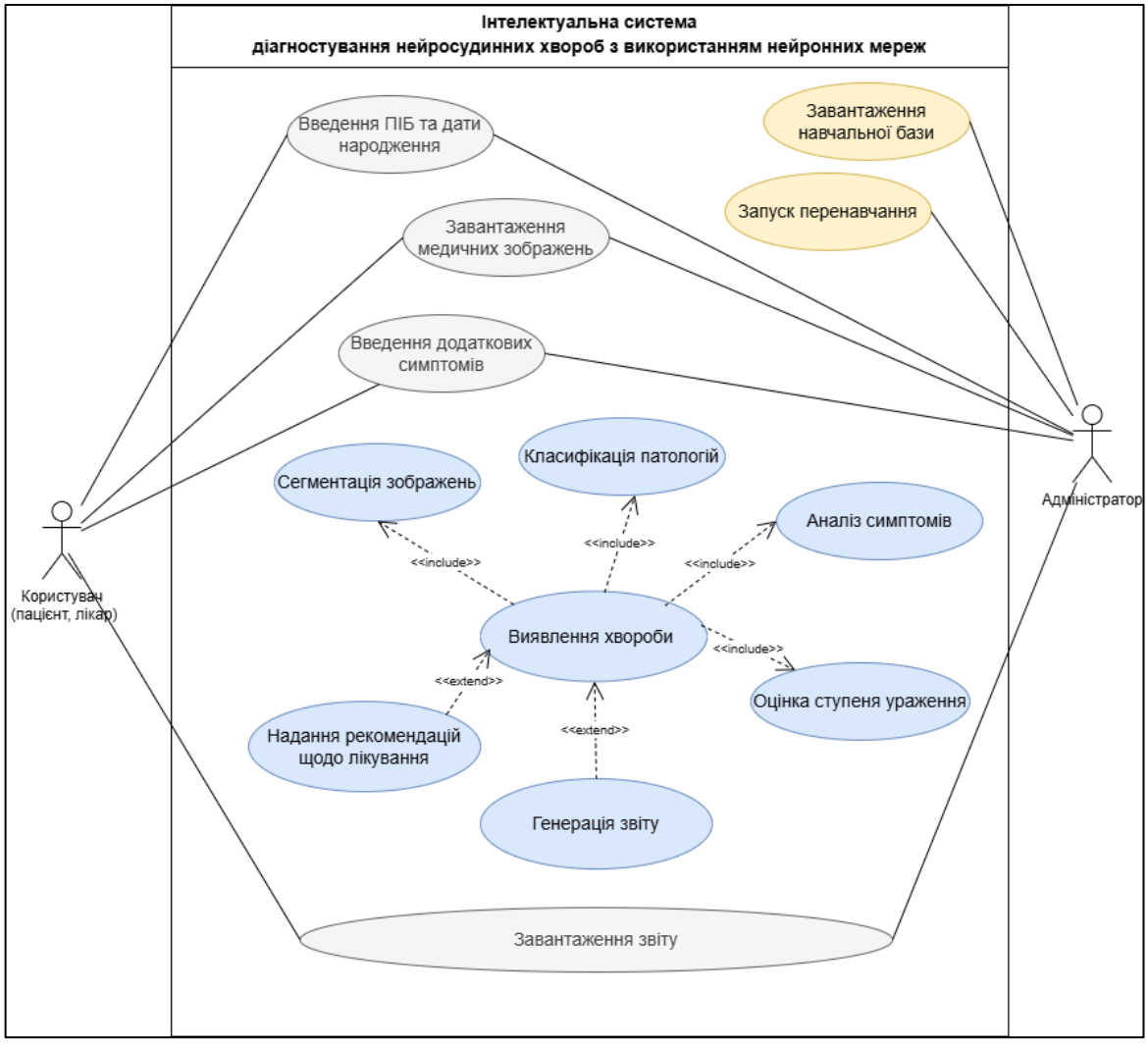

Рисунок 3.4 – Діаграма прецедентів системи

В таблиці 3.2 описано сценарій користування для користувача (лікар, пацієнт, адміністратор), який хоче перевірити наявність патологій на знімку КТ.

Таблиця 3.2 – Сценарій користування

Актор	Користувач, система
Ціль	Перевірка знімку КТ
Система	Інтелектуальна система діагностики
Передумови	Користувач має доступ до медичних зображень та до системи
Тригери	Користувач хоче дізнатися чи є на знімку КТ ознаки інсульту чи аневризми
Сценарій (основний)	1. Користувач вводить свої данні. 2. Користувач вибирає опцію "Завантажити знімок КТ". 3. Система надає можливість вибрати файл з локального комп'ютера або з електронних медичних записів. 4. Користувач обирає потрібний знімок КТ та підтверджує його завантаження. 5. Далі користувач вводить додаткові симптоми. 6. Після завантаження знімка система автоматично починає процес підготовки зображення. 7. Система аналізує сегментовані ділянки на предмет наявності ознак інсульту або аневризми. 8. Після завершення аналізу система формує результати перевірки. 9. Користувач отримує повідомлення про наявність або відсутність ознак інсульту чи аневризми. 10. У разі виявлення патологій, система також надає рекомендації щодо подальших дій. 11. Користувач переглядає результати та виходить із системи. 12. Користувач має змогу зберегти результати.
Сценарій (альтернативний)	1. Зображення не завантажується або є некоректним форматом. 2. Система видає повідомлення про помилку. 3. Користувач отримує вказівки щодо підтримуваних форматів та повторює спробу завантаження.

3.1.3 Діаграма послідовностей

Діаграма послідовностей [35] демонструє взаємодію між основними компонентами системи діагностики нейросудинних захворювань. Вона відображає порядок виконання операцій, а також передачу даних між учасниками системи (акторами та модулями). На ній зображено взаємодії між користувачем, системою, модулем класифікації, модулем сегментації та модулем аналізу симптомів (див. рис. 3.5).

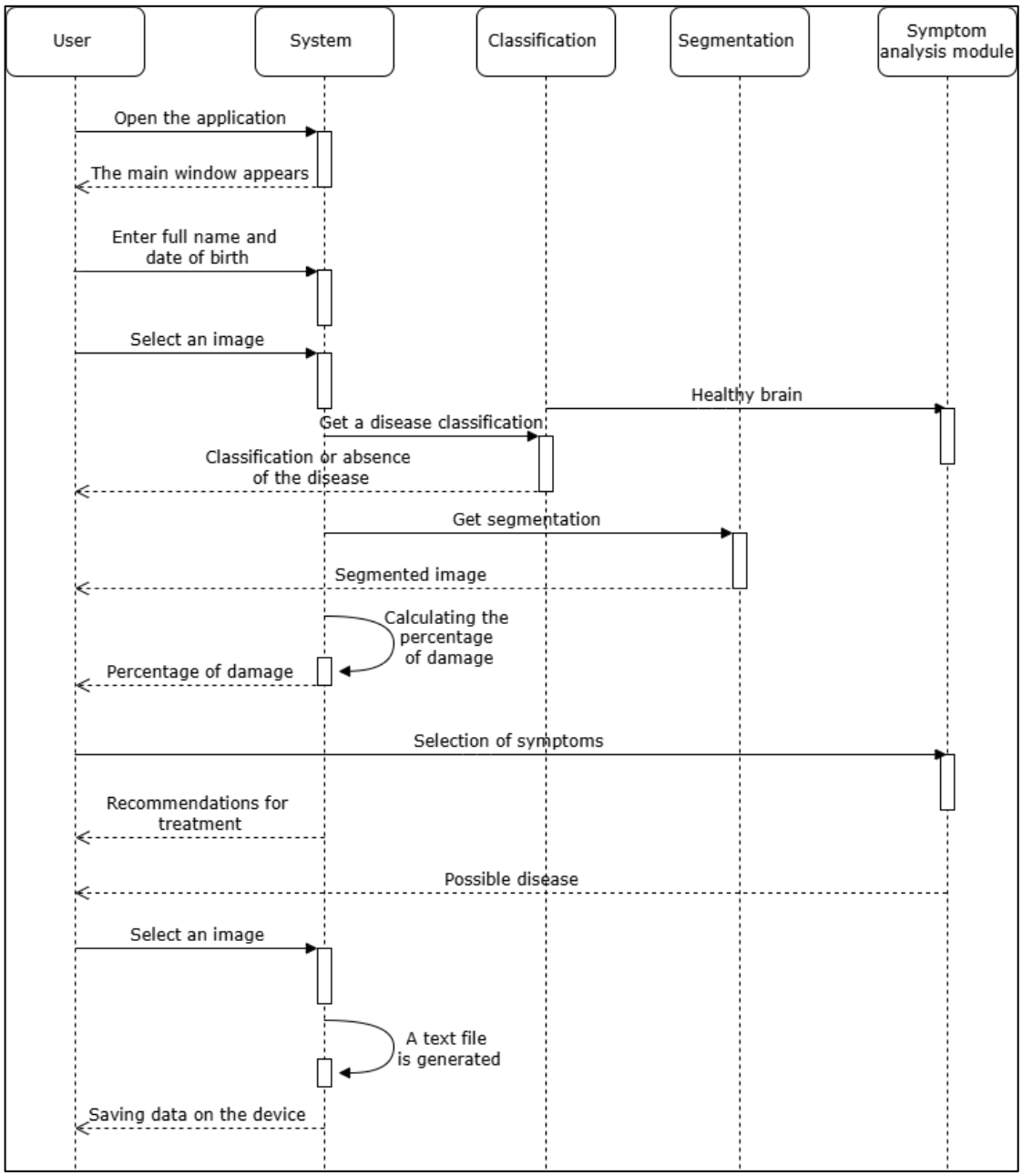

Рисунок 3.5 – Діаграма послідовностей

Ініціація процесу: користувач вводить ПІБ, завантажує зображення комп'ютерної томографії (КТ) та вводить симптоми через інтерфейс системи.

Обробка зображень:

− система передає завантажені дані до модуля класифікації;

− модуль класифікації аналізує зображення КТ, щоб визначити ймовірну патологію (ішемічний інсульт, геморагічний інсульт, аневризму) або підтвердити здоровий стан.

Сегментація зображень:

− у випадку виявлення патології модуль сегментації виділяє уражені ділянки на зображенні;

− результати сегментації повертаються до системи для включення у фінальний висновок.

Аналіз симптомів:

− якщо класифікація виявляє невідповідності або якщо користувач вводить симптоми, система звертається до модуля аналізу симптомів;

− модуль оцінює введені симптоми, щоб підтвердити чи доповнити висновки, зроблені на основі зображень.

Формування результатів:

− система інтегрує результати класифікації, сегментації та аналізу симптомів;

− користувачу надається фінальний висновок, який включає діагноз, ймовірність патології, виділені зони ураження (за їх наявності) та рекомендації;

− користувач може зберегти результати на пристрої у вигляді текстового файлу.

3.2 Аналіз та попередня обробка набору даних

3.2.1 Загальний опис датасетів

Для тренування моделі необхідно великий набір даних, оскільки якість і точність навчання штучних інтелектуальних систем значно залежать від обсягу та

різноманітності доступних даних. На платформі Kaggle представлено величезну кількість датасетів зі знімками КТ головного мозку, що були зібрані для аналізу та вивчення інсультів. Більшість із цих наборів даних просто відображає наявність інсульту, без поділу на його типи. Однак датасет [36] містить зразки знімків як ішемічного, так і геморагічного інсультів. Загальна кількість файлів у цьому датасеті становить 7324, що є досить великим обсягом даних для тренування моделі. Різноманітність представлених зображень, які показують різні розрізи мозку, дозволяє моделі вчитися на великій кількості інформації. Це, в свою чергу, підвищує ефективність навчання та точність виявлення захворювання.

Набір складається з двох папок:

− тренувальна (train) – розміром 159 МБ, складається із знімків формату .jpg мозку здорової людини (3322 файла), з ішемічним інсультом (395 файлів) та геморагічним (2113 файлів);

− тестувальна (test) – розміром 55 МБ, складається із знімків формату .jpg мозку здорової людини (783 файла), з ішемічним інсультом (135 файлів) та геморагічним (576 файлів).

Датасетів з аневризмами практично немає на Kaggle. Але вибірка [37], присвячений пухлинам, містить у собі папку із знімками аневризм. Розмір цієї папки становить всього 1,9 МБ і складається лише з 84 файлів.

Приклад частини датасету можна побачити на рисунку 3.6.

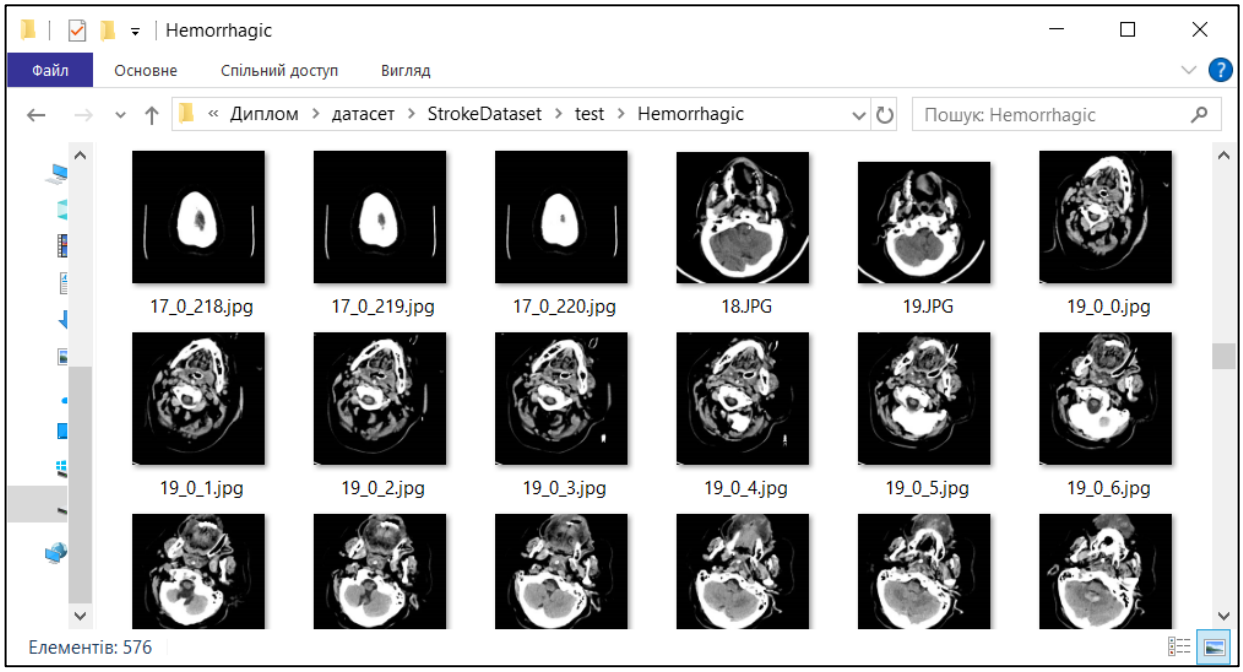

Рисунок 3.6 – Вибірка з видами інсульту

3.2.2 Попередня обробка набору даних

Попередня обробка даних важлива, оскільки вона покращує якість набору даних, видаляючи шуми та некоректні дані, що підвищує загальну якість. Вона також стандартизує формати даних, що полегшує їх аналіз і навчання моделей. Крім того, оптимізує обсяг даних, скорочуючи час навчання та використання пам'яті. Аугментація дозволяє збільшити навчальну вибірку, покращуючи стійкість моделі, а нормалізація приводить дані до одного масштабу, що сприяє ефективнішому навчальному процесу. В результаті, чистіші дані забезпечують більш точні та стабільні моделі.

