

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ
СІКОРСЬКОГО»**

Навчально-науковий інститут атомної та теплової енергетики
Кафедра інженерії програмного забезпечення в енергетиці

ДО ЗАХИСТУ ДОПУЩЕНО

В.о. завідувача кафедри

_____ Олександр КОВАЛЬ

« ____ » _____ 2026 р.

Дипломна робота

**на здобуття ступеня бакалавра
за освітньо-професійною програмою «Інженерія програмного забезпечення
інтелектуальних кібер-фізичних систем в енергетиці»
спеціальності 121 Інженерія програмного забезпечення
на тему: «Система аналізу КТ-знімків для виділення структур колінного
суглобу»**

Виконав:

студент IV курсу, групи ТВ-22

Оніщук Максим Іванович

(прізвище, ім'я, по батькові)

_____ (підпис)

Керівник:

старший викладач Дацюк О.А.

(посада, науковий ступінь, вчене звання, прізвище та ініціали)

_____ (підпис)

Консультант з назва розділу:

доцент, к.т.н. Залевська О.В.

(посада, науковий ступінь, вчене звання, прізвище та ініціали)

_____ (підпис)

Рецензент:

_____ (посада, науковий ступінь, вчене звання, прізвище та ініціали)

_____ (підпис)

Засвідчую, що у цій дипломній роботі
немає запозичень із праць інших авторів
без відповідних посилань.

Студент

_____ (підпис)

Київ – 2026

**Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»**

Навчально-науковий інститут атомної та теплової енергетики

Кафедра інженерії програмного забезпечення в енергетиці

Рівень вищої освіти перший (бакалаврський)

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення
інтелектуальних кібер-фізичних систем в енергетиці»

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри

_____ Олександр КОВАЛЬ

«___» _____ 2026р.

ЗАВДАННЯ

на дипломну роботу студенту

_____ Оніщуку Максиму Івановичу

(прізвище, ім'я, по батькові)

1. Тема роботи: Система аналізу КТ-знімків для виділення структур колінного суглобу

керівник роботи Дацюк Оксана Антонівна, старший викладач

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «28» травня 2026 р. № 1992-с

2. Строк подання студентом роботи «10» червня 2026 р.

3. Вихідні дані до роботи: мова програмування Python, середовище розробки PyCharm, набір знімків магнітно-резонансної томографії колінних суглобів SKI10

4. Зміст (дипломної роботи) пояснювальної записки (перелік завдань, які потрібно розробити): провести аналіз предметної області та відповідної літератури, дослідити наявні методи обробки та сегментації медичних знімків, розробити архітектуру нейронної мережі для сегментації зображень та реалізувати алгоритм навчання, написати код системи

5. Перелік ілюстративного матеріалу: знімки програмного продукту, знімки магнітно-резонансної томографії колін, знімки результатів навчання моделі

6. Дата видачі завдання «31» жовтня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Строки виконання етапів роботи	Примітка
1	Отримання завдання	31.10.2025	
2	Дослідження предметної області	01.11.2025 – 31.11.2025	
3	Дослідження існуючих рішень	01.12.2025 – 31.12.2025	
4	Постановка вимог до проектування системи	01.01.2026 – 31.01.2026	
5	Розробка програмного продукту	01.02.2026 – 03.05.2026	
6	Тестування	04.05.2026 – 10.05.2026	
7	Захист програмного продукту	11.05.2026 – 15.05.2026	
8	Оформлення дипломної роботи	18.05.2026 – 31.05.2026	
9	Передзахист	01.06.2026 – 05.06.2026	
10	Захист	15.06.2026 – 19.06.2026	

Студент

(підпис)

Максим ОНІЩУК

(ім'я, ПРІЗВИЩЕ)

Керівник роботи

(підпис)

Оксана ДАЦЮК

(ім'я, ПРІЗВИЩЕ)

РЕФЕРАТ

Структура та обсяг дипломної роботи. Робота містить 65 сторінки, 35 рисунків, 1 таблицю, 2 додатки та 21 посилання.

Метою роботи є розробка системи аналізу МРТ-знімків для виділення структур колінного суглобу за допомогою нейронних мереж.

Об'єктом роботи є процес семантичної сегментації КТ та МРТ-знімків за допомогою нейронних мереж.

Предметом роботи є модель нейронної мережі для сегментації медичних зображень та алгоритм її навчання.

Завдання роботи полягає у аналізі предметної області даної роботи, пошуку та підготовці набору даних для навчання моделі, створенні архітектур моделі нейронної мережі та самої системи аналізу, навчанні моделі сегментації та її тестуванні, а також розробці графічного інтерфейсу користувача.

Практичне значення одержаних результатів полягає в можливості використання нашої системи у сфері медицини для дослідження структури коліна, за допомогою нейронної мережі для семантичної сегментації МРТ-знімків колінного суглобу.

Розроблений програмний продукт дає змогу автоматично аналізувати та виділяти анатомічні структури коліна, такі як кістки та меніски, з подальшим збереженням результату

Ключові слова: Машинне навчання, нейронна мережа, обробка зображень, DICOM, Python, МНД, магнітно-резонансна томографія, семантична сегментація, колінний суглоб, датасет.

ABSTRACT

Structure and scope of the thesis. The work contains 65 pages, 35 figures, 1 table, 2 appendices and 21 references.

The purpose of the work is to develop a system for analyzing MRI images to identify knee joint structures using neural networks.

The object of the work is the process of semantic segmentation of CT and MRI images using neural networks.

The subject of the work is a neural network model for medical image segmentation and the algorithm for training it.

Tasks of the work are to analyze the subject area of this study, search for and prepare a dataset for training the model, create the architecture of the neural network model and the analysis system itself, train and test the segmentation model, and develop a graphical user interface.

The practical value of the results lies in the potential application of our system in the medical field for studying knee anatomy, using a neural network for the semantic segmentation of MRI images of the knee joint.

The developed software product enables the automatic analysis and identification of anatomical structures of the knee, such as bones and menisci, with subsequent saving of the results

Keywords: Machine learning, neural network, image processing, DICOM, Python, MHD, magnetic resonance imaging, semantic segmentation, knee joint, dataset.

.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ	8
ВСТУП.....	9
1 ПОСТАНОВКА ЗАВДАННЯ	11
1.1 Аналіз методів сегментації	12
1.2 Вимоги до системи	13
Висновки до розділу 1	15
2 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ	17
2.1 3D Slicer	17
2.2 ITK-Snap.....	19
2.3 Наявні архітектури моделей нейромереж для задачі сегментації.....	20
Висновки до розділу 2.....	25
3 ЗАСОБИ ТА ЕТАПИ РОЗРОБКИ.....	26
3.1 Етапи розробки системи.....	26
3.2 Засоби розробки системи	29
3.2.1 Мова програмування Python.....	30
3.2.2 Pydicom.....	30
3.2.3 NumPy.....	31
3.2.4 PyTorch	32
3.2.5 Matplotlib	33
3.2.6 PyQt6.....	33
3.2.7 CNN та архітектура U-Net	34
3.2.8 Інші засоби	36
3.3 Набір даних SKI10 Dataset	37
Висновки до розділу 3	38
4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ.....	40
4.1 Взаємодія користувача та системи	40
4.2 Архітектура системи	41
4.2.1 Модуль даних.....	43
4.2.2 Модуль машинного навчання	45
4.2.3 Модуль інтерфейсу	52

4.3 Проблеми системи	54
Висновки до розділу 4.....	56
5 ВЗАЄМОДІЯ КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ.....	57
5.1 Завантаження медичних знімків	58
5.2 Сегментація КТ та МРТ-знімків	58
5.3 Збереження результатів.....	60
Висновки до розділу 5.....	61
ВИСНОВКИ	62
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	63

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ML	Machine Learning (Машинне навчання)
ШІ	Штучний інтелект
MPT	Магнітно-резонансна терапія
КТ	Комп'ютерна томографія
УЗД	Ультразвукова терапія
DICOM	Digital Imaging and Communications in Medicine (Цифрові зображення та комунікації в медицині)
CPU	Central Processing Unit (Центральний процесор)
GPU	Graphics Processing Unit (Графічний процесор)
CNN	Convolutional Neural Network (Згорткова нейронна мережа)
UML	Unified Modeling Language (Уніфікована мова моделювання)
2D	2-Dimensional (Двохвимірний)
3D	3-Dimensional (Трьохвимірний)
4D	4-Dimensional (Чотирьохвимірний)
PNG	Portable Network Graphics
MHD	MetaImage Header Data
RAW	Raw Data (Сирі пікселі)
PSPNet	Pyramid Scene Parsing Network (Мережа пірамідального розбору сцени)
FCN	Fully Convolutional Network (Повністю згорткова мережа)
IoU	Intersection over Union (Перетин над об'єднанням)

ВСТУП

Останні декілька років технології штучного інтелекту розвиваються дуже стрімко. За допомогою різноманітних чат-ботів, які побудовані на основі великих мовних моделей, будь то ChatGPT, Gemini тощо, нейронні мережі змогли перейти від суто професійного використання до повсякденного, здобувши дуже велику популярність серед звичайних користувачів мережі Інтернет.

В сучасних умовах стрімкого розвитку технологій медична діагностика стикається з постійно зростаючим обсягом даних та підвищеними вимогами точності та швидкості аналізу. Для того, щоб залишатись конкурентоспроможними, медичні заклади займаються пошуком інструментів, які допомогли б автоматизувати рутинні процеси, скоротити час обробки знімків та знизити навантаження на медичний персонал. Саме тому застосування методів штучного інтелекту у медичній візуалізації стає не просто перспективним напрямком, а необхідним кроком до підвищення якості діагностичної допомоги [1].

Загалом штучний інтелект використовується в широкому спектрі сфер: від мистецтва та сільського господарства до робототехніки та медицини. Зокрема машинне навчання, яке на сьогоднішній є дуже розповсюдженим розділом ШІ та методи якого було застосовано при розробці нашої системи.

Однією з основоположних методів машинного навчання є глибоке навчання, а саме нейронні мережі. Принцип їх побудови запозичений із біології — архітектура таких мереж імітує роботу нервової системи людини, що і відображено в їхній назві. Подібно до того, як людина набуває знань через досвід, нейронні мережі адаптуються та вдосконалюють свої параметри на основі вхідних даних за допомогою спеціальних алгоритмів навчання. Варто зазначити, що нейронні мережі є стохастичними за своєю суттю: навіть за однакових вхідних даних та однакової архітектури кожен новий запуск навчання може давати модель з дещо відмінними характеристиками продуктивності [2].

Аналіз медичних зображень є важливою складовою сучасної діагностики. Ручне виділення анатомічних структур на МРТ-знімках потребує значних часових витрат та залежить від досвіду спеціаліста.

Тому актуальним є використання нейронних мереж для автоматизації цього процесу.

Розроблена система виділення структур колінного суглобу має потенціал стати хорошим помічником для лікарів для обробки медичних зображень, напівавтоматичної розмітки даних та візуалізації структур коліна.

У процесі виконання практики було реалізовано модулі завантаження та обробки медичних даних у форматах DICOM та MHD, а також алгоритми попередньої підготовки зображень. Для виконання семантичної сегментації було розроблено нейронну мережу типу U-Net. Крім цього, створено графічний інтерфейс користувача з можливістю перегляду томографічних зрізів, навігації між ними, накладання сегментаційних масок та збереження результатів у форматі PNG.

1 ПОСТАНОВКА ЗАВДАННЯ

Метою нашої роботи є створення програмного продукту для автоматизованого аналізу КТ та МРТ знімків колінного суглобу шляхом використання методів глибокого навчання.

Сегментація відіграє важливу роль у аналізі медичних зображень, будь то рентгенівські знімки, КТ чи МРТ. Даний метод дослідження зображень підвищує якість діагностики, дозволяючи лікарям правильно ідентифікувати різні новоутворення або пошкодження в тілі людини [3].

Наша система має виконувати сегментацію МРТ та КТ знімків за допомогою моделі нейронної мережі, навченої набором сегментованих медичних знімків.

Розроблювальна система повинна працювати з двома наступними форматами зберігання даних:

- DICOM (Digital Imaging and Communications in Medicine) – даний формат є міжнародним стандартом зберігання медичних знімків (КТ, МРТ, УЗД, рентген). Даний тип містить в собі не лише зображення, але й дані пацієнта та інформацію про дослідження.

- MHD – це текстові файли-заголовки медичних зображень, що містять метадані медичних зображень. Він не зберігає безпосередньо піксельні дані, а описує характеристики відповідного RAW-файлу – розмірність зображення, відстань між вокселями, тип та формат збережених даних. Пікселі зображення зберігаються в окремому RAW-файлі, на який посилається MHD-заголовок [4].

Окрім того має бути реалізований механізм зберігання результату роботи системи виділення у вигляді сегментованих 2D-знімків, які є зрізами попередньо завантаженого 3D-знімку користувачем.

Для взаємодії користувача з системою потрібно розробити графічний інтерфейс користувача, який повинен забезпечувати функціонал, вказаний вище. Окрім того, має бути реалізовано візуалізацію завантажених МРТ та КТ знімків,

результатів їх сегментації та вивід інформації про дослідження. Актуальність завдання

1.1 Аналіз методів сегментації

Сегментація зображень є методом комп'ютерного зору, що полягає у розбитті зображення на окремі області на основі таких характеристик, як колір, інтенсивність або текстура пікселів. Ціль сегментації зображень — спрощення аналізу зображення шляхом виділення змістовних областей, що полегшує подальшу обробку та інтерпретацію даних. Результатом виконання задачі сегментації є маска — зображення, в якому кожному пікселю присвоєно мітку класу, до якого він належить [5].

В основному, можна виділити три категорії завдань сегментації:

1. Семантична сегментація (англ. Semantic segmentation) передбачає класифікацію кожного пікселя зображення до певного класу, при цьому всі об'єкти одного типу отримують однакову мітку без розрізнення окремих екземплярів. Саме цей тип сегментації застосовується у розробленій системі.

2. Сегментація екземплярів (англ. Instance segmentation) є розширенням семантичного підходу — вона не лише визначає клас кожного пікселя, а й розрізняє окремі екземпляри об'єктів одного класу, присвоюючи кожному з них унікальну мітку.

3. Паноптична сегментація (англ. Panoptic segmentation) поєднує переваги обох попередніх підходів, формуючи повністю сегментоване зображення, на якому одночасно визначено класи всіх пікселів та виділено кожен окремий об'єкт сцени.

На рисунку 1.1 зображено приклади сегментацій:

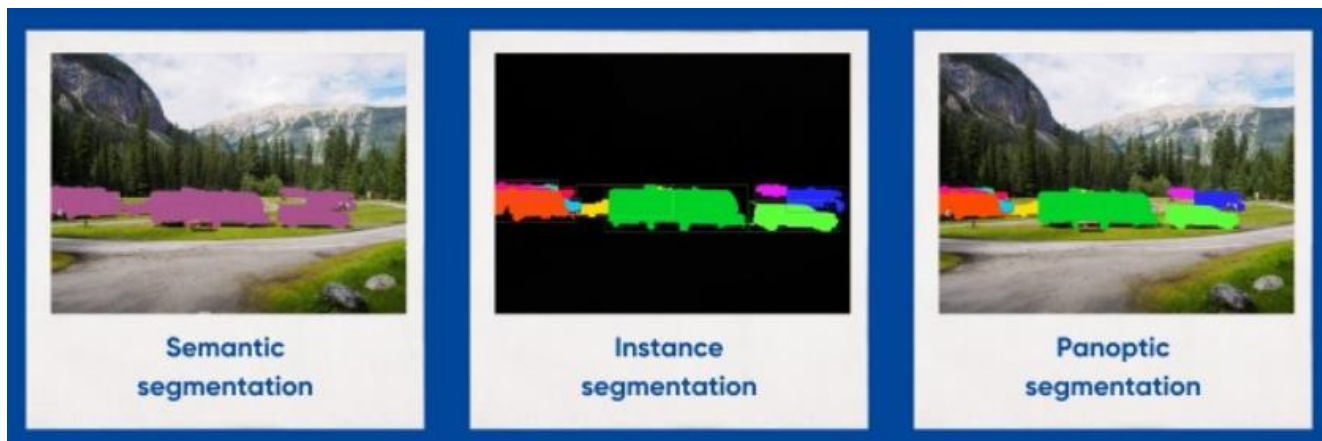


Рисунок 1.1 – Типи сегментації зображень

Окрім того, можна виділити два типи сегментації за кількістю класів які будуть виділятися:

1. Бінарна сегментація – даний тип, який працює лише з двома класами: об’єкт та фон.

2. Мультикласова сегментація – кожен піксель може належати одному з декількох класів. В контексті нашої роботи саме цей тип буде використовуватись та матиме наступні класи для класифікації:

- I. 0 – фон
- II. 1 – верхня кістка
- III. 2 – верхній меніск
- IV. 3 – нижній меніск
- V. 4 – нижня кістка

1.2 Вимоги до системи

Насамперед, потрібно було визначити перелік ключових функціональних та нефункціональних вимог до системи. Основні зусилля були направлені на розробку архітектури нейронної мережі та створення алгоритму її навчання.

Для нашої системи виділення структур колінного суглобу було сформовано наступні вимоги:

- Функціональні вимоги:
 - Можливість завантаження у систему медичних знімків типу DICOM та MHD;
 - Коректне витягування 3D/2D даних з файлів;
 - Витягування даних про пацієнта та обстеження з DICOM-файлів;
 - Виконання автоматичної сегментації медичних зображень на наступні структури:
 - Верхня кістка
 - Верхній меніск
 - Нижній меніск
 - Нижня кістка
 - Візуалізація завантажених медичних даних та результатів сегментації;
 - Збереження результатів сегментації у вигляді зрізів з форматом PNG;
 - Реалізація структури нейронної мережі та розробка алгоритму навчання і тестування моделі.
- Нефункціональні вимоги:
 - Точність моделі: система повинна забезпечувати достатньо непогану якість сегментації. Оптимізація за метриками Dice (Diverse Counterfactual Explanations) та IoU (Intersection over Union);
 - Продуктивність: система повинна підтримувати обробку великих 3D-об'ємів (100+ зрізів) та обробка одного об'єму в розумний час (залежно від CPU/GPU);
 - Вимоги до моделі: використання CNN/U-Net або подібної архітектури з підтримкою мультикласової сегментації та роботи з дизбалансованими даними (меніски дуже малі на знімках та важко розрізняються);

- Вимоги до обробки даних: підтримка роботи з 3D-знімками (наприклад shape: (67, 357, 318)), розбиття даних на тренувальні, валідаційні та тестувальні. Узгодження розмірів вхідних зображень та масок;
- Візуалізація та графічний інтерфейс: інтерфейс повинен забезпечувати зрозумілий вивід результату сегментації, виводити індикатор прогресу сегментації та можливість переглядати всі зрізи знімку;
- Стабільність моделі: уникнення ситуацій з виникненням артефактів масок, підгін розмірів знімків під потрібний розмір для роботи та обробка випадків з відсутністю об'єктів на зображенні.

Висновки до розділу 1

У першому розділі було здійснено аналіз особливостей задачі сегментації медичних знімків.

Було встановлено, що використання методів глибокого навчання є достатньо ефективним підходом для автоматизації процесу аналізу медичних зображень тому, що дозволяє виконувати виділення анатомічних структур без участі самого спеціаліста. Окрім того, було сформульовано основні вимоги до розроблювальної системи виділення структур колінного суглобу на основі нейронної мережі:

1. Підтримка роботи з медичними даними типу DICOM та MHD;
2. Обробка та нормалізація даних для подачі моделі сегментації;
3. Реалізація архітектури моделі нейронної мережі для сегментації зображень;
4. Написання алгоритму навчання та тестування моделі;
5. Розробка користувацького інтерфейсу з візуалізацією медичних знімків та результатів їх сегментації;
6. Можливість збереження результатів роботи системи.

