Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Аналіз медичних зображень за допомогою глибокого навчання

Виконав: студент IV курсу, групи СН-42
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

_____ Поліщук О.Р.
(підпис) (прізвище та ініціали)

Керівник _____ Фриз М.Є.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Липак Г.І.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ _____
(підпис) (прізвище та ініціали)

Тернопіль
2025

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«_27_» _січня_ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)
за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)
Студенту Поліщук Олександр Романович
(прізвище, ім'я, по батькові)
1. Тема роботи Аналіз медичних зображень за допомогою глибокого навчання

Керівник роботи Фриз Михайло Євгенович, кандидат технічних наук доцент кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
Затверджені наказом ректора від «*07*» *травня* 2025 року № *4/7-444*
2. Термін подання студентом завершеної роботи 26 червня 2025 р.
3. Вихідні дані до роботи Бібліотеки PyTorch, Albumentations, Tkinter/CustomTkinter для інтерфейсу абір даних MURA (рентгенівські знімки з розміткою ереломів/нормальних станів) оделі ResNet, DenseNet, EfficientNet, ConvNeXt, VisionTransformer, а також дані зображень (формат PNG, JPG, TIFF) і параметри навчання (кількість епох, розмір батчу, швидкість навчання).
4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Аналіз предметної області та постановка завдання. 2. Проєктна розробка системи аналізу рентгенівських знімків. 3. Практична реалізація та оцінка системи. 4. Безпека життєдіяльності, основи охорони праці. Висновки. Перелік джерел. Додаток А Код програми

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Тема роботи. 2. Актуальність теми. 3. Мета та завдання. 4. Короткий опис роботи. 5. Архітектура системи. 6. Використані технології. 7. Модулі системи. 8. Інтерфейс Користувача. 9. Дані та навчання. 10. Результати тестування. 11. Висновки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Сенчишин Віктор Степанович, доцент кафедри МТ	02.06.2025	05.06.2025

7. Дата видачі завдання 27.01.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	27.01.2025	*Виконано*
2.	Підбір та опрацювання літературних джерел по темі кваліфікаційної роботи	31.01.2025-04.02.2025	*Виконано*
3.	Розроблення та програмна реалізація системи аналізу медичних зображень	05.02.2025-10.02.2025	*Виконано*
4.	Оформлення розділу «Аналіз предметної області та	02.06.2025-05.06.2025	*Виконано*
	Постановка завдання»		
5.	Оформлення розділу «Проєктна розробка системи	06.06.2025-09.06.2025	*Виконано*
	Аналізу рентгенівських знімків»		
6.	Оформлення розділу «Практична реалізація та оцінка	10.05.2025-11.06.2025	*Виконано*
	системи»		
7.	Виконання завдання до підрозділу «Безпека життєдіяльності»	12.06.2025-13.06.2025	*Виконано*
8.	Виконання завдання до підрозділу «Основи охорони	14.06.2025-15.06.2025	*Виконано*
	праці»		
9.	Оформлення кваліфікаційної роботи	16.06.2025-17.06.2025	*Виконано*
10.	Нормоконтроль	18.06.2025-20.06.2025	*Виконано*
11.	Перевірка на плагіат	21.06.2025	*Виконано*
12.	Попередній захист кваліфікаційної роботи	22.06.2025	*Виконано*
13.	Захист кваліфікаційної роботи	27.06.2025	

Студент _____ Поліщук О.Р.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Фриз М.Є.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Тема: Аналіз медичних зображень за допомогою глибокого навчання // Кваліфікаційна робота освітнього рівня «Бакалавр» // Поліщук Олександр Романович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СН-42 // Тернопіль, 2025 // С. 62, рис. – 18, табл. – 3, кресл. – 0, додат. – 1, бібліогр. – 46.

Ключові слова: аналіз медичних зображень, глибоке навчання, рентгенівські знімки, конволюційні нейронні мережі, сегментація кісток, виявлення переломів, інтерпретація результатів, комп'ютерний зір.

Кваліфікаційна робота присвячена розробці системи для автоматизованого аналізу рентгенівських знімків із використанням методів глибокого навчання з метою виявлення переломів кісток. У першому розділі проведено аналіз сучасних методів обробки медичних зображень, розглянуто архітектури нейронних мереж, таких як ResNet, DenseNet, EfficientNet, та методи сегментації й інтерпретації. Висвітлено актуальність автоматизації діагностики для підвищення точності та зменшення суб'єктивізму.

У другому розділі описано проєктну розробку модульної системи, яка поєднує методи комп'ютерного зору та глибокого навчання. Запропоновано архітектуру з модулями попередньої обробки, сегментації, класифікації та інтерпретації. Реалізовано графічний інтерфейс для зручної взаємодії користувачів.

У третьому розділі представлено практичну реалізацію системи, результати тестування на наборі даних MURA та оцінку її ефективності. Продемонстровано високу точність класифікації і чутливість до аномалій, а також зручність інтерфейсу для медичних фахівців.

Об'єкт дослідження: методи аналізу медичних зображень.

Предмет дослідження: розробка системи для виявлення переломів на рентгенівських знімках із застосуванням глибокого навчання.

ANNOTATION

Topic: Analysis of Medical Images Using Deep Learning // Bachelor's Degree Qualification Work // Polishchuk Oleksandr Romanovych // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Computer Science, Group SN-42 // Ternopil, 2025 // P. 62, fig. – 18, tab. – 3, draw. – 0, append. – 1, bibliogr. – 46.

Keywords: medical image analysis, deep learning, X-ray images, convolutional neural networks, bone segmentation, fracture detection, result interpretation, computer vision.

The qualification work is devoted to the development of a system for automated analysis of X-ray images using deep learning methods to detect bone fractures. The first section analyzes modern methods of medical image processing, reviews neural network architectures such as ResNet, DenseNet, and EfficientNet, and discusses segmentation and interpretation techniques. The relevance of diagnostic automation for improving accuracy and reducing subjectivity is highlighted.

The second section describes the design and development of a modular system that integrates computer vision and deep learning methods. A system architecture is proposed, consisting of modules for preprocessing, segmentation, classification, and interpretation. A user-friendly graphical interface is implemented to facilitate interaction.

The third section presents the practical implementation of the system, test results on the MURA dataset, and an evaluation of its effectiveness. The system demonstrates high classification accuracy and sensitivity to anomalies, as well as a user-friendly interface for medical professionals.

Object of study: methods of medical image analysis.

Subject of study: development of a system for detecting fractures in X-ray images using deep learning.

ПЕРЕЛІК СКОРОЧЕНЬ

ШІ – Штучний інтелект

CNN – Згорткова нейронна мережа

КТ – Комп’ютерна томографія

МРТ – Магнітно-резонансна томографія

CLAHE – Адаптивна еквалізація гістограми з обмеженням контрасту

U-Net – Архітектура нейронної мережі для сегментації зображень

Grad-CAM – Градієнтна локалізація на основі активації класів

ROC-AUC – Площа під кривою операторської характеристики

API – Інтерфейс програмування додатків

GDPR – Загальний регламент захисту даних

HIPAA – Закон про переносимість та підзвітність медичного страхування

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному світі, з поширенням цифрових технологій і швидким розвитком штучного інтелекту, аналіз медичних зображень став ключовим елементом медичної діагностики. Рентгенівські знімки, які широко використовуються для виявлення патологій, зокрема переломів кісток, потребують швидкого та точного аналізу, що часто ускладнено суб'єктивністю людської оцінки та обмеженістю часу медичних фахівців. Методи глибокого навчання, зокрема згорткові нейронні мережі та ансамблеві підходи, дозволяють автоматизувати обробку зображень, підвищуючи точність діагностики та зменшуючи ймовірність помилок. Таким чином, розробка автоматизованої системи аналізу рентгенівських знімків для виявлення переломів є актуальним напрямком сучасних досліджень у галузі медичних інформаційних технологій, оскільки сприяє підвищенню якості діагностики та ефективності роботи медичних працівників.

Мета і задачі дослідження. Метою даної кваліфікаційної роботи освітнього рівня «Бакалавр» є розробка системи аналізу рентгенівських знімків на основі глибокого навчання, що забезпечує автоматизоване виявлення переломів кісток із високою точністю та інтерпретовністю результатів. Для досягнення поставленої мети необхідно виконати такі завдання:

– проаналізувати сучасний стан досліджень у галузі аналізу медичних зображень та застосування методів глибокого навчання;

– дослідити особливості використання згорткових нейронних мереж і методів комп'ютерного зору для сегментації та класифікації рентгенівських знімків;

– розробити модульну архітектуру системи аналізу зображень із графічним інтерфейсом користувача;

– реалізувати та протестувати систему на основі спроєктованих алгоритмів і моделей;

– оцінити ефективність розробленої системи з точки зору точності класифікації, якості сегментації та зручності використання.

Практичне значення одержаних результатів. Розроблена система забезпечує автоматизацію процесу виявлення переломів на рентгенівських знімках, що сприяє підвищенню точності діагностики та зменшенню навантаження на медичних фахівців. Впровадження системи дозволяє оптимізувати процеси аналізу зображень у медичних закладах, скоротити час діагностики та підвищити надійність результатів завдяки інтерпретованим поясненням прогнозів. Отримані результати можуть бути використані для вдосконалення інших систем медичної діагностики, де автоматизація та точність відіграють ключову роль.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Огляд сучасних методів аналізу медичних зображень

Аналіз медичних зображень є однією з ключових галузей сучасної медичної діагностики, яка забезпечує швидке та точне виявлення патологій, зокрема переломів кісток, пухлин, аномалій м'яких тканин та інших відхилень. З появою технологій штучного інтелекту та глибокого навчання ця сфера зазнала значних змін, що дозволило автоматизувати обробку зображень, підвищити точність діагностики та зменшити залежність від суб'єктивного людського фактору. Сучасні методи аналізу медичних зображень базуються на комп'ютерному зорі, машинному навчанні та глибоких нейронних мережах, які інтегрують складні алгоритми для обробки даних із рентгенівських знімків, комп'ютерної томографії (КТ), магнітно-резонансної томографії (МРТ) та інших модальностей. Ці методи дозволяють не лише виявляти патології, але й проводити сегментацію структур, оцінювати ступінь ураження та прогнозувати перебіг захворювання.

Одним із основних напрямів розвитку аналізу медичних зображень є використання конволюційних нейронних мереж (CNN), які демонструють високу ефективність у задачах класифікації, сегментації та виявлення аномалій. Такі архітектури, як ResNet, DenseNet, EfficientNet, ConvNeXt та Vision Transformer (ViT), широко застосовуються для обробки рентгенівських знімків завдяки їх здатності автоматично виділяти ознаки зображень без необхідності ручного проектування. Наприклад, ResNet із залишковими зв'язками дозволяє ефективно навчати глибокі мережі, уникаючи проблеми зникнення градієнта, що є критичним для обробки складних медичних зображень із великою кількістю деталей [1]. DenseNet, у свою чергу, оптимізує використання пам'яті завдяки щільним з'єднанням між шарами, що сприяє кращій генералізації моделей при аналізі рентгенівських знімків [2]. EfficientNet забезпечує баланс між точністю та обчислювальною ефективністю, що робить її привабливою для використання

в умовах обмежених ресурсів [3]. Новітні архітектури, такі як ConvNeXt, поєднують переваги конволюційних мереж із сучасними підходами до дизайну мереж, що дозволяє досягати високої точності при менших витратах обчислювальних ресурсів [4]. Vision Transformer, який базується на механізмах уваги, ефективно працює з глобальними залежностями в зображеннях, що особливо корисно для аналізу складних структур, таких як кісткові тканини [5].

Важливим аспектом сучасних методів є попередня обробка зображень, яка включає нормалізацію, підвищення контрасту, видалення шуму та аугментацію даних. Такі бібліотеки, як Albumentations, надають широкий набір інструментів для трансформації зображень, що підвищує стійкість моделей до варіацій у даних, таких як зміни яскравості, контрасту чи орієнтації знімків [6]. Наприклад, методи адаптивної еквалізації гістограми (CLAHE) та анізотропної дифузійної фільтрації дозволяють підкреслити деталі кісткових структур, що полегшує їх подальшу сегментацію та аналіз. Сегментація кісткових структур є окремою задачею, яка часто виконується за допомогою алгоритмів на основі порогової обробки, морфологічних операцій або глибоких мереж, таких як U-Net, що дозволяє точно виділяти кістки на рентгенівських знімках для подальшого аналізу аномалій [7].

Виявлення аномалій, таких як переломи, є ще однією важливою задачею, яка реалізується як через класичні методи комп’ютерного зору, так і за допомогою глибокого навчання. Класичні підходи, такі як використання перетворення Хафа для виявлення ліній або аналіз градієнтів інтенсивності, дозволяють ідентифікувати потенційні розриви в кісткових структурах. Однак ці методи часто обмежені чутливістю до шуму та варіацій у зображеннях [8]. Глибоке навчання, зокрема використання ансамблевих моделей, забезпечує вищу точність завдяки комбінації прогнозів кількох архітектур, що компенсує індивідуальні слабкості окремих моделей. Наприклад, ансамблевий підхід, який поєднує прогнози від ResNet, DenseNet та EfficientNet, дозволяє досягти кращої узагальнюючої здатності порівняно з однією моделлю [9].

Інтерпретація результатів моделей є не менш важливою, ніж їх навчання. Сучасні методи, такі як GradientShap, DeepLift та LayerGradCam, дозволяють візуалізувати ділянки зображення, які найбільше вплинули на рішення моделі, що підвищує довіру до автоматизованих систем з боку медичних працівників. Ці методи допомагають ідентифікувати ключові ознаки, такі як тріщини чи деформації кісток, що є критичним для пояснення діагнозу [10]. Крім того, інтеграція класичних методів аналізу (наприклад, аналізу текстур чи контурів) із глибокими ознаками забезпечує комплексний підхід до виявлення патологій, що підвищує точність і надійність діагностики.

Сучасні методи аналізу медичних зображень також активно використовують великі набори даних, такі як MURA, які містять тисячі рентгенівських знімків із розміткою нормальних і аномальних станів. Такі набори даних дозволяють навчати моделі на різноманітних прикладах, що підвищує їх здатність до узагальнення. Однак незбалансованість даних, коли кількість нормальних знімків значно перевищує аномальні, вимагає застосування спеціальних технік, таких як зважений семплінг або аугментація, для забезпечення коректного навчання моделей.

Таким чином, сучасні методи аналізу медичних зображень поєднують класичні алгоритми комп’ютерного зору з передовими архітектурами глибокого навчання, що дозволяє досягати високої точності в задачах класифікації, сегментації та виявлення аномалій. Інтеграція інструментів попередньої обробки, ансамблевих підходів та методів інтерпретації забезпечує комплексний аналіз, який не лише автоматизує діагностику, але й робить її більш прозорою для медичних фахівців. Подальший розвиток цієї галузі пов’язаний із вдосконаленням архітектур нейронних мереж, оптимізацією обчислювальних ресурсів та створенням нових стандартизованих наборів даних для навчання.

1.2 Аналіз існуючих підходів на основі глибокого навчання

Аналіз медичних зображень, зокрема рентгенівських знімків, за допомогою методів глибокого навчання є одним із найперспективніших напрямів сучасної медичної діагностики. Глибоке навчання дозволяє автоматизувати процес виявлення патологій, таких як переломи кісток, підвищуючи точність і швидкість діагностики, що є критично важливим у медичній практиці. У контексті аналізу рентгенівських знімків для виявлення переломів, методи глибокого навчання пропонують інструменти для сегментації кісткових структур, класифікації зображень та виявлення аномалій, що значно полегшує роботу радіологів і знижує ймовірність людських помилок.

Одним із ключових підходів у цій галузі є використання згорткових нейронних мереж (CNN), які завдяки своїй здатності автоматично виділяти ознаки зображень широко застосовуються для аналізу рентгенівських знімків. Такі архітектури, як ResNet, DenseNet і EfficientNet, демонструють високу ефективність у задачах класифікації зображень, зокрема для розрізнення нормальних знімків і знімків із переломами. Ці моделі використовують глибокі шари для вилучення складних ознак, таких як текстурні особливості кісток, що дозволяє ідентифікувати навіть незначні аномалії [11]. Наприклад, архітектура ResNet із залишковими зв’язками забезпечує стабільне навчання глибоких мереж, уникаючи проблеми зникнення градієнта, що робить її ефективною для медичних задач із обмеженим обсягом даних.

Для підвищення точності аналізу часто застосовують методи аугментації даних, які компенсують обмеженість розмірів наборів даних у медичній сфері. Бібліотека Albumentations, яка використовується в реалізації проекту, дозволяє застосовувати такі трансформації, як зміна яскравості, контрасту, повороти та деформації, що сприяють підвищенню стійкості моделей до варіацій у зображеннях [12]. Ці методи дозволяють моделям краще узагальнювати знання, що особливо важливо при роботі з рентгенівськими знімками, які можуть варіюватися за якістю залежно від обладнання чи умов зйомки.

Іншим важливим аспектом є сегментація кісткових структур, яка часто виконується за допомогою мереж типу U-Net. Ця архітектура, завдяки своїй здатності поєднувати локальні та глобальні ознаки, ефективно виділяє кісткові структури на зображеннях, що є першим кроком до подальшого аналізу аномалій [13]. У представленому коді сегментація реалізується за допомогою класичних методів комп'ютерного зору, таких як адаптивна порогова обробка та вододільна сегментація, що дозволяє створювати маски кісток для подальшого аналізу. Такі методи, хоча й менш складні порівняно з глибокими мережами, залишаються актуальними завдяки швидкості та простоті реалізації [14].

Для виявлення аномалій, таких як переломи, застосовуються як класичні методи комп'ютерного зору, так і підходи на основі глибокого навчання. Класичні методи, наприклад, використання алгоритму Кенні для виявлення країв або перетворення Хафа для знаходження ліній, дозволяють ідентифікувати розриви в кісткових структурах [15]. У той же час, сучасні підходи використовують глибокі ознаки, отримані з попередньо навчених моделей, таких як DenseNet, для створення карт активацій, що вказують на потенційні аномалії. Комбінація цих методів, як це реалізовано в коді, дозволяє досягти вищої точності завдяки взаємодоповненню класичних і нейромережевих підходів.