Виявлення пошкоджених або відсутніх файлів.

Під час завантаження датасетів можуть виникати проблеми з файлами, наприклад, файли можуть бути пошкоджені або нечитабельні. Перевірка наявності та коректності зображень дозволяє ідентифікувати такі проблеми до початку обробки.

```
images = []
labels = []
for filename in os.listdir(dataset_path):
    if filename.endswith('.jpg') or filename.endswith('.png'):
        img_path = os.path.join(dataset_path, filename)
        img = cv2.imread(img_path)
        if img is None:
            print(f"Не вдалося завантажити зображення: {img_path}")
            continue
```

Рисунок 3.7 – Виявлення пошкоджених або відсутніх файлів

Аугментація.

Вибірка для ішемічного інсульту є недостатньою для ефективного навчання, порівняно з вибіркою для геморагічного. Щоб покращити цю ситуацію, було вирішено збільшити вибірку вручну, додавши файли з іншого датасету [38]. Це також допоможе моделі краще справлятися з різними варіаціями зображень, що можуть зустрічатися в реальному світі.

Наявність обмеженої кількості зображень у датасеті з аневризмами значно ускладнює процес навчання. З метою вирішення цієї проблеми, було використано платформу DeepAI [39], яка надає можливості для збільшення даних (рис.3.8). DeepAI підтримує різноманітні моделі, включаючи генеративні змагальні мережі (GAN), які здатні створювати нові зображення на основі наданих даних або текстових описів. Це дає можливість не лише збільшити розмір датасету, але й згенерувати варіації зображень, які можуть допомогти покращити загальну ефективність моделей у навчанні.

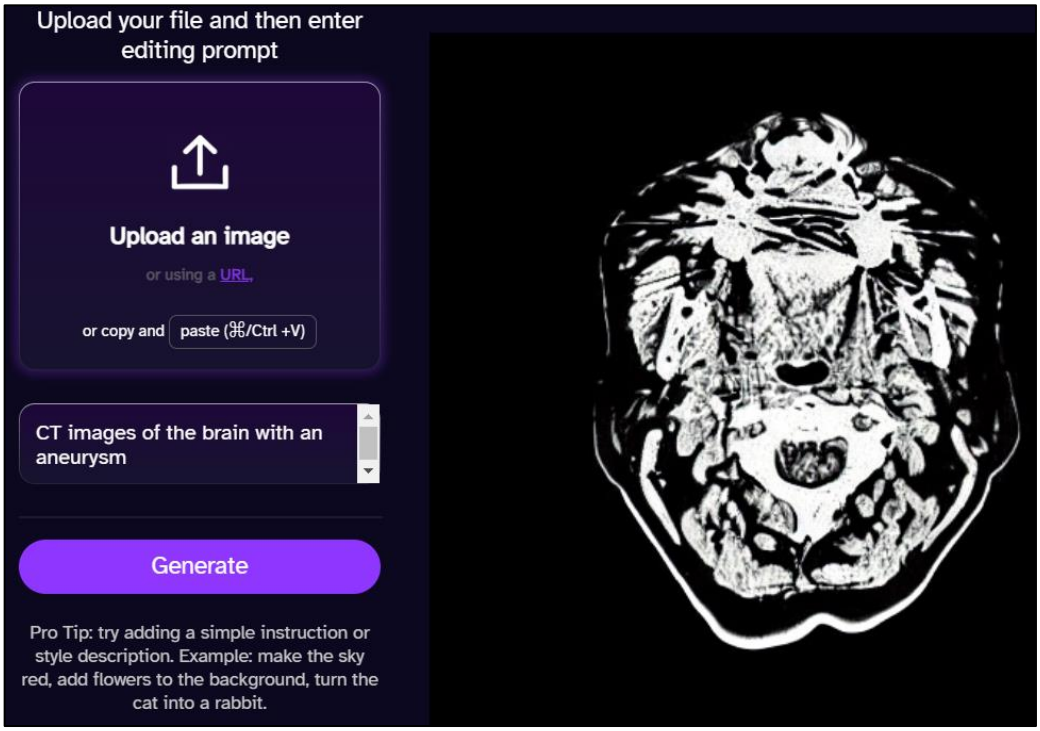

Рисунок 3.8 – Приклад використання DeepAI

Albumentations [40] – це популярна бібліотека для аугментації зображень, що широко використовується в задачах комп'ютерного зору та глибокого навчання. Вона була розроблена з акцентом на простоту використання, ефективність і можливість створення складних аугментацій. Вона підтримує різні формати зображень (включаючи NumPy масиви) і легко інтегрується з популярними фреймворками, такими як TensorFlow, PyTorch та Keras.

Вона буде використана для збільшення всіх вибірок.

Основні методи аугментації зображень:

– Rotate: випадковий поворот зображення на певний кут. Це дозволяє моделі бути стійкою до повороту об'єктів;

– HorizontalFlip: горизонтальне віддзеркалення зображення;

– VerticalFlip: вертикальне віддзеркалення;

– RandomBrightnessContrast: випадкова зміна яскравості та контрасту зображення. Це допомагає моделі вчитися в різних умовах освітлення;

– GaussianBlur: додавання розмиття до зображення. Це може допомогти моделі стати менш чутливою до шуму в зображеннях;

– RandomScale: випадкове масштабування зображення. Це може допомогти моделі навчитися розпізнавати об'єкти на різних відстанях;

– ShiftScaleRotate: зміщення, масштабування та поворот зображення. Це комбінована трансформація, яка дозволяє моделі адаптуватися до зміщених та змінених об'єктів.

```python
import albumentations as A
import cv2
import os

input_dir = 'D:/aneurysm'
output_dir = 'D:/augmented_images'
os.makedirs(output_dir, exist_ok=True)
# Аугментація
transform = A.Compose([
    A.Rotate(limit=30, p=0.5),
    A.HorizontalFlip(p=0.5),
    A.VerticalFlip(p=0.5),
    A.RandomBrightnessContrast(p=0.5),
    A.GaussianBlur(blur_limit=(3, 7), p=0.5),
    A.RandomScale(scale_limit=0.2, p=0.5),
    A.ShiftScaleRotate(shift_limit=0.1, scale_limit=0.1, rotate_limit=20, p=0.5),
])
```

Рисунок 3.9 – Приклад аугментації вибірки

Покращення якості даних.

Попередня обробка допомагає видалити шуми, непотрібні або некоректні дані, що можуть заважати навчанню. Це підвищує загальну якість даних.

```python
A.GaussianBlur(blur_limit=(1, 3), p=0.5),
A.ImageCompression(quality_lower=85, quality_upper=95, p=0.5),
```

Рисунок 3.10 – Приклад покращення якості даних

GaussianBlur – застосовується для розмиття зображення, що допомагає зменшити шум. Задано діапазон розмиття від 1 до 3 пікселів.

ImageCompression – зменшує артефакти стиснення, що може бути корисним для підвищення якості зображень після аугментації.

Уніфікація формату даних та проставлення міток.

Дані можуть бути отримані з різних джерел і можуть мати різні формати. Попередня обробка допомагає стандартизувати дані, що полегшує їх аналіз та навчання моделей.

Кожному зображенню повинна бути присвоєна відповідна мітка (label), яка вказує на клас або категорію цього зображення. Мітки мають бути уніфіковані, тобто можуть бути представлені числовими значеннями або текстовими категоріями. В даному випадку будуть ставитись мітки в такому порядку *хвороба/ її відсутність_навчальна/тренувальна/_№зображення*.

```
augmented_save_path = os.path.join(output_path, f'aneurysm_train_{i}_{j}.jpg')
cv2.imwrite(augmented_save_path, augmented_image)
```

Рисунок 3.11 – Збереження зображень у форматі .jpg та проставлення міток

Зменшення розміру даних.

За допомогою методів, таких як зменшення розміру зображень або видалення неінформативних ознак, можна зменшити обсяг даних. Це скорочує час навчання моделі та обсяги пам’яті, необхідні для зберігання даних.

```
target_size = (200, 200)
img_resized = cv2.resize(img, target_size)
```

Рисунок 3.12 – Збереження зображень розміром 200x200

Нормалізація даних.

Нормалізація даних – це процес перетворення вхідних даних у більш компактний і однорідний діапазон значень, зазвичай у межах [0, 1] або [-1, 1]. Цей процес допомагає зробити навчання нейронних мереж більш стабільним і ефективним, оскільки зменшується вплив великих або непропорційних значень у даних. В контексті зображень, нормалізація часто полягає в діленні кожного пікселя на максимальне можливе значення інтенсивності (наприклад, 255 для

зображень з 8-бітною глибиною кольору). Це переводить значення пікселів у діапазон від 0 до 1.

```
normalized_images = resized_images.astype('float32') / 255.0
```

Рисунок 3.13 – Нормалізація зображень

Після завершення всіх етапів підготовки та попередньої обробки даних, було отримано значно більший і збалансованіший набір зображень для навчання та тестування моделі. В таблиці 3.3 наведено результати розподілу кількості зображень для різних класів патологій.

Таблиця 3.3 – Розмір вибірки

	Aneurysm	**Hemorrhagic**	**Ischaemic**	**Normal**
Train	3015	4198	3604	3322
Test	639	567	675	783

Збільшення розміру вибірки для всіх класів після аугментації забезпечує рівномірне представлення кожної категорії, що є важливим для стабільної та точної роботи моделі. Такий розподіл допомагає уникнути проблеми дисбалансу даних, коли певні категорії зображень були б представлені недостатньо. Завдяки цьому модель зможе краще розпізнавати різні типи інсультів та патологій.

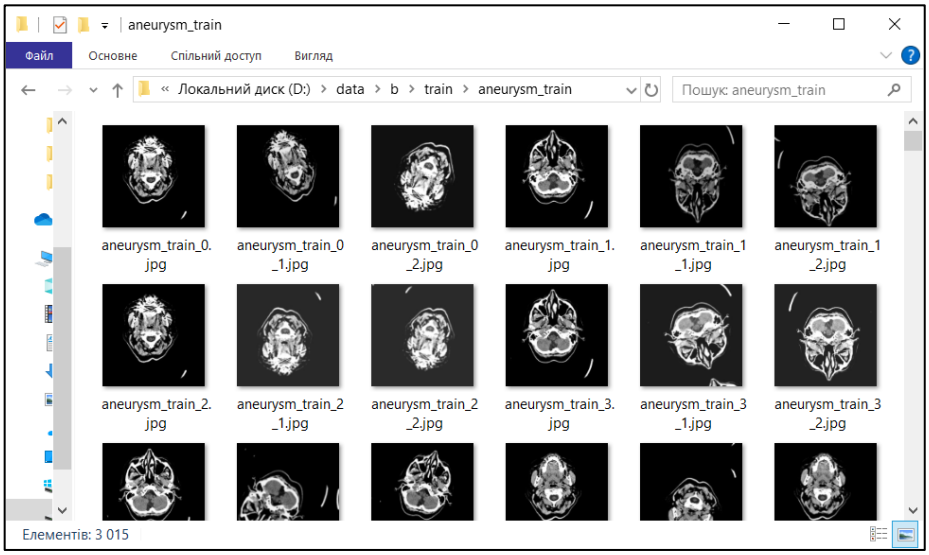

Рисунок 2.14 – Приклад вибірки після попередньої обробки

Висновки до розділу 3

В цьому розділі було детально розглянуто процес моделювання та проєктування інформаційної системи для діагностування нейросудинних хвороб. На основі розробленої структури системи було проаналізовано основні функції, відображені у вигляді дерева функцій, та побудовано діаграми IDEF0 і Use case для ілюстрації ключових етапів роботи системи.

Також проведено аналіз доступних наборів даних, які використовуються для навчання та тестування моделі. Реалізовано попередню обробку даних, включаючи процедури очищення даних, нормалізації, аугментації та балансування вибірки. У результаті цих кроків було створено збалансований датасет, що дозволяє системі ефективно розпізнавати різні типи нейросудинних патологій.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

4.1 Створення моделей з використанням архітектур U-Net та ResNet50

Для системи було обрано архітектури U-Net, яка є однією з найпопулярніших і ефективних моделей для завдань сегментації зображень, зокрема медичних зображень, та ResNet50, яка є оптимальним вибором для задач класифікації зображень.

U-Net забезпечує точне відновлення просторових деталей та має високі показники продуктивності для сегментації, тоді як ResNet50 завдяки своїй глибокій архітектурі та використанню механізму залишкових зв’язків уникає проблеми згасання градієнта, що дозволяє ефективно навчати глибокі моделі на складних наборах даних. Разом ці моделі забезпечують комплексний підхід до автоматизації аналізу медичних зображень, поєднуючи точність сегментації та класифікації для діагностики та оцінки патологій.

4.1.1 Створення моделі з використанням архітектури ResNet50

Перед початком проводиться попередня обробка та нормалізація даних.

Модель базується на ResNet50 (рис. 4.1), завантаженій з вагами *ImageNet*. ResNet50 використовується як базовий екстрактор ознак. Її шари заморожуються (параметр *trainable = False*), щоб уникнути змін у вже натренованих вагах. На виході додаються:

- шар *GlobalAveragePooling2D*, який зводить вихід ResNet50 до вектора фіксованого розміру;
- шар *Dropout* із коефіцієнтом 0.3 для зменшення ризику перенавчання;
- шар *Dense* із *Softmax*-активацією для класифікації на чотири класи.

```python
# Побудова класифікаційної моделі
def build_resnet_classification_model(input_shape=(200, 200, 3), num_classes=NUM_CLASSES):
    base_model = tf.keras.applications.ResNet50(include_top=False, input_shape=input_shape, weights='imagenet')
    base_model.trainable = False  # Заморожуємо шари ResNet50

    inputs = tf.keras.Input(shape=input_shape)
    x = base_model(inputs, training=False)
    x = tf.keras.layers.GlobalAveragePooling2D()(x)
    x = tf.keras.layers.Dropout(0.3)(x)
    outputs = tf.keras.layers.Dense(num_classes, activation='softmax')(x)

    model = tf.keras.Model(inputs, outputs)
    return model
```

Рисунок 4.1 – Побудова класифікаційної моделі

Модель компілюється з використанням оптимізатора Adam із початковою швидкістю навчання 0.001. Він забезпечує адаптивне налаштування ваг моделі. Також з допомогою функції втрат *sparse_categorical_crossentropy()*, яка є ефективною для задач мультикласової класифікації, особливо коли мітки представлені у вигляді цілих чисел.

