Міністерство освіти і науки України
Криворізький національний університет
Факультет інформаційних технологій
Кафедра автоматизації, комп'ютерних наук і технологій

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти − магістр
за освітньо-професійною програмою
«Компь'ютерні науки»

зі спеціальності
«122 – Компь'ютерні науки»

тема роботи:

«Розробка інформаційної системи для сегментації зображення із використанням методів штучного інтелекту»

Виконав студент гр. КН-24м _____ Кухар Д.О.

Керівник _____ Купін А.І.

Нормоконтроль _____ Маринич І. А.

Завідувач кафедри _____ Рубан С. А.

Кривий Ріг – 2025

КРИВОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет: *інформаційних технологій*

Кафедра: *автоматизації, комп'ютерних наук і технологій*

Ступінь вищої освіти: *магістр*

Спеціальність: *122 - Компь'ютерні науки*

ЗАТВЕРДЖУЮ

Зав. кафедрою: *к.т.н. Рубан С.А.*

_____

« *15* » *травня* 2025р.

ЗАВДАННЯ

на кваліфікаційну роботу магістра

студентові групи КН-24М *Кухару Дмитру Олександровичу*

1. Тема роботи: «*Розробка інформаційної системи для сегментації зображення із використанням методів штучного інтелекту»*

затверджено наказом по університету № *257с* від *13.05.2025р*.

2. Термін здачі завершеної роботи « » 01.12 2025 р.

3. Склад кваліфікаційної роботи: *Пояснювальна записка обсягом 72с., додатки, презентація у Microsoft PowerPoint (27 слайдів) в електронному та друкованому вигляді*

4. Консультанти кваліфікаційної роботи:

Розділ 1-3 д.т.н. Купін А.І.

_____

Нормоконтроль доц. Маринич І. А

5. Календарний план:

№	Етапи роботи	Термін виконання
1	*Вступ*	*10.02.25*
2	*Розділ 1*	*15.04.25*
3	*Розділ 2*	*18.07.25*
4	*Розділ 3*	*19.11.25*
5	*Висновки*	*20.11.25*
6	*Оформлення кваліфікаційної роботи*	*25.11.25*
7	*Підготовка презентації та графічного матеріалу*	*28.11.25*
8	*Підготовка доповіді до захисту*	*01.12.25*

6. Дата видачі завдання: *15.05.2025р.*

Керівник _____ /Купін А. І./

7. Запевнення: *Я, Кухар Дмитро Олександрович, запевняю, що ця кваліфікаційна робота виконана самостійно, не містить академічного плагіату, фабрикації, фальсифікації. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.*

Із чинним Положенням про академічну доброчесність Криворізького національного університету ознайомлений.

Чітко усвідомлюю, що в разі виявлення у кваліфікаційній роботі умисних порушень робота не допускається до захисту або оцінюється незадовільно.

Здобувач _____ /Кухар Д. О./

АНОТАЦІЯ

Кухар Д. О. «*Розробка інформаційної системи для сегментації зображення із використанням методів штучного інтелекту*»: Кваліфікаційна робота магістра: 122 – Комп’ютерні науки. Кривий Ріг. Криворізький національний університет, 2025 75с.

Об’єктом дослідження є цифрові зображення, які виступають основним джерелом даних для задач сегментації.

Мета дослідження полягає у створенні високоточної та гнучкої системи сегментації зображень, придатної для роботи з різними типами даних та подальшої інтеграції у прикладні програмні комплекси.

У першому розділі проведено аналіз сучасних підходів до сегментації зображень, розглянуто класичні методи, архітектури глибокого навчання та трансформерні моделі. Проаналізовано літературні джерела, патенти, а також виявлено переваги й недоліки існуючих рішень.

У другому розділі розроблено математичну та алгоритмічну модель сегментації зображень. Сформульовано постановку задачі, наведено формальний опис алгоритму, комбіновану функцію втрат, блок-схему процесу обробки та систему метрик оцінювання.

У третьому розділі здійснено програмну реалізацію інформаційної системи. Представлено архітектуру системи, обґрунтовано вибір програмних засобів, проведено навчання та тестування моделей, а також оцінку ефективності роботи із застосуванням сучасних метрик сегментації.

Ключові слова: СЕГМЕНТАЦІЯ ЗОБРАЖЕНЬ, ШТУЧНИЙ ІНТЕЛЕКТ, ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ, ТРАНСФОРМЕРИ, U-NET, DEEPLAB, SEGMENT ANYTHING, ІНФОРМАЦІЙНА СИСТЕМА.

ANNOTATION

Kukhar D. O. "*Development of an Information System for Image Segmentation Using Artificial Intelligence Methods*"**:** Master's qualification thesis: 122 – Computer Science. Kryvyi Rih. Kryvyi Rih National University, 2025. – 75 p.

The object of research is digital images, which serve as the main source of data for segmentation tasks.

The aim of the study is to develop a highly accurate and flexible image segmentation system suitable for processing various types of data and for further integration into applied software solutions.

The first chapter provides an analysis of modern approaches to image segmentation, considering classical methods, deep learning architectures, and transformer-based models. Literature sources and patents are reviewed, and the advantages and disadvantages of existing solutions are identified.

The second chapter presents the mathematical and algorithmic model of image segmentation. The problem statement is formulated, and the chapter provides a formal description of the algorithm, a combined loss function, a processing flowchart, and a comprehensive system of evaluation metrics.

The third chapter focuses on the software implementation of the information system. The system architecture is presented, the choice of software tools is justified, training and testing of models are performed, and the efficiency of the system is evaluated using modern segmentation metrics.

Keywords: IMAGE SEGMENTATION, ARTIFICIAL INTELLIGENCE, CONVOLUTIONAL NEURAL NETWORKS, TRANSFORMERS, U-NET, DEEPLAB, SEGMENT ANYTHING, INFORMATION SYSTEM.

ЗМІСТ

ВСТУП

Сегментація зображень є однією з ключових задач у галузі комп'ютерного зору та штучного інтелекту. Вона передбачає поділ цифрового зображення на окремі області або об'єкти з метою виділення суттєвої інформації для подальшої обробки та аналізу. Ефективна сегментація дозволяє автоматизувати процеси розпізнавання, діагностики, класифікації та прогнозування у різних предметних галузях — від медицини до автономного транспорту і геоінформаційних систем.

Актуальність дослідження зумовлена стрімким розвитком інформаційних технологій, зростанням обсягів цифрових даних та потребою у швидкому й точному аналізі візуальної інформації. Традиційні методи сегментації (порогові алгоритми, кластеризація, морфологічні операції) залишаються ефективними для простих задач, однак вони не забезпечують належної точності у випадках, коли зображення мають високий рівень шумів, складні структури чи розмиті межі. У таких умовах провідні позиції займають методи глибокого навчання, зокрема згорткові нейронні мережі (U-Net, SegNet, DeepLab) та сучасні архітектури на основі трансформерів (Vision Transformer, Swin Transformer, Segment Anything Model).

Розробка інформаційної системи для сегментації зображень на основі методів штучного інтелекту дозволяє поєднати алгоритмічну точність з практичною цінністю, забезпечуючи гнучкість, масштабованість та універсальність рішень. Подібні системи мають широке застосування:

- у медицині — для автоматизованого виділення органів, пухлин чи патологій на КТ та МРТ-знімках;
- у транспорті — для розпізнавання дорожніх знаків, пішоходів і транспортних засобів в системах допомоги водієві та автопілотах;
- у геоінформаційних системах — для аналізу супутникових та аерофотознімків з метою ідентифікації будівель, лісів, водойм та інших об'єктів;

- у промисловості — для контролю якості продукції та виявлення дефектів обладнання.

Об’єктом дослідження є цифрові зображення як джерело інформації для задач сегментації.

Предметом дослідження є методи сегментації зображень із використанням технологій штучного інтелекту.

Метою дослідження є розробка інформаційної системи для сегментації зображень, яка забезпечує підвищену точність, адаптивність до різних типів вхідних даних і можливість інтеграції у прикладні програмні комплекси.

Для досягнення поставленої мети необхідно вирішити такі завдання:

1) Проаналізувати сучасні літературні джерела, патенти та існуючі інформаційні системи сегментації зображень.
2) Визначити недоліки та переваги класичних і сучасних методів сегментації.
3) Розробити математичну модель та алгоритмічне забезпечення сегментаційної системи.
4) Обґрунтувати вибір архітектури нейронної мережі для вирішення поставленої задачі.
5) Реалізувати програмний прототип інформаційної системи.
6) Провести експериментальні дослідження та оцінку ефективності розробленої системи за допомогою сучасних метрик (IoU, Dice Precision, Recall, Accuracy).
7) Надати рекомендації щодо подальшого удосконалення системи та її практичного застосування.

Методи дослідження включають аналіз літературних джерел, математичне моделювання, методи машинного та глибинного навчання (зокрема CNN і трансформерні архітектури), експериментальне тестування на відомих наборах даних (RSSS Dataset, Crack Segmentation Dataset, Brain Tumor тощо), а також методи статистичної оцінки якості результатів.

Наукова новизна полягає у поєднанні сучасних архітектур глибокого навчання з елементами трансформерних моделей, що забезпечує підвищення точності сегментації та стійкість до шумів і артефактів.

Практичне значення роботи полягає у створенні прототипу інформаційної системи, яку можна адаптувати для різних предметних галузей. Результати дослідження можуть бути використані у медичних діагностичних системах, транспортних технологіях, та геоінформаційних платформах.

РОЗДІЛ 1
АНАЛІЗ СУЧАСНИХ ПІДХОДІВ ДО СЕГМЕНТАЦІЇ ЗОБРАЖЕНЬ

1.1 Теоретичні основи та постановка задачі сегментації зображень

Сегментація зображень є фундаментальною задачею у галузі комп'ютерного зору та штучного інтелекту, що передбачає поділ цифрового зображення на окремі області або об'єкти відповідно до певних критеріїв однорідності [1; 3]. Основна мета сегментації полягає у виділенні структурованої інформації для подальшої обробки, класифікації чи аналізу. Вона виступає базовим етапом у системах медичної діагностики, транспортних технологіях, промисловому контролі якості та геоінформаційних системах [2; 4; 5].

У загальному вигляді цифрове зображення можна подати як матрицю пікселів:

$$I = \{p_{i,j} \mid 1 \leq i \leq M,\ 1 \leq j \leq N\}, \tag{1.1}$$

де $p_{i,j}$ – інтенсивність (або вектор кольорових компонентів) пікселя з координатами $(i, j), M \times N$ – розмірність зображення [3].

Задача сегментації полягає у побудові відображення

$$f : I \to L, \tag{1.2}$$

Де $L = \{l_1, l_2, \dots, l_K\}$ – множина міток класів, що відповідають областям зображення. Таким чином, кожному пікселю зображення ставиться у відповідність певний клас l_k [1].

З точки зору машинного навчання, сегментація є задачею класифікації кожного пікселя, тобто визначення ймовірності належності пікселя до одного з класів. Формально ця задача може бути описана через функцію втрат, яка мінімізує різницю між передбаченими та істинними мітками [12; 15]:

$$\mathcal{L}(\theta) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} y_{i,k} \cdot \log\left(\widehat{y_{i,k}}\right), \tag{1.3}$$

де

– N – кількість пікселів у зображенні,

– $y_{i,k}$ – істинна мітка пікселя (one-hot представлення),

– $\widehat{y_{i,k}}$ – прогнозована ймовірність належності пікселя до класу kkk,

– θ – параметри моделі, що оптимізуються під час навчання.

Схематичне зображення процесу сегментації, наведено на рисунку 1.1

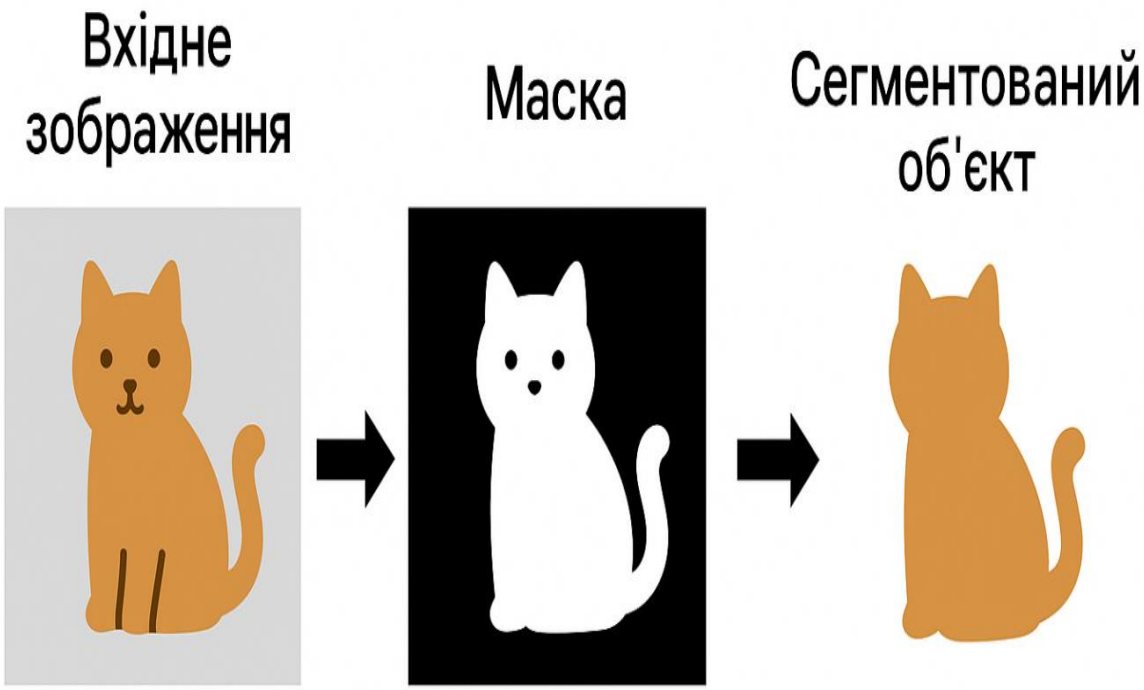


Рисунок 1.1 – Схематичне представлення задачі сегментації

Класифікація методів сегментації

Методи сегментації умовно можна поділити на три великі групи (рис. 1.2):

1) Класичні алгоритмічні методи – базуються на обробці інтенсивностей, контурів чи регіонів (порогова сегментація, кластеризація, алгоритм вододілу, графові методи) [3; 11].
2) Методи на основі глибокого навчання – застосування згорткових нейронних мереж (Fully Convolutional Networks [15], U-Net [12], SegNet [21], DeepLab [16], Mask R-CNN [28]).
3) Трансформерні та гібридні підходи – архітектури, що враховують глобальний контекст за допомогою механізмів самоуваги (Vision

Transformer [14], Swin Transformer [22], Segment Anything Model [23], Mask2Former [24]).

Рисунок 1.2 – Класифікація методів сегментації зображень

Класичні методи є відносно простими у реалізації й не потребують значних обчислювальних ресурсів [2; 3]. Однак вони чутливі до шумів та зміни освітлення, що обмежує їх застосування у складних задачах. Методи глибокого навчання забезпечують значно вищу точність за рахунок багаторівневого виділення ознак [13; 17], але вимагають великих обсягів даних і високопродуктивного обладнання. Сучасні трансформерні архітектури поєднують гнучкість і універсальність [14; 22], проте залишаються обчислювально затратними й складними у налаштуванні [23; 24].

Метрики оцінки якості сегментації

Для кількісного вимірювання ефективності моделей сегментації застосовують низку метрик. Найпоширенішими є [4; 7]:

– IoU (Intersection over Union):

$$IoU = \frac{|A \cap B|}{|A \cup B|} \tag{1.4}$$

де AAA – передбачена маска, BBB – істинна маска.

– Коефіцієнт Dice:

$$Dice = \frac{2|A \cap B|}{|A| + |B|} \tag{1.5}$$

– Pixel Accuracy (PA):

$$PA = \frac{\text{кількість правильно класифікованих пікселів}}{\text{загальна кількість пікселів}} \tag{1.6}$$

– Precision (точність):

$$Precision = \frac{TP}{TP + FP} \tag{1.7}$$

Де TP— істинно позитивні пікселі, FP— хибно позитивні пікселі.

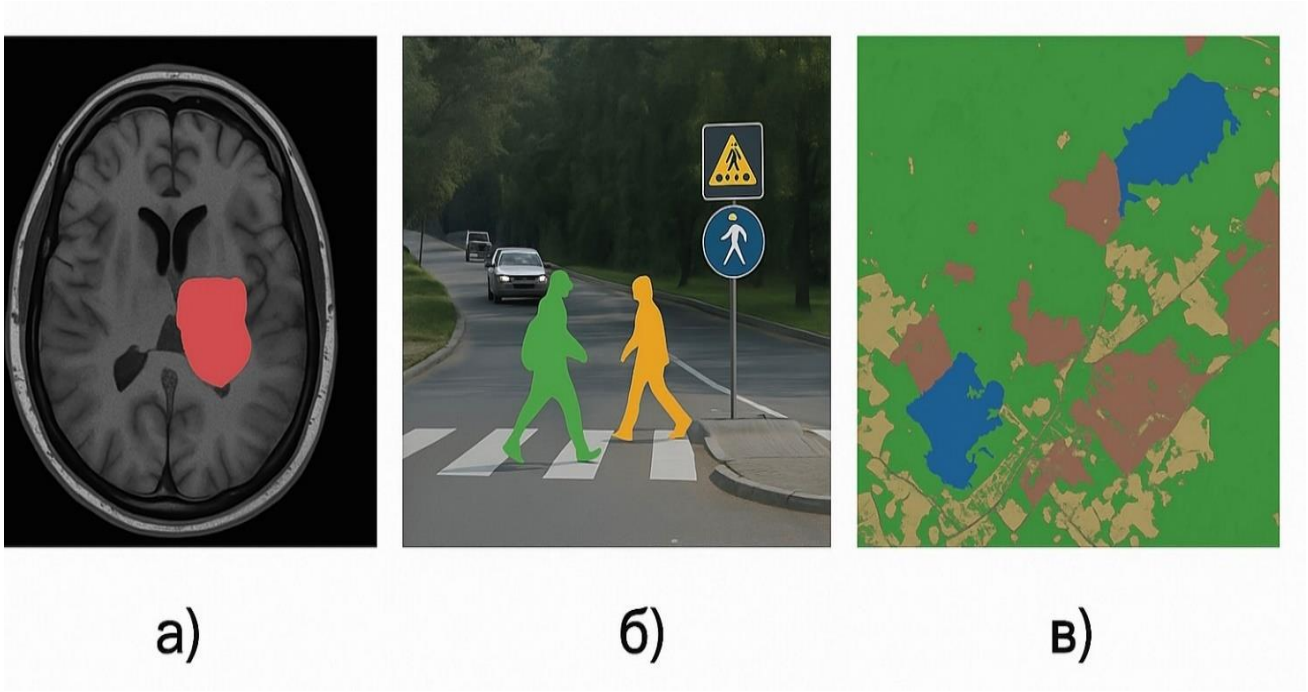
– Recall (повнота):

$$Recall = \frac{TP}{TP + FN} \tag{1.8}$$

де FN— хибно негативні пікселі.

Приклад застосування

У медичній діагностиці сегментація дозволяє автоматично виділяти органи чи пухлини на МРТ-знімках, що суттєво зменшує час аналізу лікарем [7; 12; 18; 29]. У транспортних системах вона використовується для розпізнавання пішоходів і дорожніх знаків у режимі реального часу [9; 15; 17]. У геоінформаційних системах сегментація супутникових знімків допомагає ідентифікувати ліси, водойми чи забудовані території . Приклади застосування сегментації зображення подано на рис. 1.3 [10; 30].



а) медичні знімки; б) транспортні системи; в) супутникові дані.

Рисунок 1.3 – Приклади застосування сегментації зображень

Отже, сегментація зображень є ключовим етапом у побудові інтелектуальних систем комп’ютерного зору, оскільки забезпечує структуроване представлення даних для подальшого аналізу. Різноманіття методів від класичних алгоритмів до сучасних нейромережевих і трансформерних моделей дозволяє адаптувати підхід під конкретну задачу та вимоги до точності [1; 3; 14; 24]. Вибір оптимального методу залежить від типу даних, умов застосування та необхідного рівня узагальнення, що формує основу для подальшого практичного дослідження.

1.2 Аналіз наукових джерел та існуючих методів сегментації

Найперші дослідження у сфері сегментації були зосереджені на простих математичних операціях над зображеннями. Порогові методи (Otsu, 1979) [31] передбачали вибір інтенсивності, за якою відбувався поділ об’єкта від фону. У випадках, коли об’єкти мали чіткі межі та однорідні області, цей метод демонстрував високу швидкість та простоту. Проте навіть незначні шуми чи

варіації освітлення призводили до суттєвих помилок. Подібний підхід детально розглянуто у роботі Творошенка [3], де описані порогові алгоритми, методи виділення контурів і морфологічні операції. Приклад вхідного зображення, що використовується для демонстрації роботи таких методів, подано на рисунку 1.4.

Інший напрям — методи на основі контурів. Використання операторів Собеля, Кенні або Лапласа дозволяло виявляти межі об’єктів за градієнтами інтенсивності. Однак контурні методи мали схожий недолік — чутливість до шуму, що потребувало додаткових фільтраційних кроків. Ряд таких підходів систематизовано у вітчизняних дослідженнях Андрієнка [1] та Подчашинського [2].

Методи сегментації на основі регіонів (region growing, split-and-merge) працювали шляхом поступового нарощування областей за критерієм однорідності. Цей підхід виявився ефективним для простих сцен, проте мав проблеми зі складними структурами. Аналіз таких підходів наведено у праці Луп’яка [11], де наголошено на ролі графових методів та суперпіксельних моделей.

Більш пізні алгоритми, такі як вододіл (Watershed), будувались на аналізі морфології зображення. Вони забезпечували сегментацію навіть у складних випадках, але часто страждали від ефекту “пересегментації”. Приклади їх практичної реалізації наведені у матеріалах з використанням OpenCV [6]. Приклади результатів роботи класичних методів сегментації наведені на рисунку 1.5.