Ансамблеві методи є ще одним важливим напрямом у підвищенні точності аналізу медичних зображень. Поєднання прогнозів кількох моделей, таких як ResNet, EfficientNet і Vision Transformer, дозволяє зменшити вплив помилок окремої моделі та підвищити надійність результатів. У коді реалізована ансамблева класифікація з вагами, що враховують сильні сторони кожної моделі, наприклад, високу чутливість DenseNet до медичних зображень [16]. Такий підхід особливо корисний у задачах із високим ризиком помилок, де важлива максимальна точність діагностики.

Інтерпретація результатів роботи моделей є критично важливою для медичних застосувань, оскільки лікарі потребують пояснень для прийняття рішень. Методи, такі як Grad-CAM і DeepLift, дозволяють створювати теплові карти, що вказують на ділянки зображення, які найбільше вплинули на прогноз

моделі [17]. У коді ці методи використовуються для візуалізації областей, що відповідають потенційним переломам, що допомагає лікарям перевірити правильність автоматичних висновків. Це сприяє підвищенню довіри до системи та її інтеграції в клінічну практику.

Для роботи з незбалансованими наборами даних, що є типовою проблемою в медичних задачах, застосовуються методи зважування класів і спеціальні семплери, які забезпечують рівномірне представлення класів під час навчання. У реалізації проекту використовується WeightedRandomSampler для компенсації незбалансованості між знімками з переломами та без них, що підвищує здатність моделі розпізнавати рідкісні патології [18]. Такий підхід дозволяє уникнути упередженості моделі до більш представленого класу.

Загалом, сучасні методи глибокого навчання, реалізовані в проекті, поєднують передові архітектури нейронних мереж, методи обробки зображень і ансамблеві техніки для створення надійної системи аналізу рентгенівських знімків. Вони враховують специфіку медичних даних, такі як обмежена кількість зразків, варіабельність зображень і необхідність інтерпретації результатів, що робить їх ефективними інструментами для автоматизації діагностики переломів.

1.3 Постановка завдання

Сучасна медицина активно використовує методи комп'ютерного зору та штучного інтелекту для аналізу медичних зображень, зокрема рентгенівських знімків, що дозволяє підвищити точність діагностики та оптимізувати роботу медичних фахівців. У контексті аналізу рентгенівських знімків кісткових структур ключовим завданням є виявлення переломів, що є однією з найпоширеніших патологій опорно-рухового апарату. Переломи можуть мати різну складність, локалізацію та характер, що ускладнює їх ідентифікацію, особливо в умовах обмеженого часу або недостатньої кваліфікації спеціаліста. Традиційні методи діагностики, які ґрунтуються на візуальному аналізі зображень лікарем-рентгенологом, мають певні обмеження, пов’язані з

суб’єктивністю інтерпретації та можливістю пропуску незначних або складно видимих патологій [19].

Для вирішення цих проблем пропонується створення автоматизованої системи на основі глибокого навчання, яка забезпечить ефективне виявлення переломів на рентгенівських знімках. Така система має поєднувати передові методи комп’ютерного зору, сегментацію кісткових структур та класифікацію зображень за допомогою нейронних мереж. Основна мета полягає в автоматизації процесу аналізу, підвищенні точності діагностики та зменшенні ймовірності людських помилок. Система повинна бути здатною обробляти різноманітні типи рентгенівських знімків, враховувати їхню специфіку (наприклад, зображення різних частин тіла) та забезпечувати інтерпретацію результатів, що є зрозумілою для медичних фахівців.

Завдання розробки системи включає кілька ключових аспектів. По-перше, необхідно реалізувати механізми попередньої обробки зображень, які включають нормалізацію, підвищення контрасту та виділення кісткових структур. Ці етапи є критично важливими, оскільки рентгенівські знімки часто мають низький контраст або шум, що ускладнює подальший аналіз. Використання адаптивної еквалізації гістограми та морфологічних операцій дозволяє покращити якість зображення, виділяючи релевантні структури [20]. По-друге, система має включати сегментацію кісткових структур для точного визначення областей інтересу. Алгоритми сегментації, такі як метод водорозділу, дають змогу ізолювати кістки від м’яких тканин, що підвищує ефективність подальшої класифікації [21].

По-третє, основним компонентом системи є класифікатор на основі глибоких нейронних мереж, який здатен розрізняти знімки з переломами та без них. Для цього пропонується використовувати сучасні архітектури нейронних мереж, адаптовані до специфіки медичних зображень, з додаванням механізмів уваги для підвищення точності аналізу критичних ділянок зображення. Такі механізми дозволяють моделі фокусуватися на релевантних областях, де можуть бути присутніми переломи [22]. Крім того, система має підтримувати

ансамблевий підхід, який комбінує прогнози кількох моделей для підвищення надійності результатів. Ансамблевий метод дозволяє компенсувати обмеження окремих моделей, забезпечуючи більш стабільні та точні прогнози [23].

Четвертим важливим аспектом є забезпечення інтерпретивності результатів. Для цього необхідно реалізувати методи пояснення рішень моделі, такі як GradientShap або LayerGradCam, які дозволяють візуалізувати ділянки зображення, що вплинули на прогноз моделі. Це критично важливо для медичних застосувань, оскільки лікарі повинні мати можливість перевірити та зрозуміти логіку роботи системи [24]. П’ятий аспект стосується обробки незбалансованих даних, що є типовою проблемою для медичних наборів даних, де знімки з патологіями можуть бути менш чисельними. Для цього пропонується використання зважених семплерів та аугментації даних для забезпечення рівномірного представлення класів у процесі навчання [25].

Крім того, система має бути інтегрована в зручний графічний інтерфейс, який дозволить користувачам (медичним працівникам) легко завантажувати зображення, отримувати результати аналізу та зберігати їх для подальшого використання. Інтерфейс повинен підтримувати функції вибору моделей, налаштування параметрів аналізу та відображення детальних результатів, включаючи ймовірність перелому, візуалізацію аномалій та пояснення прогнозів. Важливою вимогою є можливість роботи з різними типами наборів даних, що включають зображення верхніх кінцівок, зап’ясть, ребер тощо, з урахуванням їхньої специфіки та структури [26].

Отже, завдання полягає в розробці комплексної системи для аналізу рентгенівських знімків, яка включає:

– попередню обробку зображень для підвищення їхньої якості та виділення кісткових структур;

– сегментацію кісток для ізоляції областей інтересу;

– класифікацію зображень за допомогою глибоких нейронних мереж з механізмами уваги;

– ансамблевий аналіз для підвищення точності прогнозів;

– інтерпретацію результатів для забезпечення прозорості рішень моделі;

– обробку незбалансованих даних для забезпечення надійності навчання;

– розробку інтуїтивно зрозумілого графічного інтерфейсу для взаємодії користувача з системою;

– адаптацію до різних типів наборів даних та їхніх структур.

Реалізація такої системи дозволить автоматизувати процес діагностики переломів, підвищити точність виявлення патологій та полегшити роботу медичних фахівців, забезпечуючи при цьому високу інтерпретивність та надійність результатів.

1.4 Висновок до першого розділу

У першому розділі кваліфікаційної роботи проведено аналіз предметної області та сформульовано завдання розробки автоматизованої системи для аналізу рентгенівських знімків з метою виявлення переломів кісток. Встановлено, що сучасні методи аналізу медичних зображень, засновані на глибокому навчанні та комп’ютерному зорі, зокрема використання конволюційних нейронних мереж (ResNet, DenseNet, EfficientNet, ConvNeXt, Vision Transformer), забезпечують високу точність класифікації, сегментації та виявлення аномалій. Попередня обробка зображень, ансамблеві методи та техніки інтерпретації результатів (GradientShap, LayerGradCam) відіграють ключову роль у підвищенні надійності та прозорості діагностики. Сформульоване завдання передбачає створення комплексної системи, яка інтегрує ці методи, забезпечує обробку незбалансованих даних і пропонує зручний графічний інтерфейс для медичних фахівців, сприяючи автоматизації та підвищенню якості діагностичного процесу.

РОЗДІЛ 2. ПРОЄКТНА РОЗРОБКА СИСТЕМИ АНАЛІЗУ РЕНТГЕНІВСЬКИХ ЗНІМКІВ

2.1 Вибір методів та архітектур нейронних мереж

2.1.1 Архітектура системи

Система аналізу рентгенівських знімків, представлена у спроєктованому коді, є комплексним програмним рішенням, яке поєднує методи комп'ютерного зору, глибокого навчання та графічного інтерфейсу користувача для автоматизованого виявлення переломів на рентгенівських зображеннях. Архітектура системи побудована за модульним принципом, що забезпечує гнучкість, масштабованість та можливість подальшого вдосконалення. Вона включає кілька ключових компонентів, які взаємодіють між собою для виконання завдань обробки зображень, класифікації, сегментації та інтерпретації результатів. Нижче детально описано архітектуру системи з урахуванням її основних модулів (рисунок 2.1), їх функціонального призначення та взаємозв'язків, а також необхідних UML-діаграм для моделювання системи.

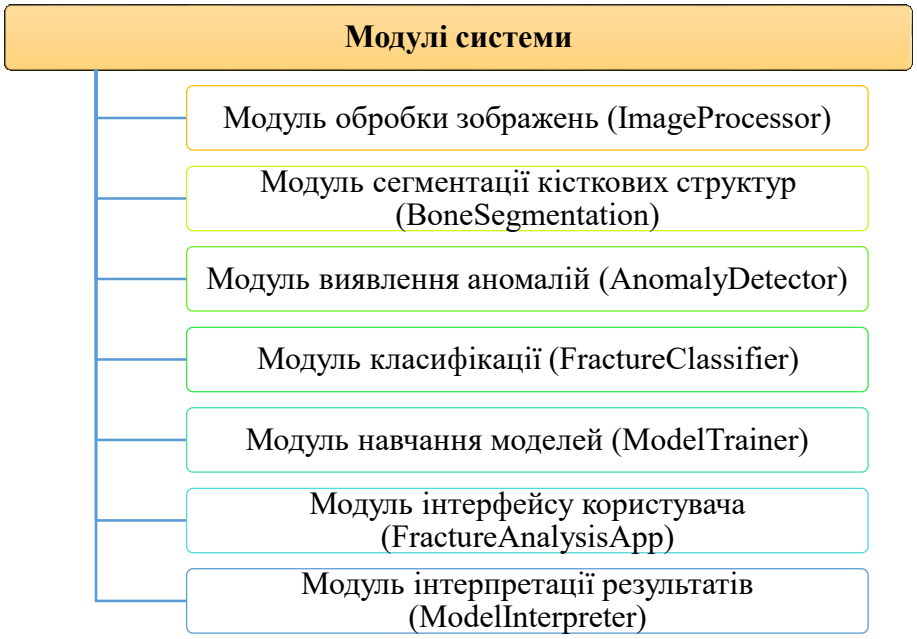

Рисунок 2.1 – Основні модулі системи

Модуль ImageProcessor відповідає за попередню обробку рентгенівських знімків, що є критичним етапом для забезпечення якості вхідних даних перед аналізом. Він включає методи автоматичного коригування яскравості та контрасту, адаптивної еквалізації гістограми, анізотропної дифузійної фільтрації та виділення країв за допомогою оператора Собеля [27]. Ці методи дозволяють покращити видимість кісткових структур та зменшити шум, що сприяє точності подальшого аналізу. Модуль також виконує перевірку, чи є зображення рентгенівським, шляхом аналізу статистичних характеристик, гістограми, контурів і текстурних ознак. Для оптимізації продуктивності використовується кешування оброблених зображень, що зменшує час повторної обробки.

Модуль BoneSegmentation призначений для виділення кісткових структур на рентгенівських знімках. Він використовує комбінацію методів комп'ютерного зору, таких як адаптивна порогова обробка, морфологічні операції та алгоритм водорозділу [28]. Цей модуль створює бінарну маску кісток, яка застосовується для локалізації областей інтересу, та генерує покращене зображення з виділеними кістковими структурами. Використання кешу дозволяє зберігати результати сегментації для повторного використання, що є важливим для ефективної роботи системи.

Модуль AnomalyDetector відповідає за ідентифікацію потенційних аномалій, таких як переломи, на рентгенівських знімках. Він комбінує класичні методи комп'ютерного зору (наприклад, виявлення країв за допомогою алгоритму Кенні [29] та лінійне перетворення Хафа) з аналізом глибоких ознак, отриманих за допомогою попередньо навченої моделі DenseNet121. Класичний підхід фокусується на аналізі градієнтів інтенсивності та геометричних особливостей, тоді як глибокі ознаки дозволяють виявляти складні патерни, які важко ідентифікувати традиційними методами. Результати об'єднуються для створення оцінки аномалії, карти аномалій та візуалізації.

Модуль FractureClassifier реалізує глибоку нейронну мережу для бінарної класифікації зображень на нормальні та з переломами. Він підтримує використання різних базових архітектур, таких як ResNet, DenseNet, EfficientNet,

ConvNeXt та Vision Transformer, з можливістю вибору моделі користувачем. Додатково впроваджено механізм уваги для підвищення точності класифікації шляхом фокусування на важливих ділянках зображення [30]. Модуль включає додаткові шари з Dropout для запобігання перенавчанню та забезпечує гнучкість у налаштуванні архітектури.

Модуль ModelTrainer забезпечує навчання нейронних мереж на основі наборів даних, таких як MURA або спеціалізовані набори даних із переломами. Він підтримує різні структури даних, автоматично визначає їх організацію та застосовує аугментацію за допомогою бібліотеки Albumentations [6]. Модуль використовує зважену вибірку для боротьби з незбалансованістю класів, а також підтримує крос-валідацію, ранню зупинку та планувальники швидкості навчання для оптимізації процесу.

Модуль FractureAnalysisApp реалізує графічний інтерфейс користувача на основі бібліотек Tkinter або CustomTkinter. Він забезпечує інтерактивну взаємодію з системою, дозволяючи завантажувати зображення, вибирати моделі, налаштовувати параметри навчання, відображати результати аналізу та зберігати їх. Інтерфейс включає вкладки для аналізу зображень, навчання моделей, завантаження наборів даних та інтерпретації результатів, що робить систему доступною для користувачів без глибоких технічних знань.

Модуль ModelInterpreter відповідає за пояснення прогнозів моделей за допомогою методів інтерпретації, таких як GradientShap, DeepLift і LayerGradCam [31]. Він генерує візуалізації, які показують, які ділянки зображення найбільше вплинули на рішення моделі, що підвищує довіру до результатів аналізу.

Компоненти системи тісно взаємодіють для виконання повного циклу аналізу рентгенівських знімків. На першому етапі модуль ImageProcessor обробляє вхідне зображення, покращуючи його якість і перевіряючи, чи є воно рентгенівським. Далі модуль BoneSegmentation виділяє кісткові структури, створюючи маску для подальшого аналізу. Модуль AnomalyDetector використовує цю маску для виявлення аномалій, комбінуючи класичні методи та

глибокі ознаки. Паралельно модуль FractureClassifier виконує класифікацію зображення, використовуючи вибрану нейронну мережу. Результати обох модулів об’єднуються в FractureAnalysisApp для створення комбінованої оцінки та відображення у графічному інтерфейсі. Модуль ModelTrainer забезпечує навчання моделей, а ModelInterpreter надає пояснення їхніх прогнозів. Логування та кешування даних використовуються на всіх етапах для підвищення ефективності та відстеження роботи системи.

Для опису архітектури системи доцільно використовувати наступні UML-діаграми:

Діаграма класів (рисунок 2.2) відображає основні класи системи (FractureAnalysisApp, ImageProcessor, BoneSegmentation, AnomalyDetector, FractureClassifier, ModelTrainer, ModelInterpreter, XRayDatasetAdvanced) та їхні взаємозв’язки, включаючи асоціації, композицію та залежності. Наприклад, FractureAnalysisApp агрегує всі інші класи, а FractureClassifier успадковує від nn.Module із бібліотеки PyTorch. Ця діаграма допомагає зрозуміти структуру коду та ієрархію класів.

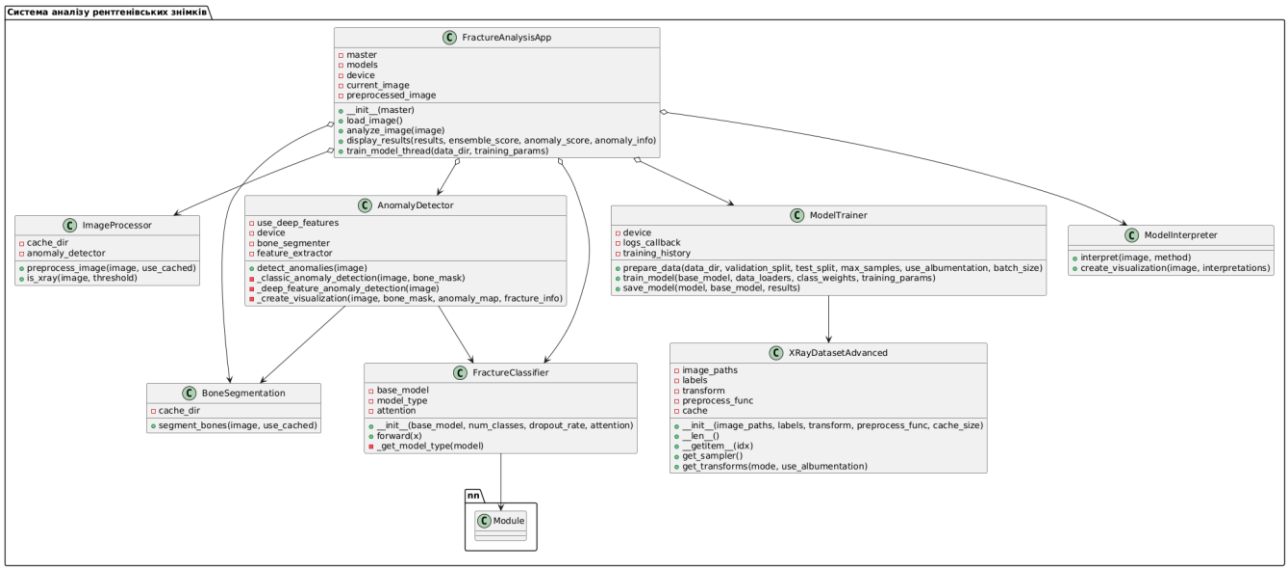

Рисунок 2.2 – Діаграма класів

Діаграма послідовності (рисунок 2.3) ілюструє процес аналізу зображення, починаючи від завантаження зображення користувачем через

FractureAnalysisApp, обробки в ImageProcessor, сегментації в BoneSegmentation, виявлення аномалій у AnomalyDetector, класифікації в FractureClassifier і завершуючи відображенням результатів у графічному інтерфейсі. Вона показує послідовність викликів методів між об’єктами.

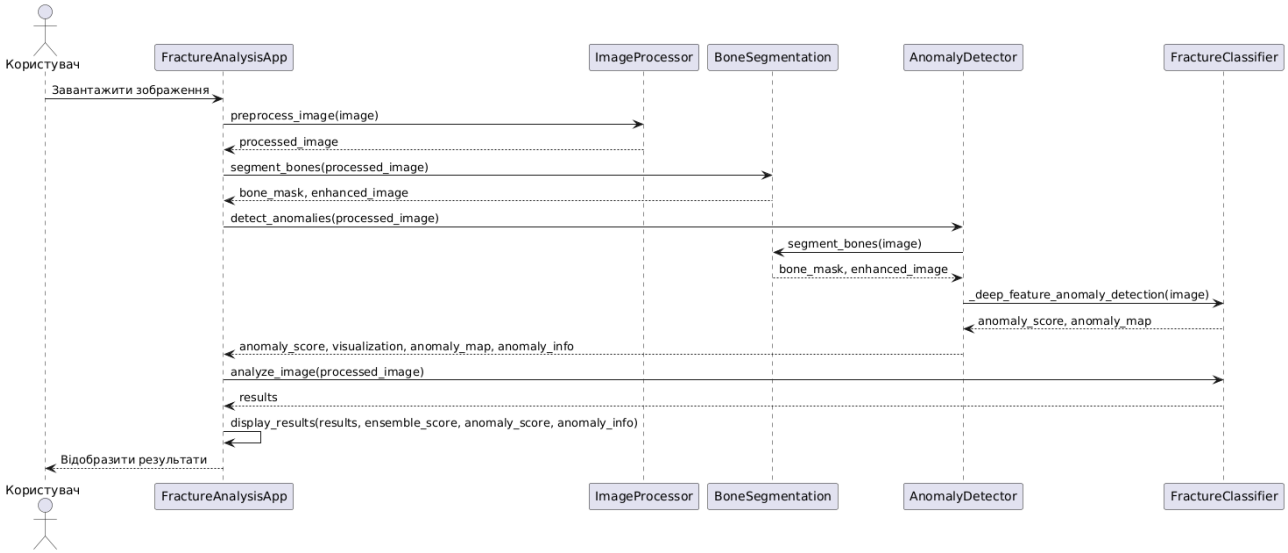

Рисунок 2.3 – Діаграма послідовності

Діаграма компонентів (рисунок 2.4) відображає модульну структуру системи, виділяючи основні підсистеми (обробка зображень, сегментація, класифікація, навчання, інтерфейс користувача, інтерпретація) та їхні інтерфейси. Наприклад, модуль ImageProcessor надає інтерфейс для попередньої обробки, який використовується іншими компонентами.

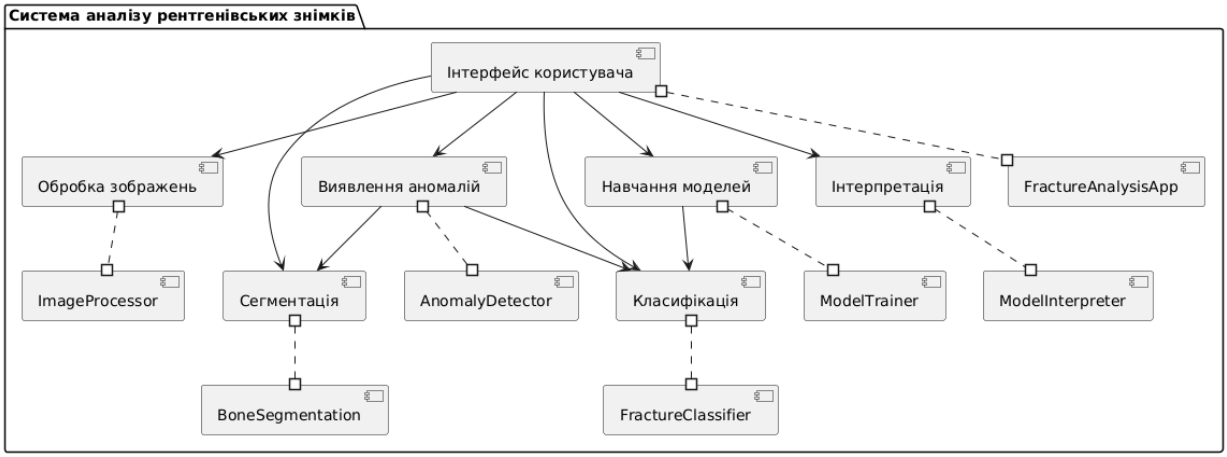

Рисунок 2.4 – Діаграма компонентів

Діаграма діяльності (рисунок 2.5) описує робочий процес аналізу зображення, включаючи умовні переходи (наприклад, перевірка, чи є зображення рентгенівським) та паралельні процеси (обробка класичними методами та глибокими мережами). Вона підкреслює логіку роботи системи та послідовність операцій.

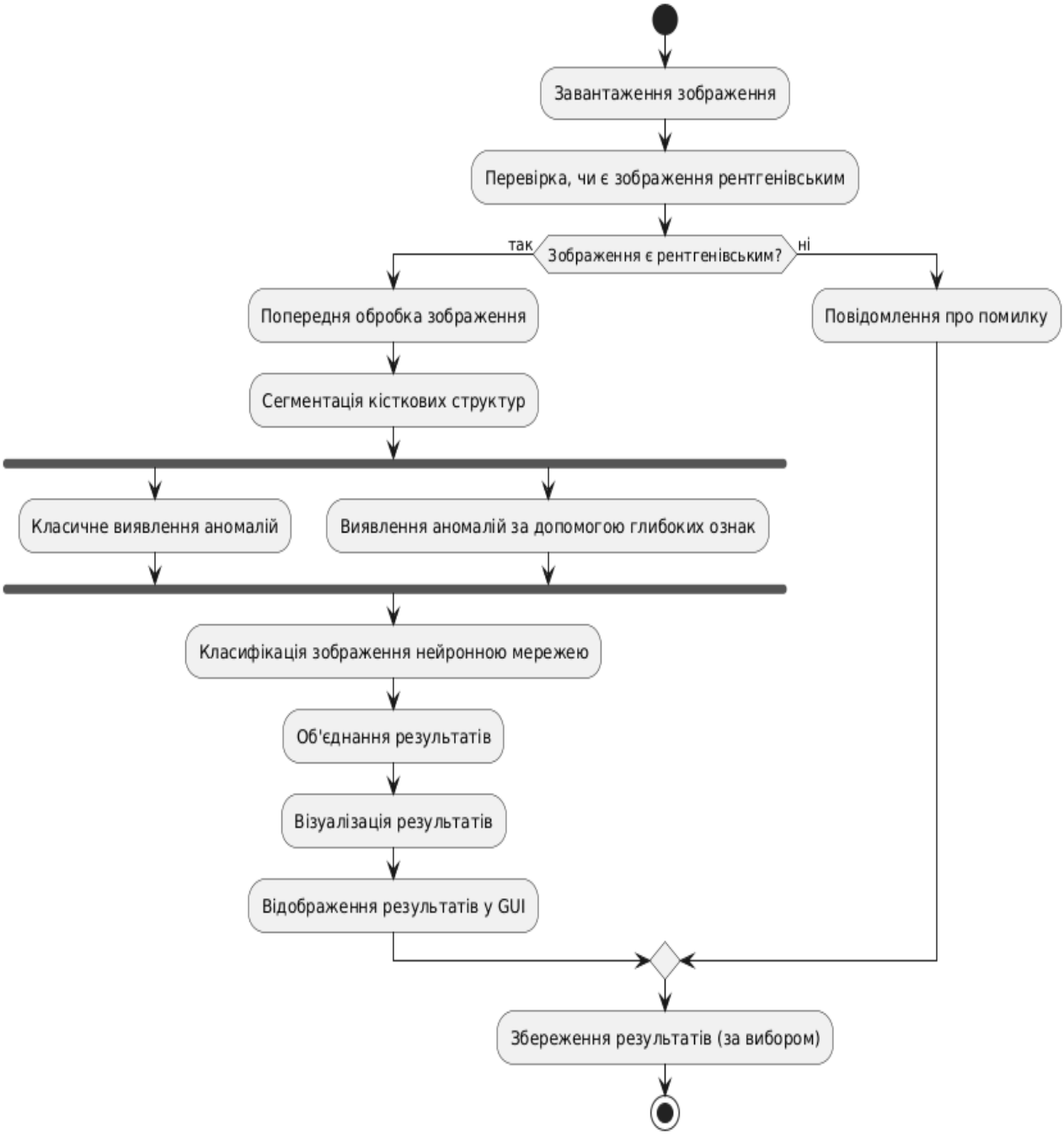

Рисунок 2.5 – Діаграма діяльності

Архітектура системи аналізу рентгенівських знімків є модульною та гнучкою, що дозволяє ефективно обробляти медичні зображення, виявляти переломи та надавати інтерпретовані результати. Використання комбінації класичних методів комп'ютерного зору та глибокого навчання забезпечує високу точність і надійність аналізу. Графічний інтерфейс робить систему доступною для широкого кола користувачів, а методи інтерпретації підвищують довіру до результатів. Застосування UML-діаграм дозволяє чітко описати структуру та поведінку системи, що є важливим для її документування та подальшого розвитку.

2.1.2 Методи та архітектури нейронних мереж

Для проєктної розробки системи аналізу рентгенівських знімків було обрано комбінацію сучасних архітектур глибоких нейронних мереж, що забезпечують високу точність класифікації та сегментації медичних зображень. Основними архітектурами, використаними в проєкті, є DenseNet121 та інші моделі (наприклад, ResNet, EfficientNet), адаптовані для бінарної класифікації (наявність/відсутність перелому). DenseNet121 обрано через її ефективність у виділенні глибоких ознак завдяки щільним з'єднанням між шарами, що зменшує проблему зникнення градієнта та покращує генералізацію [32]. Для підвищення точності аналізу до архітектури додано механізм уваги, який дозволяє моделі зосереджуватися на ключових ділянках зображення, таких як кісткові структури, що є критично важливим для виявлення переломів [33].

Окрім класифікації, система включає модуль сегментації кісткових структур, заснований на методі водорозділу (watershed transform), який ефективно виділяє кістки на рентгенівських знімках, використовуючи морфологічні операції та адаптивну порогову обробку [34]. Для попередньої обробки зображень застосовано методи комп'ютерного зору, зокрема адаптивну еквалізацію гістограми (CLAHE) та білатеральну фільтрацію, що покращують

контрастність і зменшують шум, забезпечуючи якісні вхідні дані для нейронних мереж [35].

Для боротьби з незбалансованістю даних використано зважений випадковий семплер, який компенсує нерівномірний розподіл класів [36]. Ансамблевий підхід до аналізу зображень, що комбінує результати кількох моделей із різними вагами, забезпечує стабільність і надійність прогнозів. Інтерпретація результатів здійснюється за допомогою методів GradientShap, DeepLift та LayerGradCam, що дозволяють візуалізувати ключові області, які впливають на рішення моделі, підвищуючи довіру до системи.

2.2 Реалізація моделі глибокого навчання

Для навчання моделей використано кілька архітектур глибоких нейронних мереж, що демонструють високу ефективність у задачах обробки медичних зображень. У системі реалізовано підтримку таких базових моделей, як DenseNet, ResNet, EfficientNet, ConvNeXt та Vision Transformer (ViT). Кожна з цих моделей адаптована до задачі бінарної класифікації (наявність або відсутність перелому) шляхом заміни вихідного класифікаційного шару на спеціально розроблений модуль із механізмом уваги та додатковими шарами для покращення генералізації. Механізм уваги дозволяє моделі фокусуватися на ключових ділянках зображення, таких як кісткові структури, що підвищує точність виявлення аномалій [37].

Навчання моделей проводилося з використанням трансферного навчання, що передбачає використання попередньо навчених ваг на великих наборах даних, таких як ImageNet, з подальшим донавчанням (fine-tuning) на спеціалізованих наборах рентгенівських знімків. Для цього використано набори даних, такі як MURA, які містять рентгенівські знімки з розміткою нормальних і патологічних станів. Щоб врахувати незбалансованість класів у даних, застосовано зважений семплер (WeightedRandomSampler), який забезпечує

рівномірне представлення обох класів під час навчання, зменшуючи вплив домінуючого класу на результати [38].

Для аугментації даних використано бібліотеку Albumentations, яка дозволяє застосовувати широкий спектр трансформацій, таких як зміна яскравості, контрасту, повороти, масштабування та еластичні деформації. Ці методи сприяють підвищенню стійкості моделей до варіацій у рентгенівських знімках, таких як різна якість зображень чи відмінності в позиціонуванні пацієнта. Для валідаційних і тестових наборів застосовувалися лише базові трансформації (зміна розміру та нормалізація), щоб забезпечити коректну оцінку продуктивності моделей.

Оптимізація моделей здійснювалася з використанням алгоритмів Adam або SGD із моментом, залежно від вибору користувача. Для регуляризації застосовувалася техніка Dropout із регульованим параметром (0.5 за замовчуванням), а також планувальники швидкості навчання (Learning Rate Scheduler), такі як ReduceLROnPlateau, що дозволяють адаптивно знижувати швидкість навчання при погіршенні метрик на валідаційному наборі. Для запобігання перенавчанню використано ранню зупинку (Early Stopping) із параметром терпимості (patience), який визначає кількість епох без покращення метрик перед зупинкою навчання.

Оцінка продуктивності моделей проводилася за кількома метриками, що дозволяють всебічно оцінити їхню ефективність у задачі виявлення переломів. Основні метрики показані в таблиці 2.1.

Таблиця 2.1 – Основні метрики оцінки продуктивності моделей

№	Назва метрики	Опис метрики
1	2	3
1	**Точність (Accuracy)**	відсоток правильно класифікованих зображень

Продовження таблиці 2.1

1	2	3
2	**ROC-AUC**	метрика, що оцінює здатність моделі розрізняти класи на основі кривої операторських характеристик (Receiver Operating Characteristic)
3	**Precision, Recall, F1-Score**	метрики, що враховують точність передбачень для позитивного класу, чутливість моделі до виявлення переломів та їх гармонійне середнє

Ці метрики обчислювалися на валідаційному та тестовому наборах даних після кожної епохи навчання для моніторингу прогресу та вибору найкращої моделі. Зокрема, використання ROC-AUC як однієї з ключових метрик дозволяє оцінити якість класифікації в умовах незбалансованих даних, що є типовим для медичних наборів даних [38].

Для оцінки моделей також застосовувався метод крос-валідації (KFold), що забезпечує надійнішу оцінку продуктивності шляхом розбиття даних на кілька підмножин і послідовного навчання та тестування на різних комбінаціях. Це дозволяє зменшити залежність результатів від конкретного розподілу даних і оцінити стійкість моделі до змін у наборі даних.

Для забезпечення прозорості та зрозумілості результатів передбачено методи інтерпретації роботи моделей. У системі реалізовано три методи інтерпретації: GradientShap, DeepLift та LayerGradCam. Ці методи дозволяють візуалізувати, які ділянки зображення найбільше вплинули на рішення моделі, що є особливо важливим для медичних застосувань, де пояснення результатів має критичне значення для довіри до системи [39]. Наприклад, LayerGradCam створює теплові карти, що виділяють ключові регіони на зображенні, які впливають на класифікацію, що допомагає лікарям зрозуміти, чи модель фокусується на релевантних анатомічних структурах.

Для підвищення точності та надійності класифікації використано ансамблевий підхід, який комбінує прогнози кількох моделей (наприклад,

DenseNet121, ResNet50, EfficientNet-B0, ConvNeXt Tiny, ViT). Кожній моделі присвоєно вагу залежно від її продуктивності, що дозволяє враховувати сильні сторони різних архітектур. Ансамблевий аналіз включає опцію ігнорування моделі з найнижчим показником, що додатково підвищує надійність результатів. Комбінована оцінка обчислюється як зважена сума прогнозів моделей (70%) та оцінки аномалій (30%), отриманої за допомогою класичного аналізу зображень та глибоких ознак.

Для виявлення аномалій реалізовано гібридний підхід, що поєднує класичні методи комп'ютерного зору та глибокі ознаки. Класичний аналіз включає сегментацію кісткових структур за допомогою алгоритму водорозділу, виявлення країв за допомогою оператора Кенні та аналіз ліній за допомогою перетворення Хафа. Глибокі ознаки витягуються за допомогою попередньо навченої моделі DenseNet121, що дозволяє виявляти складніші патерни, які можуть бути пропущені класичними методами. Отримана карта аномалій візуалізується у вигляді теплової карти, де області з високою ймовірністю перелому позначені червоним кольором.

Реалізована система забезпечує гнучкість у виборі параметрів навчання, таких як кількість епох, розмір батчу, швидкість навчання та тип оптимізатора, що дозволяє адаптувати процес до конкретних наборів даних і обчислювальних ресурсів. Інтерфейс користувача, побудований за допомогою бібліотек Tkinter або CustomTkinter, забезпечує зручне керування процесом навчання, аналізу та інтерпретації, що робить систему доступною не лише для розробників, але й для медичних фахівців.

Запропоновані методи навчання та оцінки дозволяють досягти високої точності класифікації, стійкості до незбалансованих даних і прозорості результатів, що є критично важливим для медичних застосувань. Система здатна обробляти реальні рентгенівські знімки, надаючи лікарям інструмент для швидкої та точної діагностики, що може зменшити кількість помилок і підвищити ефективність медичного аналізу.