Навчання виконується на 50 епох (рис. 4.2) із розміром батчу 16.

```
Epoch 1/50
177/177 ━━━━━━━━━━━━━━━━━━━━ 210s 1s/step - accuracy: 0.3547 - loss: 1.3057 - val_accuracy: 0.5537 - val_loss: 0.9396
Epoch 2/50
177/177 ━━━━━━━━━━━━━━━━━━━━ 196s 1s/step - accuracy: 0.5900 - loss: 0.9158 - val_accuracy: 0.6667 - val_loss: 0.8211
Epoch 3/50
177/177 ━━━━━━━━━━━━━━━━━━━━ 196s 1s/step - accuracy: 0.6583 - loss: 0.8121 - val_accuracy: 0.6808 - val_loss: 0.7952
Epoch 4/50
177/177 ━━━━━━━━━━━━━━━━━━━━ 194s 1s/step - accuracy: 0.6655 - loss: 0.7876 - val_accuracy: 0.7316 - val_loss: 0.7205
Epoch 5/50
177/177 ━━━━━━━━━━━━━━━━━━━━ 195s 1s/step - accuracy: 0.6905 - loss: 0.7308 - val_accuracy: 0.7500 - val_loss: 0.6848
Epoch 6/50
177/177 ━━━━━━━━━━━━━━━━━━━━ 320s 2s/step - accuracy: 0.7045 - loss: 0.7160 - val_accuracy: 0.7345 - val_loss: 0.6801
Epoch 7/50
177/177 ━━━━━━━━━━━━━━━━━━━━ 195s 1s/step - accuracy: 0.7204 - loss: 0.6896 - val_accuracy: 0.7613 - val_loss: 0.6566
Epoch 8/50
177/177 ━━━━━━━━━━━━━━━━━━━━ 195s 1s/step - accuracy: 0.7458 - loss: 0.6551 - val_accuracy: 0.7655 - val_loss: 0.6338
Epoch 9/50
177/177 ━━━━━━━━━━━━━━━━━━━━ 196s 1s/step - accuracy: 0.7372 - loss: 0.6446 - val_accuracy: 0.7853 - val_loss: 0.6202
Epoch 10/50
177/177 ━━━━━━━━━━━━━━━━━━━━ 194s 1s/step - accuracy: 0.7449 - loss: 0.6590 - val_accuracy: 0.7740 - val_loss: 0.6188
Epoch 11/50
177/177 ━━━━━━━━━━━━━━━━━━━━ 209s 1s/step - accuracy: 0.7276 - loss: 0.6490 - val_accuracy: 0.7669 - val_loss: 0.6080
Epoch 12/50
177/177 ━━━━━━━━━━━━━━━━━━━━ 201s 1s/step - accuracy: 0.7539 - loss: 0.6144 - val_accuracy: 0.7797 - val_loss: 0.5903
Epoch 13/50
177/177 ━━━━━━━━━━━━━━━━━━━━ 194s 1s/step - accuracy: 0.7549 - loss: 0.6150 - val_accuracy: 0.7938 - val_loss: 0.5728
Epoch 14/50
177/177 ━━━━━━━━━━━━━━━━━━━━ 194s 1s/step - accuracy: 0.7575 - loss: 0.5871 - val_accuracy: 0.8037 - val_loss: 0.5618
```

Рисунок 4.2 – Навчання моделі

Для покращення навчання використовуються два колбеки (рис. 4.3).

```python
# Колбеки
early_stop = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
checkpoint = tf.keras.callbacks.ModelCheckpoint(MODEL_SAVE_PATH, save_best_only=True, monitor='val_loss')
```

Рисунок 4.3 – Колбеки

1. *EarlyStopping* припиняє навчання, якщо протягом п'яти послідовних епох валідаційні втрати не покращуються. Це дозволяє уникнути перенавчання.
2. *ModelCheckpoint* зберігає найкращу модель (з найнижчими валідаційними втратами) у файл *classification_model.keras*.

4.1.2 Створення моделі з використанням архітектури U-Net

При створення моделі для задачі сегментації, U-Net потребує масок [41], оскільки вони виступають мітками для навчання. Маски відображають піксельну класифікацію зображення, де кожен піксель маски відповідає певному класу (наприклад, здоровій тканині чи ураженню). Під час навчання модель зіставляє свої прогнози зі справжніми масками (ground truth), мінімізуючи похибку. Це дозволяє моделі точно ідентифікувати патології або певні структури, зокрема у знімках мозку.

Псевдомаски [42] використовуються, якщо справжніх розміток недостатньо, або ж в даному випадку їх немає. Вони створюються автоматичними алгоритмами або попередньо навченими моделями і слугують наближеними мітками. Це дозволяє швидко розпочати навчання і побудувати модель, яку пізніше можна вдосконалити на якісніших даних.

```python
# Функція для генерації псевдомаски
def generate_pseudo_mask(image):
    _, thresh = cv2.threshold(image, 50, 255, cv2.THRESH_BINARY)
    kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (5, 5))
    cleaned = cv2.morphologyEx(thresh, cv2.MORPH_CLOSE, kernel)

    contours, _ = cv2.findContours(cleaned, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    mask = np.zeros_like(image)

    if contours:
        largest_contour = max(contours, key=cv2.contourArea)
        cv2.drawContours(mask, [largest_contour], -1, 1, -1)  # Значення 1 для ураження

    return cv2.resize(mask, IMG_SIZE).astype(np.uint8)
```

Рисунок 4.4 – Створення псевдомасок

Алгоритм генерації псевдомасок.

Порогова обробка зображення – це перший крок у генерації псевдомаски. Використовується поріг, щоб виділити області, що ймовірно належать до патології.

cv2.threshold(image, 50, 255, cv2.THRESH_BINARY) – цей метод застосовує бінаризацію зображення, щоб зробити його чорно-білим. Усі пікселі, що мають значення більше 50, будуть мати значення 255 (білий), а всі решта – значення 0 (чорний). Це допомагає виділити можливі області, які містять важливі ознаки, наприклад, пошкодження або ураження на зображенні. Параметр 50 – це поріг для бінаризації, тобто пікселі з інтенсивністю вище цього значення будуть віднесені до білих, а решта – до чорних.

Після порогової обробки застосовуються **морфологічні операції**, такі як закриття (closing) чи відкриття (opening). Це дозволяє видаляти шуми, згладжувати межі, а також з'єднувати фрагментовані області патології.

cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (5, 5)) – створює структурний елемент у формі еліпса розміру 5x5. Цей елемент використовуватиметься для морфологічних операцій.

cv2.morphologyEx(thresh, cv2.MORPH_CLOSE, kernel) – морфологічна операція закриття (morphological closing). Ця операція використовується для видалення шуму та заповнення маленьких прогалин у зображенні, покращуючи контури і роблячи їх більш гладкими.

Після цього зображення обробляється для **знаходження контурів**, що дозволяє визначити межі патологічної області. Найбільший контур вважається основною областю ураження, яка потім використовується для створення маски.

cv2.findContours – функція для пошуку контурів на обробленому бінарному зображенні.

Генерована псевдомаска використовується як **мітка для навчання моделі**, де область, що відповідає патології, позначена певним значенням (наприклад, 1), а інші області – іншим значенням (наприклад, 0).

largest_contour = max(contours, key=cv2.contourArea) – вибір найбільшого

контурного об'єкта за площею, що, відповідає найбільш значущому об'єкту на зображенні.

cv2.drawContours(mask, [largest_contour], -1, 1, -1) – малює знайдений контур на масці. Параметр 1 вказує на те, що пікселі контурного об'єкта будуть встановлені в значення 1 (білий).

mask = np.zeros_like(image) – створюється нова маска, яка має такий самий розмір, як і вхідне зображення, і заповнена нулями (чорний фон).

cv2.resize(mask, IMG_SIZE) – масштабує маску до того ж розміру, що й вхідне зображення (300x300).

.astype(np.uint8) – перетворює маску в тип даних uint8, щоб зберегти її як бінарну (0 або 1).

Зображення та маски для кожного класу (здорові тканини, ішемія, геморагія, аневризми) з'єднуються у загальні масиви x (зображення) і y (маски).

Побудова моделі U-Net.

Encoder (спадна частина).

Він відповідає за поступове зменшення розмірності вхідного зображення та витягнення високорівневих ознак (рис. 4.5).

Conv2D – 64 фільтри розміром 3x3. Активація *relu* додає нелінійність, а *padding='same'* зберігає розмір просторового виходу.

BatchNormalization – нормалізує активації, прискорюючи навчання.

MaxPooling2D – зменшує розмір зображення вдвічі для витягнення важливих ознак (200×200 → 100×100).

```
c1 = tf.keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same')(inputs)
c1 = tf.keras.layers.BatchNormalization()(c1)
p1 = tf.keras.layers.MaxPooling2D((2, 2), (2, 2), padding='same')(c1)
```

Рисунок 4.5 – Перший рівень обробки

В другому рівні обробки (рис. 4.6) застосовується те саме, що й на першому рівні, але кількість фільтрів збільшується до 128. Розмір зображення зменшується ще раз (100×100 → 50×50).

```
c2 = tf.keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same')(p1)
c2 = tf.keras.layers.BatchNormalization()(c2)
p2 = tf.keras.layers.MaxPooling2D((2, 2), (2, 2), padding='same')(c2)
```

Рисунок 4.6 – Другий рівень обробки

На цьому рівні (рис. 4.7) мережа працює з найменшою розмірністю (50×50). Витягується найвища кількість ознак (256 фільтрів).

```
c5 = tf.keras.layers.Conv2D(256, (3, 3), activation='relu', padding='same')(p2)
c5 = tf.keras.layers.BatchNormalization()(c5)
```

Рисунок 4.7 – Центральний блок

Decoder (розширювальна частина).

Декодер відповідає за відновлення просторової роздільної здатності, втраченої під час обробки в енкодері, і дозволяє відновити вихідне зображення.

Conv2DTranspose – збільшує розмір зображення вдвічі (50×50 → 100×100), використовуючи 128 фільтрів.

Concatenate – з'єднує вихід декодера з відповідним рівнем енкодера (c2), дозволяючи моделі отримати більше деталей.

Conv2D – застосовується для подальшої обробки.

```
u6 = tf.keras.layers.Conv2DTranspose(128, (2, 2), strides=(2, 2), padding='same')(c5)
u6 = tf.keras.layers.Concatenate()([u6, c2])
c6 = tf.keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same')(u6)
```

Рисунок 4.8 – Перший рівень декодування

Аналогічно першому рівню, але тепер кількість фільтрів другий рівень (рис 4.9) зменшується до 64, а розмір зображення відновлюється (100×100 → 200×200).

```
u7 = tf.keras.layers.Conv2DTranspose(64, (2, 2), strides=(2, 2), padding='same')(c6)
u7 = tf.keras.layers.Concatenate()([u7, c1])
c7 = tf.keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same')(u7)
```

Рисунок 4.9 – Другий рівень декодування

Вихідний шар.

```
outputs = tf.keras.layers.Conv2D(num_classes, (1, 1), activation='softmax')(c7)
model = tf.keras.Model(inputs=inputs, outputs=outputs)
return model
```

Рисунок 4.10 – Вихідний шар

Фільтри розміром 1×1 використовуються для перетворення ознак у ймовірності класів. Кількість фільтрів дорівнює *num_classes* (кількість класів сегментації). Softmax забезпечує нормалізацію виходу, щоб отримати ймовірності для кожного класу.

Навчання виконується на 20 епохах (рис. 4.11) із розміром батчу 16.

```
Epoch 1/20
177/177 ━━━━━━━━━━━━━━━━━━━━ 1798s 10s/step - accuracy: 0.7676 - iou_metric: 0.3344 - loss: 0.5353 - val_accuracy: 0.1606 - val_iou_metric: 0.1017 - val_loss: 1.5103
Epoch 2/20
177/177 ━━━━━━━━━━━━━━━━━━━━ 1577s 9s/step - accuracy: 0.8172 - iou_metric: 0.4048 - loss: 0.4208 - val_accuracy: 0.1707 - val_iou_metric: 0.0562 - val_loss: 1.6958
Epoch 3/20
177/177 ━━━━━━━━━━━━━━━━━━━━ 1589s 9s/step - accuracy: 0.8360 - iou_metric: 0.4506 - loss: 0.3690 - val_accuracy: 0.7513 - val_iou_metric: 0.2930 - val_loss: 0.8534
Epoch 4/20
177/177 ━━━━━━━━━━━━━━━━━━━━ 1559s 9s/step - accuracy: 0.8344 - iou_metric: 0.4471 - loss: 0.3712 - val_accuracy: 0.8315 - val_iou_metric: 0.4617 - val_loss: 0.4021
Epoch 5/20
177/177 ━━━━━━━━━━━━━━━━━━━━ 1532s 9s/step - accuracy: 0.8475 - iou_metric: 0.4759 - loss: 0.3466 - val_accuracy: 0.7858 - val_iou_metric: 0.3531 - val_loss: 1.0830
Epoch 6/20
177/177 ━━━━━━━━━━━━━━━━━━━━ 1530s 9s/step - accuracy: 0.8460 - iou_metric: 0.4717 - loss: 0.3471 - val_accuracy: 0.7837 - val_iou_metric: 0.3825 - val_loss: 0.6467
Epoch 7/20
177/177 ━━━━━━━━━━━━━━━━━━━━ 1529s 9s/step - accuracy: 0.8503 - iou_metric: 0.4978 - loss: 0.3340 - val_accuracy: 0.7474 - val_iou_metric: 0.2453 - val_loss: 1.4691
Epoch 8/20
177/177 ━━━━━━━━━━━━━━━━━━━━ 1533s 9s/step - accuracy: 0.8508 - iou_metric: 0.4926 - loss: 0.3344 - val_accuracy: 0.7473 - val_iou_metric: 0.2959 - val_loss: 0.8837
Epoch 9/20
177/177 ━━━━━━━━━━━━━━━━━━━━ 1532s 9s/step - accuracy: 0.8516 - iou_metric: 0.5081 - loss: 0.3309 - val_accuracy: 0.8475 - val_iou_metric: 0.4964 - val_loss: 0.3164
Epoch 10/20
177/177 ━━━━━━━━━━━━━━━━━━━━ 1531s 9s/step - accuracy: 0.8573 - iou_metric: 0.5148 - loss: 0.3234 - val_accuracy: 0.7977 - val_iou_metric: 0.4088 - val_loss: 0.5727
Epoch 11/20
```

Рисунок 4.11 – Навчання моделі

Симптоми впливають на діагноз наступним чином:

– якщо зображення КТ визначається як «Здоровий мозок», але пацієнт вибрав певні симптоми, система генерує припущення про ймовірний стан (інсульт чи аневризма) залежно від кількості відповідних симптомів;

– симптоми підсилюють діагностичний висновок у разі, якщо візуальна класифікація або сегментація не дають чіткої відповіді.

Симптоми групуються (рис. 4.11) в SYMPTOMS_ANEURYSM (для аневризми) та SYMPTOMS_STROKE (для інсульту), що дозволяє оцінити, яка патологія ймовірніша.

```
# Симптоми
SYMPTOMS_ANEURYSM = [
    "Головний біль", "Біль в області шиї або потилиці", "Судоми",
    "Підвищена чутливість до світла", "Погіршення слуху або дзвін у вухах"
]
SYMPTOMS_STROKE = [
    "Запаморочення", "Слабкість у кінцівках", "Порушення мови",
    "Оніміння обличчя, руки або ноги", "Асиметрія обличчя",
    "Втрата свідомості", "Нудота та блювання", "Проблеми із зором"
]
SYMPTOMS = SYMPTOMS_ANEURYSM + SYMPTOMS_STROKE
```

Рисунок 4.12 – Симптоми

Залежно від симптому і виявленої патології рекомендації змінюються. Якщо на зображенні виявлено патологію, симптоми можуть уточнити стан (наприклад, судоми можуть свідчити про важку форму аневризми).

Детальний код системи можна переглянути в додатку Б.

4.2 Оцінка створених моделей

Для оцінки якості створених моделей було застосовано наступні метрикиI:

– ***precision*** показує, яка частка з усіх передбачень класу є правильними;

– ***recall*** показує, яку частку з усіх реальних позитивних випадків модель змогла правильно виявити [43];

– ***F1-score*** [44] є середньо гармонійним значенням Precision і Recall. Це метрика, яка балансує між точністю та чутливістю, і корисна, коли важливо мати гармонійне співвідношення цих двох показників.

З рисунку 4.13 можемо побачити що модель класифікації демонструє хорошу загальну продуктивність, особливо для класів «Ішемічний» та «Аневризми», де досягаються високі показники Precision, Recall і F1-Score. Це вказує на здатність моделі точно ідентифікувати більшість зразків цих класів та мінімізувати помилкові передбачення.