Також було розглянуто основні типи сегментації зображень, серед яких для поставленої нам задачі було обрано мультикласову семантичну сегментацію, що виділяє фон, верхню та нижню кістки, а також верхній і нижній меніск.

Сформульовані нами вимоги та проведений аналіз предметної області створюють умови для подальшого проектування нашої архітектури моделі сегментації МРТ та КТ-знімків та розробки алгоритму для її навчання та тестування.

2 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

Перед початком розробки власної системи виділення структур колінного суглобу, було б доречно здійснити ознайомлення з уже наявними системами, які здатні візуалізувати, аналізувати та сегментувати МРТ та КТ-знімки. Даний процес дозволить краще спланувати наш програмний продукт, який можна було б використати в якості альтернативи.

Одними з найпопулярніших програмних рішень є програми 3D Slicer та ІТК-Snap

2.1 3D Slicer

3D Slicer - це безкоштовне програмне забезпечення з відкритим вихідним кодом призначене яке слугує для візуалізації, сегментації, реєстрації та 3D моделювання медичних знімків. Даний застосунок використовується активно у наукових дослідженнях, медичних установах та освітніх проєктах, оскільки має досить широкий набір інструментів для роботи з різноманітними медичними даними.

Розвиток даного продукту став можливим завдяки спільним зусиллям розробників, дослідників та комерційних партнерів з різних куточків світу, які долучились до його вдосконалення через розширення набору даних, виправлення помилок та надання пропозицій щодо покращення. Окрему роль у підтримці проєкту відіграли Національні Інститути Здоров'я США, які забезпечували фінансування у вигляді грантів та контрактів на конкурсній основі [6].

3D Slicer є багатофункціональною платформою для обробки та аналізу медичних зображень, що підтримує роботу зі знімками МРТ та КТ. Система забезпечує повноцінну роботу з форматом DICOM та іншими медичними форматами даних, надаючи інструменти для імпорту, експорту та візуалізації зображень у двовимірному, тривимірному та чотиривимірному представленні.

Також, за допомогою модуля Segment Editor, який підтримує ручне, напівавтоматичне та автоматичне виділення анатомічних структур. Технології ШІ, зокрема TensorFlow, MONAI та NVIDIA ClaraAI, використовуються для підвищення ефективності обробки даних.

- Також дана система має наступні корисні інструменти:
- Просторова реєстрація зображень;
- Створення тривимірних позначок і вимірювань;
- Підтримка виконання Python-скриптів;
- Підтримка експортування до 3D-принтерів;
- Встановлення додаткових модулів через систему розширень;

На рисунку 2.1 зображено інтерфейс користувача програми 3D Slicer:

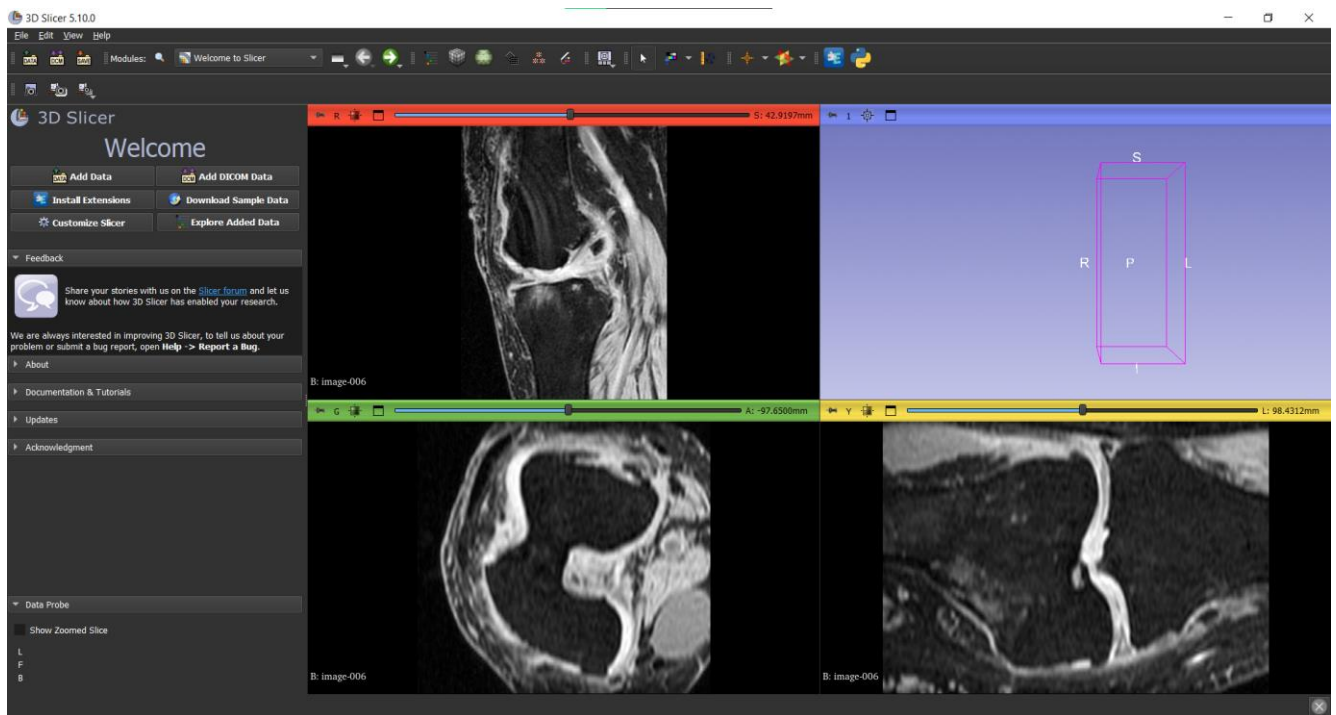


Рисунок 2.1 – Інтерфейс 3D Slicer

Завдяки широкому спектру доступних інструментів, 3D Slicer є одним із найпоширеніших програмних засобів для роботи з медичними даними, в тому числі з задачею сегментації та візуалізації МРТ- та КТ-знімків.

Однак, для нашого завдання сегментації структур колінного суглобу, серед розширень не було знайдено моделі.

2.2 ITK-Snap

ITK-Snap – це інтерактивний програмний продукт з відкритим кодом, який є кросплатформним програмним забезпеченням і використовується для сегментації медичних знімків, який надає середовище для візуалізації складних 3D та чотиривимірних (4D) наборів даних знімків, а також набір інструментів для ефективного маркування структур що цікавлять. Філософія розробки ITK-Snap полягає в тому, що інструмент повинен бути простим у вивченні та придатним для якомога більшої кількості застосувань [7].

Розробка даної системи бере свій початок в Університеті Північної Кароліни, де студентські команди під керівництвом професора Гвідо Геріга з Інженерної школи Танден Нью-Йоркського університету заклали основи її архітектури та функціоналу. Основна концепція полягала в створенні інструменту з легким освоєнням, обмеженим набором функцій, зосереджених саме на сегментації зображень [8].

ITK-Snap забезпечує напівавтоматичну сегментацію, за допомогою метод активних контурів, а також ручне розмежування та навігацію по зображеннях. На додаток до цього, дана система пропонує багато інших утиліт. Деякі з основних можливостей включають:

- Ручна сегментація одночасно в трьох ортогональних площинах;
- Підтримка багатьох різних форматів 3D-знімків, в особливості NIfTI та DICOM;
- Можливість ручної та автоматичної реєстрації знімків;
- Інтерполяція міжзрізової сегментації;
- Інструмент 3D-площини зрізу для швидкої постобробки результатів сегментації.

На рисунку 2.2 наведено інтерфейс користувача застосунку ITK-Snap:

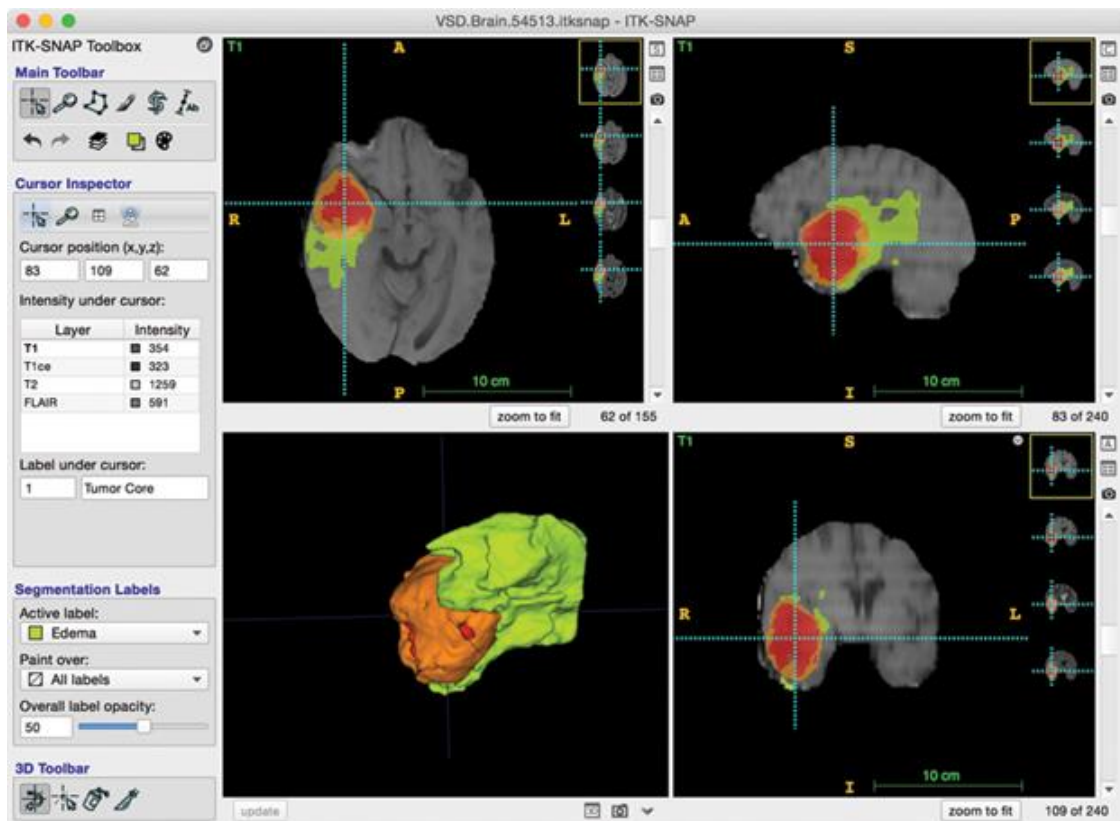


Рисунок 2.2 – Інтерфейс ITK-Snap

Інтерфейс ITK-Snap є простішим ніж у 3D Slicer, що робить його більш зручним та дружелюбним до новачків . Водночас його функціональність є дещо вужчою, оскільки вона орієнтована переважно на сегментацію та візуалізацію медичних зображень і не містить такого широкого набору інструментів для аналізу даних та інтеграції моделей ШІ, як 3D Slicer.

2.3 Наявні архітектури моделей нейромереж для задачі сегментації

Тепер перейдемо від огляду повноцінних програмних систем до моделей та їх архітектур для задачі сегментації зображень. У сучасних системах комп'ютерного зору для розпізнавання об'єктів широко використовуються згорткові нейронні мережі (CNN). Основною метою даних таких архітектур є класифікація кожного пікселя та формування маски сегментації, яка визначає

належність до певного класу. У даному підрозділі буде розглянуто лише декілька найвідоміших моделей.

Одним з перших успішних архітектур для задачі семантичної сегментації стала Fully Convolutional Network (FCN). На відміну від класичних мереж для класифікації, FCN використовує лише згорткові шари та дозволяє отримувати карту сегментації того ж розміру, що й вхідне зображення. Дана архітектура має обмежену точність виділення меж об'єктів.

PSPNet (Pyramid Scene Parsing Network) — це архітектура нейронної мережі, в основі якої лежить модуль пірамідального об'єднання. Завдяки даному модулю мережа здатна враховувати контекстну інформацію різних масштабів одночасно, що дозволяє підвищити точність сегментації при роботі зі складними сценами [9].

Схема даної архітектури наведена на рисунку 2.3:

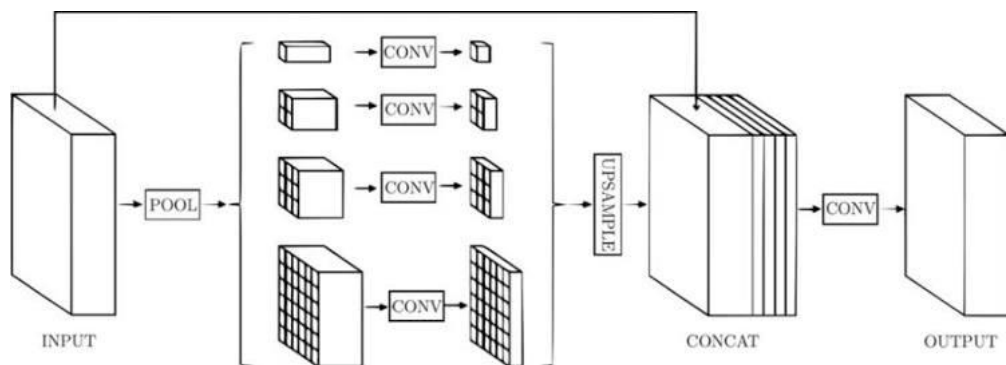


Рисунок 2.3 – Схема PSPNet

Також можна виділити архітектуру DeepLab V3. На сьогоднішній день вона є однією з найпотужніших архітектур семантичної сегментації. Дана архітектура складається з двох основних частин: енкодера та декодера. Енкодер потрібен для виділення ознак з вхідного зображення, а декодер — для відновлення просторової інформації та формування фінальної маски сегментації з межами об'єктів.

Ключовою особливістю DeepLab V3 є використання атрибутивних (англ. dilated) згорток. На відміну від звичайних згорток, вони дозволяють збільшувати область огляду мережі без збільшення кількості параметрів та втрати роздільної здатності. Завдяки цьому модель може аналізувати як дрібні деталі, так і великі структури на зображенні. Схема архітектури наведено нижче на рисунку 2.4:

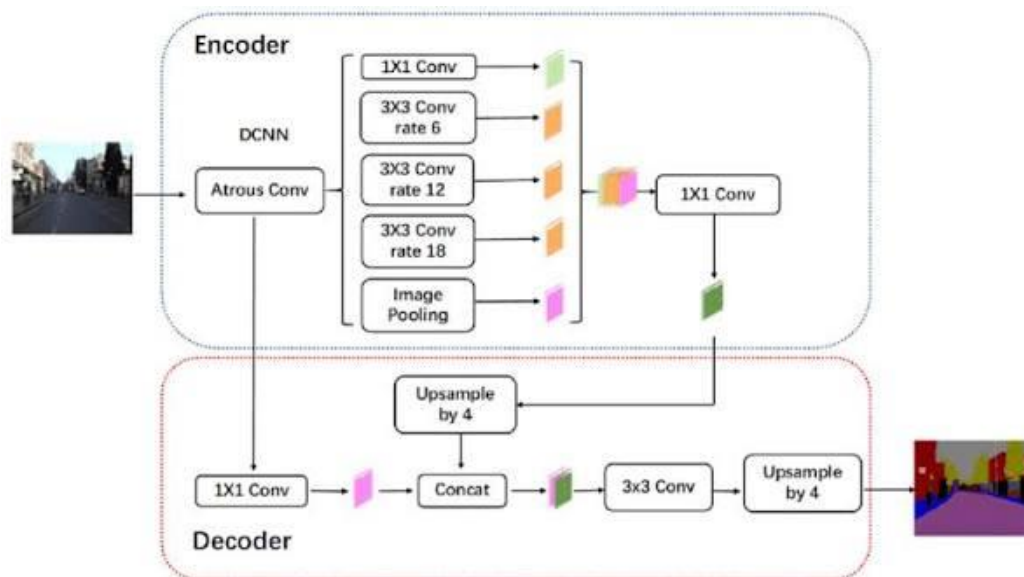


Рисунок 2.4 – Архітектура DeepLab V3

Останньою мережею яку ми б хотіли розглянути – це U-Net. Назва даної архітектури походить від її симетричної форми, яка схожа на англійську літеру «U». На сьогоднішній день, U-Net вважається однією зі стандартних архітектур нейронних мереж для задач сегментації. U-Net був створений дослідниками з Університету Фрайбурга у 2015 році, на основі раніше згаданої моделі FCN. Основною областю застосування архітектури U-Net є сегментація біомедичних зображень. Мережа була спеціально розроблена з урахуванням особливостей медичної сфери, де кількість розмічених навчальних даних, як правило, є суттєво обмеженою порівняно з іншими галузями комп'ютерного зору [10].

Схема даної мережі зображена на рисунку 2.5:

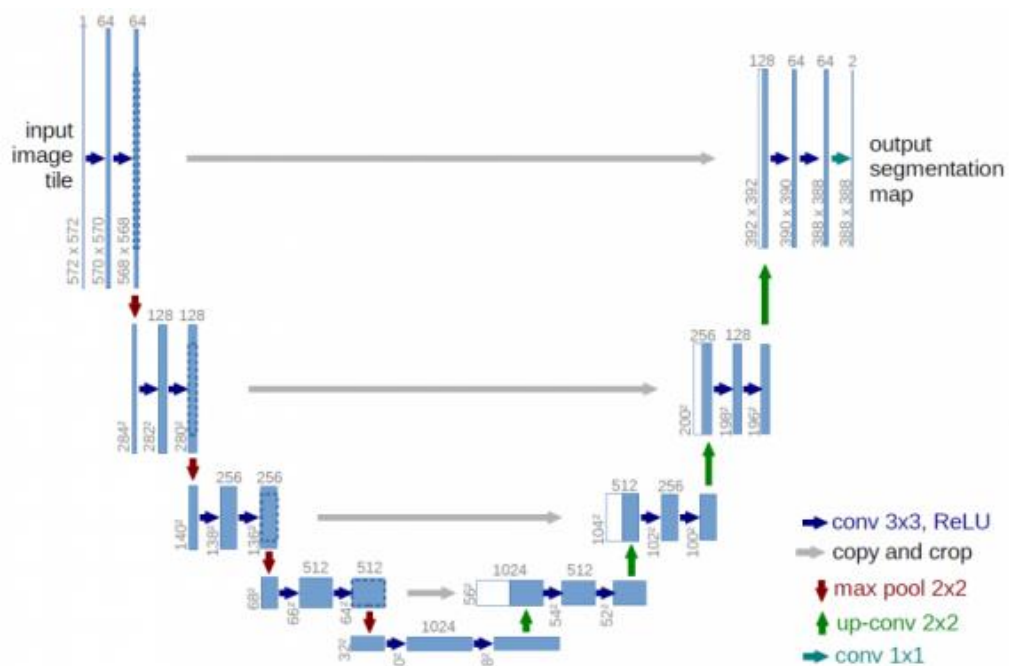


Рисунок 2.5 – Архітектура U-Net

Архітектура U-Net побудована за принципом згорткової структури енкодер-декодер. На відміну від традиційних CNN, які втрачають просторову інформацію в процесі послідовного зменшення розмірності, U-Net вирішує проблему точної локалізації об'єктів завдяки механізму skip-з'єднань, що передають карти ознак з відповідних рівнів енкодера безпосередньо до декодера, зберігаючи таким чином деталізовану просторову інформацію [11]. Описана нами, особливість дозволяє зберігати дрібні деталі зображення.

Детальніше дана архітектура нейронної мережі буде описана в підпункті 3.2.7.

Зазвичай для оцінки моделей сегментації зображень використовується метрика IoU (Intersection over Union). Вона характеризує ступінь збігу між прогнозованою маскою сегментації та еталонною розміткою. Чим вище значення IoU, тим точніше модель виконує сегментацію.