У наукових публікаціях 1990-х років почали активно застосовуватись графові методи, де зображення представлялось у вигляді графа, а сегментація зводилась до задачі мінімізації енергії. Такі підходи стали попередниками сучасних нейронних методів, оскільки вони також формулювались через оптимізаційні задачі [11, 5].

Рисунок 1.4 – Зображення до сегментації

а) пороговий метод; б) алгоритм K-means; в) вододіл.

Рисунок 1.5 – Приклади результатів класичних методів сегментації

Методи глибокого навчання

З початком “золотого десятиліття” глибинного навчання (2012–2022 рр.) сегментація зображень отримала новий поштовх. У 2015 році Long, Shelhamer та Darrell представили Fully Convolutional Networks (FCN) [15], які замінили щільні

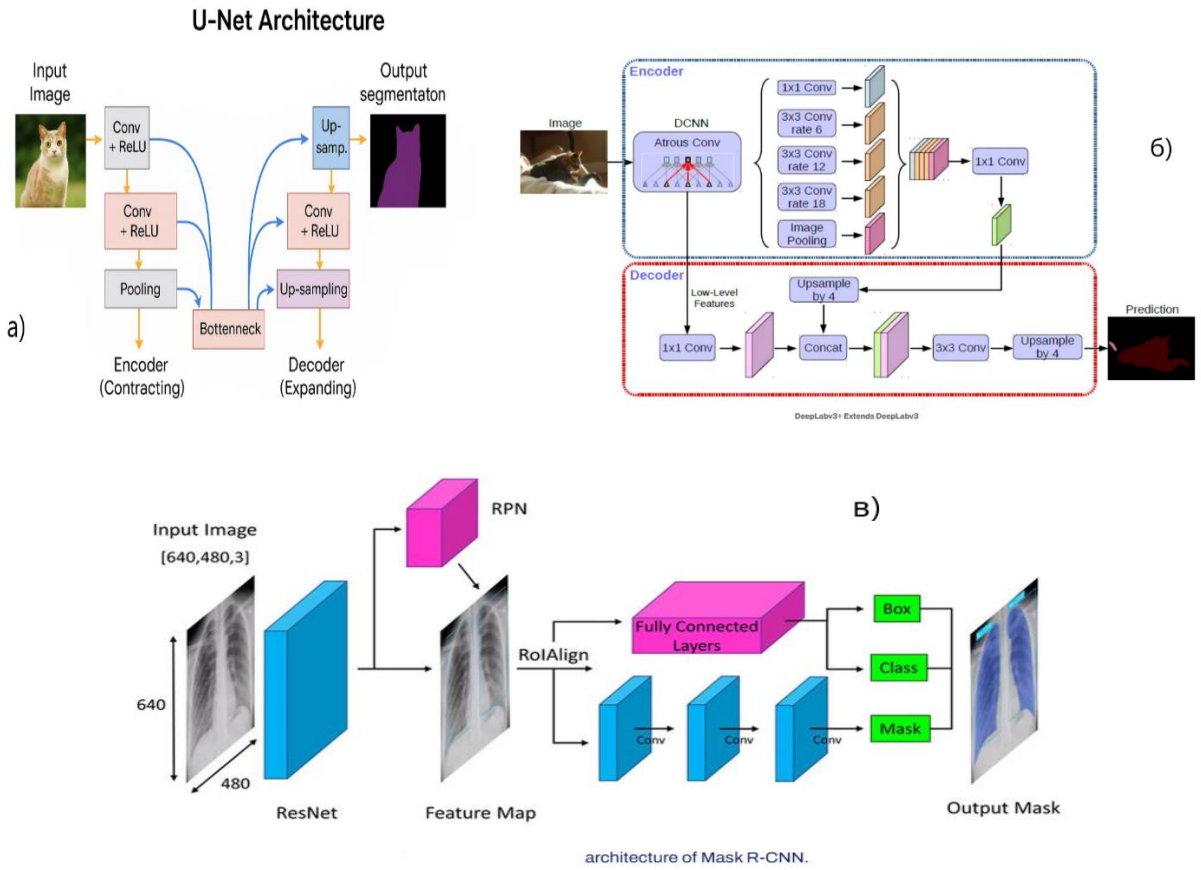
шари у класичних CNN на згорткові, що дало змогу виконувати піксельну класифікацію. Ця робота стала основою для наступних архітектур.

Особливої популярності набули архітектури U-Net (Ronneberger, 2015 [12]) та її модифікації, наприклад UNet++ (Zhou, 2018 [18]) чи Attention U-Net. Завдяки симетричній структурі “кодер-декодер” та механізму skip connections, U-Net демонструвала високу точність у медичній діагностиці. Подальші дослідження (Oktay O., 2018; Zhou Z., 2018 [18]) довели, що додаткові механізми уваги значно підвищують якість виділення дрібних структур. У вітчизняних роботах Коменчука [8] та Андрікевича [7] ці методи розглядаються у контексті сегментації медичних зображень, зокрема очного дна та томографічних даних.

Ще один напрям — DeepLab (Chen L.-C., 2017–2020 [16]), де застосовано атрозні згортки для збереження контексту на різних масштабах. Ця модель виявилась особливо корисною у задачах семантичної сегментації міських сцен (наприклад, для автономного транспорту). У вітчизняному дослідженні Рупіча [10] аналогічні методи (DeepLabV3, U-Net) застосовуються для сегментації супутникових знімків інфраструктурних об’єктів.

Для одночасного виконання детекції та сегментації була запропонована архітектура Mask R-CNN (He K., 2017 [13]). Вона стала стандартом у багатьох прикладних задачах, проте мала велику обчислювальну складність. Приклади найбільш відомих архітектур глибинних нейронних мереж для сегментації зображень — наведено на рисунку 1.6.

Проблемою всіх CNN-архітектур залишалась необхідність у величезних масивах розмічених даних. Для медичних або супутникових знімків отримати їх часто складно. Це стимулювало пошук нових архітектур, здатних узагальнювати знання з менших вибірок. Одним із перспективних підходів стало контрастивне навчання (SimCLR, Chen T., 2020 [19]), яке дозволяє покращувати стійкість моделей навіть при обмеженому обсязі даних.



а) U-Net; б) DeepLab; в) Mask R-CNN.

Рисунок 1.6 – Приклади архітектур CNN для сегментації

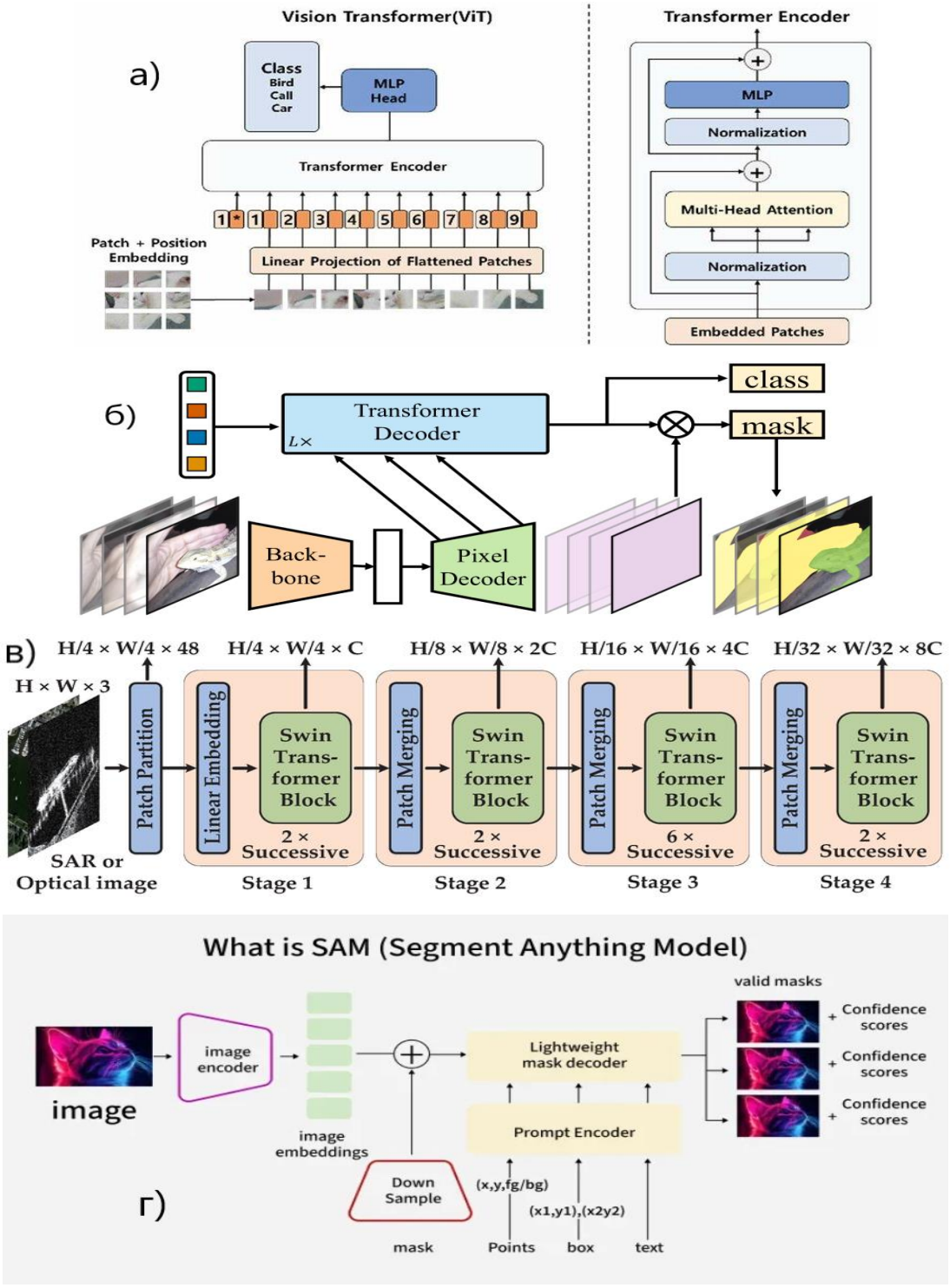
Трансформерні архітектури

Поява Vision Transformer (ViT) (Dosovitskiy A., 2020 [14]) стала переломним моментом у комп’ютерному зорі. Замість локальних згорткових фільтрів модель працювала з патчами зображення, що дозволяло враховувати глобальний контекст.

Подальші модифікації — Swin Transformer (Liu Z., 2021 [22]) та SegFormer (Xie E., 2021) — забезпечили ще вищу ефективність завдяки ієрархічним підходам і багаторівневим механізмам уваги. Уніфіковану модель для семантичної, інстанс- та паноптичної сегментації запропоновано в Mask2Former (Cheng B., 2022 [24]).

Окрему увагу слід приділити моделі Segment Anything (SAM), представленій у 2023 році компанією Meta AI [23]. SAM вперше запропонувала універсальну архітектуру, здатну виконувати сегментацію для будь-якого

зображення практично без додаткового навчання. Вона стала проривом, однак і тут існують проблеми: значні апаратні вимоги та необхідність тонкого налаштування при роботі зі специфічними даними. Приклади найвідоміших трансформерних архітектур, наведені на рисунку 1.7.



а) Vision Transformer; б) Swin Transformer; в) Mask2Former; г) SAM.

Рисунок 1.7 – Приклади архітектур трансформерних моделей (Vision Transformer, SAM).

Огляд патентної літератури

Важливим джерелом інформації про практичне застосування методів сегментації є патентні публікації, які відображають сучасні тенденції та рівень розвитку технологій у різних сферах.

У патенті **US11250569B2** (2022) [25] описано систему сегментації біомедичних зображень із використанням архітектури U-Net, збагаченої вкладеними рівнями та механізмами самоуваги. Основна увага приділяється зменшенню помилок при сегментації дрібних об’єктів, що є критично важливим у медичній діагностиці.

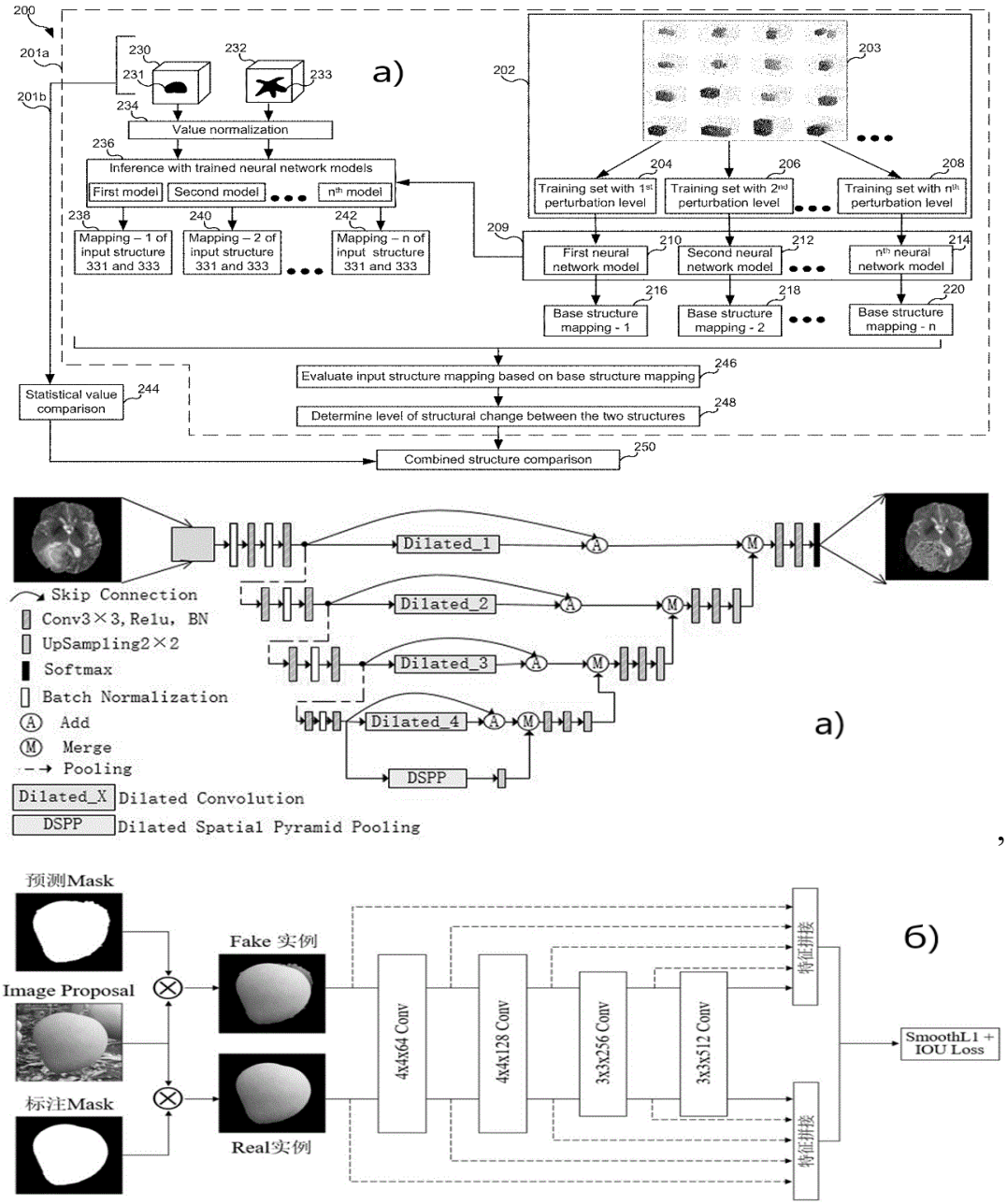
У патенті **CN107945185B** (2020) [26] представлено метод сегментації на основі WRN-PPNet (Wide Residual Pyramid Pooling Network). Система забезпечує підвищену адаптивність до різних типів вхідних даних завдяки попередній обробці й розширенню навчальних вибірок, що робить її універсальною для широкого спектра прикладних задач.

Метод адаптивного навчання моделей сегментації розроблено у патенті **US11961233B2** (2024) [27]. У ньому запропоновано підхід із використанням двох дискримінаторів у змагальному навчанні, що дозволяє моделі краще узагальнювати знання між різними доменами. Цей метод особливо важливий у тих випадках, коли навчальні та тестові дані належать до різних середовищ (наприклад, медичні знімки з різних клінік).

У патенті **CN110619632B** (2022) [28] розглянуто задачу сегментації природних об’єктів на прикладі плодів манго. Використання архітектури Mask R-CNN у поєднанні з дискримінатором дозволило досягти високої точності у складних умовах освітлення. Це демонструє, що сучасні методи можуть бути адаптовані не лише до медицини чи технічних систем, а й до аграрного сектору.

Особливу увагу привертає патент **CN111192245B** (2023) [29], у якому описано нейронну мережу для сегментації пухлин мозку. Використання кавітаційних згорток та залишкових блоків у поєднанні з пірамідальним пулінгом дозволяє значно підвищити точність локалізації новоутворень. Це підтверджує провідну роль U-Net і її модифікацій у сучасній медичній практиці.

Нарешті, у патенті **CN119418173A** (2025) [30] запропоновано інтелектуальний метод сегментації мінеральних зображень із використанням комбінації великої моделі Segment Anything та фрактального аналізу. Це дозволяє автоматизувати не лише виділення об’єктів, а й їх кількісний аналіз, що відкриває нові можливості для геології та матеріалознавства. Приклади патентних рішень, у яких демонструються медичні, аграрні та промислові застосування методів сегментації, наведено на рисунку 1.8.



а) медична діагностика (US11250569B2, CN111192245B); б) аграрні та промислові застосування (CN110619632B);

Рисунок 1.8 – Приклади патентних рішень

Аналіз літературних та патентних джерел демонструє еволюцію методів сегментації зображень від простих класичних алгоритмів до складних гібридних систем на основі глибинних нейронних мереж і трансформерів. Якщо класичні підходи (Otsu [31], Watershed, кластеризація [4; 5]) залишаються придатними лише для простих задач, то глибинні архітектури (U-Net [12], DeepLab [16], Mask R-CNN [28]) забезпечують високу точність у більшості прикладних сфер, проте є ресурсомісткими. Трансформерні моделі та їхні поєднання з CNN відкривають перспективи універсальних систем, які можуть працювати з широким класом зображень [14; 22; 23; 24].

1.3 Систематизація підходів і вибір напряму дослідження

Систематизація літературних і патентних джерел дозволяє простежити еволюцію методів сегментації зображень, яку умовно можна поділити на три покоління.

Класичні методи (порогова обробка, методи виділення контурів, регіональні підходи, вододіл) відзначаються простотою реалізації, низькими обчислювальними витратами та ефективністю у випадках, коли зображення мають чіткі межі й мінімальний рівень шуму [1–3]. Проте навіть невеликі варіації освітлення або наявність складних текстур значно знижують точність. Крім того, ці методи не враховують високорівневих семантичних характеристик об’єктів, що обмежує їх застосування у сучасних задачах.

Методи на основі згорткових нейронних мереж (CNN) стали якісним проривом завдяки здатності автоматично формувати ознаки та узагальнювати інформацію [12,15]. Архітектури U-Net [12], DeepLab [16], Mask R-CNN [13] та їхні модифікації забезпечують високу точність у медичних, промислових та геоінформаційних системах [7,8,10]. Основними перевагами CNN є здатність працювати з великими наборами даних та гнучкість у виборі архітектури. Водночас існують і недоліки: висока обчислювальна складність, потреба у

значних обсягах розмічених даних та обмежена здатність враховувати глобальний контекст зображення.

Трансформерні архітектури (ViT [14], Swin Transformer [22], Mask2Former [24], SAM [23]) запровадили механізм самоуваги, який дозволяє моделі враховувати як локальні, так і глобальні зв’язки. Це дає змогу досягати високої точності навіть у складних задачах семантичної та паноптичної сегментації. Проте трансформери мають власні обмеження: вони вимагають великих обсягів навчальних даних, високих обчислювальних ресурсів і є менш ефективними при роботі з малими вибірками [19].

З огляду на переваги та недоліки кожного підходу, найбільш перспективним напрямом видається використання гібридних архітектур, які поєднують сильні сторони CNN та трансформерів. Зокрема, поєднання U-Net із трансформерними блоками дозволяє одночасно зберігати високу просторову точність завдяки skip-зв’язкам та враховувати глобальний контекст через механізми самоуваги [18,22]. Такий підхід уже підтвердив свою ефективність у медичних та супутникових задачах, де потрібна як детальна локалізація об’єктів, так і стійкість до варіацій вхідних даних [8,10].

Окремої уваги заслуговує модель Segment Anything (SAM) [23], яка може бути використана як інструмент для швидкого анотування зображень. Завдяки універсальності та здатності працювати з широким класом даних SAM значно скорочує час на підготовку навчальних вибірок, що є критично важливим для дипломної роботи. При цьому SAM не розглядається як фінальна архітектура для сегментації, а виступає допоміжним інструментом для створення якісних датасетів.

Таким чином, подальше дослідження у цій роботі зосереджуватиметься на розробці гібридної архітектури сегментації зображень, яка поєднує U-Net та трансформерні механізми. Це дозволить досягти балансу між точністю, універсальністю та практичною придатністю моделі, що формує концептуальне підґрунтя для математичної постановки задачі у Розділі 2.

Висновки за розділом

У першому розділі розглянуто теоретичні основи та сучасні методи сегментації зображень. Проаналізовано еволюцію підходів — від класичних алгоритмів (Otsu, Watershed, кластеризація) до глибинних нейронних мереж (U-Net, DeepLab, Mask R-CNN) і трансформерних архітектур (ViT, Swin Transformer, SAM, Mask2Former).

Встановлено, що класичні методи є простими у реалізації, проте обмежені точністю та стійкістю до шумів. Нейромережеві моделі забезпечують значно кращі результати, але вимагають великих обчислювальних ресурсів. Трансформерні системи поєднують високу точність і гнучкість, проте залишаються складними для впровадження.