Проте продуктивність для класів «Здоровий» і «Геморагічний» є менш стабільною. Для «Геморагічного» класу особливо помітне зниження Recall, що свідчить про пропущення значної кількості зразків цього класу. Для "Здорового"

класу модель демонструє середній рівень Precision і Recall, що також може потребувати вдосконалення.

Загалом модель добре справляється із задачами класифікації.

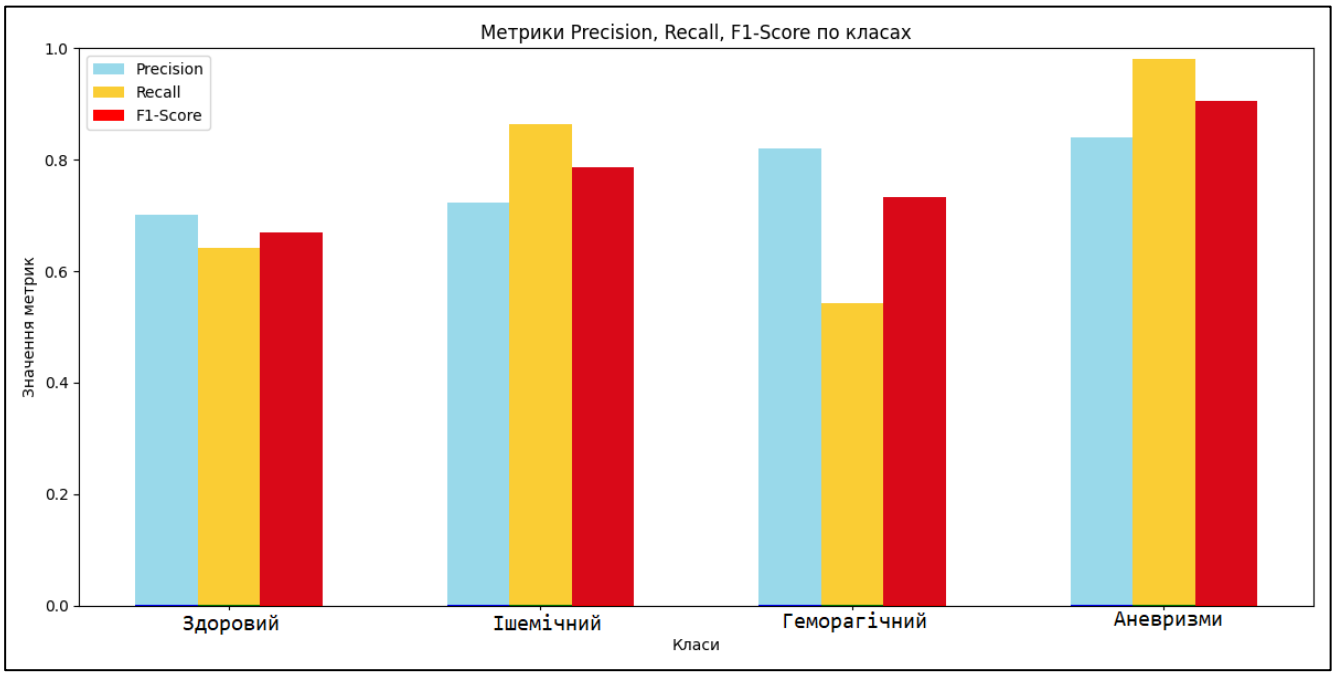

Рисунок 4.13 – Оцінка моделі класифікації

З рисунку 4.14 бачимо, що в моделі сегментації найкращі результати демонструє «Ішемічний» та «Аневризм», де всі метрики близькі до 0.8–0.85. Для «Здорового» класу метрики середні (~0.6–0.7), а найнижчі значення спостерігаються для «Геморагічного» класу (~0.5–0.6), що вказує на труднощі моделі з його розпізнаванням.

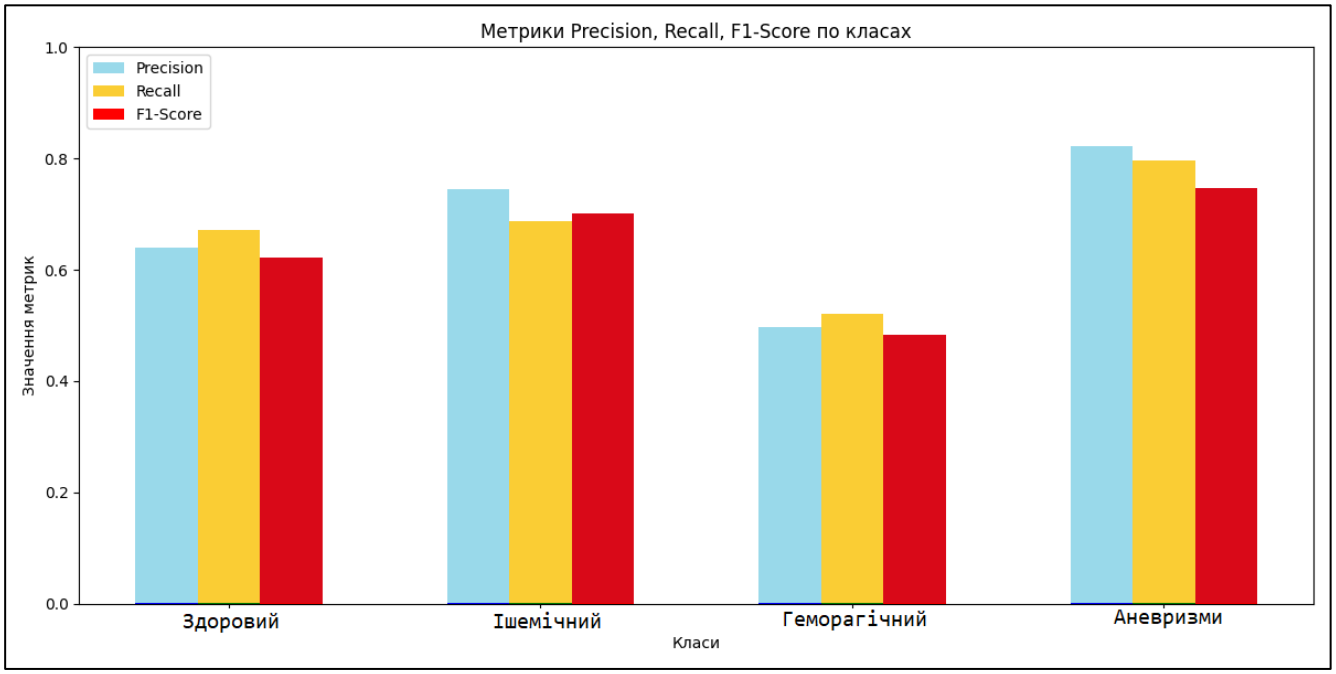

Рисунок 4.14 – Оцінка моделі сегментації

Ще було застосовано наступні метрики:

– ***accuracy*** вимірює загальну кількість правильних передбачень моделі відносно загальної кількості передбачень. Це одна з найпростіших метрик для оцінки моделі класифікації;

– ***loss*** вимірює, наскільки сильно модель відхиляється від правильного передбачення. Це числовий показник, який оцінюється під час навчання моделі. Loss є основою, на якій оптимізується модель, щоб покращити її ефективність [45].

На рисунку 4.15 зображено точність (accuracy) та втрати (loss) моделі під час навчання та валідації по епохах. Видно, що з кожною епохою точність поступово зростає, і до кінця навчання досягає значення близько 0.85 для навчальної вибірки і трохи менше для валідаційної. Втрати зменшуються протягом усього процесу навчання, що вказує на покращення моделі. Валідаційні втрати близькі до тренувальних, що свідчить про відсутність значного перенавчання.

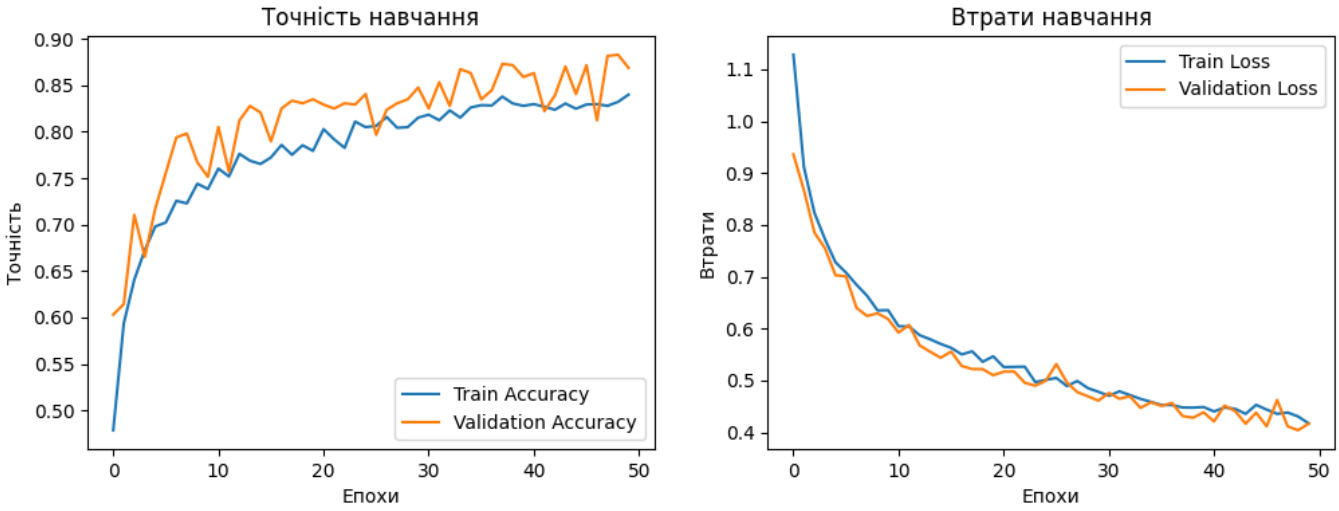

Рисунок 4.15 – Accuracy та loss моделі класифікації

На першому графіку (рис. 4.16) видно, що навчальна точність постійно зростає і наближається до 0.85 вже після кількох епох, тоді як валідаційна точність швидко зростає на початку і стабілізується на рівні, близькому до навчальної. Це вказує на ефективне навчання без значного перенавчання.

На другому графіку (рис. 4.16) навчальні втрати стабільно знижуються протягом епох, досягаючи низького рівня, що свідчить про гарну оптимізацію моделі. Втрати на валідації коливаються, але в цілому демонструють тенденцію до зменшення, хоча деякі сплески можуть вказувати на нестабільність під час оцінки.

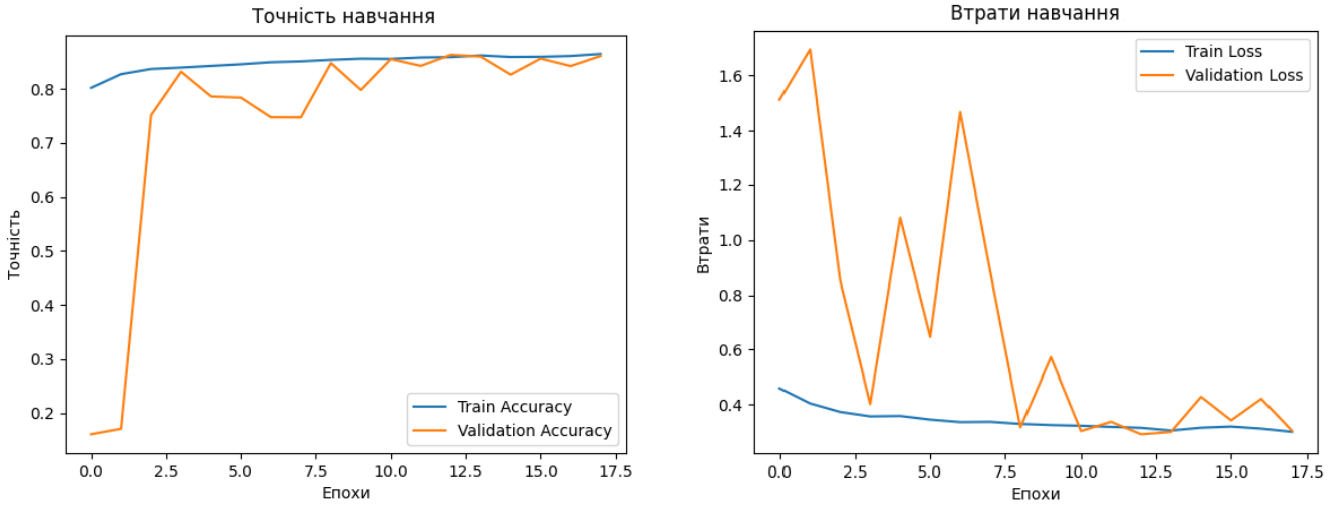

Рисунок 4.16 – Accuracy та loss моделі сегментації

4.3 Тестування системи

Початкове вікно системи зображено на рисунку 4.17.

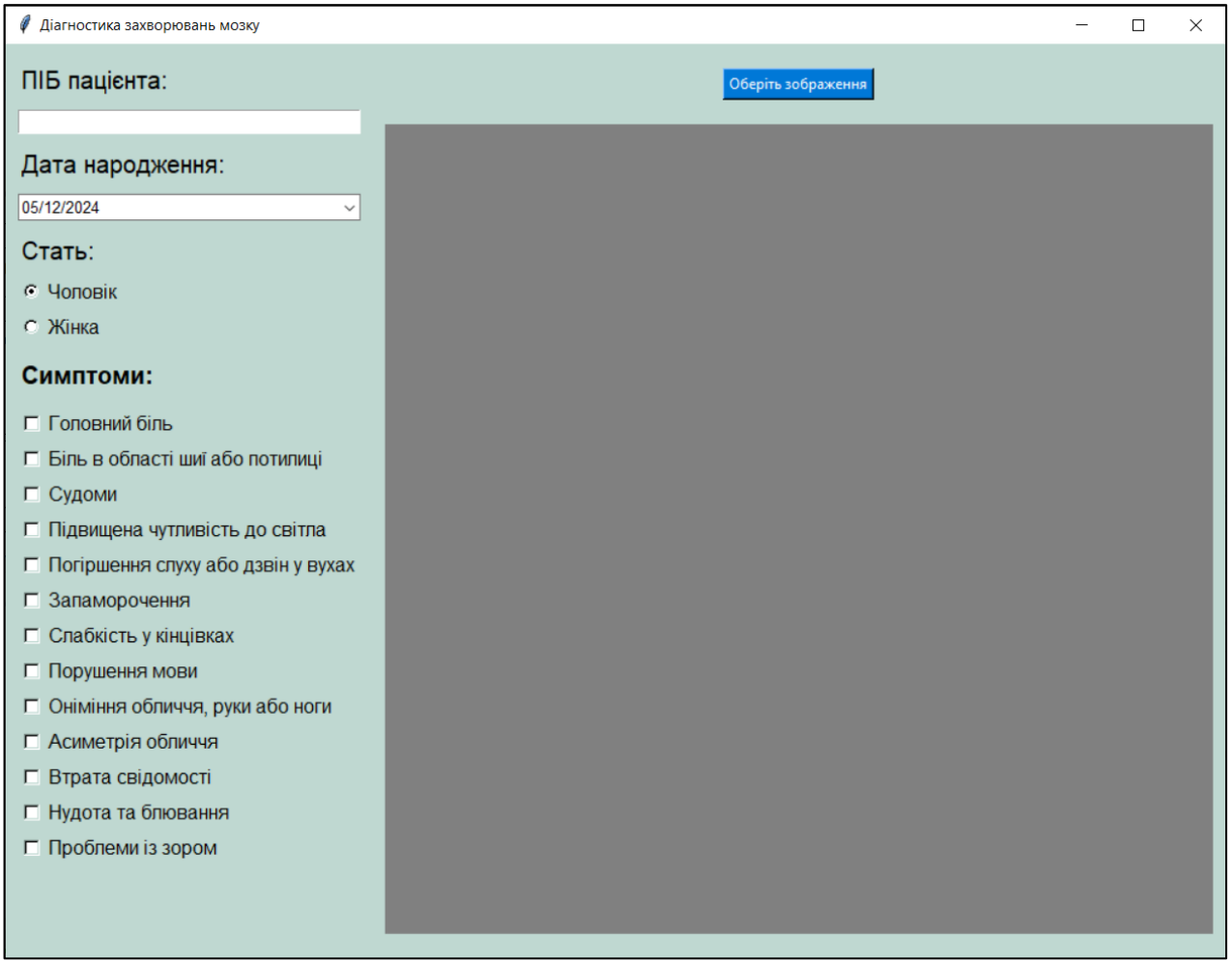

Рисунок 4.17 – Початкове вікно системи

Зліва головного вікна користувач вводить необхідні дані пацієнта, зокрема його ПІБ, дату народження та обирає стать. Далі користувач вибирає симптоми, які турбують, з наданого списку.