Оцінка моделей проводилась на однакових наборах даних. На рисунку 2.6 наведено порівняння оцінок точності різних моделей нейронних мереж для задачі сегментації зображень:

Мережа	mIoU	Переваги	Обмеження
FCN-8	~ 57	Швидкодія, хороша продуктивність	Невисока точність сегментації, необхідність адаптації для сегментації
SegNet	~ 56.1	Швидкодія, висока продуктивність	Невисока точність сегментації, необхідність адаптації для сегментації
U-Net	~ 77.5	Гарні результати сегментації на невеликій кількості даних, швидкодія, висока точність	Невисока точність сегментації для деяких задач
FC-Dence Net	~ 66.9	Висока точність та якість карток сегментації на виході	Недостатня точність сегментації для деяких задач
Refine Net	~ 73.6	Висока точність	Невисока швидкодія
PSPNet	~ 81.2	Висока точність	Невисока швидкодія
Link-Net	~ 76.4	Працює в режимі реального часу	Потрібен потужний процесор
ENet	~ 58.3	Працює в режимі реального часу	Потрібен потужний процесор
Deep Labv3	~ 81.3	Висока точність	Невисока швидкодія

Рисунок 2.6 – Порівняння точності моделей сегментації

Як можна помітити, найточнішими моделями є U-Net, PSPNet та DeepLab V3. Для розробки нашої системи було обрано за основу архітектуру U-Net, оскільки вона поєднує в собі високу точність, ефективну роботу при невеликій кількості даних та орієнтованість на біомедичні знімки.

Висновки до розділу 2

У другому розділі було проведено аналіз існуючих програм для роботи з МРТ та КТ знімками, а також було досліджено найвідоміші існуючі архітектури моделей нейронних мереж для задач сегментації.

Розглянуті системи, 3d Slicer та ITK-Snap, пропонують широкий спектр інструментів обробки та аналізу медичних даних. Дані системи є достатньо складними у експлуатації для звичайного користувача, оскільки користування ними потребує деяких додаткових знань та навичок. Крім того, інтерфейси цих програмних продуктів містять велику кількість функцій та налаштувань, що ускладнює освоєння програм і знижує зручність їх використання.

Другою проблемою даних систем є відсутність моделі для виділення структур колінного суглобу, або потребує додаткового налаштування або використання зовнішніх моделей.

Тому було прийнято рішення розробити власну систему, орієнтовану саме на сегментацію структур колінного суглоба.

Також в даному розділі було розглянуто наявні архітектури моделей нейронних мереж для сегментації зображень. Було проведено дослідження декількох з них, а саме FCN, PSPNet, DeepLab V3 та U-Net.

В рамках розробки нашої системи аналізу КТ-знімків для виділення структур колінного суглобу було обрано архітектуру U-Net, оскільки вона має високу точність за умови невеликої кількості даних. Ще однією причиною її вибору стала орієнтація даної мережі на біомедичні зображення, що в нашому випадку цілком підходить.

3 ЗАСОБИ ТА ЕТАПИ РОЗРОБКИ

Для розробки системи сегментації медичних зображень використовувалися сучасні програмні засоби та технології, призначені для обробки даних, навчання нейронних мереж і візуалізації результатів. У процесі роботи застосовувалися бібліотеки комп'ютерного зору та машинного навчання, засоби роботи з медичними зображеннями, системи керування версіями та середовища розробки

3.1 Етапи розробки системи

Розробка системи аналізу КТ-знімків для виділення структур колінного суглобу здійснювалась поетапно, завдяки чому поступово реалізувати всі ключові компоненти – від підготовки даних до інтеграції в програмний інтерфейс. Таблиця 3.1 нижче містить опис кожного етапу розробки нашого програмного продукту:

Таблиця 3.1 – Етапи розробки системи аналізу КТ-знімків для виділення структур колінного суглобу

Найменування етапу	Опис
Аналіз предметної області та постановка задачі	На першому етапі було визначено тип задачі — семантична сегментація анатомічних структур колінного суглоба на основі КТ- та МРТ-зображень. Також було проаналізовано існуючі підходи до сегментації та визначено вимоги до майбутньої системи, зокрема точність, стабільність та можливість роботи з медичними даними.

Продовження таблиці 3.1

Визначення засобів розробки	На даному етапі було проведено аналіз існуючих засобів, які були б необхідними для розробки нашої системи. Основна увага приділялася вибору мови програмування, середовища розробки, бібліотек для роботи з нейронними мережами та засобів обробки медичних даних.
Збір та підготовка даних	Було обрано медичний датасет SKI10, що містить МРТ-знімки колінного суглоба та відповідні маски сегментації. На цьому етапі виконувалося завантаження даних у форматі MHD/RAW, їх структурування та розподіл на навчальну, валідаційну і тестову вибірки.
Попередня обробка даних	Здійснювалась підготовка зображень до подачі в нейронну мережу. Вона включала розробку модуля нормалізації пікселів, приведення розмірів зображень до єдиного формату, а також формування 2D зрізів із 3D медичних об'ємів. Це дозволило стандартизувати вхідні дані та підвищити стабільність навчання моделі.

Продовження таблиці 3.1

Проектування архітектури моделі	На даному етапі було обрано архітектуру U-Net як базову модель для задачі сегментації. Було визначено структуру енкодера та декодера, а також реалізовано skip-з'єднання для збереження просторової інформації між шарами.
Навчання моделі	Нейронна мережа навчалась на підготовленому наборі даних із використанням функції втрат, адаптованої до задачі сегментації. Процес навчання включає 35 епох, під час яких виконувалась оптимізація ваг моделі методом градієнтного спуску.
Тестування та налагодження	Після навчання модель перевірялась на тестовій вибірці. Оцінювання якості виконувалося за допомогою метрики mIoU та Dice, що дозволяє визначити ступінь відповідності передбаченої маски реальній розмітці.

Продовження таблиці 3.1

Розробка інтерфейсу	Завершальним етапом стало впровадження навченої моделі в користувацький інтерфейс з метою візуалізації результатів сегментації у вигляді накладених масок на знімок. Було реалізовано можливість завантаження медичних знімків, запуску сегментації та перегляду результатів у зручному графічному вигляді.
---------------------	---

3.2 Засоби розробки системи

На етапі визначення засобів розробки важливо провести аналіз існуючих технологічних інструментів, які можуть бути в нагоді при створенні системи. Особливий акцент потрібно зосередити на засобах та бібліотеках для розробки нейронних мереж та роботи з медичними даними.

Перш за все потрібно визначитись з мовою програмування, яка буде використовуватись для написання коду нашої системи. В даному випадку було використано мову Python. Для роботи з медичними даними, зокрема типу DICOM, було обрано спеціалізовану бібліотеку Pydicom. Крім того, основним етапом розробки системи є реалізація архітектури та алгоритму навчання і тестування моделі нейронної мережі. Для цього було обрано відомий набір інструментів PyTorch. Не менш важливим завдання є створення графічного інтерфейсу, зрозумілого для звичайного користувача і здатного працювати без проблем. Для досягнення даного ефекту було використано бібліотеку PyQt6.

3.2.1 Мова програмування Python

Python – це високорівнева інтерпретована мова програмування, відома своєю простотою, читабельністю та універсальністю. Вона використовується майже скрізь – від різноманітних веб-додатків та інструментів автоматизації до обробки даних та штучного інтелекту. Логотип даної мови наведено на рисунку 3.1:



Рисунок 3.1 – Логотип Python

Мова програмування Python була розроблена Гвідо ван Россумом і була вперше випущена 20 лютого 1991 року. На сьогоднішній день, вона підтримується та розвивається Python Software Foundation, некомерційною організацією, що займається підтримкою мови та її екосистеми [12].

Дана мова має дуже великий набір різних потужних бібліотек для роботи з науковими даними, машинним навчанням та обробкою зображень. Зокрема для роботи з нейронними мережами можуть використовуватись бібліотеки TensorFlow та PyTorch, для обробки числових даних – NumPy та Pandas, а для роботи з зображеннями – OpenCV та Pillow. Саме тому дана мова була обрана як основний засіб розробки системи.

3.2.2 Pydicom

Pydicom – це бібліотека мови Python, призначена для роботи з файлами формату DICOM, що широко застосовується у медичній сфері для зберігання зображень, звітів та об'єктів променевої терапії. Вказана бібліотека дозволяє

розробнику медичних систем витягувати дані зображень, в нашому випадку МРТ-та КТ-знімків [13]. Логотип даної бібліотеки наведено на рисунку 3.2:



Рисунок 3.2 – Логотип Pydicom

Також даний пакет дозволяє отримувати метадані про пацієнта та дослідження. До таких даних можуть належати ім'я пацієнта, дата дослідження, параметри сканування, розміри зображення та інша службова інформація. Розробка нашої системи буде передбачати анонімізацію даних про пацієнтів.

3.2.3 NumPy

Отримавши пікселі з файлів DICOM або RAW, дані пікселів потрібно обробити перед поданням моделі з ціллю навчання чи тестування. Ідеальним варіантом в нашому випадку буде використання популярної бібліотеки NumPy.

NumPy (Numerical Python) — це бібліотека мови Python, орієнтована на виконання числових обчислень. Вона містить широкий спектр різноманітних засобів для роботи з багатовимірними масивами та матрицями, а також великі кількості математичних функцій для їх обробки. Бібліотека створювалась з орієнтацією на потреби фахівців у галузі аналізу даних та наукових досліджень, які працюють із великими обсягами числових даних та складними обчислювальними задачами [14]. Логотип даної бібліотеки зображено на рисунку 3.3:



Рисунок 3.3 – Логотип NumPy

На відміну від вбудованих списків Python, масиви NumPy зберігають числа в безперервному блоці пам'яті, що робить їх швидшими та ефективнішими для числових операцій. Це дозволяє вмістити більше даних у швидку оперативну пам'ять комп'ютера, що дозволяє виконувати обчислення з більшими наборами даних.

У рамках даної роботи NumPy використовується для роботи з МРТ- та КТ-знімками після їх завантаження, обробки тривимірних медичних даних, формування масок сегментації та підготовки вхідних даних для навчання і тестування моделі нейронної мережі.

3.2.4 PyTorch

PyTorch – це одна з найбільших використовуваних бібліотек для машинного навчання з відкритим кодом, яка використовується для розробки та навчання нейронних мереж.

PyTorch був створений компанією MetaAI. Він є дуже розповсюдженим серед дослідників у сфері ШІ завдяки своїй гнучкості, простоті використання та потужній підтримці спільноти [15]. Логотип PyTorch представлено на рисунку 3.4:



Рисунок 3.4 – Логотип PyTorch

Однією з ключових особливостей PyTorch є динамічний обчислювальний граф, який дозволяє гнучко змінювати архітектуру моделі під час виконання програми. Це особливо важливо для експериментальних задач та розробки складних моделей глибокого навчання.

В рамках даної роботи PyTorch має роль основного програмного фреймворку для реалізації архітектури U-Net, обробку тензорних даних МРТ та КТ-знімків, а також організацію алгоритму навчання та тестування моделі.

3.2.5 Matplotlib

Найбільш використовуваною бібліотекою для візуалізації даних на Python є Matplotlib. Даний інструмент допомагає створювати статичні, інтерактивні та анімовані графіки, в тому числі лінійні діаграми, стовпчасті діаграми, діаграми розсіювання, гістограми, тощо [16]. Логотип даної бібліотеки зображений на рисунку 3.5:



Рисунок 3.5 – Логотип Matplotlib

Ще однією потрібною в нашому випадку можливістю даної бібліотеки є здатність відображення МРТ та КТ-зрізів та візуалізації результатів сегментації. Це дозволяє користувачу наочно оцінювати якість роботи нейронної мережі виділення структур колінного суглобу.

3.2.6 PyQt6

PyQt6 – це набір бібліотек для створення графічних інтерфейсів користувача на мові Python, який є прив'язкою до фреймворку Qt. Даний інструмент є одним з найбільш розповсюджених завдяки широким функціональним можливостям та підтримці кросплатформності. Логотип даної бібліотеки наведено на рисунку 3.6:



Рисунок 3.6 – Логотип PyQt6

В рамках нашої роботи PyQt6 використовується для реалізації графічного інтерфейсу. За допомогою даної бібліотеки було створено вікно застосунку з функціями завантаження даних, візуалізації МРТ та КТ-знімків з результатами їх сегментації.

3.2.7 CNN та архітектура U-Net

Для вирішення проблеми сегментації зображень було використано згорткові нейронні мережі (англ. CNN). Основна ідея CNN полягає у поступовому виділенні ознак із вхідного зображення за допомогою операцій згортки та підвибірки (англ. pooling). На етапі згортки фільтри проходять по зображенню та формують карти ознак, які відображають різні рівні абстракції — від простих контурів і меж анатомічних структур до складних текстур і форм тканин. Дана нейронна мережа має три основні типи шарів:

- Згортковий шар (англ. Convolutional layer)
- Шар об'єднання (англ. Pooling layer)
- Повнозв'язний шар (англ. Fully-connected layer)

Згортковий шар – перший шар CNN, за яким можуть йти ще додаткові згорткові шари або шари об'єднання. Повнозв'язний шар є останнім та завершає архітектуру мережі. З кожним наступним шаром мережа аналізує дедалі складніші характеристики зображення: початкові шари виявляють прості низькорівневі ознаки, такі як краї та перепади кольорів, тоді як глибші шари поступово навчаються розпізнавати складніші структури та форми. По ходу того, як дані зображення просуваються через шари CNN, мережа починає розпізнавати більші елементи або форми об'єкту, доки нарешті не ідентифікує потрібний об'єкт [18]. Схема даної архітектури представлена на рисунку 3.7:

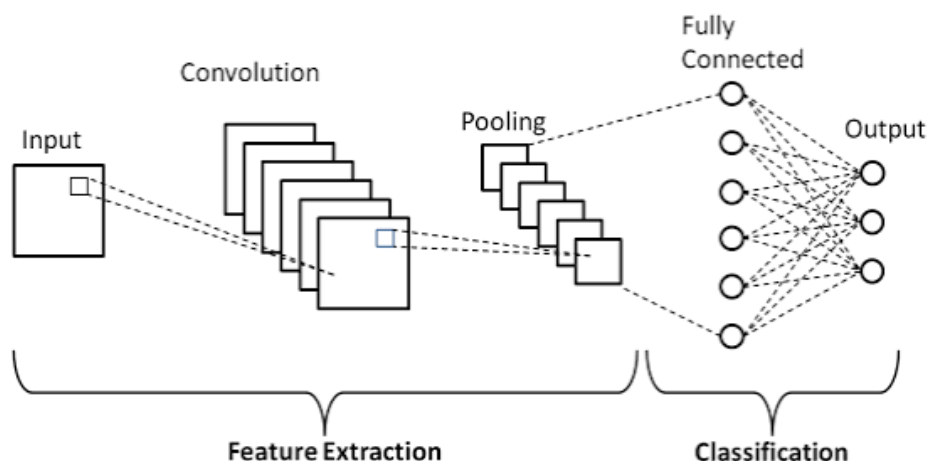


Рисунок 3.7 – Архітектура CNN

Основною перевагою CNN є здатність автоматично виділяти інформативні ознаки вхідних зображень без необхідності ручного визначення характеристик, що дозволяє ефективно працювати з великими обсягами даних. Окрім того, CNN є стійкою до незначних зсувів, масштабування та спотворень зображень.

Однак дана архітектура містить і недолік. Наприклад зі збільшенням глибини мережі виникає проблема затухання градієнта, що ускладнює навчання ранніх шарів моделі та може призводити до погіршення якості результатів.

Також однією з основних проблем є втрата просторової інформації через використання операцій підвибірки (англ. pooling), що може негативно впливати на точність локалізації об'єктів у задачах сегментації.

Крім того, CNN потребують значних обсягів навчальних даних для досягнення високої точності, а також є чутливими до якості та різноманітності набору даних. У задачах медичної сегментації це може бути суттєвим обмеженням через обмежену кількість розмічених даних.

Враховуючи зазначені проблеми CNN, зокрема втрату просторової інформації та складність точного відновлення меж об'єктів, виникає потреба у розробці більш спеціалізованої архітектури для задачі сегментації. Особливо це є актуальним у медичній сфері, де необхідна висока точність у завданні виділення структур на КТ або МРТ-знімках.

Саме тому було обрано модель U-Net, яка є розвитком ідей згорткових нейронних мереж та спеціально адаптована для задач семантичної сегментації, зокрема медичних зображень.

Основною особливістю даної моделі є симетрична структура, що складається з двох частин: енкодера (шлях звуження) та декодера (шлях розширення). Дана архітектура розглядалась у розділі 2.3 та її схема представлена на рисунку 2.5.

Важливим покращенням даної мережі в порівнянні з традиційною CNN є використання пропускових з'єднань (англ. skip-connections) між відповідними рівнями енкодера та декодера, що дозволяє поєднувати високорівневі ознаки з низькорівневою просторовою інформацією [19]. Дане нововведення забезпечує підвищення точності сегментації.

Але U-Net також і має певні недоліки:

- Чутливість до якості даних: На продуктивність U-Net можуть негативно впливати дані низької якості, такі як шумні або низькороздільні зображення.
- Схильність до перенавчання з малими наборами даних: Хоча U-Net добре працює в умовах обмежених даних, він все ще ризикує перенавчатися, якщо його не регуляризувати належним чином, особливо коли набір даних занадто малий або йому бракує різноманітності.
- Обчислювальні ресурси: Дана модель вимагає значних апаратних ресурсів для навчання

3.2.8 Інші засоби

Окрім основних засобів розробки, перелічених вище, у процесі розробки було використано ряд додаткових інструментів, що забезпечують ефективну розробку нашої системи.

В якості середовища розробки системи аналізу КТ-знімків для виділення структур колінного суглобу було обрано PyCharm. Даний інструмент розробника з відкритим кодом розробила компанія JetBrains. PyCharm доступний на операційних

системах Apple Mac, Microsoft Windows та Linux [17]. Дане середовище розробки було вибрано тому, що воно надає зручні інструменти для написання, налагодження та тестування Python-застосунків.

Бібліотека PIL (Python Imaging Library) застосовувалась для роботи з 2D зображеннями. Вона дозволяє виконувати базові операції над зображеннями, такі як відкриття, збереження, зміна розміру, конвертація форматів та попередня обробка даних перед подачею до нейронної мережі. У даній роботі бібліотека PIL використовується для приведення розміру масок сегментації до розміру вихідних медичних зображень та забезпечення коректного накладання результатів моделі на вхідні дані.

Для контролю версій вихідного коду використовувалася система Git, яка дозволяє відслідковувати зміни в проєкті, організовувати командну роботу та забезпечувати можливість відкату до попередніх версій програмного коду.

3.3 Набір даних SKI10 Dataset

При розробці систем, які використовують методи машинного навчання, не менш важливим етапом є пошук та підготовка даних для навчання моделі.

Датасет (англ. Dataset) – це організований набір даних, який використовується для навчання, тестування та валідації моделей машинного навчання.

Для розробки моделі сегментації МРТ та КТ-знімків колінного суглобу було використано датасет SKI10 (Segmentation of Knee Images 2010). Даний датасет був наданий компанією Biomet, Inc. в рамках міжнародного конкурсу з автоматичної сегментації коліна SKI10. Дане змагання було організоване як один з трьох конкурсів Grand Challenge Workshop 2010 [20].

Даний набір даних містить 100 МРТ-знімків коліна, зроблених в режимі T1. Дані зображень зберігаються у вигляді пари файлів MHD-RAW. MHD-файл – це тип файлу який містить метадані самого знімку, а саме розмірність, назву файлу в

якому зберігаються безпосередньо піксельні дані знімків. RAW-файл – це тип файлу який містить сирі пікселі зображення.

