Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук

Бакалаврська кваліфікаційна робота на тему:

ПРОГРАМНИЙ МОДУЛЬ РОЗПІЗНАВАННЯ РАКУ ШКІРИ

Виконав: студент 4 курсу, групи ЗКН-21б
спеціальності 122 Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Трегуб А. К.
(прізвище та ініціали)

Керівник: к. т. н., доцент каф. КН
Арсенюк І. Р.
(прізвище та ініціали)
«04» червня 2025 р.

Рецензент: Ph. D., ст. викл. каф. САІТ
Войцеховська О. О.
(прізвище та ініціали)
«05» червня 2025 р.

Допущено до захисту Зав.
кафедри Яровий І. А.
«06» червня 2025 р.

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти перший (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 122 Комп'ютерні науки
Освітньо - професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри КН проф.,
д.т.н. Яровий А. А.
«05» квітня 2025 р

ЗАВДАННЯ

ПРОГРАМНИЙ МОДУЛЬ РОЗПІЗНАВАННЯ РАКУ ШКІРИ

Трегубу Артему Костянтиновичу

1. Тема роботи: Програмний модуль розпізнавання раку шкіри
2. Керівник роботи: Арсенюк Ігор Ростиславович, к.т.н., доц.

Затверджені наказом вищого навчального закладу "20" березня 2025 року №97

3. Термін подання студентом роботи 30 червня 2025р.
4. Вихідні дані до роботи:

Кількість виду раку – 7; кількість зображень не менше – 5000; розмір зображення фрагментів шкіри – 256×256 пікселів; розмір маски (контур ураженої ділянки шкіри) – 256×256 пікселів; формати зображень – jpg, png; мова програмування – об'єктно орієнтована; середовище розробки – хмарне.

Зміст текстової частини (перелік питань, які потрібно розробити):

проаналізувати існуючі технології та сучасні методи комп'ютерного зору, які застосовуються для розпізнавання зображень раку шкіри; розробити алгоритм роботи програмного модуля для розпізнавання раку шкіри; вибрати технологію проєктування та реалізації модуля для розпізнавання раку шкіри; спроєктувати та реалізувати програмний модуль розпізнавання раку шкіри; тестування програмного модуля

5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень): модель роботи програмного модуля; схема алгоритму роботи модуля для ідентифікації кольорових відтінків; діаграми класів та компонентів програмного забезпечення; приклад роботи програми.
6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-6	Арсенюк І. Р., к. т. н., доц. каф. КН		

7. Дата видачі завдання 05 травня 2025 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз предметної області задачі аналізу текстів на вміст образливих висловлювань	05.05.25	09.05.25	
2	Обґрунтування методу розв'язання задачі;	10.05.25	21.05.25	
3	Проектування інтелектуального модуля аналізу текстів на вміст образливих висловлювань	15.05.25	24.05.25	
4	Програмна реалізація інтелектуального модуля аналізу текстів на вміст образливих висловлювань	22.05.25	29.05.25	
5	Аналіз результатів тестування інтелектуального модуля аналізу текстів на вміст образливих висловлювань	27.05.25	03.06.25	
6	Розробка інструкції користувача	03.06.25	03.06.25	
7	Оформлення матеріалів до захисту БКР	04.06.25	04.06.25	

Студент _____ **Трегуб А. К.**
(підпис) *(прізвище та ініціали)*

Керівник роботи _____ **Арсенюк І. Р.**
(підпис) *(прізвище та ініціали)*

АНОТАЦІЯ

Бакалаврська кваліфікаційна робота складається з 75 сторінок формату А4, на яких є 12 рисунки, список використаних джерел містить 25 найменувань.

Бакалаврська кваліфікаційна робота є частиною дослідження, присвяченого використанню сучасних технологій штучного інтелекту для покращення процесу медичної діагностики. У роботі розглянуто підхід щодо розпізнавання раку шкіри за допомогою згорткової нейронної мережі, а також виконано розробку та навчання моделі, здатної класифікувати дерматологічні зображення як доброякісні або злоякісні. У процесі реалізації було підібрано архітектуру нейронної мережі, налаштовано її параметри та проведено тестування точності класифікації з використанням відкритих медичних датасетів.

Ключові слова: нейронна мережа, згорткова нейронна мережа, рак шкіри, штучний інтелект, діагностика, медична інформатика.

ABSTRACT

The bachelor's qualification work consists of 75 pages of A4 format, on which there are 12 figures, the list of used sources contains 25 names.

The bachelor's qualification work is part of a study dedicated to the use of modern artificial intelligence technologies to improve the process of medical diagnostics. The work considers an approach to recognizing skin cancer using a convolutional neural network, and also develops and trains a model capable of classifying dermatological images as benign or malignant. During the implementation process, the architecture of the neural network was selected, its parameters were adjusted, and the classification accuracy was tested using open medical datasets.

Keywords: neural network, convolutional neural network, skin cancer, artificial intelligence, diagnostics, medical informatics.

ЗМІСТ

ВСТУП

Актуальність досліджень. Рак шкіри є одним із найпоширеніших онкологічних захворювань у світі, при цьому його рання діагностика має вирішальне значення для успішного лікування та зниження рівня смертності. Найнебезпечнішою формою є меланома — агресивне новоутворення, яке швидко метастазує і призводить до летальних наслідків у разі несвоєчасного виявлення. У зв'язку з цим, своєчасна та точна діагностика раку шкіри є критично важливою, оскільки на ранніх стадіях хвороба піддається лікуванню у більшості випадків. Традиційні методи діагностики залежать від візуального огляду лікаря-дерматолога та додаткових клінічних обстежень, що потребує значних ресурсів і може бути недоступним у віддалених або слабо розвинених регіонах. У зв'язку з цим зростає потреба в ефективних автоматизованих інструментах, здатних допомогти в первинному аналізі дерматологічних зображень. Згорткові нейронні мережі демонструють високі результати в задачах комп'ютерного зору, зокрема у розпізнаванні медичних зображень, що робить їх перспективним інструментом у сфері діагностики раку шкіри. Однак, попри успіхи сучасних систем, точність розпізнавання складних і атипових випадків новоутворень все ще залишається недостатньою. З вищенаведеного можна зробити висновок, що проблема аналізу зображень раку шкіри є актуальною та потребує подальшого дослідження.

Метою дослідження є підвищення точності розпізнавання зображень раку шкіри.

Об'єктом дослідження є процес ідентифікації зображень раку шкіри.

Предметом дослідження є програмний модуль розпізнавання раку шкіри.

Задачі дослідження:

1) проаналізувати існуючі рішення для ідентифікації зображень раку шкіри;
2) обґрунтувати підхід до розв'язання задачі розпізнавання

дерматологічних зображень шкіри та розробити відповідний алгоритм;

3) здійснити проєктування програмного модуля розпізнавання раку шкіри;
4) виконати програмну реалізацію модуля розпізнавання раку шкіри;
5) виконати тестування програмного модуля розпізнавання раку шкіри та здійснити аналіз результатів;

Апробація роботи. Робота апробувалася на конференції «LIV Всеукраїнська науково-технічна конференція факультету інтелектуальнихінформаційних технологій та автоматизації (2025)» у м. Вінниці [1].

Публікації. На науково-технічній конференції ВНТУ було опубліковано тези доповіді на тему «Перспективи розробки WEB-ресурсу підбору музичних інструментів» [1].

1 ОБҐРУНТУВАННЯ ДОЦІЛЬНОСТІ РОЗРОБКИ

1.1 Аналіз предметної області розпізнавання раку шкіри

На сьогоднішній день проблема раннього виявлення онкологічних захворювань є однією з найактуальніших у медицині. Особливе місце серед них займає рак шкіри — патологія, яка займає провідні позиції за темпами поширення у світі. Щорічно кількість випадків захворювання стрімко зростає, а ефективність лікування значною мірою залежить від стадії, на якій виявлено хворобу. Враховуючи візуальний характер проявів більшості новоутворень на шкірі, одним із перспективних напрямів сучасної медицини стало створення автоматизованих систем для розпізнавання та класифікації шкірних патологій за зображеннями [2].

Протягом тривалого часу діагностика шкірних новоутворень здійснювалася переважно за допомогою візуального огляду та дерматоскопії [2]. У основі цих методів — візуальна оцінка форми, кольору, розміру та меж новоутворення. Однак навіть досвідчені лікарі можуть стикатися з труднощами, оскільки деякі злоякісні пухлини мають зовнішню схожість із доброякісними ураженнями. Це призводить до необхідності проведення біопсії або додаткових обстежень, які є інвазивними та потребують часу. Із розвитком цифрових технологій виникла ідея автоматизувати цей процес за допомогою комп’ютерного аналізу зображень. Початково для цього застосовували традиційні методи комп’ютерного зору, такі як сегментація, фільтрація та виявлення контурів. Однак через високу варіативність форм та кольорів шкірних новоутворень, а також присутність перешкод (волосся, нерівномірне освітлення, артефакти) класичні підходи виявилися недостатньо ефективними для вирішення такого завдання.

Розпізнавання раку шкіри за допомогою комп’ютерних систем стало актуальним напрямом у сфері медичної діагностики, оскільки своєчасне

виявлення злоякісних новоутворень значно підвищує шанси на успішне лікування. Завдяки стрімкому розвитку технологій штучного інтелекту та глибокого навчання з'явилася можливість створювати автоматизовані системи, які здатні аналізувати медичні зображення, виявляти патології та допомагати лікарям у постановці діагнозів. Особливе місце серед таких методів займають нейронні мережі — математичні моделі, натхненні принципами роботи людського мозку, які дозволяють знаходити приховані залежності в складних даних [3-5].

1.2 Огляд та аналіз аналогів

Розглянемо деякі з існуючих рішень розпізнавання раку шкіри. Першим прикладом є модель на основі DenseNet [6]. Вона базується на використанні нейронної мережі DenseNet201 для задачі класифікації зображень шкірних уражень. Автор моделі застосував попередньо навчений DenseNet, адаптувавши його до набору даних HAM10000 [7]. Модель демонструє середню точність класифікації на рівні 87,53%, при цьому точність для окремих класів варіюється від 79% до 98%. Основними недоліки підходу є:

- модель не виконує сегментацію, тобто не визначає конкретні межі уражень;
- відсутність локалізації патологічної області знижує застосовність у клінічній діагностиці.

Друге рішення — модель «Skin Cancer Segmentation TensorFlow 94%», орієнтована саме на сегментацію уражень [8]. Вона побудована на архітектурі U-Net у середовищі TensorFlow. Модель досягає тестової точності 93,70%, precision 85,88%, recall 93,96%, Dice коефіцієнта 85,36%. Попри високу recall, що свідчить про ефективне виявлення уражень, значення precision залишається на рівні ~86%, що означає наявність значної кількості хибно-позитивних результатів. У

медицині такі помилки можуть призводити до необґрунтованих діагностичних або терапевтичних втручань.

Таблиця 1.1 – Порівняльна характеристика існуючих аналогів розпізнавання раку шкіри

Параметр	DenseNet модель	U-Net модель
Тип задачі	Класифікація	Сегментація
Accuracy	87,53%	93,70%
Precision	~88%	85,88%
Recall	~88%	93,96%
Dice коефіцієнт	—	85,36%
Локалізація уражень	Ні	Так
Використана архітектура	DenseNet201	U-Net
Платформа	TensorFlow	TensorFlow

На основі проведеного аналізу можна зробити висновок, що жодна з розглянутих моделей не забезпечує оптимальне рішення для задачі високоточного та надійного розпізнавання шкірних новоутворень у форматі, який би одночасно забезпечував високу точність сегментації і мінімізацію помилкових результатів. Модель на базі DenseNet орієнтована виключно на класифікацію зображень і не виконує просторової локалізації уражень, що суттєво обмежує її прикладне використання в клінічних умовах. Натомість модель, побудована на архітектурі типу U-Net, орієнтована на виконання задачі сегментації, однак характеризується недостатнім рівнем точності при виявленні істинно позитивних випадків, що створює ризик ухвалення некоректних рішень у процесі медичної діагностики. У зв’язку з цим виникає необхідність розробки удосконаленої моделі, яка б забезпечувала високу достовірність сегментації, покращені показники виявлення справжніх позитивів і зменшення кількості хибнопозитивних та хибнопозитивних результатів, а також була б адаптованою до практичного впровадження в системах комп’ютерної діагностики шкірних

захворювань. Це формує основну задачу даної роботи – розробку ефективного рішення на основі нейронних мереж, яке забезпечить більш точне, надійне і практично застосовне автоматизоване розпізнавання шкірних уражень.

1.3 Постановка задачі дослідження

Поставимо основну задачу та сформулюємо вимоги до програмного модуля, призначеного для автоматизованого аналізу дерматологічних зображень з метою виявлення ознак раку шкіри. Задача: створення програмного модуля, що забезпечує збільшену точність розпізнавання раку шкіри на медичних зображеннях, що надходять від користувача, та формує карту ймовірностей, яка дозволяє провести попередній аналіз стану шкіри для подальшого клінічного використання.

Програмний модуль має реалізовувати такі функції:

- завантаження зображень шкіри користувачем у зручному форматі;
- попередня обробка вхідних зображень;
- сегментація зображення із виділенням імовірної патологічної ділянки;
- виведення результату у вигляді візуалізованої маски поверх зображення та цифрових метрик точності;
- можливість отримання користувачем порогових значень, що визначають наявність чи відсутність підозрілих ознак;
- збереження результатів обробки у зручному форматі для подальшого аналізу.

Також необхідно провести всебічне тестування розробленого програмного модуля на предмет коректності обробки зображень, відсутності помилок, стабільності роботи та точності результатів сегментації. Програмне забезпечення має бути сумісним із сучасними операційними системами та забезпечувати зручний інтерфейс для взаємодії з користувачем. Усі компоненти модуля

повинні функціонувати узгоджено, без збоїв або непередбачуваної поведінки під час роботи з реальними клінічними зображеннями.

Початковим етапом розробки модуля розпізнавання раку шкіри є аналіз підходів до розпізнавання медичних зображень раку шкіри. Далі здійснюється побудова загальної структурної схеми програмного модуля, формулювання алгоритму його роботи та створення UML-діаграм (use case, activity, sequence, class), які деталізують логіку реалізації. На наступному етапі обираються мови програмування, бібліотеки, після чого виконується безпосередня реалізація функціоналу модуля. Завершальними етапами є тестування моделі на контрольному наборі зображень, оцінка її ефективності та створення документації користувача.

1.4 Висновок до розділу 1

Наведено обґрунтування актуальності розробки програмного модуля автоматизованого розпізнавання раку шкіри.

Здійснено аналіз предметної області, зокрема розглянуто основні методи діагностики шкірних новоутворень та підкреслено їхні недоліки, пов’язані з суб’єктивністю оцінки та обмеженою доступністю в окремих регіонах.

Здійснено огляд та аналіз аналогів програмного модуля розпізнавання шкіри. Визначено їхні особливості та недоліки. Огляд показав, що модель на основі Dense Net хороші результати класифікації, однак не виконує сегментацію, що обмежує її застосування в медичній практиці. Модель U-Net орієнтована на сегментацію, проте має недостатній рівень точності, що може спричинити хибнопозитивні результати. Зазначено необхідність підвищення точності розпізнавання раку шкіри існуючих рішень.

Здійснено детально постановку задачі розробки програмного модуля розпізнавання раку шкіри. Окреслено основні вимоги до розроблюваного модуля.

2 РОЗРОБКА ПІДХОДУ ЩОДО РОЗПІЗНАННЯ РАКУ ШКІРИ

2.1 Варіантний аналіз шляхів розв’язання поставленої задачі

Для автоматизованого розпізнавання раку шкіри можуть бути застосовані різні підходи з галузі інтелектуального аналізу зображень. У цьому підрозділі розглянемо три принципово різні напрями: методи, що базуються на ознаковому машинному навчанні, неконтрольованому навчанні (кластеризації), а також нейронні мережі. Кожен із підходів має свої сильні сторони й обмеження, що визначає їхню придатність до поставленої задачі.

Перший підхід це – ознакове машинне навчання, передбачає ручне формування векторів ознак з медичних зображень, таких як колір, текстура, форма чи геометричні характеристики уражень. Отримані ознаки подаються на вхід класичним алгоритмам, наприклад, методам опорних векторів, логістичній регресії або деревам рішень. Основною перевагою підходу є його інтерпретованість, порівняно невисокі обчислювальні вимоги та можливість навчання на обмежених обсягах даних. Проте ефективність такого методу значною мірою залежить від якості попереднього виділення ознак. У випадках складних або варіативних медичних зображень ручне виділення ознак виявляється неефективним або неточним. Крім того, модель не має змоги враховувати просторові залежності між пікселями зображення, що критично важливо для точного визначення меж патологічних утворень [9].

Другий підхід базується на методах неконтрольованого навчання. В основі цих алгоритмів лежить кластеризація схожих зразків без наявності розмічених даних. Найпоширенішими методами є k-середніх, ієрархічна кластеризація, а також аналіз головних компонент (PCA) для зменшення розмірності. Перевага неконтрольованих методів полягає у можливості виявлення прихованих структур у даних без необхідності вручну маркувати зображення. Вони є корисними на початкових етапах аналізу або як інструмент попередньої обробки.

Однак у задачах медичної діагностики, де критично важлива точність і достовірність, цей підхід виявляється недостатнім. Результати кластеризації складно інтерпретувати, вони можуть бути чутливими до вибору початкових умов, а відсутність контрольних прикладів унеможливлює оцінку якості класифікації. Також неконтрольовані алгоритми не можуть виконувати сегментацію із точною локалізацією патологій.