Після того як всі основні дані введені, користувач натискає кнопку «Оберіть зображення», щоб завантажити медичне зображення пацієнта для подальшої обробки. Підтримуються зображення у таких форматах, як .png, .jpg та .jpeg. Після вибору потрібного зображення воно відображається на екрані. У цей момент з’являється нова кнопка «Перевірити», яка дозволяє почати обробку зображення за допомогою моделі для виявлення нейросудинних захворювань. (рис. 4.18).

Рисунок 4.18 – Вибір зображення

Після цього, завантажене зображення з'являється на правій частині екрану. Під ним відображається зображення, на якому виділена уражена ділянка. Ця ділянка підсвічується червоним кольором, щоб чітко вказати на місце, яке потребує уваги. Посередині екрану знаходиться блок з діагнозом, який визначається на основі зображення та симптомів пацієнта. У цьому ж блоці відображається відсоток ураження, що допомагає визначити тяжкість стану пацієнта. Поряд з діагнозом з’являються рекомендації щодо подальшого лікування, що включають можливі методи терапії та додаткові обстеження, які можуть бути корисними для уточнення діагнозу та покращення стану пацієнта. (рис. 4.19).

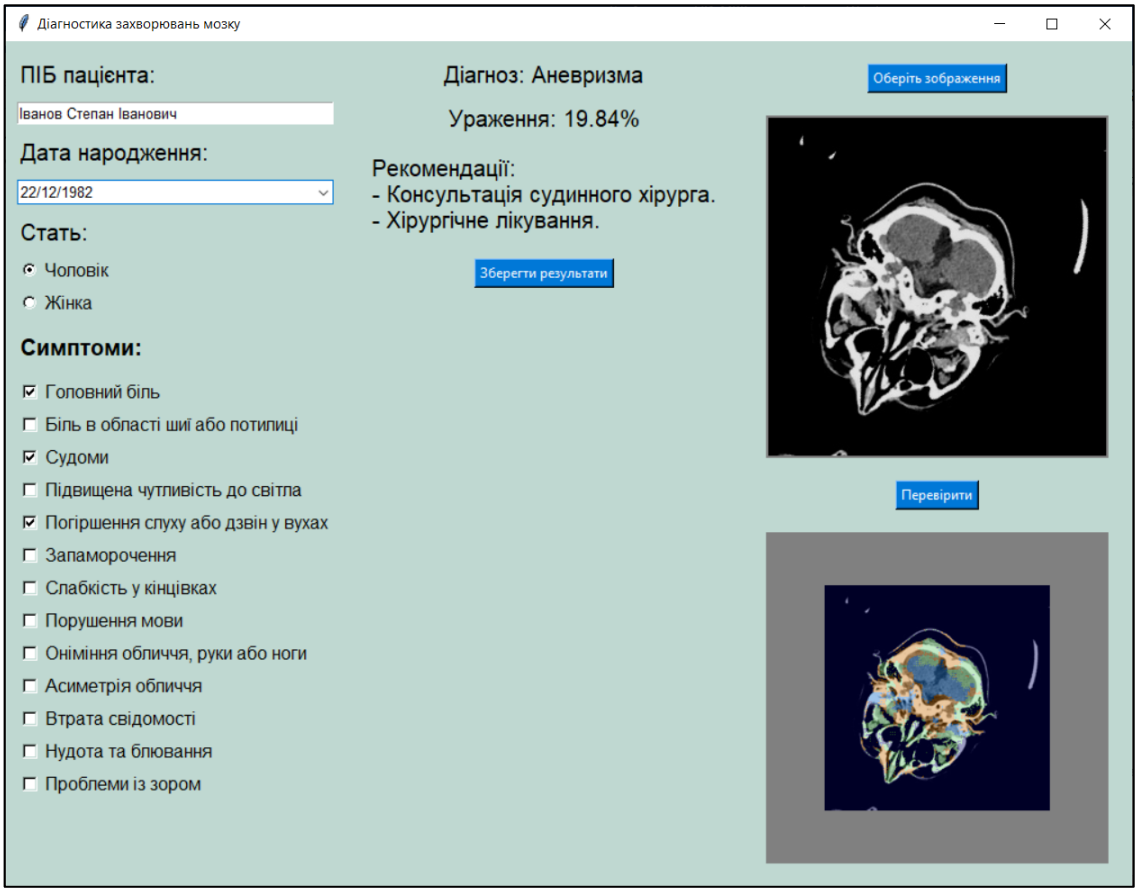

Рисунок 4.19 – Вибір зображення

Користувач може зберегти результати натиснувши кнопку «Зберегти результати». Вони зберігаються у текстовому форматі. Після успішного збереження з’являється повідомлення (рис. 4.20).

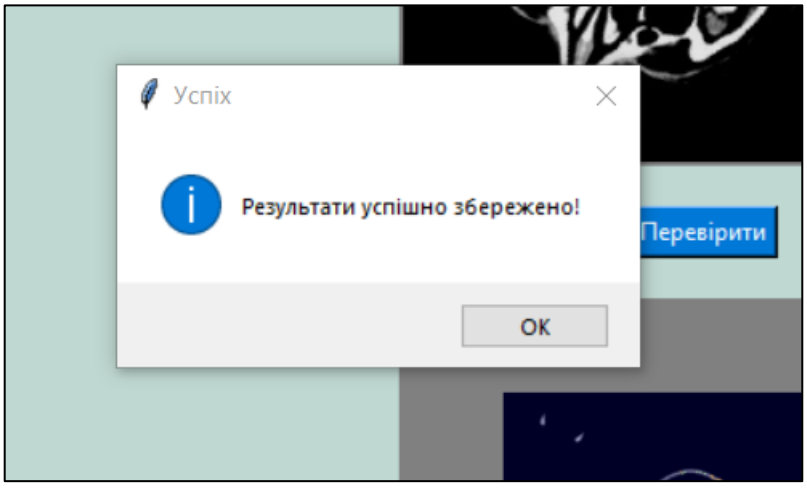

Рисунок 4.20 – Успішне збереження результатів

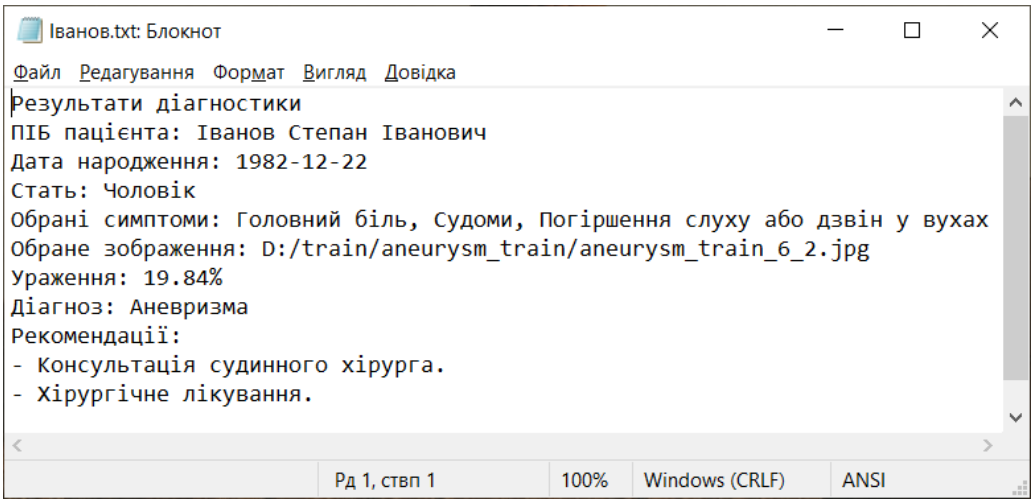

Рисунок 4.21 – Результати діагностики

ПІБ пацієнта є обов'язковим полем для введення, тому при спробі зберегти результат без заповнення цього поля, система автоматично повідомляє користувача про необхідність вказати ПІБ (рис. 4.22).

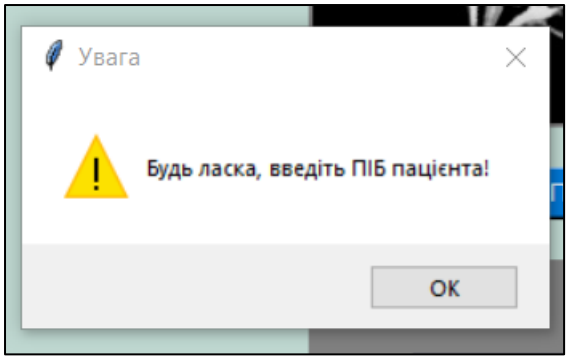

Рисунок 4.22 – Повідомлення про необхідність вказати ПІБ

Ситуація, коли на зображенні КТ не було виявлено захворювання, але симптоми вказують на це, показано на рисунках 4.23-4.25. В рекомендаціях відповідно до можливого діагнозу вказано, що саме краще робити в даному випадку.