Наша система буде здійснювати сегментацію медичних знімків у сагітальній площині. Дана площина описує знімки в профіль (зрізи «зліва направо» або «навпаки»). Часто використовуються для дослідження хребта.

Висновки до розділу 3

У третьому розділі було здійснено розгляд засобів та етапів розробки нашої системи аналізу КТ та МРТ-знімків для виділення структур колінного суглобу.

Внаслідок аналізу етапів розробки, було виділено вісім ключових пунктів:

- Аналіз предметної області та постановка задачі
- Визначення засобів розробки
- Збір та підготовка даних
- Попередня обробка даних
- Проектування архітектури моделі
- Навчання моделі
- Тестування та налагодження
- Розробка інтерфейсу

Також було обґрунтовано вибір засобів розробки. Мова програмування була обрана Python, завдяки широкій екосистемі бібліотек для машинного навчання та обробки даних. Для обробки медичних даних типу DICOM використовується пакет Pydicom, для чисельних обчислень – NumPy. Фреймворк PyTorch забезпечує реалізацію та навчання нейронної мережі. Для візуалізації медичних знімків та результатів сегментації використано matplotlib, а для розробки графічного інтерфейсу – PyQt6.

Було описано основний принцип роботи згорткових нейронних мереж (CNN) та обрано архітектуру U-Net. В порівнянні з традиційними CNN, U-Net завдяки

симетричній структурі енкодер-декодер та механізму skip-з'єднань краще зберігає просторову інформацію, що є дуже важливим для роботи з медичними знімками.

Також у даному розділі було описано набір даних SKI10, який містить 100 МРТ-знімків колінного суглобу у форматі MHD/RAW, який використовується для навчання та тестування нашої моделі нейронної мережі.

4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

У даному розділі розглянемо програмну реалізацію нашої системи сегментації структур колінного суглобу, архітектуру нашої моделі та загалом системи.

4.1 Взаємодія користувача та системи

Цільовими користувачами нашої системи можуть бути лікарі-ортопеди, травматологи, рентгенологи, а також медичні дослідники та різноманітні розробники медичних систем.

Для хорошого розуміння функціоналу системи на рисунку 4.1 зображено UML діаграму варіантів використання:

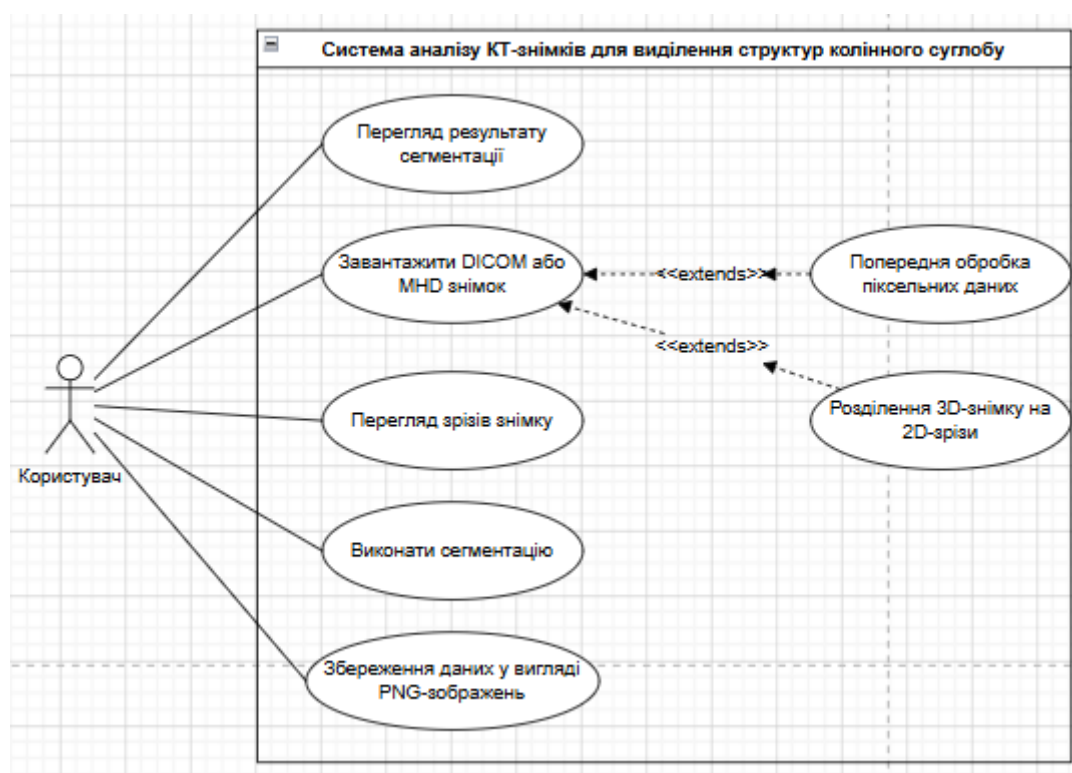


Рисунок 4.1 – Діаграма варіантів використання системи

Як можна помітити на рисунку 4.1, система надає користувачу наступний функціонал:

- Завантаження МРТ або КТ-знімку у форматі DICOM або MHD/RAW;
- Перегляд завантаженого знімку користувачем по зрізам;
- Запуск сегментації МРТ або КТ-знімку колінного суглобу;
- Перегляд результату виділення анатомічних структур;
- Збереження результату роботи у вигляді 2D-зрізів в форматі PNG.

Окрім того, на діаграмі можна побачити, що у випадку завантаження медичного знімку користувачем, система почне цикл попередньої обробки та нормалізації даних для візуалізації та можливого подання моделі сегментації. Також у випадку завантаження 3D-знімку, програма поділить його на 2D-зрізи.

Загальна послідовність роботи із системою полягає у завантаженні медичного знімка, його візуалізації, запуску процесу сегментації, перегляді отриманих результатів та їх подальшому збереженні. Такий підхід забезпечує простоту використання системи та дозволяє зосередити увагу користувача на аналізі результатів, а не на виконанні технічних операцій.

4.2 Архітектура системи

В рамках даної роботи було створено десктопний застосунок, що поєднує в собі інструменти для роботи з медичними зображеннями та методи глибокого навчання для автоматичної сегментації структур колінного суглобу та користувацький інтерфейс для діагностики. Схема архітектури системи аналізу МРТ та КТ-знімків для виділення структур колінного суглобу представлена на рисунку 4.2:

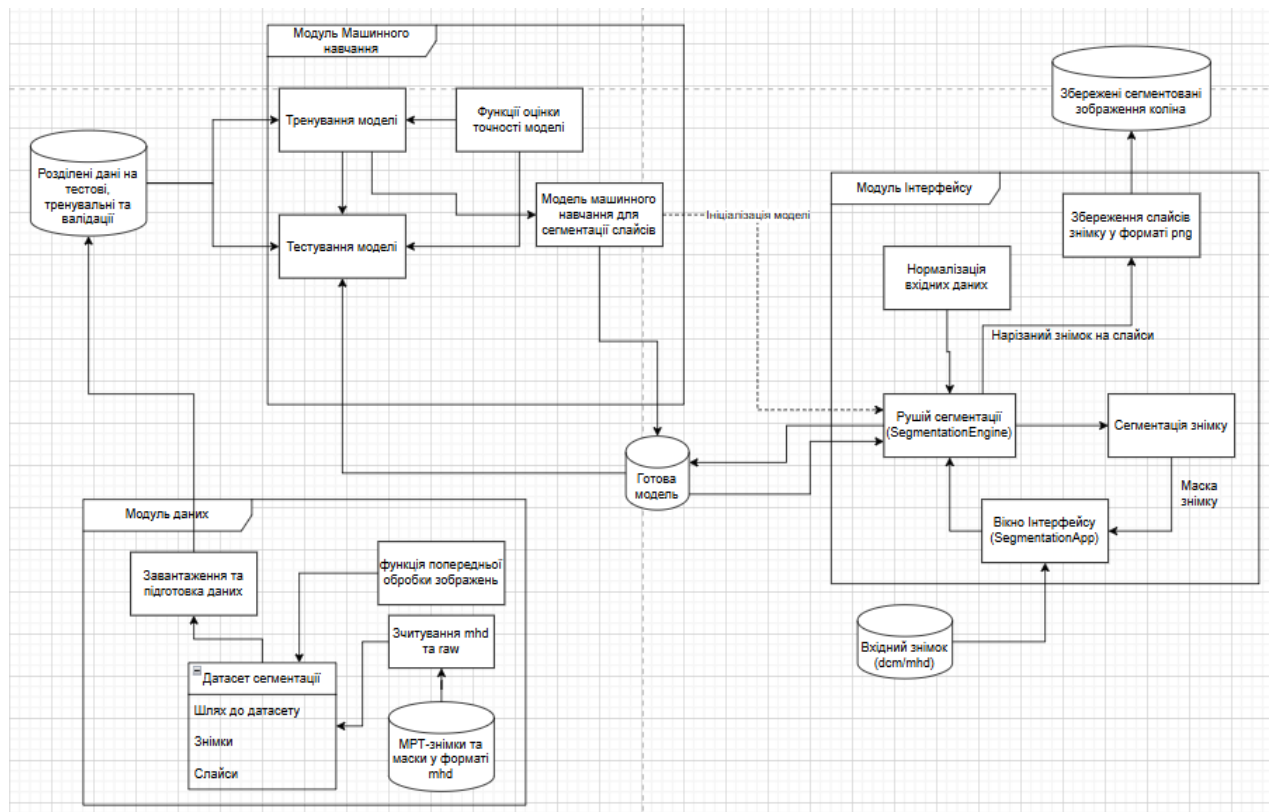


Рисунок 4.2 – Архітектури системи

Дана архітектура складається з трьох ключових модулів:

Модуль даних (Data Module): Модуль даних відповідає за завантаження, читання та підготовку медичних зображень в нейронну мережу. Його основу складають три компоненти: функція читання MHD/RAW-файлів, клас датасету та функція завантаження, обробки та розбиття даних датасету.

Модуль машинного навчання (ML Module): Модуль машинного навчання містить реалізацію архітектури нейронної мережі U-Net. В даній частині системи описана модель нейронної мережі, містяться реалізації функцій втрат та оцінки точності моделі. Також даний модуль містить реалізований алгоритм навчання та тестування моделі нейромережі.

Модуль інтерфейсу (Segmentation Interface): Даний сегмент системи складається з двох програмних файлів класів SegmentationEngine та SegmentationApp. SegmentationEngine є рушієм сегментації та забезпечує міст між навченою моделлю та інтерфейсом, коли SegmentationApp реалізує графічний інтерфейс користувача.

Такий поділ на модулі забезпечує чітке розмежування відповідальності між компонентами системи, спрощує її супровід та надає можливість незалежного вдосконалення кожної складової без впливу на інші частини застосунку.

4.2.1 Модуль даних

Перший системний компонент, який ми розглянемо буде модуль даних. Схема взаємодії складових частин даного модуля зображена на рисунку 4.3:

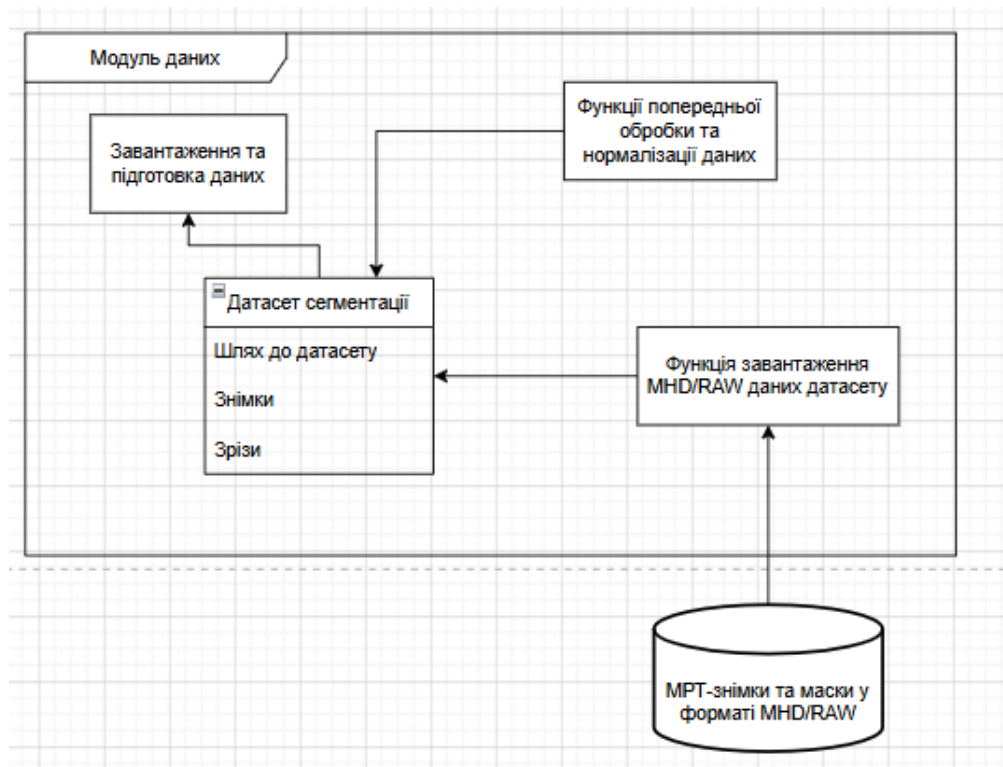


Рисунок 4.3 – Схема модуля даних

Робота даного модуля розпочинається з завантаження набору даних сегментації колін SKI10. Перш за все за допомогою інструментарію PyTorch було створено клас `SegmentationDataset`, який реалізує стандартний інтерфейс `PyTorch Dataset`. Даний клас має наступні функції:

- `__init__` - ініціалізація датасету, отримує шлях до директорії навчального набору даних моделі та алгоритм трансформації зображень та масок;
- `get_files` – сканування вказаної директорії та формує список пар файлів медичних знімків та їх масок сегментації;

- `get_slice_data` – завантажує пари файлів знімків та масок та розділяє 3D-зображення на 2D-зрізи;

- `__len__` - повертає кількість зрізів у датасеті;

- `__getitem__` - повертає окремий зріз MPT-знімку та його маску.

Окрім самого класу, який описує набір даних, даний програмний модуль містить наступні функції:

1. `read_mhd` – даний метод слугує для зчитування медичних знімків формату MHD/RAW. Відкриває MHD-файл, парсить його метадані, визначає тип даних пікселів, розмірність зображення та зчитує сирі пікселі з RAW-файлу. Повертає тривимірний масив NumPy та словник метаданих;

2. `load_dataset` – дана функція створює послідовність попередніх трансформацій зображень та масок. Створює два об'єкти `SegmentationDataset` для тренувальної та валідаційної вибірок, після чого забирає половину даних валідаційної вибірки для створення тестової. Розміри вибірок датасету наступні:

- Тренувальна вибірка – 60 пар
- Валідаційна вибірка – 20 пар
- Тестова вибірка – 20 пар

Після розбиття даних, створюється три об'єкти `PyTorch DataLoader`, який обгортає клас `SegmentationDataset` для легкого доступу до зразків, для кожної вибірки та повертає їх.

Батч (англ. `batch` — партія) — це підмножина тренувальних даних, яка подається до нейронної мережі за один крок навчання.

Замість того щоб подавати всі зображення одночасно або по одному, дані розбиваються на рівні частини — батчі. У розробленій системі розмір батчу становить 24 зрізи, тобто за один крок модель одночасно обробляє 24 зображення.

4.2.2 Модуль машинного навчання

Ключовим компонентом розробленої системи є модуль машинного навчання. Даний програмний модуль відповідальний за реалізацію архітектури моделі U-Net та алгоритму її навчання та тестування. Схема роботи даної частини програмного продукту наведена на рисунку 4.4:

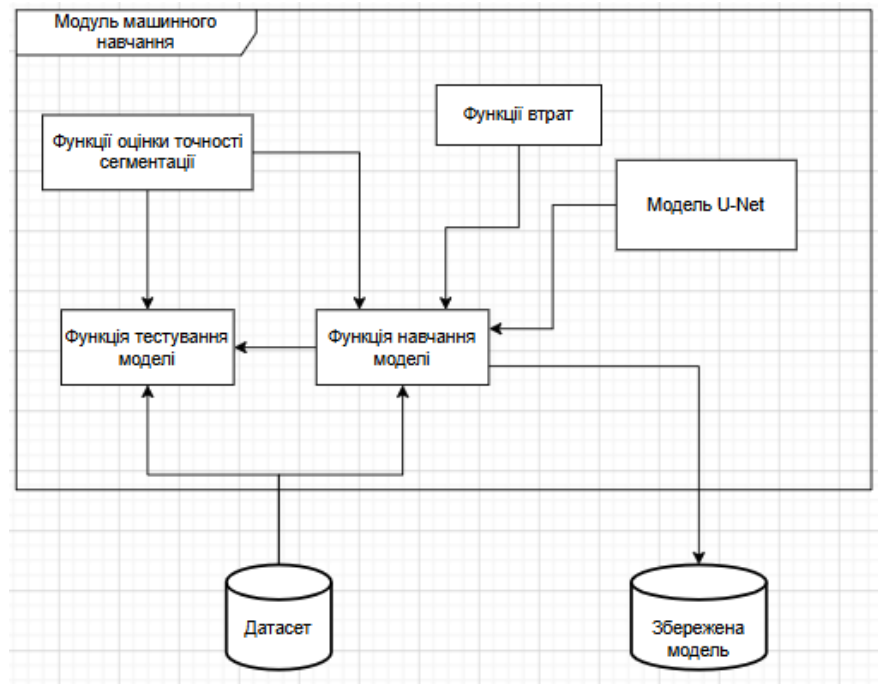


Рисунок 4.4 – Схема модуля машинного навчання

Робота модуля машинного навчання починається з створення об'єкту класу U-Net. Даний клас наслідує базовий клас `nn.Module` бібліотеки PyTorch та описує архітектуру U-Net для п'ятикласової сегментації (фон, верхня кістка, нижня кістка, верхній меніск та нижній меніск). Схема реалізації архітектури зображено на рисунку 4.5:

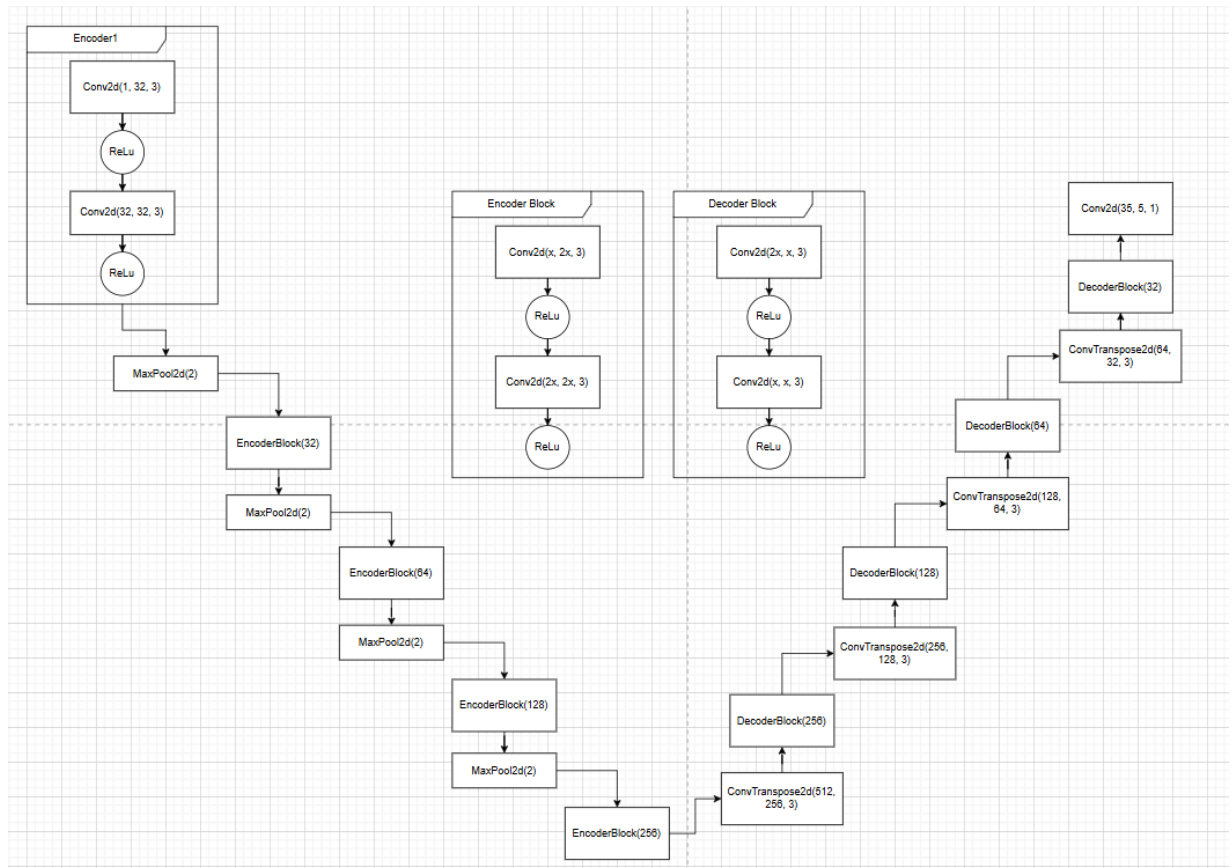


Рисунок 4.5 – Схема моделі U-Net

Метод `__init__` визначає всі шари мережі. Енкодер складається з чотирьох блоків `enc1` – `enc4` (рис. 4.6), кожен з яких містить згорткові шари 3×3 з відступом 1 та функцією активації ReLU. За допомогою `MaxPool2d` (рис. 4.6) з кроком 2, розмірність зменшується вдвічі. Кількість фільтрів подвоюється на кожному рівні: 1, 32, 64, 128 та 256.