Третій підхід – глибоке навчання на основі штучних нейронних мереж, який є найсучаснішим і найперспективнішим. Він базується на використанні багаторівневих моделей, здатних автоматично виявляти складні патерни та закономірності в медичних зображеннях. До переваг нейронних мереж належать: відсутність необхідності в ручному виділенні ознак, висока точність навіть у випадках складних або зашумлених даних, здатність до масштабування, а також можливість узагальнення на нові приклади. Основним недоліком є висока вимога до обчислювальних ресурсів і тривалий процес навчання [10].

Порівняльний аналіз засвідчує, що методи, засновані на нейронних мережа, мають найвищий потенціал для вирішення задачі автоматизованого розпізнавання раку шкіри. Саме вони здатні забезпечити водночас точну класифікацію, просторову локалізацію уражень та стійкість до варіативності вхідних зображень. У зв’язку з цим у рамках даної роботи буде реалізовано програмний модуль, заснований на архітектурі нейронної мережі, що дозволить досягти високої достовірності в автоматизованому виявленні злоякісних утворень шкіри.

2.2 Обґрунтування вибору архітектури нейронної мережі

Основною задачею в автоматизованому розпізнаванню зображень шкіри є точне виділення меж уражених ділянок та класифікація типу новоутворень. Для вирішення подібних задач найбільш ефективними виявили себе згорткові нейронні мережі, які забезпечують високу точність у задачах комп’ютерного

зору. Особливістю задачі розпізнавання раку шкіри є необхідність не лише класифікації зображення як такого, але й точного локалізованого аналізу — виявлення меж патологічних зон для подальшого прийняття клінічних рішень. У контексті завдання розпізнавання та покращення точності розпізнавання раку шкіри було розглянуто основні архітектури нейронних мереж, серед яких згорткові нейронні мережі (Convolutional Neural Networks, CNN), рекурентні нейронні мережі (Recurrent Neural Networks, RNN), трансформер-архітектури (Vision Transformers, ViT), U-Net, ResNet, DenseNet, EfficientNet.

CNN є найбільш популярним класом моделей для обробки зображень, зокрема медичних. Їхня ключова особливість полягає в здатності автоматично виділяти просторові ознаки на різних рівнях абстракції. Це досягається шляхом застосування згорткових шарів, які проходять по зображенню із фільтрами (ядрами згортки) і виявляють характерні шаблони, наприклад краї, кути або текстури [9]. На рисунку 1.1 вказана архітектура згорткової нейронної мережі AlexNet.

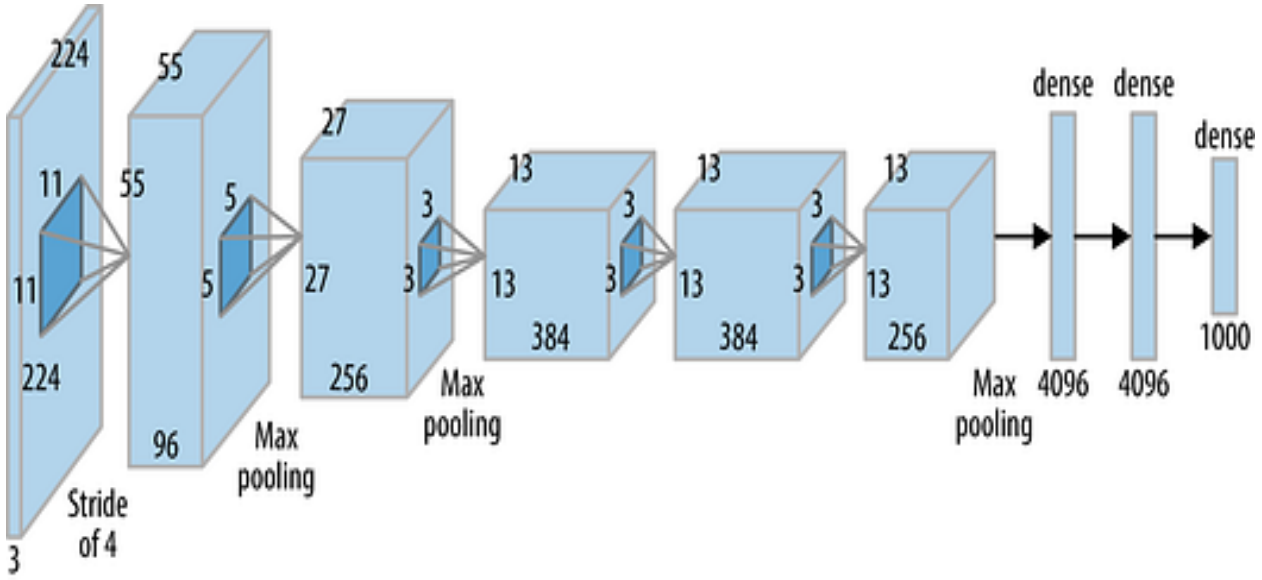

Рисунок 2.1 – Архітектура згорткової нейронної мережі AlexNet

Основна перевага CNN полягає в тому, що такі мережі можуть навчатися розпізнавати об’єкти різних розмірів і форм, що особливо важливо у медичних задачах, де новоутворення можуть бути різноманітними за своєю структурою, розташуванням і кольором. На відміну від класичних методів обробки зображень, CNN здатні самостійно формувати оптимальні ознаки для подальшої класифікації без необхідності попереднього ручного виділення особливостей. До обмежень CNN є вимога до значних обсягів даних для навчання та обчислювальних ресурсів. Крім того, такі моделі чутливі до аномалій у даних — шуму, варіацій освітлення та контрастності. Для підвищення надійності моделі часто використовують методи аугментації, які дозволяють створити додаткові варіанти зображень шляхом їх обертання, масштабування або дзеркального відображення. Ще однією проблемою CNN є обмежена здатність працювати з просторовими залежностями на великих відстанях у межах зображення, оскільки обчислення проводяться в локальних областях [9-11].

RNN (рекурентна нейронна мережа) — це тип нейронної мережі, яка на відміну від CNN, які працюють із просторовими даними, RNN призначені для обробки послідовних даних. Це мережі, в яких інформація передається від одного стану до іншого по часовій осі, що дозволяє враховувати контекст попередніх елементів послідовності. У медичній практиці RNN застосовуються для аналізу серій зображень, відео або динамічних даних пацієнтів. До недоліків RNN відносять складність навчання та обмежену здатність обробляти великі обсяги даних. Крім того, через послідовний характер обробки інформації, такі мережі мають повільніший час обробки порівняно з CNN. Також RNN схильні до проблем затухання або вибуху градієнта під час навчання, що ускладнює оптимізацію глибоких мереж [9].

Останніми роками трансформери набрали ширшого використання у галузі обробки послідовностей і поступово інтегруються в комп’ютерний зір. Їхня головна перевага – використання механізму самоуваги (self-attention), який дозволяє моделі визначати важливість різних ділянок зображення незалежно від

їхнього положення. Це особливо цінно в медичних зображеннях, де патології можуть бути розмитими, нечіткими та розташованими в непередбачуваних місцях. Трансформери демонструють високу продуктивність на великих наборах даних, що робить їх придатними для аналізу зображень високої роздільної здатності [14].

Vision Transformer (ViT) – це архітектура глибокого навчання, яка переносить принципи трансформерів, спочатку розроблених для обробки природної мови, у сферу комп'ютерного зору [9-11]. Особливістю від класичних згорткових нейронних мереж (CNN), ViT обробляє зображення як послідовність патчів (фрагментів), а не як суцільну матрицю пікселів. Такий підхід дозволяє моделі будувати глобальні представлення ознак, що критично важливо для точного аналізу складних візуальних структур. Разом із тим, трансформери мають значні недоліки. Вони потребують великих обчислювальних ресурсів для навчання та схильні до перенавчання на малих вибірках. Тому для роботи з обмеженими медичними даними їх часто комбінують із попередньо натренованими моделями або застосовують методи регуляризації. Попри складність, Vision Transformer демонструє високий потенціал у завданнях медичної діагностики, зокрема при аналізі дерматологічних зображень, де важливо враховувати як локальні, так і глобальні залежності між фрагментами шкіри. На відміну від CNN, які зосереджуються переважно на локальних шаблонах, ViT здатен ефективно моделювати взаємозв'язки між віддаленими ділянками зображення, що сприяє точнішій ідентифікації патологій. У контексті задач розпізнавання раку шкіри це дозволяє виявляти як чітко виражені, так і слабо помітні ознаки новоутворень. У поєднанні з аугментацією даних, transfer learning або гібридними підходами (наприклад, CNN + Transformer) ViT може стати ефективним інструментом для клінічного застосування в умовах обмеженого набору зображень. Ключовим структурним елементом архітектури ViT є блок трансформера, який складається з механізму багатоголового самоуваги (Multi-Head Self-Attention) та багаторівневого перцептрону (MLP), що

дозволяє моделі ефективно виявляти складні взаємозв’язки між патчами зображення. Загальна структура роботи Vision Transformer наведена на рисунку 2.2.

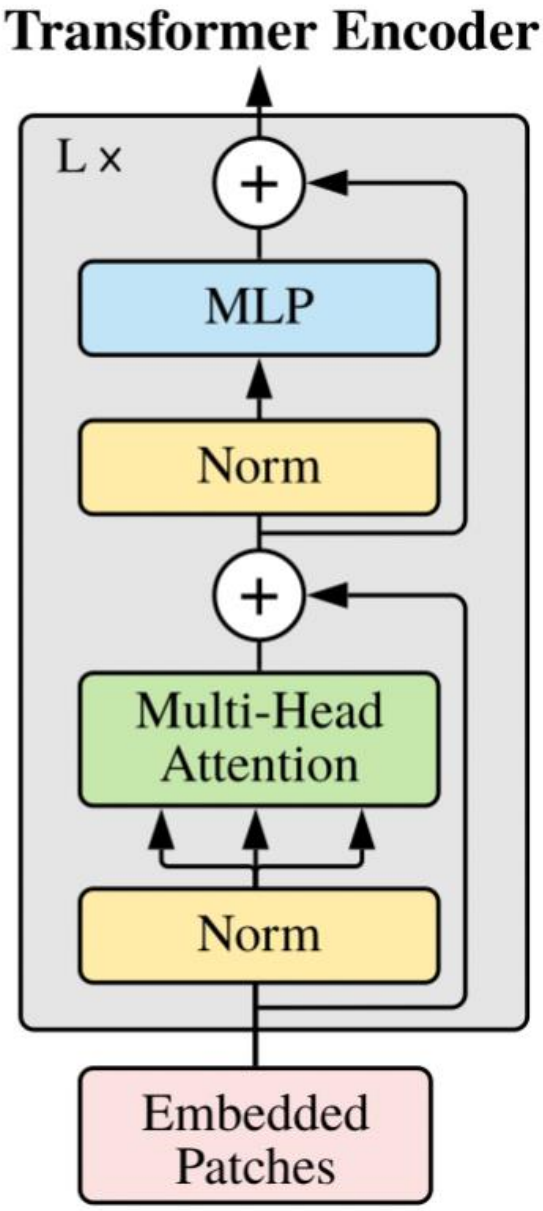

Рисунок 2.2 – Блок трансформера в ViT

Гнучкість архітектури ViT полягає також у її здатності до масштабування: вона легко адаптується до різних розмірів зображень, кількості патчів та глибини моделі. Крім того, ця архітектура успішно використовується не лише для

класифікації, а й для задач сегментації, об’єктного детектування, реставрації зображень та інших напрямів комп’ютерного зору. На великих наборах даних, зокрема ImageNet-21k [13], модель Vision Transformer демонструє точність, яка перевищує показники сучасних згорткових нейронних мереж, особливо у високорівневих задачах розпізнавання. Проте для досягнення таких результатів необхідно забезпечити значну обчислювальну потужність та доступ до великих обсягів якісних розмічених даних.

U-Net — це згорткова нейронна мережа спеціально розроблена для задач семантичної сегментації. Структурно вона складається з енкодера, який послідовно зменшує просторові розміри зображення, та декодера, що відновлює його роздільність, створюючи карту сегментації. Модель була описана Олафом Роннебергером, Філіпсом Фішером та Томасом Броксом у 2015 році у своїй роботі «UNet: Convolutional Networks for Biomedical Image Segmentation», ставши поліпшенням та подальшим розвитком архітектури FCN [15-16]. U-Net була запропонована для задач біомедичної сегментації та отримала широке застосування у медицині завдяки своїй здатності працювати з невеликими наборами даних, що є характерною умовою для задач медичної діагностики. Особливістю архітектури U-Net є наявність симетричної структури, яка складається з двох основних частин: енкодера (скорочення розмірності зображення з витягуванням суттєвих ознак) та декодера (відновлення просторових розмірів із поступовим збагаченням карти ознак). Енкодер виконує послідовні операції згортки та підвибірки, що дозволяє отримати компактне представлення вхідного зображення, зберігаючи найбільш значущі ознаки. Декодер, у свою чергу, за допомогою транспонованих згорток поступово відновлює розмірність зображення до початкової, інтегруючи додаткову інформацію з енкодера через механізм пропускових з’єднань. Механізм пропускових з’єднань дозволяє уникнути втрати важливої просторової інформації на етапах глибокої згортки, передаючи дані з відповідних рівнів енкодера безпосередньо на відповідні рівні декодера. Це критично важливо для

медичних задач, де точність визначення меж патологічних зон є вирішальною для подальшої діагностики. На рисунку 1.2 наведена архітектура U-Net для сегментації зображення.

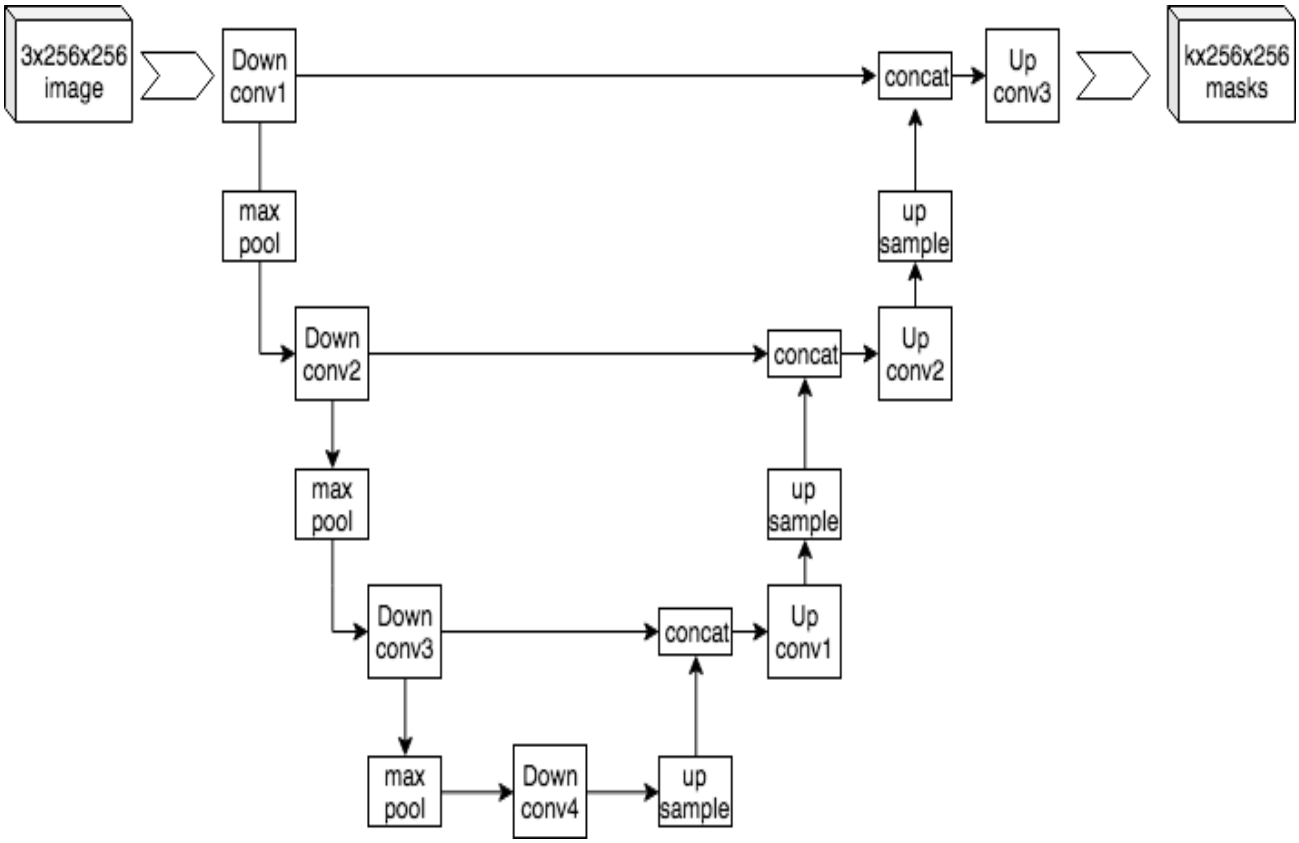

Рисунок 2.3 — Архітектура U-Net для сегментації зображення

Особливістю U-Net є наявність пропускних з’єднань між відповідними шарами енкодера та декодера, що забезпечує збереження просторової інформації та покращує точність виділення меж об’єктів. Це дозволяє моделі ефективно працювати навіть із обмеженими наборами медичних зображень. Попри високу продуктивність, U-Net має свої обмеження. Зокрема, модель може погано працювати з дуже великими зображеннями через обмеження пам’яті графічного процесора. Крім того, складні структури патологій можуть потребувати додаткового постпроцесингу для усунення помилкових класифікацій [12].

ResNet (Residual Network) — це глибока згорткова нейронна мережа, яка впроваджує залишкові (residual) з'єднання з метою полегшення процесу навчання дуже глибоких моделей. Головна ідея полягає у впровадженні shortcut-з'єднань, які дозволяють передавати сигнал та градієнти в обхід кількох шарів без їх модифікації. Такий підхід ефективно усуває проблему затухання градієнтів, що зазвичай виникає при навчанні глибоких мереж, і відкриває можливість тренування моделей з сотнями та навіть тисячами шарів [17].