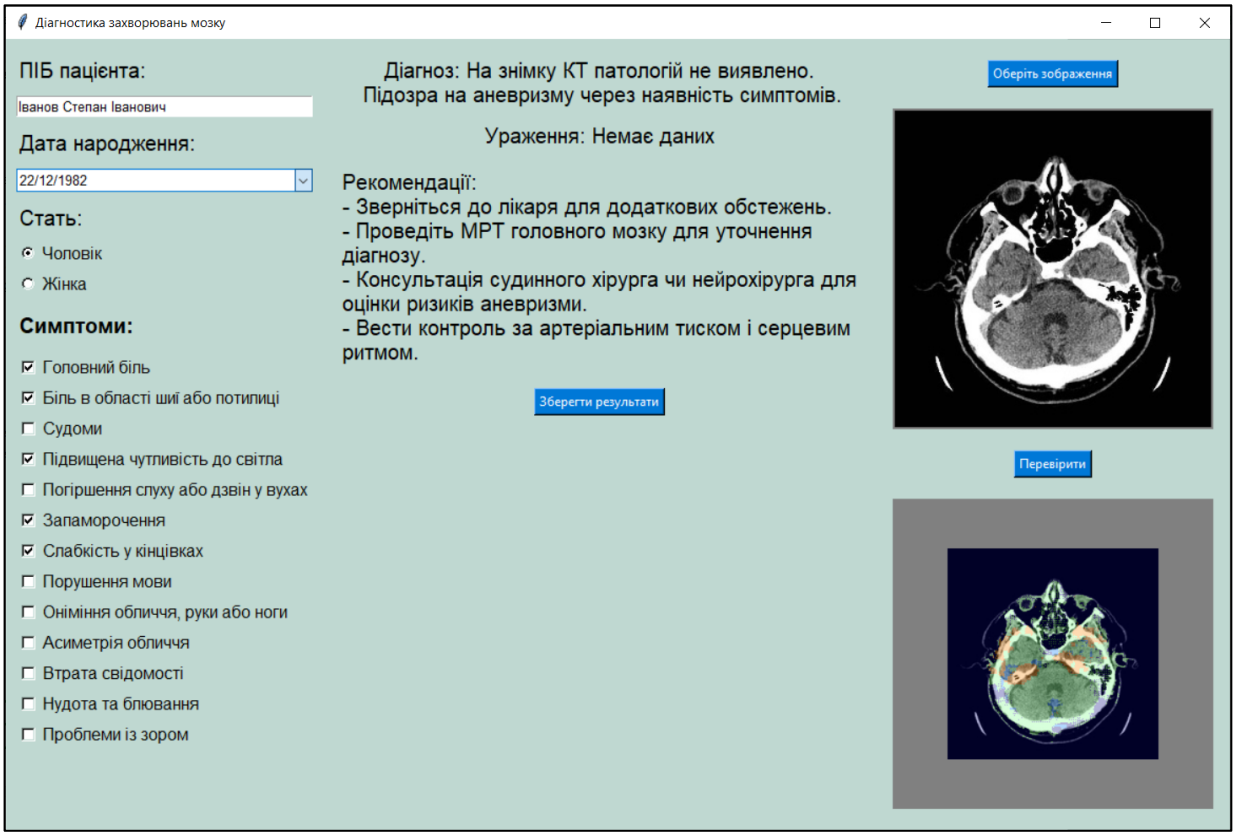

Рисунок 4.23 – Симптоми вказують на аневризму

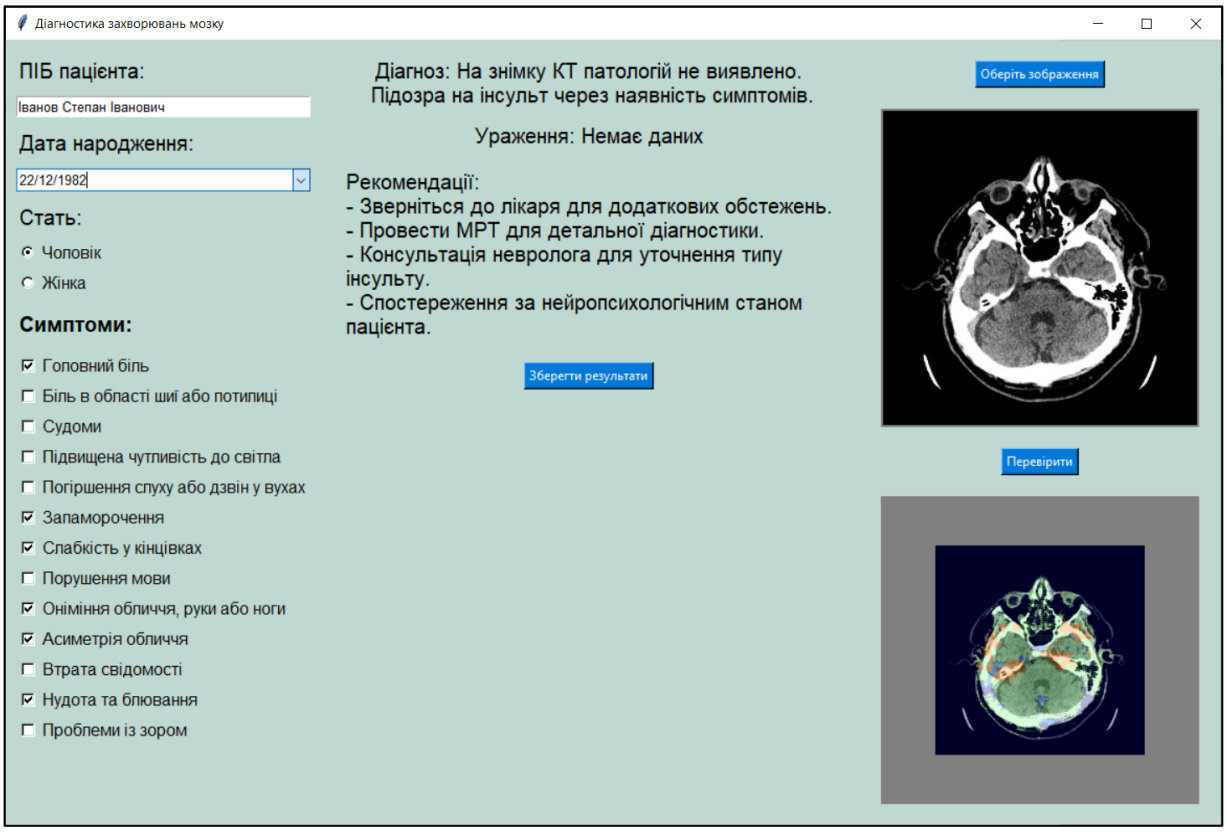

Рисунок 4.24 – Симптоми вказують на інсульт

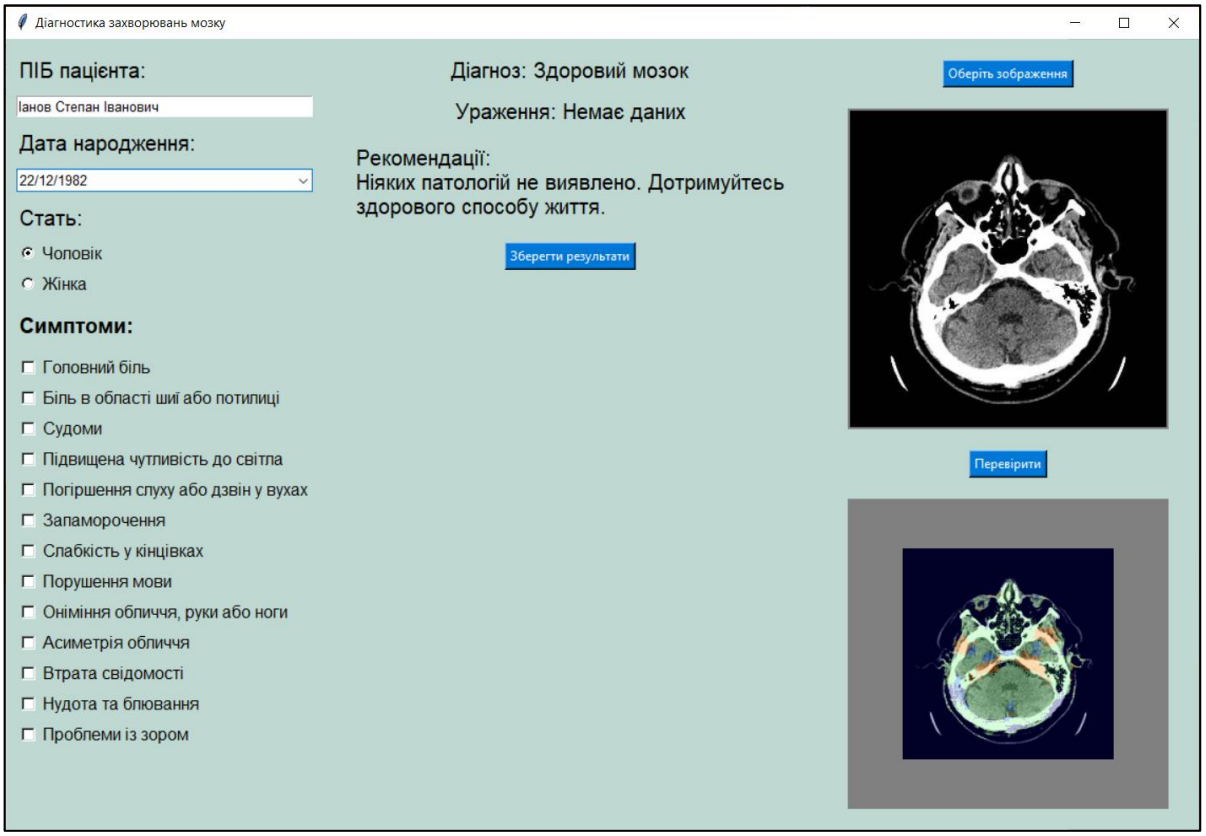

Рисунок 4.25 – КТ здорового мозку без симптомів

Отже, результати показали, що модель класифікації працює добре. Вона здатна коректно розпізнавати здоровий мозок та основні патології, такі як інсульт чи аневризма.

Однак сегментаційна частина моделі демонструє гірші результати, особливо на складних знімках КТ, де структура мозку має неоднорідності або присутні артефакти. Псевдомаски, згенеровані для навчання, можуть бути недостатньо точними, що негативно впливає на здатність сегментаційної моделі коректно ідентифікувати область ураження. Через це хвороба в таких випадках розпізнається менш ефективно. Покращення сегментації, наприклад, використання більш точних масок може значно підвищити точність і якість ідентифікації патологій на складних знімках КТ.

Висновки до розділу 4

Створення моделей для діагностики нейросудинних захворювань, показують, що використання архітектур U-Net та ResNet50 забезпечило високий рівень точності в аналізі медичних зображень. Архітектура U-Net продемонструвала ефективність у задачах сегментації, дозволяючи виділяти патологічні ділянки на КТ-знімках, що суттєво покращило точність виявлення уражених зон головного мозку. ResNet50 показав чудові результати в класифікації зображень за категоріями, що забезпечило високу точність розпізнавання здорового мозку, інсульту та аневризми.

Моделі, що були створені, продемонстрували надійні результати в контексті метрик точності, втрат та F1-міри, що свідчить про їх здатність правильно класифікувати зображення та виявляти патології з високою впевненістю.

Тестування системи підтвердило, що програмне забезпечення є стабільним і ефективним, зручним у використанні та здатним швидко обробляти медичні зображення. Точність результатів, отриманих під час тестування, підтверджує надійність розробленої системи для автоматизованого аналізу КТ-знімків.

ВИСНОВКИ

У кваліфікаційній роботі було розроблено та впроваджено інформаційну систему для діагностики нейросудинних захворювань на основі аналізу КТ-знімків із використанням сучасних нейронних мереж.

На першому етапі було проведено аналіз предметної області, зокрема, розглянуто основні типи нейросудинних захворювань, їх симптоматику та методи діагностики. Було вивчено існуючі інтелектуальні системи в медицині, що дало змогу визначити основні проблеми, які необхідно вирішити, а також обґрунтувати доцільність використання нейромереж для підвищення точності діагностики.

Другий етап роботи присвячено вибору методів машинного навчання та технологій для розробки системи. Було обрано архітектури U-Net для сегментації зображень та ResNet50 для їх класифікації. Проведено аналіз інструментів, таких як TensorFlow, що забезпечили високу гнучкість і потужність у процесі розробки.

На третьому етапі здійснено моделювання та проєктування інформаційної системи, описано її структуру, функціональні можливості та етапи обробки даних. Проведено детальну роботу з підготовки та аналізу набору даних, що дозволило оптимізувати якість моделі на основі нормалізації даних.

Четвертий етап охоплює створення моделей, їх тестування та оцінку. Було доведено ефективність використання обраних архітектур для задач сегментації та класифікації. Розроблена система здатна виявляти патології, такі як ішемічний інсульт, геморагічний інсульт та аневризми, або підтверджувати відсутність уражень. Особливістю системи є можливість врахування симптомів, які вказує користувач під час введення даних. У разі, якщо патологій не виявлено, але симптоми вказують на їх наявність, система формує повідомлення про ймовірність патології та рекомендує додаткові обстеження. Завдяки цьому підхід дозволяє враховувати як об’єктивні дані КТ-знімків, так і клінічні прояви, що робить діагностику більш точною та інформативною. Система також забезпечує зручний інтерфейс для користувача, автоматизуючи процес аналізу медичних зображень і

формування рекомендацій для лікування.

Загалом, результати роботи демонструють, що впровадження інтелектуальних систем на основі нейронних мереж у медичну практику має значний потенціал для покращення діагностики та зменшення часу аналізу, що особливо важливо в умовах ургентної медицини. Система є ефективним і перспективним інструментом, який може бути використаний у медичних закладах для покращення точності та оперативності діагностики нейросудинних захворювань.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Neurovascular Disease | VMFH. VMFH. URL: https://www.vmfh.org/our-services/center-for-neurosciences-spine/neurovascular-disease (date of access: 23.09.2024).

2. Stroke: What It Is, Causes, Symptoms, Treatment & Types. Cleveland Clinic. URL: https://my.clevelandclinic.org/health/diseases/5601-stroke (date of access: 23.09.2024).

3. Brain aneurysm - Symptoms and causes. Mayo Clinic. URL: https://www.mayoclinic.org/diseases-conditions/brain-aneurysm/symptoms-causes/syc-20361483 (date of access: 24.09.2024).

4. Computed Tomography (CT). National Institute of Biomedical Imaging and Bioengineering. URL: https://www.nibib.nih.gov/science-education/science-topics/computed-tomography-ct (date of access: 25.09.2024).

5. Magnetic Resonance Imaging (MRI). Johns Hopkins Medicine. URL: https://www.hopkinsmedicine.org/health/treatment-tests-and-therapies/magnetic-resonance-imaging-mri (date of access: 25.09.2024).

6. CT Angiography (CTA). Radiologyinfo.org. URL: https://www.radiologyinfo.org/en/info/angioct (date of access: 26.09.2024).

7. Magnetic resonance angiography: MedlinePlus Medical Encyclopedia. MedlinePlus - Health Information from the National Library of Medicine. URL: https://medlineplus.gov/ency/article/007269.htm (date of access: 26.09.2024).

8. Digital Substruction Angiography (DSA) - ACIBADEM. Acıbadem Healthcare Services. URL: https://acibademinternational.com/technology/digital-substruction-angiography-dsa/ (date of access: 26.09.2024).

9. Ultrasound of blood vessels (head, neck, legs) Doppler. URL: https://universum.clinic/en/service/uz-doslidzhennya-vsi-vidi/uzd-sudin-golovi-shii-nig-dopler/ (date of access: 28.09.2024).

10. Advancing Breast Cancer Detection with AI - Google Health. What Is Google Health? - Google Health. URL: https://health.google/caregivers/mammography/ (date of access: 28.09.2024).

11. McKinney, S.M., Sieniek, M., Godbole, V. et al. International evaluation of an AI system for breast cancer screening. Nature 577, 89–94 (2020). https://doi.org/10.1038/s41586-019-1799-6.

12. Yim, J., Chopra, R., Spitz, T. et al. Predicting conversion to wet age-related macular degeneration using deep learning. Nat Med 26, 892–899 (2020). https://doi.org/10.1038/s41591-020-0867-7

13. Aidoc Always On Healthcare AI. Healthcare AI | Aidoc Always-on AI. URL: https://www.aidoc.com/eu/ (date of access: 26.09.2024).

14. Viz.ai | AI-Powered Care Coordination. Viz.ai, the Proven AI-Powered Care Coordination Platform. URL: https://www.viz.ai/ (date of access: 30.10.2024).

15. C. Wantaka, P. Kitidumrongsuk, P. Soontornpipit and J. Sillabutra, "Design and Development of Data Model for Stroke FAST Track System," 2019 International Electrical Engineering Congress (iEECON), Krabi, Thailand, 2019, pp. 1-4, doi: 10.1109/IEECON.2018.8712241.

16. M. Lee, J. -H. Jeong, Y. -H. Kim and S. -W. Lee, "Decoding Finger Tapping With the Affected Hand in Chronic Stroke Patients During Motor Imagery and Execution," in IEEE Transactions on Neural Systems and Rehabilitation Engineering, vol. 29, pp. 1099-1109, 2021, doi: 10.1109/TNSRE.2021.3087506.

17. J. Chaki and M. Woźniak, "Deep Learning and Artificial Intelligence in Action (2019–2023): A Review on Brain Stroke Detection, Diagnosis, and Intelligent Post-Stroke Rehabilitation Management," in IEEE Access, vol. 12, pp. 52161-52181, 2024, doi: 10.1109/ACCESS.2024.3383140.

18. Data classification methods–ArcGIS Pro | Documentation. URL: https://pro.arcgis.com/en/pro-app/latest/help/mapping/layer-properties/data-classification-methods.htm (date of access: 05.10.2024).

19. Frost J. Regression Analysis: An Intuitive Guide for Using and Interpreting Linear Models. Statistics By Jim Publishing, 2020. 355 p.

20. Чару Аггарвал Нейронні мережі і глибоке навчання. Навчальний курс, Вильямс, 2020. 645-650 с.

21. Саймон Х. Нейронні мережі. Повний курс. 2-е видання, 2020. 870 с.

22. Taye M. M. Theoretical Understanding of Convolutional Neural Network: Concepts, Architectures, Applications, Future Directions. Computation. 2023. Vol. 11, no. 3. P. 52. URL: https://doi.org/10.3390/computation11030052 (date of access: 06.10.2024).

23. API Documentatio|TensorFlow. URL: https://www.tensorflow.org/api_docs (date of access: 07.10.2024).

24. Keras documentation: The Model class. Keras: Deep Learning for humans. URL: https://keras.io/api/models/model/ (date of access: 07.10.2024).

25. Image Classification Algorithm Based on Improved AlexNet / S. Li et al. Journal of Physics: Conference Series. 2021. Vol. 1813, no. 1. P. 012051. URL: https://doi.org/10.1088/1742-6596/1813/1/012051 (date of access: 08.10.2024).

26. Classification of Simple CNN Model and ResNet50 / L. M. K. Sheikh et al. International Journal for Research in Applied Science and Engineering Technology. 2024. Vol. 12, no. 4. P. 4606–4610. URL: https://doi.org/10.22214/ijraset.2024.60677 (date of access: 08.10.2024).

27. Zhao R. Brain Tumor Identification Based on AlexNet and VGG. Highlights in Science, Engineering and Technology. 2023. Vol. 57. P. 57–61. URL: https://doi.org/10.54097/hset.v57i.9897 (date of access: 09.10.2024).

28. Retinal Vessel Automatic Segmentation Using SegNet / X. Xu et al. Computational and Mathematical Methods in Medicine. 2022. Vol. 2022. P. 1–11. URL: https://doi.org/10.1155/2022/3117455 (date of access: 09.10.2024).

29. Fang H. Semantic Segmentation Based on Improved DeeplabV3+. Mathematical Problems in Engineering. 2022. Vol. 2022. P. 1–8. URL: https://doi.org/10.1155/2022/6228532 (date of access: 09.10.2024).

30. Attention-augmented U-Net (AA-U-Net) for semantic segmentation / K. T. Rajamani et al. Signal, Image and Video Processing. 2022. URL: https://doi.org/10.1007/s11760-022-02302-3 (date of access: 09.10.2024).

31. Mayer A., Napel S. Weighted Scoring Committees. Games. 2021. Vol. 12, no. 4. P. 94. URL: https://doi.org/10.3390/g12040094 (date of access: 12.10.2024).

32. Marzouk M., Elmansy N. Roadmapping BIM Implementation Processes Using IDEF0 Diagrams. International Journal of 3-D Information Modeling. 2019. Vol. 7, no. 1. P. 49–63. URL: https://doi.org/10.4018/ij3dim.2018010104 (date of access: 22.10.2024).