Найвужчий шар розширює кількість до 512, що є найглибшим рівнем представлення ознак. Так само як і енкодер, декодер містить також 4 блоки `dec1` – `dec4`, кожному з яких передують транспонована згортка `ConvTranspose2d` (рис. 4.7), з кроком 2 для збільшення розмірності. Вихідний шар `conv_last` зі згорткою 1×1 перетворює 32 канали на 5 карт активацій відповідно до класів.

На рисунку 4.6 зображено фрагмент коду перших двох блоків `enc1` та `enc2` енкодера:

```

# Encoder
self.enc1 = nn.Sequential(
    nn.Conv2d(in_channels=1, out_channels=32, kernel_size=3, padding=1),
    nn.ReLU(inplace=True),

    nn.Conv2d(in_channels=32, out_channels=32, kernel_size=3, padding=1),
    nn.ReLU(inplace=True)
)
self.pool1 = nn.MaxPool2d(2)

self.enc2 = nn.Sequential(
    nn.Conv2d(in_channels=32, out_channels=64, kernel_size=3, padding=1),
    nn.ReLU(inplace=True),

    nn.Conv2d(in_channels=64, out_channels=64, kernel_size=3, padding=1),
    nn.ReLU(inplace=True)
)
self.pool2 = nn.MaxPool2d(2)

```

Рисунок 4.6 – Перші два блоки енкодера моделі

На рисунку 4.7 зображено фрагмент коду найглибшої шару та перших двох блоків dec1 та dec2 декодера:

```

# Bottleneck
self.bottleneck = nn.Sequential(
    nn.Conv2d(in_channels=256, out_channels=512, kernel_size=3, padding=1),
    nn.ReLU(inplace=True),

    nn.Conv2d(in_channels=512, out_channels=512, kernel_size=3, padding=1),
    nn.ReLU(inplace=True)
)

# Decoder
self.up4 = nn.ConvTranspose2d(in_channels=512, out_channels=256, kernel_size=2, stride=2)

self.dec4 = nn.Sequential(
    nn.Conv2d(in_channels=512, out_channels=256, kernel_size=3, padding=1),
    nn.ReLU(inplace=True),

    nn.Conv2d(in_channels=256, out_channels=256, kernel_size=3, padding=1),
    nn.ReLU(inplace=True)
)

self.up3 = nn.ConvTranspose2d(in_channels=256, out_channels=128, kernel_size=2, stride=2)

self.dec3 = nn.Sequential(
    nn.Conv2d(in_channels=256, out_channels=128, kernel_size=3, padding=1),
    nn.ReLU(inplace=True),

    nn.Conv2d(in_channels=128, out_channels=128, kernel_size=3, padding=1),
    nn.ReLU(inplace=True)
)

```

Рисунок 4.7 – Код найглибшого шару та двох блоків декодера

Метод `forward` класу `U-Net` визначає прямий прохід: послідовно обчислюються виходи всіх рівнів енкодера з пулінгом, боттлнек, після чого декодер відновлює просторову розмірність із конкатенацією `skip-з'єднань` на кожному рівні. Програмна реалізація даного методу представлена на рисунку 4.8:

```
def forward(self, x):
    e1 = self.enc1(x)
    e2 = self.enc2(self.pool1(e1))
    e3 = self.enc3(self.pool2(e2))
    e4 = self.enc4(self.pool3(e3))

    b = self.bottleneck(self.pool4(e4))

    d4 = self.dec4(torch.cat(tensors: [self.up4(b), e4], dim=1))
    d3 = self.dec3(torch.cat(tensors: [self.up3(d4), e3], dim=1))
    d2 = self.dec2(torch.cat(tensors: [self.up2(d3), e2], dim=1))
    d1 = self.dec1(torch.cat(tensors: [self.up1(d2), e1], dim=1))

    return self.conv_last(d1)
```

Рисунок 4.8 – Метод `forward`

Функція `train_model` реалізовує алгоритм навчання та тестування моделі сегментації медичних знімків. За допомогою функції `load_dataset` завантажуються тренувальна, валідаційна та тестова вибірки:

1. Тренувальна вибірка використовується безпосередньо для навчання моделі.
2. Валідаційна вибірка – для моніторингу якості навчання після кожної епохи.
3. Тестова вибірка використовується одноразово лише після завершення навчання для фінальної оцінки моделі

Ініціалізується модель `U-Net`. Алгоритм навчання даної мережі зображено на рисунку 4.9:

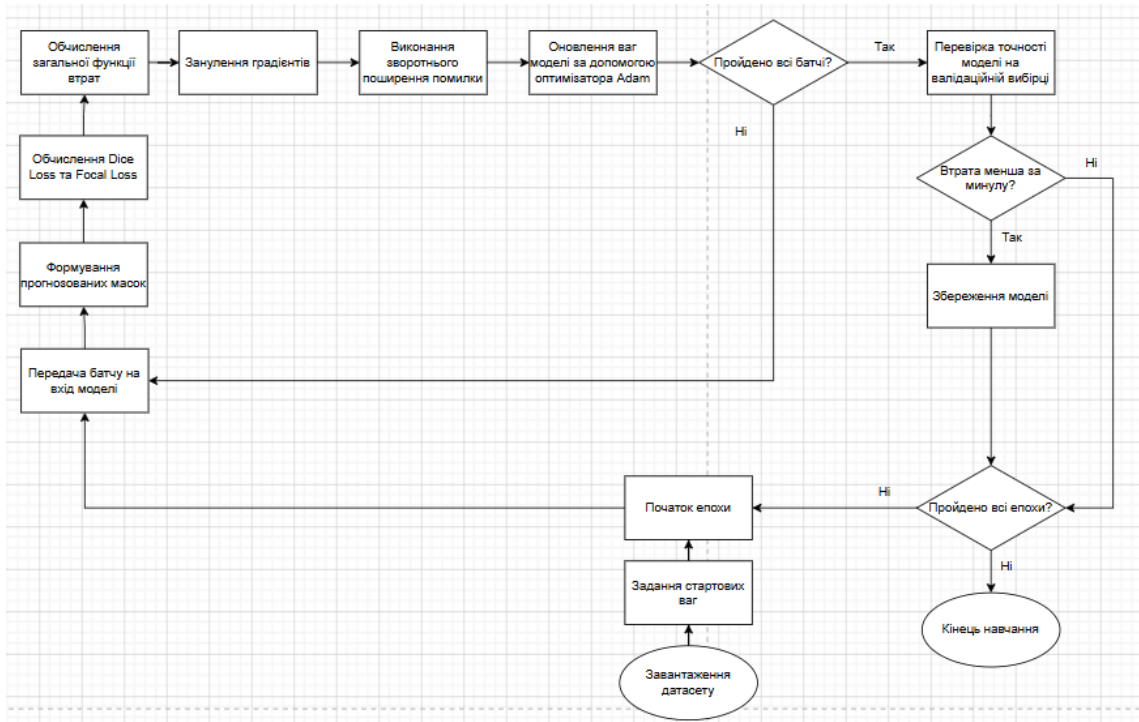


Рисунок 4.9 – Алгоритм навчання моделі

На початку кожної епохи виконується тренувальна фаза, що складається з послідовних кроків обробки батчів. Спочатку черговий батч зображень передається на вхід моделі, яка формує прогнозовані маски сегментації. На основі отриманих прогнозів обчислюються значення комбінованої функції втрат, яка дорівнює сумі складових функцій втрат — Dice Loss та Focal Loss.

Focal Loss – це спеціалізована функція втрат, яка використовується для вирішення проблеми серйозного дизбалансу класів. Вона була використана оскільки більшість пікселів займає фон, тоді як пікселі анатомічних структур займають меншу частину

Принцип роботи даної функції втрат полягає у динамічному перерозподілі уваги моделі під час навчання: внесок легко класифікованих зразків у загальну втрату зменшується, тоді як градієнтне оновлення зосереджується на складних, помилково розпізнаних випадках. Це досягається завдяки коефіцієнту згасання γ , що входить до формули 4.1:

$$FL = (1 - p_t)^\gamma \cdot CE, \quad (4.1)$$

де p_t – оцінена моделлю ймовірність для правильного класу ($p_t = e^{-CE}$),

γ – коефіцієнт фокусування, що визначає інтенсивність зменшення впливу простих прикладів на навчання.

CE – значення стандартної крос-ентропії для поточного зразка.

Dice Loss обчислюється на основі коефіцієнта Дайса, що вимірює ступінь перекриття між передбаченою маскою та еталонною розміткою. Коефіцієнт Дайса визначається як подвоєний розмір перетину двох множин, поділений на суму їх розмірів, і набуває значень від 0 до 1, де 1 відповідає повному збігу масок. На рисунку 4.10 зображено принцип обчислення коефіцієнту Дайса:

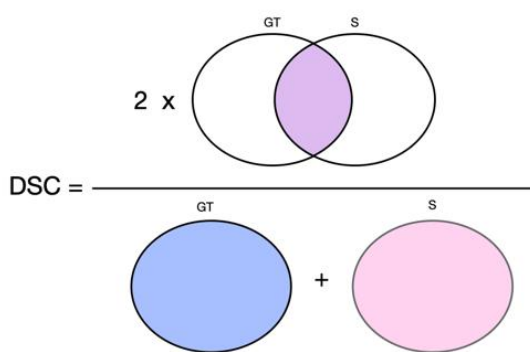


Рисунок 4.10 – Принцип обчислення коефіцієнта Дайса

На відміну від Focal Loss, який працює на рівні окремих пікселів, Dice Loss оцінює якість сегментації на рівні цілих регіонів, що робить його більш чутливим до точності відновлення форми та меж анатомічних структур.

Далі виконується занулення градієнтів, зворотне поширення помилки та оновлення ваг моделі за допомогою оптимізатора Adam. Якщо у поточній епосі залишились необроблені батчі, цикл повертається до передачі наступного батчу на вхід моделі.

Після того як усі батчі пройдено, розпочинається валідаційна фаза. Модель перевіряється на валідаційній вибірці та обчислюється поточне значення втрат. Отримане значення порівнюється з найкращим збереженим результатом попередніх епох. Якщо поточна втрата є меншою, модель зберігається як найкраща.

Після завершення валідаційної фази перевіряється умова завершення навчання. Якщо не всі епохи пройдено, алгоритм повертається до початку

наступної епохи. Після проходження всіх 35 епох навчання завершується, а збережена модель із найкращими ваговими коефіцієнтами використовується для подальшого тестування та використання.

Для оцінювання точності використовуються метрики IoU (Intersection over Union) та раніше згаданий коефіцієнт Дайса. IoU обчислюється як відношення площі перекриття масок на сумарну площу покриту обома масками. На рисунку 4.11 зображено схему обчислення IoU:

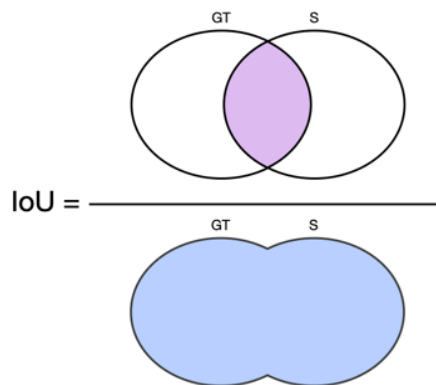


Рисунок 4.11 – Принцип обчислення IoU

Це співвідношення дає значення від 0 до 1:

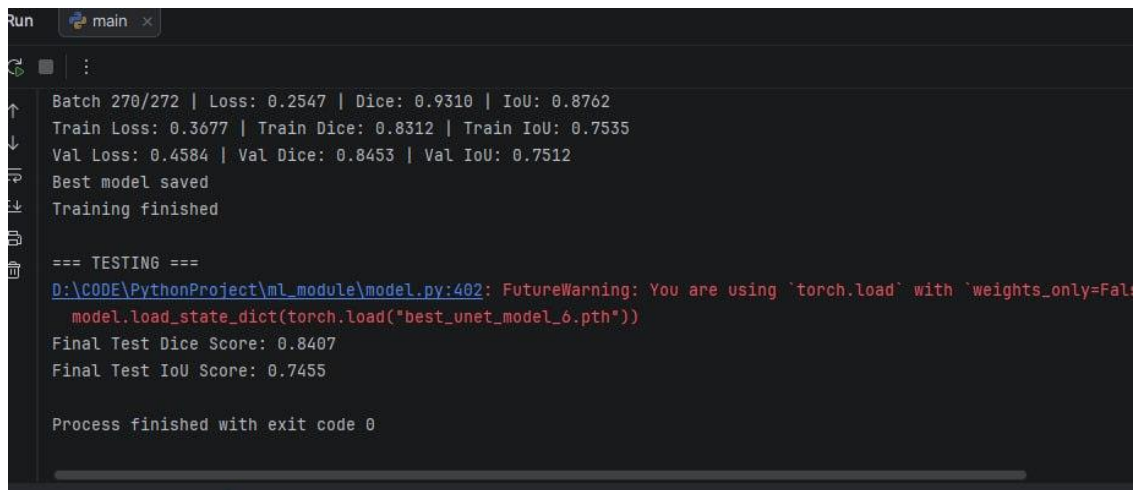
Якщо $\text{IoU} = 1$, то рамка прогнозування точно відповідає граничній рамці істинного рівня.

Коли $\text{IoU} = 0$, перекриття взагалі відсутнє.

Проміжні значення, такі як 0,5 або 0,75, показують, що рамки частково перекриваються [21].

В нашому випадку, коли сегментація є мультикласовою дані метрики обчислюються як сума значень поділена на кількість класів, тобто 5.

Результат навчання нашої моделі зображено на рисунку 4.12:



```
Run main x
Batch 270/272 | Loss: 0.2547 | Dice: 0.9310 | IoU: 0.8762
Train Loss: 0.3677 | Train Dice: 0.8312 | Train IoU: 0.7535
Val Loss: 0.4584 | Val Dice: 0.8453 | Val IoU: 0.7512
Best model saved
Training finished

=== TESTING ===
D:\CODE\PythonProject\ml_module\model.py:402: FutureWarning: You are using `torch.load` with `weights_only=False` which
model.load_state_dict(torch.load("best_unet_model_6.pth"))
Final Test Dice Score: 0.8407
Final Test IoU Score: 0.7455

Process finished with exit code 0
```

Рисунок 4.12 – Результат навчання моделі

На рисунку 4.12 видно, що метрики Dice Score та IoU Score під час тестування дорівнюють 0.84 та 0.745 відповідно.

Для виконання обчислень під час навчання моделі використовується графічний процесор NVIDIA GeForce RTX 4060, оснащений 24 ГБ відеопам'яті. Використання GPU дозволяє ефективно виконувати паралельні обчислення, необхідні для навчання глибоких нейронних мереж, що суттєво скорочує час тренування моделі та обробки великих обсягів даних.

Навчання тривало від 2 до 3 годин.

4.2.3 Модуль інтерфейсу

Останнім програмним компонентом системи аналізу МРТ та КТ-знімків для виділення структур колінного суглобу є модуль інтерфейсу, архітектура якого зображена на рисунку 4.13. Даний модуль забезпечує взаємодію користувача з системою.

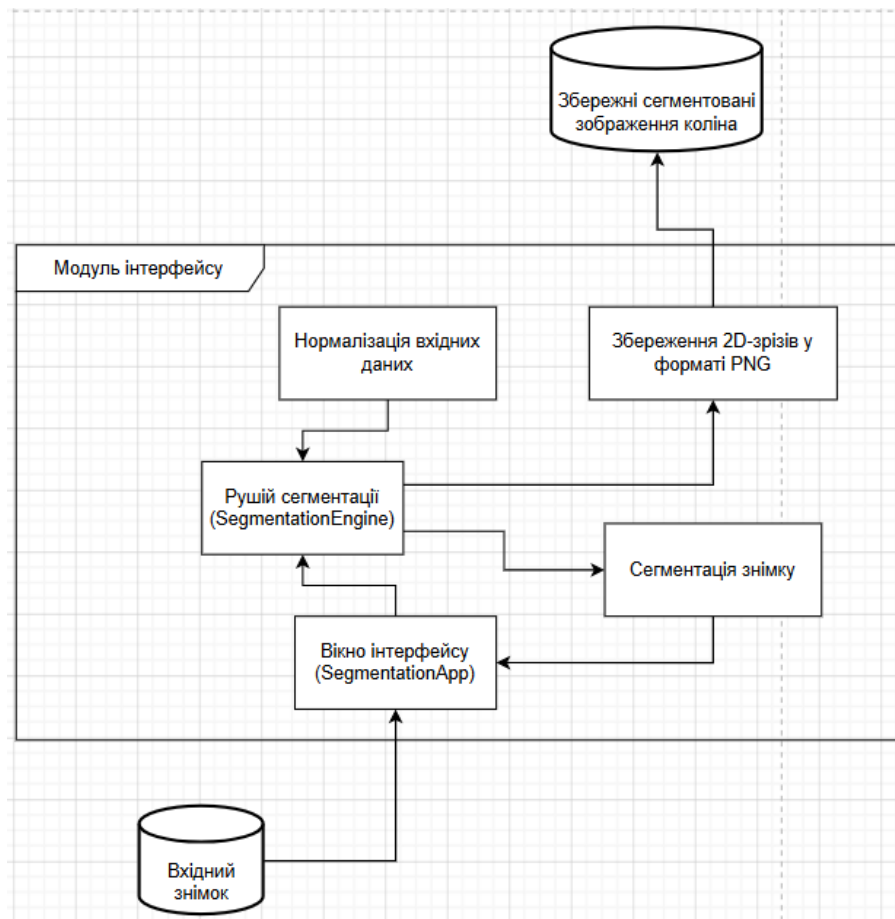


Рисунок 4.13 – Архітектура модуля інтерфейсу

Модуль складається з двох основних підмодулів SegmentationApp та SegmentationEngine. Процес взаємодії користувача з системою проходить саме через SegmentationApp, який реалізує вікно програми на основі PyQt6, яке дозволяє завантаження медичних знімків у форматі DICOM або MHD/RAW, запуск сегментації та перегляд її результатів у вигляді накладеної маски поверх знімка.