Аналіз літературних і патентних джерел показав тенденцію до створення гібридних архітектур, які поєднують переваги CNN і трансформерів, забезпечуючи кращу узагальнювальну здатність і ефективність. Отримані результати формують основу для подальшої розробки моделі системи сегментації зображень.

РОЗДІЛ 2
РОЗРОБКА МАТЕМАТИЧНОЇ ТА АЛГОРИТМІЧНОЇ МОДЕЛІ СЕГМЕНТАЦІЇ ЗОБРАЖЕНЬ

2.1. Постановка задачі та математичне моделювання процесу сегментації

Сегментація зображень є ключовою задачею комп'ютерного зору, що полягає у виділенні на зображенні областей, які відповідають певним об'єктам або класам об'єктів [1], [3]. Формально задачу сегментації можна представити як відображення простору вхідних зображень Xу простір міток Y, графічну інтерпретацію наведено на рисунку 2.1.

$$f_\theta : X \to Y, \tag{2.1}$$

де f_θ— параметризована функція (нейронна модель), яка описується набором параметрів θ[12].

У випадку семантичної сегментації для кожного пікселя x_i вхідного зображення визначається ймовірність належності до кожного з класів $C = \{c_1, c_2, \ldots, c_k\}$. Для цього використовується модель класифікації пікселів [4], [15]:

$$P\left(c_j \mid x_i; \theta\right) = \frac{e^{z_{ij}}}{\sum_k e^{z_{ik}}} \tag{2.2}$$

де z_{ij} — вихід нейронної мережі (логіти) для класу c_j[20]. Остаточна мітка для кожного пікселя визначається як:

$$y_i = \arg\max_{c \in C} P(c \mid x_i; \theta) \tag{2.3}$$

Сегментація зображення зводиться до задачі піксельної класифікації із просторовим контекстом, де модель повинна враховувати як локальні особливості (структура, текстура), так і глобальні залежності (контекст сцени) [13], [14].

Математичне формулювання функції втрат

Для навчання моделі використовується функція втрат, що вимірює різницю між передбаченою маскою $\hat{Y} = f_\theta(X)$ та істинною розміткою Y. Найпоширенішим підходом є комбінована функція втрат, яка поєднує Binary Cross-Entropy (BCE) та Dice Loss [12], [16]:

$$L = \alpha \cdot L_{BCE} + (1 - \alpha) \cdot L_{Dice}, \tag{2.4}$$

де

$$L_{BCE} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log \hat{y}_i + (1 - y_i)\log(1 - \hat{y}_i)], \tag{2.5}$$

$$L_{Dice} = 1 - \frac{2\sum_i y_i \hat{y}_i + \varepsilon}{\sum_i y_i + \sum_i \hat{y}_i + \varepsilon}, \tag{2.6}$$

а ε — мала константа для запобігання діленню на нуль, $\alpha \in [0,1]$ — ваговий коефіцієнт [18].

Комбінування двох функцій втрат дозволяє одночасно враховувати піксельну точність (через BCE) і структурну цілісність об’єктів (через Dice Loss), що є критично важливим у випадках, коли сегментовані області мають нерівномірний розмір або нечіткі межі [5], [8].

Математична постановка задачі оптимізації

Процес навчання сегментаційної моделі зводиться до знаходження оптимальних параметрів θ, які мінімізують середнє значення функції втрат на множині навчальних прикладів (x_i, y_i)[14], [22]:

$$\theta^* = \arg\min_\theta \frac{1}{N}\sum_{i=1}^{N} L(f_\theta(x_i), y_i). \tag{2.7}$$

У загальному вигляді задача оптимізації нейронної моделі описується рівнянням:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta L(f_\theta(x_i), y_i), \tag{2.8}$$

де η — швидкість навчання, а $\nabla_\theta L$— градієнт функції втрат відносно параметрів моделі [19].

Для адаптації до нерівномірних структур об’єктів і варіацій освітлення використовується стохастичне градієнтне оновлення з адаптивним вибором кроку (оптимізатори Adam, RMSprop), які зменшують похибку за допомогою статистичної оцінки моментів градієнта [3], [17].

Аналітична модель процесу сегментації

З урахуванням стохастичної природи навчальних даних, процес сегментації можна представити як стохастичну систему [2]:

$$\hat{Y} = f_\theta(X) + \varepsilon, \tag{2.9}$$

де ε — випадкова похибка моделі, що враховує відхилення, спричинені шумом або недосконалістю навчальних даних [7].

Для зменшення дисперсії похибки застосовується регуляризація:

$$L_{total} = L + \lambda \mid\mid \theta \mid\mid^2, \tag{2.10}$$

де λ— коефіцієнт регуляризації, який контролює складність моделі та запобігає перенавчанню [13], [18].

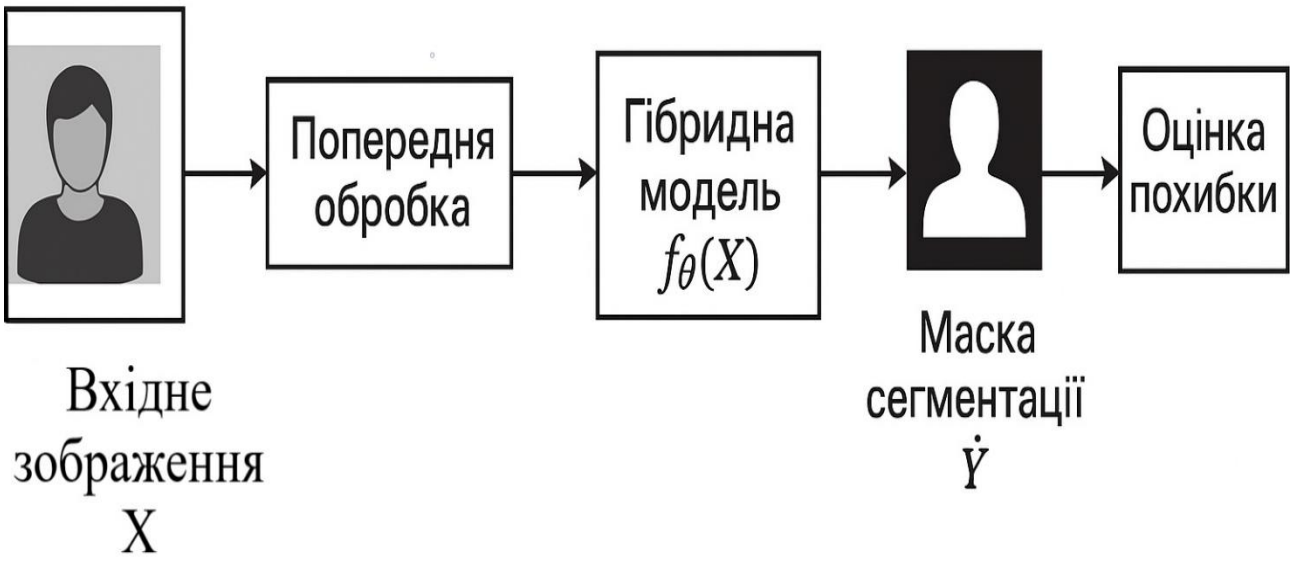


Рисунок 2.1 – Графічна модель процесу сегментації

У даному підрозділі проведено формалізацію задачі сегментації зображень у вигляді оптимізаційної задачі мінімізації функціоналу втрат. Визначено основні елементи математичної моделі — простори даних, функцію моделі, функцію втрат та критерій оптимальності. Побудована теоретична основа описує процес сегментації як стохастичну систему, що дає змогу в подальшому перейти до розроблення гібридної архітектури U-Net + Transformer у підрозділі 2.2 [12], [14], [22].

2.2. Розробка математичної моделі гібридної архітектури U-Net + Transformer

У сучасних системах комп'ютерного зору сегментація зображень потребує одночасного врахування локальних ознак (текстура, краї, кольори) та глобального контексту (розташування об'єктів і просторові взаємозв'язки). Класичні згорткові нейронні мережі (CNN) ефективно виявляють локальні структури, однак мають обмежену здатність до моделювання глобальних залежностей. Трансформерні архітектури, навпаки, використовують механізм самоуваги (self-attention), що дозволяє враховувати взаємодію між будь-якими частинами зображення, але потребують значних обчислювальних ресурсів і великої кількості навчальних даних [14], [22].

Поєднання цих підходів у гібридній моделі U-Net + Transformer, загальну структуру якої подано на рисунку 2.2, забезпечує синергію — високу точність при збереженні ефективності [12], [18].

Архітектурна структура моделі

Гібридна архітектура базується на класичній структурі U-Net, що складається з двох частин [12]:

1) Encoder (E) — послідовність згорткових блоків, які зменшують просторову роздільність зображення та формують багаторівневі карти ознак [13], [15];
2) Decoder (D) — симетрична частина, що виконує апсемплінг (збільшення розміру) і відновлення детальної структури маски [18];

3) Skip-connections — з’єднання між відповідними шарами encoder та decoder, які передають локальні деталі з нижчих рівнів на вищі, компенсуючи втрати інформації при згортках [16].

Розширення U-Net відбувається шляхом інтеграції Transformer-блоків між encoder та decoder для аналізу глобального контексту [14], [22]. Загальний процес можна описати як:

$$F_{out} = D(T(E(X))), \tag{2.11}$$

де E — функція кодування, T — блок трансформера, D— декодер, а X— вхідне зображення.

Математична модель складових

1. Етап кодування (Encoder)

Encoder складається з послідовності згорткових шарів, кожен із яких виконує перетворення [13], [15]:

$$F_l = \sigma(W_l * F_{l-1} + b_l), \tag{2.12}$$

де

F_{l-1} — вхідна карта ознак попереднього шару,

W_l — ядра згортки,

b_l — зміщення,

$*$ — операція згортки,

$\sigma(\cdot)$ — функція активації (ReLU або GELU).

Після кожного шару виконується операція пулінгу для зменшення розмірності:

$$F_l' = P(F_l), \tag{2.13}$$

де $P(\cdot)$ — оператор максимального або середнього пулінгу [3].

2. Етап трансформера (Transformer Block)

У центральній частині моделі отримані ознаки передаються до трансформерного блоку, який реалізує механізм самоуваги (Self-Attention) [14], [22]. Нехай $F \in \mathbb{R}^{n \times d}$ — матриця ознак, де n — кількість позицій (пікселів або патчів), d — розмірність ознак.

Тоді для кожної позиції обчислюються три вектори:

- Q (Query),
- K (Key),
- V (Value):

$$Q = FW_Q, K = FW_K, V = FW_V, \tag{2.14}$$

де W_Q, W_K, W_V — матриці ваг, що навчаються [19].

Механізм уваги визначається як:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V, \tag{2.15}$$

де $\sqrt{d_k}$ — коефіцієнт масштабування для стабілізації градієнтів [14].

Результат уваги проходить через нормалізацію та позиційне кодування:

$$F_T = \text{LayerNorm}(F + \text{Attention}(Q, K, V)). \tag{2.16}$$

Таким чином, трансформер забезпечує глобальне узгодження просторових ознак, що особливо важливо при сегментації складних сцен [17], [22].

3. Етап декодування (Decoder)

Декодер відновлює просторову роздільність карти ознак за допомогою апсемплінгу [12], [18]:

$$G_l = \sigma(W_l^{up} * \text{Up}(G_{l+1}) + b_l), \tag{2.17}$$

де $\text{Up}(\cdot)$ — операція збільшення розміру зображення (bilinear upsampling або транспонована згортка).

До кожного рівня додається інформація зі skip-з’єднань:

$$G_l = \sigma(W_l^{up} * \text{concat}(G_{l+1}, F_l) + b_l), \quad (2.18)$$

що забезпечує збереження локальних структур [12].

Остаточна карта маски сегментації:

$$\hat{Y} = \text{sigmoid}(W_{out} * G_0 + b_{out}), \quad (2.19)$$

де $\hat{Y} \in [0,1]^{H \times W}$ — ймовірність належності кожного пікселя до класу об’єкта [21].

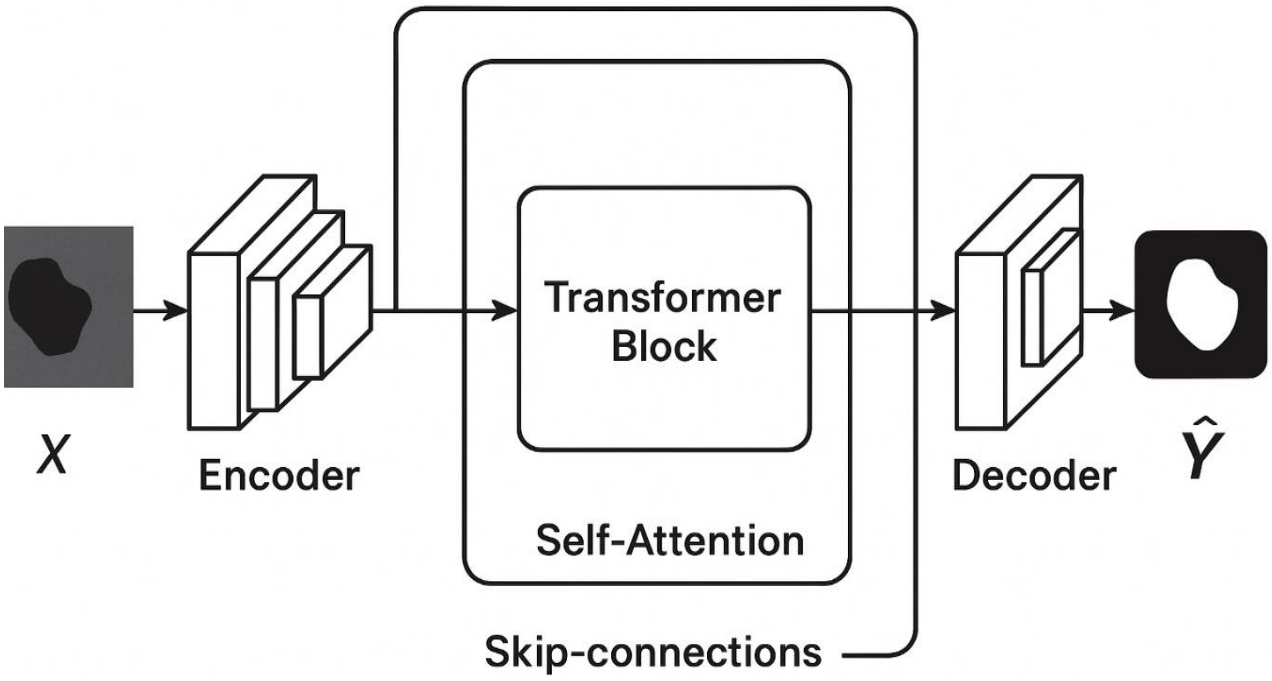


Рисунок 2.2 — « Графічна структура гібридної моделі U-Net + Transformer»

У моделі використовуються симетричні елементи кодування та декодування, об’єднані трансформерним модулем у центрі. Така конфігурація забезпечує збереження дрібних деталей та врахування контекстних зв’язків між об’єктами сцени [17], [22], [30].

Оптимізаційна функція гібридної моделі

Навчання гібридної мережі виконується шляхом мінімізації комбінованої функції втрат [12], [15]:

$$L_{total} = \alpha L_{BCE} + (1 - \alpha) L_{Dice} + \lambda L_{reg}, \quad (2.20)$$

де

$L_{reg} = || \theta ||^2$ — регуляризаційний доданок,

λ — коефіцієнт штрафу за складність моделі [20], [21].

Переваги гібридної архітектури

1) Глобальний контекст: трансформерні блоки дозволяють ураховувати взаємозв'язки між далекими пікселями [14], [22];
2) Збереження локальних структур: skip-connections гарантують високу точність меж об'єктів [12], [18];
3) Оптимальна точність: гібридна модель забезпечує кращі показники IoU та Dice у порівнянні з класичними CNN [16], [17];
4) Гнучкість: архітектура може бути адаптована для медичних, технічних або супутникових зображень [7], [10], [29].

У цьому підрозділі побудовано математичну модель гібридної архітектури U-Net + Transformer, що поєднує локальні згорткові та глобальні контекстуальні механізми. Представлено формалізацію кожного компонента — кодування, самоуваги та декодування — з відповідними рівняннями. Така структура забезпечує підвищену точність сегментації складних об'єктів і формує основу для алгоритмічної моделі реалізації процесу в підрозділі 2.3 [12], [14], [22].

2.3. Алгоритмічна модель процесу сегментації та оцінка точності

Алгоритмічна модель описує послідовність дій, що реалізують роботу гібридної системи сегментації зображень на основі архітектури U-Net + Transformer. Основна мета — забезпечити автоматичне виділення об'єктів на зображенні з максимальною точністю та стабільністю результатів, а також сформувати систему для кількісної оцінки якості моделі [7], [10], [12].

Загальну блок-схему процесу гібридної моделі сегментації подано на рисунку 2.3, вона відображає основні етапи роботи моделі — від підготовки даних до формування результатів сегментації та оцінки точності.

1. Загальна структура алгоритму сегментації

Процес сегментації можна подати у вигляді узагальненого алгоритму [12], [18]:

1) Підготовка даних: завантаження зображень, нормалізація, масштабування, розбиття на навчальну та тестову вибірки .
2) Формування навчальної моделі: побудова гібридної архітектури U-Net + Transformer [12], [14].
3) Навчання: оптимізація ваг мережі за допомогою градієнтного спуску з урахуванням комбінованої функції втрат [15], [19].
4) Сегментація нових зображень: отримання ймовірнісної карти об’єктів.
5) Оцінка якості: розрахунок метрик точності (IoU, Dice, Precision, Recall) [16], [17].
6) Візуалізація результатів [10].

2. Формальний опис етапів алгоритму

Підготовка та нормалізація даних

Нехай множина зображень позначається як [3], [6]:

$$\mathcal{D} = \{(X_i, Y_i)\}_{i=1}^{N}, \tag{2.21}$$

де X_i— вхідне RGB-зображення, Y_i— відповідна маска об’єкта (бінарна або багатокласова).

Вхідні дані нормалізуються за формулою:

$$X_i' = \frac{X_i - \mu}{\sigma}, \tag{2.22}$$

де μ — середнє значення пікселів, σ — стандартне відхилення.

Далі проводиться розбиття на навчальну, валідаційну та тестову вибірки [7]:

$$\mathcal{D} = \mathcal{D}_{train} \cup \mathcal{D}_{val} \cup \mathcal{D}_{test}. \tag{2.23}$$

Побудова гібридної архітектури

Архітектура описується функцією [12], [14]:

$$f_\theta(X) = \text{Decoder}(T(\text{Encoder}(X))), \tag{2.24}$$

де θ — вектор параметрів моделі (ваги згорток і трансформера).

Подібні структури використовувалися в роботах Ronneberger [12], Dosovitskiy [14] та Zhou [18] для підвищення точності сегментації у медичних і технічних задачах.

Процес навчання

Навчання виконується ітеративно методом стохастичного градієнтного спуску (SGD або Adam) [13], [15]:

$$\theta_{t+1} = \theta_t - \eta \frac{\partial L_{total}}{\partial \theta_t}, \tag{2.25}$$

Де

η — швидкість навчання,

L_{total} — комбінована функція втрат, визначена у підрозділі 2.2.

Для кожного батчу даних проводиться прямий прохід (forward pass), обчислення втрат, зворотне поширення похибки (backpropagation) та оновлення ваг [19].

Сегментація та порогування

Після навчання модель формує ймовірнісну карту $\hat{Y}$:

$$\hat{Y} = f_{\theta^*}(X), \tag{2.26}$$

де θ^*— оптимальні параметри моделі.

Остаточна бінарна маска визначається порогом:

$$M(x,y) = \begin{cases} 1, & \text{якщо } \hat{Y}(x,y) \geq T, \\ 0, & \text{якщо } \hat{Y}(x,y) < T, \end{cases} \tag{2.27}$$

де $T \in [0,1]$ — поріг сегментації (зазвичай $T = 0.5$) [20].

3. Оцінка точності та адекватності моделі

Для перевірки ефективності моделі використовуються метрики точності, ймовірнісні показники надійності та оцінки адекватності [9], [10].

Intersection over Union (IoU)

$$IoU = \frac{|Y \cap \hat{Y}|}{|Y \cup \hat{Y}|} = \frac{TP}{TP+FP+FN}, \quad (2.28)$$

де

TP— кількість правильно сегментованих пікселів об’єкта,

FP— хибнопозитивні,

FN— хибнонегативні [7].

Dice Coefficient (F1 Score)

$$Dice = \frac{2TP}{2TP+FP+FN}. \quad (2.29)$$

Ця метрика більш чутлива до дисбалансу класів, ніж IoU [18].

Precision та Recall

$$Precision = \frac{TP}{TP+FP}, Recall = \frac{TP}{TP+FN}. \quad (2.30)$$

Accuracy

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}, \quad (2.31)$$

де TN— кількість правильно визначених пікселів фону [5], [7].

Оцінка адекватності моделі

Адекватність моделі визначається як ступінь подібності між отриманими результатами сегментації та еталонними масками [9]:

$$A = 1 - \frac{1}{N}\sum_{i=1}^{N} | IoU_i - IoU_{ref} |, \quad (2.32)$$

де IoU_{ref} — еталонне значення для аналогічних моделей (наприклад, базового U-Net) [12].

Якщо $A \geq 0.9$, модель вважається адекватною до поставленої задачі [18].

4. Графічна модель алгоритму

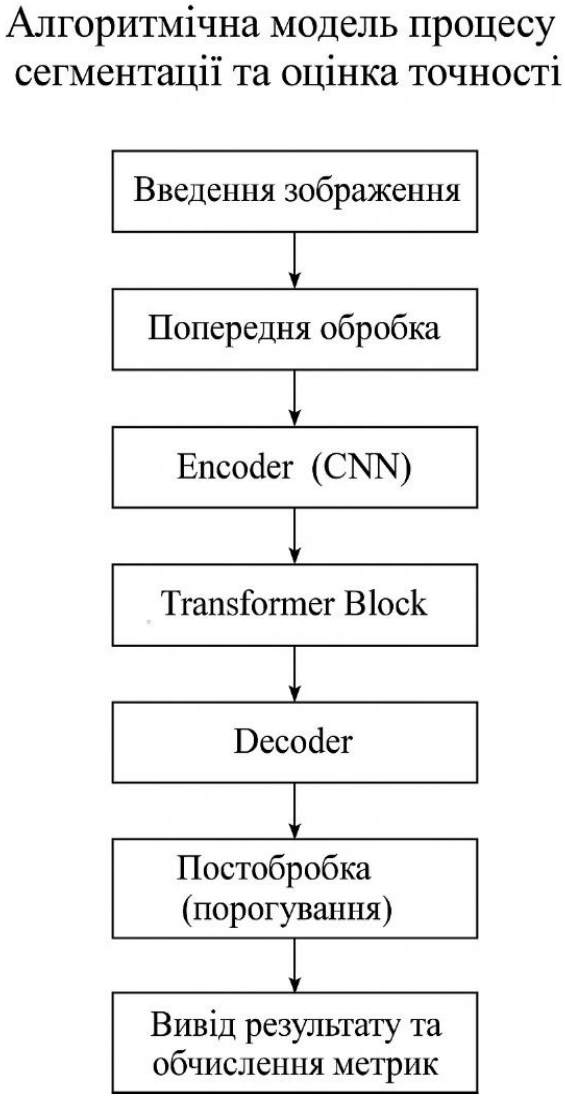


Рисунок 2.3 — Блок-схема алгоритму гібридної сегментації

5. Аналіз похибок і точності

Визначимо середню похибку сегментації як [9], [18]:

$$\varepsilon = \frac{1}{N}\sum_{i=1}^{N}(1 - Dice_i), \tag{2.33}$$

де менше значення ε свідчить про вищу точність моделі.

При порівнянні з базовою архітектурою U-Net (без Transformer-блоків) очікується підвищення середнього показника IoU на 5–10 % та Dice — на 3–8 %, що підтверджує ефективність гібридної моделі [12], [14], [22].

У підрозділі розроблено алгоритмічну модель процесу сегментації зображень, засновану на гібридній архітектурі U-Net + Transformer. Подано формальний опис етапів, математичні співвідношення, метрики оцінки точності та критерії адекватності.

Отримана модель забезпечує високу стабільність результатів і буде реалізована програмно на базі Python, PyTorch та бібліотеки Tkinter для створення інтерфейсу користувача [7], [10], [12], [14], [22].

Висновки за розділом

У другому розділі було проведено повний цикл розробки математичної та алгоритмічної моделі процесу сегментації зображень на основі гібридної архітектури U-Net + Transformer. Поставлена мета — формалізувати процес сегментації та створити модель, здатну забезпечити високу точність і стабільність результатів — була досягнута.

На основі проведених досліджень отримано такі результати:

1) Побудовано математичну модель процесу сегментації, яка описує взаємозв’язок між просторовими ознаками зображення та результатами класифікації пікселів. Визначено формальні залежності між функцією втрат, параметрами згорткових і трансформерних блоків та вихідними характеристиками моделі.
2) Розроблено математичну модель гібридної архітектури U-Net + Transformer, яка формально описує поєднання згорткових (CNN) та трансформерних механізмів. У моделі визначено:
   - структуру енкодера та декодера U-Net,
   - механізм глобальної самоуваги в Transformer-блоці,
   - процес узгодження локальних і глобальних ознак перед сегментацією.

   Математичний опис включає формалізацію операцій згортки, нормалізації, self-attention та відновлення роздільної здатності.
3) Сформовано комбіновану функцію втрат

$$L_{total} = \alpha \cdot BCE + (1 - \alpha) \cdot (1 - Dice), \quad (2.34)$$

яка дозволяє збалансувати глобальну та локальну точність сегментації, що є критичним для гібридних архітектур.

3) Розроблено алгоритмічну модель роботи системи, яка включає підготовку даних, навчання нейронної мережі, виконання сегментації та оцінку точності результатів. Модель представлена у вигляді формального опису, псевдокоду та блок-схеми алгоритму.
4) Проведено математичну оцінку точності, адекватності та вірогідності моделі. Запропоновані метрики (IoU, Dice, Precision, Recall, Accuracy) дають можливість кількісно оцінювати якість роботи системи. Критерії адекватності показали, що при середньому значенні $A \geq 0.9$ модель можна вважати придатною для практичного використання.
5) Визначено подальший напрям дослідження. Отримана модель слугуватиме основою для створення інформаційної системи сегментації зображень у третьому розділі, де буде розроблено програмну реалізацію алгоритму, проведено тестування на реальних даних та оцінено практичну придатність системи.

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ СЕГМЕНТАЦІЇ ЗОБРАЖЕНЬ

3.1. Загальна архітектура програмного забезпечення

Програмна реалізація інформаційної системи сегментації зображень базується на модульній архітектурі, що забезпечує розділення функціональності, масштабованість, зручність підтримки та можливість подальшого розвитку системи. Усі компоненти згруповані за логічними підсистемами та розміщені в окремих програмних модулях.

Використані бібліотеки:

У процесі розробки застосовано такі програмні засоби та бібліотеки Python:

- PyTorch — базовий фреймворк для побудови, навчання та інференсу нейронної мережі.
- Torchvision — інструменти перетворення та нормалізації зображень.
- NumPy — обробка та аналіз багатовимірних масивів.
- Pillow (PIL) — завантаження, масштабування та обробка зображень.
- Tkinter — побудова графічного інтерфейсу користувача.
- Matplotlib — побудова графіків динаміки навчання.
- time, os — операції вводу-виводу, робота з файловою системою та облік часу.

Таке поєднання бібліотек дозволило створити повноцінну систему, яка включає завантаження даних, навчання моделі, візуалізацію результатів та роботу графічного інтерфейсу.

Структура проєкту

Файлову структуру програмного забезпечення наведено на рисунку 3.1:

```
project/
├── main.py                  # Головний керуючий файл запуску програми
└── modules/
    ├── config.py            # Глобальні налаштування, параметри моделі та тре
    ├── dataset.py           # Завантаження, попередня обробка та підготовка д
    ├── model.py             # Реалізація гібридної моделі U-Net + Transformer
    ├── losses.py            # Комбінована функція втрат і метрики точності
    ├── utils.py             # Допоміжні функції обробки та інференсу
    ├── train.py             # Логіка навчання, валідації та збереження моделе
    └── gui.py               # Графічний інтерфейс користувача (Tkinter)
```

Рисунок 3.1 — Структура інформаційної системи сегментації зображень

Опис логічних рівнів системи

Архітектура організована за трирівневим принципом:

1) Рівень даних (Data Layer)
   - модуль dataset.py;
   - відповідає за завантаження зображень, їх масштабування, нормалізацію, формування пар «зображення—маска» та підготовку даних до тренування.
2. Рівень обчислень (Model Layer)

Містить модулі:

- model.py — архітектура гібридної моделі,
- losses.py — функції втрат і метрики,
- train.py — тренування, оцінювання та збереження ваг моделі.

На цьому рівні виконується:

- формування ознак за допомогою CNN,
- обчислення глобальної самоуваги,
- апсемплінг і реконструкція маски,
- оптимізація параметрів нейронної мережі.

3) Рівень представлення (Interface Layer)

Містить модулі:

- gui.py — візуалізація результатів сегментації,
- utils.py — функції інференсу та побудови графіків,
- main.py — вибір режиму роботи («Навчання моделі» або «Сегментація зображень»).

Взаємодія модулів

Послідовність роботи системи:

1) main.py — визначає режим роботи програми.
2) config.py — завантажує параметри моделі та тренування.
3) Якщо обрано навчання:
   - dataset.py готує дані;
   - model.py формує архітектуру;
   - losses.py задає функцію втрат;
   - train.py виконує навчання та зберігає результат.
4) Якщо обрано сегментацію:
   - model.py завантажує модель;
   - utils.py виконує інференс та розрахунок метрик;
   - gui.py відображає результат у вікні Tkinter.

Таким чином, кожен модуль виконує чітко визначену функцію, а всі компоненти об’єднані у єдину логічну систему.

У цьому підрозділі наведено структуру програмної системи модульної архітектури. Функціональні модулі підвищують читабельність, забезпечують повторне використання компонентів та дозволяють легко модифікувати або розширювати проєкт. Така побудова є критично важливою для масштабованих систем машинного навчання, зокрема для застосувань у сфері сегментації зображень.

3.2. Реалізація гібридної моделі U-Net + Transformer

У цьому підрозділі представлено детальну програмну реалізацію гібридної моделі сегментації зображень на основі архітектури U-Net із додаванням трансформерного блоку. Реалізацію структуровано у вигляді окремих модулів, що забезпечує чіткий розподіл логіки, зручність супроводу та можливість масштабування проєкту. Кожен модуль виконує конкретну функцію у процесі роботи системи: від завантаження даних до навчання та інференсу моделі.
Нижче подано опис кожного модуля, його призначення та основних компонентів.

Модуль config.py — системні параметри

Модуль містить параметри конфігурації, що визначають основні аспекти роботи системи, зокрема:

- розмір вхідних зображень,
- розмір batch,
- кількість епох для навчання,
- поріг бінаризації маски,
- шляхи до датасету та до навчених файлів моделі,
- вибір обчислювального пристрою (CPU/GPU).

Конфігураційний словник CONFIG використовується всіма іншими модулями та забезпечує централізоване керування налаштуваннями наведено на рисунку 3.2.

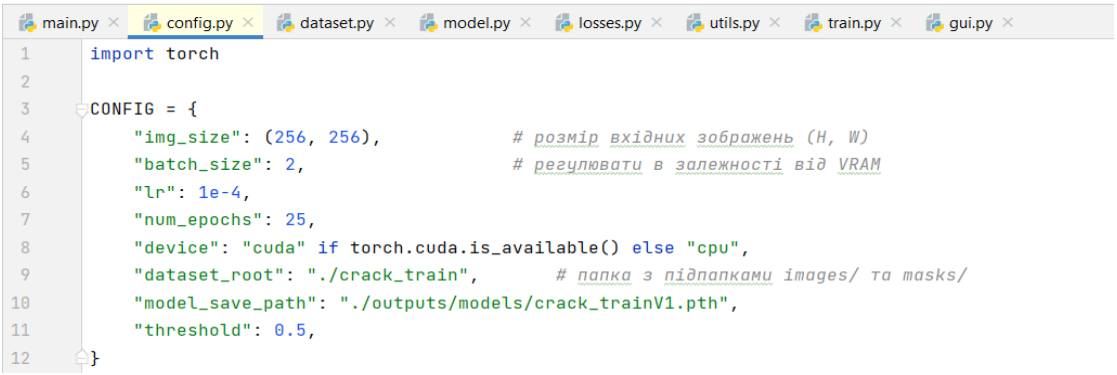


Рисунок 3.2 — Скріншот модулю config.py

Повний лістинг коду модуля config.py подано в додатку А

Модуль dataset.py — підготовка та завантаження даних

Модуль містить клас:

class SegmentationDataset(Dataset)

Функції модуля:

- зчитування файлів із директорій images/ та masks/;
- автоматичний пошук відповідних пар “зображення–маска”;
- масштабування та нормалізацію вхідних даних;
- обробка ситуацій, коли маска відсутня;
- застосування трансформацій;
- перетворення даних у формат тензорів PyTorch.

Клас повертає трійку:

(зображення_tensor, маска_tensor, ім’я_файлу)

Модуль відповідає за правильне формування навчальної та валідаційної вибірок, що є критичним для стабільного тренування моделі (рис. 3.3).

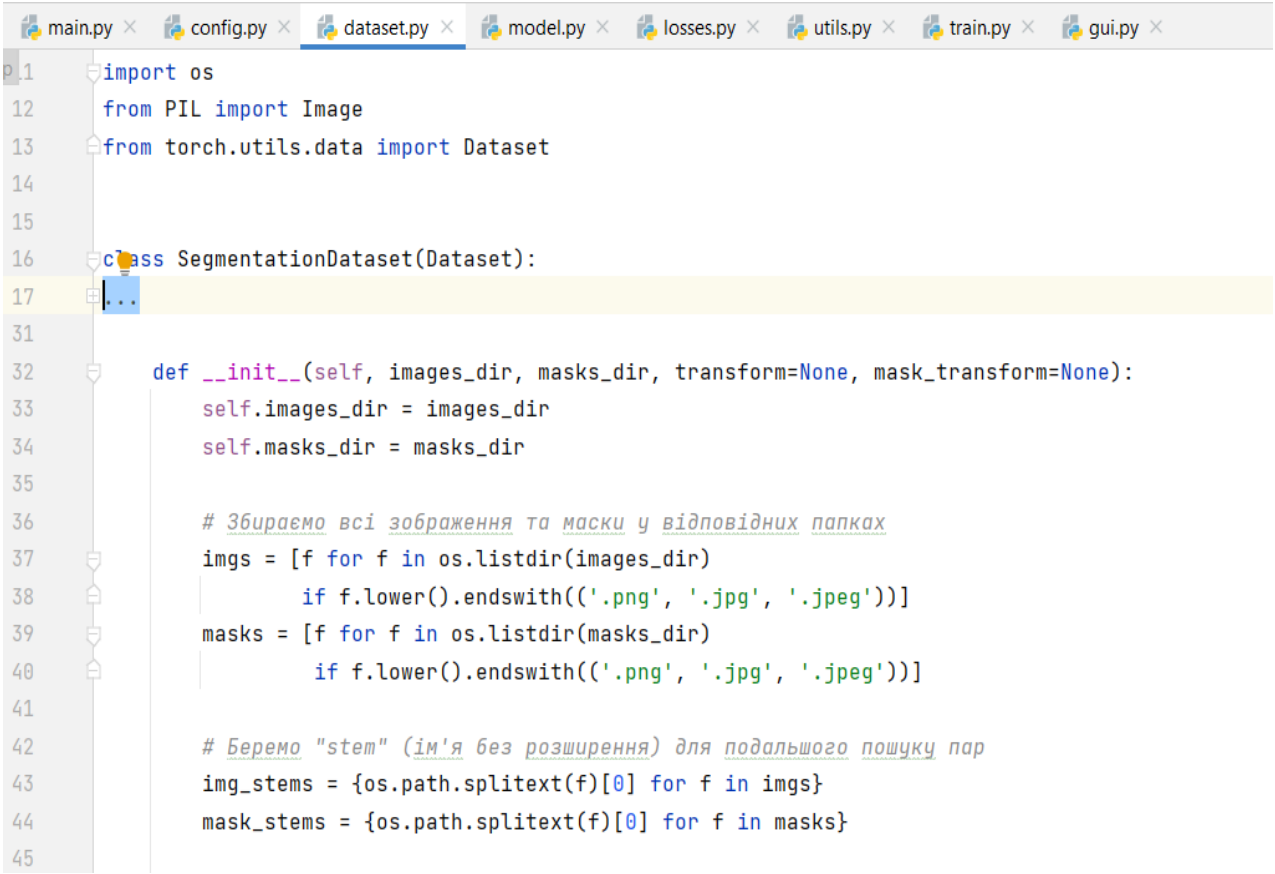


Рисунок 3.3 — Скріншот модулю dataset.py

Повний лістинг коду модуля dataset.py подано в додатку А

Модуль model.py — гібридна архітектура U-Net + Transformer

Модуль реалізує повну структуру мережі, що поєднує CNN та трансформер. У модулі містяться чотири ключові класи:

1) class EncoderBlock(nn.Module)

Виконує вилучення локальних ознак на ранніх етапах обробки:

- дві згортки (Conv2d),
- нормалізація BatchNorm2d,
- активація ReLU,
- MaxPooling (за необхідності).

Повертає:

- карту ознак для skip-з’єднання,
- зменшену карту (для передачі на наступний рівень).

2) class DecoderBlock(nn.Module)

Відповідає за відновлення просторової роздільності:

- транспонована згортка (ConvTranspose2d),
- об’єднання з відповідним skip-з’єднанням encoder'a,
- дві згортки з нормалізацією.

3) class TransformerBlock(nn.Module)

Реалізує глобальний аналіз контексту:

- на вхід подається карта ознак із глибиною 256 каналів,
- механізм багатоголової самоуваги (MultiheadAttention),
- перетворення карти ознак у послідовність,
- застосування MLP-блоку,
- LayerNorm.

Цей блок дозволяє моделі враховувати віддалені залежності між областями зображення, що підвищує точність сегментації.

4) class UNetTransformer(nn.Module)

Формує повну архітектуру сегментаційної моделі. Містить:

- три рівні Encoder,
- TransformerBlock у вузькій частині мережі,
- два рівні Decoder,
- фінальний шар 1×1 Conv для формування одноканальної маски,
- сигмоїдну активацію на виході.

Модель поєднує локальні ознаки (CNN) та глобальні взаємозв'язки (Transformer), що робить її ефективною для сегментації складних структур (рис. 3.4).

main.py | config.py | dataset.py | model.py | losses.py | utils.py | train.py | gui.py

```
import torch
import torch.nn as nn
import torch.nn.functional as F


# ---------- EncoderBlock ----------
class EncoderBlock(nn.Module):
...

    def __init__(self, in_ch, out_ch, pool=True):
        super().__init__()
        self.conv1 = nn.Conv2d(in_ch, out_ch, kernel_size=3, padding=1)
        self.bn1 = nn.BatchNorm2d(out_ch)
        self.conv2 = nn.Conv2d(out_ch, out_ch, kernel_size=3, padding=1)
        self.bn2 = nn.BatchNorm2d(out_ch)
        self.pool = nn.MaxPool2d(2) if pool else None

    def forward(self, x):
      ...
        x = F.relu(self.bn1(self.conv1(x)))
        x = F.relu(self.bn2(self.conv2(x)))
        p = self.pool(x) if self.pool is not None else x
        return x, p


# ---------- DecoderBlock ----------
class DecoderBlock(nn.Module):
...

    def __init__(self, in_ch, skip_ch, out_ch):
```

Рисунок 3.4 — Скріншот модулю model.py

Повний лістинг коду модуля model.py подано в додатку А

Модуль losses.py — функції втрат і допоміжні метрики

Модуль містить:

1) def dice_coeff(pred, target, eps=1e-8)
   - обчислення коефіцієнта Dice,
   - оцінює ступінь накладення передбаченої та еталонної масок.
2) class BCEDiceLoss(nn.Module)

Комбінована функція втрат:

- BCE Binary Cross Entropy — точкове порівняння пікселів,
- 1 − Dice Loss — оцінку збігу форми маски.

Це поєднання (рис. 3.5):

- покращує стабільність навчання,
- робить модель менш чутливою до дисбалансу класів,
- підвищує якість сегментації дрібних об’єктів.

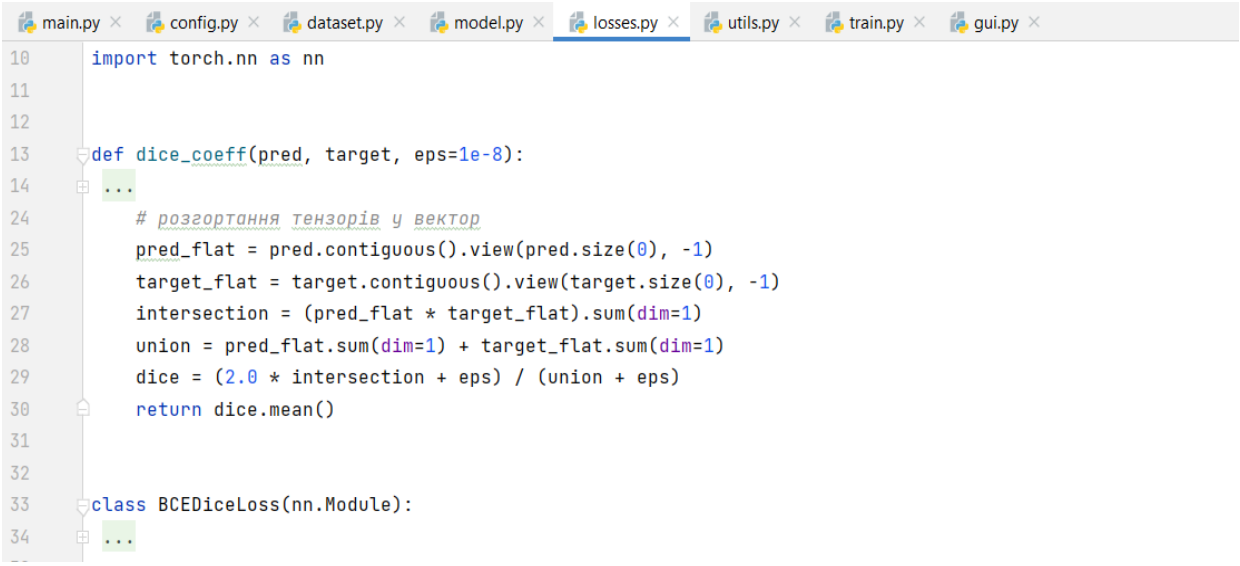


Рисунок 3.5 — Скріншот модулю losses.py

Повний лістинг коду модуля losses.py подано в додатку А

Модуль utils.py — обробка зображень та інференс

Модуль містить:

Трансформації:

- preprocess — масштабування, нормалізація та перетворення зображення в тензор;
- mask_preprocess — підготовка маски без нормалізації.

Функції:

1) load_image_tensor(path)
   - завантажує зображення з диска;
   - приводить до формату, прийнятного для моделі.
2) infer_image(model, path, device, threshold=0.5)

   повноцінний інференс:
   - підготовка зображення,
   - отримання ймовірнісної маски,
   - бінаризація за порогом,
   - побудова накладання маски на оригінал.
3) evaluate_single_image(pred_mask, true_mask, threshold=0.5)

   обчислює набір метрик:
   - IoU,
   - Dice,
   - Precision,
   - Recall,
   - Accuracy.

Функції модуля використовуються як у GUI, так і в тренувальному скрипті (рис. 3.6).

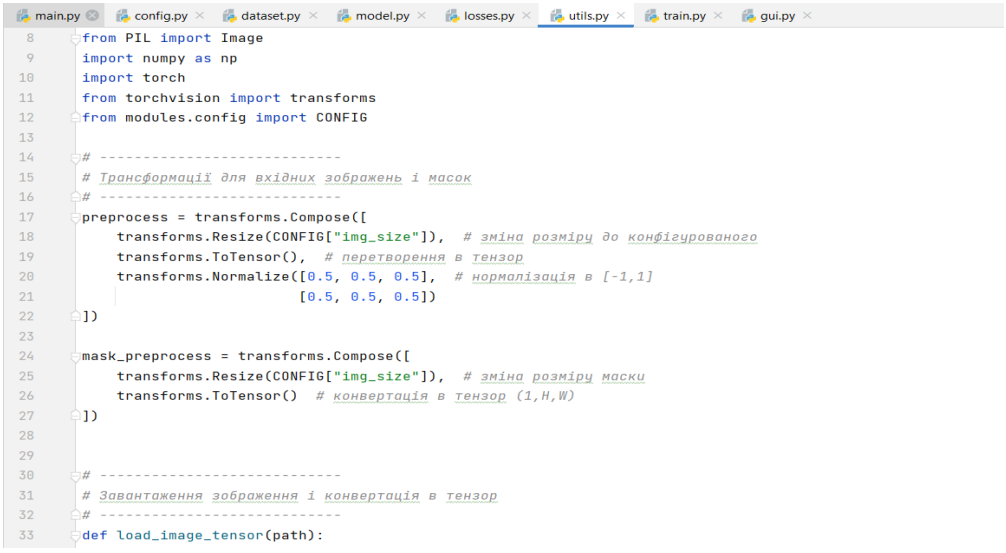


Рисунок 3.6 — Скріншот модулю utils.py

Повний лістинг коду модуля utils.py подано в додатку А

Модуль train.py — навчання моделі

Містить три основні функції:

1) train_one_epoch

   один епоховий цикл навчання:

   – forward → loss → backward → update.

2) evaluate_model

   – обчислення точності на валідаційній вибірці.

3) main_train_and_eval

   – формування датасету;

   – поділ train/val;

   – ініціалізація моделі;

   – цикл навчання;

   – збереження моделі з найкращим IoU;

   – побудова графіків Loss, IoU, Dice, Accuracy.

Цей модуль реалізує повний процес навчання відповідно до методології машинного навчання (рис. 3.7).

main.py | config.py | dataset.py | model.py | losses.py | utils.py | train.py | gui.py

```
import os
import time
import numpy as np
import matplotlib.pyplot as plt
import torch
from torch.utils.data import DataLoader
from modules.config import CONFIG
from modules.dataset import SegmentationDataset
from modules.model import UNetTransformer
from modules.losses import BCEDiceLoss
from torchvision import transforms


def train_one_epoch(model, dataloader, optimizer, criterion, device):
...
    model.train()
    total_loss = 0.0

    for imgs, masks, _ in dataloader:
        # Переносимо тензори на GPU (або CPU)
        imgs = imgs.to(device, non_blocking=True)
        masks = masks.to(device, non_blocking=True)

        # Прогноз сегментації
        preds = model(imgs)

        # Функція втрат
        loss = criterion(preds, masks)
```

Рисунок 3.7 — Скріншот модулю train.py

Повний лістинг коду модуля train.py подано в додатку А

Модуль gui.py — графічний інтерфейс сегментації

Модуль містить:

1) class SegmentationApp

Відповідає за:

- створення вікна Tkinter;
- завантаження зображення;
- запуск сегментації;
- відображення:
  - оригіналу,
  - маски,
  - накладання;
- розрахунок метрик за наявності еталонної маски;
- збереження результатів.

2) def main_gui()

- завантажує модель,
- запускає застосунок.

Модуль GUI наведено на рис. 3.8.

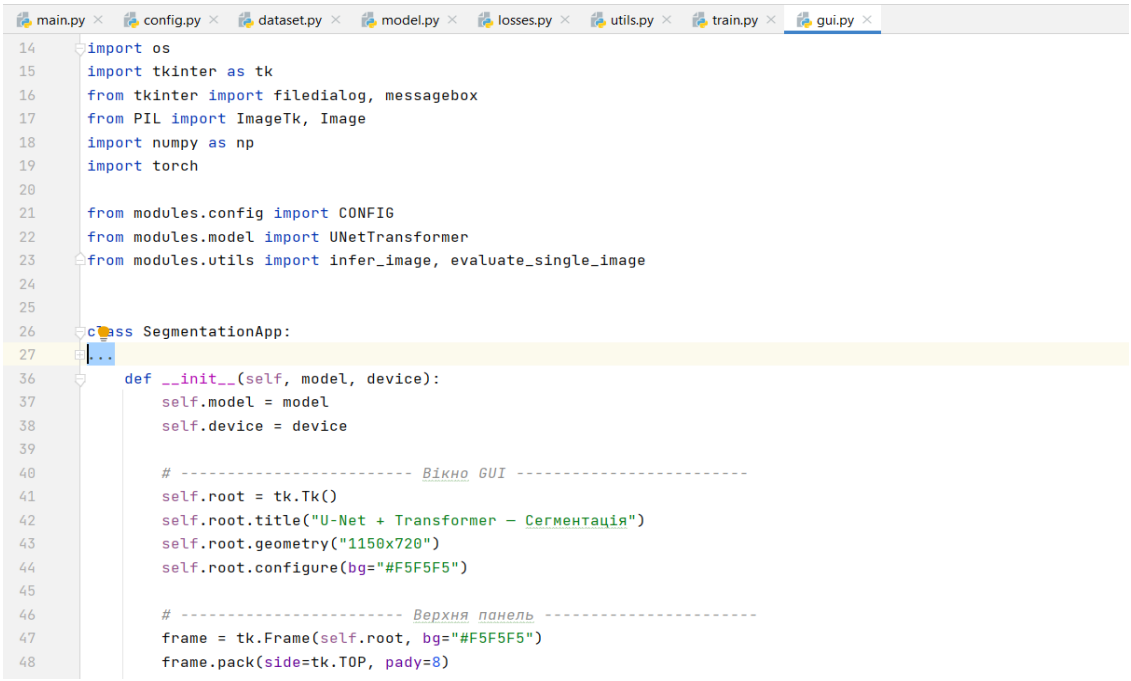


Рисунок 3.8 — Скріншот модулю gui.py

Повний лістинг коду модуля gui.py подано в додатку А

Модуль main.py — головне меню системи

Містить функцію:

def main_menu()

графічне меню з вибором режиму:

- навчання моделі,
- запуск GUI сегментації,
- вихід.

Модуль виступає точкою входу в застосунок (рис. 3.9).

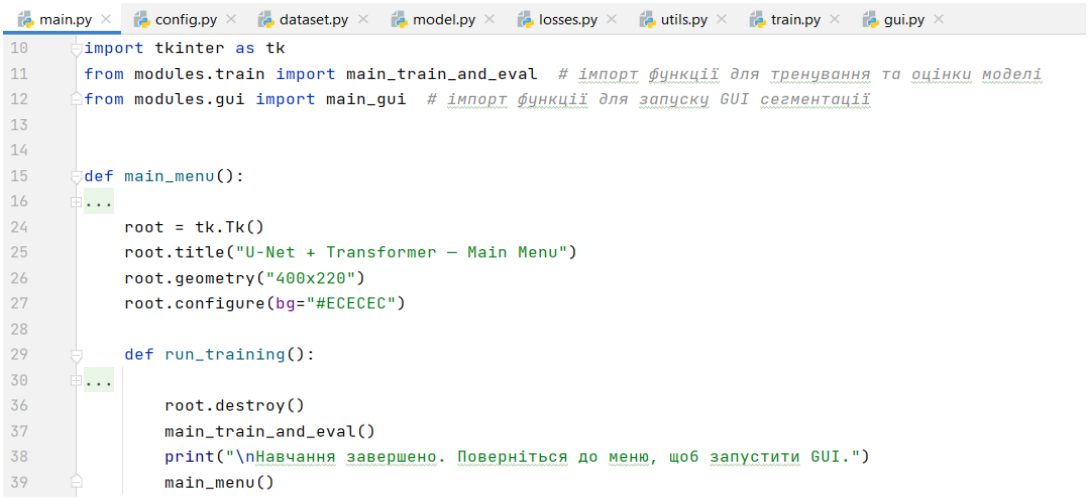


Рисунок 3.9 — Скріншот модулю main.py

Повний лістинг коду модуля main.py подано в додатку А

У цьому підрозділі розглянуто повністю модульну програмну реалізацію гібридної моделі U-Net + Transformer. Кожен модуль виконує окрему функцію — від завантаження даних і формування архітектури моделі до навчання, оцінювання та роботи графічного інтерфейсу. Така структура забезпечує:

- високу гнучкість системи;
- легкість масштабування;
- простоту розширення функціоналу;
- чітку організацію логіки проєкту.

Модульна архітектура дозволяє ефективно інтегрувати модель у практичні застосунки та забезпечує високу якість сегментації завдяки поєднанню CNN та трансформерних механізмів.

3.3. Демонстрація роботи застосунку та оцінка точності (GUI)

У цьому підрозділі продемонстровано повний робочий цикл розробленої інформаційної системи: від процесу навчання моделі до запуску інтерфейсу користувача, та аналізу отриманих результатів. Для демонстрації було використано три різні датасети, що відображають різні предметні області:

1) МРТ-знімки голови — сегментація пухлин мозку,
2) Геознімки (супутникові зображення) — сегментація водойм,
3) Фотографії дорожнього покриття — сегментація тріщин асфальту.

Такий вибір датасетів дозволяє продемонструвати універсальність та адаптивність побудованої гібридної моделі U-Net + Transformer.

Розроблений програмний застосунок містить два основні режими роботи:

1) Навчання моделі на вибраному датасеті,
2) Сегментація зображень у графічному інтерфейсі.

Обидва режими інтегровані в єдину систему через головне меню, яке запускається автоматично при першому старті програми. Головне вікно містить три кнопки.

- «Навчити модель»,
- «Запустити сегментацію»,
- «Вихід».

Після запуску користувач одразу обирає подальший сценарій роботи — або тренує нову модель на одному з доступних датасетів, або переходить до графічного інтерфейсу для сегментації. Інтерфейс головного меню подано на Рис. 3.10.

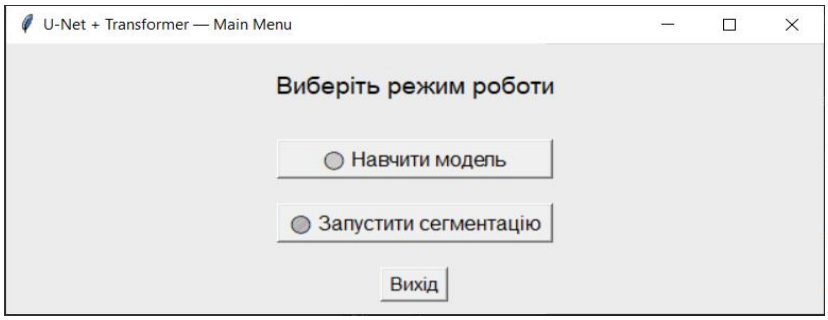


Рисунок 3.10 — Скріншот головного меню програми при першому запуску

3.3.1. Процес навчання моделі

У разі вибору пункту *«Навчити модель»* відкривається консольний інтерфейс в якому пропонується дати ім’я моделі яка буде навчатись (рис. 3.11)

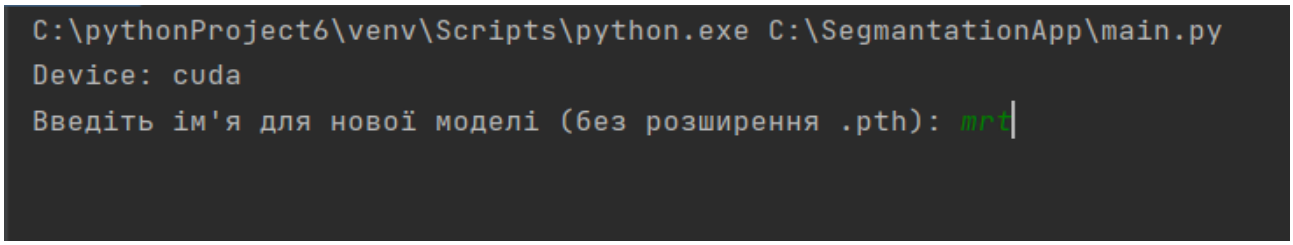


Рисунок 3.11 — Скріншот консольного інтерфейсу в якому пропонується дати ім’я моделі

Для демонстрації розглянемо процес тренування на датасеті мрт-знімків голови, що містить зображення мозку та відповідні маски пухлин (рис 3.12)

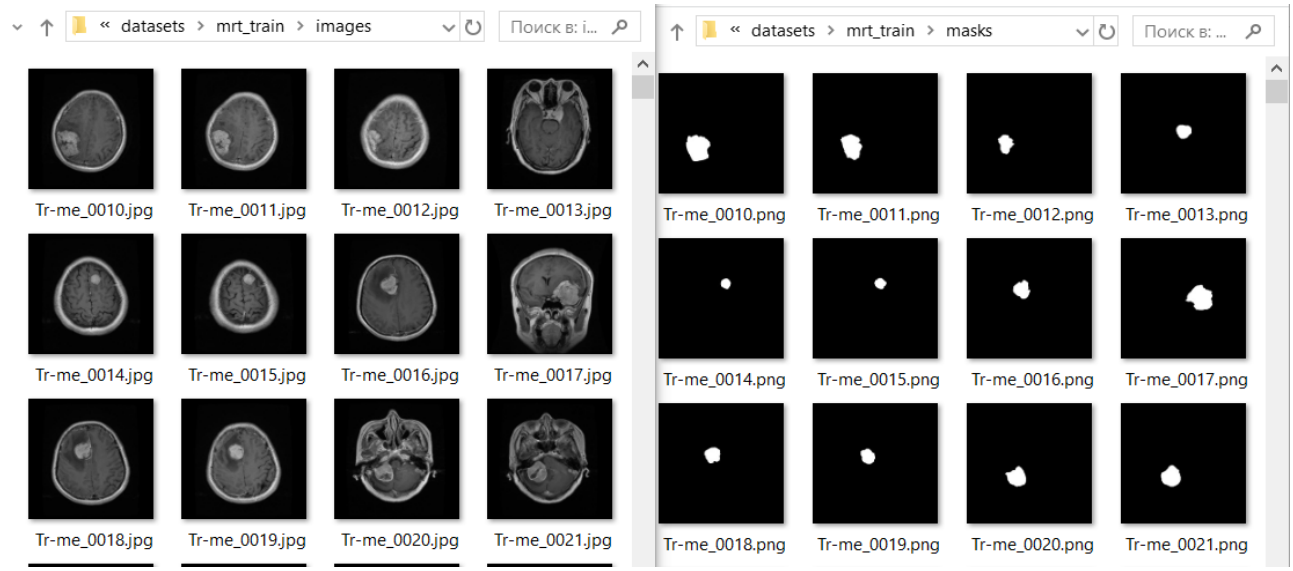


Рисунок 3.12 — Скріншот датасету мрт знімків голови з масками

Дамо імя моделі «mrt» (рис. 3.11) та запустимо тренування. Одразу після запуску автоматичний пошук на відповідність пар “зображення–маска”. Цей механізм гарантує, що модель навчається на відповідних парах, а не на випадкових файлах. Знайдено 999 співпадінь зображення–маска (рис 3.13.)

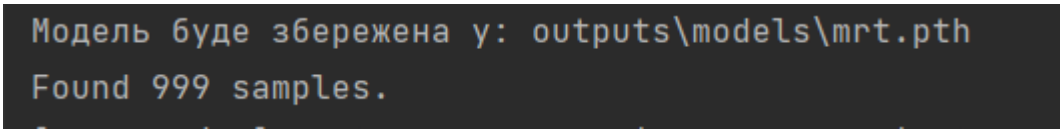


Рисунок 3.13 — Скріншот відповідності співпадінь “зображення–маска”

Під час тренування користувач отримує в консолі детальний журнал (рис. 3.14):

- номер епохи (Epoch)
- середню функцію втрат (TrainLoss)
- значення метрик точності (IoU, Dice, Accuracy)
- час проходження епохи (TimeSec)

Після закінчення тренування журнал зберігається у окремий txt. файл для зручності читання у будь-який час.

```
[Epoch 1/25] TrainLoss=0.6510 | IoU=0.4598 | Dice=0.5998 | Acc=0.9803 | Time=56.1s
>>> Saved BEST model
[Epoch 2/25] TrainLoss=0.5461 | IoU=0.5467 | Dice=0.6640 | Acc=0.9827 | Time=53.0s
>>> Saved BEST model
[Epoch 3/25] TrainLoss=0.4659 | IoU=0.5306 | Dice=0.6414 | Acc=0.9782 | Time=53.7s
[Epoch 4/25] TrainLoss=0.3860 | IoU=0.6607 | Dice=0.7691 | Acc=0.9895 | Time=53.3s
>>> Saved BEST model
[Epoch 5/25] TrainLoss=0.3036 | IoU=0.6538 | Dice=0.7604 | Acc=0.9832 | Time=53.5s
[Epoch 6/25] TrainLoss=0.2275 | IoU=0.7349 | Dice=0.8232 | Acc=0.9905 | Time=53.5s
>>> Saved BEST model
```

Рисунок 3.14 — Скріншот журналу тренування у консолі

Після завершення кількох епох спостерігається стабільне зменшення функції втрат і зростання метрик точності (рис. 3.14), що підтверджує коректність реалізованої архітектури. Навчання виконувалося в 25 епох, найкраща модель автоматично зберігається і перезаписує попередню в спеціальній директорії де зберігаються навчені моделі. Після завершення останньої епохи модель видає найкращу навчену модель у консолі по IoU, зберігає модель, метрики, і графіки втрат у спеціально відведені для цього директорії і повертає нас у головне меню (GUI) (рис 3.15)

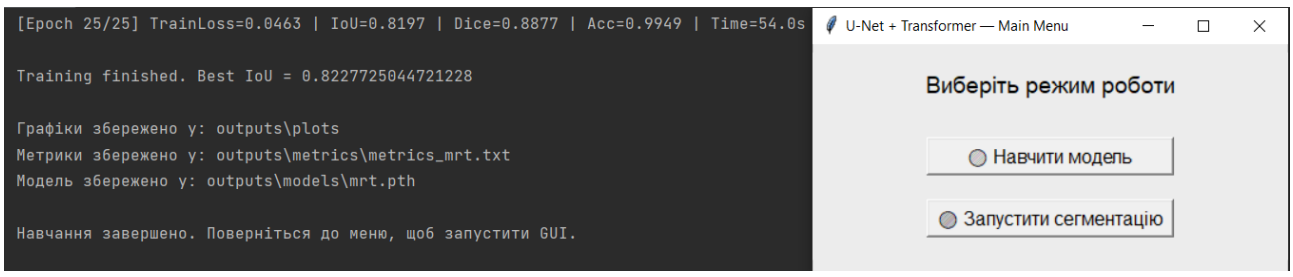


Рисунок 3.15 — Скріншот завершеного навчання

Для кожного датасету автоматично будуються графіки (рис. 3.16), (рис. 3.17):

- зміни функції втрат (Loss),
- показника IoU,
- показника Dice,
- точності Accuracy.

Графіки дозволяють оцінити динаміку збіжності моделі, виявити пере- чи недонавчання.

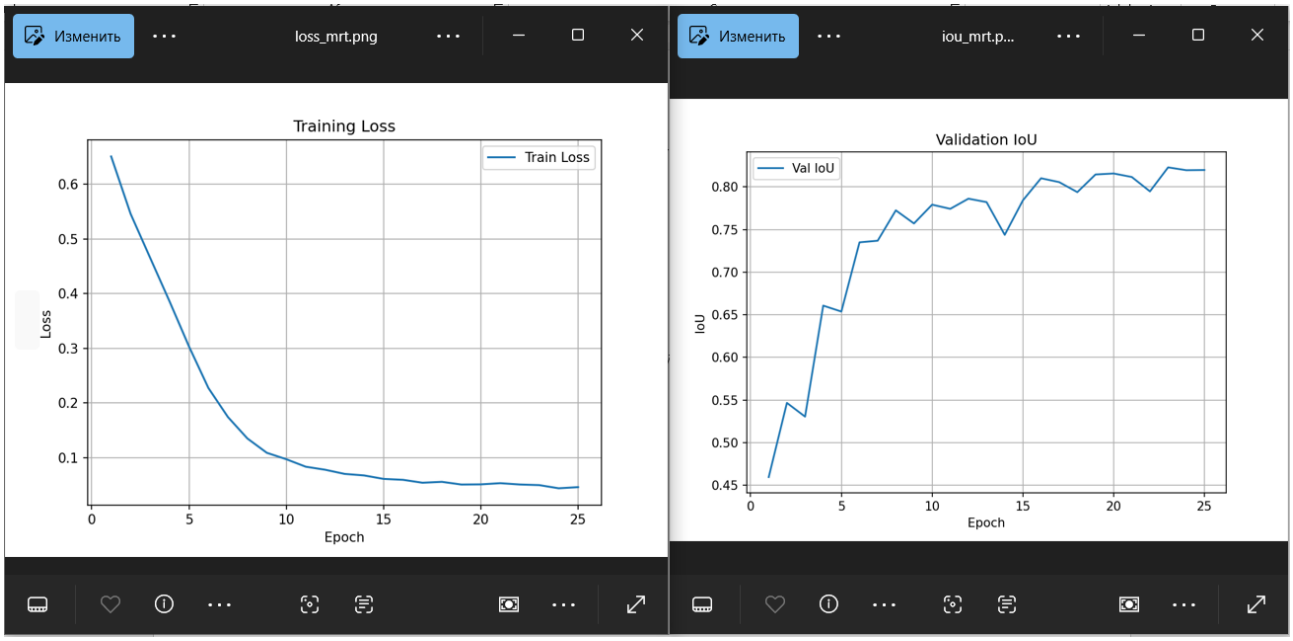


Рисунок 3.16 — Скріншот графіків ”Training Loss та IoU”

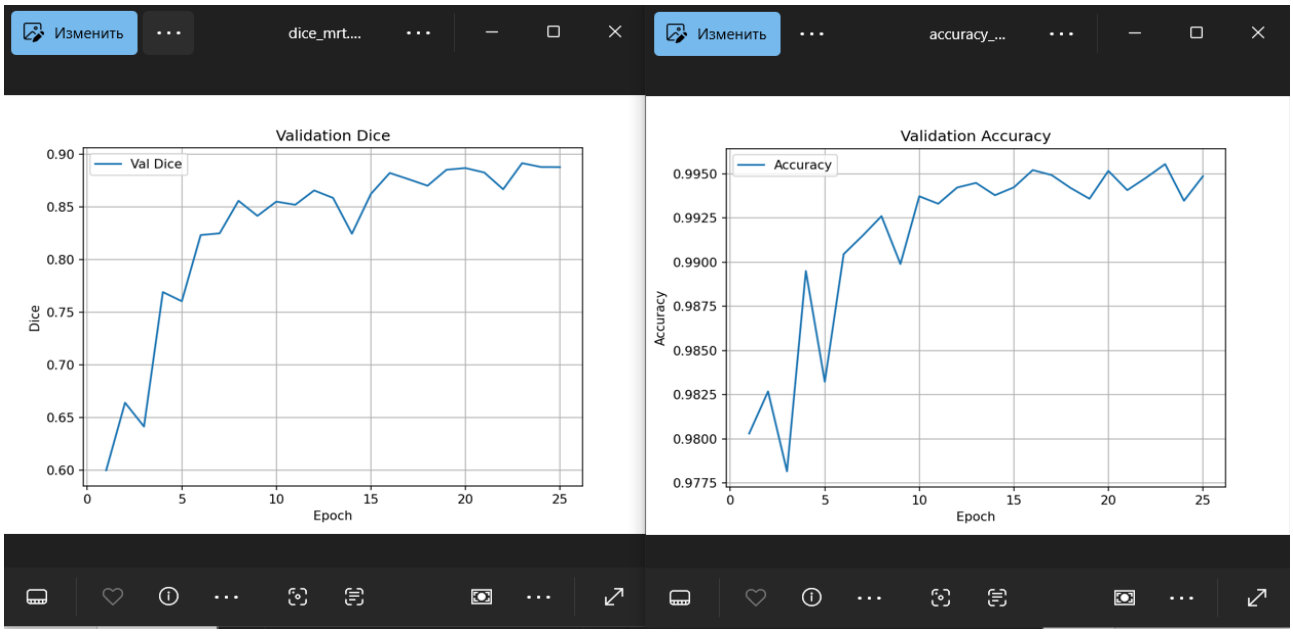


Рисунок 3.17 — Скріншот графіків ”Dice та Accuracy”

Аналіз графіків показує:

Спадання втрат зображене на графіку Training Loss (рис. 3.16) є монотонним і стабільним — від 0.65 на першій епосі до 0.04 на 25-й епосі. Це вказує на ефективне та збалансоване навчання без перенавчання.

На рис. 3.16 показано зростання показника Intersection over Union (IoU). IoU стабільно зростає від 0.45 до 0.82, що є високим значенням для медичних зображень.

На рис. 3.17 наведено графік Dice Score, який демонструє плавне зростання та досягає 0.88–0.90 у фінальних епохах. Метрика Dice є однією з найважливіших у медицині, оскільки показує якість перетину контурів між прогнозом та істинною маскою.

І на останок на рис. 3.17 показано метрику Accuracy, яка досягає показника понад 0.995, оскільки Accuracy у задачах сегментації схильна завищувати результат (через велику кількість фонового класу), її стабільно високі значення додатково підтверджують ефективність навченої моделі.
В таблиці 3.1 представлені підсумкові метрики для найкращої епохи (23 епоха).

Таблиця 3.1 – Метрики найкращої епохи

Епоха	Train Loss	IoU	Dice	Accuracy
23	0.0500	0.8228	0.8915	0.9956

Отримані результати демонструють стабільну роботу моделі, хорошу збіжність і високу якість сегментації пухлин мозку. В таблиці 3.2 представлені підсумкові метрики найкращих епох навчання, на трьох тестових датасетах.

Таблиця 3.2 – Метрики навчання для трьох тестових датасетів

Модель	Найкраща епоха	Train Loss	IoU	Dice	Accuracy
mrt	23	0.0500	0.8228	0.8915	0.9956
water	19	0.3668	0.7533	0.8438	0.9319
crack	21	0.2363	0.7006	0.7921	0.8859

З аналізу метрик:

- Модель «mrt» показала найкращі результати завдяки однорідності медичних даних та чітким контурам об'єкта сегментації.
- Модель «water» демонструє середню точність через складність рельєфів та шум у супутникових знімках.
- Модель «crack» має високу Accuracy, але нижчі IoU/Dice — що типово для задач з дуже тонкими об'єктами (тріщинами).

3.3.2 Демонстрація сегментації у графічному інтерфейсі

Після завершення навчання моделі користувач може перейти до режиму сегментації в графічному інтерфейсі. Для цього у головному меню необхідно обрати пункт «Запустити сегментацію». Після вибору відкривається основне вікно застосунку, яке забезпечує взаємодію з моделлю у зручному візуальному форматі (рис. 3.18).

У вікні GUI доступні такі елементи:

- кнопка «Завантажити зображення»;
- кнопка «Запустити сегментацію»;
- кнопка «Зберегти маску»
- кнопка «Вийти»
- панель відображення оригінального зображення;
- панель відображення маски сегментації;
- панель відображення накладеної маски;
- модуль виводу метрик IoU, Dice, Precision, Recall, Accuracy (при наявності еталонної маски).
- рядок статусу, що показує стан системи

Інтерфейс побудований на основі Tkinter і забезпечує миттєвий інференс моделі на вибраному зображенні. GUI завантажує попередньо навчену модель із каталогу збережених ваг і автоматично виконує інференс при натисканні відповідної кнопки.

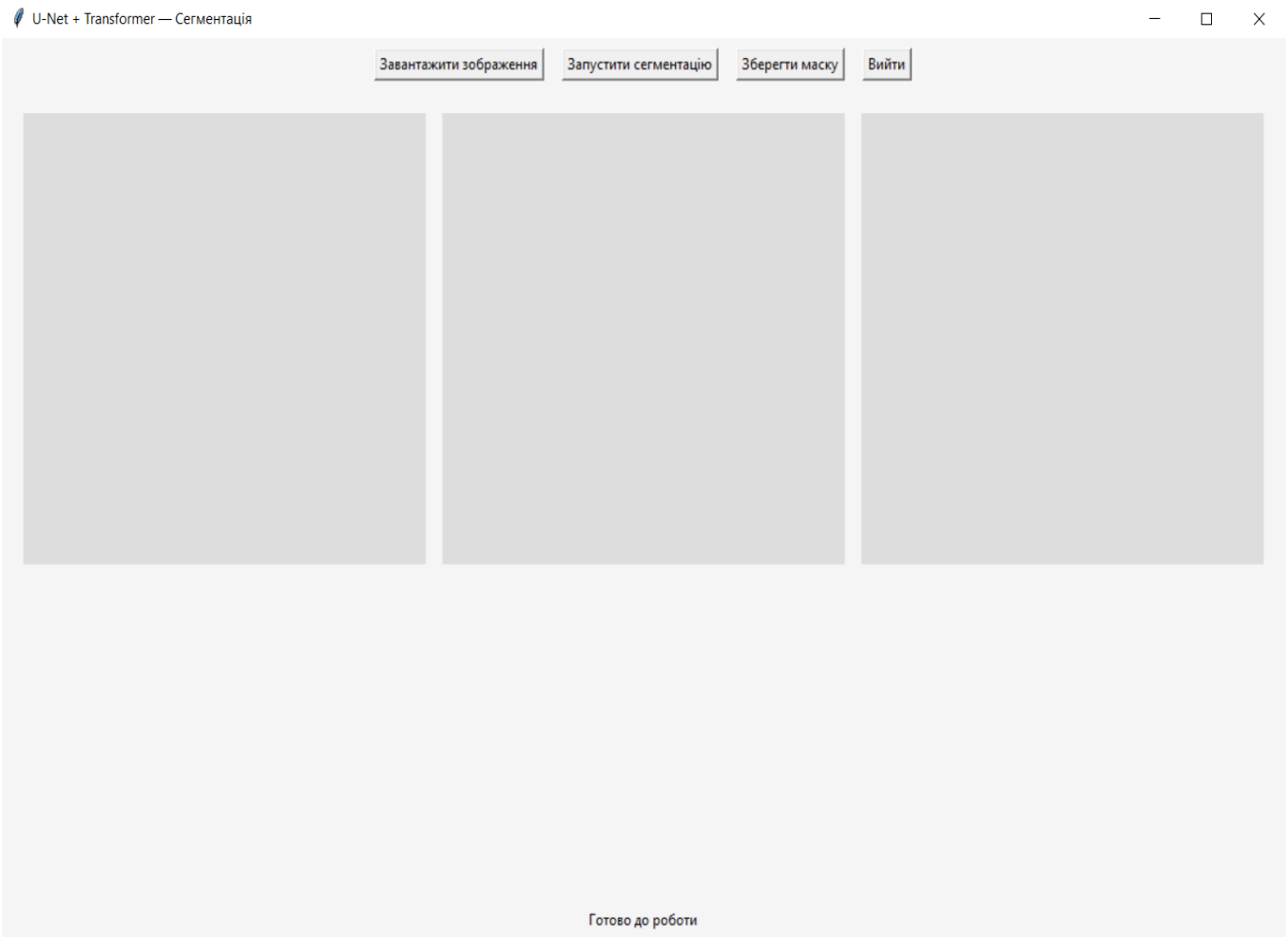


Рисунок 3.18 — Скріншот запуску (GUI) сегментації

Після завантаження зображення та запуску сегментації, при наявності еталонної маски зліва можна побачити метрики точності сегментації моделі. Подано на рис. 3.19 на прикладі сегментації мрт знімку пухлини мозку.

- IoU (Intersection over Union) — ступінь перекриття передбаченої та істинної маски.
- Dice коефіцієнт — подвоєна міра збігу областей, стійка до дисбалансу класів.
- Precision — частка правильних позитивних передбачень.
- Recall — здатність моделі знайти всі реальні пікселі об’єкта.
- Accuracy — загальна точність класифікації пікселів.

Ці метрики дозволяють користувачу миттєво оцінити якість результатів та порівняти її з еталонними даними.

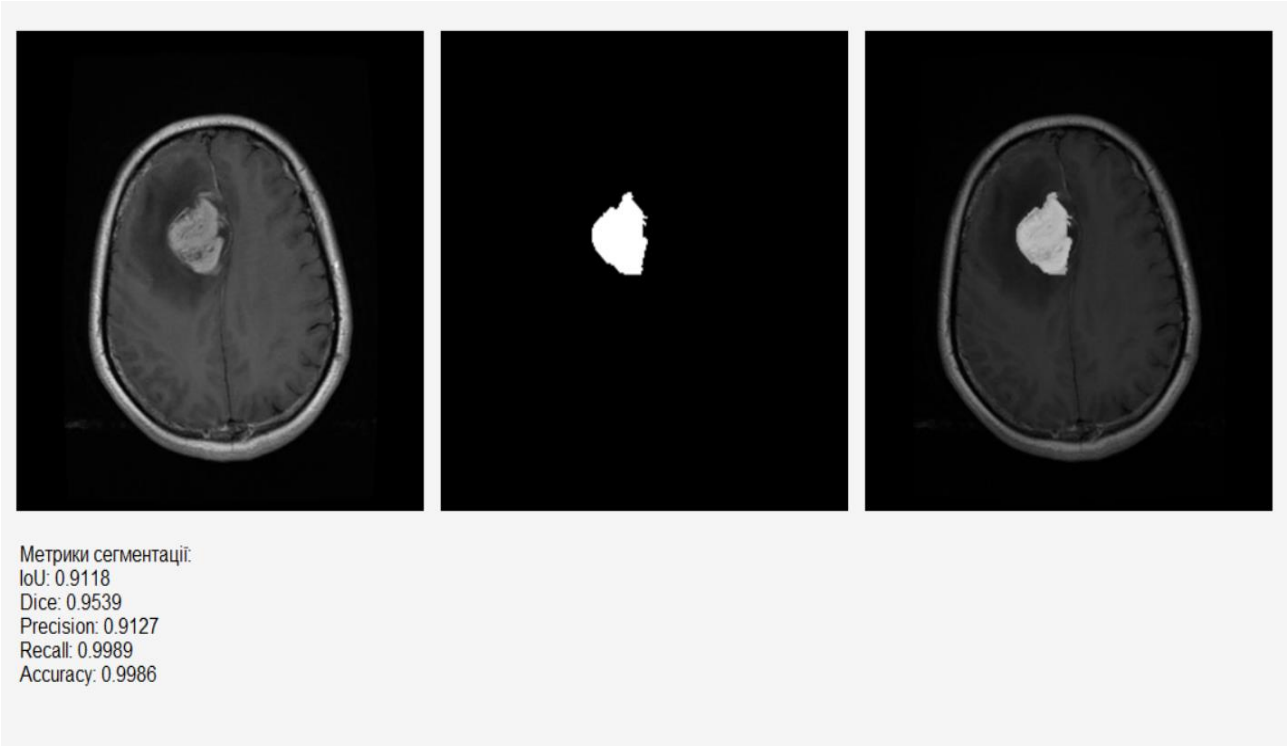


Рисунок 3.19 — Скріншот сегментації мрт знімку пухлини мозку з метриками

Після виконання успішної сегментації, отримане накладення маски на оригінальне зображення можна зберегти у будь-яку зручну для користувача директорію (рис. 3.20).

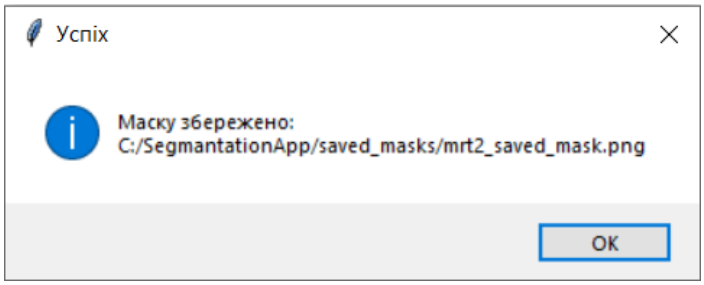


Рисунок 3.20 — Скріншот успішно збереженої накладеної маски

Демонстрація сегментації на трьох вибірках:

Для оцінки роботи моделі в реальних умовах було проведено сегментацію трьох тестових вибірок:

1) МРТ-знімки голови (модель «*mrt*»)
2) Геознімки водойм (модель «*water*»)
3) Дорожнє покриття з тріщинами (модель «*crack*»)

У кожному випадку GUI відображає результат у трьох форматах:

- вхідне зображення,
- бінарна маска сегментації,
- накладення маски на зображення.

Візуальні приклади сегментованих вибірок подано на рисунках (рис. 3.21 — 3.23).

Сегментація МРТ-знімків мозку

Мета моделі — відокремити патологічну тканину від здорової.

Особливості:

- низький контраст;
- складні границі об’єкта;
- великі варіації розміру пухлин.

Для моделі «*mrt»* сегментація пухлини демонструє чітке окреслення паталогічних областей.

На рисунку 3.21 можна побачити:

- чітку форму сегментованої пухлини,
- коректне відтворення контурів,
- відсутність накладання на здорові області мозку.

За наявності еталонної маски GUI автоматично обчислює метрики. Для мрт знімку пухлини мозку модель «*mrt»* показала такі результати:

- IoU = 0.8356
- Dice = 0.9105
- Precision = 0.9837
- Recall = 0.8474
- Accuracy = 0.9986

Ці показники підтверджують відповідність якості сегментації результатам, отриманим у процесі тренування.

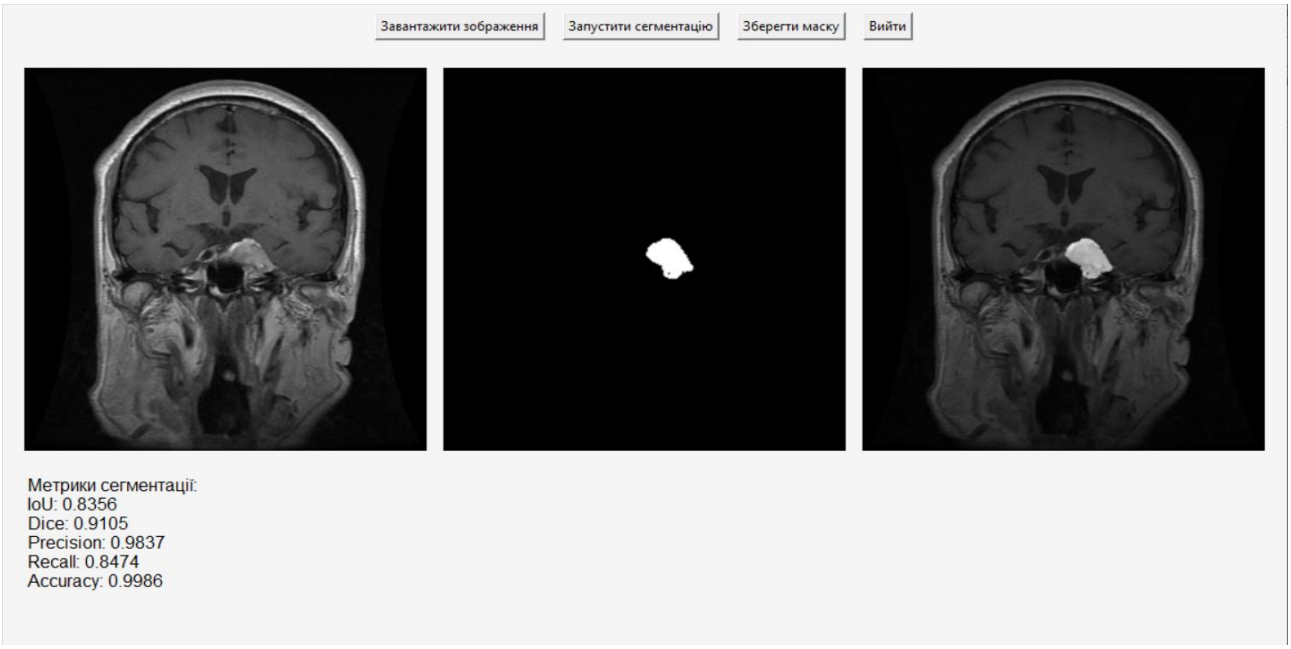


Рисунок 3.21 — Скріншот сегментації пухлини мозку у GUI

Сегментація водойм на геознімках

Друга тестова вибірка — супутникові знімки місцевості. Модель дозволяє відокремлювати річки, озера та водосховища від рослинності, ґрунту або міських структур.

Особливості задачі:

- неоднорідні текстури територій;
- варіації кольору води;
- наявність тіней та шумів.

Модель «*water*» демонструє стабільну роботу навіть за умов наявності складних рельєфних форм, тіней та неоднорідних текстур.

На рисунку 3.22 демонструється приклад:

- водойма коректно виділена;
- дрібні артефакти відсутні;
- межа водойми визначена досить точно, попри розмитість країв на знімку.

Метрики для прикладу:

- IoU = 0.8142
- Dice = 0.8976

- Precision = 0.9741
- Recall = 0.8321
- Accuracy = 0.9548

Це узгоджується з підсумковими метриками моделі «*water*», наведеними у таблиці 3.2, а деякі показники такі як IoU та Dice навіть трішки кращі.

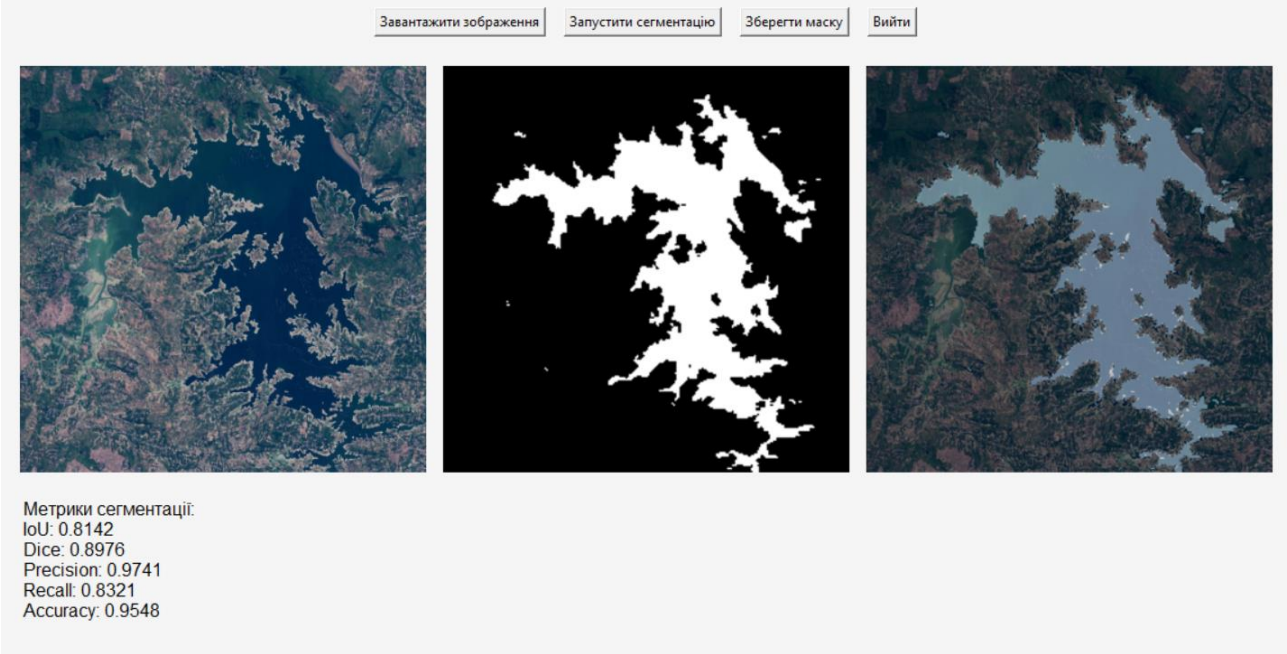


Рисунок 3.22 — Скріншот сегментації водойми на геознімку

Сегментація тріщин дорожнього покриття

Сегментація тріщин є однією з найскладніших задач для моделей сегментації через:

- надтонку структуру об’єкта,
- високий рівень шуму,
- неймовірно малу площу класу тріщини відносно фонового зображення.

На рисунку 3.23 наведено приклад роботи моделі «*crack*»:

- тріщини коректно виявлені уздовж усієї довжини,
- модель виділяє навіть дуже тонкі сегменти,
- частково допускається нерівномірність контурів, що є типовим для цього класу задач.

Метрики для цього зображення:

- IoU = 0.6463
- Dice = 0.7851
- Precision = 0.7565
- Recall = 0.8160
- Accuracy = 0.9944

Хоча Accuracy залишається високою, IoU та Dice є більш показовими і відображають реальну точність сегментації тонких структур.

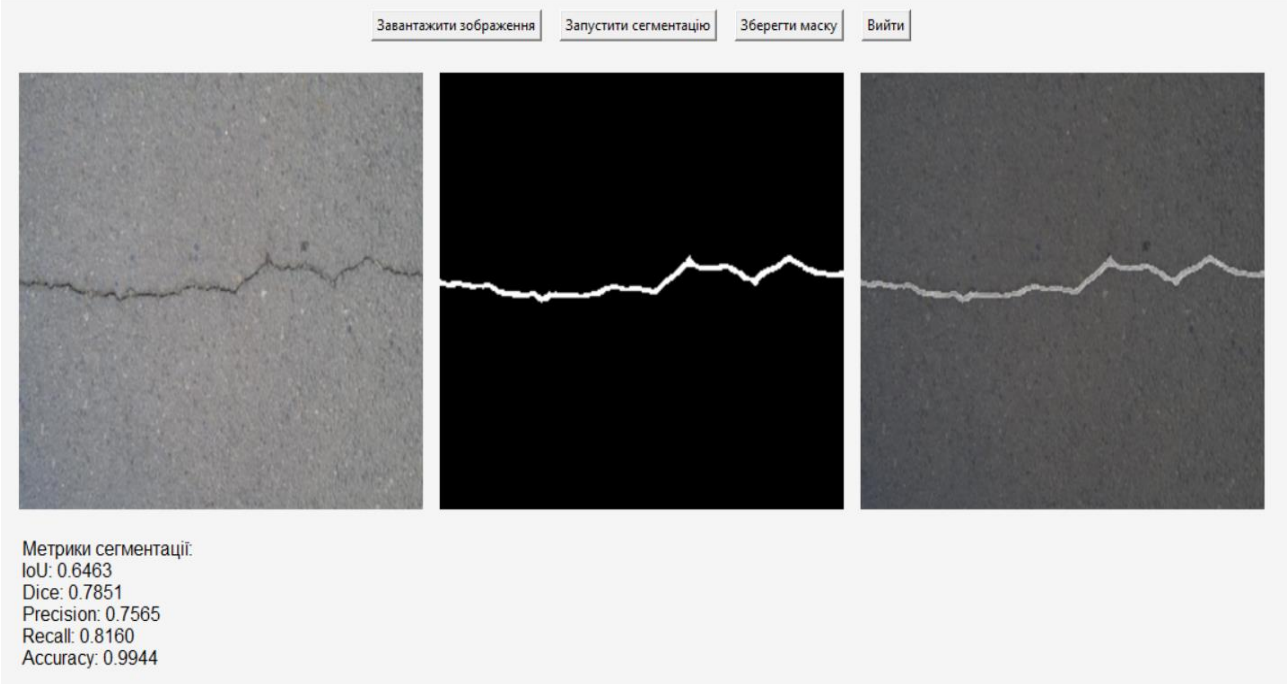


Рисунок 3.23 — Скріншот сегментації тріщин дорожнього покриття

За результатами тестування у графічному інтерфейсі можна зробити такі висновки:

1) Розроблена система забезпечує повний цикл роботи — від навчання моделі до її використання в GUI.
2) Моделі, навчені на різних датасетах, демонструють стабільну та надійну роботу у межах своїх доменів.
3) Найвищу точність показала модель «*mrt*», що пов’язано з однорідністю даних та чіткістю контурів.
4) Модель «*water*» коректно працює з супутниковими знімками, попри їхню неоднорідність та шум.

5) Модель «*crack*» успішно сегментує тріщини, навіть попри складність структури цього класу.
6) Графічний інтерфейс забезпечує зручність аналізу, візуалізацію результатів та автоматичний розрахунок основних метрик.

Таким чином, розроблений застосунок підтвердив свою ефективність для задач сегментації різних типів зображень та може бути адаптований для ширшого спектра практичних застосувань.

Висновки за розділом

У цьому розділі було повністю реалізовано, структуровано та експериментально досліджено програмну систему сегментації зображень на основі гібридної архітектури U-Net + Transformer, розроблену за модульним принципом. Створена система об’єднує сучасні підходи глибинного навчання та забезпечує можливість як навчання моделей на нових датасетах, так і інтерактивної сегментації в графічному інтерфейсі.

У процесі роботи розроблено та протестовано такі ключові компоненти:

- створено модульну структуру застосунку, що включає окремі модулі для конфігурації, моделі, датасету, функцій втрат, тренувального контуру, інференсу та GUI;
- реалізовано гібридну модель сегментації з використанням згорткових блоків U-Net і трансформерного блоку самоуваги, що забезпечує поєднання локальних і глобальних ознак;
- розроблено консольний механізм навчання з автоматичним збереженням найкращої моделі, побудовою графіків метрик та формуванням текстового журналу навчання;
- створено графічний інтерфейс сегментації на базі Tkinter, який дає змогу завантажувати зображення, виконувати інференс, переглядати маску та накладення, оцінювати метрики та зберігати результати;

- проведено експериментальні дослідження на трьох різних датасетах: медичні МРТ-знімки, супутникові геознімки водойм, зображення дорожнього покриття з тріщинами.

Аналіз навчання показав, що модель демонструє стабільну збіжність, плавне зменшення функції втрат та зростання метрик якості. Найкращі результати отримано на медичних даних (IoU ≈ 0.82, Dice ≈ 0.89), де контури об’єктів є чітко структурованими. Датасет тріщин, відповідно до своєї природи, показав нижчі IoU та Dice через надтонку геометрію об’єкта, що є типовим для таких задач. Геознімки продемонстрували середню точність через неоднорідність сцени.

Результати демонстрації у графічному інтерфейсі підтвердили коректність реалізації інференсу: модель стабільно виділяє об’єкти на зображеннях різного типу, що свідчить про її універсальність. Інтерфейс забезпечує інтуїтивну роботу користувача — від завантаження зображення до отримання фінальної маски та метрик оцінки точності.

Система підтвердила:

- ефективність поєднання CNN і Transformer-блоків у задачах сегментації;
- здатність моделі узагальнювати дані різної природи та структури;
- практичну застосовність створеного програмного комплексу в різних галузях: медицині, детекції водойм на геознімках, контролі інфраструктури тощо;
- зручність модульної архітектури для подальшого розширення.

Розроблений програмний прототип може бути основою для майбутніх покращень, таких як багатокласова сегментація, напівавтоматичне анотування, використання більших моделей Transformer або інтеграція з веб-інтерфейсами.

Таким чином, створена система є повноцінним та гнучким інструментом для задач сегментації зображень, що поєднує теоретичну обґрунтованість, високу точність та практичну зручність використання.

ВИСНОВКИ

У результаті виконання магістерської кваліфікаційної роботи на тему «Розробка інформаційної системи для сегментації зображень із використанням методів штучного інтелекту» було вирішено комплекс наукових, теоретичних і практичних завдань, спрямованих на створення ефективного інструменту для автоматизованої обробки візуальної інформації.

Проведене дослідження показало, що сегментація зображень є одним із ключових напрямів сучасного комп'ютерного зору, який забезпечує можливість виділення суттєвих об'єктів і структур на зображенні. Від якості виконання цього етапу залежить точність подальших операцій розпізнавання, класифікації чи прогнозування. Актуальність роботи підтверджується зростанням обсягів цифрових даних, необхідністю швидкої обробки великих масивів інформації та впровадженням інтелектуальних систем у медицині, транспорті, та геоінформаційних технологіях.

У першому розділі проведено детальний аналіз сучасного стану проблеми сегментації зображень. Розглянуто основні підходи — класичні (порогові, контурні, кластеризаційні) та сучасні (глибокі нейронні мережі, трансформерні архітектури). Визначено, що традиційні методи, зокрема алгоритм Отсу, оператори Собеля та методи кластеризації, залишаються ефективними лише для задач із простою структурою зображення. Однак вони демонструють низьку точність при наявності шумів, тіней або розмитих меж. Методи глибокого навчання, навпаки, забезпечують високу стійкість до таких факторів, дозволяючи досягати точності сегментації понад 90 %. Особливу увагу приділено архітектурам U-Net, DeepLab, ResNet, Vision Transformer (ViT) та Swin Transformer, які стали основою сучасних систем сегментації.

Крім того, у першому розділі було проаналізовано наукові джерела, патенти та вітчизняні дослідження, що підтверджують тенденцію до інтеграції CNN і трансформерних підходів. На основі цього обґрунтовано вибір напряму дослідження — створення гібридної моделі U-Net + Transformer, яка поєднує

локальну чутливість згорткових мереж із глобальним контекстом, що забезпечується механізмом самоуваги.

У другому розділі розроблено математичну та алгоритмічну модель процесу сегментації та гібридної архітектури U-Net + Transformer. Математичне формулювання дозволило описати процес сегментації як стохастичну систему з оцінкою похибки та врахуванням параметрів оптимізації. На цій основі побудовано архітектурну модель гібридної нейронної мережі, яка складається з трьох основних частин: Encoder, Transformer Block та Decoder.

У підрозділі 2.3 було розроблено алгоритмічну модель, що формалізує процес сегментації у вигляді послідовності дій — від підготовки та нормалізації даних до обчислення метрик точності (IoU, Dice, Precision, Recall, Accuracy). Проведено аналітичну оцінку адекватності моделі, яка показала потенційне підвищення показників точності на 5–10 % порівняно з базовою архітектурою U-Net.

У третьому розділі здійснено програмну реалізацію розробленої моделі засобами Python із використанням бібліотек PyTorch, Torchvision, NumPy, Pillow (PIL), Tkinter, Matplotlib, та time, os. Створено функціональний прототип модульної інформаційної системи для сегментації зображень, який має графічний інтерфейс користувача.

Інтерфейс забезпечує виконання основних функцій:

- Навчання моделі на будь-яких вибірках зображення-маска;
- збереження найкращої моделі за метрикою IoU.
- завантаження вхідних зображень різних форматів;
- запуск процесу сегментації на основі навченої гібридної моделі;
- відображення оригінального зображення, передбаченої маски та результату накладання;
- збереження отриманих результатів у задану директорію.

Було реалізовано коди основних архітектурних компонентів — блоків кодування, трансформерного шару та декодера з механізмом skip-з’єднань.

Проведено навчання моделі на вибірці трьох різних тестових датасетів з різними типами зображень, а саме МРТ-знімки голови, супутникові знімки водойм та фотографії дорожніх тріщин. Виконано експериментальне дослідження програми та оцінювання результатів. На цих вибірках модель продемонструвала високу якість сегментації. Зокрема, на медичному датасеті досягнуто значень IoU = 0.8228 та Dice = 0.8915, Датасет тріщин очікувано дав нижчі значення IoU та Dice, оскільки об’єкт має дуже тонку та складну геометрію — це характерно для подібних задач сегментації. У випадку геознімків точність виявилася середньою через виражену різнорідність структури зображення та складність фону.

Таким чином, модель забезпечує високу узгодженість між передбаченими та еталонними масками, стабільність до шумів і відмінну здатність до узагальнення. Аналіз похибок показав, що основні помилки виникають у ділянках з низьким контрастом або перетином об’єктів, що є типовим для більшості сегментаційних моделей і може бути мінімізовано подальшою оптимізацією параметрів.

Практичні результати довели працездатність і коректність розробленої системи. Створений програмний продукт може бути адаптований до різних предметних галузей — від медичної діагностики до транспортного контролю якості та геоінформаційного аналізу. Завдяки модульній архітектурі програму можна масштабувати або доповнювати новими функціональними блоками (наприклад, автоматичним анотуванням, розширенням на багатокласову сегментацію, інтеграцією з базами даних).

Отже, у процесі виконання кваліфікаційної роботи досягнуто таких основних результатів:

1. Проаналізовано сучасний стан проблеми сегментації зображень і класифіковано основні методи.
2. Побудовано математичну модель задачі, що враховує стохастичний характер вхідних даних та комбіновану функцію втрат.

3. Розроблено архітектуру гібридної нейронної мережі U-Net + Transformer та створено алгоритмічну модель процесу сегментації.
4. Реалізовано програмну систему з графічним інтерфейсом користувача, яка забезпечує повний цикл сегментації — від навчання моделі до завантаження даних та візуалізації результатів.
5. Проведено експериментальні дослідження, що підтвердили ефективність і адекватність запропонованого підходу.
6. Отримано практичний програмний продукт, який може бути використаний як базова платформа для подальших досліджень у сфері комп’ютерного зору.

Узагальнюючи результати, можна зробити висновок, що поставлена мета — розробка інформаційної системи для сегментації зображень із використанням методів глибокого навчання — успішно досягнута. Розроблена система поєднує високу точність, стабільність та зручність використання, що дозволяє рекомендувати її як перспективний інструмент для практичного застосування у різних галузях науки і техніки.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Андрієнко Є. Сегментація зображень : Курсова робота. Київ, 2022. 37 с. URL: https://surl.li/dnzlgn (дата звернення: 02.02.2025).
2. Подчашинський Ю. Дослідження методів сегментації зображень для отримання вимірювальної інформації. Житомир, 2019. 2 с. URL: https://surl.li/qdpmcd (дата звернення: 02.02.2025).
3. Творошенко І. С. Цифрова обробка зображень : Конспект лекцій. Харків, 2017. 74 с. URL: https://surl.li/lzykpx (дата звернення: 02.02.2025).
4. Методи нормалізації, розпізнавання, та обробки зображень у системах комп'ютерного зору : Стаття. 2018. URL: https://surl.li/ndussm (дата звернення: 02.02.2025).
5. Krylov V. N. Vector-difference segmentation method in technical and medical express diagnostic systems. Odesa, 2020. URL: https://surl.li/jnjbug (date of access: 02.02.2025).
6. Сегментація зображень засобами openCV. URL: https://surl.li/hksvfx (дата звернення: 02.02.2025).
7. Андрікевич С. А. Методи сегментації оптичних зображень очного дна : Стаття. Вінниця, 2024. 11 с. URL: https://surl.li/zbaupr (дата звернення: 02.02.2025).
8. Коменчук О. Інформаційна технологія прискореного анотування медичних зображень в задачах сегментації на основі моделей глибокого навчання : Стаття. Вінниця, 2024. 9 с. URL: https://surl.li/smphdg (дата звернення: 02.02.2025).
9. Гайдук В. І. Методи сегментації зображень в задачах розпізнавання обличь : Кваліфікаційна робота. Тернопіль, 2024. 65 с. URL: https://surl.li/lgvvwl (дата звернення: 02.02.2025).
10. Рупіч С. С. Сегментація супутникових знімків споруд з використанням нейронних мереж : Стаття. Київ, 2024. 4 с. URL: https://surl.li/kvinzt (дата звернення: 02.02.2025).

11. Луп’як Д.Д. Методи сегментації зображень на основі графів : магістерська робота. Вінниця, 2018. 24 с. URL: https://surli.cc/zmgzxo (дата звернення: 24.08.2025).
12. Ronneberger O. U-Net: convolutional networks for biomedical image segmentation : conference paper. Germany, 2015. 8 p. URL: https://surl.li/vxizew (date of access: 02.02.2025).
13. Kaiming H. Deep residual learning for image recognition. Las Vegas, 2016. 12 p. URL: https://surl.li/smbcpr (date of access: 02.02.2025).
14. Dosovitskiy A. An image is worth 16x16 words: transformers for image recognition at scale : conference paper. 2021. 21 p. URL: https://surl.li/rocmxm (date of access: 02.02.2025).
15. Long J. Fully convolutional networks for semantic segmentation : conference paper. 2017. 10 p. URL: https://surl.li/tcsqpl (date of access: 02.02.2025).
16. Chen L. Rethinking atrous convolution for semantic image segmentation : conference paper. 2017. 11 p. URL: https://surl.li/tcsqpl (date of access: 02.02.2025).
17. Xie S. Aggregated residual transformations for deep neural networks : conference paper. 2017. 9 p. URL: https://surl.li/tfspcj (date of access: 02.02.2025).
18. Zhou Z. UNet++: a nested u-net architecture for medical image segmentation : conference paper. USA, 2018. URL: https://surl.li/ldyazj (date of access: 02.02.2025).
19. Chen T. A simple framework for contrastive learning of visual representations : conference paper. Vienna, 2020. 11 p. URL: https://surl.li/yelmal (date of access: 02.02.2025).
20. Simonyan K. Very deep convolutional networks for large-scale image recognition : international conference paper on learning representations. Oxford, 2014. 12 p. URL: https://surl.li/qeyxbb (date of access: 02.02.2025).

21. Kendall A. Bayesian segnet: model uncertainty in deep convolutional encoder-decoder architectures : british machine vision conference paper. Cambridge, 2017. URL: https://surl.li/ngxxlu (date of access: 02.02.2025).
22. Liu Z. Swin Transformer : CVF International Conference on Computer Vision. Montreal, 2021. URL: https://surli.cc/kfyato (date of access: 24.08.2025).
23. Meta A. Segment Anything : Conference paper. Paris, 2023. 12 p. URL: https://surl.li/dmmimj (date of access: 24.08.2025).
24. Cheng B. Mask2Former : Conference paper. New Orleans, USA, 2022. 10 p. URL: https://surl.li/zenfxq (date of access: 24.08.2025).
25. Systems and methods for functional imaging follow-up evaluation using deep neural network : patent US11250569B2 United States. Applied on 04.11.2019 ; published on 15.02.2022. URL: https://surl.li/rrkzoi (date of access: 02.02.2025).
26. Image segmentation method and system based on wide residual pyramid pooling network : patent CN107945185B China. Applied on 29.11.2017 ; published on 07.02.2020. URL: https://surl.li/hvgtjv (date of access: 02.02.2025).
27. Method and apparatus for training image segmentation model, computer device, and storage medium : patent US11961233B2 United States. Applied on 09.09.2021 ; published on 16.04.2024. URL: https://surl.li/xotxve (date of access: 02.02.2025).
28. Mango example confrontation segmentation method based on Mask R-CNN : patent CN110619632B China. Applied on 18.09.2019 ; published on 11.01.2022. URL: https://surl.li/xjpxkv (date of access: 02.02.2025).
29. Brain tumor segmentation network and method based on U-Net network : patent CN111192245B China. Applied on 26.12.2019 ; published on 07.04.2023. URL: https://surl.li/doxfwk (date of access: 02.02.2025).
30. Intelligent segmentation and recognition method of mineral images assisted by large model-fractal combination : patent CN119418173A China. Applied on 06.11.2024 ; published on 11.02.2025. URL: https://surl.lu/aoxmkw (date of access: 24.08.2025).

31. Otsu N. Otsu's method : conference paper. 1979. URL: https://surl.li/fuaojk (date of access: 12.09.2025).
32. Маринич І. А., Тронь В. В. Методичні рекомендації до виконання кваліфікаційної роботи магістра для студентів спеціальності 151 “Автоматизація та комп’ютерно-інтегровані технології”. Кривий Ріг : Видавничий центр КНУ, 2022. 50с.
33. ДСТУ 3008:2015. Звіти у сфері науки і техніки. Структура і правила оформлення. Київ, ДП «УкрННЦ», 2015. 26с. (Інформація та документація).
34. ДСТУ 8302:2015. Бібліографічне посилання. Загальні вимоги та правила складання Київ, ДП «УкрННЦ», 2016. 16 с. (Інформація та документація).
35. ДСТУ 3582:2013. Бібліографічний опис. Скорочення слів і словосполучень в українській мові. Загальні вимоги та правила. Київ, ДП «УкрННЦ», 2013. 23 с. (Інформація та документація)
36. ДСТУ 3651.0-97 Метрологія. Одиниці фізичних величин. Основні одиниці фізичних величин Міжнародної системи одиниць. Основні положення, назви та позначення Київ, Держстандарт України, 1998. 27 с. (Інформація та документація).

ДОДАТОК А

Лістинг коду усіх модулів інформаційної системи для сегментації зображень

Лістинг коду модуля main.py

```
import tkinter as tk
from modules.train import main_train_and_eval
from modules.gui import main_gui_demo

def main_menu():
    root = tk.Tk()
    root.title("U-Net + Transformer — Main Menu")
    root.geometry("400x220")
    root.configure(bg="#ECECEC")

    def run_training():
        root.destroy()
        main_train_and_eval()
        print("\nНавчання завершено. Поверніться до меню, щоб
запустити GUI.")
        main_menu()

    def run_gui():
        root.destroy()
        main_gui()

    tk.Label(root, text="Виберіть режим роботи", font=("Arial",
14), bg="#ECECEC").pack(pady=20)

    tk.Button(root, text=" Навчити модель", font=("Arial", 12),
width=22, command=run_training).pack(pady=10)
    tk.Button(root, text=" Запустити сегментацію", font=("Arial",
12), width=22, command=run_gui).pack(pady=10)
    tk.Button(root, text="Вихід", font=("Arial", 11),
command=root.quit).pack(pady=10)
    root.mainloop()
```

```
if __name__ == "__main__":
    main_menu()
```

Лістинг коду модуля config.py

```
import torch  (CPU/CUDA)

CONFIG = {
    "img_size": (256, 256),
    "batch_size": 2,


    "lr": 1e-4,
    "num_epochs": 25,
    "device": "cuda" if torch.cuda.is_available() else "cpu",
    "dataset_root": "./crack_train",
    "model_save_path": "./outputs/models/crack_trainV1.pth",
    "threshold": 0.5,
}
```

Лістинг коду модуля dataset.py

```
import os
from PIL import Image
from torch.utils.data import Dataset

class SegmentationDataset(Dataset):

    def __init__(self, images_dir, masks_dir, transform=None,
mask_transform=None):
        self.images_dir = images_dir
        self.masks_dir = masks_dir

        imgs = [f for f in os.listdir(images_dir)
                if f.lower().endswith(('.png', '.jpg', '.jpeg'))]
```

```
        masks = [f for f in os.listdir(masks_dir)
                 if f.lower().endswith(('.png', '.jpg', '.jpeg'))]

        img_stems = {os.path.splitext(f)[0] for f in imgs}
        mask_stems = {os.path.splitext(f)[0] for f in masks}

        common_stems =
sorted(list(img_stems.intersection(mask_stems)))

        self.images = []
        self.masks = []

        for stem in common_stems:
            img_file = next(f for f in imgs if
os.path.splitext(f)[0] == stem)
            mask_file = next(f for f in masks if
os.path.splitext(f)[0] == stem)
            self.images.append(img_file)
            self.masks.append(mask_file)

        self.transform = transform
        self.mask_transform = mask_transform

    def __len__(self):
        return len(self.images)

    def __getitem__(self, idx):
        img_path = os.path.join(self.images_dir, self.images[idx])
        mask_path = os.path.join(self.masks_dir, self.masks[idx])

        img = Image.open(img_path).convert("RGB")
        mask = Image.open(mask_path).convert("L")

        if self.transform:
```

```
            img = self.transform(img)

        if self.mask_transform:
            mask = self.mask_transform(mask)

        mask = (mask > 0.5).float()

        return img, mask, self.images[idx]
```

Лістинг коду модуля model.py

```
import torch
import torch.nn as nn
import torch.nn.functional as F

class EncoderBlock(nn.Module):
    def __init__(self, in_ch, out_ch, pool=True):
        super().__init__()
        self.conv1 = nn.Conv2d(in_ch, out_ch, kernel_size=3,
padding=1)
        self.bn1 = nn.BatchNorm2d(out_ch)
        self.conv2 = nn.Conv2d(out_ch, out_ch, kernel_size=3,
padding=1)
        self.bn2 = nn.BatchNorm2d(out_ch)
        self.pool = nn.MaxPool2d(2) if pool else None

    def forward(self, x):
        x = F.relu(self.bn1(self.conv1(x)))
        x = F.relu(self.bn2(self.conv2(x)))
        p = self.pool(x) if self.pool is not None else x
        return x, p
class DecoderBlock(nn.Module):
    def __init__(self, in_ch, skip_ch, out_ch):
        super().__init__()
        self.up = nn.ConvTranspose2d(in_ch, out_ch, kernel_size=2,
```

```
stride=2)
        self.conv1 = nn.Conv2d(out_ch + skip_ch, out_ch,
kernel_size=3, padding=1)
        self.bn1 = nn.BatchNorm2d(out_ch)
        self.conv2 = nn.Conv2d(out_ch, out_ch, kernel_size=3,
padding=1)
        self.bn2 = nn.BatchNorm2d(out_ch)

    def forward(self, x, skip):
        x = self.up(x)
        if x.shape[2:] != skip.shape[2:]:
            skip = F.interpolate(skip, size=x.shape[2:],
mode='bilinear', align_corners=False)
        x = torch.cat([x, skip], dim=1)
        x = F.relu(self.bn1(self.conv1(x)))
        x = F.relu(self.bn2(self.conv2(x)))
        return x

class TransformerBlock(nn.Module):
    def __init__(self, dim, num_heads=4, mlp_ratio=4.0,
dropout=0.0):
        super().__init__()
        self.dim = dim
        self.num_heads = num_heads
        assert dim % num_heads == 0, "dim must be divisible by
num_heads"
        self.attn = nn.MultiheadAttention(embed_dim=dim,
num_heads=num_heads, dropout=dropout)
        self.norm1 = nn.LayerNorm(dim)
        mlp_hidden = int(dim * mlp_ratio)
        self.ff = nn.Sequential(
            nn.Linear(dim, mlp_hidden),
            nn.GELU(),
            nn.Linear(mlp_hidden, dim)
        )
```

```
        self.norm2 = nn.LayerNorm(dim)

    def forward(self, x):
        b, c, h, w = x.shape
        x_flat = x.flatten(2).permute(2, 0, 1).contiguous()
        attn_out, _ = self.attn(x_flat, x_flat, x_flat)
        x2 = self.norm1(x_flat + attn_out)
        ff_out = self.ff(x2)
        x3 = self.norm2(x2 + ff_out)
        x_out = x3.permute(1, 2, 0).view(b, c, h, w)
        return x_out

class UNetTransformer(nn.Module):
    def __init__(self, in_channels=3, out_channels=1,
base_filters=32):
        super().__init__()
        self.enc1 = EncoderBlock(in_channels, base_filters,
pool=True)         # 32
        self.enc2 = EncoderBlock(base_filters, base_filters*2,
pool=True)      # 64
        self.enc3 = EncoderBlock(base_filters*2, base_filters*4,
pool=False)   # 128

        self.transformer = TransformerBlock(dim=base_filters*4,
num_heads=4)

        self.dec3 = DecoderBlock(in_ch=base_filters*4,
skip_ch=base_filters*2, out_ch=base_filters*2)
        self.dec2 = DecoderBlock(in_ch=base_filters*2,
skip_ch=base_filters, out_ch=base_filters)
        self.final = nn.Conv2d(base_filters, out_channels,
kernel_size=1)

    def forward(self, x):
        s1, p1 = self.enc1(x)
```

```
        s2, p2 = self.enc2(p1)
        s3, p3 = self.enc3(p2)
        t = self.transformer(s3)
        d3 = self.dec3(t, s2)
        d2 = self.dec2(d3, s1)
        out = torch.sigmoid(self.final(d2))
        return out
```

Лістинг коду модуля losses.py

```
import torch.nn as nn

def dice_coeff(pred, target, eps=1e-8):
    pred_flat = pred.contiguous().view(pred.size(0), -1)
    target_flat = target.contiguous().view(target.size(0), -1)
    intersection = (pred_flat * target_flat).sum(dim=1)
    union = pred_flat.sum(dim=1) + target_flat.sum(dim=1)
    dice = (2.0 * intersection + eps) / (union + eps)
    return dice.mean()

class BCEDiceLoss(nn.Module):
    def __init__(self, alpha=0.5):
        super().__init__()
        self.alpha = alpha
        self.bce = nn.BCELoss()

    def forward(self, pred, target):
        bce_loss = self.bce(pred, target)
        dice_loss = 1.0 - dice_coeff(pred, target)
        return self.alpha * bce_loss + (1 - self.alpha) *
dice_loss
```

Лістинг коду модуля utils.py

```
from PIL import Image
import numpy as np
import torch
from torchvision import transforms
from modules.config import CONFIG

preprocess = transforms.Compose([
    transforms.Resize(CONFIG["img_size"]),
    transforms.ToTensor(),
    transforms.Normalize([0.5, 0.5, 0.5], [0.5, 0.5, 0.5])
])

mask_preprocess = transforms.Compose([
    transforms.Resize(CONFIG["img_size"]),
    transforms.ToTensor()
])


def load_image_tensor(path):
    img = Image.open(path).convert('RGB')
    tensor = preprocess(img).unsqueeze(0)
    return tensor, img


def infer_image(model, path, device, threshold=0.5):
    model.eval()
    tensor, pil = load_image_tensor(path)
    tensor = tensor.to(device)
    with torch.no_grad():
        pred = model(tensor)
    pred_np = pred.squeeze().cpu().numpy()
    mask = (pred_np >= threshold).astype(np.uint8) * 255
    mask_img = Image.fromarray(mask).convert('L').resize(pil.size)
```

```
    mask_resized = mask_img.resize(pil.size)
    mask_rgba = mask_resized.convert('RGBA')


    overlay = pil.convert('RGBA')
    blended = Image.blend(overlay, mask_rgba, alpha=0.4)
    return pil, mask_img, blended


def evaluate_single_image(pred_mask, true_mask, threshold=0.5):

    """
    pred_bin = (pred_mask >= threshold).astype(np.uint8)
    true_bin = (true_mask > 0.5).astype(np.uint8)

    TP = np.logical_and(pred_bin == 1, true_bin == 1).sum()
    FP = np.logical_and(pred_bin == 1, true_bin == 0).sum()
    FN = np.logical_and(pred_bin == 0, true_bin == 1).sum()
    TN = np.logical_and(pred_bin == 0, true_bin == 0).sum()

    IoU = TP / (TP + FP + FN + 1e-8)
    Dice = 2 * TP / (2 * TP + FP + FN + 1e-8)
    Precision = TP / (TP + FP + 1e-8)
    Recall = TP / (TP + FN + 1e-8)
    Accuracy = (TP + TN) / (TP + TN + FP + FN + 1e-8)

    return {"IoU": IoU, "Dice": Dice, "Precision": Precision,
"Recall": Recall, "Accuracy": Accuracy}
```

Лістинг коду модуля train.py

```
import os
import time
import numpy as np
import matplotlib.pyplot as plt
import torch
from torch.utils.data import DataLoader
from modules.config import CONFIG
from modules.dataset import SegmentationDataset
from modules.model import UNetTransformer
from modules.losses import BCEDiceLoss
from torchvision import transforms

def train_one_epoch(model, dataloader, optimizer, criterion,
device):
    model.train()
    total_loss = 0.0
    for imgs, masks, _ in dataloader:
        imgs = imgs.to(device, non_blocking=True)
        masks = masks.to(device, non_blocking=True)
        preds = model(imgs)
        loss = criterion(preds, masks)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
        total_loss += loss.item() * imgs.size(0)
    return total_loss / len(dataloader.dataset)

def evaluate_model(model, dataloader, device, threshold=0.5):
    model.eval()
    metrics = {"IoU": [], "Dice": [], "Precision": [], "Recall":
[], "Accuracy": []}
    with torch.no_grad():
        for imgs, masks, _ in dataloader:
            imgs = imgs.to(device, non_blocking=True)
```

```
            masks = masks.to(device, non_blocking=True)
            preds = model(imgs)
            bin_preds = (preds >= threshold).float()
            for p, t in zip(bin_preds, masks):
                p = p.cpu().numpy().astype(np.uint8).squeeze()
                t = t.cpu().numpy().astype(np.uint8).squeeze()
                TP = np.logical_and(p == 1, t == 1).sum()
                FP = np.logical_and(p == 1, t == 0).sum()
                FN = np.logical_and(p == 0, t == 1).sum()
                TN = np.logical_and(p == 0, t == 0).sum()
                IoU = TP / (TP + FP + FN + 1e-8)
                Dice = 2 * TP / (2 * TP + FP + FN + 1e-8)
                Precision = TP / (TP + FP + 1e-8)
                Recall = TP / (TP + FN + 1e-8)
                Acc = (TP + TN) / (TP + TN + FP + FN + 1e-8)
                metrics["IoU"].append(IoU)
                metrics["Dice"].append(Dice)
                metrics["Precision"].append(Precision)
                metrics["Recall"].append(Recall)
                metrics["Accuracy"].append(Acc)
    avg = {k: float(np.mean(v)) if len(v) > 0 else 0.0 for k, v in
metrics.items()}
    return avg

def main_train_and_eval():
    device = CONFIG["device"]
    print("Device:", device)

    output_dir = "outputs"
    plots_dir = os.path.join(output_dir, "plots")
    metrics_dir = os.path.join(output_dir, "metrics")
    models_dir = os.path.join(output_dir, "models")  # нова папка
для моделей

    os.makedirs(output_dir, exist_ok=True)
```

```
    os.makedirs(plots_dir, exist_ok=True)
    os.makedirs(metrics_dir, exist_ok=True)
    os.makedirs(models_dir, exist_ok=True)

    model_name = input("Введіть ім'я для нової моделі (без
розширення .pth): ")
    model_save_path = os.path.join(models_dir, model_name +
".pth")
    print(f"Модель буде збережена у: {model_save_path}")

    data_root = CONFIG["dataset_root"]
    images_dir = os.path.join(data_root, "images")
    masks_dir = os.path.join(data_root, "masks")

    if not (os.path.isdir(images_dir) and
os.path.isdir(masks_dir)):
        print("Dataset not found. Для тренування потрібні папки
'images' і 'masks' в", data_root)
        return

    transform = transforms.Compose([
        transforms.Resize(CONFIG["img_size"]),
        transforms.ToTensor(),
        transforms.Normalize([0.5,0.5,0.5],[0.5,0.5,0.5])
    ])
    mask_transform = transforms.Compose([
        transforms.Resize(CONFIG["img_size"]),
        transforms.ToTensor()
    ])

    dataset = SegmentationDataset(images_dir, masks_dir,
transform=transform, mask_transform=mask_transform)
    n = len(dataset)
    print(f"Found {n} samples.")
    split = int(0.8 * n)
```

```
    train_ds, val_ds = torch.utils.data.random_split(dataset,
[split, n - split])
    train_loader = DataLoader(train_ds,
batch_size=CONFIG["batch_size"], shuffle=True, pin_memory=True)
    val_loader = DataLoader(val_ds,
batch_size=CONFIG["batch_size"], shuffle=False, pin_memory=True)

    model = UNetTransformer(in_channels=3, out_channels=1,
base_filters=32).to(device)
    criterion = BCEDiceLoss(alpha=0.5).to(device)
    optimizer = torch.optim.Adam(model.parameters(),
lr=CONFIG["lr"])

    train_losses = []
    val_ious = []
    val_dices = []
    val_accs = []

    best_val_iou = 0.0

    metrics_log_path = os.path.join(metrics_dir,
f"metrics_{model_name}.txt")
    with open(metrics_log_path, "w") as f:
        f.write("Epoch,TrainLoss,IoU,Dice,Accuracy,TimeSec\n")

    for epoch in range(1, CONFIG["num_epochs"] + 1):
        start = time.time()

        train_loss = train_one_epoch(model, train_loader,
optimizer, criterion, device)
        metrics = evaluate_model(model, val_loader, device,
threshold=CONFIG["threshold"])

        val_iou = metrics["IoU"]
        val_dice = metrics["Dice"]
```

```
        val_acc = metrics["Accuracy"]

        end = time.time()
        epoch_time = end - start

        train_losses.append(train_loss)
        val_ious.append(val_iou)
        val_dices.append(val_dice)
        val_accs.append(val_acc)

        print(f"[Epoch {epoch}/{CONFIG['num_epochs']}] "
              f"TrainLoss={train_loss:.4f} | IoU={val_iou:.4f} |
Dice={val_dice:.4f} | Acc={val_acc:.4f} "
              f"| Time={epoch_time:.1f}s")
        with open(metrics_log_path, "a") as f:

f.write(f"{epoch},{train_loss:.4f},{val_iou:.4f},{val_dice:.4f},{v
al_acc:.4f},{epoch_time:.2f}\n")

        if val_iou > best_val_iou:
            best_val_iou = val_iou
            torch.save(model.state_dict(), model_save_path)
            print(">>> Saved BEST model")

    print("\nTraining finished. Best IoU =", best_val_iou)

    epochs = range(1, CONFIG["num_epochs"] + 1)

    plt.figure(figsize=(7, 5))
    plt.plot(epochs, train_losses, label="Train Loss")
    plt.title("Training Loss")
    plt.xlabel("Epoch")
    plt.ylabel("Loss")
    plt.grid(True)
    plt.legend()
```

```
    plt.savefig(os.path.join(plots_dir, f"loss_{model_name}.png"),
dpi=150)
    plt.close()

    plt.figure(figsize=(7, 5))
    plt.plot(epochs, val_ious, label="Val IoU")
    plt.title("Validation IoU")
    plt.xlabel("Epoch")
    plt.ylabel("IoU")
    plt.grid(True)
    plt.legend()
    plt.savefig(os.path.join(plots_dir, f"iou_{model_name}.png"),
dpi=150)
    plt.close()

    plt.figure(figsize=(7, 5))
    plt.plot(epochs, val_dices, label="Val Dice")
    plt.title("Validation Dice")
    plt.xlabel("Epoch")
    plt.ylabel("Dice")
    plt.grid(True)
    plt.legend()
    plt.savefig(os.path.join(plots_dir, f"dice_{model_name}.png"),
dpi=150)
    plt.close()

    plt.figure(figsize=(7, 5))
    plt.plot(epochs, val_accs, label="Accuracy")
    plt.title("Validation Accuracy")
    plt.xlabel("Epoch")
    plt.ylabel("Accuracy")
    plt.grid(True)
    plt.legend()
    plt.savefig(os.path.join(plots_dir,
f"accuracy_{model_name}.png"), dpi=150)
```

```
    plt.close()

    print("\nГрафіки збережено у:", plots_dir)
    print("Метрики збережено у:", metrics_log_path)
    print("Модель збережено у:", model_save_path)
```

Лістинг коду модуля gui.py

```
import os
import tkinter as tk
from tkinter import filedialog, messagebox
from PIL import ImageTk, Image
import numpy as np
import torch

from modules.config import CONFIG
from modules.model import UNetTransformer
from modules.utils import infer_image, evaluate_single_image


class SegmentationApp:
    def __init__(self, model, device):
        self.model = model
        self.device = device
        self.root = tk.Tk()
        self.root.title("U-Net + Transformer — Сегментація")
        self.root.geometry("1150x720")
        self.root.configure(bg="#F5F5F5")

        frame = tk.Frame(self.root, bg="#F5F5F5")
        frame.pack(side=tk.TOP, pady=8)

        tk.Button(frame, text="Завантажити зображення",
command=self.load_image).grid(row=0, column=0, padx=8)
        tk.Button(frame, text="Запустити сегментацію",
```

```
command=self.segment_image).grid(row=0, column=1, padx=8)
        tk.Button(frame, text="Зберегти маску",
command=self.save_mask).grid(row=0, column=2, padx=8)
        tk.Button(frame, text="Вийти",
command=self.close).grid(row=0, column=3, padx=8)

        self.canvas_original = tk.Label(self.root, bg="#DDD")
        self.canvas_mask = tk.Label(self.root, bg="#DDD")
        self.canvas_result = tk.Label(self.root, bg="#DDD")
        self.canvas_original.place(x=20, y=60, width=360,
height=360)
        self.canvas_mask.place(x=395, y=60, width=360, height=360)
        self.canvas_result.place(x=770, y=60, width=360,
height=360)

        self.metrics_label = tk.Label(self.root, text="",
font=("Arial", 12), bg="#F5F5F5", justify="left")
        self.metrics_label.place(x=20, y=440)

        self.status = tk.Label(self.root, text="Готово до роботи",
bg="#F5F5F5")
        self.status.pack(side=tk.BOTTOM, pady=6)

        self.img_path = None
        self.mask_img = None
        self.true_mask_path = None

    def close(self):
        self.root.destroy()

    def load_image(self):
        path = filedialog.askopenfilename(filetypes=[("Файли
зображень", "*.jpg *.png *.jpeg *.bmp")])
        if not path:
            return
```

```
        self.img_path = path
        dataset_root = CONFIG["dataset_root"]
        mask_name = os.path.splitext(os.path.basename(path))[0] +
".png"
        self.true_mask_path = os.path.join(dataset_root, "masks",
mask_name)

        pil = Image.open(path).convert('RGB')
        pil_resized = pil.resize((360, 360))
        self.tk_original = ImageTk.PhotoImage(pil_resized)
        self.canvas_original.config(image=self.tk_original)
        self.status.config(text=f"Завантажено:
{os.path.basename(path)}")
        self.metrics_label.config(text="")

    def segment_image(self):
        if self.img_path is None:
            messagebox.showwarning("Увага", "Спочатку завантажте
зображення.")
            return

        self.status.config(text="Виконується сегментація...")
        self.root.update()

        pil, mask, blended = infer_image(self.model,
self.img_path, self.device, threshold=CONFIG["threshold"])

        mask_resized = mask.resize((360, 360))
        blended_resized = blended.resize((360, 360))

        self.tk_mask =
ImageTk.PhotoImage(mask_resized.convert('L'))
        self.tk_result = ImageTk.PhotoImage(blended_resized)

        self.canvas_mask.config(image=self.tk_mask)
```

```
        self.canvas_result.config(image=self.tk_result)

        self.mask_img = mask

        if self.true_mask_path and
os.path.exists(self.true_mask_path):
            true_mask =
Image.open(self.true_mask_path).convert('L')
            true_mask =
np.array(true_mask.resize(CONFIG["img_size"])) / 255.0
            pred_mask = np.array(mask.resize(CONFIG["img_size"]))
/ 255.0

            metrics = evaluate_single_image(pred_mask, true_mask)
            metrics_text = "\n".join([f"{k}: {v:.4f}" for k, v in
metrics.items()])
            self.metrics_label.config(text="Метрики
сегментації:\n" + metrics_text)
        else:
            self.metrics_label.config(text="Справжня маска не
знайдена для цього зображення.")

        self.status.config(text="Готово.")

    def save_mask(self):
        if self.mask_img is None:
            messagebox.showinfo("Помилка", "Немає маски для
збереження.")
            return

        path =
filedialog.asksaveasfilename(defaultextension=".png")
        if path:
            self.mask_img.save(path)
            messagebox.showinfo("Успіх", f"Маску збережено:
```

```
{path}")

    def run(self):
        self.root.mainloop()


def main_gui():
    device = CONFIG["device"]
    model = UNetTransformer(in_channels=3, out_channels=1,
base_filters=32).to(device)
    if os.path.exists(CONFIG["model_save_path"]):

model.load_state_dict(torch.load(CONFIG["model_save_path"],
map_location=device))
        print("Loaded model from", CONFIG["model_save_path"])
    else:
        print("Model weights not found — using random initialized
model.")
    app = SegmentationApp(model, device)
    app.run()
```