33. Naylor M. Decision Tree. Mathematics Teacher: Learning and Teaching PK-12. 2020. Vol. 113, no. 7. P. 612. URL: https://doi.org/10.5951/mtlt.2020.0081 (date of access: 23.10.2024).

34. Miller G., Armour F. Advanced Use of Case Modelling. Pearson Education, Limited, 2020. 192 p.

35. Kulkarni D. R. N., Srinivasa C. K. Novel approach to transform UML Sequence diagram to Activity diagram. Journal of University of Shanghai for Science and Technology. 2021. Vol. 23, no. 07. P. 1247–1255. URL: https://doi.org/10.51201/jusst/21/07300 (date of access: 25.10.2024).

36. StrokeDataset. Kaggle: Your Machine Learning and Data Science Community. URL: https://www.kaggle.com/datasets/chelebi/strokedataset?select=train (date of access: 01.11.2024).

37. Brain CT Tumor Classification Dataset. Kaggle: Your Machine Learning and Data Science Community. URL: https://www.kaggle.com/datasets/killa92/brain-ct-tumor-classification-dataset (date of access: 01.11.2024).

38. Finaldata. Kaggle: Your Machine Learning and Data Science Community. URL: https://www.kaggle.com/datasets/biraramarew/finaldata (date of access: 01.11.2024).

39. DeepAI. DeepAI. URL: https://deepai.org/ (date of access: 07.11.2024).

40. GitHub - albumentations-team/albumentations: Fast and flexible image augmentation library. Paper about the library: https://www.mdpi.com/2078-2489/11/2/125. GitHub. URL: https://github.com/albumentations-team/albumentations (date of access: 14.11.2024).

41. Image segmentation masks. Labelbox | Data factory for the next GenAI. URL: https://labelbox.com/guides/image-segmentation/ (date of access: 27.11.2024).

42. Learning Pseudo Labels for Semi-and-weakly Supervised Semantic Segmentation / Y. Wang et al. Pattern Recognition. 2022. P. 108925. URL: https://doi.org/10.1016/j.patcog.2022.108925 (date of access: 27.11.2024).

43. Precision and Recall in Classification Models | Built In. Built In. URL: https://builtin.com/data-science/precision-and-recall (date of access: 29.11.2024).

44. F1 Score in Machine Learning. V7 | AI Document Processing & Data Labelling. URL: https://www.v7labs.com/blog/f1-score-guide (date of access: 29.11.2024).

45. Liu S., Zhou Z.-H. Machine Learning. Springer, 2020.

ДОДАТОК А

Апробація роботи

Робота пройшла апробацію під час Всеукраїнської науково-практичної конференції молодих вчених, аспірантів і студентів, 2–4 грудня 2024 р., м. Миколаїв.

Міністерство освіти і науки України
Чорноморський національний університет ім. Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Інформаційний лист

Всеукраїнська науково-практична конференція молодих вчених, аспірантів і студентів

Інтелектуальні інформаційні системи

2 – 4 грудня 2024 року

Миколаїв

УДК 004.42

Слободенюк А. І., Сіденко Є. В.
Чорноморський національний університет ім. Петра Могили, Миколаїв, Україна

ІНТЕЛЕКТУАЛЬНА СИСТЕМА ДІАГНОСТУВАННЯ НЕЙРОСУДИННИХ ХВОРОБ З ВИКОРИСТАННЯМ НЕЙРОННИХ МЕРЕЖ

Сучасна медицина знаходиться на стадії активного розвитку завдяки впровадженню новітніх технологій, серед яких особливе місце займають методи штучного інтелекту. Однією з ключових проблем охорони здоров'я є своєчасна та точна діагностика складних захворювань, зокрема нейросудинних хвороб, які займають одне з перших місць за рівнем смертності та інвалідності у світі. Такі захворювання, як інсульт, аневризми та тромбози мозкових судин, вимагають швидкої та точної діагностики для збереження життя пацієнтів та уникнення важких наслідків.

Інтелектуальна система діагностики нейросудинних захворювань базується на кількох ключових компонентах, які забезпечують її функціональність і продуктивність. Як і в будь-якій інформаційній системі, вона складається з таких основних елементів: апаратне та програмне забезпечення, мережеві комунікації, дані, люди та процеси. Всі ці компоненти взаємодіють, щоб створити єдину інтегровану систему, здатну виконувати завдання з автоматизованої діагностики.

Система діагностики нейросудинних хвороб на основі КТ-зображень реалізується через інтеграцію кількох ключових компонентів. На першому етапі здійснюється збір [1-3] і підготовка даних, що включає формування бази КТ-знімків із патологіями, такими як ішемічний та геморагічний інсульти, аневризми, а також із зображеннями здорового мозку. Ці дані проходять попередню обробку:

ДОДАТОК Б

Програмний код системи

File «pre_processing.py»

```
import os
import cv2
import numpy as np
import albumentations as A

# Шляхи до датасету та вихідної директорії
dataset_path = 'D:/data/a/test/normal_test'  # Змініть на шлях до вашого
датасету
output_path = 'D:/data/b/test/normal_test'  # Директорія для збереження
оброблених та аугментованих зображень

# Створення директорії, якщо не існує
os.makedirs(output_path, exist_ok=True)
# Аугментація
transform = A.Compose([
    A.Rotate(limit=30, p=0.5),
    A.HorizontalFlip(p=0.5),
    A.VerticalFlip(p=0.5),
    A.RandomBrightnessContrast(p=0.5),
    A.RandomScale(scale_limit=0.2, p=0.5),
    A.ShiftScaleRotate(shift_limit=0.1, scale_limit=0.1, rotate_limit=20,
p=0.5),
    A.GaussianBlur(blur_limit=(1, 3), p=0.5),
    A.ImageCompression(quality_lower=85, quality_upper=95, p=0.5),
])

# Перебір зображень у директорії
for i, filename in enumerate(os.listdir(dataset_path)):
    if filename.endswith(('.jpg', '.jpeg', '.png')):
        img_path = os.path.join(dataset_path, filename)

        # Спроба завантажити зображення
        img = cv2.imread(img_path)
        if img is None:
            print(f"Не вдалося завантажити зображення: {img_path}")
            continue

        # Зміна розміру зображення до одного формату
        target_size = (200, 200)
        img_resized = cv2.resize(img, target_size)

        # Нормалізація зображення
        normalized_img = img_resized.astype('float32') / 255.0
```

```
        # Збереження обробленого зображення
        processed_save_path = os.path.join(output_path,
f'normal_test_{i}.jpg')
        cv2.imwrite(processed_save_path, (normalized_img *
255).astype(np.uint8))

        # Аугментація зображення кілька разів
        for j in range(0):  # Кількість аугментацій
            augmented = transform(image=(normalized_img *
255).astype(np.uint8))
            augmented_image = augmented['image']

            augmented_save_path = os.path.join(output_path,
f'normal_test_{i}_{j + 1}.jpg')
            cv2.imwrite(augmented_save_path, augmented_image)

print(f"Оброблені та аугментовані зображення збережено в {output_path}")
```

File «classification_model.py»

```
import tensorflow as tf
import numpy as np
import os
import cv2
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt

# Параметри
IMG_SIZE = (200, 200)
NUM_CLASSES = 4
BATCH_SIZE = 16
EPOCHS = 50
MODEL_SAVE_PATH = " classification_model.keras"

# Шляхи до папок із зображеннями
HEALTHY_FOLDER = "train/normal_train"
ISCHEMIC_FOLDER = "train/ischaemic_train"
HEMORRHAGIC_FOLDER = "train/hemorrhagic_train"
ANEURYSM_FOLDER = "train/aneurysm_train"

# Функція для попередньої обробки
def preprocess_image(img_path):
    img = cv2.imread(img_path)
    if img is None:
        return None
    resized_img = cv2.resize(img, IMG_SIZE)
    normalized_img = resized_img / 255.0
    return normalized_img
```

```
# Завантаження зображень
def load_images_and_labels(folder_path, label):
    images, labels = [], []
    for filename in os.listdir(folder_path):
        img_path = os.path.join(folder_path, filename)
        img = preprocess_image(img_path)
        if img is not None:
            images.append(img)
            labels.append(label)
    return np.array(images), np.array(labels)

# Завантаження всіх зображень і міток
def load_all_images():
    healthy_images, healthy_labels = load_images_and_labels(HEALTHY_FOLDER,
label=0)
    ischemic_images, ischemic_labels =
load_images_and_labels(ISCHEMIC_FOLDER, label=1)
    hemorrhagic_images, hemorrhagic_labels =
load_images_and_labels(HEMORRHAGIC_FOLDER, label=2)
    aneurysm_images, aneurysm_labels =
load_images_and_labels(ANEURYSM_FOLDER, label=3)

    X = np.concatenate((healthy_images, ischemic_images,
hemorrhagic_images, aneurysm_images), axis=0)
    y = np.concatenate((healthy_labels, ischemic_labels,
hemorrhagic_labels, aneurysm_labels), axis=0)
    return X, y

# Побудова класифікаційної моделі
def build_resnet_classification_model(input_shape=(200, 200, 3),
num_classes=NUM_CLASSES):
    base_model = tf.keras.applications.ResNet50(include_top=False,
input_shape=input_shape, weights='imagenet')
    base_model.trainable = False  # Заморожуємо шари ResNet50

    inputs = tf.keras.Input(shape=input_shape)
    x = base_model(inputs, training=False)
    x = tf.keras.layers.GlobalAveragePooling2D()(x)
    x = tf.keras.layers.Dropout(0.3)(x)
    outputs = tf.keras.layers.Dense(num_classes, activation='softmax')(x)

    model = tf.keras.Model(inputs, outputs)
    return model

# Навчання класифікаційної моделі
def train_resnet_classification_model():
    X, y = load_all_images()
    if X.shape[0] == 0 or y.shape[0] == 0:
```

```
        raise ValueError("Завантажено 0 зображень або міток. Перевірте
дані.")

    # Розбивка на тренувальний і валідаційний набори
    X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2,
random_state=42, stratify=y)

    # Побудова моделі
    model = build_resnet_classification_model()

    # Компіляція моделі
    model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])

    # Колбеки
    early_stop = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
patience=5, restore_best_weights=True)
    checkpoint = tf.keras.callbacks.ModelCheckpoint(MODEL_SAVE_PATH,
save_best_only=True, monitor='val_loss')

    # Навчання
    history = model.fit(X_train, y_train, batch_size=BATCH_SIZE,
epochs=EPOCHS,
                        validation_data=(X_val, y_val),
callbacks=[early_stop, checkpoint])

    print(f"Модель збережено у {MODEL_SAVE_PATH}")

    # Використання неінтерактивного бекенду
    import matplotlib
    matplotlib.use('Agg')
    import matplotlib.pyplot as plt

    # Побудова графіків
    plt.figure(figsize=(12, 4))

    plt.subplot(1, 2, 1)
    plt.plot(history.history['accuracy'], label='Train Accuracy')
    plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
    plt.xlabel('Епохи')
    plt.ylabel('Точність')
    plt.legend()
    plt.title('Точність навчання')

    plt.subplot(1, 2, 2)
    plt.plot(history.history['loss'], label='Train Loss')
    plt.plot(history.history['val_loss'], label='Validation Loss')
    plt.xlabel('Епохи')
```

```
    plt.ylabel('Втрати')
    plt.legend()
    plt.title('Втрати навчання')

    # Збереження у файл
    plt.savefig('training_results.png')
# Запуск навчання
train_resnet_classification_model()
```

File «model.py»

```
import tensorflow as tf
import numpy as np
import os
import cv2
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt

# Параметри
IMG_SIZE = (200, 200)
NUM_CLASSES = 4
BATCH_SIZE = 16
EPOCHS = 20
MODEL_SAVE_PATH = "neurovascular_model.h5"
HISTORY_SAVE_PATH = "training_history.npy"

# Шляхи до папок із зображеннями
HEALTHY_FOLDER = "train/normal_train"
ISCHEMIC_FOLDER = "train/ischaemic_train"
HEMORRHAGIC_FOLDER = "train/hemorrhagic_train"
ANEURYSM_FOLDER = "train/aneurysm_train"

# Функція для попередньої обробки
def preprocess_image(img_path):
    img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
    if img is None:
        return None
    resized_img = cv2.resize(img, IMG_SIZE)
    normalized_img = resized_img / 255.0
    return normalized_img

# Функція для генерації псевдомаски
def generate_pseudo_mask(image):
    _, thresh = cv2.threshold(image, 50, 255, cv2.THRESH_BINARY)
    kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (5, 5))
    cleaned = cv2.morphologyEx(thresh, cv2.MORPH_CLOSE, kernel)

    contours, _ = cv2.findContours(cleaned, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
```

```
    mask = np.zeros_like(image)

    if contours:
        largest_contour = max(contours, key=cv2.contourArea)
        cv2.drawContours(mask, [largest_contour], -1, 1, -1)

    return cv2.resize(mask, IMG_SIZE).astype(np.uint8)

# Завантаження зображень і створення псевдомасок
def load_images_and_generate_pseudo_masks(image_folder, label):
    images, masks = [], []
    for filename in os.listdir(image_folder):
        img_path = os.path.join(image_folder, filename)
        img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
        if img is None:
            print(f"Зображення {img_path} не знайдено. Пропускаємо.")
            continue

        processed_img = preprocess_image(img_path)
        pseudo_mask = generate_pseudo_mask(img)

        if processed_img is not None:
            images.append(processed_img)
            masks.append(pseudo_mask * label)

    return np.array(images), np.array(masks)

# Завантаження всіх зображень і псевдомасок
def load_all_images_with_pseudo_masks():
    healthy_images, healthy_masks =
load_images_and_generate_pseudo_masks(HEALTHY_FOLDER, label=0)
    ischemic_images, ischemic_masks =
load_images_and_generate_pseudo_masks(ISCHEMIC_FOLDER, label=1)
    hemorrhagic_images, hemorrhagic_masks =
load_images_and_generate_pseudo_masks(HEMORRHAGIC_FOLDER, label=2)
    aneurysm_images, aneurysm_masks =
load_images_and_generate_pseudo_masks(ANEURYSM_FOLDER, label=3)

    X = np.concatenate((healthy_images, ischemic_images,
hemorrhagic_images, aneurysm_images), axis=0)
    y = np.concatenate((healthy_masks, ischemic_masks, hemorrhagic_masks,
aneurysm_masks), axis=0)
    return X, y

# Модель U-Net
def build_unet(input_shape=(200, 200, 1), num_classes=NUM_CLASSES):
    inputs = tf.keras.Input(shape=input_shape)
```

```
    c1 = tf.keras.layers.Conv2D(64, (3, 3), activation='relu',
padding='same')(inputs)
    c1 = tf.keras.layers.BatchNormalization()(c1)
    p1 = tf.keras.layers.MaxPooling2D((2, 2), (2, 2), padding='same')(c1)

    c2 = tf.keras.layers.Conv2D(128, (3, 3), activation='relu',
padding='same')(p1)
    c2 = tf.keras.layers.BatchNormalization()(c2)
    p2 = tf.keras.layers.MaxPooling2D((2, 2), (2, 2), padding='same')(c2)

    c5 = tf.keras.layers.Conv2D(256, (3, 3), activation='relu',
padding='same')(p2)
    c5 = tf.keras.layers.BatchNormalization()(c5)

    u6 = tf.keras.layers.Conv2DTranspose(128, (2, 2), strides=(2, 2),
padding='same')(c5)
    u6 = tf.keras.layers.Concatenate()([u6, c2])
    c6 = tf.keras.layers.Conv2D(128, (3, 3), activation='relu',
padding='same')(u6)

    u7 = tf.keras.layers.Conv2DTranspose(64, (2, 2), strides=(2, 2),
padding='same')(c6)
    u7 = tf.keras.layers.Concatenate()([u7, c1])
    c7 = tf.keras.layers.Conv2D(64, (3, 3), activation='relu',
padding='same')(u7)

    outputs = tf.keras.layers.Conv2D(num_classes, (1, 1),
activation='softmax')(c7)
    model = tf.keras.Model(inputs=inputs, outputs=outputs)
    return model

# Метрика IoU
def iou_metric(y_true, y_pred):
    y_pred = tf.argmax(y_pred, axis=-1)
    y_true = tf.cast(y_true, tf.int64)
    ious = []
    for i in range(NUM_CLASSES):
        intersection = tf.reduce_sum(tf.cast((y_true == i) & (y_pred == i),
tf.float32))
        union = tf.reduce_sum(tf.cast((y_true == i) | (y_pred == i),
tf.float32))
        ious.append((intersection + 1e-10) / (union + 1e-10))
    return tf.reduce_mean(ious)

# Навчання моделі
def train_and_save_model():
    X, y = load_all_images_with_pseudo_masks()
    if X.shape[0] == 0 or y.shape[0] == 0:
```

```
        raise ValueError("Завантажено 0 зображень або масок. Перевірте
дані.")

    X = X[..., np.newaxis]
    X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2,
random_state=42)

    model = build_unet(input_shape=(200, 200, 1))
    model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
metrics=['accuracy', iou_metric])

    early_stop = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
patience=5, restore_best_weights=True)

    history = model.fit(X_train, y_train, batch_size=BATCH_SIZE,
epochs=EPOCHS, validation_data=(X_val, y_val),
                        callbacks=[early_stop])
    model.save(MODEL_SAVE_PATH)
    print(f"Модель збережено у {MODEL_SAVE_PATH}")

    # Збереження історії тренувань
    history_dict = history.history
    np.save(HISTORY_SAVE_PATH, history_dict)
    print(f"Історія тренувань збережена у {HISTORY_SAVE_PATH}")

# Запуск навчання
train_and_save_model()
```

File «interface.py»

```
import tkinter as tk
from tkinter import filedialog, Label, Button, Entry, IntVar, Checkbutton,
messagebox, Radiobutton
from tkcalendar import DateEntry
from PIL import Image, ImageTk
import tensorflow as tf
import numpy as np
import cv2

# Параметри
CLASSIFICATION_MODEL_PATH = "classification_model.keras"  # Модель для
класифікації
SEGMENTATION_MODEL_PATH = "neurovascular_model.h5"  # Модель для
сегментації
IMG_SIZE = (200, 200)
NUM_CLASSES = 4
CLASS_LABELS = ["Здоровий мозок", "Ішемічний інсульт", "Геморагічний
інсульт", "Аневризма"]
```

```
# Симптоми
SYMPTOMS_ANEURYSM = [
    "Головний біль", "Біль в області шиї або потилиці", "Судоми",
    "Підвищена чутливість до світла", "Погіршення слуху або дзвін у вухах"
]
SYMPTOMS_STROKE = [
    "Запаморочення", "Слабкість у кінцівках", "Порушення мови",
    "Оніміння обличчя, руки або ноги", "Асиметрія обличчя",
    "Втрата свідомості", "Нудота та блювання", "Проблеми із зором"
]
SYMPTOMS = SYMPTOMS_ANEURYSM + SYMPTOMS_STROKE

# Глобальні змінні
selected_symptoms_aneurysm = []
selected_symptoms_stroke = []
file_path = None

# Завантаження моделей
def load_classification_model():
    try:
        model = tf.keras.models.load_model(CLASSIFICATION_MODEL_PATH)
        return model
    except Exception as e:
        messagebox.showerror("Помилка", f"Не вдалося завантажити модель
класифікації: {str(e)}")
        exit()

def load_segmentation_model():
    try:
        model = tf.keras.models.load_model(SEGMENTATION_MODEL_PATH,
custom_objects={"iou_metric": iou_metric})
        return model
    except Exception as e:
        messagebox.showerror("Помилка", f"Не вдалося завантажити модель
сегментації: {str(e)}")
        exit()

# Метрика IoU
def iou_metric(y_true, y_pred):
    y_pred = tf.argmax(y_pred, axis=-1)
    y_true = tf.cast(y_true, tf.int64)
    ious = []
    for i in range(NUM_CLASSES):
        intersection = tf.reduce_sum(tf.cast((y_true == i) & (y_pred == i),
tf.float32))
        union = tf.reduce_sum(tf.cast((y_true == i) | (y_pred == i),
tf.float32))
        ious.append((intersection + 1e-10) / (union + 1e-10))
    return tf.reduce_mean(ious)
```

```
# Сегментація зображення
def segment_image(model, img_path):
    try:
        img = Image.open(img_path).convert("L").resize(IMG_SIZE)
        img_array = np.array(img) / 255.0
        img_array = img_array[np.newaxis, ..., np.newaxis]

        prediction = model.predict(img_array)
        predicted_mask = np.argmax(prediction[0], axis=-1)

        total_pixels = predicted_mask.size
        affected_pixels = np.sum(predicted_mask > 0)
        damage_percentage = (affected_pixels / total_pixels) * 100

        class_counts = np.bincount(predicted_mask.flatten(),
minlength=NUM_CLASSES)
        most_likely_class = np.argmax(class_counts[1:]) + 1 if
np.any(class_counts[1:] > 0) else 0
        pathology_label = CLASS_LABELS[most_likely_class]

        segmented_mask = (predicted_mask * (255 //
NUM_CLASSES)).astype(np.uint8)
        mask_colored = cv2.applyColorMap(segmented_mask, cv2.COLORMAP_JET)

        img_original = cv2.cvtColor(np.array(img), cv2.COLOR_GRAY2BGR)
        highlighted = cv2.addWeighted(img_original, 0.7, mask_colored, 0.3,
0)

        return Image.fromarray(cv2.cvtColor(highlighted,
cv2.COLOR_BGR2RGB)), damage_percentage, pathology_label
    except Exception as e:
        messagebox.showerror("Помилка", f"Помилка сегментації: {str(e)}")
        return None, 0, "Невідомо"

# Функція вибору зображення
def choose_image():
    global file_path
    file_path = filedialog.askopenfilename(filetypes=[("Image Files",
"*.png;*.jpg;*.jpeg")])
    if file_path:
        img = Image.open(file_path).resize((300, 300))
        img_tk = ImageTk.PhotoImage(img)
        input_image_label.configure(image=img_tk)
        input_image_label.image = img_tk


def generate_final_diagnosis(pathology_label):
    recommendations = ""
```

```
    if pathology_label == "Ішемічний інсульт":
        recommendations = "- Госпіталізація.\n- Лікування
антикоагулянтами.\n- Контроль артеріального тиску."
    elif pathology_label == "Геморагічний інсульт":
        recommendations = "- Негайна госпіталізація.\n- Контроль тиску.\n-
Нейрохірургічна консультація."
    elif pathology_label == "Аневризма":
        recommendations = "- Консультація судинного хірурга.\n- Хірургічне
лікування."
    elif pathology_label == "Здоровий мозок":
        recommendations = "Ніяких патологій не виявлено. Дотримуйтесь
здорового способу життя."
    return pathology_label, recommendations


# Функція перевірки
def segment_and_classify():
    if file_path:
        global selected_symptoms_aneurysm, selected_symptoms_stroke
        selected_symptoms_aneurysm = [symptom for i, symptom in
enumerate(SYMPTOMS_ANEURYSM) if symptom_vars[i].get() == 1]
        selected_symptoms_stroke = [symptom for i, symptom in
enumerate(SYMPTOMS_STROKE, len(SYMPTOMS_ANEURYSM)) if symptom_vars[i].get()
== 1]

        # Сегментація
        segmented_img, damage_percentage, pathology_label =
segment_image(segmentation_model, file_path)

        # Класифікація
        img_class = Image.open(file_path).resize(IMG_SIZE)
        img_class_array = np.array(img_class) / 255.0
        img_class_array = np.expand_dims(img_class_array, axis=0)

        classification_result =
classification_model.predict(img_class_array)
        class_idx = np.argmax(classification_result, axis=1)[0]
        classified_label = CLASS_LABELS[class_idx]

        # Генерація діагнозу та рекомендацій
        final_diagnosis, recommendations =
generate_final_diagnosis(classified_label)

        # Якщо виявлений "Здоровий мозок", але є вибрані симптоми, вивести
ймовірність патології
        if classified_label == "Здоровий мозок":
            if selected_symptoms_aneurysm or selected_symptoms_stroke:
                if len(selected_symptoms_aneurysm) >
len(selected_symptoms_stroke):
```

```
                    suspected_condition = "Підозра на аневризму"
                    final_diagnosis = f"На знімку КТ патологій не
виявлено.\n {suspected_condition} через наявність симптомів."
                    recommendations = (
                        "- Зверніться до лікаря для додаткових
обстежень.\n"
                        "- Проведіть МРТ головного мозку для уточнення
діагнозу.\n"
                        "- Консультація судинного хірурга чи нейрохірурга
для оцінки ризиків аневризми.\n"
                        "- Вести контроль за артеріальним тиском і серцевим
ритмом."
                    )
                else:
                    suspected_condition = "Підозра на інсульт"
                    final_diagnosis = f"На знімку КТ патологій не
виявлено.\n {suspected_condition} через наявність симптомів."
                    recommendations = (
                        "- Зверніться до лікаря для додаткових
обстежень.\n"
                        "- Провести МРТ для детальної діагностики.\n"
                        "- Консультація невролога для уточнення типу
інсульту.\n"
                        "- Спостереження за нейропсихологічним станом
пацієнта."
                    )

            # Якщо жодних симптомів не вибрано, можна вивести загальне
повідомлення про здоров'я:
            else:
                final_diagnosis = "На знімку КТ патологій не виявлено."
                recommendations = "- Потрібно регулярно проводити
профілактичні обстеження.\n- Слідкуйте за здоров'ям та ведіть активний
спосіб життя."

            # Якщо мозок здоровий, не виводити відсоток ураження
            damage_percentage = None

        if segmented_img:
            segmented_img = segmented_img.resize((200, 200))
            img_tk = ImageTk.PhotoImage(segmented_img)
            output_image_label.configure(image=img_tk)
            output_image_label.image = img_tk

        # Виведення результатів
        result_label.config(text=f"Діагноз: {final_diagnosis}")
        if damage_percentage is not None:
            damage_label.config(text=f"Ураження: {damage_percentage:.2f}%")
        else:
```

```
            damage_label.config(text="Ураження: Немає даних")

recommendation_label.config(text=f"Рекомендації:\n{recommendations}")
    else:
        messagebox.showwarning("Увага", "Оберіть зображення для
перевірки!")


# Зазначте функцію save_results перед її використанням:
def save_results():
    if not file_path:
        messagebox.showwarning("Увага", "Немає результатів для
збереження!")
        return

    name = name_entry.get().strip()
    dob = dob_entry.get_date()
    gender = "Чоловік" if gender_var.get() == 1 else "Жінка"
    if not name:
        messagebox.showwarning("Увага", "Будь ласка, введіть ПІБ
пацієнта!")
        return

    symptoms = ", ".join([symptom for i, symptom in enumerate(SYMPTOMS) if
symptom_vars[i].get() == 1])
    damage = damage_label.cget("text")
    diagnosis = result_label.cget("text")
    recommendations = recommendation_label.cget("text")

    save_path = filedialog.asksaveasfilename(defaultextension=".txt",
filetypes=[("Text Files", "*.txt")])
    if save_path:
        try:
            with open(save_path, "w") as f:
                f.write("Результати діагностики\n")
                f.write(f"ПІБ пацієнта: {name}\n")
                f.write(f"Дата народження: {dob}\n")
                f.write(f"Стать: {gender}\n")
                f.write(f"Обрані симптоми: {symptoms if symptoms else 'Не
вказано'}\n")
                f.write(f"Обране зображення: {file_path}\n")
                f.write(f"{damage}\n")
                f.write(f"{diagnosis}\n")
                f.write(f"{recommendations}\n")
            messagebox.showinfo("Успіх", "Результати успішно збережено!")
        except Exception as e:
            messagebox.showerror("Помилка", f"Не вдалося зберегти
результати: {str(e)}")
```

```
# Головне вікно
root = tk.Tk()
root.title("Діагностика захворювань мозку")
root.geometry("1000x750")
root.configure(bg="#bfd8d1")

# Ліва панель
left_frame = tk.Frame(root, bg="#bfd8d1", width=300)
left_frame.pack(side="left", fill="y", padx=10, pady=10)

tk.Label(left_frame, text="ПІБ пацієнта:", font=("Arial", 15),
bg="#bfd8d1", fg="black").pack(anchor="w", pady=5)
name_entry = Entry(left_frame, font=("Arial", 10))
name_entry.pack(fill="x", pady=5)

tk.Label(left_frame, text="Дата народження:", font=("Arial", 15),
bg="#bfd8d1", fg="black").pack(anchor="w", pady=5)
dob_entry = DateEntry(left_frame, font=("Arial", 10),
date_pattern="dd/mm/yyyy")
dob_entry.pack(fill="x", pady=5)

tk.Label(left_frame, text="Стать:", font=("Arial", 15), bg="#bfd8d1",
fg="black").pack(anchor="w", pady=5)
gender_var = IntVar(value=1)
Radiobutton(left_frame, text="Чоловік", variable=gender_var, value=1,
font=("Arial", 12), bg="#bfd8d1").pack(anchor="w")
Radiobutton(left_frame, text="Жінка", variable=gender_var, value=2,
font=("Arial", 12), bg="#bfd8d1").pack(anchor="w")

tk.Label(left_frame, text="Симптоми:", font=("Arial", 15, "bold"),
bg="#bfd8d1", fg="black").pack(anchor="w", pady=10)
symptom_vars = [IntVar() for _ in SYMPTOMS]
for i, symptom in enumerate(SYMPTOMS):
    Checkbutton(left_frame, text=symptom, variable=symptom_vars[i],
font=("Arial", 12), bg="#bfd8d1", fg="black").pack(anchor="w")

# Права панель
right_frame = tk.Frame(root, bg="#bfd8d1")
right_frame.pack(side="right", fill="y", expand=True, padx=10, pady=10)

Button(right_frame, text="Оберіть зображення", command=choose_image,
bg="#0078D7", fg="white").pack(pady=10)

input_image_label = Label(right_frame, bg="gray", width=300, height=300)
input_image_label.pack(pady=10)

Button(right_frame, text="Перевірити", command=segment_and_classify,
bg="#0078D7", fg="white").pack(pady=10)
```

```
output_image_label = Label(right_frame, bg="gray", width=300, height=300)
output_image_label.pack(pady=10)

# Права панель
middle_frame  = tk.Frame(root, bg="#bfd8d1")
middle_frame .pack(side="left", fill="y", expand=True, padx=10, pady=10)

result_label = Label(middle_frame , text="", bg="#bfd8d1", fg="black",
font=("Arial", 15))
result_label.pack(pady=5)

damage_label = Label(middle_frame , text="", bg="#bfd8d1", fg="black",
font=("Arial", 15))
damage_label.pack(pady=5)

recommendation_label = Label(middle_frame , text="", bg="#bfd8d1",
fg="black", font=("Arial", 15), wraplength=500, justify="left")
recommendation_label.pack(pady=10)

Button(middle_frame , text="Зберегти результати", command=save_results,
bg="#0078D7", fg="white").pack(pady=10)

# Завантаження моделі
classification_model = load_classification_model()
segmentation_model = load_segmentation_model()

# Запуск
root.mainloop()
```