Рушій сегментації (SegmentationEngine) є проміжним шаром між навченою моделлю нейронної мережі та графічним інтерфейсом SegmentationApp. Він відповідає за завантаження моделі, підготовку вхідних даних та виконання інференсу. Інференс виконується послідовно: кожен зріз нормалізується, приводиться до розміру 384×288 пікселів та конвертується у тензор PyTorch. Модель повертає маску сегментації у вигляді масиву NumPy. Результати можна зберегти у форматі PNG як з накладеною кольоровою маскою, так і без нього.

Для читання медичних файлів реалізовано два методи. Метод `read_dicom` зчитує DICOM-файл за допомогою бібліотеки `Pydicom` та зберігає його метадані для подальшого відображення в інтерфейсі. Метод `read_mhd` делегує читання MHD/RAW-файлів відповідній функції модуля даних.

4.3 Проблеми системи

У процесі розробки та тестування даної системи було виявлено низку проблем, які впливають на якість та практичність її використання:

- **Обмежений спектр сегментованих знімків:** Модель навчалась виключно на МРТ-знімках колінного суглоба з датасету SKI10, отриманих у сагітальній площині в режимі T1. Це означає, що система не здатна коректно працювати зі знімками інших суглобів, іншими режимами МРТ-сканування або знімками, отриманими в інших площинах. Таке обмеження суттєво звужує область застосування розробленої системи в умовах реальної клінічної практики. З КТ-знімками також можуть бути проблеми. Наприклад спробуємо завантажити МРТ-знімок зроблений іншим пристроєм у фронтальній площині. Результат сегментації даного зображення наведено на рисунку 4.14:

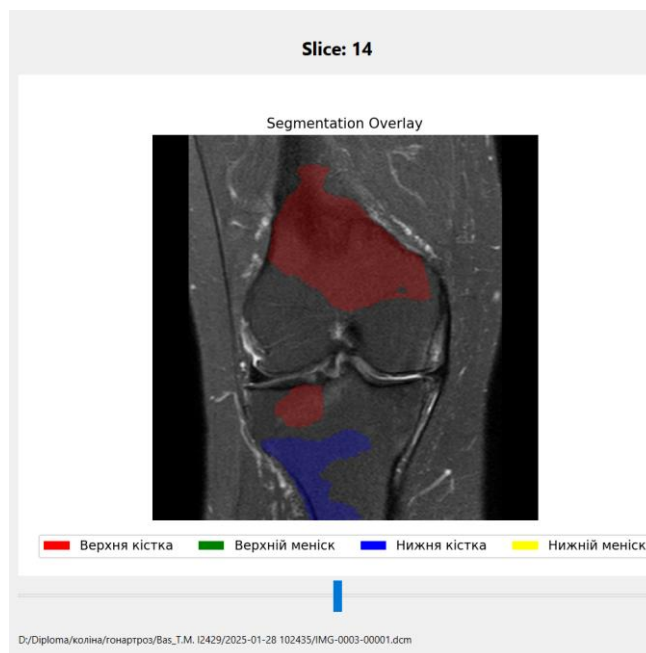


Рисунок 4.14 – Сегментація МРТ-знімку у фронтальній площині

- Відсутність даних з патологіями: Використаний нами набір даних SKI10 містить в основному МРТ-знімки здорових або незначно пошкоджених колінних суглобів. Модель не навчалась на зображеннях із вираженими патологіями — такими як розрив менісків, переломи кісток або значні дегенеративні зміни. Внаслідок цього при роботі з медичними знімками з серйозними патологіями, анатомічна структура може суттєво відрізнятись від навчальних даних, що призводить до некоректного результату сегментації
- Поява артефактів за відсутності об'єктів: При обробці зрізів, на яких анатомічні структури колінного суглоба відсутні або слабо виражені — наприклад, на крайніх зрізах об'єму, — модель може генерувати хибні маски сегментації. Оскільки мережа намагається виділити структури на кожному зрізі незалежно від їх фактичної наявності, на таких зображеннях з'являються артефакти у вигляді розрізнених областей неправильної форми, що не відповідають жодній реальній анатомічній структурі. На рисунку 4.15 зображено зріз сегментованого МРТ-знімку на якому немає ніяких об'єктів для виділення:

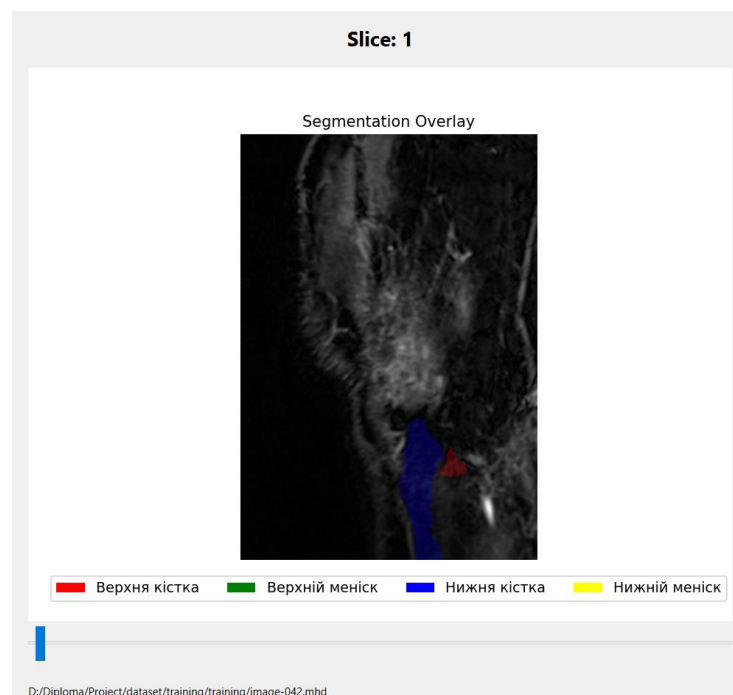


Рисунок 4.15 – Сегментація зрізу без об'єктів

Як можна помітити, система показує що на зрізові зображена нижня кістка та трішки верхня, хоча насправді їх там не має.

Висновки до розділу 4

У четвертому розділі було описано архітектуру розробленої системи, її програмну реалізацію та виявлені проблеми під час її створення.

Система складається з трьох взаємопов'язаних модулів. Модуль даних забезпечує завантаження та підготовку медичних даних, включаючи нормалізацію та зміну розмірності знімків та масок, формату MHD/RAW. Окрім того, до подачі в нейронну мережу здійснюється розбиття набору даних на тренувальну, валідаційну та тестувальну вибірки.

Модуль машинного навчання містить реалізацію архітектури U-Net, комбіновану функцію втрат на основі Focal Loss та Dice Loss, а також алгоритм навчання моделі протягом 35 епох із збереженням найкращої версії за валідаційною втратою.

Модуль інтерфейсу забезпечує зручну взаємодію користувача з системою через графічний застосунок на основі PyQt6, що підтримує завантаження медичних файлів, виконання сегментації та візуалізацію результатів у вигляді кольорового оверлею.

Розроблений алгоритм навчання реалізує повний цикл – від завантаження датасету та ініціалізації моделі до тренувальної та валідаційної фаз на кожній епосі. Комбінування Focal Loss та Dice Loss дозволяє ефективно вирішувати проблему дисбалансу класів, характерну для задач медичної сегментації.

Разом з тим було виявлено ряд обмежень системи. Модель навчалась на вузькоспеціалізованому датасеті без патологій, що обмежує її застосування в умовах реальної клінічної практики. Двовимірний підхід до сегментації не враховує просторові зв'язки між зрізами, а на крайніх зрізах об'єму спостерігається поява артефактів за відсутності анатомічних структур. Усунення зазначених обмежень є перспективним напрямком подальшого вдосконалення системи.

5 ВЗАЄМОДІЯ КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

У даному розділі було наведено можливості користувача для взаємодії з системою аналізу КТ та МРТ-знімків структур колінного суглобу.

Розроблений програмний продукт розрахований на користування лікарями-рентгенологами, як основною цільовою аудиторією. Саме тому графічний інтерфейс спроектований таким чином, щоб забезпечити максимально простий та зрозумілий робочий процес без необхідних додаткових технічних навичок.

Запустивши застосунок, користувач побачить вікно, яке складається з наступних атрибутів:

- Зона перегляду медичних знімків
- Зона перегляду інформації про дослідження
- Кнопка «Завантажити файл»
- Кнопка «Сегментувати»
- Кнопка «Конвертація в PNG»

Вигляд графічного інтерфейсу користувача представлений на рисунку 5.1:

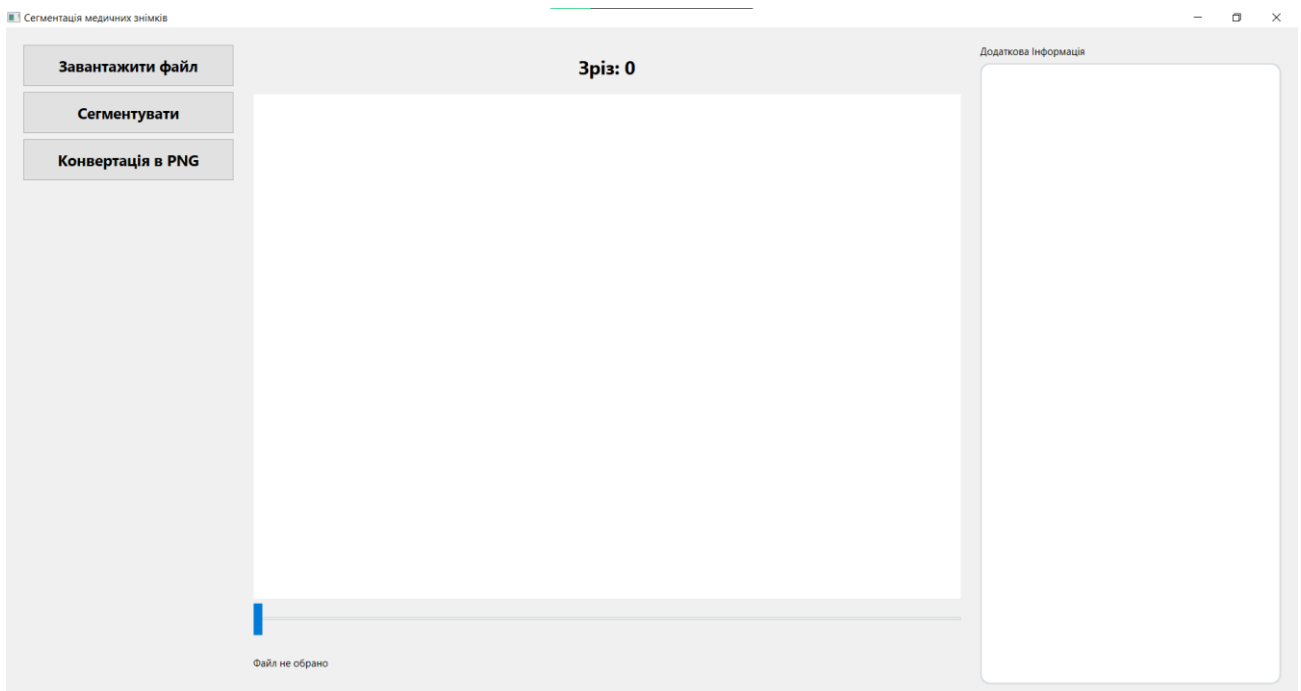


Рисунок 5.1 – Інтерфейс користувача

Даний інтерфейс створений у мінімалістичному стилі.

5.1 Завантаження медичних знімків

Розпочавши роботу з системою, користувач захоче завантажити медичні дані для попереднього перегляду. Після завантаження МРТ-знімку, можна буде переглянути його зрізи та інформацію про дослідження (рис. 5.2).

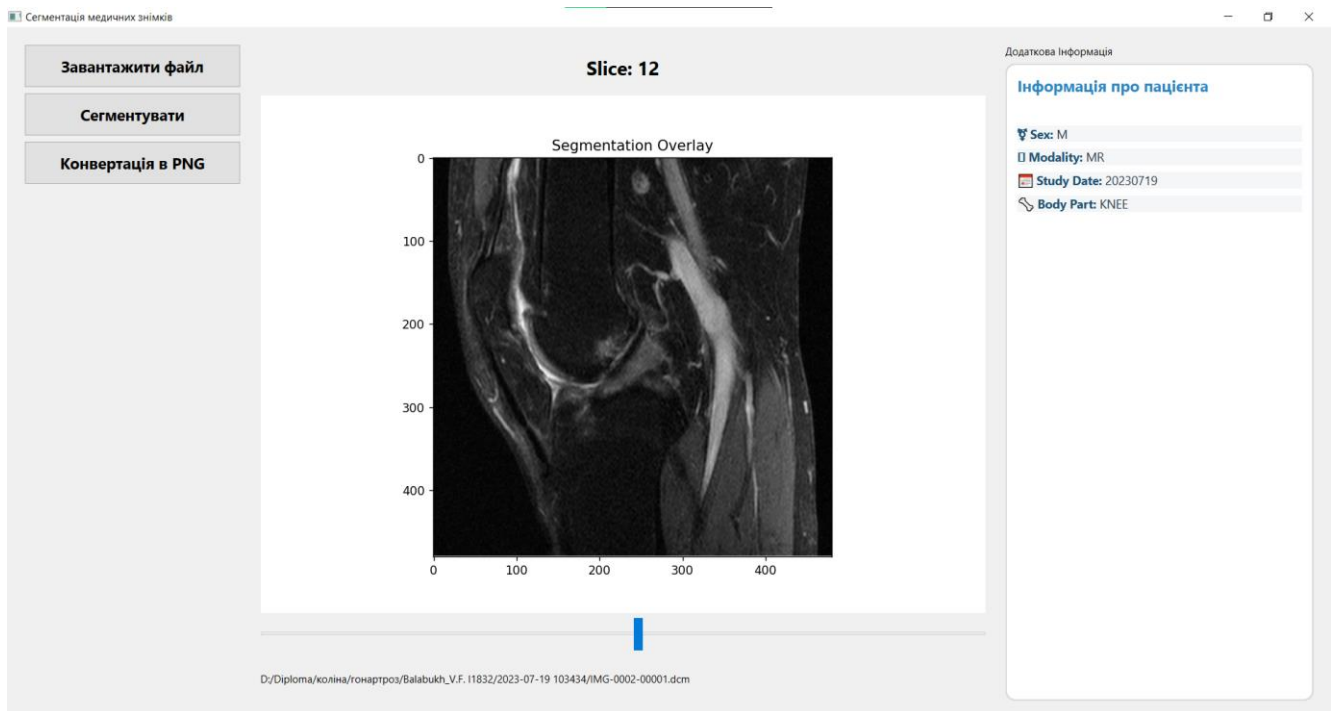


Рисунок 5.2 – Вікно з завантаженим МРТ-знімком

У полі справа можна побачити інформацію про те що тип знімку МРТ, досліджувана частина тіла – коліна, стать пацієнта – чоловіча. Дослідження було зроблено 19 серпня 2023 року.

5.2 Сегментація КТ та МРТ-знімків

Для перевірки роботи завантажимо з тренувального набору даних МРТ-знімок у форматі MHD/RAW(рис. 5.3).

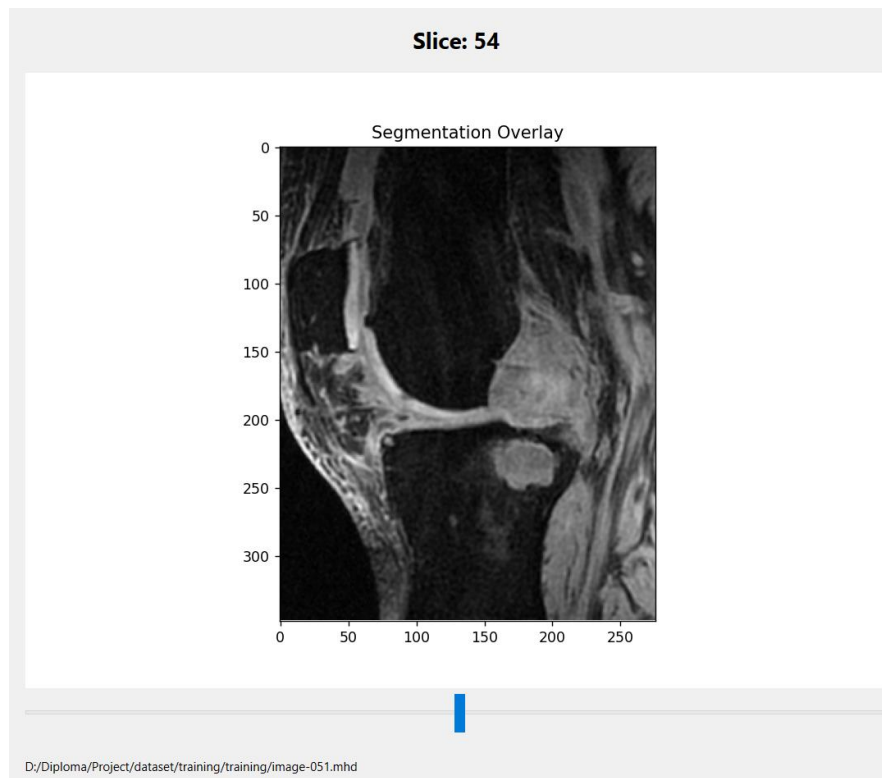


Рисунок 5.3 – Завантажений МРТ-знімок для сегментації

Тепер перевіримо, натиснувши на кнопку «Сегментувати». З'явиться діалогове вікно, яке буде відображати прогрес виконаної роботи (рис. 5.4)

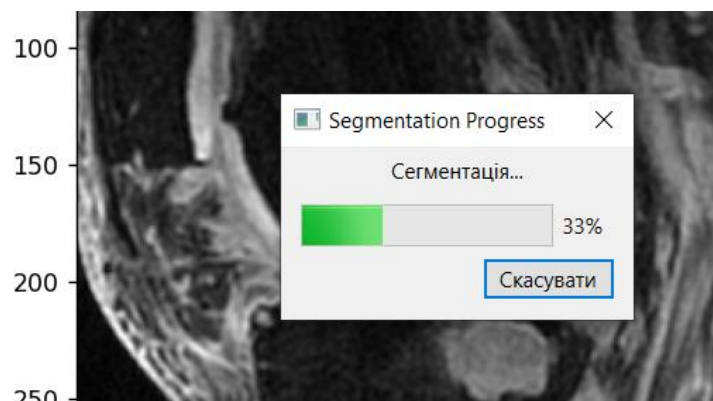


Рисунок 5.4 – Відображення прогресу сегментації

По завершенню процесу сегментації, з'являться результати у вигляді накладених масок на знімок (рис. 5.5).