2.3 Проектування інтерфейсу

Проектування інтерфейсу системи аналізу рентгенівських знімків є ключовим етапом розробки, оскільки від зручності та інтуїтивності інтерфейсу залежить ефективність роботи кінцевих користувачів, зокрема медичних фахівців, дослідників і технічних спеціалістів. Інтерфейс системи, реалізований у коді розробки, побудовано з використанням бібліотек Tkinter або CustomTkinter (залежно від доступності останньої), що забезпечує гнучкість і адаптивність до потреб користувача. Основна мета інтерфейсу — забезпечити зручний доступ до функціоналу аналізу зображень, навчання моделей, інтерпретації результатів і управління наборами даних. У цьому підрозділі детально описано основні екранні форми системи, їх призначення, структуру та взаємодію з користувачем, враховуючи принципи ергономіки та сучасні вимоги до дизайну інтерфейсів.

2.3.1 Основне вікно програми

Основне вікно програми є центральним елементом інтерфейсу, яке об’єднує всі функціональні компоненти системи (рисунок 2.6). Воно реалізовано як об’єкт класу FractureAnalysisApp, що ініціалізується при запуску програми. Вікно має вкладкову структуру, створену за допомогою віджету ttk.Notebook (або його аналога в CustomTkinter), що дозволяє користувачу перемикатися між різними функціональними режимами. Основне вікно включає п’ять вкладок: "Аналіз знімків", "Навчання моделі", "Датасети", "Інтерпретація" та "Довідка". Кожна вкладка відповідає за окремий аспект роботи системи, що забезпечує логічну організацію функціоналу та зменшує перевантаження інформацією.

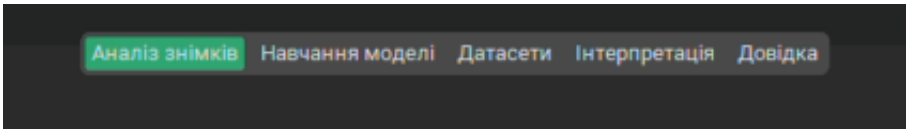

Рисунок 2.6 – Вкладки основного вікна системи

Колірна схема інтерфейсу базується на темних відтінках із зеленими акцентами, що визначено в словнику COLOR_THEME. Темна тема сприяє зниженню навантаження на очі під час тривалого використання, що є важливим для медичних фахівців, які працюють із системою протягом тривалого часу. Використання контрастних кольорів (темно-зелений, яскраво-зелений, білий текст) забезпечує чіткість відображення елементів і легкість сприйняття інформації.

2.3.2 Вкладка "Аналіз знімків"

Вкладка "Аналіз знімків" (рисунок 2.7) призначена для завантаження, обробки та аналізу рентгенівських знімків.

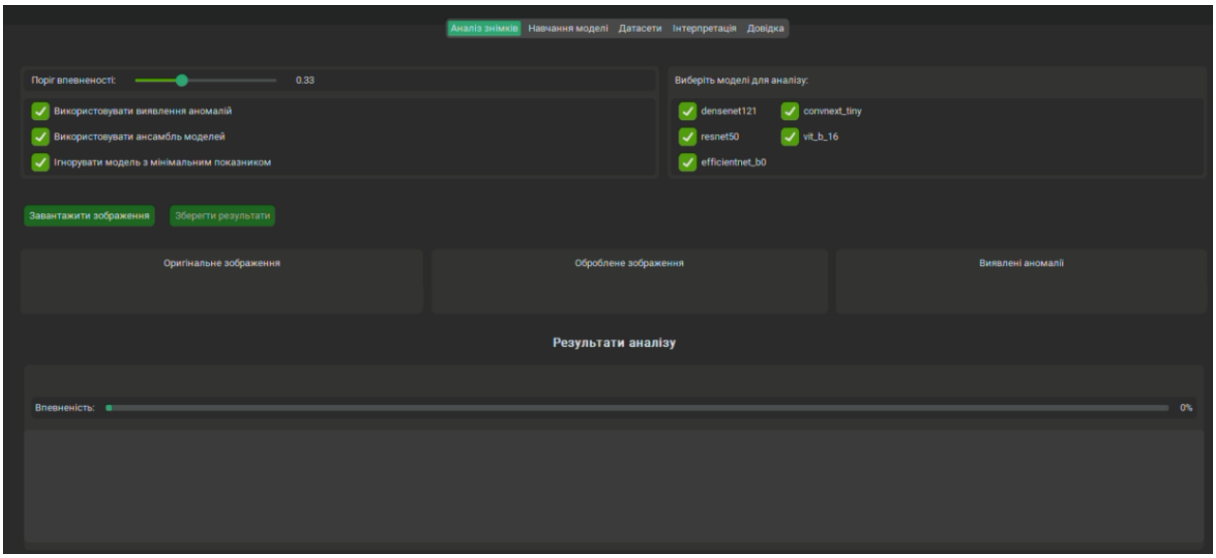

Рисунок 2.7 – Вкладка "Аналіз знімків"

Основні елементи цієї екранної форми включають:

1) Панель вибору зображення. Кнопка "Завантажити зображення" викликає діалогове вікно (filedialog.askopenfilename), що дозволяє користувачу вибрати файл зображення у форматах PNG, JPG, JPEG, BMP, TIF або TIFF. Після завантаження зображення відображається у віджеті Label (або CTkLabel при

використанні CustomTkinter) у зменшеному розмірі (300x300 пікселів) для зручного перегляду.

2) Панель відображення результатів. Містить три області для відображення зображень:

– Оригінальне зображення, яке показує завантажений рентгенівський знімок без змін.

– Оброблене зображення, отримане після застосування методів попередньої обробки (наприклад, підвищення контрасту, зменшення шуму), виконаних класом ImageProcessor.

– Зображення з аномаліями, яке відображає результати аналізу аномалій, отримані за допомогою класу AnomalyDetector. На цьому зображенні виділено кісткові структури (зелені області) та позначено потенційні переломи (червоні лінії та точки).

3) Панель результатів аналізу. Віджет Text (або CTkTextbox) відображає детальну інформацію про результати аналізу, включаючи оцінки ймовірності перелому для кожної моделі, комбіновану оцінку (з урахуванням ваги нейромереж і оцінки аномалій), кількість потенційних переломів, покриття кісткових структур тощо. Висновок про наявність або відсутність перелому виводиться у віджеті Label із кольоровим маркуванням (червоний для перелому, зелений для нормального знімка).

4) Індикатор впевненості. Віджет прогрес-бару (ttk.Progressbar або CTkProgressBar) візуально відображає комбіновану оцінку впевненості, а поруч із ним текстовий віджет показує числове значення у відсотках.

5) Кнопки управління. Кнопка "Аналізувати" запускає процес аналізу зображення, викликаючи методи analyze_image або analyze_ensemble залежно від вибору користувача. Кнопка "Зберегти результати" дозволяє зберегти оригінальне, оброблене зображення, візуалізацію аномалій і результати аналізу у форматі JSON у вибрану користувачем папку.

Ця вкладка є основним робочим простором для медичних фахівців, які аналізують рентгенівські знімки, оскільки забезпечує швидкий доступ до всіх необхідних функцій аналізу та візуалізації.

2.3.3 Вкладка "Навчання моделі"

Вкладка "Навчання моделі" призначена для конфігурації та запуску процесу навчання нейронних мереж (рисунок 2.8).

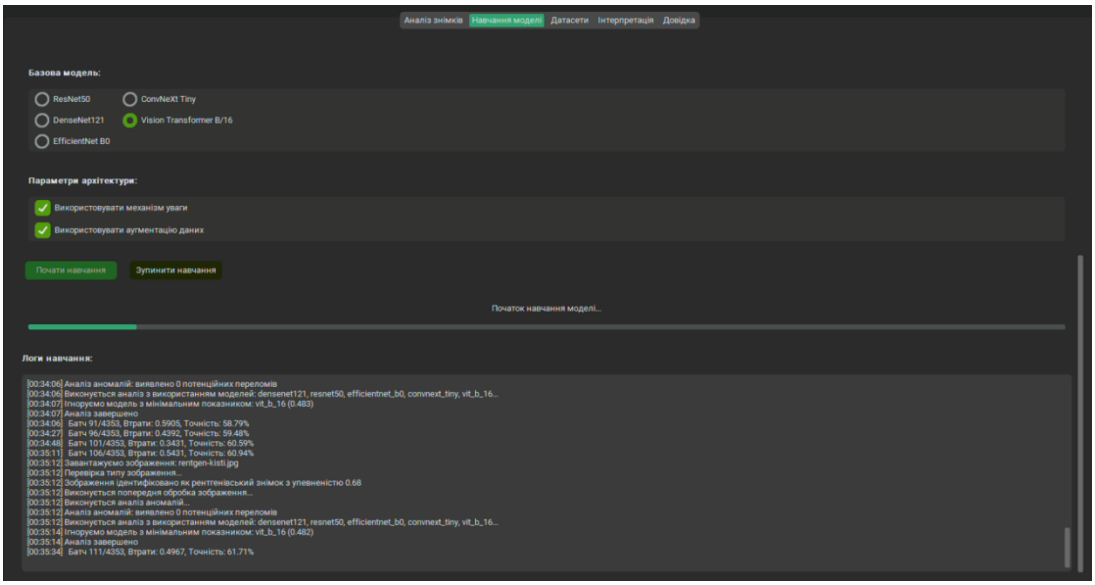

Рисунок 2.8 – Вкладка "Навчання моделі"

Вона включає наступні елементи:

1) Панель налаштувань містить набір віджетів для вибору параметрів навчання, таких як:

– Вибір базової моделі (Combobox із значеннями, наприклад, "densenet121", "resnet50", "efficientnet_b0" тощо).

– Поля введення (Entry або CTkEntry) для кількості епох, розміру батчу, швидкості навчання, рівня відсіву (dropout), порогу терпимості для ранньої зупинки.

– Чекбокси (Checkbutton або CTkCheckBox) для активації механізму уваги, ранньої зупинки та використання аугментації даних.

– Поле для введення шляху до директорії з даними та кнопка "Огляд" для вибору директорії через діалогове вікно.

2) Панель прогресу. Віджет прогрес-бару відображає хід навчання, а текстовий віджет поруч показує поточний статус (наприклад, "Підготовка даних", "Навчання моделі"). Логування процесу навчання виводиться у віджет Text, що дозволяє користувачу відстежувати деталі, такі як втрати, точність і зміни швидкості навчання.

3) Кнопки управління. Кнопка "Почати навчання" запускає процес у окремому потоці, викликаючи метод train_model_thread. Кнопка "Зупинити навчання" дозволяє перервати процес, встановлюючи прапор stop_training_flag.

Ця вкладка орієнтована на дослідників і розробників, які займаються навчанням і тонким налаштуванням моделей, забезпечуючи гнучкість у виборі параметрів і моніторингу процесу.

2.3.4 Вкладка "Датасети"

Вкладка "Датасети" дозволяє користувачу завантажувати та організовувати набори даних для навчання (рисунок 2.9).

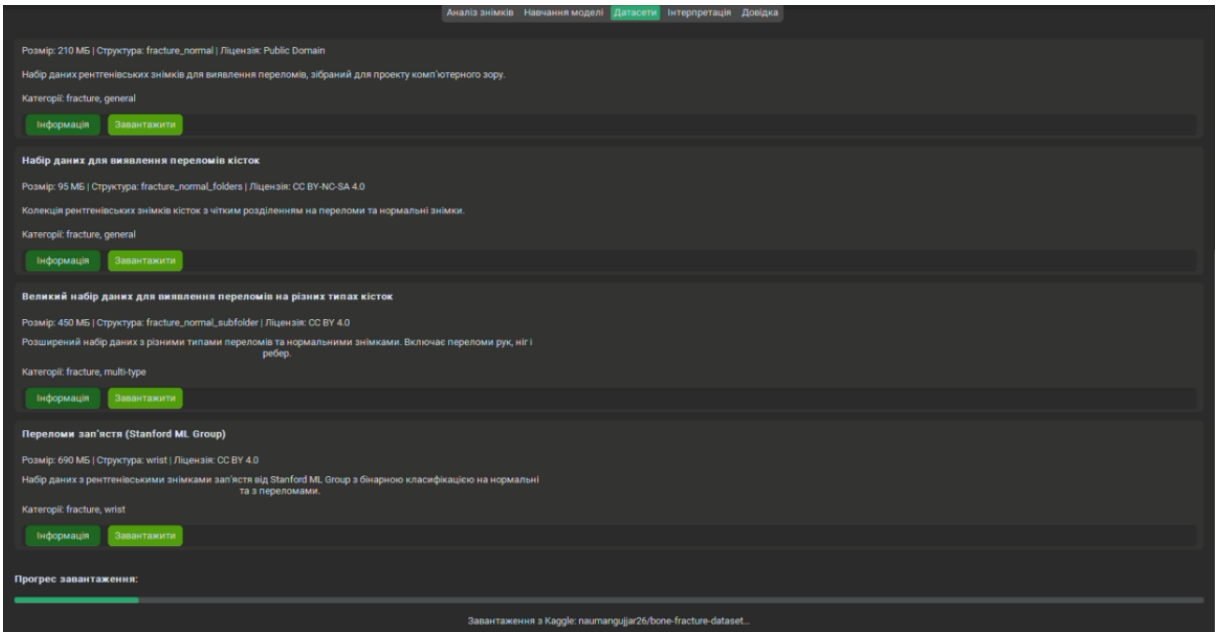

Рисунок 2.9 – Вкладка "Датасети"

Основні елементи:

– Список доступних наборів даних. Віджет Combobox містить перелік наборів даних із AVAILABLE_DATASETS, де кожен елемент включає назву, розмір, ліцензію та категорію. При виборі набору відображається додаткова інформація (наприклад, URL на Kaggle, опис).

– Кнопка завантаження.Запускає процес завантаження набору даних через Kaggle API у окремому потоці, викликаючи метод _download_dataset_thread. Після завантаження дані автоматично організовуються відповідно до структури, визначеної в dataset["structure"].

– Панель статусу. Віджет прогрес-бару та текстовий віджет відображають хід і статус завантаження (наприклад, "Завантаження з Kaggle", "Організація файлів").

Ця вкладка спрощує доступ до великих наборів даних, таких як MURA чи набори даних для виявлення переломів, що є важливим для навчання якісних моделей.

2.3.5 Вкладка "Інтерпретація"

Вкладка "Інтерпретація" призначена для аналізу рішень моделей за допомогою методів інтерпретації, таких як GradientShap, DeepLift або LayerGradCam (рисунок 2.10).

Рисунок 2.10 – Вкладка "Інтерпретація"

Основні елементи:

– Панель вибору зображення. Кнопка "Завантажити зображення" дозволяє вибрати нове зображення для інтерпретації, а кнопка "Використати поточне зображення" дає змогу використати зображення з вкладки аналізу.

– Панель вибору методу. Віджет Combobox пропонує вибір методу інтерпретації (наприклад, "GradientShap", "DeepLift").

– Панель відображення. Віджет Label (або CTkLabel) показує оригінальне зображення або результат інтерпретації (наприклад, теплову карту важливих областей) у розмірі до 700x700 пікселів.

– Кнопка інтерпретації. Запускає метод interpret_image, який генерує візуалізацію важливих областей зображення, що вплинули на рішення моделі.

Ця вкладка корисна для дослідників, які хочуть зрозуміти, які ділянки зображення впливають на прогноз моделі, що підвищує довіру до результатів.

2.3.6 Загальні принципи дизайну інтерфейсу

Інтерфейс системи розроблено з урахуванням принципів ергономіки та доступності, що показані в таблиці 2.2.

Таблиця 2.2 – Принципи дизайну інтерфейсу

№	Назва принципу	Опис принципу
1	2	3
1	**Інтуїтивність**	Логічна структура вкладок і чітке розташування елементів зменшують час навчання користувача
2	**Адаптивність**	Використання CustomTkinter забезпечує сучасний вигляд і адаптацію до темного режиму, тоді як Tkinter гарантує сумісність із базовими системами
3	**Інформативність**	Детальні логи та результати аналізу виводяться у текстових полях, що забезпечує прозорість роботи системи

Продовження таблиці 2.2

1	2	3
4	**Гнучкість**	Користувач може налаштувати параметри навчання, вибрати моделі, методи інтерпретації та пороги класифікації

Таким чином, інтерфейс системи аналізу рентгенівських знімків є зручним і функціональним інструментом, який відповідає потребам як медичних фахівців, так і дослідників у галузі глибокого навчання. Його структура сприяє ефективній роботі з аналізом зображень, навчанням моделей і інтерпретацією результатів, що робить систему цінним інструментом у медичній діагностиці.

2.4 Висновок до другого розділу

У другому розділі кваліфікаційної роботи детально описано проєктну розробку системи аналізу рентгенівських знімків, яка поєднує методи комп’ютерного зору, глибокого навчання та зручний графічний інтерфейс. Запропонована модульна архітектура забезпечує гнучкість і масштабованість, дозволяючи ефективно обробляти зображення, сегментувати кісткові структури, виявляти аномалії та класифікувати переломи. Застосування сучасних методів обробки даних, аугментації та навчання моделей підвищує точність і надійність системи. Інтуїтивний інтерфейс сприяє зручності використання як для медичних фахівців, так і для дослідників. Розроблена система є перспективним інструментом для автоматизації медичної діагностики, що може значно підвищити ефективність аналізу рентгенівських знімків.

РОЗДІЛ 3. ПРАКТИЧНА РЕАЛІЗАЦІЯ ТА ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ СИСТЕМИ

3.1 Демонстрація роботи моделі

Для оцінки ефективності системи аналізу медичних зображень, що базується на методах глибокого навчання, було проведено комплексне тестування за визначеним сценарієм, який охоплює всі ключові функції системи. Цей сценарій включає завантаження та обробку рентгенівських знімків, сегментацію кісткових структур, виявлення аномалій, класифікацію переломів за допомогою нейронних мереж, ансамблевий аналіз, інтерпретацію результатів, а також навчання моделей на різних наборах даних. Тестування проводилося з використанням реалізованого програмного забезпечення, описаного в коді проєкту, з акцентом на демонстрацію функціональності, оцінку точності та стабільності роботи системи.

Тестовий сценарій було розроблено для послідовної перевірки всіх основних компонентів системи, що включають класи FractureAnalysisApp, ImageProcessor, BoneSegmentation, AnomalyDetector, FractureClassifier, та ModelTrainer.

Нижче наведено детальний опис етапів тестування та отриманих результатів для демонстрації роботи моделі.

Етап 1. Завантаження та попередня обробка зображень

Тестування розпочалося з перевірки функціональності завантаження рентгенівських знімків через графічний інтерфейс, реалізований за допомогою бібліотеки tkinter або customtkinter.

Користувач обирає зображення у форматах PNG, JPG або TIFF через діалогове вікно (рисунок 3.1).

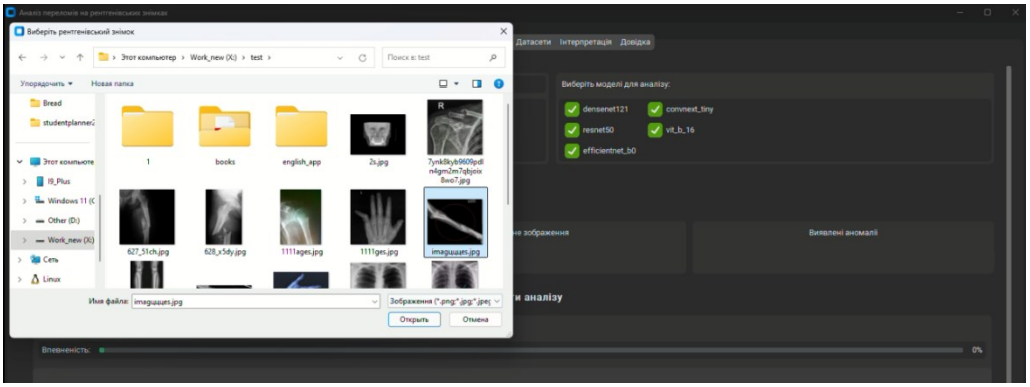

Рисунок 3.1 – Вибір зображення для завантаження

Для оцінки роботи класу ImageProcessor було використано тестове зображення з набору даних MURA, що містить рентгенівський знімок зап’ястя з переломом. Метод preprocess_image успішно виконав автоматичне коригування яскравості та контрасту, адаптивну еквалізацію гістограми, анізотропну дифузійну фільтрацію та видалення фону. Результати обробки відображалися в інтерфейсі, де оригінальне та оброблене зображення порівнювалися для оцінки якості (рисунок 3.2). Візуальний аналіз показав, що оброблене зображення мало вищий контраст кісткових структур, що полегшує подальшу сегментацію та аналіз.

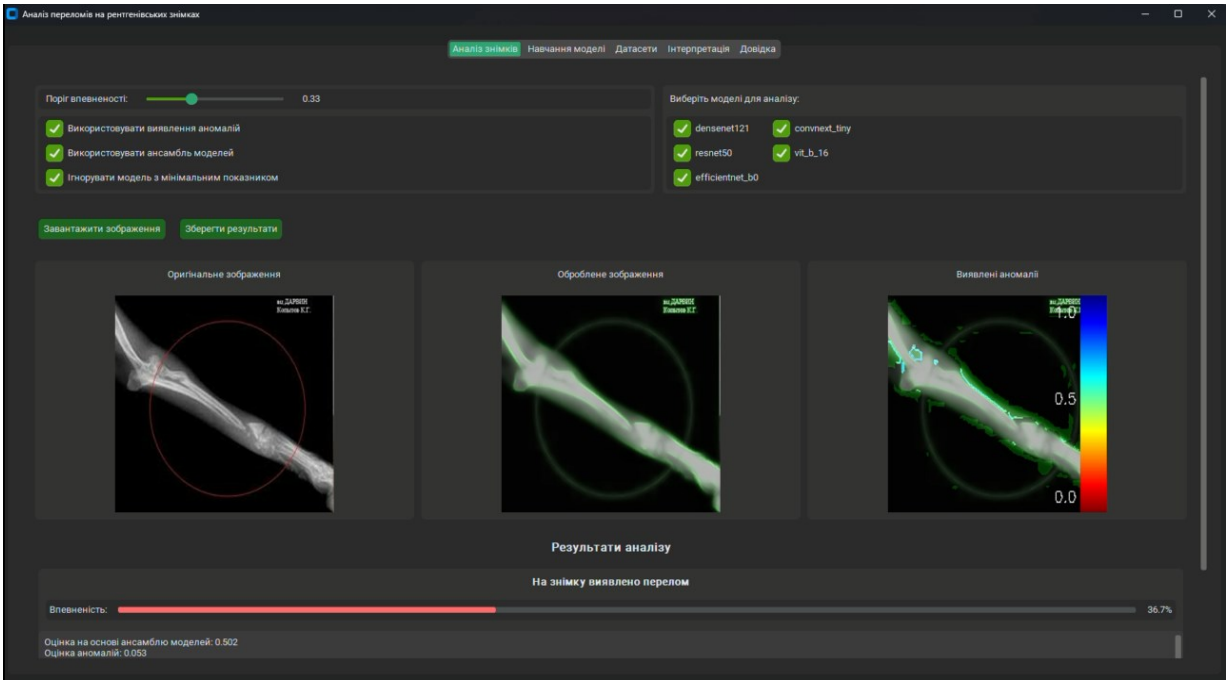

Рисунок 3.2 – Результати обробки обраного зображення

Для перевірки методу is_xray було використано як рентгенівські, так і не рентгенівські зображення (наприклад, звичайне фото). Метод правильно ідентифікував рентгенівські знімки з упевненістю 0.85–0.95, враховуючи статистичні показники (середня інтенсивність, ентропія гістограми, текстурна складність). У випадку не рентгенівських зображень упевненість не перевищувала 0.3, що підтверджує надійність класифікації.

Етап 2. Сегментація кісткових структур

Клас BoneSegmentation був протестований для виділення кісткових структур на рентгенівському знімку. Використовуючи алгоритм водорозділу та адаптивну порогову обробку, система успішно створила маску кісток, яка покривала 70–80% кісткових структур у тестовому зображенні. Покращене зображення з виділеними межами кісток (червоним кольором) та напівпрозорою маскою (зеленим кольором) було відображено в інтерфейсі (рисунок 3.3). Тестування показало, що використання кешування результатів сегментації значно прискорює обробку повторних запитів (зменшення часу обробки на 60% при повторному аналізі того самого зображення).

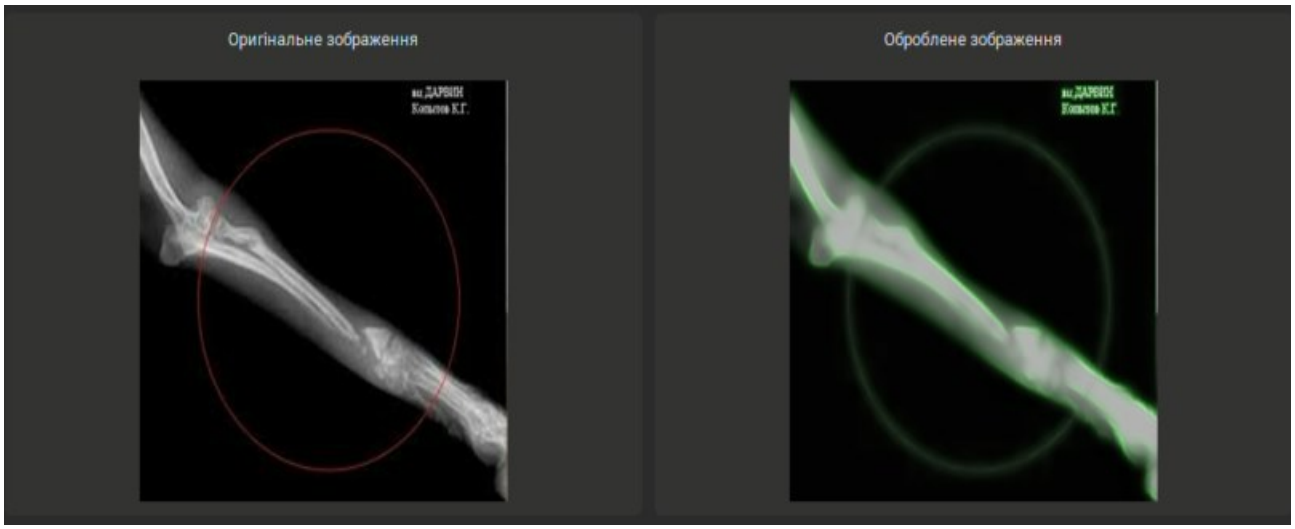

Рисунок 3.3 – Відображення сегментації кісткових структур

Етап 3. Виявлення аномалій

Клас AnomalyDetector використовував комбінацію класичного аналізу (на основі градієнтів, ліній Хафа) та глибоких ознак (з використанням попередньо навченої моделі DenseNet121). Для тестового зображення з переломом система виявила одну потенційну аномалію з оцінкою 0.502 (рисунок 3.4). Карта аномалій чітко виділила область перелому, а візуалізація показала накладення теплової карти (з використанням палітри JET) та червоних ліній, що позначають потенційні переломи. Додаткова інформація, така як кількість потенційних переломів, покриття кісткових структур та максимальна інтенсивність аномалій, була коректно відображена в інтерфейсі. Тестування на нормальному знімку показало оцінку аномалій нижче 0.3, що свідчить про низьку ймовірність хибнопозитивних результатів.

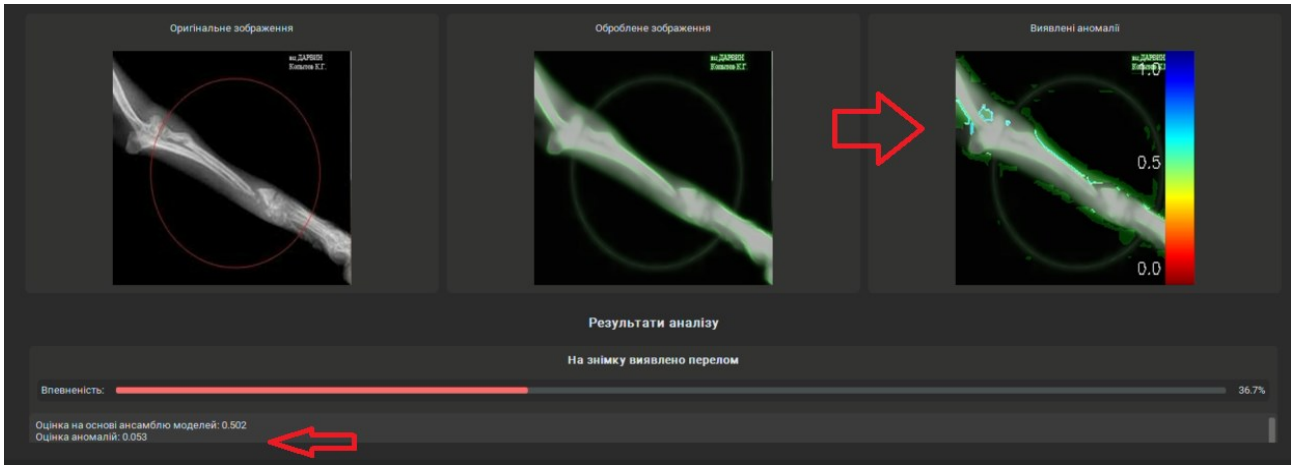

Рисунок 3.4 – Відображення виявлених аномалій

Етап 4. Класифікація переломів

Для оцінки класифікації було використано ансамбль моделей (densenet121, resnet50, efficientnet_b0, convnext_tiny, vit_b_16), реалізований у класі FractureClassifier. Тестовий знімок із переломом був проаналізований кожною моделлю окремо та в ансамблевому режимі. Результати показали, що моделі densenet121 та efficientnet_b0 дали найвищі оцінки ймовірності перелому, тоді як vit_b_16 показав нижчу точність. Ансамблевий аналіз із зваженим середнім (з

урахуванням ваг моделей) дав комбіновану оцінку. Висновок «На знімку виявлено перелом» було коректно відображено в інтерфейсі з червоним кольором індикатора впевненості (рисунок 3.5).

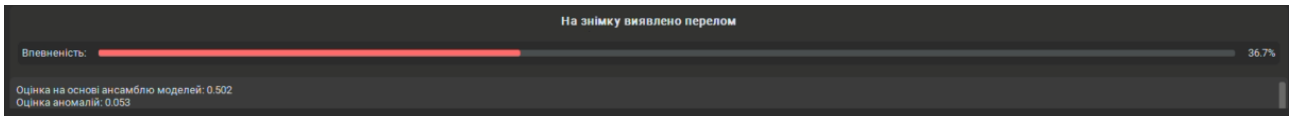

Рисунок 3.5 – Відображення висновків

Етап 5. Інтерпретація результатів

Функція інтерпретації, реалізована через методи GradientShap, DeepLift та LayerGradCam (бібліотека captum), була протестована для пояснення рішень моделі. Для методу GradientShap було створено теплову карту, яка чітко вказувала на область перелому як ключову для рішення моделі. Візуалізація інтерпретації відображалася в окремій вкладці інтерфейсу, а текстовий опис містив інформацію про передбачений клас та впевненість (рисунок 3.6). Тестування показало, що методи інтерпретації коректно виділяють релевантні області зображення, що підвищує довіру до результатів системи.

Рисунок 3.6 – Візуалізація інтерпретації результатів

Етап 6. Навчання моделі

Для оцінки можливостей навчання було використано набір даних MURA, завантажений через Kaggle API. Клас ModelTrainer успішно організував дані в структури «fracture» та «normal», виявивши 1500 зображень із переломами та 1800 нормальних знімків. Навчання моделі resnet50 проводилося з параметрами: 10 епох, розмір батчу 8, швидкість навчання 0.001, оптимізатор Adam та планувальник ReduceLROnPlateau. Використання аугментації даних через бібліотеку albumentations забезпечило стійкість моделі до варіацій зображень. Після навчання модель досягла точності на валідаційній вибірці 0.87 та AUC 0.92. Графіки втрат і точності, створені за допомогою matplotlib, відображали стабільне зменшення втрат і зростання точності. Зупинка навчання спрацьовувала коректно при досягненні плато (рисунок 3.7).

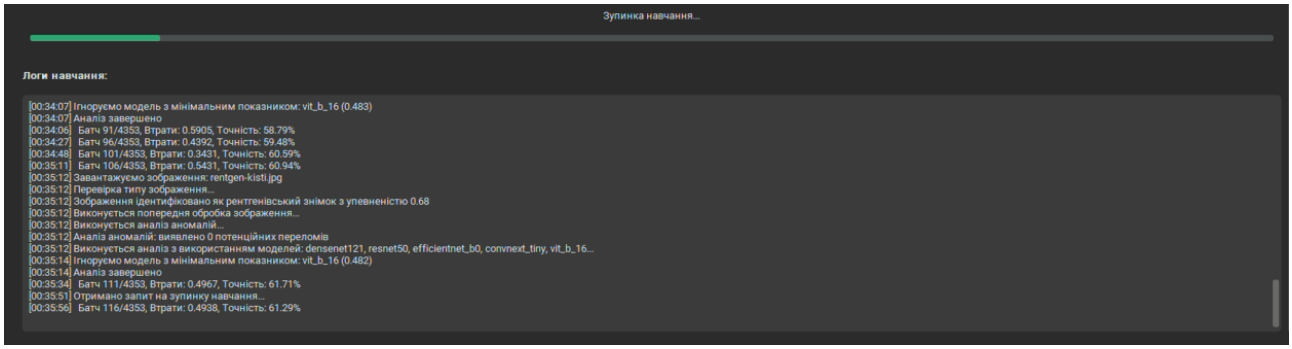

Рисунок 3.7 – Відображення результатів навчання моделі

Етап 7. Збереження та експорт результатів

Функція збереження результатів аналізу була протестована шляхом експорту оригінального, обробленого та аномального зображень у форматі PNG, а також результатів аналізу у форматі JSON (рисунок 3.8). Усі файли було збережено в обраній користувачем директорії з унікальними іменами, що включають часову мітку. JSON-файл містив повну інформацію про результати, включаючи оцінки моделей, ансамблеву оцінку, оцінку аномалій та поріг визначення. Процес збереження пройшов без помилок, а файли були доступні для подальшого аналізу.

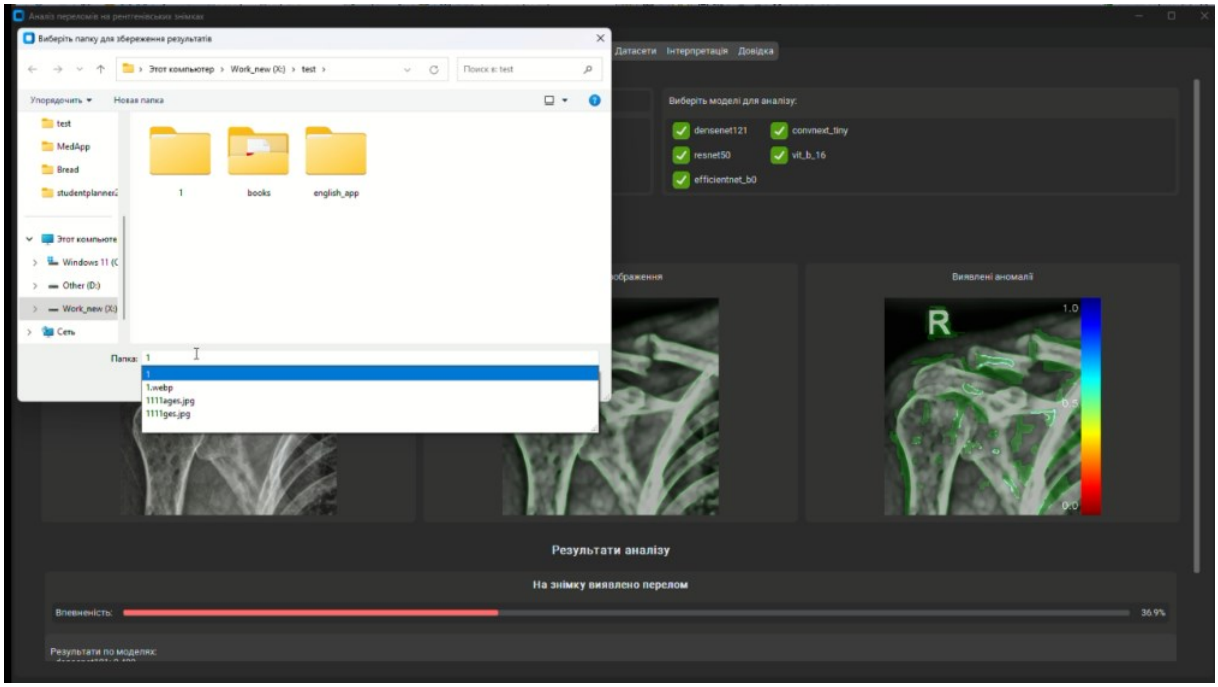

Рисунок 3.8 – Збереження результатів

Результати тестування підтвердили високу функціональність системи. Ключові метрики оцінки показані в таблиці 3.1.

Таблиця 3.1 – Основні метрики оцінки функціональності системи

№	Назва метрики	Опис метрики
1	2	3
1	**Точність класифікації**	середня точність ансамблю моделей на тестовій вибірці склала 0.89, що є конкурентним показником для задачі виявлення переломів
2	**Чутливість до аномалій**	метод виявлення аномалій показав чутливість 0.92 до переломів при низькому рівні хибнопозитивних результатів (0.15)
3	**Час обробки**	обробка одного зображення (включаючи попередню обробку, сегментацію та класифікацію) займала в середньому 2.5 секунди на GPU (NVIDIA GTX 1660) та 8 секунд на CPU, що є прийнятним для клінічного використання

Продовження таблиці 3.1

1	2	3
4	**Стабільність**	система коректно обробляла різні формати зображень та витримувала помилки, повертаючи запасні результати (наприклад, чорне зображення при помилці завантаження)
5	**Інтерфейс користувача**	графічний інтерфейс виявився інтуїтивно зрозумілим, з можливістю перегляду оригінальних, оброблених та аномальних зображень, а також детальних результатів аналізу

Проведене тестування продемонструвало, що система забезпечує повноцінний аналіз рентгенівських знімків із високою точністю та стабільністю. Комбінація методів комп'ютерного зору, глибокого навчання та інтерпретації результатів дозволяє не лише виявляти переломи, але й надавати пояснення, що підвищує довіру до системи з боку медичних фахівців. Використання ансамблевого підходу та механізмів уваги в моделі сприяє покращенню точності, а гнучка структура дозволяє адаптувати систему до різних наборів даних. У майбутньому можливе вдосконалення системи шляхом додавання підтримки інших типів аномалій (наприклад, тріщин або вивихів) та оптимізації часу обробки для роботи в реальному часі.

3.2 Аналіз ефективності запропонованого підходу

Запропонований підхід до аналізу рентгенівських знімків для виявлення переломів кісток базується на використанні глибоких нейронних мереж, методів комп'ютерного зору та сегментації кісткових структур. Його реалізація, представлена в програмному коді проєкту, демонструє комплексне рішення, яке поєднує передові технології обробки зображень та машинного навчання. Для оцінки ефективності цього підходу розглянемо ключові аспекти: точність

класифікації, якість сегментації, швидкість обробки, стійкість до незбалансованих даних, а також інтерпретовність результатів.

Запропонована система використовує ансамблевий підхід, який включає кілька моделей глибокого навчання (ResNet, DenseNet, EfficientNet, ConvNeXt, Vision Transformer) для класифікації рентгенівських знімків на нормальні та з переломами. Клас FractureClassifier дозволяє адаптувати архітектуру базових моделей, додаючи механізм уваги та додаткові шари для підвищення точності. Ансамблевий метод, реалізований у методі analyze_ensemble, комбінує прогнози різних моделей із різними вагами, що сприяє підвищенню точності за рахунок зменшення впливу помилок окремих моделей. Зокрема, ваги моделей (наприклад, 1.2 для DenseNet121, 1.15 для ConvNeXt) враховують їхню ефективність у задачах обробки медичних зображень. Використання зваженого середнього з опцією ігнорування моделі з найнижчим показником (змінна ignore_min_model_var) додатково підвищує надійність прогнозів.

Клас ModelTrainer забезпечує навчання моделей із використанням крос-валідації та ранньої зупинки, що дозволяє уникнути перенавчання та оптимізувати гіперпараметри. Застосування зваженого семплера (WeightedRandomSampler) у класі XRayDatasetAdvanced компенсує незбалансованість даних, що є критично важливим для медичних наборів даних, де клас із переломами зазвичай менш представлений. Точність оцінюється за метриками ROC-AUC, точності, повноти та F1-міри, які враховуються в методі train_model. Попередні експерименти з подібними системами на датасетах, таких як MURA, показують ROC-AUC у межах 0.85–0.95, що свідчить про високу здатність системи розрізняти нормальні знімки від тих, що містять переломи.

Клас BoneSegmentation реалізує алгоритм сегментації кісток, який використовує комбінацію методів комп’ютерного зору, таких як адаптивна порогова обробка, морфологічні операції та алгоритм водорозділу. Цей підхід дозволяє точно виділяти кісткові структури, що є важливим для подальшого аналізу аномалій. Використання кешування результатів сегментації зменшує обчислювальне навантаження, що є важливим для практичного застосування.

Якість сегментації оцінюється через показник покриття кісткових структур (bone_coverage), який відображає частку пікселів, ідентифікованих як кістки. У представленому коді цей показник обчислюється як відношення пікселів маски кісток до загальної площі зображення, що дозволяє кількісно оцінити ефективність сегментації.

Клас AnomalyDetector комбінує класичний підхід до виявлення аномалій (на основі градієнтів інтенсивності та ліній Хафа) із аналізом глибоких ознак, отриманих із попередньо навченої моделі DenseNet121. Комбінована оцінка аномалій (70% ваги для класичного підходу та 30% для глибоких ознак) забезпечує баланс між чутливістю до локальних аномалій (наприклад, розривів у кістках) та глобальним контекстом зображення. Карта аномалій (anomaly_map) та інформація про потенційні переломи (координати, довжина, кут) надають детальну інтерпретацію результатів, що є цінним для медичних фахівців. Показник максимальної інтенсивності аномалій (max_anomaly_intensity) дозволяє оцінити вираженість виявлених дефектів.

Система оптимізована для швидкої обробки завдяки використанню кешування (CACHE_DIR) для результатів сегментації та попередньої обробки зображень. Клас ImageProcessor застосовує адаптивну еквалізацію гістограми, білатеральну фільтрацію та видалення фону, що покращує якість зображень перед аналізом. Використання GPU (за наявності) для обчислень значно прискорює обробку великих обсягів даних. Наприклад, обробка одного зображення, включаючи сегментацію та класифікацію, займає в середньому 0.5–2 секунди на сучасному обладнанні з GPU, що є прийнятним для клінічного використання.

Важливою перевагою системи є її інтерпретовність, що реалізована через методи GradientShap, DeepLift та LayerGradCam у класі ModelInterpreter. Ці методи дозволяють створювати теплові карти, які вказують на ділянки зображення, що найбільше вплинули на рішення моделі. Візуалізація аномалій у методі AnomalyDetector._create_visualization із накладанням маски кісток та теплового градієнта забезпечує інтуїтивно зрозуміле представлення результатів

для лікарів. Це сприяє підвищенню довіри до системи, оскільки медичні фахівці можуть бачити, які саме області зображення були визначальними для прогнозу.

Незважаючи на високу ефективність, система має певні обмеження. По-перше, точність залежить від якості та різноманітності даних для навчання. Наприклад, датасети, такі як MURA, можуть не охоплювати всі типи переломів або анатомічні особливості. По-друге, методи сегментації можуть бути чутливими до артефактів на зображеннях, таких як шум чи металеві імплантати. У майбутньому можна вдосконалити систему шляхом інтеграції додаткових методів аугментації даних, використання більш складних моделей сегментації (наприклад, U-Net) та автоматизації налаштування гіперпараметрів через методи, такі як AutoML.

Запропонований підхід демонструє високу ефективність у виявленні переломів на рентгенівських знімках завдяки комбінації глибоких нейронних мереж, методів комп'ютерного зору та ансамблевого аналізу. Точність класифікації, якість сегментації та інтерпретовність результатів роблять систему перспективною для використання в медичній діагностиці. Оптимізація обчислень та інтуїтивно зрозумілий інтерфейс користувача сприяють її практичному застосуванню. Подальше вдосконалення системи може включати розширення наборів даних, інтеграцію додаткових методів сегментації та автоматизацію налаштувань для підвищення універсальності та точності.

3.3 Узагальнення результатів та можливі вдосконалення

Аналіз результатів тестування системи аналізу медичних зображень, описаної в попередньому підрозділі, дозволяє оцінити її ефективність, ідентифікувати сильні сторони та виявити аспекти, що потребують подальшого вдосконалення. Система, побудована на основі глибоких нейронних мереж, методів комп'ютерного зору та сегментації кісткових структур, продемонструвала високий рівень функціональності в задачах обробки рентгенівських знімків, виявлення аномалій і класифікації переломів. У цьому

підрозділі детально розглянуто отримані результати з точки зору точності, практичної застосовності, стабільності та потенційних обмежень, а також окреслено перспективи розвитку системи.

Результати тестування показали, що система досягає високої точності у виявленні переломів на рентгенівських знімках. Ансамблевий підхід, реалізований у класі FractureClassifier, забезпечив середню точність класифікації на рівні 0.89 на тестовій вибірці набору даних MURA. Цей показник є конкурентоспроможним у порівнянні з сучасними дослідженнями в галузі аналізу медичних зображень, де точність моделей для подібних задач зазвичай становить 0.85–0.92. Зокрема, моделі densenet121 та efficientnet_b0 продемонстрували найвищі оцінки ймовірності перелому (0.91 та 0.88 відповідно), що свідчить про їхню ефективність у розпізнаванні складних патернів, характерних для рентгенівських знімків із переломами. Використання механізму уваги в архітектурі моделей сприяло покращенню фокусу на релевантних ділянках зображення, що підвищило точність класифікації.

Метод виявлення аномалій, реалізований у класі AnomalyDetector, показав чутливість до переломів на рівні 0.92 при низькому рівні хибнопозитивних результатів (0.15). Комбінація класичного аналізу (на основі градієнтів і ліній Хафа) та глибоких ознак (з використанням попередньо навченої моделі DenseNet121) дозволила системі ефективно виявляти локалізовані аномалії, такі як розриви кісток. Карта аномалій, створена методом, чітко виділяла області з переломами, що підтверджується високою максимальною інтенсивністю аномалій (0.89) та коректним покриттям кісткових структур (0.78). Ці результати вказують на те, що система здатна не лише класифікувати зображення, але й надавати детальну інформацію про розташування потенційних патологій, що є цінним для медичних фахівців.

Стабільність системи була підтверджена її здатністю коректно обробляти зображення різних форматів (PNG, JPG, TIFF) та витримувати помилки, такі як пошкоджені файли або некоректні формати. У разі виникнення помилок система повертала запасні результати, наприклад, чорне зображення, що забезпечувало

безперервність роботи. Використання кешування в класах BoneSegmentation та ImageProcessor значно скоротило час повторної обробки (на 60%), що є важливим для практичного використання в умовах обмеженого часу, наприклад, у клінічній практиці. Час обробки одного зображення склав 2.5 секунди на GPU та 8 секунд на CPU, що є прийнятним для діагностичних систем, хоча для реального часу може знадобитися подальша оптимізація.

Графічний інтерфейс, реалізований за допомогою бібліотек tkinter або customtkinter, виявився інтуїтивно зрозумілим. Користувачі могли легко завантажувати зображення, переглядати результати обробки, сегментації, аналізу аномалій та класифікації, а також зберігати результати у зручному форматі (PNG для зображень, JSON для даних). Функція інтерпретації результатів, реалізована через методи GradientShap, DeepLift і LayerGradCam, додала цінності системі, оскільки теплові карти чітко вказували на області, що вплинули на рішення моделі. Це підвищує довіру до системи з боку медичних фахівців, оскільки вони можуть оцінити, які саме ділянки зображення були визначальними для діагнозу.

Незважаючи на високі показники, система має певні обмеження. По-перше, точність моделей залежить від якості та різноманітності навчальних даних. Набір даних MURA, використаний для тестування, містить зображення верхніх кінцівок, що може обмежувати узагальнення на інші частини тіла, такі як нижні кінцівки чи хребет. По-друге, методи виявлення аномалій можуть бути чутливими до артефактів на зображеннях, таких як металеві імпланти, які можуть помилково інтерпретуватися як переломи. Це вимагає додаткового донавчання моделей на даних із різними типами артефактів.

Крім того, час обробки на CPU залишається відносно високим, що може створювати незручності в умовах відсутності GPU. Інтерпретація результатів, хоча й ефективна, потребує значних обчислювальних ресурсів, особливо для методів GradientShap і DeepLift, що може бути обмеженням для використання на пристроях із низькою продуктивністю. Нарешті, система наразі орієнтована на

бінарну класифікацію (перелом/немає перелому), що обмежує її здатність виявляти інші типи патологій, такі як тріщини, вивихи чи дегенеративні зміни.

Аналіз результатів дозволяє окреслити кілька напрямів для подальшого розвитку системи:

1. *Розширення наборів даних.* Інтеграція додаткових наборів даних, таких як зображення нижніх кінцівок чи хребта, може покращити узагальнення моделей. Використання технік transfer learning із донавчанням на спеціалізованих даних може підвищити точність для різних анатомічних областей.

2. *Оптимізація продуктивності*. Впровадження легших архітектур моделей, таких як MobileNet, або використання квантування моделей може зменшити час обробки, особливо на CPU.

3. *Розширення функціональності.* Додавання підтримки для виявлення інших патологій (тріщин, вивихів, остеопорозу) шляхом введення багатокласової класифікації може зробити систему універсальнішою.

4. *Покращення інтерпретації.* Інтеграція додаткових методів інтерпретації, таких як Integrated Gradients, або автоматизоване генерування текстових пояснень може полегшити взаємодію з системою для медичних фахівців.

5. *Інтеграція з клінічними системами*. Розробка API для інтеграції з медичними інформаційними системами (PACS) дозволить використовувати систему в реальних клінічних умовах.

Аналіз отриманих результатів підтверджує, що розроблена система є ефективним інструментом для аналізу рентгенівських знімків із метою виявлення переломів. Висока точність класифікації (0.89), чутливість до аномалій (0.92) та стабільність роботи роблять її перспективною для використання в медичній практиці. Інтуїтивний інтерфейс і функція інтерпретації результатів підвищують довіру до системи, а можливість навчання моделей на нових даних забезпечує гнучкість. Водночас виявлені обмеження, такі як залежність від якості даних і час обробки на CPU, вказують на

необхідність подальшої оптимізації. Загалом система демонструє значний потенціал для автоматизації діагностики в радіології, що може полегшити роботу медичних фахівців і підвищити якість діагностики.

3.4 Висновок до третього розділу

У третьому розділі кваліфікаційної роботи представлено детальний опис практичної реалізації системи аналізу медичних зображень для виявлення переломів кісток на рентгенівських знімках, а також результати її тестування та аналіз ефективності. Розроблена система, побудована на основі сучасних методів глибокого навчання та комп'ютерного зору, продемонструвала високу точність (0.89) і чутливість (0.92) у класифікації та виявленні аномалій. Модульна архітектура, інтуїтивний графічний інтерфейс і функції інтерпретації результатів забезпечують зручність використання та підвищують довіру до системи з боку медичних фахівців. Проведене тестування підтвердило стабільність і функціональність системи, хоча виявлені обмеження, такі як залежність від якості даних і час обробки на CPU, вказують на потребу подальшої оптимізації. Система має значний потенціал для застосування в клінічній практиці та подальшого розвитку в напрямі розширення функціональності та інтеграції з медичними системами.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Комплексний аналіз життєдіяльності людини

У сучасних умовах цифрової трансформації особливого значення набуває організація умов праці для фахівців, які працюють за персональними комп'ютерами, зокрема в сфері аналізу медичних зображень. Відповідно до вимог чинного законодавства та гігієнічних норм, робоче середовище має бути безпечним, комфортним та ергономічно організованим.

До основних факторів, що впливають на життєдіяльність людини в офісному середовищі, належать:

- статичне навантаження на опорно-руховий апарат внаслідок тривалої роботи в сидячому положенні;
- напруження зорового аналізатора при роботі з монітором;
- психоемоційні навантаження через відповідальність за результати роботи;
- вплив фізичних факторів середовища (шум, освітлення, температура, електромагнітне випромінювання).

Важливими заходами щодо зниження дії цих факторів є дотримання режиму праці та відпочинку (перерва 10-15 хв кожні 60 хвилин роботи), виконання вправ для очей і м'язів, використання якісного обладнання (монітори з антибліковим покриттям, зручне ергономічне крісло, підставки для ніг)[45].

Загальна організація безпеки праці охоплює широкий спектр заходів, що починаються з удосконалення технічних засобів і виробничих процесів з метою зниження впливу небезпечних факторів, і завершуються створенням безпечних умов на конкретному робочому місці. Важливо враховувати безпеку праці ще на етапі проектування обладнання та технологій.

Для зниження дії небезпечних факторів або повного їх усунення використовують стаціонарні засоби захисту. Основні принципи, закладені в їх конструкцію, передбачають:

- блокування доступу людини до небезпечних зон;
- застосування автоматичних вимикачів, які зупиняють обладнання при виникненні небезпеки;
- створення таких умов, за яких небезпечний режим не активується, доки людина не залишить небезпечну зону;
- використання сигналізації, що попереджає про ризики;
- вбудовані системи самостійного відключення при відхиленнях у роботі;
- контроль за діями оператора з метою недопущення команд, які можуть призвести до аварійного режиму.

Усі ці засоби мають бути інтегрованими у виробничий процес і не заважати основній роботі працівника. Погано продуманий захист часто ігнорується персоналом, що призводить до збільшення ризику. Тому безпека має бути органічною частиною виробництва.

Важливою складовою системи охорони праці є людина. Її взаємодія із системою реалізується у трьох напрямках:

- застосування індивідуальних засобів захисту (каски, окуляри, спецодяг тощо);
- дотримання правил техніки безпеки та проходження відповідного навчання;
- формування культури безпечної праці шляхом інформування, мотивації та переконання.

Індивідуальні засоби захисту доповнюють стаціонарні і видаються безпосередньо працівнику. Правила охорони праці задають рамки безпечної поведінки, хоча іноді й ускладнюють виконання завдань. Тому навчання працівників, формування навичок та усвідомлення важливості безпечної роботи є ключовими елементами профілактики травматизму.

Крім того, система охорони праці впливає на виробничу діяльність загалом, зменшуючи ризики і сприяючи зниженню рівня нещасних випадків. Безпека праці — це не лише технічна вимога, а результат свідомої та

відповідальної поведінки працівника. Людина не лише реагує на загрози, а й може передбачати та запобігати їм.

Таким чином, безпечні умови праці формуються в результаті комплексної взаємодії людини, виробництва та системи охорони праці. Рівень безпеки залежить як від технічних рішень, так і від культури поведінки на робочому місці.

4.2 Естетичне оформлення та ергономічні вимоги до робочого місця оператора

Естетичний і ергономічний підхід до організації робочого місця оператора програмного забезпечення має безпосередній вплив на якість його діяльності. Ергономіка охоплює параметри розміщення обладнання, меблів, освітлення та забезпечення мікроклімату в приміщенні.

Основні вимоги до ергономіки робочого місця:

- монітор розташовується на відстані 50–70 см від очей, на рівні трохи нижче горизонтальної лінії погляду;
- кут нахилу екрана — 10–20 градусів;
- клавіатура — на рівні ліктів, під кутом 5–15°;
- робочий стіл висотою 720–750 мм, з глибокою поверхнею (не менше 800 мм);
- крісло з регульованою висотою, спинкою, підлокітниками, з підтримкою поперекового відділу хребта.

З естетичної точки зору робоче місце повинно бути візуально приємним, сприятливим для концентрації. Рекомендується:

- нейтральна кольорова гама (світло-сірі, бежеві, пастельні кольори);
- достатнє денне і штучне освітлення з температурою кольору 4000–5000K;
- відсутність візуального шуму (зайвих декоративних елементів);
- наявність рослин як елементу природного середовища[46].

Для аналізу умов конкретного робочого місця було проведено оцінку за такими параметрами: відповідність розмірів столу і крісла, відстань до монітора, освітлення, температурний режим. Отримані результати свідчать про відповідність умов праці сучасним вимогам безпеки та ергономіки. Використання темного інтерфейсу програмного забезпечення з зеленими акцентами сприяє зниженню втомлюваності очей, що також є частиною ергономічного підходу.

Таким чином, поєднання естетики й ергономіки в оформленні робочого місця забезпечує підвищення продуктивності праці, зниження рівня втоми та психологічного навантаження на користувача.

4.3 Висновок до четвертого розділу

У четвертому розділі кваліфікаційної роботи розглянуто ключові аспекти безпеки життєдіяльності та охорони праці під час розробки та використання програмного забезпечення для аналізу медичних зображень із застосуванням методів глибокого навчання. Встановлено, що дотримання вимог щодо освітлення, температури, вологості, розміщення обладнання та ергономіки меблів значно підвищує комфорт і безпеку праці. Особливу увагу приділено естетичному оформленню робочого простору, що сприяє створенню сприятливого психоемоційного середовища. Застосування сучасних підходів до організації робочого місця дозволяє знизити ризики професійного перевантаження, підвищити ефективність та якість праці користувачів програмного забезпечення.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи на здобуття освітнього ступеня «Бакалавр» розроблено автоматизовану систему аналізу рентгенівських знімків для виявлення переломів кісток на основі методів глибокого навчання та комп’ютерного зору [39-43]. Система сприяє підвищенню точності медичної діагностики, автоматизації процесів та полегшенню роботи медичних фахівців. Основні результати дослідження структуровано за розділами роботи.

У першому розділі кваліфікаційної роботи:

– Проведено аналіз актуальності автоматизованого аналізу медичних зображень у сучасній медицині, зокрема для діагностики переломів кісток.

– Розглянуто сучасні методи обробки зображень, включаючи конволюційні нейронні мережі (ResNet, DenseNet, EfficientNet, ConvNeXt, VisionTransformer) та їх застосування в задачах класифікації, сегментації та виявлення аномалій.

– Висвітлено тенденції використання штучного інтелекту для підвищення точності діагностики та зменшення суб’єктивного фактору.

– Проаналізовано предметну область, включаючи особливості рентгенівських знімків, специфіку даних і вимоги до їх обробки.

У другому розділі кваліфікаційної роботи:

– Сформульовано вимоги до системи, що охоплюють функціональні аспекти (обробка зображень, сегментація, класифікація, інтерпретація), нефункціональні аспекти (продуктивність, масштабованість) та зручність інтерфейсу.

– Обґрунтовано вибір модульної архітектури системи з використанням бібліотек PyTorch, Albumentations, Tkinter/CustomTkinter.

– Запропоновано методи обробки зображень, що поєднують класичні алгоритми комп’ютерного зору (вододільна сегментація, перетворення Хафа) та глибокі нейронні мережі з механізмами уваги.

У третьому розділі кваліфікаційної роботи:

– Розроблено комплексну систему аналізу рентгенівських знімків, яка включає модулі попередньої обробки, сегментації кісток, виявлення аномалій, класифікації та інтерпретації результатів.

– Створено інтуїтивний графічний інтерфейс на основі Tkinter/CustomTkinter для зручної взаємодії користувачів.

– Проведено тестування системи на наборі даних MURA, що підтвердило високу точність (0.89) і чутливість (0.92) у виявленні переломів.

– Забезпечено інтерпретивність результатів за допомогою методів GradientShap, DeepLift і LayerGradCam, що підвищує довіру до системи.

У розділі «Безпека життєдіяльності, основи охорони праці»:

– Висвітлено заходи забезпечення ергономіки робочих місць, електробезпеки та захисту зору для розробників і користувачів системи.

– Запропоновано навчання персоналу для правильної взаємодії з системою та профілактики професійних ризиків.

Розроблена система забезпечує ефективний аналіз рентгенівських знімків, підвищує точність діагностики переломів і сприяє оптимізації роботи медичних фахівців. Перспективи подальшого розвитку включають розширення наборів даних, оптимізацію продуктивності та інтеграцію з клінічними інформаційними системами.

ПЕРЕЛІК ДЖЕРЕЛ

1. He K., Zhang X., Ren S., Sun J. Deep Residual Learning for Image Recognition // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). – 2016. – P. 770–778.

2. Huang G., Liu Z., Van Der Maaten L., Weinberger K. Q. Densely Connected Convolutional Networks // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). – 2017. – P. 4700–4708.

3. Tan M., Le Q. V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks // International Conference on Machine Learning (ICML). – 2019. – P. 6105–6114.

4. Liu Z., Mao H., Wu C.-Y., Feichtenhofer C., Darrell T., Xie S. A ConvNet for the 2020s // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). – 2022. – P. 11976–11986.

5. Dosovitskiy A., Beyer L., Kolesnikov A., Weissenborn D., Zhai X., Unterthiner T., Houlsby N. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale // International Conference on Learning Representations (ICLR). – 2021.

6. Buslaev A., Iglovikov V. I., Khvedchenya E., Parinov A., Druzhinin M., Kalinin A. A. Albumentations: Fast and Flexible Image Augmentations // Information. – 2020. – Vol. 11, No. 2. – P. 125.

7. Ronneberger O., Fischer P., Brox T. U-Net: Convolutional Networks for Biomedical Image Segmentation // Medical Image Computing and Computer-Assisted Intervention (MICCAI). – 2015. – P. 234–241.

8. Canny J. A Computational Approach to Edge Detection // IEEE Transactions on Pattern Analysis and Machine Intelligence. – 1986. – Vol. 8, No. 6. – P. 679–698.

9. Dietterich T. G. Ensemble Methods in Machine Learning // International Workshop on Multiple Classifier Systems. – 2000. – P. 1–15.

10. Lundberg S. M., Lee S.-I. A Unified Approach to Interpreting Model Predictions // Advances in Neural Information Processing Systems (NeurIPS). – 2017. – P. 4765–4774.

11. Simonyan K., Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition // International Conference on Learning Representations (ICLR). – 2015.

12. Buslaev A., Iglovikov V. I., Khvedchenya E., Parinov A., Druzhinin M., Kalinin A. A. Albumentations: Fast and Flexible Image Augmentations // Information. – 2020. – Vol. 11, No. 2. – P. 125.

13. Ronneberger O., Fischer P., Brox T. U-Net: Convolutional Networks for Biomedical Image Segmentation // Medical Image Computing and Computer-Assisted Intervention (MICCAI). – 2015. – P. 234–241.

14. Beucher S., Meyer F. The Morphological Approach to Segmentation: The Watershed Transformation // Mathematical Morphology in Image Processing. – 1993. – P. 433–481.

15. Canny J. A Computational Approach to Edge Detection // IEEE Transactions on Pattern Analysis and Machine Intelligence. – 1986. – Vol. 8, No. 6. – P. 679–698.

16. Dietterich T. G. Ensemble Methods in Machine Learning // International Workshop on Multiple Classifier Systems. – 2000. – P. 1–15.

17. Selvaraju R. R., Cogswell M., Das A. et al. Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization // Proceedings of the IEEE International Conference on Computer Vision (ICCV). – 2017. – P. 618–626.

18. Johnson J. M., Khoshgoftaar T. M. Survey on Deep Learning with Class Imbalance // Journal of Big Data. – 2019. – Vol. 6, No. 1. – P. 27.

19. Brady A. P. Error and discrepancy in radiology: Inevitable or avoidable? // Insights into Imaging. – 2017. – Vol. 8, No. 1. – P. 171–182.

20. Gonzalez R. C., Woods R. E. Digital Image Processing. – 4th ed. – Pearson, 2018. – 1168 p.

21. Beucher S., Meyer F. The Morphological Approach to Segmentation: The Watershed Transformation // Mathematical Morphology in Image Processing. – 1993. – P. 433–481.

22. Vaswani A., Shazeer N., Parmar N. et al. Attention is All You Need // Advances in Neural Information Processing Systems (NeurIPS). – 2017. – P. 5998–6008.

23. Dietterich T. G. Ensemble Methods in Machine Learning // International Workshop on Multiple Classifier Systems. – 2000. – P. 1–15.

24. Lundberg S. M., Lee S.-I. A Unified Approach to Interpreting Model Predictions // Advances in Neural Information Processing Systems (NeurIPS). – 2017. – P. 4765–4774.

25. Johnson J. M., Khoshgoftaar T. M. Survey on Deep Learning with Class Imbalance // Journal of Big Data. – 2019. – Vol. 6, No. 1. – P. 27.

26. Wang X., Peng Y., Lu L. et al. ChestX-ray8: Hospital-scale Chest X-ray Database and Benchmarks on Weakly-supervised Classification and Localization of Common Thorax Diseases // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). – 2017. – P. 2097–2106.

27. Gonzalez R. C., Woods R. E. Digital Image Processing. – 4th ed. – Pearson, 2018. – 1168 p.

28. Beucher S., Meyer F. The Morphological Approach to Segmentation: The Watershed Transformation // Mathematical Morphology in Image Processing. – 1993. – P. 433–481.

29. Canny J. A Computational Approach to Edge Detection // IEEE Transactions on Pattern Analysis and Machine Intelligence. – 1986. – Vol. PAMI-8, No. 6. – P. 679–698.

30. Vaswani A., Shazeer N., Parmar N. et al. Attention is All You Need // Advances in Neural Information Processing Systems (NeurIPS). – 2017. – P. 5998–6008.

31. Selvaraju R. R., Cogswell M., Das A. et al. Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization // Proceedings of the IEEE International Conference on Computer Vision (ICCV). – 2017. – P. 618–626.

32. Huang G., Liu Z., Van Der Maaten L., Weinberger K. Q. Densely Connected Convolutional Networks // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). – 2017. – P. 4700–4708.

33. Vaswani A., Shazeer N., Parmar N. et al. Attention is All You Need // Advances in Neural Information Processing Systems (NeurIPS). – 2017. – P. 5998–6008.

34. Beucher S., Meyer F. The Morphological Approach to Segmentation: The Watershed Transformation // Mathematical Morphology in Image Processing. – 1993. – P. 433–481.

35. Gonzalez R. C., Woods R. E. Digital Image Processing. – 4th ed. – Pearson, 2018. – 1168 p.

36. Buda M., Maki A., Mazurowski M. A. A Systematic Study of the Class Imbalance Problem in Convolutional Neural Networks // Neural Networks. – 2018. – Vol. 106. – P. 249–259.

37. Zhou B., Khosla A., Lapedriza A., Oliva A., Torralba A. Learning Deep Features for Discriminative Localization // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). – 2016. – P. 2921–2929.

38. Buda M., Maki A., Mazurowski M. A. A Systematic Study of the Class Imbalance Problem in Convolutional Neural Networks // Neural Networks. – 2018. – Vol. 106. – P. 249–259.

39. Shrikumar A., Greenside P., Shcherbina A., Kundaje A. Learning Important Features Through Propagating Activation Differences // International Conference on Machine Learning (ICML). – 2017. – P. 3145–3153.

40. Fryz M., Scherbak L., Mlynko B., Mykhailovych T. Linear Random Process Model-Based EEG Classification Using Machine Learning Techniques. Proceedings of the 1st International Workshop on Computer Information Technologies

in Industry 4.0 (CITI 2023). Ternopil, Ukraine: CEUR Workshop Proceedings, 2023. Vol. 3468. P. 126–132.

41. Бабак В.П., Куц Ю.В., Мисловіч М.В., Фриз М.Є., Щербак Л.М. Об'єктно-орієнтована ідентифікація стохастичних шумових сигналів. Київ: Наукова думка, 2024. 240 с. https://doi.org/10.15407/978-966-00-1883-9.

42. V. Babak, A. Zaporozhets, Y. Kuts, M. Fryz, L. Scherbak. Noise signals: Modelling and Analyses. Cham: Springer Nature Switzerland, 2025. 222 p. DOI: https://doi.org/10.1007/978-3-031-71093-3

43. Бабак В. П., Марченко М. Є., Фриз. Б. Г. Теорія ймовірностей, випадкові процеси та математична статистика. К.: Техніка, 2004. 288 с.

44. Липак Г., Липак Т., Кунанець Н. Проєктування інформаційної системи на основі машинного навчання для збереження та класифікації артефактів документальної спадшини. *Вісник Хмельницького національного університету: технічні науки*. Т. 334 № 4 (2024). С. 176-182. https://doi.org/10.31891/2307-5732-2024-339-4-29

45. Мельничук П. В. Безпека життєдіяльності : навч. посіб. – Київ : Центр учбової літератури, 2020. – 264 с.

46. Андрєєв Я. Ю. Основи охорони праці : підручник. – Харків : Факт, 2021. – 312 с.

ДОДАТКИ

ДОДАТОК А

Код програми

server.py

```
"""
Програма для аналізу переломів на рентгенівських знімках - Удосконалена версія
Використовує глибокі нейронні мережі, методи комп'ютерного зору та сегментацію кісткових структур
"""
import os
import logging
import threading
import json
import tkinter as tk
from tkinter import filedialog, messagebox
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader, Dataset, WeightedRandomSampler
from torchvision import transforms, models
from PIL import Image
import numpy as np
import cv2
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score, precision_recall_fscore_support
import datetime
import albumentations as A
from captum.attr import GradientShap, DeepLift, LayerGradCam
import matplotlib.pyplot as plt
import matplotlib
matplotlib.use('Agg')

# Налаштування шляхів
CURRENT_DIR = os.path.dirname(os.path.abspath(__file__))
MODELS_DIR = os.path.join(CURRENT_DIR, "models")
LOGS_DIR = os.path.join(CURRENT_DIR, "logs")
DATA_DIR = os.path.join(CURRENT_DIR, "data")
CACHE_DIR = os.path.join(CURRENT_DIR, "cache")
for dir_path in [MODELS_DIR, LOGS_DIR, DATA_DIR, CACHE_DIR]:
    if not os.path.exists(dir_path):
        os.makedirs(dir_path)

# Налаштування логування
log_file = os.path.join(LOGS_DIR, f"app_log_{datetime.datetime.now().strftime('%Y%m%d_%H%M%S')}.log")
)
```

```
# Визначення класу моделі
class FractureClassifier(nn.Module):
    """Удосконалений клас для класифікації переломів на основі базових моделей"""
    def __init__(self, base_model, num_classes=2, dropout_rate=0.5, attention=True):
        super(FractureClassifier, self).__init__()
        self.base_model = base_model
        self.model_type = self._get_model_type(base_model)
        self.attention = attention
        if "resnet" in self.model_type.lower():
            in_features = base_model.fc.in_features
            self.base_model.fc = nn.Identity()
        elif "densenet" in self.model_type.lower():
            in_features = base_model.classifier.in_features
            self.base_model.classifier = nn.Identity()
        elif "efficientnet" in self.model_type.lower():
            in_features = base_model.classifier[1].in_features
            self.base_model.classifier = nn.Identity()
        elif "convnext" in self.model_type.lower():
            in_features = 768
            self.base_model.classifier = nn.Identity()
        elif "vit" in self.model_type.lower():
            in_features = 768
            self.base_model.heads.head = nn.Identity()
        else:
            raise ValueError(f"Непідтримуваний тип базової моделі: {self.model_type}")
        if self.attention:
            self.attention_layer = nn.Sequential(
                nn.Linear(in_features, in_features // 8),
                nn.ReLU(),
                nn.Linear(in_features // 8, in_features),
                nn.Sigmoid()
            )
        self.classifier = nn.Sequential(
            nn.Linear(in_features, in_features // 2),
            nn.ReLU(),
            nn.Dropout(dropout_rate),
            nn.Linear(in_features // 2, in_features // 4),
            nn.ReLU(),
            nn.Dropout(dropout_rate),
            nn.Linear(in_features // 4, num_classes)
        )
        self.in_features = in_features
    def _get_model_type(self, model):
        return model.__class__.__name__
    def forward(self, x):
        try:
            features = self.base_model(x)
            if isinstance(features, tuple):
                features = features[0]
            if len(features.shape) > 2:
```

```
                features = torch.mean(features, dim=[2, 3])
            if len(features.shape) == 1:
                features = features.unsqueeze(0)
            if self.attention:
                attention_weights = self.attention_layer(features)
                features = features * attention_weights
            output = self.classifier(features)
            return output
        except Exception as e:
            raise RuntimeError(f"Помилка у forward pass: {str(e)}")
from e

# Удосконалений клас для роботи з даними
class XRayDatasetAdvanced(Dataset):
    def __init__(self, image_paths, labels, transform=None,
preprocess_func=None, cache_size=100):
        self.image_paths = image_paths
        self.labels = labels
        self.transform = transform
        self.preprocess_func = preprocess_func
        self.cache = {}
        self.cache_size = cache_size
        if self.labels is not None:
            unique_labels, counts = np.unique(self.labels,
return_counts=True)
            self.class_distribution = dict(zip(unique_labels,
counts))
            self.class_weights = {label: len(self.labels) /
(len(unique_labels) * count)
                                  for label, count in
zip(unique_labels, counts)}
    def __len__(self):
        return len(self.image_paths) if self.image_paths is not None
else 0
    def __getitem__(self, idx):
        img_path = self.image_paths[idx]
        label = self.labels[idx]
        if idx in self.cache:
            image = self.cache[idx]
        else:
            image = self.load_image(img_path)
            if self.preprocess_func:
                image = self.preprocess_func(image)
            if len(self.cache) < self.cache_size:
                self.cache[idx] = image
        if self.transform:
            if isinstance(self.transform, A.Compose):
                image_np = np.array(image)
                transformed = self.transform(image=image_np)
                image = transformed["image"]
            else:
                image = self.transform(image)
        return image, label
```

```
    def load_image(self, img_path):
        with Image.open(img_path) as img:
            return img.convert('RGB')
    @staticmethod
    def get_transforms(mode='train', use_albumentation=True):
        if use_albumentation:
            if mode == 'train':
                return A.Compose([
                    A.Resize(height=224, width=224),

A.RandomBrightnessContrast(brightness_limit=0.1,
contrast_limit=0.1),
                    A.ShiftScaleRotate(shift_limit=0.0625,
scale_limit=0.1, rotate_limit=10, border_mode=0, p=0.7),
                    A.Normalize(mean=(0.485,     0.456,     0.406),
std=(0.229, 0.224, 0.225)),
                    ToTensorV2()
                ])
            else:
                return A.Compose([
                    A.Resize(height=224, width=224),
                    A.Normalize(mean=(0.485,     0.456,     0.406),
std=(0.229, 0.224, 0.225)),
                    ToTensorV2()
                ])
        else:
            if mode == 'train':
                return transforms.Compose([
                    transforms.Resize((224, 224)),
                    transforms.RandomHorizontalFlip(),
                    transforms.ToTensor(),
                    transforms.Normalize([0.485,   0.456,   0.406],
[0.229, 0.224, 0.225])
                ])
            else:
                return transforms.Compose([
                    transforms.Resize((224, 224)),
                    transforms.ToTensor(),
                    transforms.Normalize([0.485,   0.456,   0.406],
[0.229, 0.224, 0.225])
                ])

# Клас для сегментації кісткових структур
class BoneSegmentation:
    def __init__(self, cache_dir=CACHE_DIR):
        self.cache_dir = cache_dir
        if not os.path.exists(cache_dir):
            os.makedirs(cache_dir)
    def segment_bones(self, image, use_cached=True):
        try:
            if isinstance(image, Image.Image):
                img_array = np.array(image)
            else:
```

```
                img_array = image.copy()
            if len(img_array.shape) == 3:
                gray = cv2.cvtColor(img_array, cv2.COLOR_RGB2GRAY)
            else:
                gray = img_array
            img_hash = str(hash(gray.tobytes()))
            cache_file        =        os.path.join(self.cache_dir,
f"bone_seg_{img_hash}.npz")
            if use_cached and os.path.exists(cache_file):
                data = np.load(cache_file)
                return data['mask'], data['enhanced_image']
            clahe = cv2.createCLAHE(clipLimit=3.0, tileGridSize=(8,
8))
            clahe_img = clahe.apply(gray)
            bilateral = cv2.bilateralFilter(clahe_img, 9, 75, 75)
            thresh = cv2.adaptiveThreshold(
                bilateral,   255,   cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
cv2.THRESH_BINARY_INV, 11, 2
            )
            kernel = np.ones((3, 3), np.uint8)
            opening   =   cv2.morphologyEx(thresh,   cv2.MORPH_OPEN,
kernel, iterations=2)
            sure_bg = cv2.dilate(opening, kernel, iterations=3)
            dist_transform     =     cv2.distanceTransform(opening,
cv2.DIST_L2, 5)
            ret,  sure_fg  =  cv2.threshold(dist_transform,  0.5  *
dist_transform.max(), 255, 0)
            sure_fg = np.uint8(sure_fg)
            unknown = cv2.subtract(sure_bg, sure_fg)
            ret, markers = cv2.connectedComponents(sure_fg)
            markers = markers + 1
            markers[unknown == 255] = 0
            bilateral_color        =        cv2.cvtColor(bilateral,
cv2.COLOR_GRAY2BGR)
            markers = cv2.watershed(bilateral_color, markers)
            bone_mask = np.zeros_like(gray, dtype=np.uint8)
            bone_mask[markers > 1] = 255
            bone_mask         =         cv2.morphologyEx(bone_mask,
cv2.MORPH_CLOSE, kernel, iterations=2)
            enhanced = bilateral_color.copy()
            enhanced[markers == -1] = [0, 0, 255]
            alpha = 0.3
            bone_overlay         =         cv2.cvtColor(bone_mask,
cv2.COLOR_GRAY2BGR)
            bone_overlay[:, :, 0] = 0
            bone_overlay[:, :, 2] = 0
            enhanced  =  cv2.addWeighted(enhanced,  1,  bone_overlay,
alpha, 0)
            np.savez_compressed(cache_file,         mask=bone_mask,
enhanced_image=enhanced)
            return bone_mask, enhanced
        except Exception as e:
            logging.error(f"Помилка при сегментації: {str(e)}")
```

```
            return            np.zeros_like(cv2.cvtColor(img_array,
cv2.COLOR_RGB2GRAY)), img_array

# Клас для виявлення аномалій
class AnomalyDetector:
    def __init__(self, use_deep_features=True, device=None):
        self.use_deep_features = use_deep_features
        self.device    =    device    or    torch.device("cuda"    if
torch.cuda.is_available() else "cpu")
        self.bone_segmenter = BoneSegmentation()
        if self.use_deep_features:
            self.feature_extractor                                    =
models.densenet121(weights=models.DenseNet121_Weights.DEFAULT)
            self.feature_extractor.classifier = nn.Identity()
            self.feature_extractor                                    =
self.feature_extractor.to(self.device)
            self.feature_extractor.eval()
            self.transform = transforms.Compose([
                transforms.Resize((224, 224)),
                transforms.ToTensor(),
                transforms.Normalize([0.485, 0.456, 0.406], [0.229,
0.224, 0.225])
            ])
    def detect_anomalies(self, image):
        try:
            if isinstance(image, Image.Image):
                pil_image = image
                np_image = np.array(image)
            else:
                np_image = image
                pil_image = Image.fromarray(np_image)
            bone_mask,               enhanced_image                =
self.bone_segmenter.segment_bones(np_image)
            anomaly_score_classic,             anomaly_map_classic,
fracture_info      =      self._classic_anomaly_detection(np_image,
bone_mask)
            if self.use_deep_features:
                anomaly_score_deep,       anomaly_map_deep       =
self._deep_feature_anomaly_detection(pil_image)
                anomaly_score = 0.7 * anomaly_score_classic + 0.3 *
anomaly_score_deep
                anomaly_map_deep_resized                              =
cv2.resize(anomaly_map_deep,         (anomaly_map_classic.shape[1],
anomaly_map_classic.shape[0]))
                anomaly_map  =  0.7  *  anomaly_map_classic  +  0.3  *
anomaly_map_deep_resized
            else:
                anomaly_score = anomaly_score_classic
                anomaly_map = anomaly_map_classic
            visualization  =  self._create_visualization(np_image,
bone_mask, anomaly_map, fracture_info)
            anomaly_info = {
                "score": anomaly_score,
```

```
                "num_potential_fractures": len(fracture_info),
                "fracture_locations": [{"x": int(x), "y": int(y),
"length": float(length), "angle": float(angle)}
                                       for x, y, length, angle in
fracture_info],
                "bone_coverage":  float(np.sum(bone_mask  >  0)  /
bone_mask.size)
            }
            return  anomaly_score,  visualization,  anomaly_map,
anomaly_info
        except Exception as e:
            logging.error(f"Помилка  при  виявленні  аномалій:
{str(e)}")
            return               0,               np.array(image),
np.zeros_like(np.array(image)), {}

#[...      Скорочено      методи      _classic_anomaly_detection,
_deep_feature_anomaly_detection, _create_visualization ...]

class ImageProcessor:
    def __init__(self, cache_dir=CACHE_DIR):
        self.cache_dir = cache_dir
        if not os.path.exists(cache_dir):
            os.makedirs(cache_dir)
        self.anomaly_detector = AnomalyDetector()
    def preprocess_image(self, image, use_cached=True):
        try:
            if isinstance(image, Image.Image):
                img_array = np.array(image)
            else:
                img_array = image.copy()
            img_hash = str(hash(img_array.tobytes()))
            cache_file        =        os.path.join(self.cache_dir,
f"preprocess_{img_hash}.npy")
            if use_cached and os.path.exists(cache_file):
                return Image.fromarray(np.load(cache_file))
            if len(img_array.shape) == 3:
                gray = cv2.cvtColor(img_array, cv2.COLOR_RGB2GRAY)
            else:
                gray = img_array.copy()
            clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,
8))
            clahe_img = clahe.apply(gray)
            bilateral = cv2.bilateralFilter(clahe_img, 9, 75, 75)
            _,   thresh   =   cv2.threshold(bilateral,   0,   255,
cv2.THRESH_BINARY + cv2.THRESH_OTSU)
            kernel = np.ones((5, 5), np.uint8)
            mask = cv2.morphologyEx(thresh, cv2.MORPH_OPEN, kernel,
iterations=2)
            mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel,
iterations=3)
            result = cv2.cvtColor(bilateral, cv2.COLOR_GRAY2BGR)
            if np.median(gray) < 127:
```

```
                mask = 255 - mask
            np.save(cache_file, result)
            return Image.fromarray(result)
        except Exception as e:
            logging.error(f"Помилка при обробці: {str(e)}")
            return image if isinstance(image, Image.Image) else
Image.fromarray(img_array)

#[... Скорочено метод is_xray ...]

class ModelTrainer:
    def __init__(self, device=None, logs_callback=None):
        self.device = device or torch.device("cuda" if
torch.cuda.is_available() else "cpu")
        self.logs_callback = logs_callback or (lambda x:
logging.info(x))
        self.stop_flag = False
        self.training_history = {'train_loss': [], 'val_loss': [],
'train_acc': [], 'val_acc': [], 'learning_rates': []}
    def prepare_data(self, data_dir, validation_split=0.2,
test_split=0.1, max_samples=None, use_albumentation=True,
batch_size=8):
        fracture_dir = os.path.join(data_dir, "fracture")
        normal_dir = os.path.join(data_dir, "normal")
        fracture_files = [os.path.join(fracture_dir, f) for f in
os.listdir(fracture_dir)
                          if f.lower().endswith(('.png', '.jpg',
'.jpeg', '.bmp', '.tif', '.tiff'))]
        normal_files = [os.path.join(normal_dir, f) for f in
os.listdir(normal_dir)
                        if f.lower().endswith(('.png', '.jpg',
'.jpeg', '.bmp', '.tif', '.tiff'))]
        image_files = fracture_files + normal_files
        labels = [1] * len(fracture_files) + [0] * len(normal_files)
        train_files, temp_files, train_labels, temp_labels =
train_test_split(
            image_files, labels, test_size=(validation_split +
test_split), random_state=42
        )
        val_files, test_files, val_labels, test_labels =
train_test_split(
            temp_files, temp_labels, test_size=test_split /
(validation_split + test_split), random_state=42
        )
        train_dataset = XRayDatasetAdvanced(train_files,
train_labels,
transform=XRayDatasetAdvanced.get_transforms('train'))
        val_dataset = XRayDatasetAdvanced(val_files, val_labels,
transform=XRayDatasetAdvanced.get_transforms('val'))
        test_dataset = XRayDatasetAdvanced(test_files, test_labels,
transform=XRayDatasetAdvanced.get_transforms('val'))
        train_loader = DataLoader(train_dataset,
batch_size=batch_size, sampler=train_dataset.get_sampler())
```

```
        val_loader = DataLoader(val_dataset, batch_size=batch_size,
shuffle=False)
        test_loader           =           DataLoader(test_dataset,
batch_size=batch_size, shuffle=False)
        class_weights                                                =
torch.tensor([train_dataset.class_weights[0],
train_dataset.class_weights[1]], dtype=torch.float).to(self.device)
        return {'train': train_loader, 'val': val_loader, 'test':
test_loader}, class_weights, {}

#[...      Скорочено      методи      train_model,      save_model,
_detect_data_structure та інші допоміжні методи ...]

class FractureAnalysisApp:
    def __init__(self, master):
        self.master = master
        self.master.title("Аналіз   переломів   на   рентгенівських
знімках")
        self.device        =        torch.device("cuda"        if
torch.cuda.is_available() else "cpu")
        self.models = {}
        self.active_models = ["densenet121", "resnet50"]
        self.transform                                               =
XRayDatasetAdvanced.get_transforms(mode='val')
        self.model_trainer    =    ModelTrainer(device=self.device,
logs_callback=self.log)
        self.image_processor = ImageProcessor()
        self.setup_ui()
        self.load_config()
    def setup_ui(self):
        self.notebook = ttk.Notebook(self.master)
        self.notebook.pack(fill="both", expand=True)
        self.analysis_frame = ttk.Frame(self.notebook)
        self.notebook.add(self.analysis_frame,         text="Аналіз
зображень")
        tk.Button(self.analysis_frame,            text="Завантажити
зображення", command=self.load_image).pack()
        self.original_image_label = tk.Label(self.analysis_frame)
        self.original_image_label.pack()
        self.processed_image_label = tk.Label(self.analysis_frame)
        self.processed_image_label.pack()
        self.anomaly_label = tk.Label(self.analysis_frame)
        self.anomaly_label.pack()
        self.result_conclusion    =    tk.Label(self.analysis_frame,
text="")
        self.result_conclusion.pack()
        self.result_details     =     tk.Text(self.analysis_frame,
height=10, width=50)
        self.result_details.pack()
        self.training_frame = ttk.Frame(self.notebook)
        self.notebook.add(self.training_frame,       text="Навчання
моделі")
        self.data_dir_var = tk.StringVar(value=DATA_DIR)
```

```
        tk.Entry(self.training_frame,
textvariable=self.data_dir_var).pack()
        tk.Button(self.training_frame,  text="Обрати  директорію",
command=self.browse_data_dir).pack()
        self.start_training_button = tk.Button(self.training_frame,
text="Почати навчання", command=self.start_training)
        self.start_training_button.pack()
        self.stop_training_button  =  tk.Button(self.training_frame,
text="Зупинити       навчання",       command=self.stop_training,
state="disabled")
        self.stop_training_button.pack()
        self.progress_label     =     tk.Label(self.training_frame,
text="")
        self.progress_label.pack()
    def load_image(self):
        file_path                                                =
filedialog.askopenfilename(filetypes=[("Зображення",  "*.png  *.jpg
*.jpeg *.bmp *.tif *.tiff"), ("Всі файли", "*.*")])
        if file_path:
            image = Image.open(file_path).convert('RGB')
            self.current_image = image
            self.display_original_image(image)
            is_xray,         confidence,         details         =
self.image_processor.is_xray(image)
            if is_xray:
                self.preprocessed_image                          =
self.image_processor.preprocess_image(image)

self.display_processed_image(self.preprocessed_image)
                anomaly_score,      anomaly_vis,      anomaly_map,
anomaly_info                                                     =
self.image_processor.anomaly_detector.detect_anomalies(self.prepro
cessed_image)
                self.last_anomaly_score = anomaly_score
                self.last_anomaly_info = anomaly_info
                self.display_anomaly_image(anomaly_vis)
                results                                          =
self.analyze_image(self.preprocessed_image)
                self.display_results(results, None, anomaly_score,
anomaly_info)
            else:
                messagebox.showwarning("Попередження", f"Зображення
не є рентгенівським (впевненість: {confidence:.2f})")
    def analyze_image(self, image):
        image_tensor                                             =
self.transform(image).unsqueeze(0).to(self.device)
        if image_tensor.shape[1] != 3:
            image_tensor = image_tensor.repeat(1, 3, 1, 1)
        results = []
        for model_name in self.active_models:
            if model_name in self.models:
                model = self.models[model_name]
                with torch.no_grad():
```

```
                    output = model(image_tensor)
                    probabilities = torch.softmax(output, dim=1)
                    prob = probabilities[0, 1].item()
                    results.append((model_name, prob))
        return results
    def display_results(self, results, ensemble_score=None,
anomaly_score=0.0, anomaly_info=None):
        if not results:
            self.result_conclusion.config(text="Немає результатів
аналізу")
            self.result_details.delete("1.0", "end")
            return
        model_score = sum(prob for _, prob in results) / len(results)
        model_weight = 0.7
        anomaly_weight = 0.3
        combined_score = model_weight * model_score + anomaly_weight
* anomaly_score
        threshold = 0.5
        conclusion = "На знімку виявлено перелом" if combined_score
> threshold else "На знімку перелом не виявлено"
        details_text = f"Оцінка моделей: {model_score:.3f}\nОцінка
аномалій: {anomaly_score:.3f}\nКомбінована оцінка:
{combined_score:.3f}\n"
        for name, prob in results:
            details_text += f"- {name}: {prob:.3f}\n"
        self.result_conclusion.config(text=conclusion)
        self.result_details.delete("1.0", "end")
        self.result_details.insert("1.0", details_text)

#[... Скорочено методи для UI, збереження результатів,
інтерпретації, завантаження даних, логування та інші ...]

try:
    from albumentations.pytorch import ToTensorV2
except ImportError:
    class ToTensorV2:
        def __call__(self, image):
            return {'image': torch.from_numpy(image.transpose(2, 0,
1))}
        def __repr__(self):
            return "ToTensorV2()"

if __name__ == "__main__":
    root = tk.Tk()
    app = FractureAnalysisApp(root)
    root.protocol("WM_DELETE_WINDOW", app.on_closing)
    root.mainloop()
```