Ключовим елементом архітектури ResNet є залишковий блок (residual block). Він зазвичай містить два або більше згорткових шарів, між якими реалізовані операції нормалізації за міні-пакетами (batch normalization) та нелінійності за допомогою функції ReLU. Особливість блоку полягає в тому, що вхідний сигнал додається до виходу блоку — це так зване shortcut-з'єднання. У результаті, мережа не вчить саму функцію перетворення, а лише різницю (residual) між вхідними та вихідними ознаками, що робить процес оптимізації більш стабільним. Архітектура ResNet добре масштабована: існують моделі різної глибини, такі як ResNet-50, ResNet-101, ResNet-152, які відрізняються кількістю залишкових блоків. Для зменшення обчислювальної складності в глибших моделях застосовуються bottleneck-блоки, які включають згортки 1×1 для зменшення розмірності ознак перед виконанням обчислювально важчих 3×3 згорток. Це дозволяє зменшити кількість параметрів і швидше тренувати мережу без втрати якості.

DenseNet (Densely Connected Convolutional Network) – це архітектура згорткової нейронної мережі, в якій реалізовано прямі з'єднання між кожним шаром і всіма попередніми шарами в межах одного блоку. Це означає, що кожен шар отримує на вхід не лише результат попереднього шару, а й виходи всіх шарів, що передували йому. Така стратегія сприяє максимально ефективному повторному використанню ознак. Ці щільні з'єднання (dense connections) забезпечують ефективний потік інформації та градієнтів по всій мережі, що зменшує ймовірність переобчислення однакових ознак у різних шарах. У межах

кожного dense block накопичуються ознаки, що збагачують представлення вхідних даних. Для контролю зростання розміру ознак між dense blocks застосовуються перехідні шари (transition layers), які містять 1×1 згортки для зменшення кількості каналів та операції pooling (зазвичай 2×2) для зменшення просторових розмірів. Ще однією ключовою характеристикою є параметр growth rate (k), який визначає, скільки нових ознак додається на кожному новому шарі. Цей параметр впливає на розмірність представлень та на обчислювальні витрати. DenseNet демонструє високу ефективність та стабільність навіть на відносно невеликих вибірках, а також добре підходить для задач класифікації та сегментації в медичних зображеннях [18].

EfficientNet – це сімейство моделей згорткових нейронних мереж, яке вирізняється застосуванням принципу збалансованого масштабування (compound scaling). Замість того, щоб збільшувати лише один з архітектурних параметрів (глибину, ширину або роздільну здатність вхідного зображення), EfficientNet одночасно масштабує всі три параметри за спеціально підібраною формулою. Це забезпечує оптимальний баланс між точністю, продуктивністю та обчислювальними витратами.

Архітектура EfficientNet базується на MBConv-блоках — мобільних згорткових блоках, які поєднують глибокі згортки (depthwise separable convolutions) для зменшення обчислень і SE-механізми (squeeze-and-excitation) для динамічного зважування важливості каналів. Ці SE-блоки дозволяють моделі фокусуватись на найбільш інформативних ознаках, адаптуючи активацію каналів до контексту кожного зображення. Результати численних експериментів показують, що EfficientNet досягає вищої точності при меншій кількості параметрів і обчислювальних операцій (FLOPs) у порівнянні з ResNet і DenseNet. Це робить її однією з найефективніших архітектур для застосування в задачах комп’ютерного зору з обмеженими ресурсами, зокрема на мобільних пристроях або в умовах хмарної інфраструктури з лімітованою обчислювальною потужністю [19].

У таблиці 2.1 наведено порівняльна характеристика різних архітектур нейронних мереж.

Таблиця 2.1 – Порівняльна характеристика різних архітектур нейронних мереж

Тип архітектури	Переваги	Недоліки
RNN (Recurrent Neural Network)	– ефективна в роботі з часовими рядами; – підходить для моделювання послідовностей; – може запам’ятовувати інформацію про попередній контекст.	– не здатна ефективно аналізувати просторові залежності у зображеннях; – складне навчання через затухання або вибух градієнтів; – низька точність у задачах класифікації медичних зображень.
CNN (Convolutional Neural Network)	– орієнтована на обробку зображень; – виявляє локальні ознаки за допомогою згорток; – зберігає просторову структуру зображення; – менше параметрів порівняно з FCNN; – масштабується для сегментаційних та класифікаційних задач.	– потребує великого обсягу навчальних даних; – вимагає значних обчислювальних ресурсів для навчання; – погано пояснює прийняті рішення без додаткових методів інтерпретації.
ViT (Vision Transformer)	– глобальний контекст через self-attention; – висока точність у задачах класифікації на великих наборах;	– потребує величезних обсягів даних для якісного навчання; – високі обчислювальні витрати;

Продовження табл. 2.1

Тип архітектури	Переваги	Недоліки
U-Net	– спеціально адаптована до задач сегментації; – ефективна при малій кількості даних; – зберігає просторову інформацію через пропускові з'єднання; – добре підходить для медичних зображень.	– складна архітектура з великою кількістю параметрів; – важче масштабувати для класифікаційних задач; – може перенавчитись на малих вибірках без регуляризації.
ResNet	– ефективна для глибоких моделей за рахунок residual-зв'язків; – покращене поширення градієнта; – хороша точність при класифікації зображень.	– складніша реалізація; – все ще потребує великого обсягу даних для тренування; – не призначена для сегментації без модифікацій.
DenseNet	– покращене повторне використання ознак завдяки щільним зв'язкам; – зменшує кількість параметрів; – хороша ефективність при класифікації.	– повільне навчання через велику кількість з'єднань; – надлишковість інформації між шарами при великій глибині; – потребує значної пам'яті.
EfficientNet	– оптимальний баланс між точністю та ефективністю; – масштабована архітектура (від B0 до B7); – менше параметрів за аналогічної точності.	– складна пошарова побудова; – гірша інтерпретованість; – може вимагати складного тюнінгу при специфічних задачах.

З огляду на поставлену задачу найкращою архітектурою для підвищеня точності розпізнавання раку шкіри вибір саме U-Net архітектури є доцільним з огляду на її перевірену ефективність у медичних задачах сегментації, гнучкість

налаштувань, здатність працювати з обмеженими наборами даних та забезпечення високої точності при прийнятних обчислювальних витратах.

- висока точність сегментації на невеликих та атипових вибірках медичних зображень;
- збереження просторових характеристик об’єкта завдяки використанню пропускові з’єднання;
- гнучкість налаштування кількості рівнів абстракції та ширини мережі кількості фільтрів на кожному рівні;
- оптимальний баланс між складністю архітектури та необхідними обчислювальними ресурсами.

2.3 Розробка структурної схеми програмного модуля

Розробимо загальну структурну схему програмного модуля розпізнавання раку шкіри.

Розробимо загальну структурну схему програмного модуля розпізнавання раку шкіри. Програмний модуль складається з шести основних компонентів:

- Модуль підготовки даних
- Модуль візуалізації даних
- Модуль формування наборів даних
- Модуль побудови та навчання моделі
- Модуль оцінки якості моделі
- Модуль збереження результатів

Опишемо кожен із компонентів.

Модуль підготовки даних відповідає за завантаження зображень та відповідних масок із датасету, перевірку їх відповідності, а також виключення неповних або некоректних зразків. На цьому етапі здійснюється попередня

обробка зображень: масштабування, нормалізація, приведення до єдиного розміру.

Модуль візуалізації даних забезпечує можливість перегляду прикладів зображень та масок, а також їх накладання для візуального контролю якості даних. Це дозволяє переконатися у коректності підготовки даних перед навчанням моделі.

Модуль формування наборів даних виконує розподіл даних на тренувальний, валідаційний та тестовий набори із заданими пропорціями для найбільшої точності. Також формує пайплайни для ефективного зчитування та обробки даних у процесі навчання моделі.

Модуль побудови та навчання моделі реалізує архітектуру сегментаційної нейронної мережі, визначає функцію втрат, метрики якості, а також організовує процес навчання з використанням оптимізатора Adam, callback-функцій для збереження найкращих ваг та динамічного зменшення швидкості навчання.

Модуль оцінки якості моделі здійснює оцінку навченої моделі на тестовому наборі даних за основними метриками (точність, повнота, Dice-коефіцієнт). Також забезпечує візуалізацію динаміки навчання (графіки втрат, точності, Dice-коефіцієнта) та візуалізацію результатів сегментації на тестових зображеннях.

Модуль збереження результатів використовується для збереження навченої моделі, ваг, а також результатів оцінки та візуалізацій, що дозволяє повторно використовувати або аналізувати результати в майбутньому. Основна структурна схема роботи програмного модуля розпізнавання раку шкіри наведена на рисунку 2.4

Рисунок 2.4 – Основна структурна схема роботи програмного модуля ропізнавання раку шкіри

2.4 Розробка загального алгоритму роботи програмного модуля

Для відображення циклу роботи програмного забезпечення використаємо текстовий та графічний опис загального алгоритму програмного модуля розпізнавання раку шкіри.

Кроки алгоритму:

1. Початок
2. Завантаження зображень і масок з датасету
3. Перевірка відповідності масок зображенням
4. Чи є відповідні маски? Якщо так — перейти до кроку 7. Якщо ні — перейти до кроку 5.
5. Виключити неповні зразки
6. Продовжити
7. Візуалізація прикладів зображень і масок
8. Формування наборів даних (train/val/test)
9. Попередня обробка зображень
10. Побудова моделі (U-net)
11. Компіляція моделі
12. Навчання моделі
13. Збереження навчальної моделі.
14. Алгоритм завершує роботу.

Графічна інтерпретація описаного алгоритму наведена на рисунку 2.1:

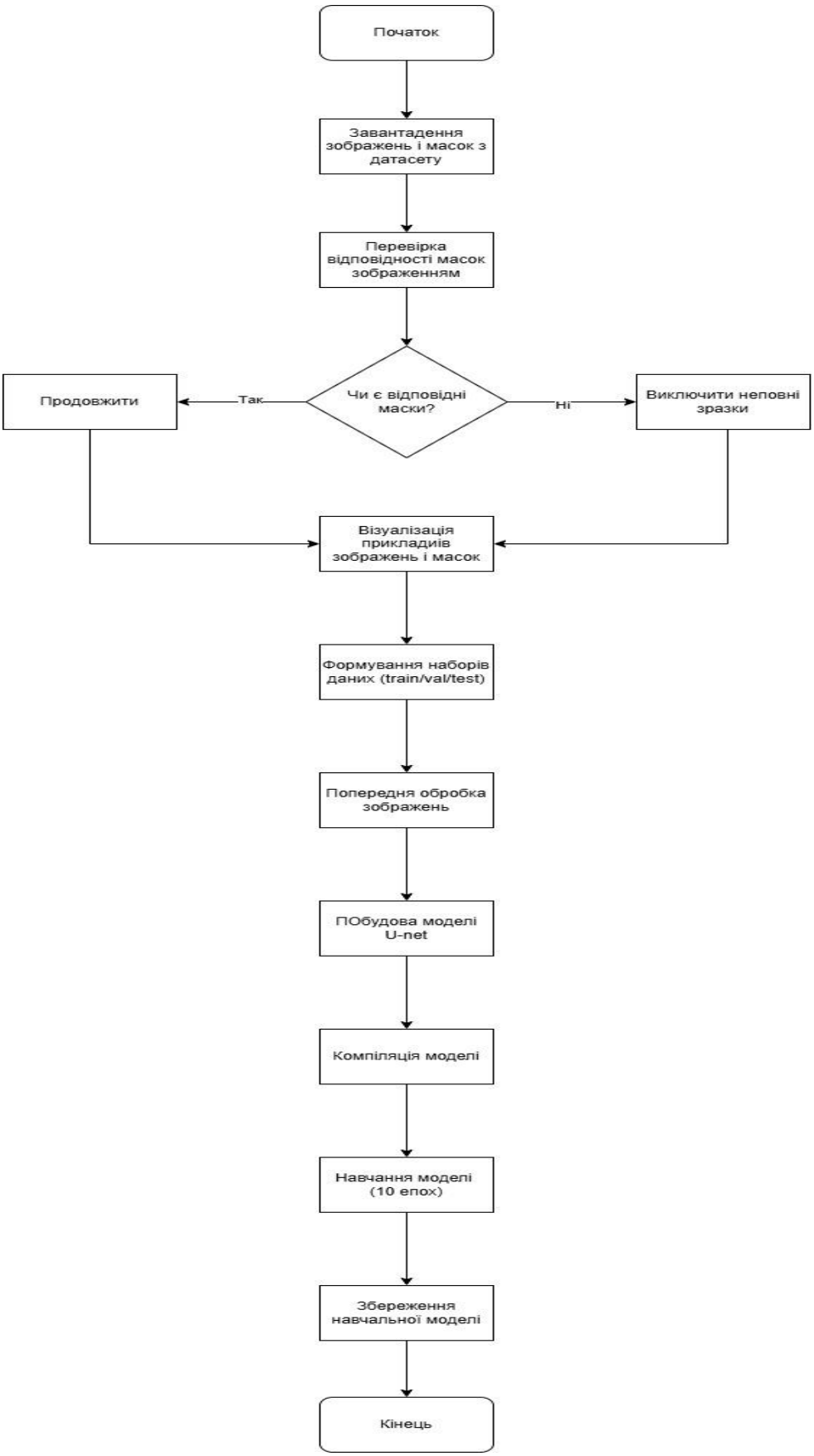

Рисунок 2.5 – Загальний алгоритм роботи програмного модуля

2.5 Обґрунтування вибору математичної моделі програмного модуля

Для реалізації задачі сегментації дерматоскопічних зображень було розроблено математичну модель, що ґрунтується на архітектурі згорткової нейронної мережі U-Net. Математична модель описує послідовність обчислювальних операцій, що перетворюють вхідне зображення у карту ймовірностей для кожного пікселя щодо належності до ураженої ділянки. Ця модель ефективно поєднує локальні та глобальні ознаки, що є критичним для точного виявлення та сегментації уражених ділянок шкіри з нерівними, нечіткими межами задля збільшення точності у всіх можливих випадках.

Основним елементом моделі є згортковий шар (Convolutional Layer), що виконує операцію згортки над вхідним зображенням або над картою ознак з використанням ядра згортки розміром $k \times k$ [12]. З математичної точки зору операція згортки для одного фільтра описується наступним чином:

$$Y(i,j) = \sum_{m=0}^{k-1} \sum_{m=0}^{k-1} X(i+m, j+n) * K(m,n) + b, \quad (2.1)$$

де:

- $X(i,j)$ — значення пікселя вхідного зображення на позиції (i,j);
- $K(m,n)$ — коефіцієнти ядра згортки;
- b — зміщення (bias);
- $Y(i,j)$ — результат згортки в позиції (i,j).

Після кожної операції згортки до результату застосовується нелінійна функція активації. У запропонованій моделі використано функцію активації ReLU (Rectified Linear Unit), яка задається виразом:

$$f(x) = \max(0, x). \quad (2.2)$$

Вона дозволяє уникнути проблеми зникаючих градієнтів та пришвидшує навчання моделі. Після кожних двох послідовних згорткових шарів

застосовується операція підвибірки (max pooling) для зменшення розмірності карти ознак та виділення найбільш суттєвих ознак:

$$\mathrm{Y(i,j)} = \max\{\mathrm{X(m,n)}\}, \tag{2.3}$$

де (m, n) — елементи підматриці розміром 2×2 на відповідній позиції. У зворотній (декодерній) частині моделі для відновлення розмірності застосовується операція транспонованої згортки (Conv2DTranspose), яка математично є оберненою до звичайної згортки, дозволяючи збільшити просторовий розмір карти ознак [12].

Особливістю моделі є використання механізму пропускових з'єднань, які здійснюють конкатенацію карт ознак відповідних рівнів енкодера та декодера. Це дозволяє зберегти просторові ознаки з ранніх рівнів та передати їх до фінальних етапів обробки. На вихідному шарі застосовується згортка з ядром 1×1, що дозволяє перетворити багатоканальну карту ознак у карту ймовірностей сегментації для кожного пікселя. Остаточне рішення щодо наявності патологічної ділянки для кожного пікселя приймається на основі порогового значення функції sigmoid:

$$\sigma(\mathrm{x}) = \frac{1}{1+e^{-x}}, \tag{2.4}$$

яка нормалізує значення у діапазон [0, 1].

В якості функції втрат для оптимізації параметрів мережі було обрано комбіновану функцію Binary Cross-Entropy (BCE) та Dice Loss, що дозволяє одночасно враховувати як піксельну точність, так і збалансованість між помилками пропуску та хибного виявлення для отримання найбільшої точності:

$$Loss = \alpha \cdot BCE + (1 - \alpha) \cdot (1 - Dice), \tag{2.5}$$

де BCE — бінарна крос-ентропія:

$$BCE = -\frac{1}{N}\sum_{i=1}^{N}[y_i \cdot \log(p_i) + (1-y_i)\cdot\log(1-p_i)], \quad (2.6)$$

де $Dice$ – коефіцієнт схожості:

$$Dice = \frac{2\sum y_i p_i}{\sum y_i + \sum p_i}. \quad (2.7)$$

Параметр α дозволяє регулювати вагу кожної складової у загальній функції втрат.

2.6 Розробка UML діаграми програмного модуля розпізнавання раку шкіри

Модель – це абстракція, створена для осмислення та формалізації певної предметної області чи системи перед її реалізацією. Абстрагування полягає у вибірковому виділенні суттєвих аспектів проблеми, ізоляції важливих деталей і відкиданні другорядних [5].

UML (Unified Modeling Language) – уніфікована мова моделювання, що є стандартом для візуалізації, опису та документування об’єктно-орієнтованих систем. UML використовує графічні позначення для створення абстрактної моделі системи, яка допомагає розробникам краще розуміти структуру та взаємодію її компонентів.

Одним із ключових типів UML-діаграм є діаграма класів – статичне представлення структури системи, що відображає класи, їх атрибути, операції та зв’язки між ними. Діаграма класів допомагає формалізувати основні сутності

системи, їх властивості та взаємодії, що є фундаментом для подальшої розробки програмного забезпечення.

Для опису структури програмного модуля розпізнавання раку шкіри розроблено UML-діаграму класів, яка включає основні компоненти модуля: клас моделі U-Net, клас результатів сегментації, модуль збереження результатів, а також допоміжні класи для попередньої обробки даних та візуалізації. Така діаграма відображає статичні зв'язки між класами, їх функціональні можливості та потоки даних у межах модуля. Нижче наведена розроблена діаграма класів програмного модуля розпізнавання раку шкіри (рисунок 2.5).

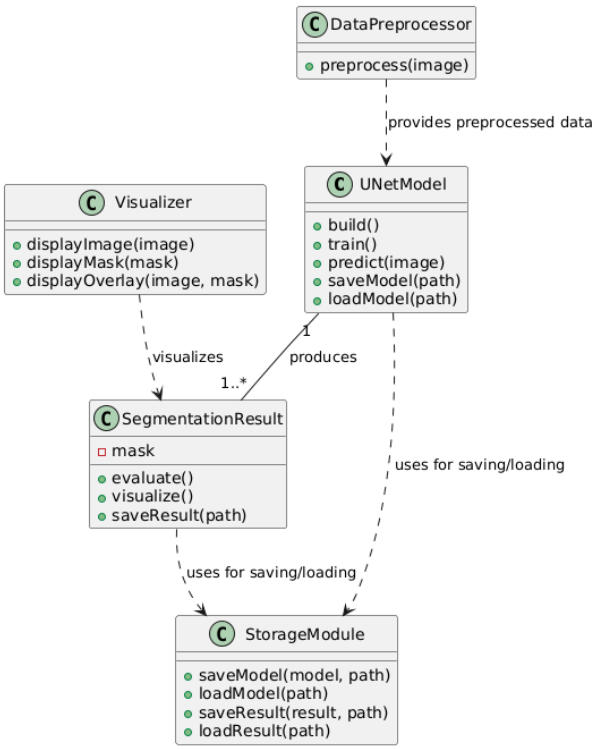

Рисунок 2.6 – UML-діаграма класів програмного модуля розпізнавання раку шкіри

2.7 Висновок до розділу 2

Здійснено варіантний аналіз трьох різних підходів до вирішення поставленої задачі. Розглянуто методи ознакового машинного навчання,

алгоритми неконтрольованого навчання та нейронні мережі. Кожен з підходів відрізняється рівнем складності, точністю, потребами в розмічених даних та обчислювальними ресурсами. На основі порівняльного аналізу обрано підхід на основі нейронних мереж, оскільки саме він забезпечує автоматичне виділення ознак, високу точність класифікації та здатність до просторової локалізації патологій, що є ключовим у завданні розпізнавання раку шкіри.

Виділено основні компоненти програмного модуля, що включають обробку вхідних даних, побудову та навчання моделі, оцінку результатів і збереження отриманих даних. Також було розроблено загальний алгоритм роботи програмного модуля, який послідовно описує етапи від підготовки зображень до отримання кінцевого результату сегментації.

Проведено аналіз сучасних архітектур нейронних мереж для задачі сегментації медичних зображень, зокрема дерматоскопічних знімків. Розглянуто такі моделі, як ResNet, DenseNet, EfficientNet та Vision Transformer, які відрізняються за структурою, підходами до обробки даних і ефективністю.

Обґрунтовано вибір архітектури U-Net для розробки програмного модуля, що спеціалізується на сегментації медичних зображень і має унікальну особливість – пропускові з’єднання, які зберігають просторову інформацію.

Розроблено математичну модель, що описує послідовність операцій згорткових шарів, активацій, підвибірки та транспонованих згорток із застосуванням комбінованої функції втрат (Binary Cross-Entropy та Dice Loss). Це дозволяє моделі одночасно оптимізувати точність піксельної класифікації та якість сегментації, що є ключовим для розпізнавання складних уражень шкіри та підвищення точності.

Розроблено UML-діаграму класів програмного модуля, яка формалізує архітектуру системи, включаючи основні компоненти: модель U-Net, модуль збереження результатів, обробку вхідних даних та візуалізацію. Такий підхід забезпечує зрозумілу структуру, гнучкість та можливість масштабування модуля.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ПРОГРАМНОГО МОДУЛЯ РОЗПІЗНАВАННЯ РАКУ ШКІРИ

3.1 Обґрунтування вибору мови програмування та бібліотек

Для розробки програмного модуля розпізнавання раку шкіри обрано хмарне середовище Kaggle через його потужні обчислювальні ресурси, доступ до GPU та широкий набір готових бібліотек для машинного навчання.

Визначимо інструменти, за допомогою яких буде створюватись програмна реалізація програмного модуля розпізнавання раку шкіри. Серед великої кількості мов програмування, що можуть бути використані для реалізації систем штучного інтелекту, особливо для задач комп'ютерного зору, виділимо три найбільш релевантні у контексті розробки модуля розпізнавання раку шкіри: JavaScript, C++ та Python. Саме ці мови найчастіше застосовуються для створення та інтеграції нейронних мереж, а також забезпечують широкий вибір бібліотек, фреймворків і середовищ розробки, необхідних для реалізації повного циклу машинного навчання — від обробки даних до візуалізації результатів.

JavaScript – це високорівнева, інтерпретована мова програмування, що спочатку створювалася для клієнтської частини веб-додатків. Завдяки поширенню екосистеми Node.js, JavaScript також широко використовується на стороні серверу. Незважаючи на це, JavaScript рідко застосовується для складних обчислювальних задач та глибинного навчання, оскільки не має розвинутого інструментарію для роботи з великими обсягами даних, GPU-обчисленнями та нейронними мережами. Наявні бібліотеки для машинного навчання такі як, TensorFlow.js або Brain.js не забезпечують такої ж гнучкості та продуктивності, як їхні аналоги для Python [20].

C++ – є мовою програмування з високою продуктивністю та низькорівневим контролем пам'яті. Вона широко використовується в системному програмному забезпеченні, розробці ігор та інших

високопродуктивних програмах. Завдяки своїй передовій технології компіляції вона має деякі суттєві переваги при використанні для створення нейронних мереж, такі як швидкість та низькі накладні витрати [21].

Python – інша мова програмування, яка відзначається простотою та читабельністю. Вона має багату екосистему бібліотек, таких як TensorFlow, PyTorch та Scikit-learn, що робить її популярною у сфері штучного інтелекту та машинного навчання. Python дозволяє швидко створювати прототипи та експериментувати з моделями, що є важливим для досліджень та розробки [22].

У таблиці 3.1 наведена порівняльна характеристика мов програмування.

Таблиця 3.1 – Порівняльна характеристика мов програмування

Критерій	JavaScript	C++	Python
Синтаксис	Простий	Складний	Простий
Продуктивність	Середня	Висока	Висока (з бібліотеками)
Простота реалізації ML	Низька	Середня	Висока
Підтримка бібліотек AI/ML	Обмежена (TF.js)	Низька	Найвища (TensorFlow, PyTorch)
Спільнота та підтримка	Висока	Висока	Дуже висока
Кросплатформеність	Веб/Node.js	Висока	Висока

З огляду на вищенаведене, для реалізації даного проєкту було обрано мову Python, оскільки вона поєднує простоту розробки, багатство екосистеми, активне спільнота, а також підтримку широкого спектру бібліотек для аналізу даних та розробки нейронних мереж. Крім того, Python дозволяє легко інтегрувати наукові дослідження з візуалізацією результатів та побудовою інтерфейсів для взаємодії з користувачем.

А тепер розглянемо питання обґрунтування вибору бібліотек розробки нейронних мереж: TensorFlow, PyTorch та Scikit-learn.

TensorFlow – це масштабована платформа машинного навчання з відкритим вихідним кодом, створена та підтримувана компанією Google. Вона є однією з найпоширеніших бібліотек у сфері глибинного навчання завдяки своїй гнучкості, продуктивності та активному розвитку. TensorFlow надає інструменти як для низькорівневого програмування обчислювальних графів, так і для високорівневого опису нейронних мереж через API Keras, що суттєво полегшує створення моделей для користувачів із різним рівнем досвіду. Однією з найсильніших сторін TensorFlow є її здатність ефективно працювати на різних типах обчислювального обладнання – від звичайних процесорів до графічних процесорів (GPU) та спеціалізованих тензорних процесорів (TPU), що дозволяє масштабувати обчислення та пришвидшувати навчання великих моделей. Вбудована підтримка метрик, моніторингу процесу навчання, колбеків, системи збереження чекпоїнтів і валідації моделей на окремих вибірках робить TensorFlow придатним як для дослідницької, так і для комерційної розробки [23].

PyTorch – ще одна популярна бібліотека для глибинного навчання, створена лабораторією Facebook AI Research (FAIR). Вона відзначається гнучкістю завдяки динамічній обчислювальній графі, яка надає змогу виконувати обчислення в реальному часі, що особливо зручно на етапах дослідження, розробки нових архітектур і експериментів з гіперпараметрами. PyTorch дозволяє програмісту змінювати архітектуру моделі «на льоту» без необхідності компіляції всього графа обчислень, що вигідно вирізняє її серед інших бібліотек, зокрема у сфері академічних досліджень. Бібліотека має зручний API, гарно інтегрується з іншими науковими бібліотеками Python, як-от NumPy та SciPy, а також активно використовується у найсучасніших публікаціях з машинного навчання. Однак, незважаючи на беззаперечні переваги для прототипування та досліджень, PyTorch дещо поступається TensorFlow у питаннях масштабування, інструментів розгортання на мобільних пристроях та підтримки виробничих середовищ, хоча останніми роками ця різниця поступово нівелюється [24].

Scikit-learn – є бібліотекою машинного навчання, орієнтованою переважно на класичні алгоритми аналізу даних, такі як регресія, класифікація, кластеризація, зниження розмірності, побудова дерев рішень та інші методи, що не потребують побудови глибоких нейронних мереж. Вона не забезпечує повноцінної підтримки глибинного навчання, однак має надзвичайно простий і зручний інтерфейс, що дозволяє швидко створювати базові моделі, проводити порівняльний аналіз, а також здійснювати попередню обробку даних, перехресну валідацію, побудову пайплайнів тощо. Scikit-learn широко використовується для побудови базових прототипів, а також як допоміжна бібліотека в більш складних проєктах, де класичні методи аналізу поєднуються з нейронними мережами. Через відсутність вбудованої підтримки GPU та обмеження у роботі з тензорними структурами Scikit-learn не розглядається як основний інструмент для реалізації задач глибинного навчання, однак залишається важливим компонентом екосистеми Python у сфері машинного навчання [25].

На основі аналізу було обрано бібліотеку TensorFlow, оскільки вона забезпечує високу продуктивність, широкі можливості для побудови та експорту моделей, а також має вбудовану інтеграцію з Keras для швидкої розробки.

3.2 Програмна реалізація компонентів модуля розпізнавання раку шкіри

Розглянемо основні моменти програмної реалізації програмного модуля розпізнавання раку шкіри. Завантаження та підготовка вхідних даних. На початковому етапі виконується імпорт необхідних бібліотек, зокрема TensorFlow, NumPy, PIL, scikit-learn та ін., а також завантаження датасету з Kaggle. Після цього зображення та відповідні маски сегментації зберігаються в окремих директоріях та фільтруються за розширенням. Для забезпечення коректного зіставлення масок та відповідних зображень перевіряється наявність усіх пар:

```
img_files = sorted([f for f in os.listdir(img_dir) if f.endswith(".jpg")])
mask_files = sorted([f for f in os.listdir(mask_dir) if f.endswith(".png")])
missing_masks = [f for f in image_names if f not in mask_names]
```

Цей блок дозволяє впевнитися, що жодне зображення не залишилося без відповідної маски сегментації. Для наочного представлення даних реалізовано два способи виведення зображень, виведення зображення та маски окремо та накладання маски на зображення з використанням прозорості (alpha=0.5). Це дозволяє швидко оцінити якість та точність наявної розмітки перед тренуванням моделі.

```
def display_image_and_mask(n=5, seed=None):
def display_image_with_mask(n=5, seed=None):
```

Для забезпечення найбільшої точності було поділено вибірки у наступному співвідношенні:

- 90% — тренувальна вибірка;
- 5% — валідаційна;
- 5% — тестова.

Використовується функція train_test_split з модуля sklearn.model_selection, що гарантує рандомізований і збалансований розподіл:

```
X_train, X_temp, y_train, y_temp = train_test_split(imgs_path, masks_path, train_size=0.90, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, train_size=0.5, random_state=42)
```

Використано функціональність tf.data.Dataset для ефективного завантаження, масштабування та нормалізації зображень і масок. Функція map_fn обробляє дані, приводячи їх до одного розміру (256×256 пікселів), нормалізуючи значення пікселів у діапазон [0, 1].

```
def map_fn(img_path, mask_path):
train_set = tf.data.Dataset.from_tensor_slices((X_train, y_train))
```

Додатково використовуються буферизація (prefetch) та паралельна обробка (num_parallel_calls) для оптимізації швидкості обробки.

Основу модуля сегментації становить згорткова нейронна мережа архітектури U-Net. Для збереження просторової інформації застосовуються "skip connections".

```
def UNET():
    inputs = Input(shape=(IMG_SIZE, IMG_SIZE, 3))
    x = Conv2D(32, 3, strides=1, padding='same')(inputs)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    skip_connections = []
    for filters in [64, 128, 256, 512]:
        x = Conv2D(filters, 3, strides=1, padding='same')(x)
        x = BatchNormalization()(x)
        x = Activation('relu')(x)
        x = Conv2D(filters, 3, strides=1, padding='same')(x)
        x = BatchNormalization()(x)
        x = Activation('relu')(x)
        skip_connections.append(x)
        x = MaxPooling2D(2, strides=2, padding='same')(x)
    x = Conv2D(1024, 3, strides=1, padding='same')(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = Conv2D(1024, 3, strides=1, padding='same')(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    for filters in [512, 256, 128, 64]:
        x = Conv2DTranspose(filters, 2, strides=2, padding='same')(x)
        skip_connection = skip_connections.pop()
```

```
        x = add([x, skip_connection])
        x = Conv2D(filters, 3, strides=1, padding='same')(x)
        x = BatchNormalization()(x)
        x = Activation('relu')(x)
        x = Conv2D(filters, 3, strides=1, padding='same')(x)
        x = BatchNormalization()(x)
        x = Activation('relu')(x)
    outputs = Conv2D(1, 1, strides=1, activation='sigmoid')(x)
    model = Model(inputs=[inputs], outputs=[outputs])
    return model
```

Мережа завершується одним згортковим шаром з активацією sigmoid, що дозволяє отримати двійкову маску сегментації.

Далі модель компілюється з функцією втрат BinaryCrossentropy, оптимізатором Adam та метриками точності, точності класифікації (Precision), повноти (Recall) і коефіцієнтом Dice. Останній використовується для оцінки якості сегментації:

```
model.compile(optimizer=Adam(0.0002),
              loss=BinaryCrossentropy(),
              metrics=['accuracy', Precision(name='precision'), Recall(name='recall'),
dice_coefficient])
```

Для зменшення перенавчання використовуються колбеки: ModelCheckpoint для збереження найкращих ваг та ReduceLROnPlateau для зменшення швидкості навчання, що забезпечує покращення точності. Навчання триває 10 епох із валідаційною перевіркою:

```
history = model.fit(train_set, epochs=10, validation_data=val_set,
callbacks=[checkpoint_cb, reduce_lr_cb])
```

Після навчання модель зберігається у форматі .keras. У подальшому вона завантажується для проведення тестування:

```
model.save('skin-cancer-segmentation.keras')
```

loaded_model = tf.keras.models.load_model(..., custom_objects={...})

Для підрахунку точності модель оцінюється на тестовій вибірці:

loss, accuracy, precision, recall, dice_coefficient = model.evaluate(test_set)

3.3 Тестування роботи програмного модуля та аналіз його результатів

Після завершення процесу навчання нейронної мережі було проведено оцінювання на тестовій вибірці з метою перевірки точності та сегментації. коректність її роботи. На графіках було візуалізовано зміну значень функції втрат (Loss), точності (Accuracy) та коефіцієнта Dice під час навчання. На рисунках 3.1, 3.2 зображено точність на етапах навчання та перевірки та результати створеної моделі для розпізнавання раку шкіри відповідно.

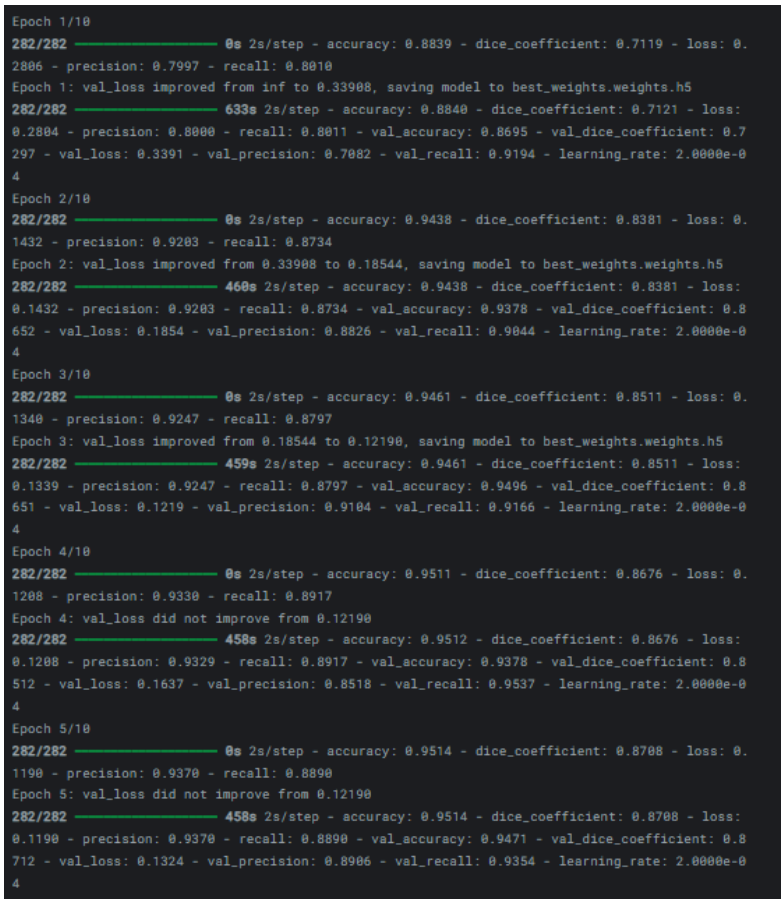

Рисунок 3.1 – Точність на етапах навчання та перевірки

```
loss, accuracy, precision, recall, dice_coefficient = model.evaluate(test_set)

print(f"Test Loss: {loss:.4f}")
print(f"Test Accuracy: {accuracy:.4f}")
print(f"Test Precision: {precision:.4f}")
print(f"Test Recall: {recall:.4f}")
print(f"Test Dice Coefficient: {dice_coefficient:.4f}")
```

```
16/16 ━━━━━━━━━━━━━━━━━━━━ 11s 511ms/step - accuracy: 0.9577 - dice_coefficient: 0.8900 - loss: 0.1078 - precision: 0.9393 - recall: 0.9
138
Test Loss: 0.1120
Test Accuracy: 0.9558
Test Precision: 0.9367
Test Recall: 0.9108
Test Dice Coefficient: 0.8892
```

Рисунок 3.2 – Результати створеної моделі для розпізнавання раку шкіри

Після демонстрації візуальних результатів сегментації на рисунку 3.2 доцільно розглянути динаміку навчання моделі. На рисунку 3.3 зображено зміну функції втрат протягом 10 епох навчання.

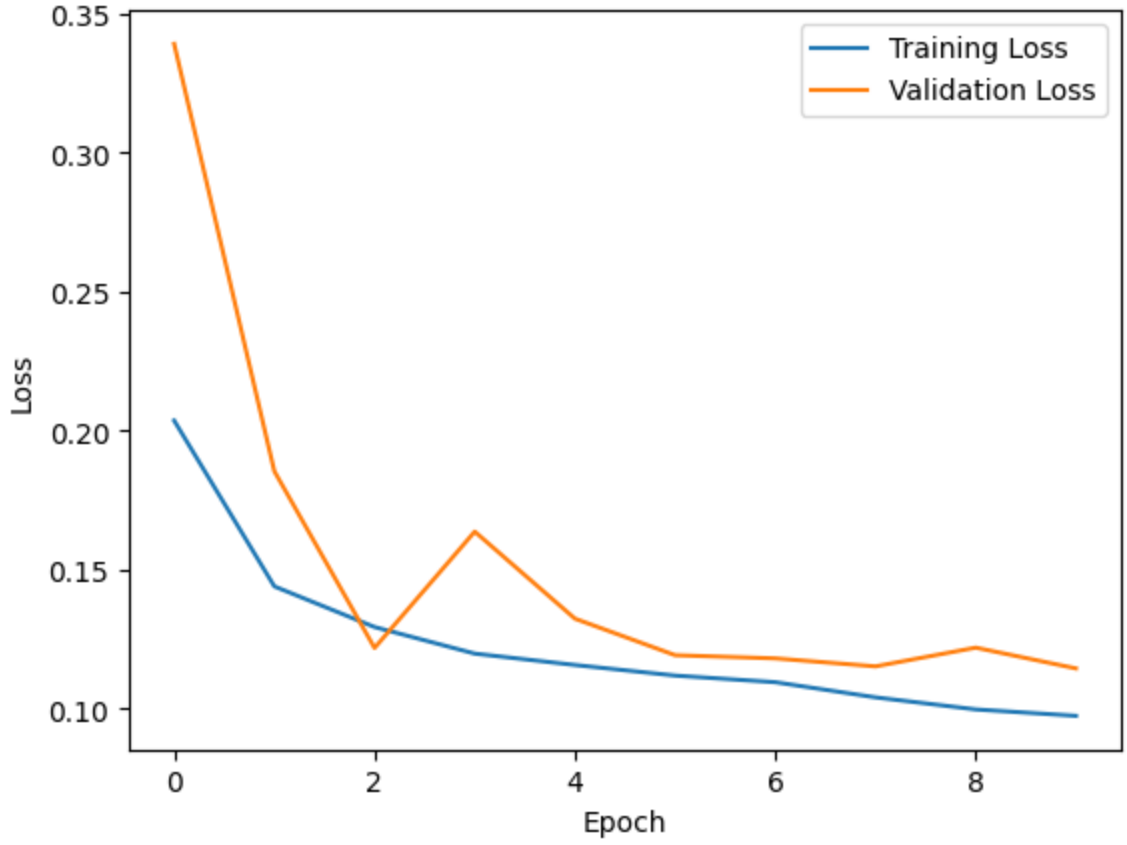

Рисунок 3.3 – Графік зміни функції втрат протягом 10 епох

На графіку значення Training Loss демонструє стабільне зменшення, що вказує на поступове покращення здатності моделі до узгодження з тренувальними даними. Водночас Validation Loss спершу швидко знижується, однак у подальших епохах спостерігаються коливання, що може свідчити про початок перенавчання. Проте загальна динаміка залишилася позитивною.

На рисунку 3.4 наведено зміну коефіцієнта Dice – метрики, що відображає ступінь схожості між предикованими та фактичними масками.

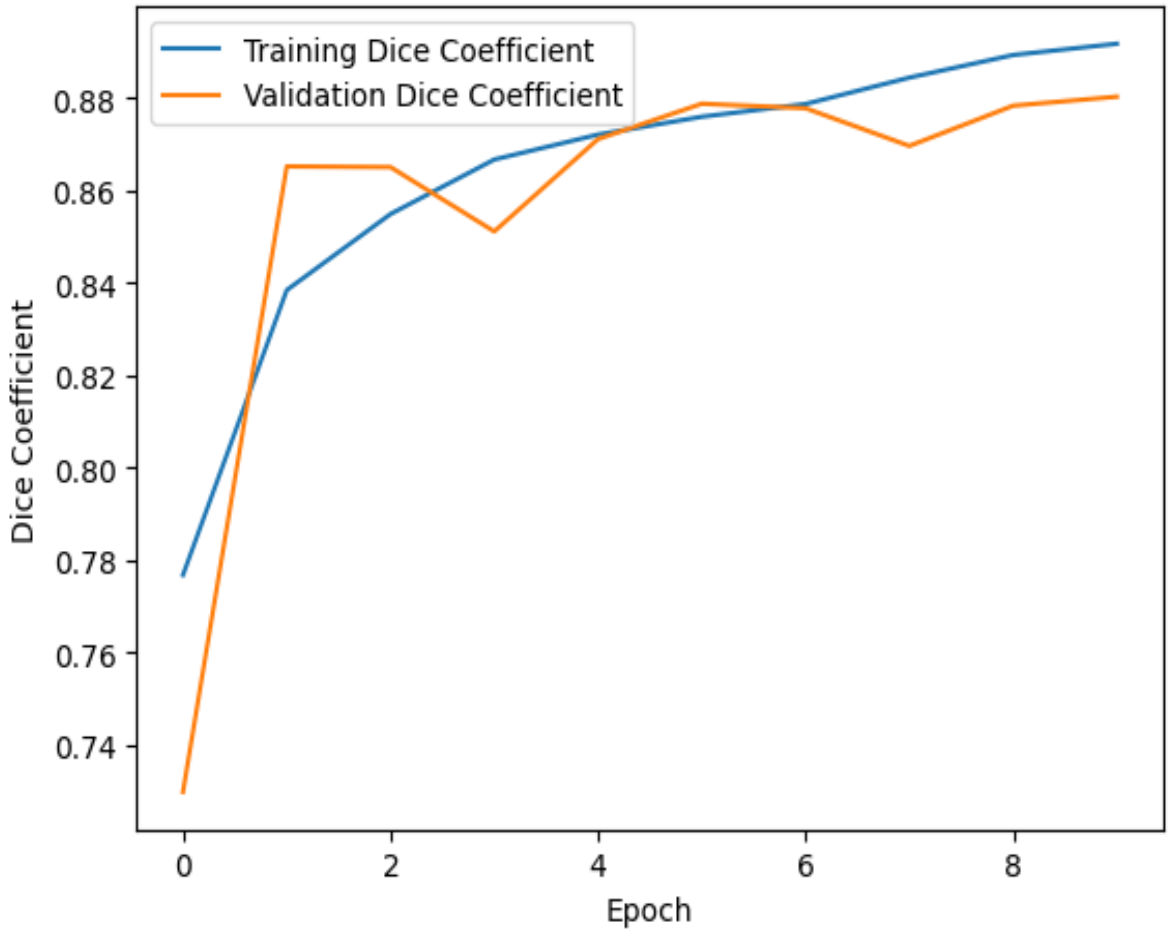

Рисунок 3.4 – Графік зміни коефіцієнта Dice протягом 10 епох

Training Dice Coefficient демонструє стійке зростання протягом усього процесу навчання, досягаючи значення понад 0.89 на фінальних епохах. Значення Validation Dice Coefficient також є високим, особливо на середніх епохах, однак у фінальних епохах помітне незначне зниження, що може бути

ознакою перенавчання або впливу шуму у валідаційній вибірці. Загалом, високе значення коефіцієнта Dice на обох множинах свідчить про ефективність моделі у виконанні задачі сегментації.

На рисунку 3.5 та 3.6 наведено приклад візуалізації результатів сегментації зображень шкіри, отриманих за допомогою розробленої моделі U-Net. Перший рядок зображень містить вихідні зображення дерматологічних утворень. Другий рядок демонструє відповідні еталонні маски (ground truth), які були використані для навчання та перевірки моделі. У третьому рядку наведені результати, передбачені моделлю, а також значення коефіцієнта Dice для кожного з прикладів.

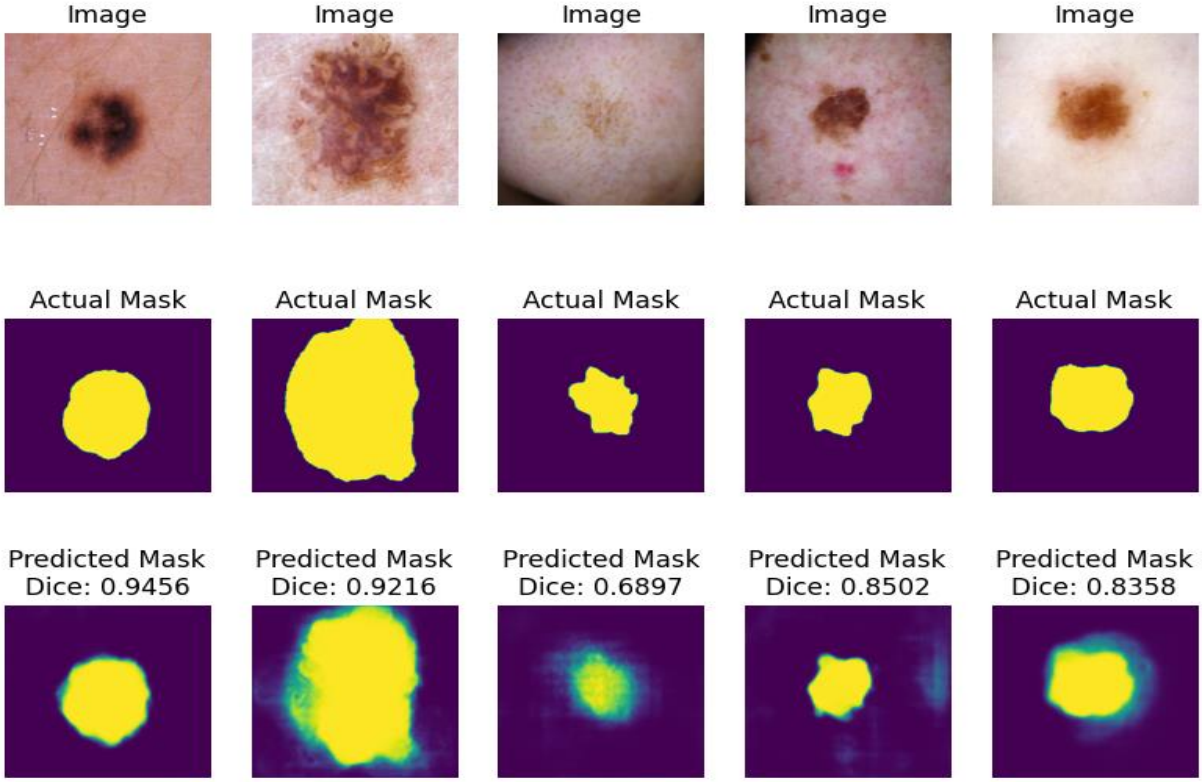

Рисунок 3.5 – Візуалізація результатів сегментації зображень

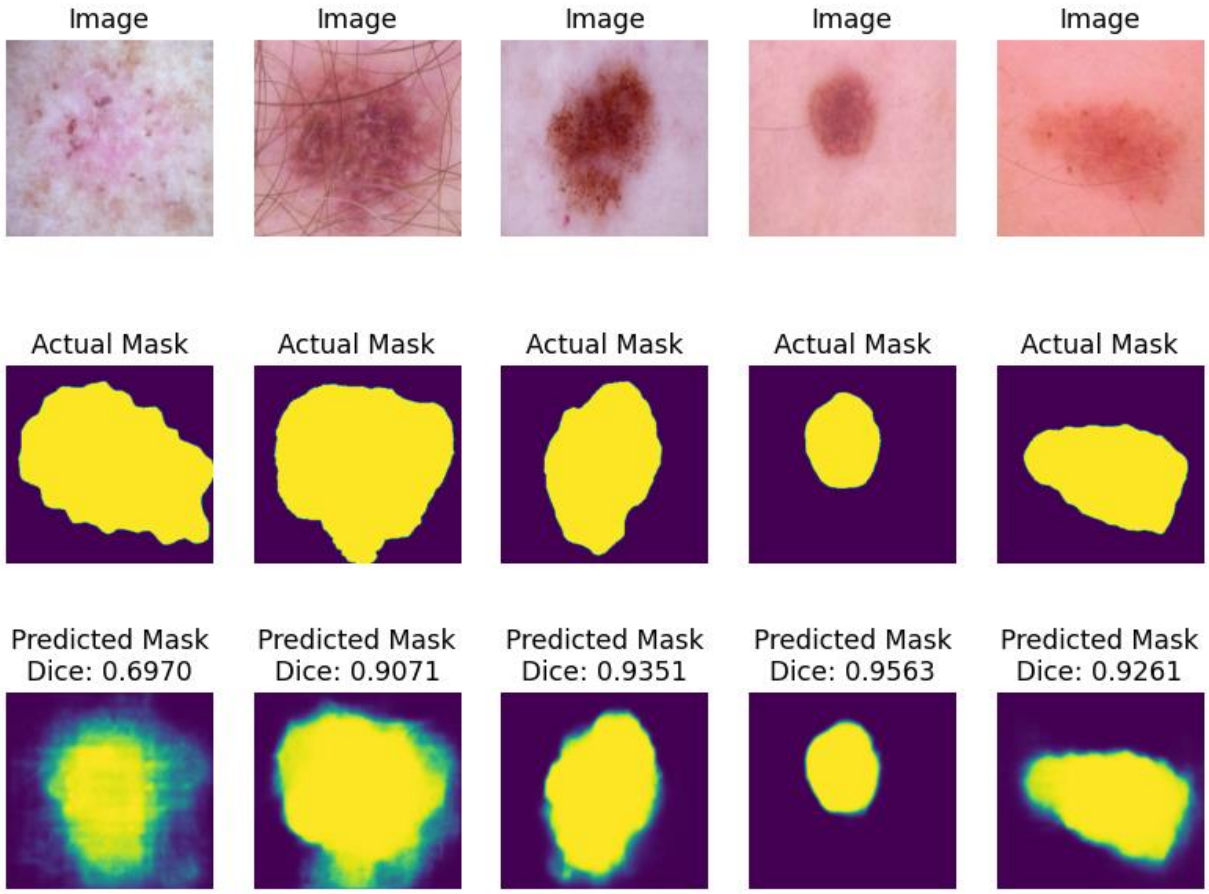

Рисунок 3.6 – Візуалізація результатів сегментації зображень

Загалом модель показала високу точність відновлення контурів уражених ділянок. У більшості прикладів коефіцієнт Dice перевищує 0,90, що свідчить про високий ступінь накладання передбаченої маски на еталонну. Зокрема, для деяких зображень (наприклад, дев’ятого та десятого прикладів) точність наближається до 0,95. Це підтверджує здатність моделі ефективно локалізувати межі патологічних утворень.

Проте у випадках, де межі ураження нечіткі або зображення мають низький контраст (наприклад, третій та шостий приклади), якість сегментації знижується — відповідні значення Dice становлять 0,6897 та 0,6970. Це вказує на обмеження моделі у складніших сценаріях, однак навіть у таких випадках вона продовжує виділяти загальний контур ураженої ділянки.

Розроблена модель U-Net демонструє найвищі результати серед розглянутих аналогів: точність становить 95,58%, precision – 93,67%, recall – 91,08%, а коефіцієнт Dice – 88,92%. Це свідчить про здатність моделі не лише виявляти більшість уражених ділянок, але й з високою точністю узгоджувати їх із еталонними масками. Порівняно з попередньою U-Net моделлю, яка мала точність близько 93,70% та Dice коефіцієнт 85,36%, запропонована модель покращила точність на 1,88%, а Dice коефіцієнт — на 3,56%, що є суттєвим покращенням для задачі медичної сегментації. Результати порівняння наведені у таблиці 3.1 наведена Порівняльна характеристика розробленої моделі з її аналогами.

Таблиця 3.1 – Порівняльна характеристика розробленої моделі з її аналогами

Параметр	U-Net модель	DenseNet модель	Розроблена U-Net модель
Тип задачі	Сегментація	Класифікація	Сегментація
Accuracy	93,70%	87,53%	95,58%
Precision	85,88%	~88%	93,67%
Recall	93,96%	~88%	91,08%
Dice коефіцієнт	85,36%	–	88,92%
Локалізація уражень	Так	Ні	Так
Використана архітектура	U-Net	DenseNet201	U-Net
Платформа	TensorFlow	TensorFlow	TensorFlow

Як видно з таблиці, розроблена модель перевершує існуючі аналоги за ключовими метриками, що є критично важливим у медичній діагностиці, де точне локалізування уражених ділянок безпосередньо впливає на якість

постановки діагнозу та вибір лікування. Покращення точності та Dice коефіцієнта свідчить про більш надійне виділення патологічних областей і зменшення кількості помилкових спрацьовувань, що підвищує клінічну цінність розробленої моделі.

3.4 Висновок до розділу 3

Обґрунтовано вибір інструментів для розробки програмного модуля розпізнавання раку шкіри. Для реалізації програмного модуля використано мову Python та бібліотеку TensorFlow у хмарному середовищі Kaggle, що дозволило ефективно використовувати обчислювальні ресурси та прискорити процес розробки.

Було виконано повну програмну реалізацію модуля: розроблено алгоритм підготовки та попередньої обробки зображень, побудовано та навчено модель U-Net, реалізовано функції збереження результатів та візуалізації сегментації. Особливу увагу приділено оптимізації структури даних, використанню сучасних пайплайнів для обробки та розподілу вибірки на тренувальну, валідаційну і тестову частини.

Здійснено тестування роботи програмного модуля на незалежній тестовій вибірці. Навчання моделі проводилося протягом 10 епох із використанням функції втрат Binary Cross-Entropy та метрики Dice, а також моніторингом точності, precision та recall. Аналіз графіків зміни функції втрат і метрик показав стабільне зниження loss, зростання Dice-коефіцієнта та високі значення точності й precision. За підсумками тестування, розроблена модель досягла точності 95,58%, precision 93,67%, recall 91,08% та Dice-коефіцієнта 88,92%, що перевищує результати попередніх аналогів.

ВИСНОВКИ

Усі завдання поставлені перед бакалаврською дипломною роботою виконані в повному об’ємі, а саме:

1. Проаналізовано існуючі рішення для розпізнавання зображень раку шкіри.
2. Обґрунтовано підхід до розв’язання задачі розпізнавання зображень раку шкіри та розроблено відповідний алгоритм.
3. Спроектовано програмний модуль розпізнавання раку шкіри.
4. Програмно реалізовано відповідний програмний модуль.
5. Проведено тестування розробленого програмного модуля та проведено аналіз результатів тестування.

У роботі було розглянуто поточні програмні аналоги розпізнавання раку та проаналізовано основні рішення, доступні користувачам. Відзначено, що їх є досить багато, але всі вони мають недостатню точність та надійність результатів в окремих умовах.

Обґрунтовано підхід до розв’язання задачі розпізнавання, зокрема вибір архітектури U-Net, яка завдяки пропусковим з’єднанням і багаторівневій структурі забезпечує досить високу точність сегментації патологічних ділянок. Розроблено відповідний алгоритм роботи програмного модуля, що містить етапи підготовки даних, навчання моделі, оцінки результатів та збереження отриманих даних.

Здійснено проєктування програмного модуля, виділено основні компоненти – обробку вхідних зображень, побудову та навчання моделі, модуль збереження результатів і візуалізацію. Розроблено UML-діаграму класів, що формалізувала структуру системи та полегшила подальшу реалізацію.

Обґрунтовано вибір мови програмування для реалізації програмного засобу. Обґрунтовано доцільність використання Python, оскільки вона повністю задовольняють вимогам розробки моделі нейронної мережі та дозволяє створювати високоточний програмний модуль. Для мови Python обрано

бібліотеку TensorFlow для розробки програмного модуля розпізнавання раку шкіри. На основі обраних вище інструментів розроблено відповідний програмний модуль розпізнавання раку шкіри.

Здійснено тестування програмного модуля на незалежній тестовій вибірці протягом 10 епох. Аналіз графіків функції втрат, точності, precision, recall та коефіцієнта Dice показав стабільне покращення метрик. Розроблена модель досягла точності 95,58%, precision — 93,67%, recall — 91,08%, а Dice коефіцієнт склав 88,92%, що перевищує показники попередніх аналогів. Це свідчить про підвищену здатність моделі точно розпізнавати уражені ділянки та зменшувати кількість помилкових спрацьовувань.

Результати виконаної роботи свідчать про те, що обране середовище розробки, інструменти та підходи дозволили створити програмний модуль з підвищеною точністю розпізнавання раку шкіри, отже, мету роботи досягнуто та поставлені задачі виконано в повному обсязі.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Трегуб А. К., Арсенюк І. Р. Підхід щодо розпізнавання раку шкіри за допомогою нейронної мережі // Тези доповідей LIV науково-технічної конференції факультету інтелектуальних інформаційних технологій та автоматизації (2025). – Вінниця: ВНТУ, 2025. Електронний ресурс (режим доступу https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2025/paper/view/24244/20014)
2. Базальноклітинний рак шкіри клінічна настанова, заснована на доказах. ДП "Державний експертний центр МОЗ України" – 2024 URL: https://moz.gov.ua/uploads/ckeditor/%D0%93%D1%80%D0%BE%D0%BC%D0%B0%D0%B4%D1%81%D1%8C%D0%BA%D0%B5%20%D0%BE%D0%B1%D0%B3%D0%BE%D0%B2%D0%BE%D1%80%D0%B5%D0%BD%D0%BD%D1%8F/2024/19012024/2024_01_03_%D0%9A%D0%9D%20%D0%91%D0%9A%D0%A0%-D0%A8.pdf (дата звернення: 22.02.2025).
3. Суботін С. О. Нейронні мережі: теорія та практика: навч. посіб./ С. О. Субботін. – Житомир: Вид. О. О. Євенок, 2020. – 184 с. URL: https://eir.zp.edu.ua/server/api/core/bitstreams/2abb401b-9ee6-4afca92a-2de5c332d12f/content (дата звернення: 20.05.2025).
4. Goodfellow I., Bengio Y., Courville A., Bach F. Deep Learning (Adaptive Computation and Machine Learning series). – The MIT Press book. – 2016 – 776 p. URL: https://www.deeplearningbook.org/ (дата звернення: 20.05.2025).
5. Арсенюк І. Р. Дослідження та розробка програмного блоку розпізнавання емоцій людини/ І. Р. Арсенюк, В. О. Денисюк, Є. В. Зінов'єв// Наука і техніка сьогодні. Серія «Техніка». – 2024. – № 3 (31) С. 767 – 782. URL: https://doi.org/10.52058/2786-6025-2024-3(31)-767-782
6. Hari Naga Sai Perisetla. Skin Lesion HAM10000 Using DenseNet URL: https://www.kaggle.com/code/harinagasaiperisetla/skin-lesion-ham10000-using-densenet (дата звернення: 20.05.2025).

7. K. Scott Mader., Skin Cancer MNIST: HAM10000 URL: https://www.kaggle.com/datasets/kmader/skin-cancer-mnist-ham10000/ (дата звернення: 20.05.2025).
8. mostafa-ml. Skin Lesion HAM10000 Using DenseNet URL: https://www.kaggle.com/code/mostafaessam33/skin-cancer-segmentation-tensorflow-94#The-Dice-coefficient-is-a-measure-of-the-similarity-between-two-sets,-A-and-B.-The-coefficient-ranges-from-0-to-1,-where-1-indicates-that-the-two-sets-are-identical,-and-0-indicates-that-the-two-sets-have-no-overlap.-It-is-defined-as (дата звернення: 20.05.2025).
9. From Convolution to Neural Network. URL: https://gregorygundersen.com/blog/2017/02/24/cnns/ (дата звернення: 20.05.2025).
10. Colah`s Blog, Christopher Olah. URL: https://colah.github.io/ (дата звернення: 20.05.2025).
11. Mingxuan X., Yufeng L., Xu Y., Min G., Weimin W. Convolutional neural network classification of cancer cytopathology images: taking breast cancer as an example// ICMVA '24: Proceedings of the 2024 7th International Conference on Machine Vision and Applications. – P. 145 – 149 https://doi.org/10.1145/3653946.3653968 URL: https://www.semanticscholar.org/paper/Convolutional-neural-networkclassification-of-as-Xiao-Li/6f5b551df53174c833d0716d56926a14799546a2 (дата звернення: 20.05.2025).
12. Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In Medical Image Computing and Computer-Assisted Intervention (MICCAI). URL: https://arxiv.org/abs/1505.04597 (дата звернення: 20.05.2025).
13. Tal Ridnik, Emanuel Ben-Baruch, Asaf Noy, Lihi Zelnik-Manor. ImageNet-21K Pretraining for the Masses. URL: https://arxiv.org/pdf/2104.10972 (дата звернення: 20.05.2025).

14. Litjens, G. et al. (2017). A survey on deep learning in medical image analysis. In Medical Image Analysis, 42, pp. 60–88. URL: https://doi.org/10.1016/j.media.2017.07.005 (дата звернення: 20.05.2025).
15. Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely Connected Convolutional Networks. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR). URL: https://arxiv.org/abs/1608.06993 (дата звернення: 20.05.2025).
16. Dosovitskiy, A. et al. (2020). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. URL: https://arxiv.org/abs/2010.11929 (дата звернення: 20.05.2025).
17. Sasha Targ, Diogo Almeida[1], Kevin Lyman. (2024). Resnet in Resnet: Generalizing Residual Architectures. URL: https://ar5iv.labs.arxiv.org/html/1603.08029 (дата звернення: 20.05.2025).
18. Tao Zhou, XinYu Ye, HuiLing Lu, Xiaomin Zheng, Shi Qiu, YunCan Liu. (2022). Dense Convolutional Network and Its Application in Medical Image Analysis. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC9060995/ (дата звернення: 20.05.2025).
19. Mingxing Tan, Quoc V. Le. (2020). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. URL: https://arxiv.org/abs/1905.11946 (дата звернення: 20.05.2025).
20. JavaScript Tutorial. URL: https://www.w3schools.com/js/default.asp (дата звернення: 20.05.2025).
21. C++ Tutorial. URL: https://www.w3schools.com/cpp/cpp_intro.asp (дата звернення: 20.05.2025).
22. Python Tutorial. URL: https://www.w3schools.com/cpp/cpp_intro.asp (дата звернення: 20.05.2025).
23. Tensorflow URL: https://www.tensorflow.org/ (дата звернення: 20.05.2025).
24. PyTorch Tutorial. URL: https://docs.pytorch.org/docs/stable/index.html (дата звернення: 20.05.2025).

25. Scikit-learn Tutorial. URL: https://scikit-learn.org/stable/user_guide.html (дата звернення: 20.05.2025).
26. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів спеціальності 122 «Комп'ютерні науки» [Електронний ресурс] / уклад.: А. А. Яровий, О. В. Сілагін, І. В. Богач. – Вінниця : ВНТУ, 2023. – (PDF, 54 с.).

Додатки

ДОДАТОК А (ОБОВ'ЯЗКОВИЙ) ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Програмний модуль розпізнавання раку шкіри

Тип роботи: бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра комп'ютерних наук
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 4,78 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

✓ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Яровий А.А., зав. каф. КН _____
(прізвище, ініціали, посада) (підпис)

Колесницький О.К., проф. каф КН _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Озеранський В.С.
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Арсенюк І.Р.
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Трегуб А.К.
(підпис) (прізвище, ініціали)

ДОДАТОК Б (ОБОВ'ЯЗКОВИЙ)

ЛІСТИНГ ПРОГРАМИ

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import glob as gb
from PIL import Image
import os
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow.keras.layers import Input, Conv2D, BatchNormalization, Activation,
MaxPooling2D, Conv2DTranspose, add
from tensorflow.keras import Model
from tensorflow.keras import backend as K
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.losses import BinaryCrossentropy
from tensorflow.keras.metrics import Precision, Recall
from tensorflow.keras.callbacks import ModelCheckpoint, ReduceLROnPlateau
# Clear GPU Memory
tf.keras.backend.clear_session()
!pip install kaggle
!kaggle datasets download -d surajghuwalewala/ham1000-segmentation-and-
classification
!unzip -q ham1000-segmentation-and-classification.zip
img_dir = "/kaggle/input/ham1000-segmentation-and-classification/images"
mask_dir = "/kaggle/input/ham1000-segmentation-and-classification/masks"

img_files = sorted([f for f in os.listdir(img_dir) if f.endswith(".jpg")])
mask_files = sorted([f for f in os.listdir(mask_dir) if f.endswith(".png")])
image_names = [os.path.splitext(f)[0] for f in img_files]    # ('ISIC_0024306', '.jpg')
mask_names = [os.path.splitext(f)[0].replace('_segmentation', '') for f in mask_files]
# ('ISIC_0024306_segmentation', 'png')

missing_masks = [f for f in image_names if f not in mask_names]

if len(missing_masks) == 0:
    print('No missing masks found.')
else:
    print(f"There are {len(missing_masks)} missing masks found:")
    print(missing_masks)
```

```
len(img_files), len(mask_files)

def display_image_and_mask(n=5, seed=None):
    if seed:
        np.random.seed(seed)

    fig, axs = plt.subplots(2, n, figsize=(20, 6))
    for i in range(n):
        idx = np.random.randint(0, len(img_files))
        img_path = os.path.join(img_dir, img_files[idx])
        mask_path = os.path.join(mask_dir, os.path.splitext(img_files[idx])[0] +
'_segmentation.png')    # mask_files[idx]

        img = Image.open(img_path)
        mask = Image.open(mask_path)

        axs[0, i].imshow(img)
        axs[0, i].set_title('Image')
        axs[0, i].axis('off')

        axs[1, i].imshow(mask)
        axs[1, i].set_title('Mask')
        axs[1, i].axis('off')

plt.show()
display_image_and_mask(n=5, seed=42)
def display_image_with_mask(n=5, seed=None):
    if seed:
        np.random.seed(seed)

    fig, axs = plt.subplots(1, n, figsize=(20, 5))
    for i in range(n):
        idx = np.random.randint(0, len(img_files))
        img_path = os.path.join(img_dir, img_files[idx])
        mask_path = os.path.join(mask_dir, os.path.splitext(img_files[idx])[0] +
'_segmentation.png')    # mask_files[idx]

        img_np = np.array(Image.open(img_path))
        mask_np = np.array(Image.open(mask_path))

        axs[i].imshow(img_np)
        axs[i].imshow(mask_np, cmap='Reds', alpha=0.5)
        axs[i].set_title('Image with Mask')
```

```
        axs[i].axis('off')

plt.show()
display_image_with_mask(n=5, seed=42)
IMG_SIZE = 256
BATCH_SIZE = 32
BUFFER_SIZE = 1000
AUTOTUNE = tf.data.experimental.AUTOTUNE
# img_dir
# mask_dir
def img_mask_paths(img_dir, mask_dir):
    img_path = sorted(gb.glob(os.path.join(img_dir, '*.jpg')))
    mask_path = sorted(gb.glob(os.path.join(mask_dir, '*.png')))
    return np.array(img_path), np.array(mask_path)

imgs_path, masks_path = img_mask_paths(img_dir, mask_dir)
X_train, X_temp, y_train, y_temp = train_test_split(imgs_path, masks_path,
train_size=0.90, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, train_size=0.5,
random_state=42)
print(f"{'Training:':<15}{len(X_train)}")
print(f"{'Validation:':<15}{len(X_val)}")
print(f"{'Testing:':<15}{len(X_test)}")
def map_fn(img_path, mask_path):
    img = tf.io.read_file(img_path)
    img = tf.image.decode_jpeg(img, channels=3)
    img = tf.image.resize(img, (IMG_SIZE, IMG_SIZE))
    img = tf.cast(img, tf.float32) / 255.0

    mask = tf.io.read_file(mask_path)
    mask = tf.image.decode_jpeg(mask, channels=1)
    mask = tf.image.resize(mask, (IMG_SIZE, IMG_SIZE))
    mask = tf.cast(mask, tf.float32) / 255.0

    return img, mask
# train_set
train_set = tf.data.Dataset.from_tensor_slices((X_train, y_train))
train_set = train_set.map(map_fn, num_parallel_calls=AUTOTUNE)
train_set =
train_set.shuffle(buffer_size=BUFFER_SIZE).batch(batch_size=BATCH_SIZE).pref
etch(buffer_size=AUTOTUNE)

# val_set
```

```
val_set = tf.data.Dataset.from_tensor_slices((X_val, y_val))
val_set = val_set.map(map_fn, num_parallel_calls=AUTOTUNE)
val_set =
val_set.shuffle(buffer_size=BUFFER_SIZE).batch(batch_size=BATCH_SIZE).prefe
tch(buffer_size=AUTOTUNE)

# test_set
test_set = tf.data.Dataset.from_tensor_slices((X_test, y_test))
test_set = test_set.map(map_fn, num_parallel_calls=AUTOTUNE)
test_set =
test_set.shuffle(buffer_size=BUFFER_SIZE).batch(batch_size=BATCH_SIZE).prefe
tch(buffer_size=AUTOTUNE)
# U-Net Model
def UNET():
    inputs = Input(shape=(IMG_SIZE, IMG_SIZE, 3))

    x = Conv2D(32, 3, strides=1, padding='same')(inputs)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)

    skip_connections = []

    # Encoder
    for filters in [64, 128, 256, 512]:
        x = Conv2D(filters, 3, strides=1, padding='same')(x)
        x = BatchNormalization()(x)
        x = Activation('relu')(x)

        x = Conv2D(filters, 3, strides=1, padding='same')(x)
        x = BatchNormalization()(x)
        x = Activation('relu')(x)

        skip_connections.append(x)
        x = MaxPooling2D(2, strides=2, padding='same')(x)

    # Bottleneck
    x = Conv2D(1024, 3, strides=1, padding='same')(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)

    x = Conv2D(1024, 3, strides=1, padding='same')(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
```

```
    # Decoder
    for filters in [512, 256, 128, 64]:
        x = Conv2DTranspose(filters, 2, strides=2, padding='same')(x)
        skip_connection = skip_connections.pop()
        x = add([x, skip_connection])

        x = Conv2D(filters, 3, strides=1, padding='same')(x)
        x = BatchNormalization()(x)
        x = Activation('relu')(x)

        x = Conv2D(filters, 3, strides=1, padding='same')(x)
        x = BatchNormalization()(x)
        x = Activation('relu')(x)

    outputs = Conv2D(1, 1, strides=1, activation='sigmoid')(x)

    model = Model(inputs=[inputs], outputs=[outputs])
    return model
model = UNET()
model.summary()
def dice_coefficient(y_true, y_pred, smooth=1):
    y_true_f = K.flatten(y_true)
    y_pred_f = K.flatten(y_pred)
    intersection = K.sum(y_true_f * y_pred_f)
    return (2. * intersection + smooth) / (K.sum(y_true_f) + K.sum(y_pred_f) +
smooth)

def dice_loss(y_true, y_pred):
    return 1 - dice_coefficient(y_true, y_pred)

def combined_loss(y_true, y_pred):
    bce_loss = BinaryCrossentropy()(y_true, y_pred)
    dice_l = dice_loss(y_true, y_pred)
    return bce_loss + dice_l
model.compile(optimizer=Adam(0.0002),
              loss=combined_loss, # Зміни тут
              metrics=['accuracy', Precision(name='precision'), Recall(name='recall'),
dice_coefficient])
checkpoint_cb = ModelCheckpoint('best_weights.weights.h5',
                                monitor='val_loss',
                                verbose=1,
                                save_best_only=True,
```

```
                        save_weights_only=True)

reduce_lr_cb = ReduceLROnPlateau(monitor='val_loss',
                         factor=0.1,
                         patience=5,
                         verbose=1)
history = model.fit(train_set, epochs=10, validation_data=val_set,
callbacks=[checkpoint_cb, reduce_lr_cb])
# save the model
model.save('skin-cancer-segmentation.keras')
# Load the model

import tensorflow as tf
from tensorflow.keras import backend as K

def dice_coefficient(y_true, y_pred, smooth=1):
    y_true_f = K.flatten(y_true)
    y_pred_f = K.flatten(y_pred)
    intersection = K.sum(y_true_f * y_pred_f)
    return (2. * intersection + smooth) / (K.sum(y_true_f) + K.sum(y_pred_f) +
smooth)

def dice_loss(y_true, y_pred):
    return 1 - dice_coefficient(y_true, y_pred)

loaded_model = tf.keras.models.load_model('/kaggle/input/unet-
ham10000/tensorflow2/default/1/skin-cancer-segmentation.keras',
custom_objects={'dice_coefficient': dice_coefficient})
loaded_model = UNET()
loaded_model.load_weights('/kaggle/input/weights/tensorflow2/default/1/best_weigh
ts.weights (1).h5')
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
plt.plot(history.history['dice_coefficient'], label='Training Dice Coefficient')
plt.plot(history.history['val_dice_coefficient'], label='Validation Dice Coefficient')
plt.xlabel('Epoch')
plt.ylabel('Dice Coefficient')
plt.legend()
plt.show()
```

```
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
loss, accuracy, precision, recall, dice_coefficient_score = model.evaluate(test_set)

print(f"Test Loss: {loss:.4f}")
print(f"Test Accuracy: {accuracy:.4f}")
print(f"Test Precision: {precision:.4f}")
print(f"Test Recall: {recall:.4f}")
print(f"Test Dice Coefficient: {dice_coefficient_score:.4f}")
print(f"{'Test Loss:':<25}{loss:.4f}")
print(f"{'Test Accuracy:':<25}{accuracy:.4f}")
print(f"{'Test Precision:':<25}{precision:.4f}")
print(f"{'Test Recall:':<25}{recall:.4f}")
print(f"{'Test Dice Coefficient:':<25}{dice_coefficient_score:.4f}")
n_imgs = 10
test_imgs, test_masks = next(iter(test_set))
y_pred = loaded_model.predict(test_imgs[:n_imgs], verbose=1)
def display_predictions(n, test_imgs, test_masks, y_pred):
    fig, axs = plt.subplots(3, n, figsize=(20, 8))
    for i in range(n):
        # Calculate Dice score
        score = dice_coefficient(test_masks[i], y_pred[i])

        # Display the test image
        axs[0, i].imshow(test_imgs[i])
        axs[0, i].set_title('Image')
        axs[0, i].axis('off')

        # Display the actual mask
        axs[1, i].imshow(test_masks[i], cmap='gray')
        axs[1, i].set_title('Actual Mask')
        axs[1, i].axis('off')

        # Display the predicted mask
        axs[2, i].imshow(y_pred[i], cmap='gray')
        axs[2, i].set_title(f'Predicted Mask\nDice: {score:.4f}')
        axs[2, i].axis('off')

    plt.tight_layout()  # Adjust layout to prevent overlap
```

```
    plt.show()
display_predictions(n=n_imgs,
                    test_imgs=test_imgs[:n_imgs],
                    test_masks=test_masks[:n_imgs],
                    y_pred=y_pred)
```

ДОДАТОК В (ОБОВ'ЯЗКОВИЙ)

ІЛЮСТРАТИВНА ЧАСТИНА

«ПРОГРАМНИЙ МОДУЛЬ РОЗПІЗНАВАННЯ РАКУ ШКІРИ»

Виконав: студент 4 курсу,групи ЗКН-21б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Трегуб А. К.
(прізвище та ініціали)

Керівник: к.т.н., проф. каф.КН Арсенюк І. Р.
(прізвище та ініціали)
«03» червня 2025 р.

Вінниця ВНТУ - 2025 рік

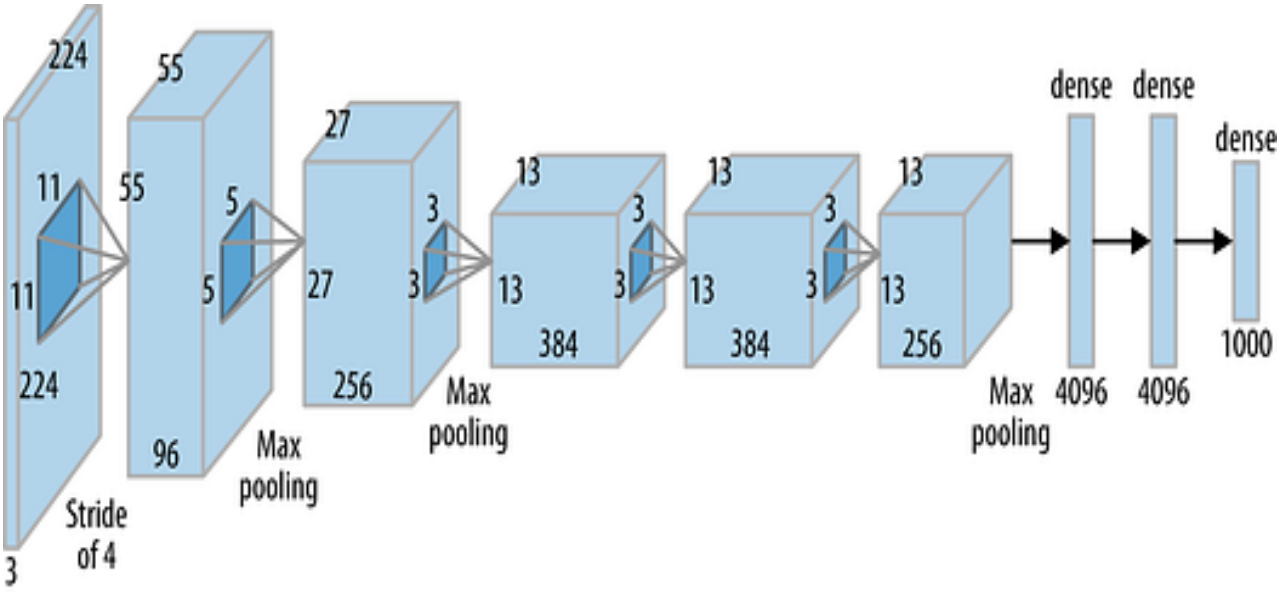

Рисунок В.1 – Архітектура згорткової нейронної мережі AlexNet

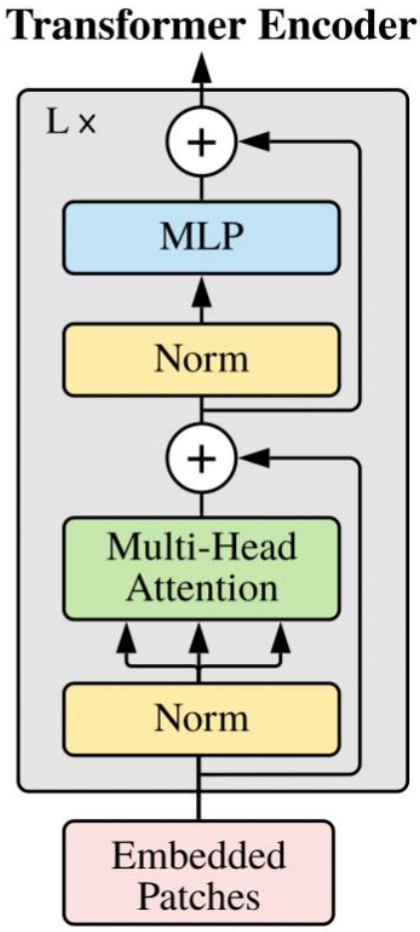

Рисунок В.2 – Блок трансформера в ViT

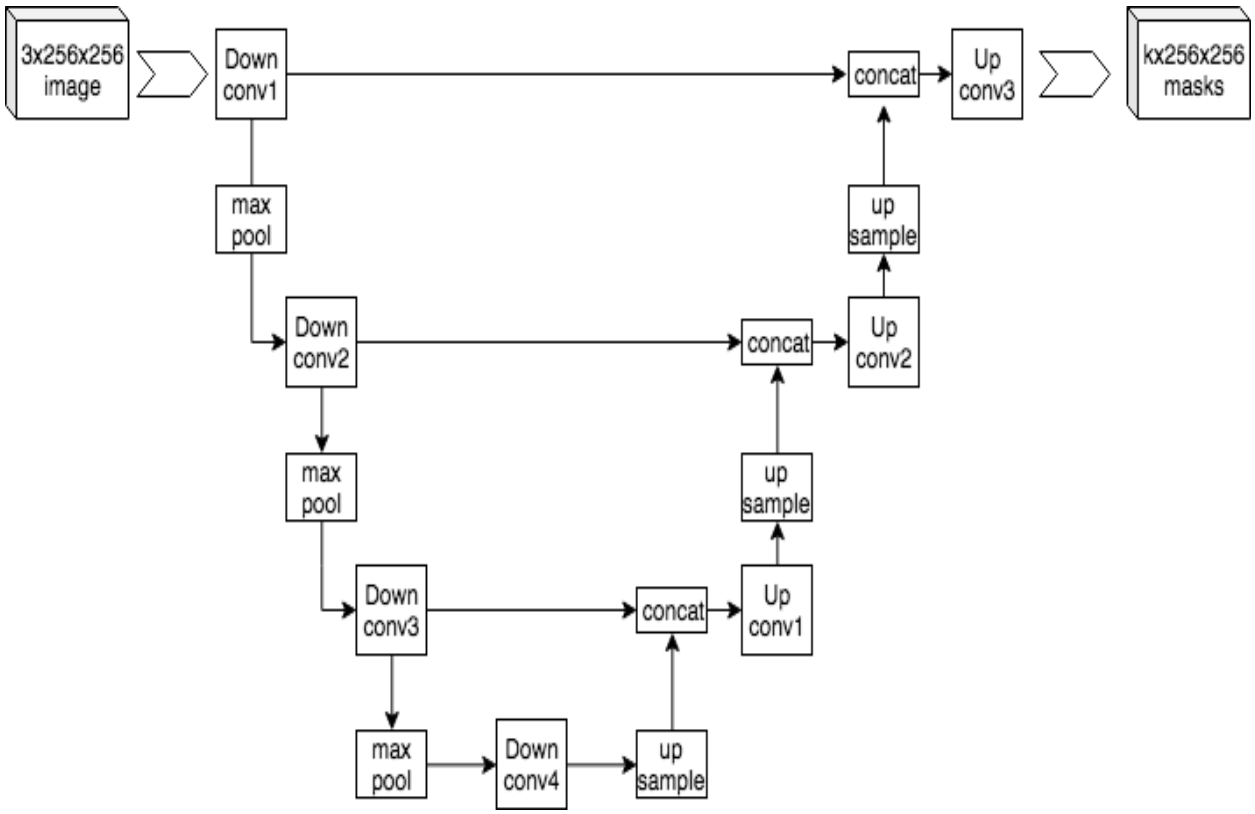

Рисунок В.3 — Архітектура U-Net для сегментації зображення

Рисунок В.4 – Основна структурна схема роботи програмного модуля ропізнавання раку шкіри

Рисунок В.5 – Загальний алгоритм роботи програмного модуля

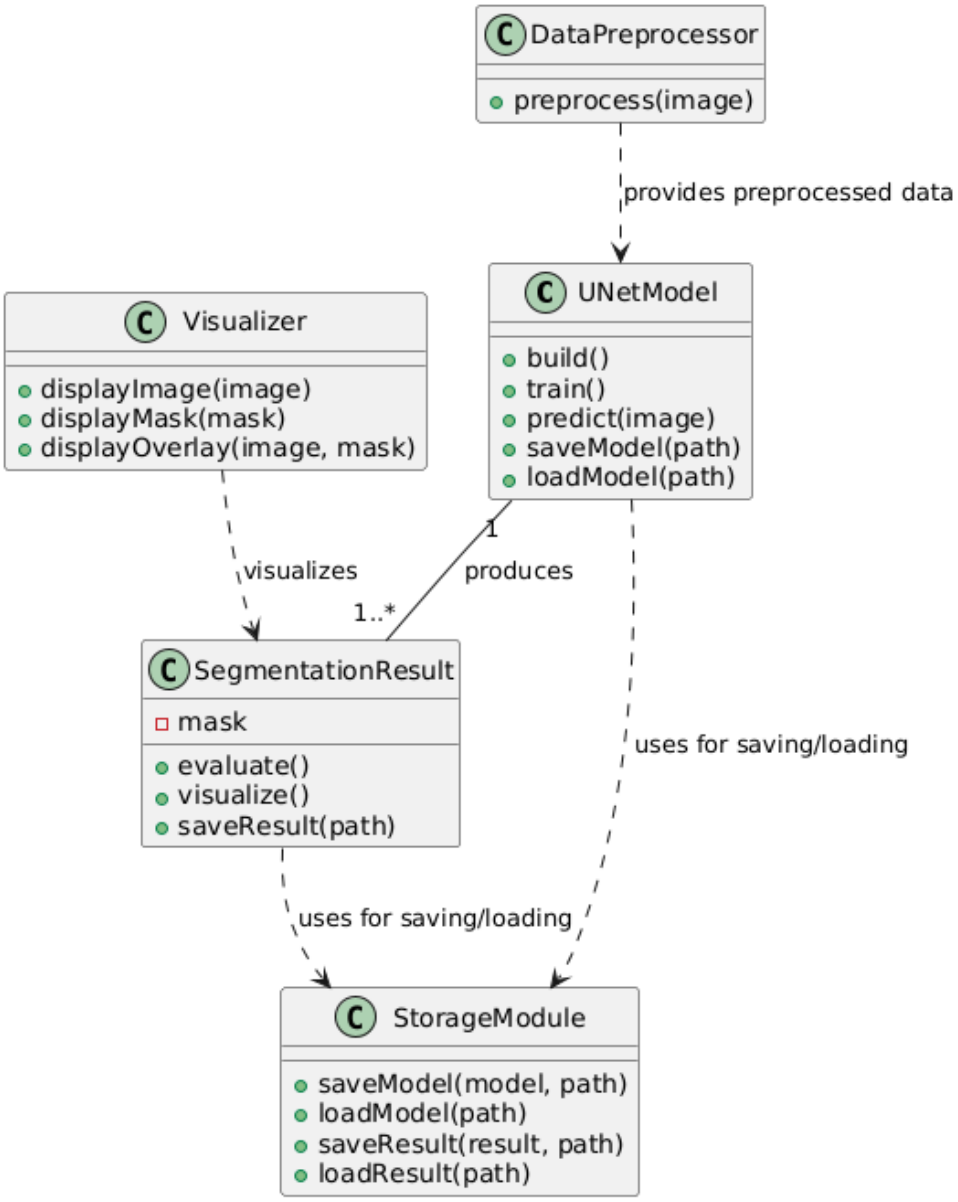

Рисунок В.6 – UML-діаграма класів програмного модуля розпізнавання раку шкіри

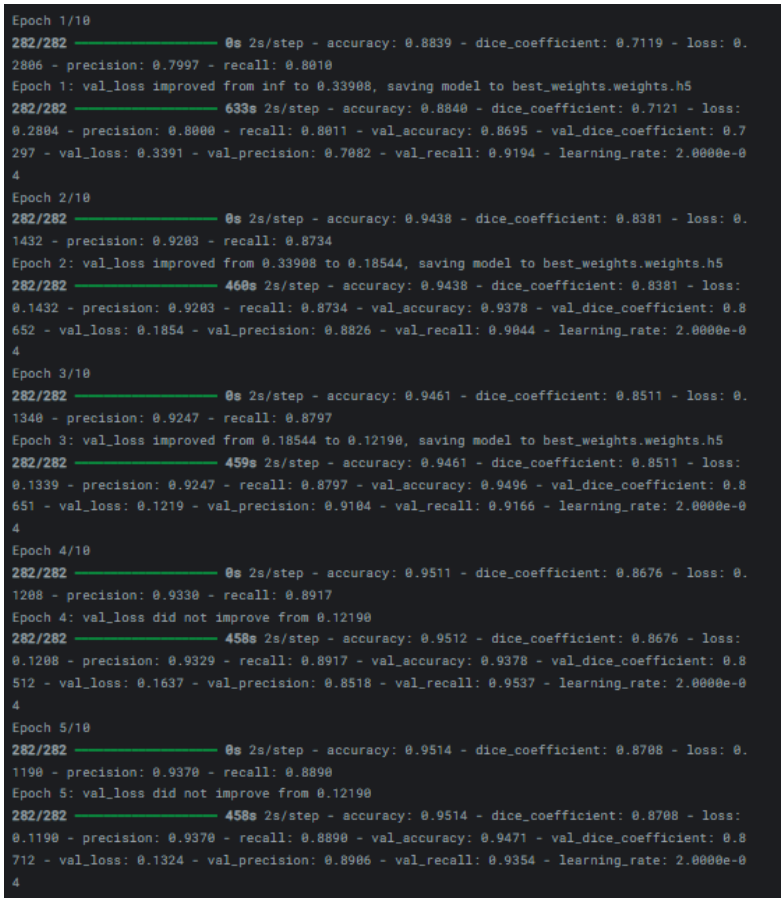

Рисунок В.7 – Точність на етапах навчання та перевірки

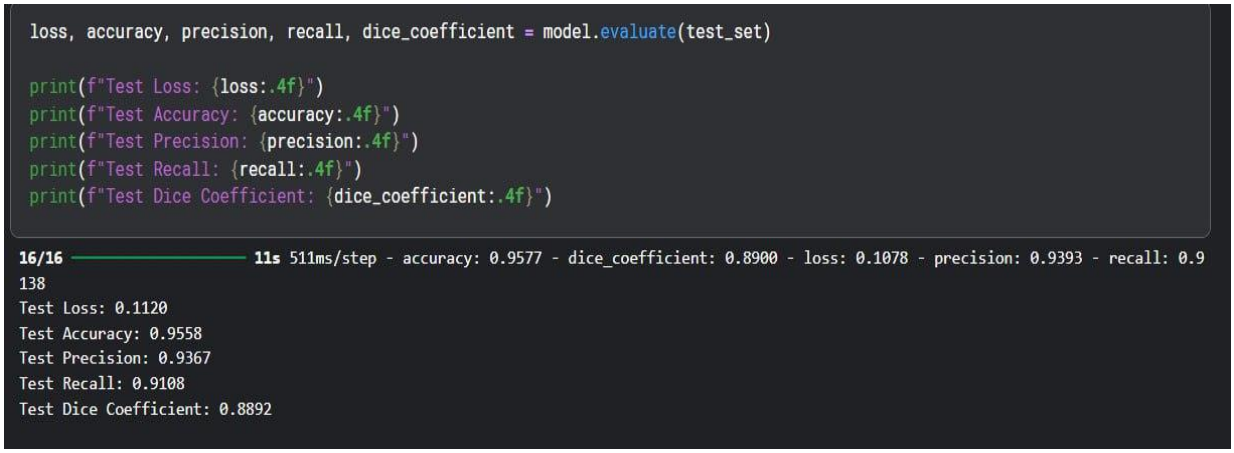

Рисунок В.8 – Результати створеної моделі для розпізнавання раку шкіри

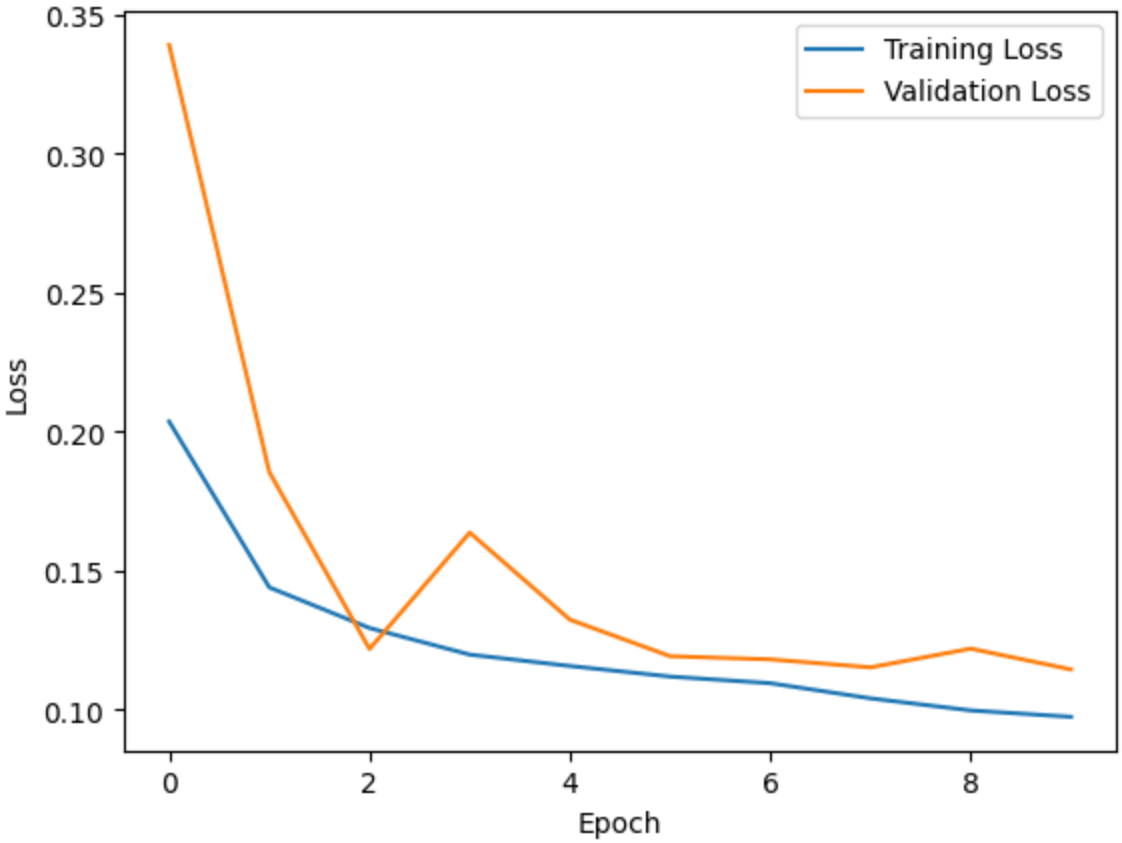

Рисунок В.9 – Графік зміни функції втрат протягом 10 епох

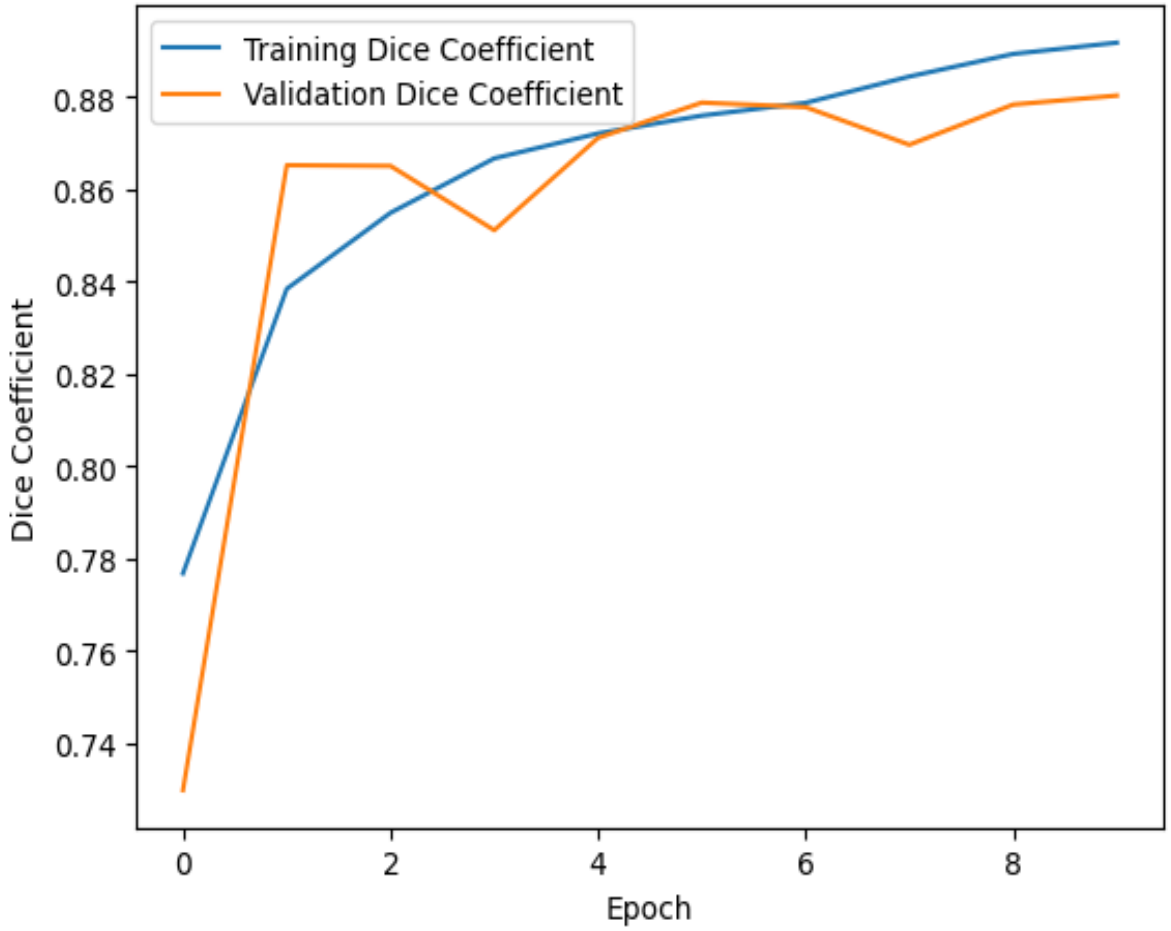

Рисунок В.10 – Графік зміни коефіцієнта Dice протягом 10 епох

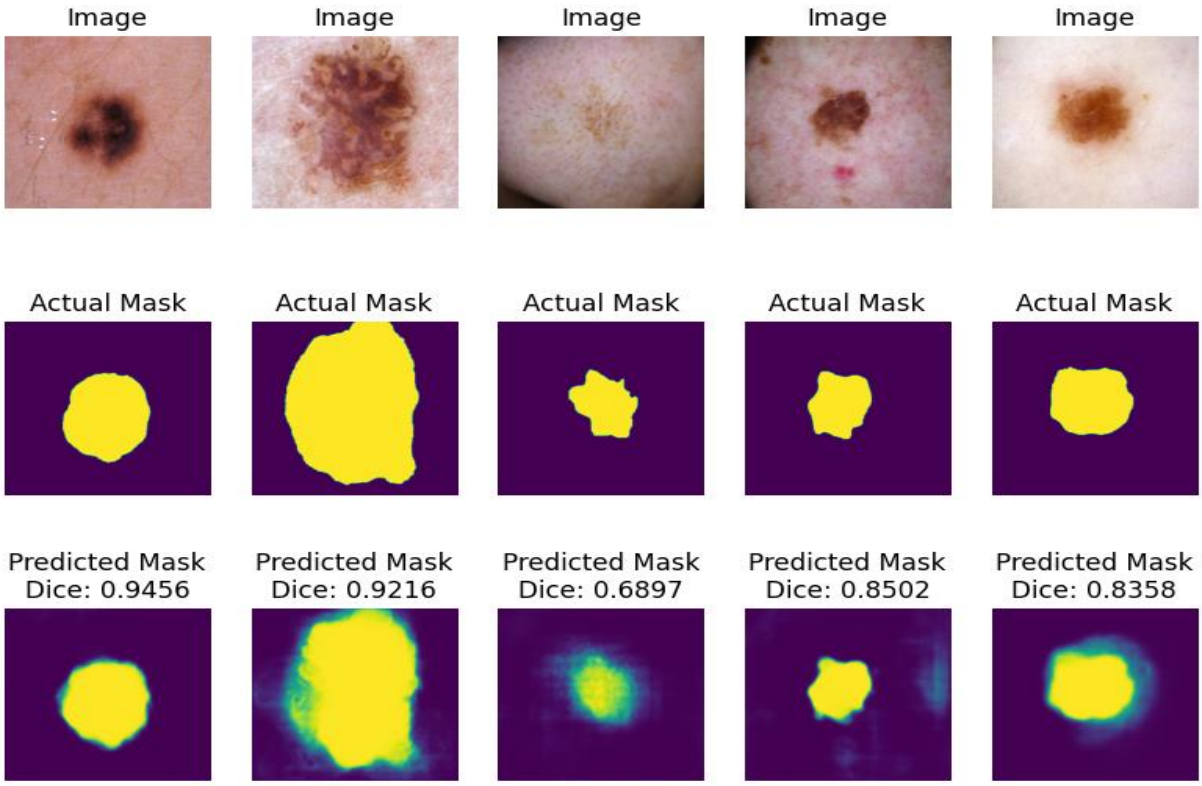

Рисунок В.11 – Візуалізація результатів сегментації зображень

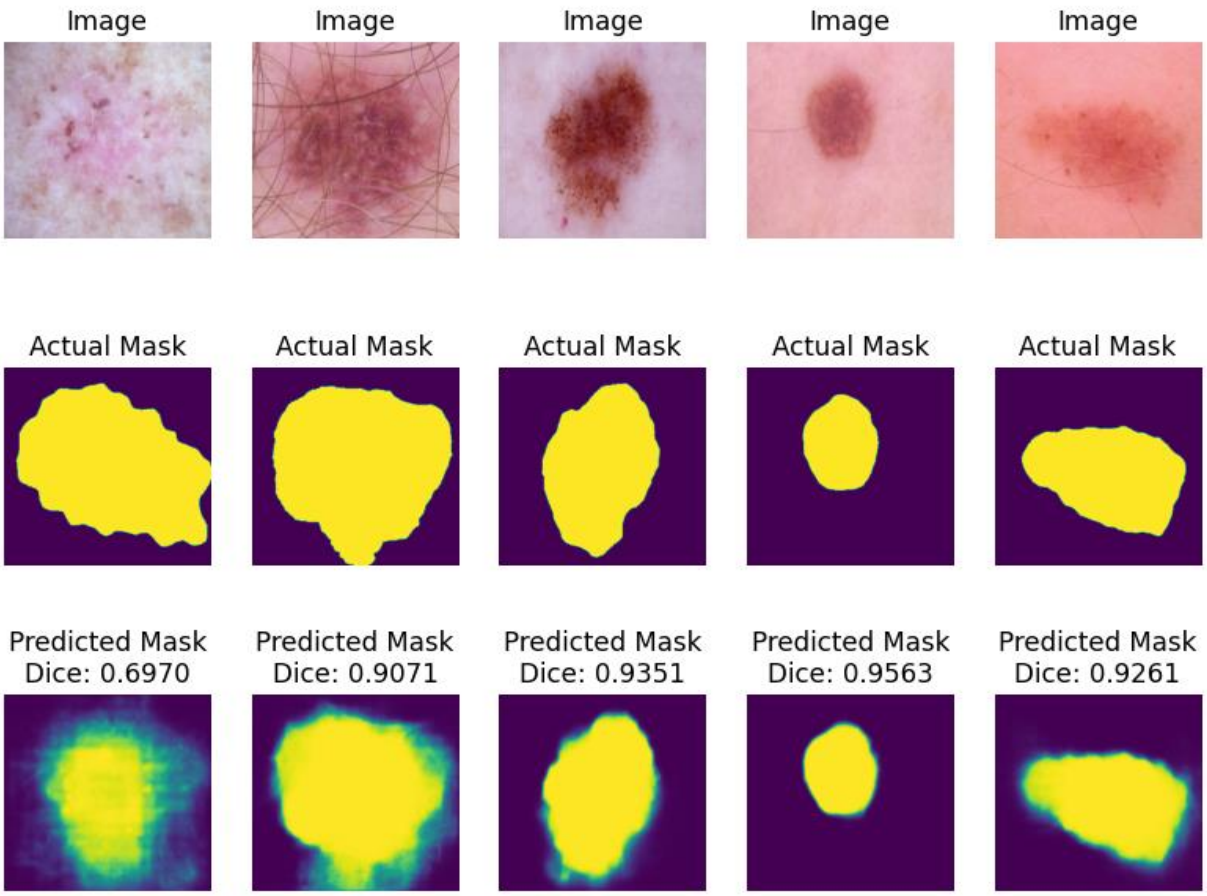

Рисунок В.12 – Візуалізація результатів сегментації зображень