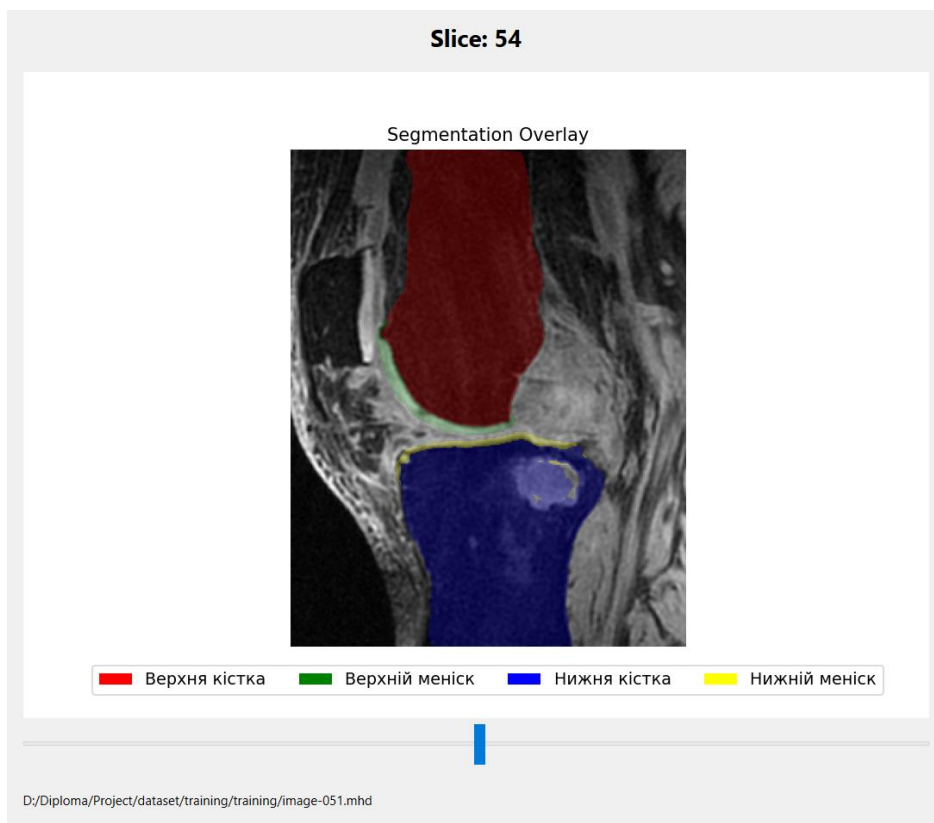


Рисунок 5.5 – Результат сегментації МРТ-знімку

Користувач може продовжувати навігацію між зрізами за допомогою повзунка — на кожному зрізі результати сегментації відображатимуться автоматично.

5.3 Збереження результатів

Для того, щоб зберегти результати сегментації користувач може натиснути кнопку «Конвертувати в PNG». Після цього користувачу потрібно буде обрати директорію куди зберегти. Зробивши вибір, система конвертує 2D-зрізи у зображення формату PNG. Перевірка збережених даних зображена на рисунку 5.6:

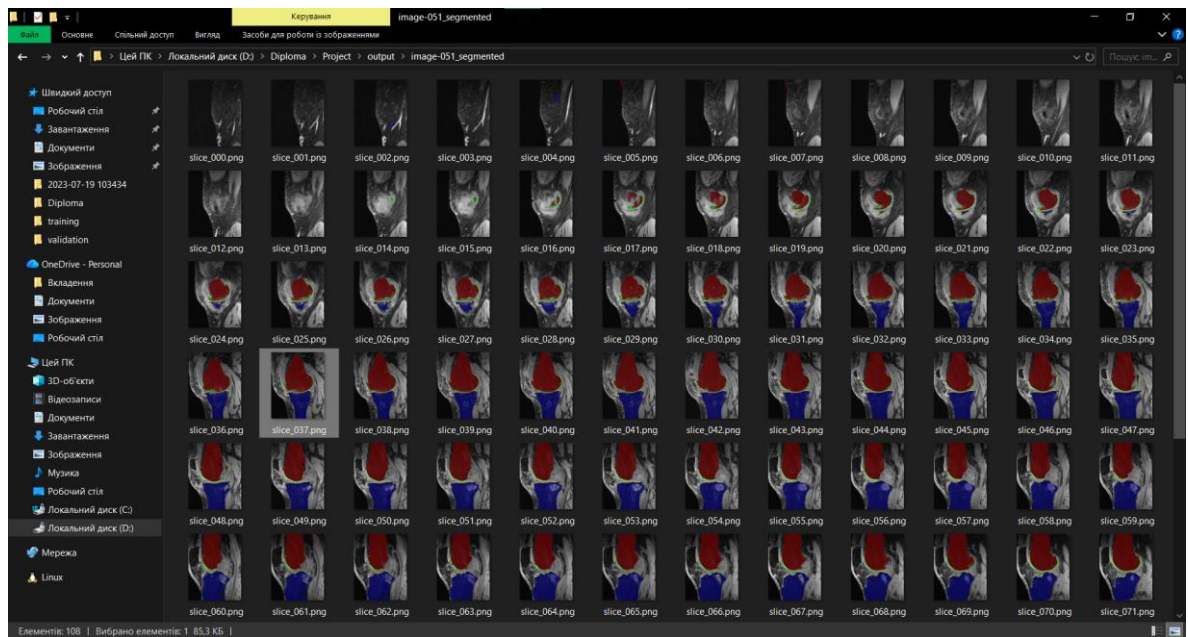


Рисунок 5.6 – Збереження сегментованих 2D-зрізів

Якщо сегментація не виконувалась, зберігаються оригінальні зрізи без оверлею. Після успішного збереження відображається повідомлення із зазначенням шляху до збережених файлів.

Висновки до розділу 5

У п'ятому розділі було продемонстровано процес взаємодії користувача з розробленою системою аналізу КТ та МРТ-знімків для виділення структур колінного суглобу.

Створений графічний інтерфейс забезпечує повний робочий цикл користувача – від завантаження медичного знімку до збереження результатів сегментації. Він спроектований таким чином, що лікар міг виконувати всі необхідні операції без додаткових технічних знань.

Результати сегментації відображаються у вигляді кольорового накладання отриманої маски безпосередньо поверх оригінального знімка, що дозволяє лікарю наочно оцінювати виділені анатомічні структури в контексті вихідного зображення. Крім того, передбачена можливість збереження результатів у форматі PNG для подальшого використання в діагностичному процесі.

ВИСНОВКИ

У даній дипломній роботі було створено систему аналізу МРТ та КТ-знімків для виділення структур колінного суглобу за допомогою нейронних мереж. Даний застосунок працює з 2D або 3D медичними зображення у форматі DICOM або парі файлів MHD/RAW.

Було проаналізовано предметну область, розглянуто анатомічну будову колінного суглоба та обґрунтовано актуальність автоматизації процесу сегментації медичних зображень. Визначено основні вимоги до розроблюваної системи з точки зору точності, стабільності та зручності використання.

Здійснено розгляд існуючих програмних рішень в галузі обробки медичних даних та використання методів машинного навчання для задач сегментації. Програми 3D Slicer та ITK-Snap є досить продвинутими та популярними інструментами в даній сфері. Дані застосунки не містять моделі сегментації колінного суглобу та мають досить складний інтерфейс.

Окрім того, було досліджено наявні архітектури нейронних мереж для сегментації зображень. Вибір впав на модель U-Net. Було реалізовано алгоритм навчання вибраної мережі.

Було проведено дослідження для можливого покращення роботи моделі, шляхом розширення спектру знімків для сегментації, завдяки додаванням у набір даних для навчання знімків, зроблених не лише у сагітальній площині, але й і у аксіональній та фронтальній.

Розроблено простий графічний інтерфейс користувача, який дозволяє користувачу взаємодіяти з додатком та переглядати медичні зображення і результат їх сегментації.

Система може виступати як допоміжний інструмент для лікарів, забезпечуючи автоматичне первинне виділення структур та звільняючи фахівця для аналізу складніших клінічних випадків, що потребують поглибленої експертної оцінки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. AI для бізнесу: з чого почати впровадження штучного інтелекту у власний бізнес. *Ukrainian Digital Community*. 13.08.2025. URL: <https://ukrainiandigital.com/ai-dlia-biznesu-z-choho-pochaty-vprovadzhennia-shtuchnoho-intelektu-u-vlasnyy-biznes/#shtuchniy-intelekt-u-schodenniy-praktitsi> (access date: 05.06.2026).
2. Artificial Neural Network. *IT Wiki. Data science*. 31.07.2023. URL: <https://itwiki.dev/data-science/ml-reference/ml-glossary/artificial-neural-network> (access date: 05.06.2026).
3. Георгій К. Як ми тренуємо ML-моделі для створення сегментаційних масок. *DOU. Форум*. 18.09.2024. URL: <https://dou.ua/forums/topic/50396/> (access date: 05.06.2026).
4. All about MHD Files. *FILEExt*. URL: <https://filext.com/file-extension/MHD> (access date: 05.06.2026).
5. LTS Global Digital Services. An Extensive Overview: 3 types of image segmentation. *LinkedIn*. 27.05.2024. URL: <https://www.linkedin.com/pulse/extensive-overview-3-types-image-segmentation-ltsgds-4hkmc> (access date: 05.06.2026).
6. 3D Slicer image computing platform. *3D Slicer*. URL: <https://www.slicer.org/> (access date: 05.06.2026).
7. ITK-Snap. *IEEE EMBS. IEEE Pulse*. 27.07.2017. URL: <https://www.embs.org/pulse/articles/itk-snap/> (access date: 05.06.2026).
8. *ITK-SNAP. Home Page*. URL: <https://www.itksnap.org/pmwiki/pmwiki.php> (access date: 05.06.2026).
9. PSPNet (Pyramid Scene Parsing Network) for Image Segmentation. *GeekforGeeks*. URL: <https://www.geeksforgeeks.org/computer-vision/pspnet-pyramid-scene-parsing-network-for-image-segmentation/> (access date: 05.06.2026).

10. Abirami V. A guide on U-Net architecture and its applications. *Ultralytics*. 15.07.2025. URL: <https://www.ultralytics.com/blog/a-guide-on-u-net-architecture-and-its-applications#the-history-of-image-segmentation> (access date: 05.06.2026).
11. U-Net. *ScienceDirect*. URL: <https://www.sciencedirect.com/topics/computer-science/u-net> (access date: 05.06.2026).
12. Python the language of today and tomorrow. *Python Institute*. URL: <https://pythoninstitute.org/about-python> (access date: 05.06.2026).
13. Omar A. Introducing PyDicom, Its Classes, Methods, and Attributes. *Medium. TDS Archive*. 04.01.2023. URL: <https://medium.com/data-science/introducing-pydicom-its-classes-methods-and-attributes-518c1d71162> (access date: 05.06.2026).
14. What is NumPy?. *Boston University. Faculty of Computing & Data Sciences*. URL: <https://www.bu.edu/cds-faculty/stay-connected/data-science-resources/what-is-numpy/> (access date: 05.06.2026).
15. How to take PyTorch?. *Codefinity*. URL: <https://codefinity.com/ua/courses/v2/1dd2b0f6-6ec0-40e6-a570-ed0ac2209666/8a329f24-2e0c-4e15-bc32-802e817f6856/72b7811c-1d6e-4bef-b86e-e7640359b773> (access date: 05.06.2026).
16. Kevin B. Matplotlib Tutorial: Line, Bar, and Scatter Plots in Python. *Datacamp. Tutorials*. URL: <https://www.datacamp.com/tutorial/matplotlib-tutorial-python> (access date: 05.06.2026).
17. Xavier S. PyCharm Community Edition. *Softonic. Web Development*. URL: <https://pycharm-community-edition.en.softonic.com/> (access date: 05.06.2026).
18. What are convolutional neural networks?. *IBM. Think*. URL: <https://www.ibm.com/think/topics/convolutional-neural-networks> (access date: 05.06.2026).
19. How U-net works?. *Esri Developer. ArcGIS API for Python*. URL: <https://developers.arcgis.com/python/latest/guide/how-unet-works/> (access date: 05.06.2026).

20. SKI10. *Grand Challenge*. URL: <https://ski10.grand-challenge.org/> (access date: 05.06.2026).

21. Intersection Over Union (IoU): From Theory to Practice. *Lightly. Blog*. URL: <https://www.lightly.ai/blog/intersection-over-union> (access date: 05.06.2026).

ДОДАТОК А Лістинг розробленої системи

model.py:

```
import torch.nn as nn
import torch.nn.functional as F
import torch
import torch.optim as optim
from data_module.DataLoader import load_dataset
```

```
# U-NET
```

```
class UNet(nn.Module):
```

```
    def __init__(self):
```

```
        super(UNet, self).__init__()
```

```
        # Encoder
```

```
        self.enc1 = nn.Sequential(
```

```
            nn.Conv2d(1, 32, 3, padding=1),
```

```
            nn.ReLU(inplace=True),
```

```
            nn.Conv2d(32, 32, 3, padding=1),
```

```
            nn.ReLU(inplace=True)
```

```
        )
```

```
        self.pool1 = nn.MaxPool2d(2)
```

```
        self.enc2 = nn.Sequential(
```

```
            nn.Conv2d(32, 64, 3, padding=1),
```

```
            nn.ReLU(inplace=True),
```

```
            nn.Conv2d(64, 64, 3, padding=1),
```

```
            nn.ReLU(inplace=True)
```

```
        )
```

```
        self.pool2 = nn.MaxPool2d(2)
```

```
        self.enc3 = nn.Sequential(
```

```
            nn.Conv2d(64, 128, 3, padding=1),
```

```
            nn.ReLU(inplace=True),
```

```
            nn.Conv2d(128, 128, 3, padding=1),
```

```
            nn.ReLU(inplace=True)
```

```

)
self.pool3 = nn.MaxPool2d(2)

self.enc4 = nn.Sequential(
    nn.Conv2d(128, 256, 3, padding=1),
    nn.ReLU(inplace=True),

    nn.Conv2d(256, 256, 3, padding=1),
    nn.ReLU(inplace=True)
)

self.pool4 = nn.MaxPool2d(2)

# Bottleneck
self.bottleneck = nn.Sequential(
    nn.Conv2d(256, 512, 3, padding=1),
    nn.ReLU(inplace=True),

    nn.Conv2d(512, 512, 3, padding=1),
    nn.ReLU(inplace=True)
)

# Decoder
self.up4 = nn.ConvTranspose2d(512, 256, 2, stride=2)

self.dec4 = nn.Sequential(
    nn.Conv2d(512, 256, 3, padding=1),
    nn.ReLU(inplace=True),

    nn.Conv2d(256, 256, 3, padding=1),
    nn.ReLU(inplace=True)
)

self.up3 = nn.ConvTranspose2d(256, 128, 2, stride=2)

self.dec3 = nn.Sequential(
    nn.Conv2d(256, 128, 3, padding=1),
    nn.ReLU(inplace=True),

    nn.Conv2d(128, 128, 3, padding=1),

```

```

        nn.ReLU(inplace=True)
    )

    self.up2 = nn.ConvTranspose2d(128, 64, 2, stride=2)

    self.dec2 = nn.Sequential(
        nn.Conv2d(128, 64, 3, padding=1),
        nn.ReLU(inplace=True),

        nn.Conv2d(64, 64, 3, padding=1),
        nn.ReLU(inplace=True)
    )

    self.up1 = nn.ConvTranspose2d(64, 32, 2, stride=2)

    self.dec1 = nn.Sequential(
        nn.Conv2d(64, 32, 3, padding=1),
        nn.ReLU(inplace=True),

        nn.Conv2d(32, 32, 3, padding=1),
        nn.ReLU(inplace=True)
    )

    self.conv_last = nn.Conv2d(32, 5, 1)

def forward(self, x):
    e1 = self.enc1(x)
    e2 = self.enc2(self.pool1(e1))
    e3 = self.enc3(self.pool2(e2))
    e4 = self.enc4(self.pool3(e3))

    b = self.bottleneck(self.pool4(e4))

    d4 = self.dec4(torch.cat([self.up4(b), e4], dim=1))
    d3 = self.dec3(torch.cat([self.up3(d4), e3], dim=1))
    d2 = self.dec2(torch.cat([self.up2(d3), e2], dim=1))
    d1 = self.dec1(torch.cat([self.up1(d2), e1], dim=1))

    return self.conv_last(d1)

```

```

class DiceLoss(nn.Module):
    def __init__(self, smooth=1e-5):
        super(DiceLoss, self).__init__()
        self.smooth = smooth

    def forward(self, outputs, targets):

        num_classes = outputs.shape[1]

        # softmax probabilities
        outputs = F.softmax(outputs, dim=1)

        # one-hot targets
        targets_onehot = F.one_hot(
            targets,
            num_classes=num_classes
        ).permute(0, 3, 1, 2).float()

        # flatten
        outputs = outputs.contiguous().view(outputs.shape[0], num_classes, -1)
        targets_onehot = targets_onehot.contiguous().view(
            targets_onehot.shape[0],
            num_classes,
            -1
        )

        intersection = (outputs * targets_onehot).sum(dim=2)

        dice = (2. * intersection + self.smooth) / (
            outputs.sum(dim=2) +
            targets_onehot.sum(dim=2) +
            self.smooth
        )

        dice = dice[:, 1:]
        dice_loss = 1 - dice.mean()

        return dice_loss

```

```
# FOCAL LOSS
```

```
class FocalLoss(nn.Module):
```

```
    def __init__(self, alpha=None, gamma=2):
        super(FocalLoss, self).__init__()
```

```
        self.alpha = alpha
        self.gamma = gamma
```

```
    def forward(self, outputs, targets):
```

```
        ce_loss = F.cross_entropy(
            outputs,
            targets,
            reduction='none',
            weight=self.alpha
        )
```

```
        pt = torch.exp(-ce_loss)
```

```
        focal_loss = ((1 - pt) ** self.gamma) * ce_loss
```

```
        return focal_loss.mean()
```

```
# DICE SCORE
```

```
def multiclass_dice_score(preds, targets, num_classes=5):
```

```
    preds = torch.argmax(preds, dim=1)
    dices = []
```

```
    for cls in range(1, num_classes):
```

```
        pred_cls = (preds == cls).float()
        target_cls = (targets == cls).float()
```

```
        intersection = (pred_cls * target_cls).sum()
```

```
        dice = (2 * intersection + 1e-5) / (pred_cls.sum() + target_cls.sum() + 1e-5)
        dices.append(dice.item())
```

```

    return sum(dices) / len(dices)

# IOU SCORE
def multiclass_iou_score(preds, targets, num_classes=5):

    preds = torch.argmax(preds, dim=1)

    ious = []

    for cls in range(1, num_classes):

        pred_cls = (preds == cls).float()
        target_cls = (targets == cls).float()

        intersection = (pred_cls * target_cls).sum()

        union = (
            pred_cls.sum() +
            target_cls.sum() -
            intersection
        )

        iou = (intersection + 1e-5) / (union + 1e-5)
        ious.append(iou.item())

    return sum(ious) / len(ious)

def train_model():
    print("Program started")

    device = "cuda" if torch.cuda.is_available() else "cpu"

    print(f"Device: {device}")

    dataset, train_loader, val_loader, test_loader = load_dataset()

    print(f"Total samples: {len(dataset)}")
    print(f"Train batches: {len(train_loader)}")

```

```

print(f"Val batches: {len(val_loader)}")
print(f"Test batches: {len(test_loader)}")

# MODEL
model = UNet().to(device)
weights = torch.tensor([0.5, 1.0, 1.0, 2.0, 2.0]).to(device)

focal_loss = FocalLoss(alpha=weights, gamma=2)

dice_loss = DiceLoss()
optimizer = optim.Adam(model.parameters(), lr=1e-4)

# TRAINING
best_val_loss = float("inf")

for epoch in range(35):
    print(f"--- Epoch {epoch + 1} started ---")

    model.train()
    train_loss = 0
    train_dice = 0
    train_iou = 0

    for batch_idx, (images, masks) in enumerate(train_loader):

        images = images.to(device)
        masks = masks.to(device)

        masks = masks.squeeze(1).long()

        if batch_idx == 0 and epoch == 0:
            print("Mask classes:", torch.unique(masks))

        outputs = model(images)

        loss_focal = focal_loss(outputs, masks)
        loss_dice = dice_loss(outputs, masks)

        loss = loss_focal + loss_dice

```

```

optimizer.zero_grad()
loss.backward()
optimizer.step()

train_loss += loss.item()

dice = multiclass_dice_score(outputs, masks)
train_dice += dice

iou = multiclass_iou_score(outputs, masks)
train_iou += iou

# простий лог батчів
if batch_idx % 10 == 0:
    print(
        f"Batch {batch_idx}/{len(train_loader)} | "
        f"Loss: {loss.item():.4f} | "
        f"Dice: {dice:.4f} | "
        f"IoU: {iou:.4f}"
    )

train_loss /= len(train_loader)
train_dice /= len(train_loader)
train_iou /= len(train_loader)

# Validation
model.eval()
val_loss = 0
val_dice = 0
val_iou = 0

with torch.no_grad():
    for images, masks in val_loader:
        images = images.to(device)
        masks = masks.to(device)

        masks = masks.squeeze(1).long()

        outputs = model(images)

```

```

    loss_focal = focal_loss(outputs, masks)
    loss_dice = dice_loss(outputs, masks)

    loss = loss_focal + loss_dice

    val_loss += loss.item()

    dice = multiclass_dice_score(outputs, masks)
    val_dice += dice

    iou = multiclass_iou_score(outputs, masks)
    val_iou += iou

val_loss /= len(val_loader)
val_dice /= len(val_loader)
val_iou /= len(val_loader)

print(
    f"Train Loss: {train_loss:.4f} | "
    f"Train Dice: {train_dice:.4f} | "
    f"Train IoU: {train_iou:.4f}"
)

print(
    f"Val Loss: {val_loss:.4f} | "
    f"Val Dice: {val_dice:.4f} | "
    f"Val IoU: {val_iou:.4f}"
)

# SAVE BEST MODEL
if val_loss < best_val_loss:
    best_val_loss = val_loss

    torch.save(model.state_dict(), "best_unet_model_6.pth")

    print("Best model saved")

print("Training finished")

# TESTING

```

```

print("\n=== TESTING ===")

model.load_state_dict(torch.load("best_unet_model_6.pth"))
model.eval()

test_dice = 0
test_iou = 0

with torch.no_grad():

    for images, masks in test_loader:
        images = images.to(device)
        masks = masks.to(device)

        masks = masks.squeeze(1).long()

        outputs = model(images)

        dice = multiclass_dice_score(outputs, masks)
        test_dice += dice

        iou = multiclass_iou_score(outputs, masks)
        test_iou += iou

test_dice /= len(test_loader)
test_iou /= len(test_loader)

print(f"Final Test Dice Score: {test_dice:.4f}")
print(f"Final Test IoU Score: {test_iou:.4f}")

```



Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Навчально-науковий інститут атомної та теплової енергетики
Кафедра інженерії програмного забезпечення в енергетиці

СИСТЕМА АНАЛІЗУ МРТ-ЗНІМКІВ ДЛЯ ВИДІЛЕННЯ СТРУКТУР КОЛІННОГО СУГЛОБУ

виконав: студент групи ТВ-22 Оніщук Максим Іванович

керівник: старший викладач Дацюк Оксана Антонівна

2026

Актуальність теми

Розроблена система дозволяє автоматично виділяти кістки та меніски колінного суглобу за допомогою нейронної мережі типу U-Net, що спрощує обробку медичних зображень та прискорює аналіз даних.

Система може бути корисною для:

- лікарів-рентгенологів;
- ортопедів і травматологів;
- медичних дослідників;
- студентів та викладачів;
- розробників медичних AI-систем.



Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

2/18

Постановка задачі

Мета: Розробка системи аналізу МРТ та КТ-знімків для виділення структур колінного суглобу за допомогою нейронних мереж

Об'єктом роботи є процес семантичної сегментації КТ та МРТ-знімків за допомогою нейронних мереж.

Предметом роботи є модель нейронної мережі для сегментації медичних зображень та алгоритм її навчання.

Завдання:

1. Провести аналіз предметної області та існуючих рішень.
2. Розробити архітектуру моделі нейромережі та написання модулів машинного навчання, даних та інтерфейсу
3. Протестувати реалізовану модель та провести оцінювання її точності
4. Розробити графічний інтерфейс користувача

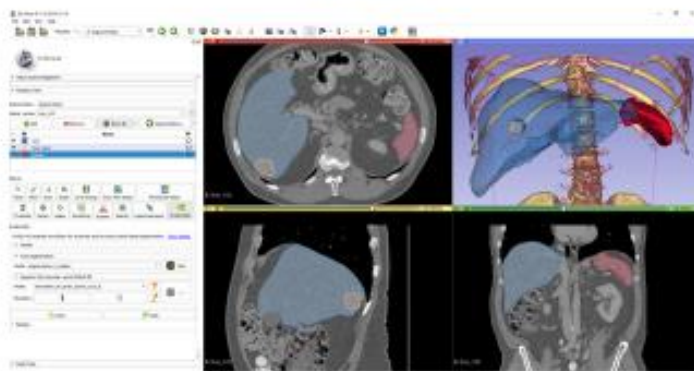


Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

3 / 18

Огляд існуючих рішень

3D Slicer



ITK-Snap



Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

4 / 18

Засоби розробки



Середовище розробки PyCharm



Бібліотека для візуалізації зображень



Мова програмування Python



Бібліотека для створення графічного інтерфейсу



Бібліотека для роботи з машинним навчанням



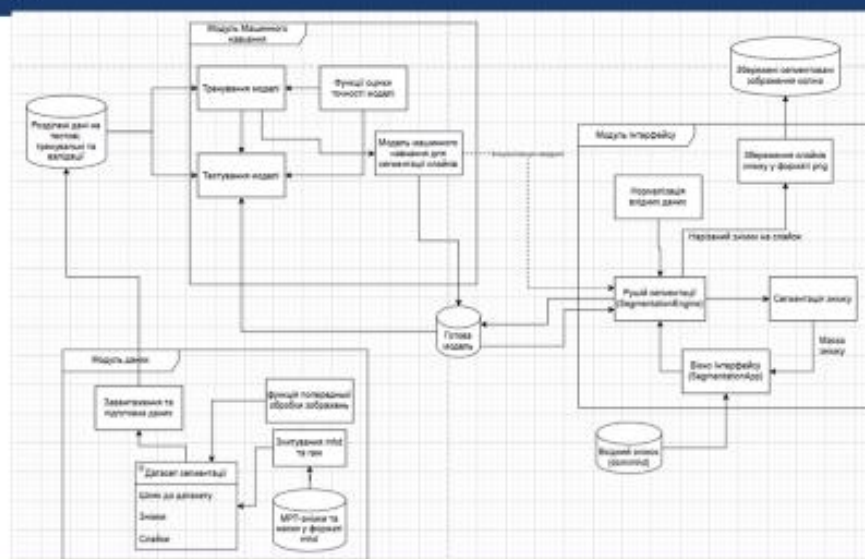
Стандартний формат для зберігання медичних знімків



Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

5 / 18

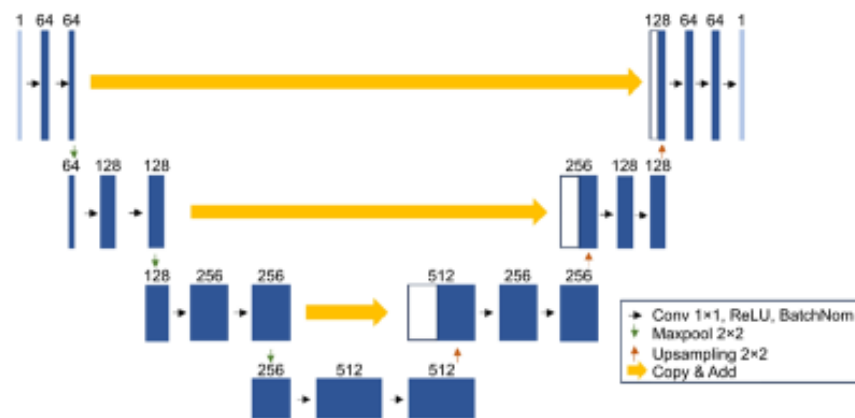
Архітектура ПЗ



Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

6 / 18

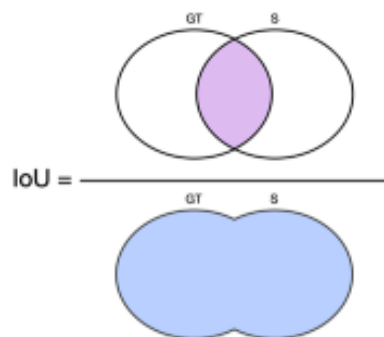
Структура мережі



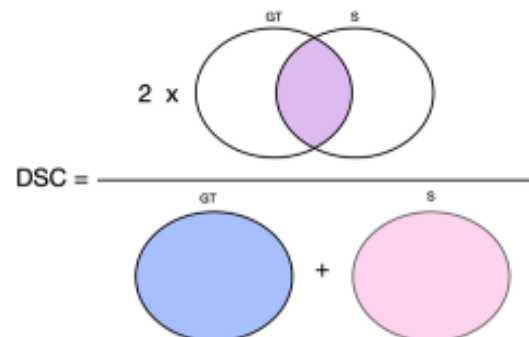
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

7 / 18

Використання метрик точності моделі сегментації



IoU (Intersection per union) Score



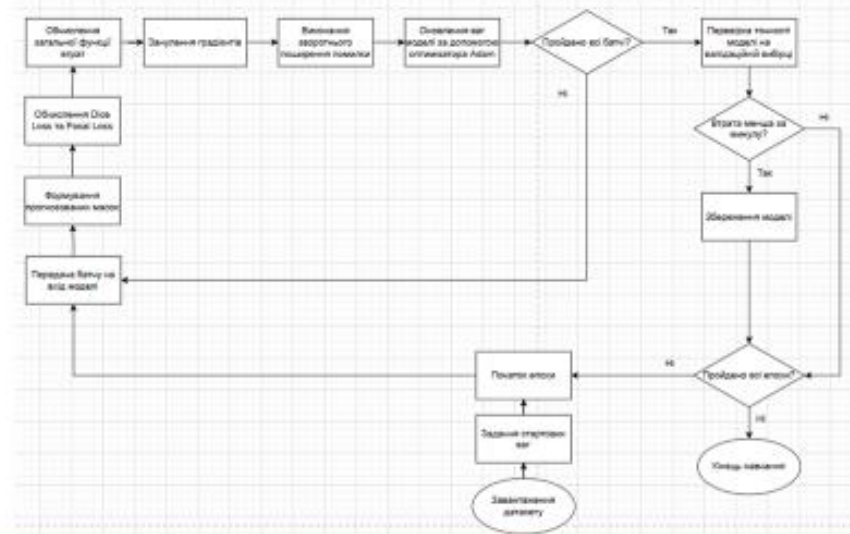
Dice Score



Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

8 / 18

Алгоритм навчання моделі



Процес тренування

Використані параметри навчання:

- 35 епох
- Розмір батчів – 24 зрізів
- Стартові ваги класів:
 - 0.5 – фон
 - 1.0 – верхня кістка
 - 1.0 – нижня кістка
 - 2.0 – верхній меніск
 - 2.0 – нижній меніск

```

Batch 270/272 | Loss: 0.2947 | Dice: 0.9019 | IoU: 0.8762
Train Loss: 0.2677 | Train Dice: 0.8332 | Train IoU: 0.7535
Val Loss: 0.4584 | Val Dice: 0.8452 | Val IoU: 0.7512
Best model saved
Training finished

=== TESTING ===
FileNotFoundError: [Errno 2] No such file or directory: 'best_model_e.pth'
Final Test Dice score: 0.8407
Final Test IoU score: 0.7455

Process finished with exit code 0
    
```

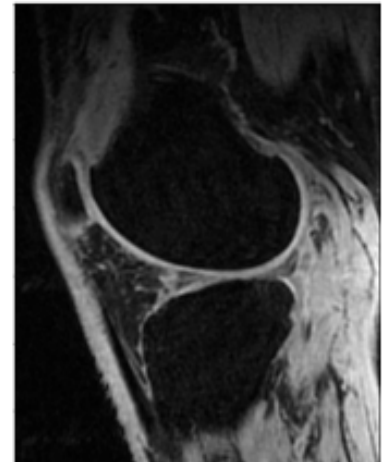
Результат тренування та тестування моделі



Набір МРТ-знімків для навчання моделі

SKI10 Dataset

- Містить 100 МРТ-знімків, в середньому по 100 зрізів
- Містяться у форматі MHD (хедерний файл) та RAW (сирі пікселі)
- Режим T1
- Зроблені у сагітальній площині
- Дані розділені наступним чином:
 - 60 знімків – тренувальна вибірка
 - 20 знімків – валідаційна вибірка
 - 20 знімків – тестувальна вибірка



Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

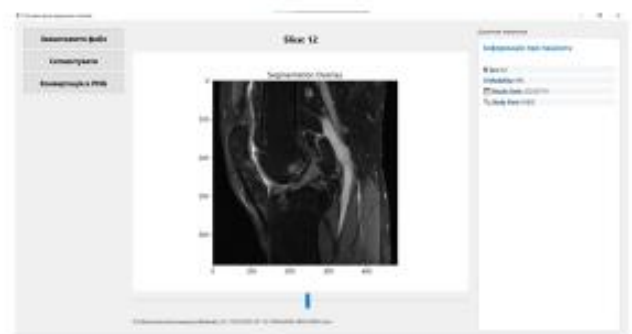
11 / 18

Інтерфейс користувача



Відкриється файл зі зрізами знімку та інформацією про дослідження (у випадку DICOM)

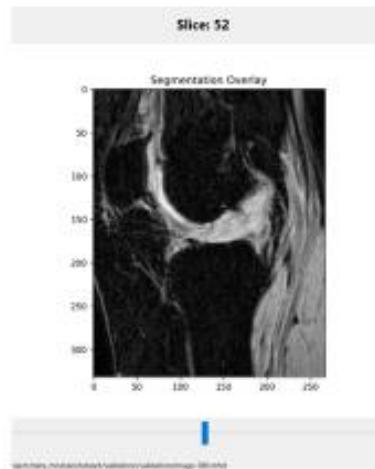
Натискаємо на «Завантажити файл» та вибираємо потрібний файл у провіднику



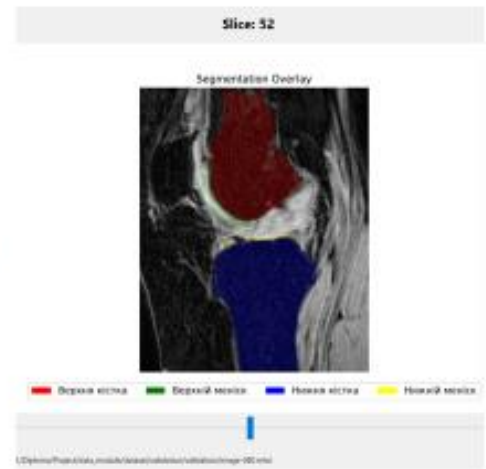
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

12 / 18

Вхідні дані



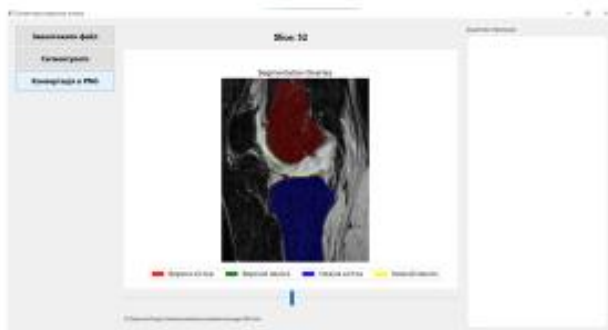
Відкритий MHD-файл знімку



Результат сегментації



Інтерфейс користувача



Збережені сегментовані зрізи будуть збережені у PNG наступним чином

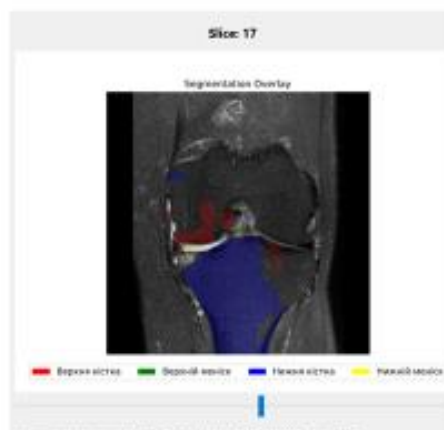
Після проведення сегментації натискаємо «Конвертувати в PNG» та вибираємо директорію куди зберігаємо



Проблеми системи

Система містить наступні недоліки:

- Обмежений спектр сегментації знімків. Тобто система працює добре лише при наступних умовах:
 - МРТ-знімок у режимі T1
 - Знімок повинен бути зроблений лише у сагітальній площині
- Відсутність серед тренувальних даних знімків з різноманітними патологіями.
- При відсутності на зображенні жодних структур модель може видавати випадкові артефакти



Результат сегментації МРТ-знімку у фронтальній площині

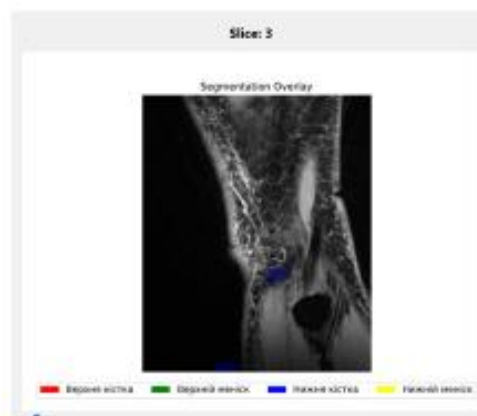
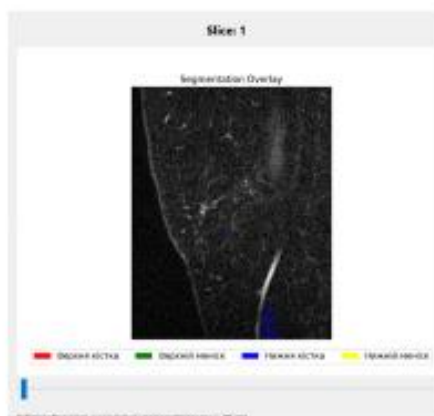


Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

15 / 18

Проблеми системи

Проблема відсутності об'єктів



Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

16 / 18

Висновки

Результати: Розроблено систему аналізу МРТ-знімків для сегментації структур колінного суглобу з використанням методів глибокого навчання

Досягнення:

- Реалізовано повний цикл обробки медичних зображень у форматах DICOM та MHD.
- Розроблено алгоритм навчання нейронної мережі типу U-Net
- Реалізовано виділення анатомічних структур колінного суглобу: кісток та менісків
- Створено інтуїтивно зрозумілий інтерфейс користувача для перегляду знімків та інформації про дослідження з подальшим збереженням роботи.

Практична цінність: Система може використовуватись як допоміжний інструмент для аналізу медичних зображень, напівавтоматичної розмітки даних, візуалізації структур колінного суглобу

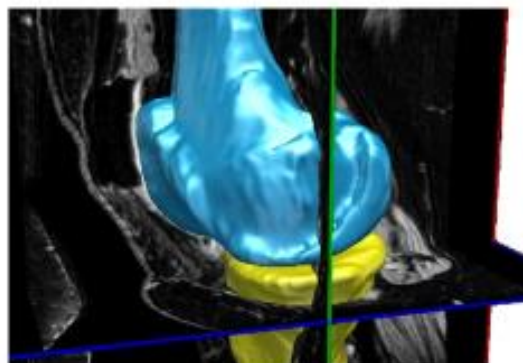


Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

17 / 18

Перспективи розвитку

- Для навчання моделі можна використати додатково інші датасети які містять знімки з різних пристроїв та осей для розширення діапазону знімків які можна сегментувати.
- Окрім того можна додати підтримку роботи з іншими форматами зберігання даних.
- Також можна здійснити перехід від 2D до повноцінної 3D-сегментації даних, з метою покращення точності аналізу.
- Можна створити 3D моделювання суглобу, отриманого в результаті сегментації знімку



Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

18 / 18



Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute"