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

(повна назва факультету)

(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

(назва освітнього ступеня)

на тему: _____

Виконав(ла): студент(ка) _____ курсу, групи _____

спеціальності _____

(шифр і назва спеціальності)

(підпис) (прізвище та ініціали)

Керівник _____ _____
(підпис) (прізвище та ініціали)

Нормоконтроль _____ _____
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ _____
(підпис) (прізвище та ініціали)

Рецензент _____ _____
(підпис) (прізвище та ініціали)

Тернопіль
20___

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

Факультет _____
(повна назва факультету)

Кафедра _____
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ _____
(підпис) (прізвище та ініціали)
« » 20___ р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня _____
(назва освітнього ступеня)

за спеціальністю _____
(шифр і назва спеціальності)

студенту _____
(прізвище, ім'я, по батькові)

1. Тема роботи _____

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

Затверджені наказом ректора від «___» _____ 20___ року № _____

2. Термін подання студентом завершеної роботи _____

3. Вихідні дані до роботи _____

4. Зміст роботи (перелік питань, які потрібно розробити)

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка

Студент _____ _____
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота бакалавра на тему «Розробка веб-додатку для аналізу зображень флюоресцентної ангіографії ока з використанням технології YOLOv8» написана Задворним Олександром Миколайовичем, студентом Тернопільського національного технічного університету імені Івана Пулюя, Факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра програмної інженерії, група СП-41.

Відомості про обсяг: сторінок – 69, рисунків – 21, частин – 4, додатків – 2, посилань – 16, лістингів коду – 7.

Метою наукової роботи є створення веб-додатку для автоматизованого аналізу зображень флюоресцентної ангіографії ока з використанням технології YOLOv8. Для цього будуть застосовані методи глибокого навчання для обробки вхідних зображень та виявлення патологій ока. Бібліотека алгоритмів та програмне забезпечення дозволять створювати моделі для аналізу зображень та візуалізувати результати, а також забезпечать зручність використання та інтеграції.

У результаті роботи було створено систему "FluoroVision", яка аналізує зображення флюоресцентної ангіографії та автоматично виявляє до 25 типів уражень. Для досліджень було зібрано та перевірено набори даних з публічно доступних джерел. Розроблений додаток використовує модель YOLOv8, яка показала високі результати точності та повноти.

Веб-додаток продемонстрував високу ефективність у ранньому виявленні хронічних захворювань ока, що підтверджується виконаними тестами. Інструменти для розробки додатку включають мову програмування Python, бібліотеки OpenCV, PyTorch, Streamlit та інші, які забезпечують повну функціональність та зручність використання додатку.

Ключові слова: веб-додаток, флюоресцентна ангіографія ока, YOLO, глибоке навчання, аналіз зображень, Python, офтальмологія, Streamlit, автоматизоване тестування.

ANNOTATION

Bachelor's qualification work on the topic "Development of a Web Application for Analysing Fluorescein Angiography Images of the Eye Using YOLOv8 Technology" was written by Zadvornyi Oleksandr Mykolaiovych, a student of Ivan Puluj Ternopil National Technical University, Faculty of Computer and Information Systems and Software Engineering, Department of Software Engineering, group SP-41.

Information about the scope: pages - 69, figures - 21, sections - 4, appendices - 2, references – 16, code listings – 7.

The aim of this research is to develop a web application for automated analysis of fluorescein angiography images of the eye using YOLOv8 technology. This will involve the application of deep learning methods for processing input images and detecting eye pathologies. The library of algorithms and software will enable the creation of models for image analysis and visualization of results, ensuring ease of use and integration.

As a result of the work, the "FluoroVision" system was created, which analyses fluorescein angiography images and automatically detects up to 25 types of lesions. For the research, datasets from publicly available sources were collected and validated. The developed application uses the YOLOv8 model, which has demonstrated high precision and recall rates. The average precision exceeds 0.8, confirming its feasibility for image analysis.

The web application has shown high efficiency in the early detection of chronic eye diseases, as evidenced by conducted tests. The program provides real-time usability, reducing the time required for manual lesion detection and increasing diagnostic accuracy. The tools for developing the application include the Python programming language, OpenCV, PyTorch, Streamlit, and other libraries that ensure the full functionality and ease of use of the application.

Keywords: web application, fluorescein angiography of the eye, YOLO, deep learning, image analysis, Python, ophthalmology, Streamlit, automated testing

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

FFA (Fundus Fluorescein Angiography) – флюоресцентна ангіографія ока, метод діагностики, що використовується для візуалізації кровоносних судин у сітківці.

YOLO (You Only Look Once) – алгоритм глибокого навчання для швидкого та точного виявлення об’єктів на зображеннях.

API (Application Programming Interface) – набір інструментів та протоколів для створення програмного забезпечення.

CNN (Convolutional Neural Network) – тип нейронної мережі, що використовується для аналізу візуальних даних.

PT – файл, що містить збережені ваги моделі, використовується для завантаження навченої моделі.

IoU (Intersection over Union) – показник, що використовується для оцінки якості виявлення об’єктів на зображенні. Він вимірює співвідношення площі перетину передбаченого об’єкта та площі об’єкта в реальному світі до об’єднаної площі цих об’єктів.

RGB (Red, Green, Blue) – колірна модель, що використовується для кодування кольорів у зображеннях та дисплеях.

FP (False Positive) – випадок, коли модель неправильно виявляє об’єкт, який насправді відсутній.

FN (False Negative) – випадок, коли модель не виявляє об’єкт, який насправді присутній.

ЗМІСТ

ВСТУП

Флюоресцентна ангіографія ока (Fundus Fluorescein Angiography) є важливою інвазивною процедурою діагностики, яка використовується в офтальмології для оцінки ретинального і хоріоїдального кровотоку, а також інших компонентів, розташованих в задній частині ока. Ідентифікація уражень на зображеннях флюоресцентної ангіографії, які вказують на аномалії сітківки ока та судинної оболонки, відбувається з використанням ін’єкції спеціального флуюоресцеїнового барвника. Ця процедура є критично важливим кроком у діагностиці та лікуванні хронічних очних захворювань. Такі ураження часто вказують на ряд очних захворювань, включаючи діабетичну ретинопатію, макулодистрофію та оклюзію сітківки. Зазвичай дані зображення інтерпретуються вручну кваліфікованим лікарем офтальмологом. Через складну природу цих зображень їх аналіз часто виконується вручну, що займає багато часу та підлягає індивідуальному тлумаченню. Розвиток передових інформаційних технологій спонукає на розробку автоматизованих методів виявлення уражень, тим самим полегшуючи роботу спеціалістів у сфері офтальмології.

Метою даної роботи є розробка веб-додатку для автоматизованого виявлення уражень на зображеннях флюоресцентної ангіографії ока. Для досягнення цієї мети необхідно вирішити наступні задачі:

1) Провести аналіз існуючих методів автоматизованого виявлення уражень на зображеннях флюоресцентної ангіографії.
2) Розробити архітектуру веб-додатку для обробки зображень та виявлення уражень.
3) Провести тестування розробленого додатку на реальних даних та оцінити його ефективність.

Об’єктом дослідження є процес автоматизованого виявлення уражень на зображеннях флюоресцентної ангіографії ока. Предметом дослідження є методи і

засоби автоматизованої обробки зображень та виявлення відповідних уражень.

Актуальність цієї роботи полягає у необхідності створення автоматизованих методів виявлення уражень на зображеннях флюоресцентної ангіографії для підвищення ефективності діагностики та лікування хронічних очних захворювань. Впровадження таких технологій дозволить знизити навантаження на лікарів та покращити точність і швидкість постановки діагнозу, що є важливим для розвитку сучасної офтальмології.

У роботі використовувались такі методи дослідження:

- аналіз літературних джерел з метою виявлення існуючих підходів до автоматизації діагностики очних захворювань;
- розробка алгоритмів обробки зображень з використанням сучасних бібліотек машинного навчання;
- тестування та оцінка ефективності розробленого додатку на реальних даних.

Наукова новизна роботи полягає у розробці веб-додатку для автоматизованого виявлення поширених уражень на зображеннях флюоресцентної ангіографії ока. Практичне значення полягає у теоретичній можливості використання розробленого додатку в клінічній практиці для підвищення ефективності діагностики та лікування хронічних очних захворювань. Результати дослідження можуть бути впроваджені в офтальмологічних дослідженнях, що дозволить знизити навантаження на лікарів та покращити якість медичних послуг.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд флюоресцентної ангіографії ока

Флюоресцентна ангіографія ока є важливою діагностичною процедурою, що використовується в офтальмології для оцінки стану кровоносних судин сітківки та хоріоідеї. Ця методика дозволяє візуалізувати кровотік у задній частині ока та виявляти різноманітні патологічні зміни, що можуть бути пов’язані з різними очними захворюваннями [1]. Основний принцип методу полягає в введенні флюоресцеїнового барвника в венозну систему пацієнта, після чого проводиться серія знімків задньої частини ока за допомогою спеціальної камери. Барвник, циркулюючи по кровоносних судинах, дозволяє візуалізувати їх на зображеннях. Приклад застосування флюоресцентної ангіографії в медичному закладі наведено на рисунку 1.1. Проведення процедури складається з наступних етапів:

- Пацієнту вводиться флюоресцеїновий барвник внутрішньовенно.
- Після введення барвника проводиться серія фотографій ока з інтервалами в кілька секунд. Це дозволяє створити динамічну картину кровообігу в сітківці та хоріоідеї.
- Отримані зображення аналізуються для виявлення порушень, таких як витоки барвника (вказують на пошкодження судин), зони неперфузії (відсутність кровотоку) та новоутворення судин.

Рисунок 1.1 – Використання флюоресцентної ангіографії для лікування пацієнта

Флюоресцентна ангіографія знайшла широке застосування з метою діагностики діабетичної ретинопатії – захворювання очей, яке виникає внаслідок діабету. Хоча флюоресцентна ангіографія є високоефективним методом діагностики, її використання стикається з низкою викликів, таких як потреба в інвазивному втручанні та залежність від кваліфікації лікаря при інтерпретації зображень. Сучасні тенденції розвитку технологій включають автоматизацію процесів аналізу зображень за допомогою алгоритмів машинного навчання та штучного інтелекту, що може значно підвищити ефективність та точність діагностики.

1.2 Характеристика основних типів уражень на зображеннях

Одним з найпоширеніших захворювань сітківки ока є діабетична ретинопатія, яка виникає внаслідок пошкодження існуючих та утворення нових патологічних кровоносних судин, що можуть призвести до повної або часткової втрати зору. Захворювання може прогресувати до діабетичного макулярного набряку – запущеної та небезпечної для зору стадії, при якій пошкоджується центральна частина світлочутливого шару ока. При діагностиці цукрового діабету 2-го типу до 21% пацієнтів мають ретинопатію, а в більшості розвивається певний ступінь ретинопатії впродовж певного часу [2].

Міжнародна асоціація із запобігання сліпоти оцінює, що в 2015 році 145 мільйонів людей мали ту чи іншу форму діабетичної ретинопатії, а 45 мільйонів постраждали від неї. У зв’язку з зростанням захворюваності на діабет, прогнозується поширення хвороби діабетичної ретинопатії, підкреслюючи важливість її раннього виявлення та подальшого лікування [3].

Флюоресцентна ангіографія дозволяє детально візуалізувати кровоносні судини сітківки, що є критично важливим для діагностики різноманітних патологій. На отриманих зображеннях можна ідентифікувати кілька основних

типів уражень, кожен з яких має свої характерні ознаки. Приклад чорно-білого знімку васкуляризації ока, отриманого в результаті флюоресцентної ангіографії зображено на рисунку 1.2. Загалом можна виділити наступні основні типи уражень, які зазвичай легко виявляються за допомогою технології флюоресцентної ангіографії:

- Мікроаневризми – маленькі випинання на стінках капілярів сітківки, виглядають як невеликі яскраві точки. Мікроаневризми часто є першою ознакою діабетичної ретинопатії та можуть призводити до витоків крові.
- Неперфузійні області – зони, де відсутній кровообіг через обструкцію або оклюзію судин, які виявляються як темні ділянки, що не забарвлюються флуоресцеїном. Неперфузійні області можуть призводити до подразнення сітківки, стимулюючи розвиток новоутворених судин.
- Неоваскуляризація – утворення нових, патологічних кровоносних судин, що виглядають як хаотично розташовані, тонкі та звивисті судини, які забарвлюються на пізніх фазах ангіограми. Неоваскуляризація є ознакою важких захворювань, таких як діабетична ретинопатія та вікова макулодистрофія. Може призводити до крововиливів та рубців.
- Макулярний набряк – накопичення рідини в центральній частині сітківки (макулі), що виглядають як область потовщення та зниженого контрасту. Макулярний набряк є однією з основних причин зниження зору при діабетичній ретинопатії та інших захворюваннях.

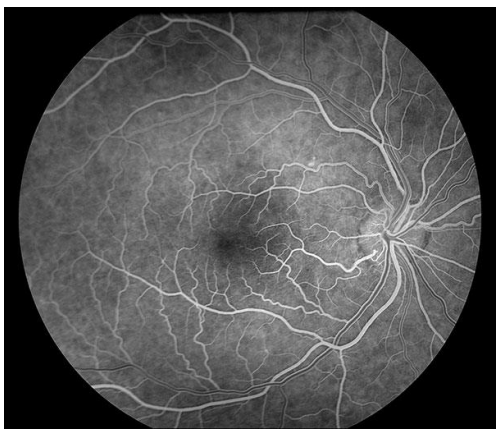

Рисунок 1.2 – Знімок ока при виконанні флюоресцентної ангіографії

Загальні методи діагностики діабетичної ретинопатії включають в себе фотозйомку очного дна сітківки, оптичну когерентну томографію та флюоресцентну ангіографію ока. Оскільки дане захворювання в основному впливає на внутрішні судини сітківки, флюоресцентна ангіографія є безцінним методом для виявлення судинних витоків та інших мікроаневризмів, які інші методи діагностики не в змозі ідентифікувати. Автоматизований аналіз на наявність діабетичної ретинопатії з використанням штучного інтелекту на основі зроблених зображень сітківки ока може бути не тільки точним, але й економічно ефективним [4,5].

Розуміння характеристик основних типів уражень на зображеннях флюоресцентної ангіографії є критично важливим для діагностики та лікування різних очних захворювань. Кожен тип ураження має свої специфічні візуальні ознаки, які можуть бути автоматизовано ідентифіковані за допомогою сучасних технологій машинного навчання, таких як YOLO. Це значно підвищує ефективність і точність діагностичного процесу, зменшуючи навантаження на лікарів і покращуючи результати лікування пацієнтів.

1.3 Методи автоматизованої діагностики уражень ока

Виявлення уражень сітківки ока при діабетичній ретинопатії за допомогою зображень очного дна та флюоресцентної ангіографії, включаючи неоваскуляризацію та інші мікроаневризми, є важливим для планування точних місць для подальшого лазерного лікування. На сьогодні вже існують різноманітні дослідження, направлені на автоматизування різних класифікацій патологій сітківки на зображеннях очного дна для скринінгу діабетичної ретинопатії, але небагато з них задокументували автоматизований метод виявлення уражень ока на зображеннях флюоресцентної ангіографії. Виявлення уражень вручну є досить дорогою та затратною в часі задачею, тому автоматизація аналізу зображень та

виявлення патологій сітківки може допомогти спеціалістам у сфері офтальмології у прийнятті рішень щодо лікування та направлення пацієнтів.

Автоматизована діагностика уражень ока є важливим напрямом в сучасній офтальмології, який дозволяє підвищити точність і ефективність виявлення патологій. З розвитком технологій машинного навчання та обробки зображень з'явилися різні методи автоматизованої діагностики, які використовуються для аналізу зображень флюоресцентної ангіографії. Розглянемо основні методи, що застосовуються в цій галузі.

Класичні методи обробки включають використання алгоритмів фільтрації, сегментації та виявлення країв для аналізу зображень. Наприклад, застосування фільтрів для зменшення шуму та покращення якості зображень, методи сегментації, такі як порогова сегментація, вододіл та активні контури, використовуються для виділення кровоносних судин та уражень. Також широко застосовуються алгоритми виявлення країв з метою ідентифікації контурів судин. Переваги класичних методів полягають у простоті реалізації та швидкості обробки, однак вони мають обмежену точність і залежать від якості вхідних зображень.

Методи машинного навчання використовуються для класифікації та виявлення уражень на основі аналізу зображень. Наприклад, підтримка векторних машин застосовується для класифікації різних типів уражень на основі певних ознак, а випадкові ліси, що складаються з багатьох рішучих дерев, використовуються для класифікації та регресії. Основними перевагами методів машинного навчання є можливість виявлення складних патернів, проте вони вимагають великого обсягу навчальних даних для досягнення високої точності.

Глибоке навчання використовує нейронні мережі з багатьма шарами для автоматичного виявлення уражень. Згорткові нейронні мережі застосовуються для автоматичного виділення ознак та класифікації зображень, моделі YOLO (You Only Look Once) інтегруються в реальному часі для виявлення уражень. Основними перевагами є висока точність та можливість автоматичного виявлення складних ознак, однак ці методи мають високі вимоги до обчислювальних ресурсів та великого обсягу навчальних даних.

Комбіновані підходи використовують переваги різних методів для покращення точності та надійності автоматизованої діагностики. Наприклад, гібридні моделі поєднують класичні методи обробки зображень з методами машинного та глибокого навчання, а інтеграція даних з різних джерел, таких як флюоресцентної ангіографії та оптичної когерентної томографії, покращує точність діагностики. Переваги комбінованих підходів включають підвищену точність та гнучкість, проте їх реалізація може бути складною та вимагати інтеграції різних систем.

Методи автоматизованої діагностики уражень ока значно підвищують ефективність і точність діагностичного процесу. Від класичних методів обробки зображень до сучасних підходів глибокого навчання, кожен метод має свої переваги та недоліки. Використання глибоких нейронних мереж, зокрема моделей YOLO, є перспективним напрямом, що дозволяє автоматизувати виявлення уражень на зображеннях з високою точністю. Комбіновані підходи, що інтегрують різні методи, можуть забезпечити ще кращі результати, сприяючи розвитку автоматизованої діагностики та покращенню клінічних стратегій лікування очних захворювань.

1.4 Сучасні підходи до розпізнавання зображень у медицині

Сучасні підходи до розпізнавання зображень у медицині базуються на використанні передових технологій машинного навчання та глибокого навчання, які значно підвищують точність і ефективність діагностики. Завдяки розвитку цих технологій, аналіз медичних зображень стає більш автоматизованим, швидким та точним, що сприяє покращенню якості медичної допомоги.

Застосування моделей глибокого навчання для виявлення уражень у медичній візуалізації здобуло значну увагу в останні роки. Моделі глибокого навчання продемонстрували значний потенціал у виявленні уражень у різних

медичних методах візуалізації, включаючи ураження шкіри, рак легень та рак молочної залози. Ці дослідження зафіксували виняткові показники продуктивності з точки зору точності та повноти аналізу.

Глибоке навчання є найбільш перспективним напрямом у розпізнаванні медичних зображень. Згорткові нейронні мережі демонструють високу точність у завданнях класифікації, сегментації та виявлення об’єктів на медичних зображеннях, автоматично екстрагуючи важливі ознаки, що дозволяє виявляти складні патерни, які можуть бути недоступними для традиційних методів. Наприклад, моделі ResNet використовуються для діагностики захворювань легень за допомогою рентгенівських знімків, а U-Net ефективно сегментує зображення для виявлення патологій на знімках магнітно-резонансної томографії. Приклад використання глибокого навчання в медичній сфері зображено на рисунку 1.3.

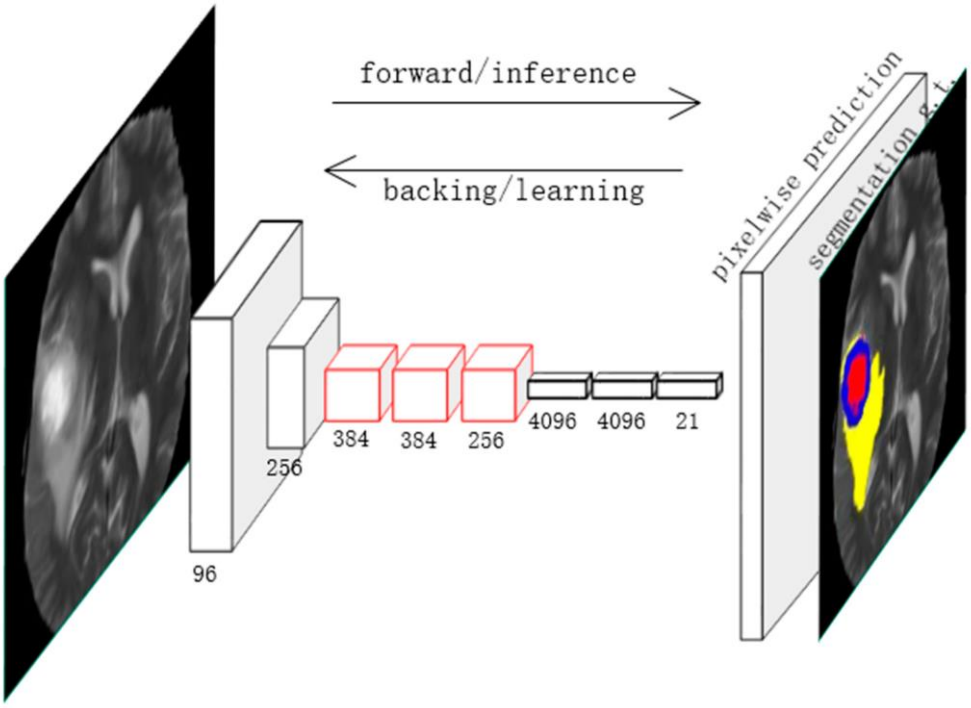

Рисунок 1.3 – Використання нейронних мереж в медицині

Хоча флюоресцентна ангіографія залишається стандартною технікою для діагностики та моніторингу офтальмологічних захворювань, новіші технології, такі як оптична когерентна томографія, поступово набувають все більше популярності

у медичній спільноті. Нові дослідження підкреслюють важливу роль флюоресцентної ангіографії у діагностиці та моніторингу різних нейро-офтальмологічних розладів, таких як вікова макулярна дегенерація, діабетична ретинопатія та судинні захворювання сітківки. Крім того, поєднання цієї технології та оптичної томографії може надати більш повне уявлення про судинну систему сітківки та підлеглу тканину сітківки, що сприяє кращій діагностиці та лікуванню цих захворювань [6].

Сучасні підходи до розпізнавання зображень у медицині відкривають нові можливості для автоматизації та покращення діагностичних процесів. Використання глибокого навчання, підкріплювального навчання, гібридних моделей дозволяє підвищити точність, ефективність і надійність медичної діагностики, що в кінцевому результаті сприяє покращенню якості медичної допомоги та здоров’я пацієнтів.

1.5 Технологія YOLO: еволюція та характеристики версії YOLOv8

Технологія “You Only Look Once” (YOLO) є однією з провідних методик для швидкого та ефективного розпізнавання об’єктів на зображеннях. Запропонована у 2015 році Джозефом Редмоном та іншими дослідниками, YOLO стала революційною завдяки своєму підходу до детекції об’єктів в реальному часі. На відміну від традиційних алгоритмів розпізнавання, які використовують регіональні пропозиції або багатокрокові процеси для виявлення об’єктів, YOLO застосовує єдиний нейронний мережевий підхід. Це дозволяє системі обробляти цілі зображення за один прохід, значно скорочуючи час обробки та забезпечуючи високу швидкість роботи [7].

Модель YOLO представляє собою алгоритм реального часу для виявлення об’єктів, спеціально розроблений для ідентифікації та класифікації об’єктів на зображеннях та відеопотоках. Основна методологія, втілена в моделях YOLO

(YOLOv3, YOLOv5, YOLOv5 (evolved), YOLOv7 і YOLOv8), може бути підсумована наступним чином:

1) Вхідним даними для алгоритму є зображення або кадр відео.
2) Після цього алгоритм ділить зображення на сітку з комірок. Для кожної з цих комірок він прогнозує наявність і розташування об’єктів у ній.
3) Далі алгоритм передбачає клас кожного об’єкта (наприклад, автомобіль, людина тощо) і присвоює кожному з цих прогнозів оцінку впевненості.
4) Алгоритм застосовує видалення немаксимальних значень для усунення накладених обмежувальних рамок, тим самим вибираючи прогнози з найвищою впевненістю.
5) В кінцевому результаті алгоритм виводить список обмежувальних рамок, кожна з яких має присвоєну мітку класу та оцінку впевненості.

Ключовою особливістю YOLO є її здатність до одночасної локалізації та класифікації об`єктів на зображенні. Модель ділить зображення на сітку і передбачає межі об’єктів та їх клас у межах кожної комірки. Завдяки цьому підходу, YOLO забезпечує не лише високу швидкість, але й високу точність розпізнавання об’єктів, роблячи її ефективною навіть для завдань реального часу, таких як відеоспостереження, автономне водіння та, звісно, медична діагностика.

З моменту свого впровадження, YOLO зазнала кількох значних покращень та еволюційних змін. Кожна нова версія моделі включала вдосконалення, спрямовані на підвищення точності та продуктивності, що робило технологію ще більш потужною та адаптивною до різних завдань. Зокрема, сучасні версії YOLO включають вдосконалені архітектурні елементи, методи навчання та оптимізації, що дозволяють моделі ефективно працювати з великими обсягами даних та складними зображеннями.

Остання модель YOLOv8 будується на основі своїх попередників, покращуючи точність, швидкість і простоту. Архітектура даної моделі базується на згортковій нейронній мережі (CNN), яка включає в себе кілька вхідних, вихідних та прихованих шарів. Повна архітектура моделі зображена на рисунку 1.4.

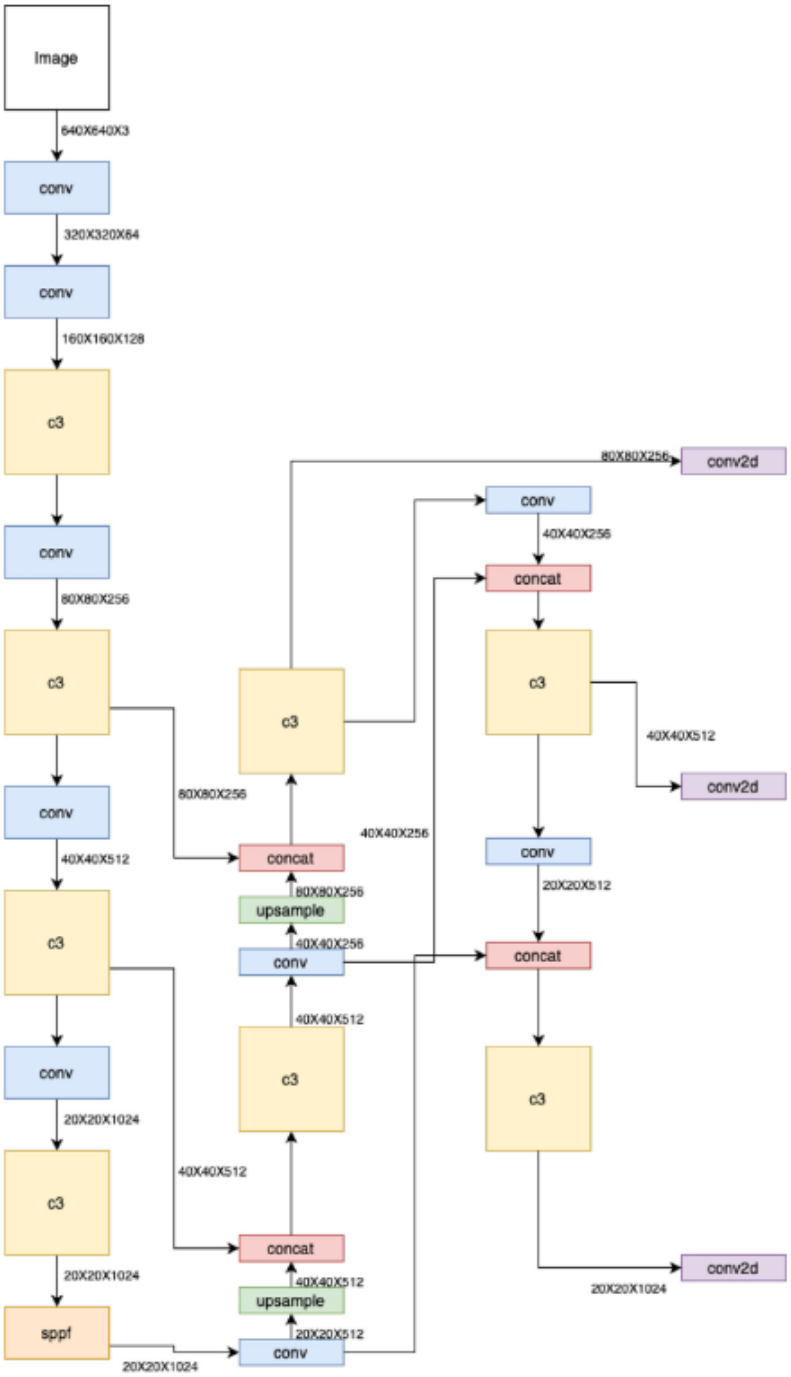

Рисунок 1.4 – Архітектура YOLOv8

В останніх версіях YOLO були додані такі вдосконалення, як використання більш глибоких і широких мереж, а також застосування новітніх методів регуляризації і нормалізації, що дозволило значно покращити продуктивність моделі. Версія YOLOv8, зокрема, включає передові техніки глибинного навчання, що забезпечують ще кращу точність і швидкість. Принцип роботи моделі подано на рисунку 1.5.

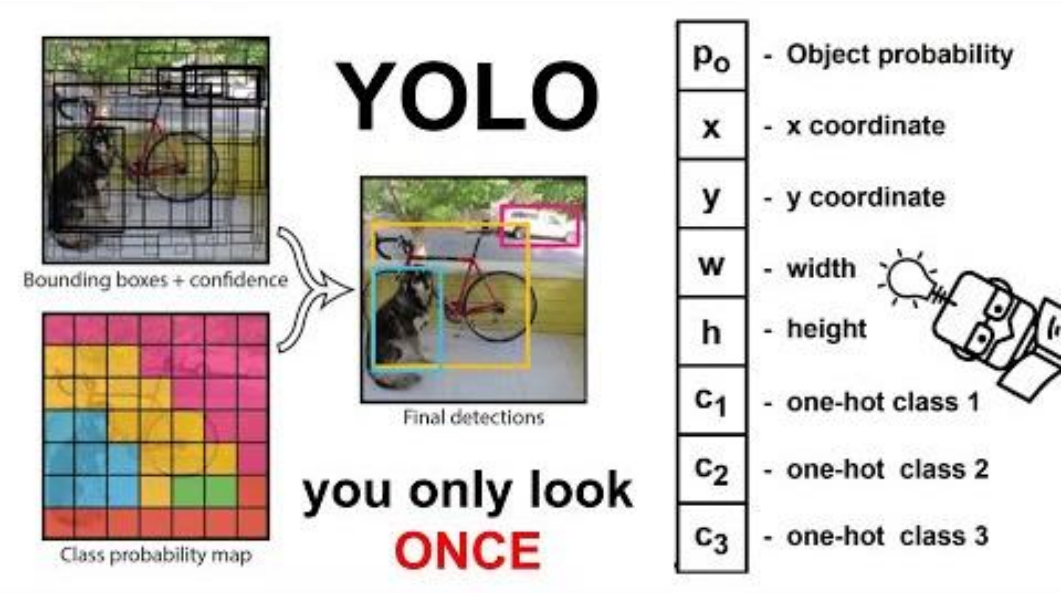

Рисунок 1.5 – Принцип роботи моделі YOLO

Модель YOLOv8 вводить кілька функцій і покращень порівняно зі своїми попередниками, а саме:

- Модель YOLOv8 використовує за основу принцип “EfficientNet” як нейронну мережу для визначення ознак, яка оптимізована для ефективного та дієвого вилучення ознак із зображень.
- YOLOv8 застосовує просторову піраміду (spatial pyramid) для поділу вхідного зображення на кілька областей різних масштабів, що дозволяє виявляти об’єкти різних розмірів.
- YOLOv8 включає навчання для підвищення стійкості та здатності моделі до розвитку, що сприяє покращенню продуктивності в широкому спектрі завдань виявлення об’єктів.
- Модель використовує мозаїчну аугментацію даних для підвищення стійкості моделі до різних трансформацій і збурень зображень.

Таким чином, YOLO продовжує залишатися однією з найпопулярніших та найбільш використовуваних технологій для розпізнавання об’єктів у різних галузях, підтверджуючи свою цінність та ефективність у практичному застосуванні.

2 ПРОЕКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

2.1 Вимоги до програмного забезпечення

При розробці системи для аналізу зображень флюоресцентної ангіографії ока, важливо визначити вимоги, які забезпечать успішну реалізацію та функціонування системи. Їх можна поділити на функціональні та нефункціональні.

Функціональні вимоги описують конкретні дії, які програмне забезпечення повинно виконувати для задоволення потреб користувачів. Вони визначають, як система реагує на певні входи та як вона повинна поводитися в різних ситуаціях. До функціональних вимог веб-додатку “FluoroVision” можна віднести наступні:

- Завантаження та обробка зображень – система повинна підтримувати завантаження зображень флюоресцентної ангіографії різних форматів (наприклад, JPEG, PNG, TIFF). Після завантаження зображення повинно бути автоматично підготовлено для аналізу.
- Виявлення уражень – система повинна автоматично визначати та класифікувати різні типи уражень на отриманих зображеннях з використанням моделі YOLO.
- Відображення результатів – після аналізу зображення, результати повинні бути відображені у вигляді зображень з накладеними прямокутниками, що позначають виявлені ураження, та відповідними назвами класів.
- Налаштування параметрів моделі – користувач повинен мати можливість налаштовувати параметри моделі, такі як пороги впевненості та значення перетину об’єктів (IoU).
- Збереження та експорт результатів – система повинна дозволяти користувачам зберігати результати аналізу.
- Інтерактивний інтерфейс – система повинна мати зручний та інтуїтивно зрозумілий користувацький інтерфейс, який дозволяв би легко завантажувати зображення, запускати аналіз, переглядати результати та налаштовувати параметри.

Нефункціональні вимоги визначають характеристики та обмеження системи, такі як її продуктивність, надійність, безпека та масштабованість. Вони описують, як система повинна працювати, а не що вона повинна робити. Для розроблюваної системи можна виділити наступні нефункціональні вимоги:

- Продуктивність – система повинна забезпечувати швидкий аналіз зображень, щоб результати були доступні у реальному часі або близькому до нього.
- Надійність – програмне забезпечення повинно бути стійким до помилок і забезпечувати коректну роботу навіть при некоректних вхідних даних.
- Масштабованість – система повинна підтримувати обробку великої кількості зображень без значного зниження продуктивності.
- Безпека – система повинна забезпечувати захист даних користувачів та гарантувати конфіденційність медичної інформації.
- Сумісність – програмне забезпечення повинно бути сумісним з різними операційними системами (Windows, MacOS, Linux) та підтримувати різні браузери (Chrome, Firefox, Safari).

Визначені вимоги забезпечать створення ефективного, надійного та зручного програмного забезпечення для автоматизованого аналізу зображень флюоресцентної ангіографії ока, що сприятиме покращенню діагностики та лікування очних захворювань.

2.2 Аналіз акторів та варіантів використання

Головним актором, що взаємодіє із системою, є користувач. Користувачем може бути офтальмолог або інший медичний спеціаліст, який має необхідні знання для інтерпретації результатів аналізу зображень флюоресцентної ангіографії ока (FFA). Користувач є центральною фігурою у взаємодії з системою, його дії спрямовані на забезпечення точного та ефективного аналізу зображень FFA. Цей

процес значно полегшується за допомогою розробленого програмного забезпечення, що дозволяє автоматизувати рутинні завдання та підвищити ефективність діагностики.

Варіанти використання є одним з інструментів для збору функціональних вимог до системи. Вони визначають набір орієнтованих на ціль взаємодій між зовнішніми акторами (сторони, що знаходяться поза системою і взаємодіють з нею) та системою. Варіант використання ініціюється користувачем з певною метою і завершується успішно, коли ця мета досягається. Сама система розглядається як “чорний ящик”, а взаємодії з нею сприймаються як зовнішні по відношенню до системи, що дозволяє варіанту використання фіксувати, хто і що робить з системою та з якою метою, не розглядаючи внутрішні механізми системи [8].

Варіанти використання (use cases) описують сценарії взаємодії користувача з системою для досягнення конкретної мети. Для даної системи можна виділити наступні варіанти використання:

- Завантаження зображень FFA – користувач завантажує зображення флюоресцентної ангіографії ока до системи для подальшої обробки та аналізу.
- Аналіз зображень – користувач запускає процес автоматизованого аналізу зображень для ідентифікації та класифікації уражень.
- Налаштування параметрів аналізу – користувач налаштовує параметри обробки зображень, такі як моделі YOLO, порогові значення та інші налаштування, щоб адаптувати систему під свої потреби.
- Отримання результатів аналізу – після завершення аналізу користувач переглядає результати, які включають визначені ураження та їх локалізацію на зображеннях. Ці результати допомагають у діагностиці та плануванні лікування.
- Збереження та експорт результатів – користувач зберігає результати аналізу або експортує їх для подальшого використання чи обміну з іншими спеціалістами.

На основі щойно описаних варіантів використання, створимо відповідну діаграму, яка б візуально представила користувача та його взаємодію з системою. Це допоможе краще зрозуміти функціональні можливості програмного забезпечення та його структуру. Створена діаграма варіантів використання зображена на рисунку 2.1.

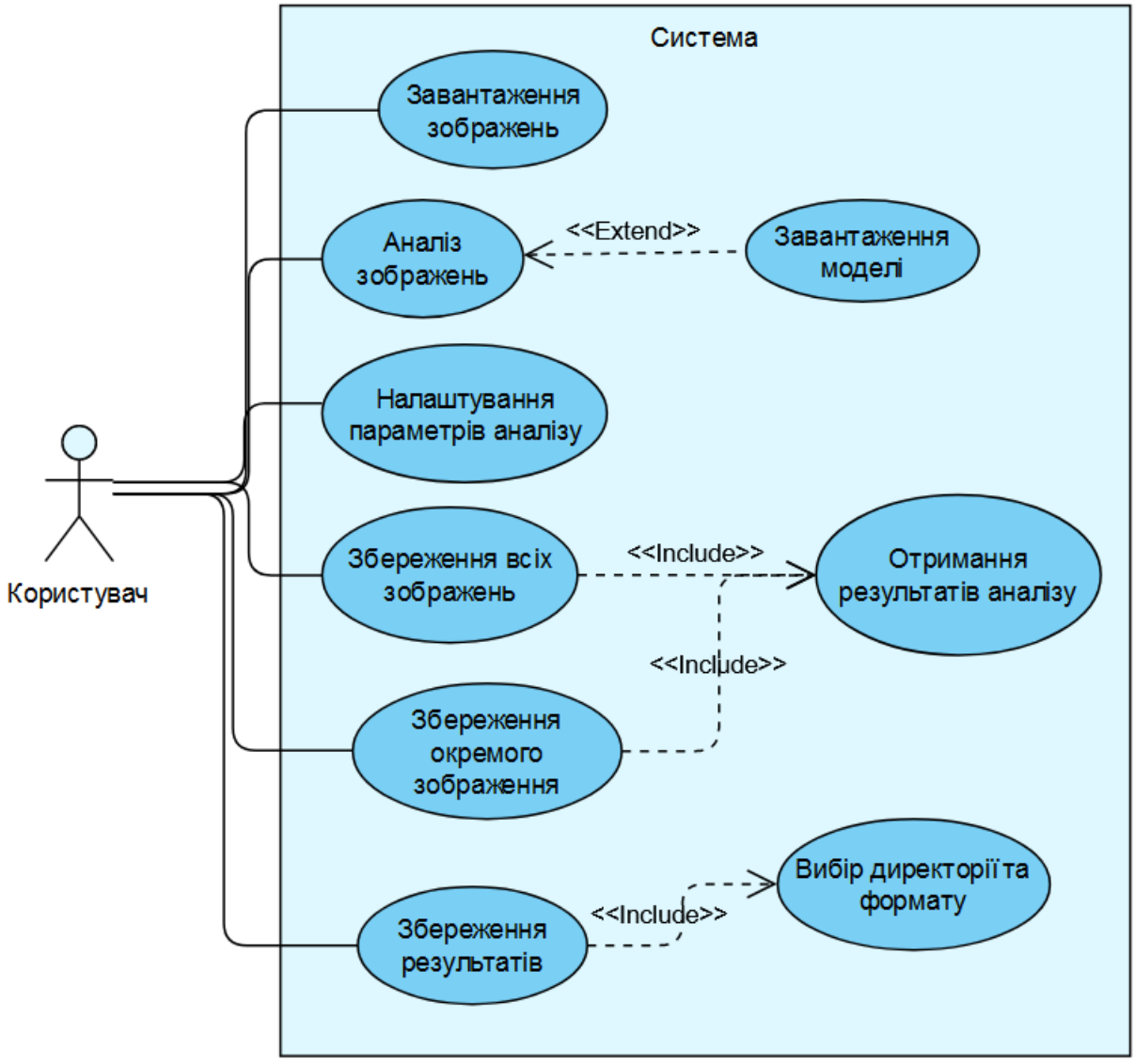

Рисунок 2.1 – Діаграма варіантів використання веб-додатку “FluoroVision”

Дана діаграма ілюструє всі можливі дії користувача в системі, забезпечуючи зрозуміле представлення функціональних можливостей та їх реалізації.

2.3 Архітектура системи

Архітектура системи є важливим аспектом розробки програмного забезпечення, оскільки вона визначає структуру, компоненти та їх взаємодію для досягнення функціональних та нефункціональних вимог. Архітектура розробленої системи для аналізу зображень флюоресцентної ангіографії ока складається з декількох ключових компонентів, кожен з яких виконує специфічні завдання для забезпечення коректної роботи всього програмного забезпечення. Застосунок загалом складається з трьох основних модулів, як показано на рисунку 2.2.

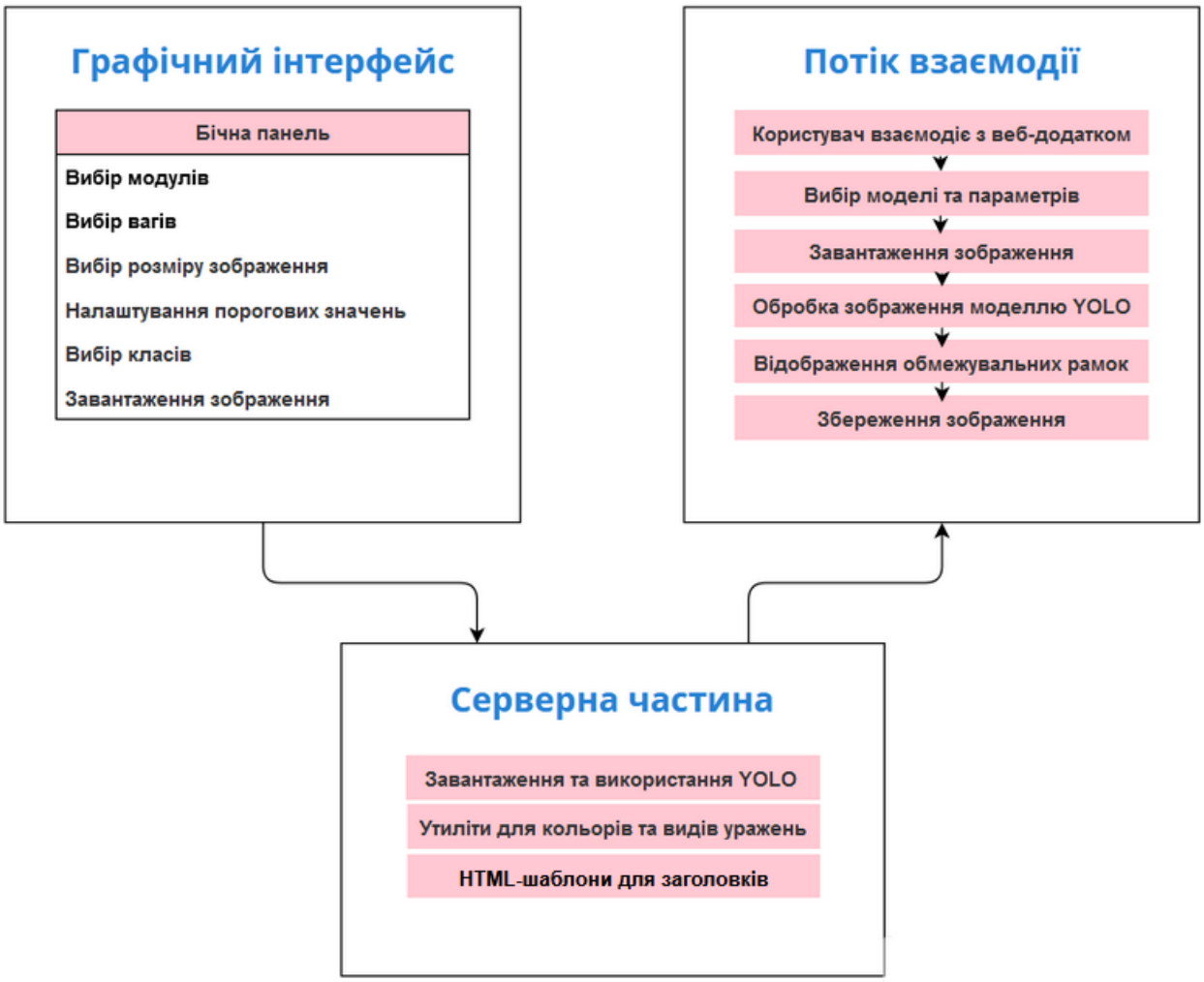

Рисунок 2.2 – Загальна архітектура веб-додатку

Модуль графічного інтерфейсу, використовуючи бібліотеку “Streamlit” на мові Python, забезпечує легкий та швидкий спосіб створення інтерактивних веб-додатків. Користувачі даного веб-застосунку можуть налаштовувати різноманітні параметри, такі як модель YOLO, відповідні ваги та розміри, пороги та класи тощо, щоб врахувати різні випадки використання та налаштувати виявлення об’єктів. Застосування бібліотеки “Streamlit” дозволяє завантажувати зображення для обробки моделлю YOLO, генерувати прогнози та відображати результати з обмежувальними рамками для візуального аналізу. Кнопка “Завантажити” дозволяє користувачам зберігати оброблені зображення з обмежувальними рамками для подальшого опрацювання.

Модуль серверної частини обробляє основні функції, завантажуючи моделі YOLO, вказані користувачем, з обраними вагами для експериментування. Він включає допоміжні функції для керування кольорами та генерації візуальної складової, забезпечуючи обмежувальні рамки та чітке представлення класів виявлених захворювань. Додаток також використовує спеціальний конфігураційний файл, що відображає номери уражень на їхні назви, допомагаючи користувачеві зрозуміти виявлені ураження та їхній тип. Поєднуючи простоту використання бібліотеки “Streamlit” з розробленою функціональністю серверної частини, веб-застосунок надає зручний та потужний інструментарій для виявлення уражень на завантажених зображеннях.

2.4 Вибір інструментів та технологій

Розумний вибір технологій та фреймворків є важливим етапом в кожному проєкті, оскільки від нього залежить ефективність, продуктивність та надійність розробленої системи. Правильно обрані інструменти дозволяють оптимізувати процес розробки, зменшити час на реалізацію функціоналу та підвищити якість кінцевого продукту. Крім того, відповідні технології забезпечують можливість

масштабування системи, її легку інтеграцію з іншими програмними продуктами та адаптацію до вимог користувачів, що вкрай важливо для успішної реалізації проекту. При розробці програмної системи для аналізу зображень флюоресцентної ангіографії ока були обрані наступні інструменти та технології з метою забезпечення ефективності, точності та надійності системи.

Для реалізації функціональності та забезпечення високої продуктивності програмного забезпечення були обрані наступні фреймворки та бібліотеки:

- YOLO (You Only Look Once) – це один з найбільш ефективних та швидких алгоритмів для визначення об’єктів на зображеннях у реальному часі. Версія YOLOv8 забезпечує покращену точність та швидкість порівняно з попередніми версіями, що є критично важливим для аналізу медичних зображень.
- Streamlit – бібліотека, що дозволяє швидко створювати інтерактивні веб-додатки для аналізу та візуалізації даних. Завдяки своїй простоті у використанні та широким можливостям інтеграції з іншими бібліотеками, Streamlit є оптимальним вибором для розробки користувацького інтерфейсу.
- OpenCV – потужна бібліотека для обробки зображень, яка включає інструменти для зчитування, маніпуляції та візуалізації зображень. Вона широко використовується в комп’ютерному зорі та машинному навчанні.
- NumPy – інструмент для забезпечення ефективних операцій з багатовимірними масивами та матрицями, що є основою для обробки даних у Python.
- PyTorch – бібліотека для побудови та тренування нейронних мереж. Це один з найбільш популярних фреймворків для машинного навчання завдяки своїй гнучкості та зручності у використанні.
- Matplotlib – бібліотека для візуалізації даних та результатів аналізу. Вона дозволяє створювати різноманітні графіки та діаграми для полегшення інтерпретації даних.

Для розробки даної програмної системи була обрана мова програмування Python. Python є однією з найбільш популярних мов програмування в галузі наукових досліджень, машинного навчання та обробки зображень. Основні причини вибору Python:

- Python має великий набір бібліотек, таких як OpenCV, NumPy, PyTorch та інші, що полегшують обробку зображень та реалізацію алгоритмів машинного навчання.
- Мова Python відома своєю простотою у синтаксисі та читабельністю коду, що дозволяє швидко розробляти та підтримувати програмне забезпечення.
- Широка спільнота розробників забезпечує постійну підтримку, оновлення та обмін досвідом, що важливо для вирішення складних задач.

Обрані інструменти та технології забезпечують високу продуктивність, надійність та гнучкість розробленого програмного забезпечення, що дозволяє ефективно вирішувати поставлені задачі з аналізу зображень флюоресцентної ангіографії ока.

2.5 Інтерфейс користувача

Інтерфейс користувача є ключовим елементом програмного забезпечення, що забезпечує взаємодію між користувачем та системою. Він має бути інтуїтивно зрозумілим, зручним для використання та надавати користувачеві всі необхідні інструменти для ефективного виконання завдань. У розроблюваній системі інтерфейс влаштований таким чином, щоб медичні спеціалісти могли легко завантажувати зображення флюоресцентної ангіографії, налаштовувати параметри аналізу та отримувати результати з мінімальними зусиллями.

Дизайн інтерфейсу розроблений з урахуванням потреб кінцевих користувачів – офтальмологів та медичних спеціалістів. Основний акцент зроблено на простоті та зручності використання, що дозволяє швидко та ефективно

виконувати всі необхідні дії. Інтерфейс включає в себе бокову панель для налаштування параметрів аналізу, таких як вибір моделі, ваги, розміру зображення, порогових значень та класів уражень. Крім того, є можливість завантаження зображень та кнопка для завантаження отриманих результатів аналізу. Всі елементи інтерфейсу розташовані логічно та зручно для користувача, що забезпечує швидкий доступ до всіх функцій.

Функціональні можливості інтерфейсу користувача включають в себе:

- Завантаження зображень флюоресцентної ангіографії – користувач може завантажити зображення FFA для подальшого аналізу.
- Налаштування параметрів аналізу – Користувач може вибрати модель YOLO, налаштувати ваги, розмір зображення, порогові значення та класи уражень відповідно до своїх потреб.
- Запуск аналізу – після налаштування всіх параметрів, користувач може запустити процес автоматизованого аналізу зображень для ідентифікації та класифікації уражень.
- Відображення результатів – користувач може переглянути результати аналізу, включаючи визначені ураження та їх локалізацію на зображеннях.
- Збереження та експорт результатів – користувач може зберегти результати аналізу або експортувати їх для подальшого використання чи обміну з іншими спеціалістами.

Ці функціональні можливості дозволяють користувачеві ефективно взаємодіяти з системою та отримувати необхідні результати для діагностики та лікування хронічних очних захворювань.

3 РОЗРОБКА ВЕБ-ДОДАТКУ

3.1 Підготовка та обробка даних

Підготовка та обробка даних є одним з найважливіших етапів розробки системи, в якій використовується штучний інтелект, особливо коли мова йде про роботу з медичними зображеннями. У даному підрозділі розглядаються основні кроки, необхідні для забезпечення якісної підготовки даних для аналізу зображень флюоресцентної ангіографії ока.

Першим кроком був збір даних, який включав в себе отримання великого набору зображень флюоресцентної ангіографії з різних джерел. Ці дані можуть бути зібрані з клінічних баз даних, наукових досліджень або надані медичними закладами. Важливо було забезпечити різноманітність даних, щоб охопити різні типи уражень і забезпечити більш точне навчання моделей.

Набір даних FFA-IR (FFA Images and Reports) є великою збіркою медичних даних, що складається близько з мільйона зображень флюоресцентної ангіографії ока, які використовуються для діагностики захворювань сітківки, таких як вікова макулярна дегенерація та діабетична ретинопатія. Цей набір даних, який охоплює зображення сітківки, отримані під час флюоресцентної ангіографії, має на меті сприяти розвитку як медичної діагностики, так і обробки природної мови, надаючи зрозумілі анотації та надійні механізми оцінки для автоматизованих систем генерації медичних звітів.

Використаний набір включає 21 тип уражень, кожен з яких представлений не менш ніж 100 зображеннями. Таким чином, для цілей цього аналізу було вибрано 25 найпоширеніших уражень, кожне з яких представлено більш ніж 100 зображеннями. Всі отримані зображення були поділені на навчальні (75%), валідаційні (10%) та тестові (15%) набори. Всі зображення було проаналізовано та промарковано, щоб визначити різні типи уражень і їх локалізацію.

3.2 Моделювання та тренування моделі

З метою забезпечення послідовності модель було навчено загалом протягом 200 епох в однаковому середовищі GPU. Результати, отримані в кінці процесу навчання, зображені на рисунку 3.1. У цій роботі використовується метрики оцінки точності та повноти, які зазвичай застосовуються в завданнях виявлення об’єктів. Точність є пропорцією істинно позитивних виявлень серед усіх позитивних виявлень, тоді як повнота відноситься до пропорції істинно позитивних виявлень серед усіх фактичних об’єктів. Отримані показники точності та повноти становили приблизно 0.856 та 0.834 відповідно. Ці результати свідчать про високий рівень точності та всеосяжності виявлення об’єктів даною моделлю.

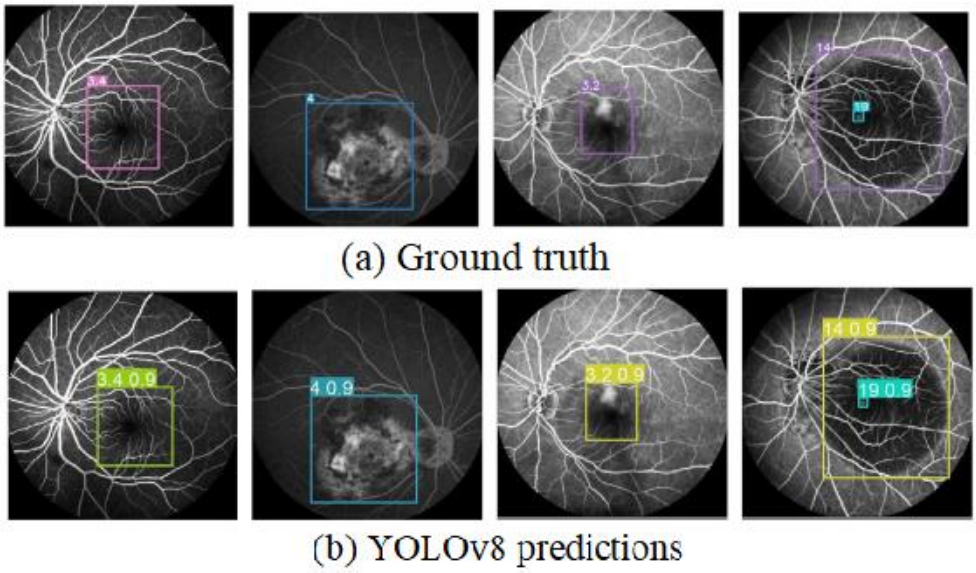

Рисунок 3.1 – Результати навчання моделі

Крім точності та повноти, також була розрахована середня точність (Mean Average Precision). Це популярна метрика у виявленні об’єктів, що представляє собою статистичний показник, який підсумовує загальну продуктивність моделі, враховуючи різні масштаби та співвідношення різних сторін об’єктів. Середня точність була розрахована як при пороговому значенні IoU 0.5, так і в діапазоні від

0.5 до 0.95. Результати становили приблизно 0.838 та 0.524 відповідно. Ці показники свідчать про високу продуктивність моделі у точному та всеосяжному виявленні об’єктів, навіть з урахуванням різних порогів перетину об’єктів.

Загалом, показники продуктивності, отримані при використанні моделі YOLOv8 (середня точність та повнота), демонструють доволі непогані результати, які свідчать про надійну здатність моделі виявляти об’єкти з високим рівнем точності та повноти. З рисунку 5 видно, що модель YOLOv8 демонструє помітну здатність точно прогнозувати мітки уражень. Прогнозовані обмежувальні рамки показують високий ступінь відповідності з істинними обмежувальними рамками, що вказує на надійність моделі у виявленні та локалізації уражень на зображеннях флюоресцентної ангіографії ока.

3.3 Інтеграція моделі з веб-додатком

Інтеграція моделі YOLOv8 з веб-додатком є важливим кроком для забезпечення зручності використання та ефективності системи. Процес включає декілька етапів, від підготовки серверної частини до забезпечення взаємодії з користувачем через інтерфейс. Нижче наведені основні кроки інтеграції.

Веб-застосунок надає зручний та орієнтований на користувача інтерфейс для розгортання моделей YOLO з метою прогнозування уражень на зображеннях ока. Таким чином, він має значний потенціал для застосування в умовах реальних клінічних досліджень, допомагаючи у діагностиці та лікуванні різноманітних очних захворювань. Додаток використовує кілька бібліотек та модулів, включаючи OpenCV, PyTorch, NumPy, Streamlit та Ultralytics.

Загалом код веб-додатку було написано на мові Python, яка дозволяє швидко та ефективно реалізувати функціонал завдяки широкому набору бібліотек. Таким чином, для реалізації графічного інтерфейсу користувача та подальшого розгортання додатку було обрано бібліотеку Streamlit. Даний інструмент спрощує

процес розробки програмного забезпечення, надаючи можливість створення веб-інтерфейсів без глибоких знань веб-програмування. Streamlit використовує декларативний підхід, при якому розробники можуть писати мінімум коду для відображення графіків, таблиць, форм та інших елементів інтерфейсу. Завдяки своїй простоті, Streamlit є популярним інструментом серед розробників, для швидкої демонстрації своїх результатів.

Отже, щоб додати до сторінки заголовок та вказати його розташування та колір, достатньо написати лише декілька рядків коду. Код на мові Python для відображення заголовку сторінки “Виявлення уражень зору” подано в лістингу 3.1.

Лістинг 3.1 – Розміщення заголовку сторінки використовуючи Streamlit

```
import streamlit as st

# HTML код для заголовка сторінки Streamlit
markdown_html = """
    <section style = "background-color: #FF5733; padding: 16px;">
        <h2 style="color: #FFFFF0; text-align: center;">
            Виявлення уражень зору
        </h2>
    </section>
"""

# Відображення заголовка сторінки
st.markdown(markdown_html, unsafe_allow_html=True)
```

З лістингу можна побачити, що спочатку імпортується бібліотека Streamlit та визначається змінна “html”, яка містить HTML-код для створення заголовка сторінки, який розташовується в центрі, має білий колір тексту та світло-червоний фон, за допомогою методу “st.markdown” вміст змінної додається на веб-сторінку з параметром “unsafe_allow_html” рівним значенню True, що дозволяє використовувати HTML-розмітку для відображення заголовка. Вигляд, якого набуде веб-сторінка в результаті запуску цього коду, зображено на рисунку 3.2.

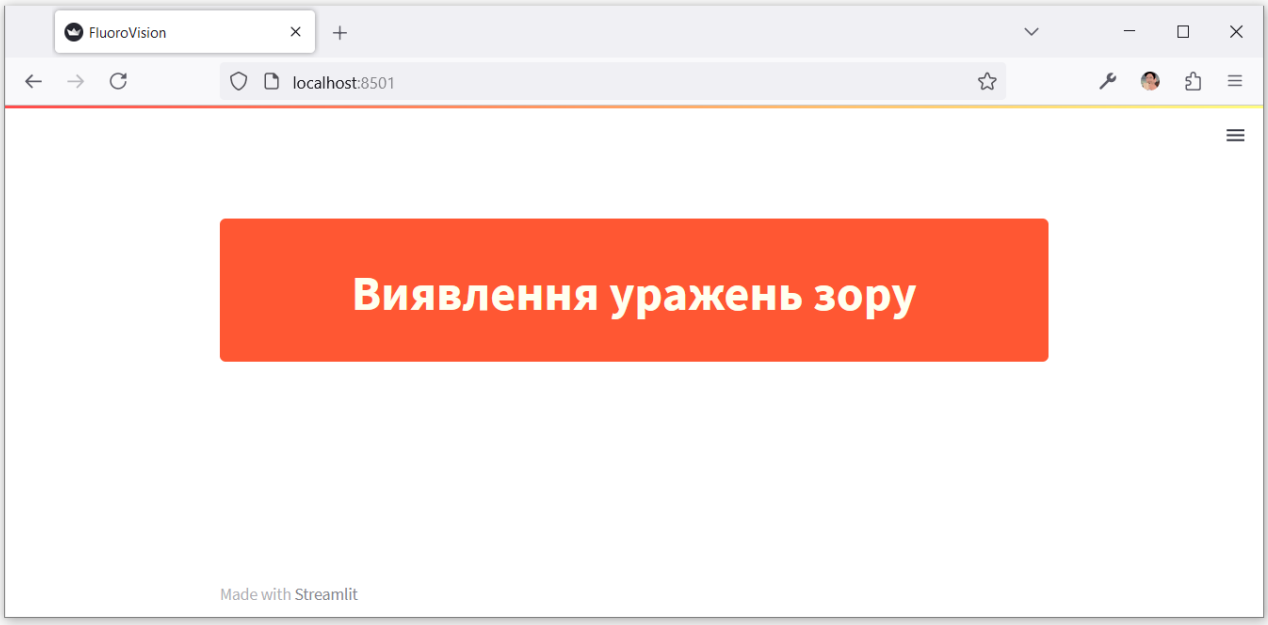

Рисунок 3.2 – Відображення заголовку сторінки

Схожим чином до сторінки можна додати бічну панель (sidebar), на якій будуть розташовані різноманітні списки (selectbox) та повзунки (slider), які дозволятимуть налаштовувати параметри аналізу. Все це реалізовується за допомогою спеціальної змінної “sidebar” об’єкта бібліотеки Streamlit. Приклад додання списків до бічної панелі подано в лістингу 3.2.

Лістинг 3.2 – Додання списків до бічної панелі

```
# Сайдбар для налаштувань користувача
st.sidebar.markdown(
    '<p class="format-style">Параметри</p>',
    unsafe_allow_html=True
)

# Вибір моделі користувачем
st.sidebar.selectbox(
    'Модель',
    ('YOLOv8', 'YOLOv5', 'YOLOv3'),
    format_func=lambda a: a[:len(a)]
)

# Вибір ваг користувачем
st.sidebar.selectbox(
    'Ваги',
    ('yolov8.pt')
)
```

Цей код додає до бічної панелі налаштувань користувача два основні елементи. Спочатку створюється заголовок “Параметри” для панелі, використовуючи метод “markdown” з можливістю включення HTML-коду. Потім додаються два випадаючі списки (selectbox), які дозволяють користувачу вибирати модель та ваги. Це забезпечує гнучкість у налаштуванні параметрів аналізу, надаючи можливість адаптувати додаток під конкретні потреби користувача. В результаті веб-сторінка набуде наступного вигляду (див. рисунок 3.3).

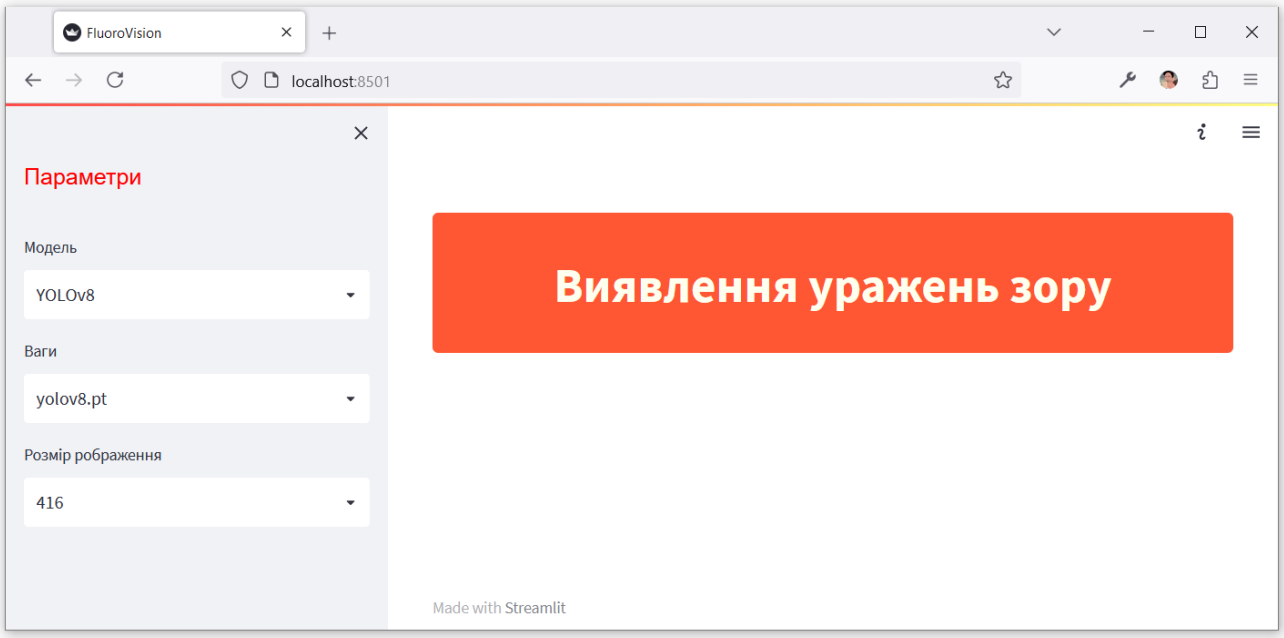

Рисунок 3.3 – Використанні бічної панелі на веб-сторінці

Для використання повзунків, потрібно лише скористатись відповідним методом “slider”, якому в якості параметрів передати назву мітки для слайдера, а також мінімальне та максимальне значення, разом з кроком переміщення. Повзунки дозволяють користувачам налаштовувати числові параметри шляхом переміщення маркера вздовж осі, що відображає діапазон значень. Це дуже корисно для налаштувань, які вимагають регулювання, таких як визначення порогу впевненості або порогу перетину об’єктів. Використання повзунків на мові Python показано в лістингу 3.3.

Лістинг 3.3 – Використанні повзунків на бічній панелі

```
# Вибір порогу впевненості користувачем
st.sidebar.slider(
    'Поріг впевненості', 0.00, 1.00, 0.5
)

# Вибір порогу перетину об'єктів (IoU) користувачем
st.sidebar.slider(
    'Поріг перетину об\'єктів (IoU)', 0.00, 1.00, 0.35
)

# Вибір класів для обробки користувачем
st.sidebar.multiselect(
    'Класи',
    [i for i in range(len(CLASSES1))],
    format_func=lambda index: CLASSES1[index]
)

# Вибір усіх класів для обробки
all_classes = st.sidebar.checkbox('Всі класи', value=True)
```

Даний код створює елементи користувацького інтерфейсу на бічній панелі веб-додатка з використанням бібліотеки Streamlit. Зокрема, додаються повзунки для налаштування параметрів аналізу зображень, таких як поріг впевненості моделі та поріг перетину об’єктів (IoU), а також список для вибору класів об’єктів, які будуть оброблятися моделлю.

Повзунки дозволяють користувачу інтуїтивно встановлювати числові значення в межах заданого діапазону, що спрощує процес налаштування. Додатково, множинний вибір (multiselect) дозволяє вибрати конкретні класи об’єктів для аналізу, а прапорець (checkbox) забезпечує можливість обробки всіх класів одночасно.

Цей підхід дозволяє забезпечити високу гнучкість та зручність використання інтерфейсу, дозволяючи користувачам швидко та легко налаштовувати параметри аналізу відповідно до своїх потреб. Таким чином, реалізація налаштувань через бічну панель сприяє кращій взаємодії користувача з веб-додатком, підвищуючи його ефективність та функціональність. Вигляд повністю побудованої бічної панелі зображено на рисунку 3.4.

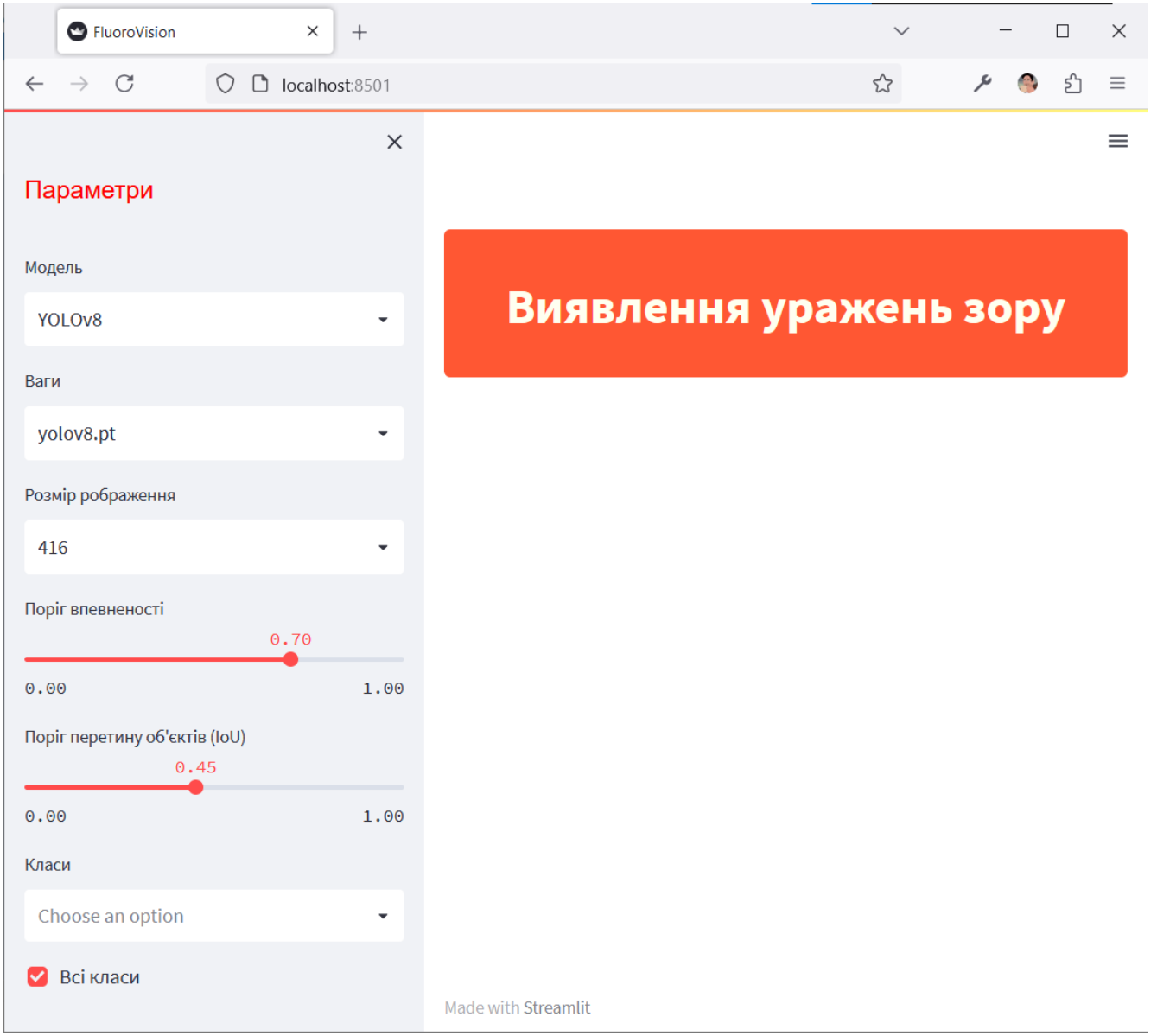

Рисунок 3.4 – Вигляд бічної панелі

Для візуального відображення процесу завантаження моделі можна скористатись методом “spinner” з бібліотеки Streamlit. Цей метод створює візуальний індикатор завантаження та приймає в якості параметра текстове повідомлення, наприклад “Завантаження моделі...” , яке буде відображатися під час виконання завантаження. Після успішного завершення завантаження буде відображено повідомлення, яке підтверджуватиме цю подію. Це можна реалізувати, використавши метод “success”, все з тієї ж бібліотеки. Він також приймає стрічку в якості аргументів – текст, який виводиться на екран в якості успішного завантаження моделі.

Код на мові Python, наведений в лістингу 3.4, реалізує цей функціонал. Використовуючи метод “spinner”, відбувається завантаження моделі на основі вибраних користувачем параметрів (у цьому випадку обрана модель YOLOv8). Після завершення завантаження моделі виводиться повідомлення про успішне завершення завантаження. Вигляд веб-сторінки при успішному завантаженні моделі зображено на рисунку 3.5.

Лістинг 3.4 – Відображення статусу завантаження моделі

```
# Завантаження моделі з вибраними параметрами
with st.spinner('Завантаження моделі...'):
    if modelSelect == 'YOLOv8':
        model = get_yolo8(weights)

st.success('Модель "' + modelSelect + '" завантажено успішно!')
```

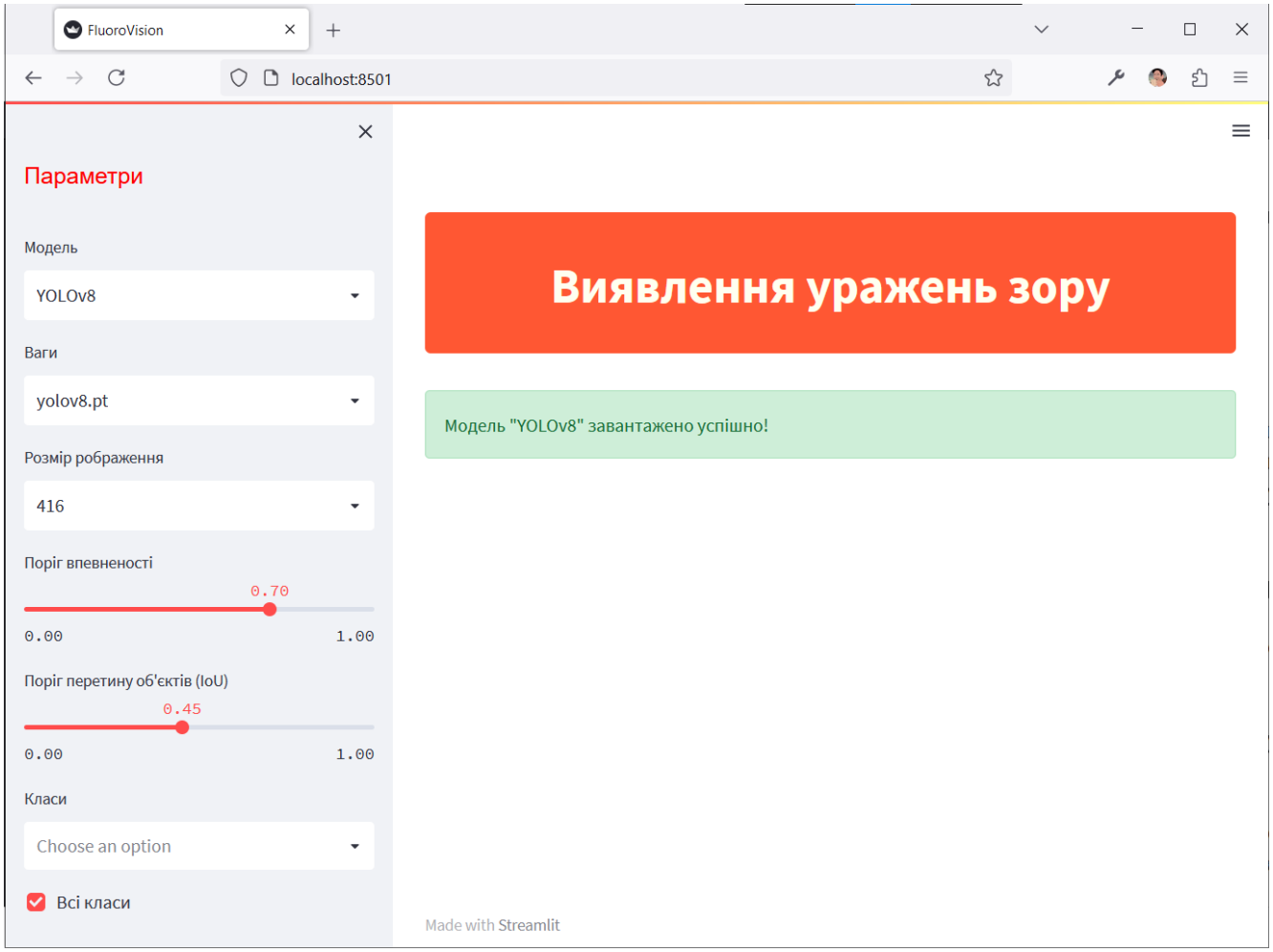

Рисунок 3.5 – Виведення повідомлення про успішне завантаження моделі

Останнім елементом графічного інтерфейсу буде віджет для завантаження зображень для їх подальшого аналізу. Це можна реалізувати за допомогою методу “file_uploader” бібліотеки Streamlit. Він приймає в параметрах текстове повідомлення для відображення над віджетом, допустиме розширення зображення та прапорець для вказання, чи можна завантажувати декілька зображень одночасно. Програмний код, який реалізує даний функціонал, подано в лістингу 3.5.

Лістинг 3.5 – Відображення віджета для завантаження зображень

```
# Функція для завантаження зображень
def load_images(file_uploader):
    uploaded_files = file_uploader(
        "Виберіть зображення",
        type=['png', 'jpg']
    )
    image_filenames, img_collection = [], []
    for uploaded_file in uploaded_files:
        bytes_data = uploaded_file.getvalue()
        file_bytes = np.asarray(
            bytearray(bytes_data),
            dtype=np.uint8
        )
        img = cv2.imdecode(file_bytes, cv2.IMREAD_COLOR)
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        img_collection.append(img)
        image_filenames.append(uploaded_file.name)

    return [images, image_filenames]
```

Даний код створює функцію “load_images”, яка обробляє завантажені зображення через віджет “file_uploader”. Використовуючи метод “file_uploader”, користувач може завантажувати файли типу “png” або “jpg”. Якщо вказано параметр “accept_multiple_files” рівний значенню True, користувач може завантажувати декілька файлів одночасно. Функція “load_images” обробляє кожен завантажений файл, зчитуючи його байти, перетворюючи їх у масив NumPy, декодуючи зображення за допомогою OpenCV та конвертуючи його з BGR у RGB. Оброблені зображення зберігаються в списку “img_collection”, а імена файлів – у списку “image_filenames”. Функція повертає обидва списки для подальшого використання. Вигляд даного віджета зображено на рисунку 3.6.

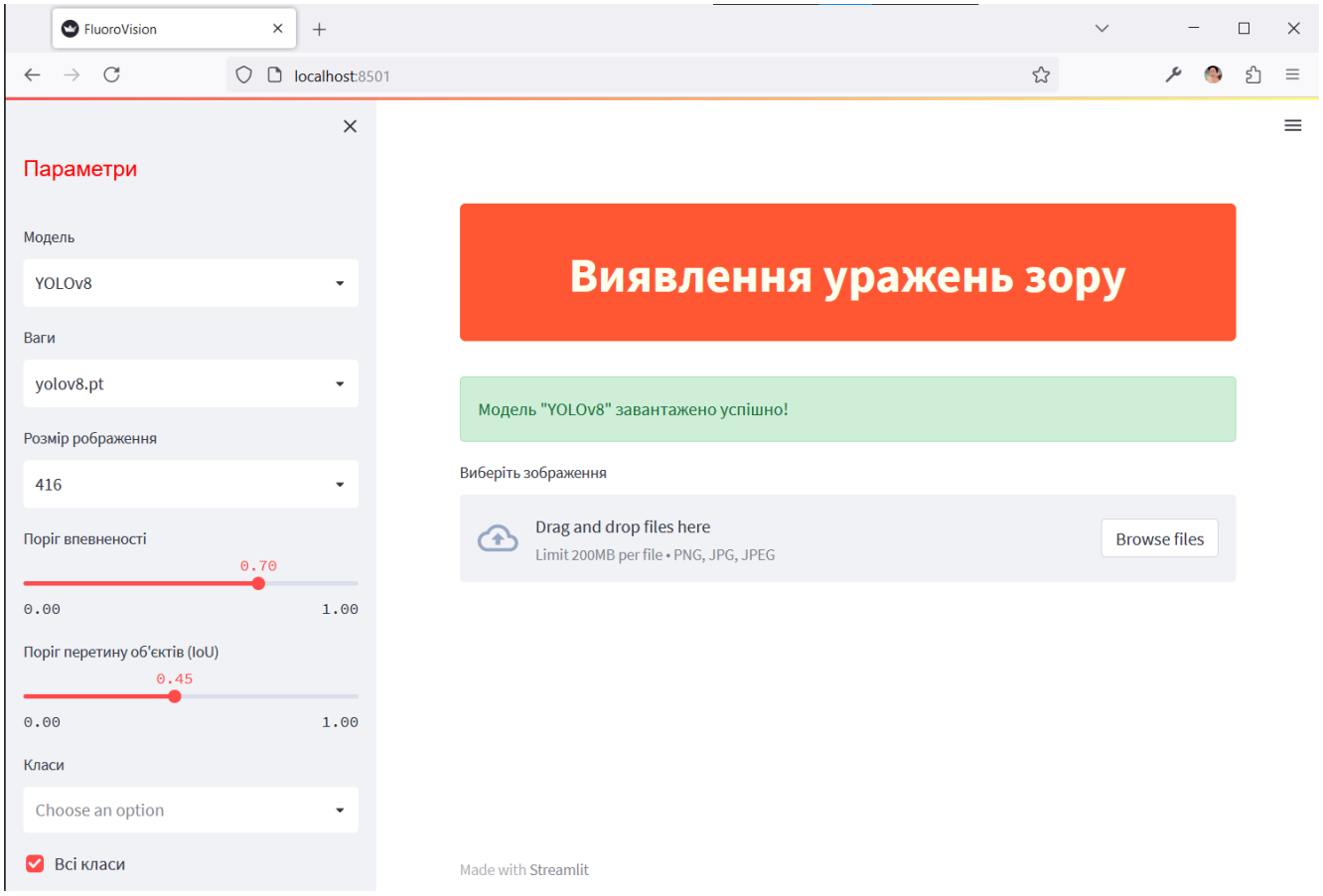

Рисунок 3.6 – Реалізований інтерфейс веб-сторінки із завантаженням файлів

На рисунку можна побачити, що віджет для завантаження зображень розташований у центральній частині сторінки веб-додатка, безпосередньо під заголовком “Виявлення уражень зору” та повідомленням про успішне завантаження моделі. Віджет представлений у вигляді області для перетягування файлів, яка має підпис “Drag and drop files here” та кнопку "Browse files" для вибору файлів з комп’ютера.

Також під віджетом вказано обмеження на розмір файлу та підтримувані формати файлів. Віджет забезпечує простий та інтуїтивно зрозумілий спосіб завантаження зображень для аналізу, надаючи користувачам можливість легко додавати файли до системи. Вигляд діалогового меню, яке відкривається при натисненні на кнопку “Browse files”, подано на рисунку 3.7.

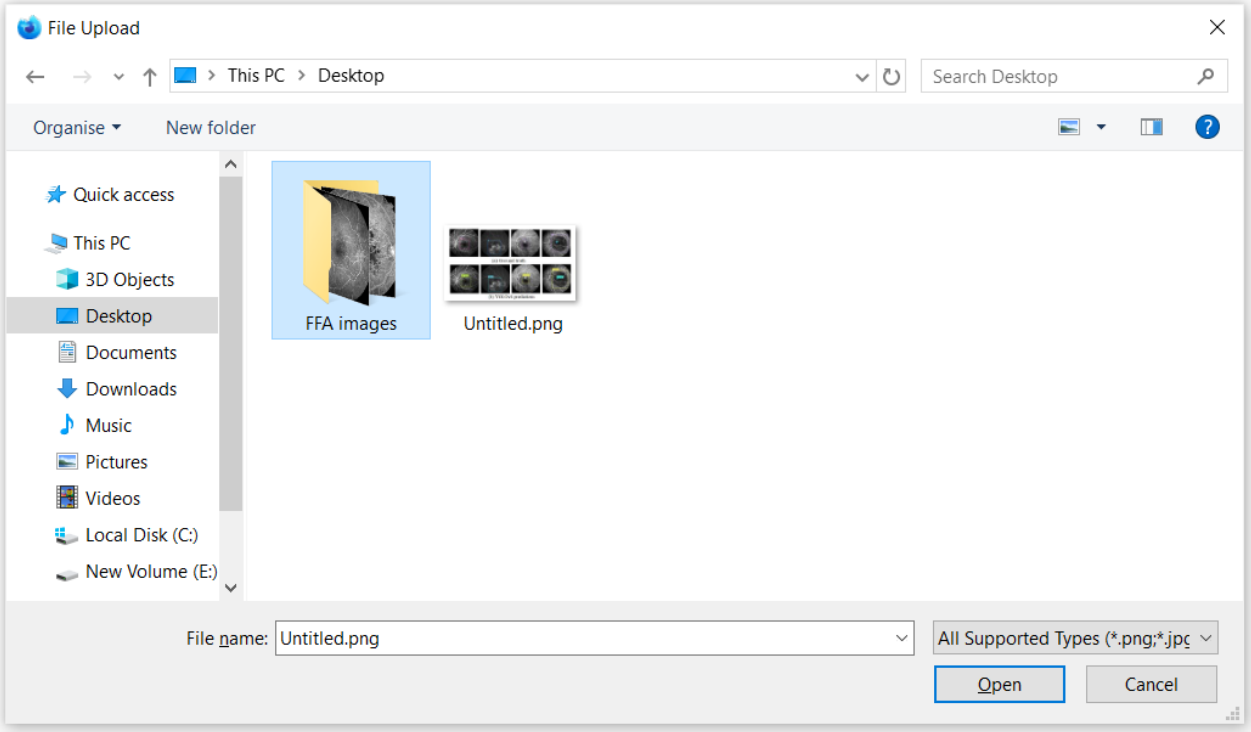

Рисунок 3.7 – Діалогове вікно для завантаження зображень

Після завантаження зображень для аналізу, система відразу починає їх обробляти за допомогою попередньо завантаженої моделі YOLO. За це відповідає спеціальна функція “process_images”, код якої подано в лістингу 3.6.

Лістинг 3.6 – Функція для аналізу завантажених зображень

```
# Функція для обробки зображень
def process_images(image_list, class_ids, model_weights):
    processed_results = []

    for image in image_list:
        predictions = get_preds(image, imgsz)
        filtered_predictions = predictions.copy()
        filtered_predictions =
            filtered_predictions[
                np.isin(filtered_predictions[:, -1], class_ids)
            ]
        processed_results.append(filtered_predictions)

    return processed_results
```

Даний код виконує обробку списку зображень, фільтруючи їх за певними класами. Для кожного зображення у списку викликається функція “get_preds”, яка

повертає передбачення для цього зображення. Потім ці передбачення копіюються та фільтруються за допомогою методу “np.isin”, який залишає тільки ті передбачення, класи яких відповідають вказаним у “class_ids”. Відфільтровані передбачення додаються до списку “processed_results”, який повертається в кінці функції.

Функція “process_images” є корисною для обробки зображень у рамках задачі комп’ютерного зору, де потрібно аналізувати тільки певні типи об’єктів на зображеннях. Вона забезпечує зручний спосіб фільтрації результатів, що дозволяє зосередитися на релевантних класах, зменшуючи шум від інших передбачень.

За відображення обмежувальних рамок відповідатиме функція “draw_bounding_boxes”, повний код якої наведено в додатку А. Функція проходить через кожне зображення та його відповідні результати обробки. Якщо впевненість (confidence) передбачення перевищує заданий поріг, вона малює обмежувальну рамку навколо виявленого об’єкта, а також додає текстову мітку з номером класу і впевненістю. Результати, включаючи зображення з намальованими рамками та списки виявлених класів, додаються до списку зображень, який повертається в кінці функції. В результаті отримується нове зображення, вигляд якого подано на рисунку 3.8.

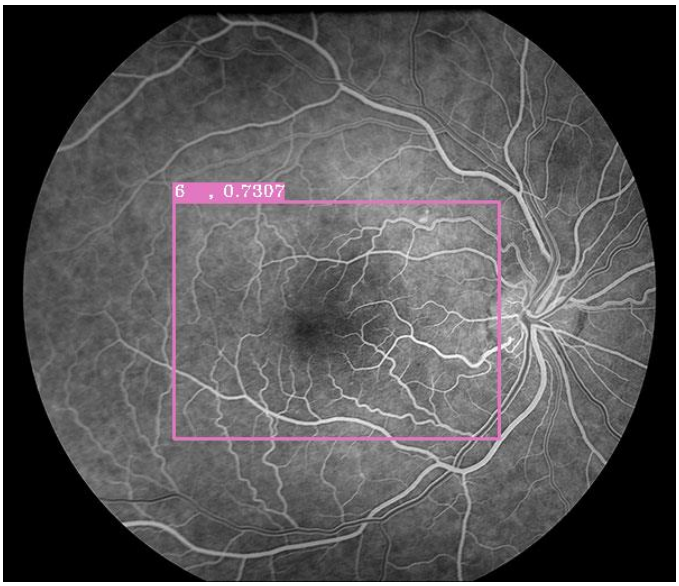

Рисунок 3.8 – Результат обробки зображення та малювання обмежувальної рамки

Після обробки, нове зображення з обмежувальною рамкою можна завантажити на свій комп’ютер. Це реалізується за допомогою методу “download_button” бібліотеки Streamlit, яка створює віджет у вигляді кнопки, при натисненні на яку починається завантаження файлу. В програмному коді це реалізовано у вигляді функції “save_single_image”, код якої наведено в лістингу 3.7. Вигляд веб-сторінки після аналізу зображення подано на рисунку 3.9.

Лістинг 3.7 – Код для збереження проаналізованих зображень

```
# Функція для завантаження одного зображення
def save_single_image(image_data, output_name):
    _, modified_image = image_data
    image_as_bytes = image_to_bytes(modified_image)
    output_file_name = f"{output_name.rsplit('.', 1)[0]}.png"
    st.download_button(
        label="Завантажити дане зображення",
        data=image_as_bytes,
        file_name=output_file_name,
        key=download_single'
    )
```

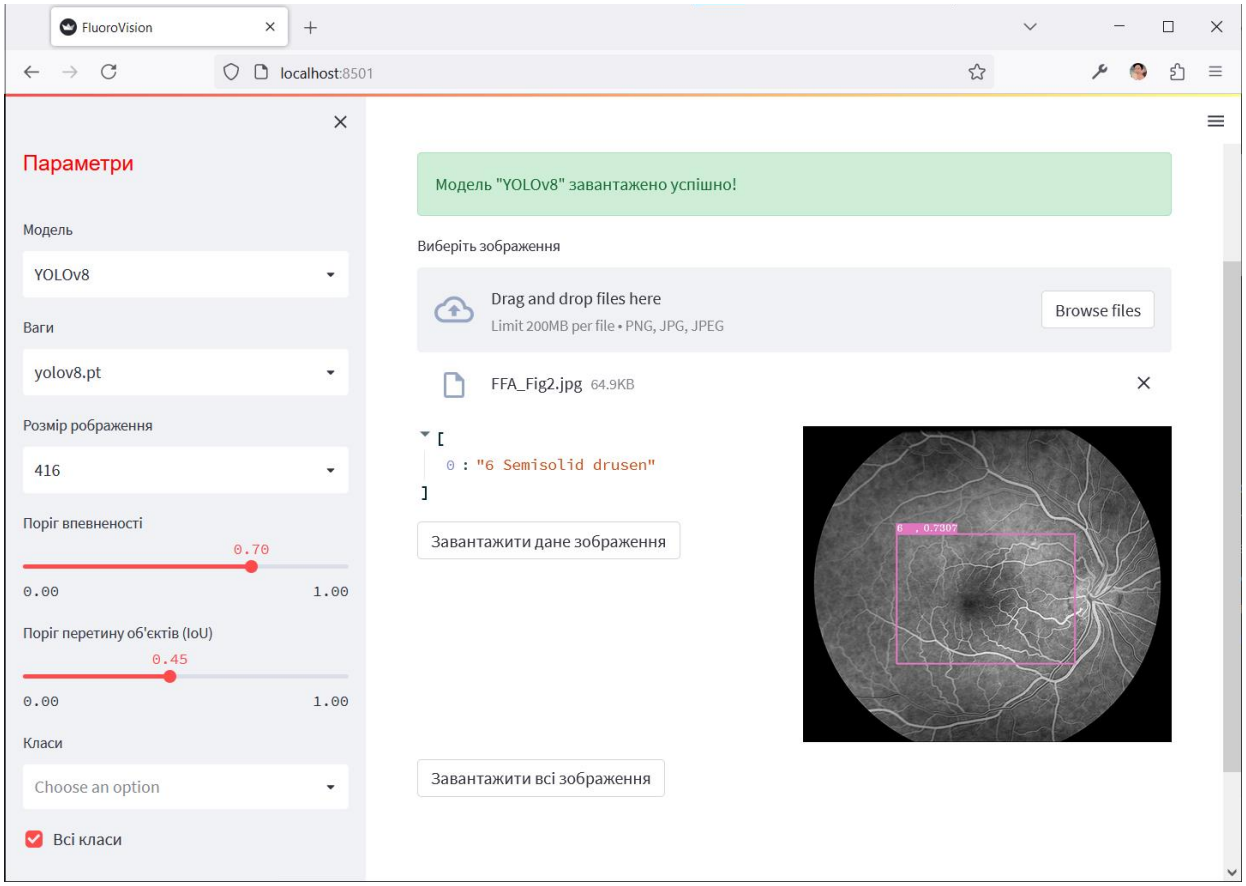

Рисунок 3.9 – Відображення кнопок для збереження проаналізованих зображень

В даній функції зображення конвертується у байтовий формат за допомогою окремої функції “image_to_bytes”, після чого визначається нове ім’я файлу, у який буде збережено отримане зображення. За саму функцію завантаження відповідає метод “download_button” з бібліотеки Streamlit, яка відображає саму кнопку, що відповідає за збереження файлу.

Схожим чином реалізована функція “save_zip”, яка дозволяє зберегти не одне окреме зображення, а всі разом у вигляді ZIP-архіву. Це виконано з використанням бібліотеки “io” та методу “BytesIO” для створення буфера, в який буде записано сформований архів. В кінці викликається вже знайомий метод “download_button”, який відображає кнопку з назвою “Завантажити всі зображення” для збереження всіх проаналізованих фотографій.

3.4 Тестування веб-додатку

Однією з основних частин життєвого циклу розробки програмного забезпечення є тестування. Це набір завдань, спрямованих на те, щоб переконатися, чи програмне забезпечення працює як слід, є зручним у використанні і виконує свої функції. Тестування програмного забезпечення проводиться для виявлення багів або помилок.

Точність (precision) та повнота (recall) є мірами точності моделі. Висока точність означає, що модель робить мало хибних виявлень, тоді як висока повнота означає, що модель виявляє більшість об’єктів істинно (ground truth). Середнє значення точності є значенням по всіх класах у наборі даних і надає загальну міру точності моделі. Дана метрика розраховується на різних порогах IoU (Intersection over Union), два з яких становлять 0.5 та 0.5:0.95. Значення 0.5 в даному випадку означає, що 50 відсотків площі передбаченої обмежувальної рамки та істинної обмежувальної рамки повинні перекриватися, щоб передбачення вважалося позитивним. Результат 0.5:0.95 вказує на те, що перекриття повинно бути між 50 та

95 відсотками. В загальному, для визначення загальної продуктивності моделі необхідно враховувати компроміс між точністю та повнотою.

Існує декілька можливих причин, чому модель демонструє кращу продуктивність за певною метрикою, деякі з яких наступні:

- Різні моделі мають різну архітектуру та здатність вивчати й представляти ознаки об’єктів, що може впливати на їхню продуктивність за різними метриками.
- Якість та різноманітність навчальних даних можуть впливати на продуктивність моделі за різними метриками.
- Значення гіперпараметрів, таких як швидкість навчання та кількість епох, можуть мати значний вплив на продуктивність моделі за різними метриками.
- Характеристики даних, такі як складність об’єктів, рівень шуму та розподіл екземплярів об’єктів, можуть впливати на продуктивність моделі за різними метриками.

Для перевірки моделі системи було використано тестовий набір, отриманий з набору даних FFA-IR, за допомогою запропонованого веб-застосунку. Середнє значення точності для проаналізованих зображень становило приблизно 0.8. Це підкреслює виняткову продуктивність програмного забезпечення в анотації уражень, що демонструє його потенціал як ефективного інструменту для прийняття клінічних рішень.

Використання автоматизації в тестуванні програмного забезпечення має значне значення завдяки своїй здатності підвищувати операційну ефективність, дозволяючи швидше завершувати тести. Крім того, автоматизація забезпечує високу точність тестів, що дозволяє досягти ширшого охоплення тестів, особливо в складних випадках [9]. Крім того, автоматизація підтримує однорідність у тестувальних процедурах, зменшуючи ймовірність виникнення людських помилок. Крім того, вона сприяє ефективному регресійному тестуванню та дозволяє одночасне виконання тестів.

В кінцевому підсумку, впровадження автоматизації в тестування програмного забезпечення призводить до створення програмного забезпечення високої якості з меншою кількістю дефектів. Основною метою автоматизованого тестування програмного забезпечення є досягнення багатьох цілей, які включають дослідження багатьох інструментів тестування, що широко використовуються в індустрії, розуміння успішних підходів до розробки автоматизованих тестових випадків і оволодіння процесом написання сценаріїв і швидкого виконання цих тестів.

Для забезпечення ефективного та надійного тестування розробленого веб-додатку було обрано інструмент Selenium IDE. Це інтегроване середовище розробки для створення веб-тестів на основі Selenium. Воно реалізоване як розширення для браузерів Firefox та Chrome і забезпечує ефективний спосіб розробки, запису та налагодження тестів. Selenium IDE включає весь Selenium Core, що дозволяє легко та швидко записувати та виконувати тести в реальному середовищі. Selenium WebDriver широко використовується для автоматизації браузерів, тоді як Selenium IDE дозволяє користувачам створювати та виконувати тести через розширення для браузера [10].

Основні переваги Selenium IDE:

- Простота використання – інтерфейс інтуїтивно зрозумілий і не потребує глибоких знань у програмуванні.
- Швидкість тестування – можливість швидко створювати та виконувати тести, суттєво прискорюючи процес тестування.
- Можливість редагування тестів – усі тести можна легко редагувати та налаштовувати відповідно до потреб застосунку.
- Підтримка різних браузерів – автоматизоване тестування можна проводити в багатьох браузерах, що забезпечує крос-браузерну сумісність системи.

Перед початком роботи необхідно створити новий проєкт для збереження та виконання автоматизованих тестів. Для цього варто лише відкрити встановлене розширення Selenium IDE, після чого з’явиться діалогове вікно для створення нового проєкту. Вигляд цього вікна зображено на рисунку 3.10.

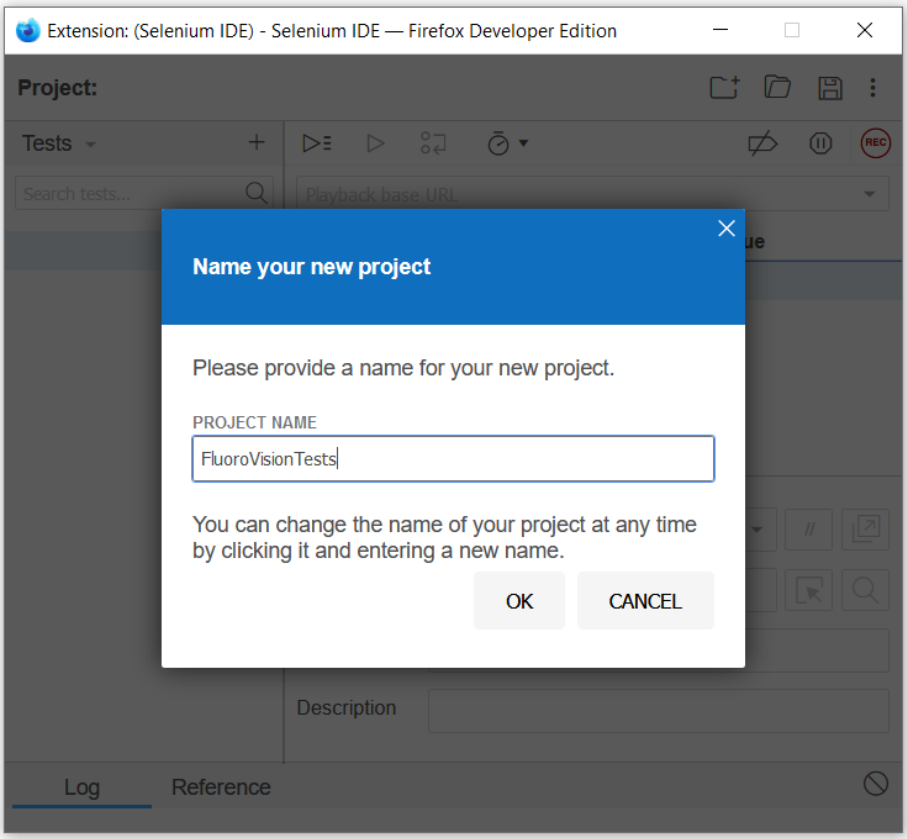

Рисунок 3.10 – Створення нового проєкту в Selenium IDE

Для створення нових тестів (сценаріїв) потрібно записати взаємодію користувача з веб-додатком. Selenium IDE записує всі дії користувача, такі як натискання кнопок, введення тексту, навігація між сторінками тощо. Після запису тесту можна відредагувати тестовий сценарій для оптимізації та налаштування. Це включає додавання перевірок (assertions) для верифікації правильності роботи додатку. Після запису створені тести можна запустити, Selenium IDE дозволяє запускати тести автоматично, перевіряючи коректність роботи додатку відповідно до записаного сценарію. Після виконання тесту аналізуються результати для виявлення помилок та неточностей у роботі додатку. У разі виявлення помилок проводяться виправлення та повторне тестування.

Створимо декілька автоматизований тестів для перевірки реалізованого функціоналу веб-застосунку, такого як: налаштування параметрів аналізу, завантаження моделі, аналіз зображень та коректне відображення результатів тощо. Результат запуску створених тестів зображено на рисунку 3.11.

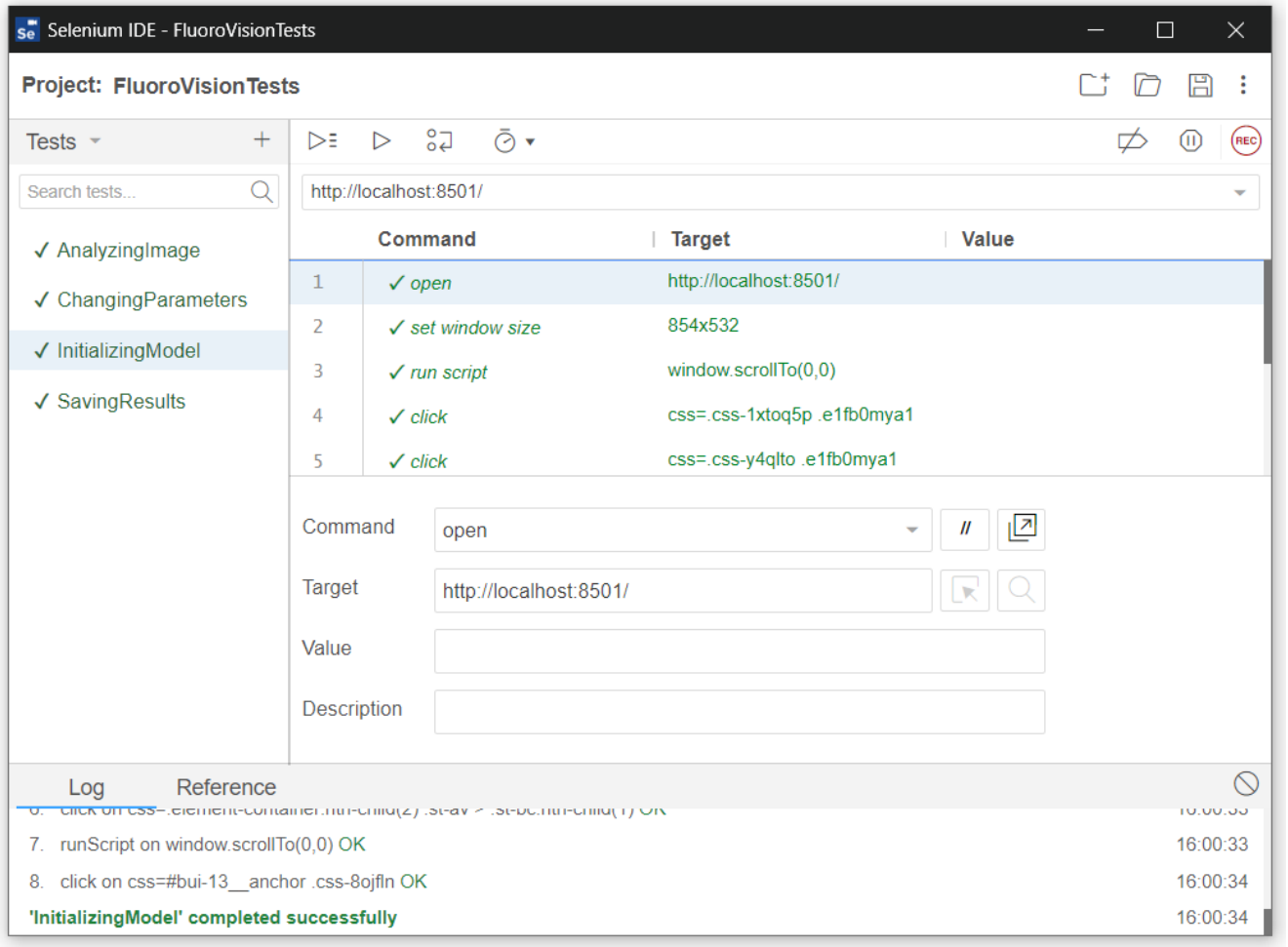

Рисунок 3.11 – Успішне виконання автоматизованих тестів

З рисунку видно, що всі основні функціональні можливості працюють коректно. Веб-додаток успішно завантажується, параметри конфігурації обираються без помилок, зображення завантажуються та обробляються належним чином, а результати аналізу відображаються та можуть бути завантажені користувачем.

Усі тести пройшли успішно, що свідчить про високу якість і надійність розробленого веб-додатку. Це забезпечує впевненість у його готовності до використання кінцевими користувачами. Таким чином, використання Selenium IDE дозволяє автоматизувати процес тестування веб-додатку, забезпечуючи високу якість та надійність програмного забезпечення.

3.5 Розгортання системи

Розгортання системи є заключним етапом розробки програмного забезпечення, який включає підготовку середовища та розміщення веб-додатку на сервері або хмарній платформі. Правильне розгортання забезпечує доступність додатку для кінцевих користувачів та його стабільну роботу.

Для розгортання веб-додатку було обрано платформу Streamlit Community Cloud, яка забезпечує просте та швидке розгортання веб-додатків з використанням Git-репозиторіїв. Streamlit Community Cloud надає можливість безкоштовного хостингу для додатків, розроблених за допомогою бібліотеки Streamlit. Основні переваги Streamlit Community Cloud:

- Інтуїтивно зрозумілий інтерфейс та простий процес розгортання.
- Можливість автоматичного розгортання додатку безпосередньо з GitHub-репозиторію.
- Відсутність грошових витрат на хостинг для невеликих проектів.

Перед розгортанням необхідно підготувати середовище, включаючи встановлення необхідних залежностей та налаштування конфігурацій. Для початку потрібно вказати всі використані бібліотеки та залежності у файлі “requirements.txt”. Цей файл використовується для автоматичної інсталяції залежностей під час розгортання, що дозволить Streamlit Community Cloud автоматично встановити всі необхідні пакети під час розгортання. Приклад файлу “requirements.txt” може містити такі пакети, як streamlit, numpy, opencv-python, ultralytics тощо.

Streamlit Community Cloud запускає додатки безпосередньо з особистого репозиторію GitHub, тому весь необхідний код додатку та залежності мають бути на GitHub перед тим, як починати розгорнути додаток. Після створення репозиторію та переміщення до нього всіх потрібних файлів проєкту, потрібно перейти на платформу Streamlit Community Cloud, увійти в систему за допомогою облікового запису GitHub, та створити новий додаток, вибравши відповідний

репозиторій і вказавши основні налаштування додатку. Потрібно вказати назву основного файлу додатку, зазвичай це “app.py”, а також гілку репозиторію, яку потрібно розгорнути. Процес розгортання веб-додатку зображено на рисунку 3.12.

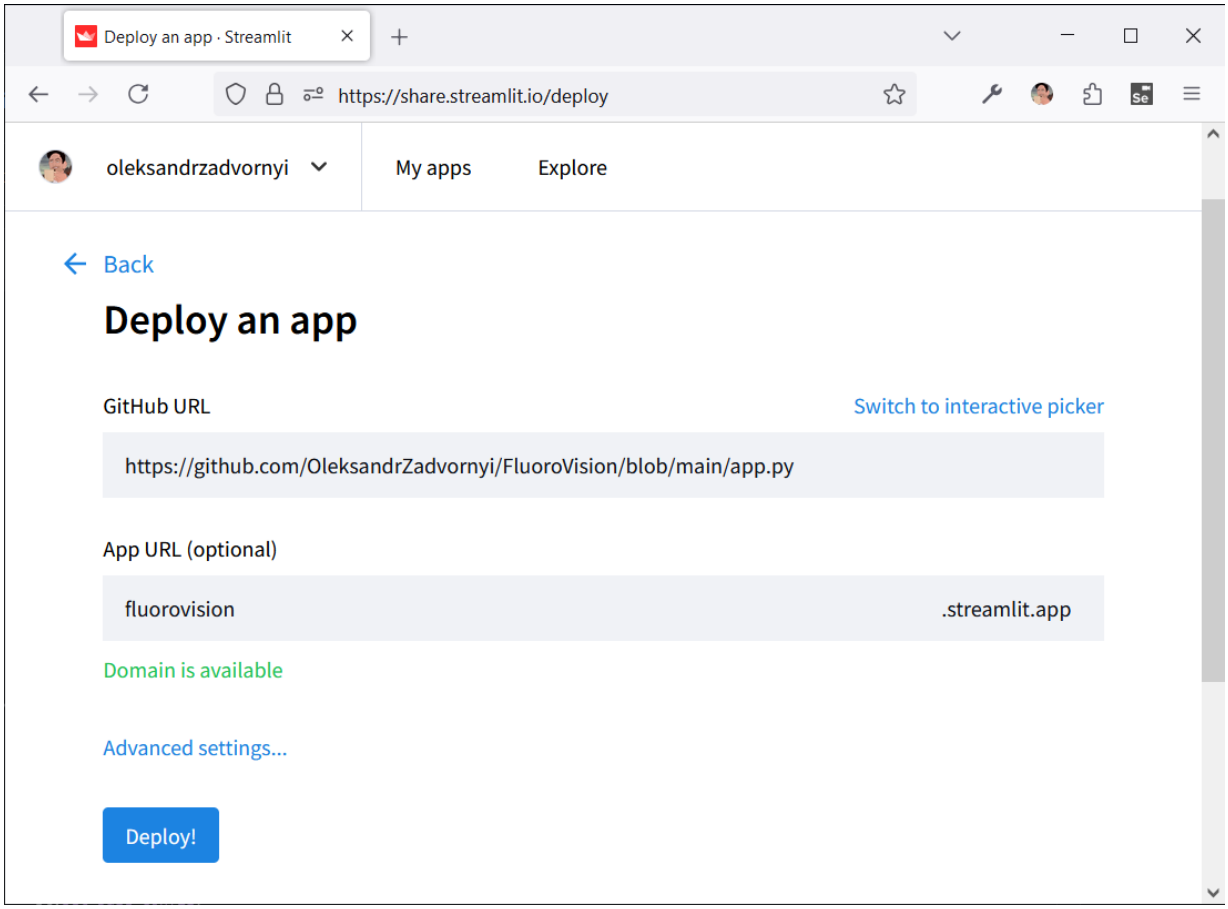

Рисунок 3.12 – Розгортання додатку за допомогою Streamlit Community Cloud

Після налаштування конфігурації, Streamlit Community Cloud автоматично розгорне додаток та надасть посилання для доступу до нього. В даному випадку, веб-додаток було розгорнуто за посиланням https://fluorovision.streamlit.app/. Можна перейти за цією адресою, щоб переконатися, що додаток працює коректно. У разі необхідності внесення змін до коду, достатньо оновити репозиторій на GitHub, і Streamlit Community Cloud автоматично перезапустить додаток з новими змінами. Приклад коректної роботи системи за наведеним посиленням зображено на рисунку 3.11.

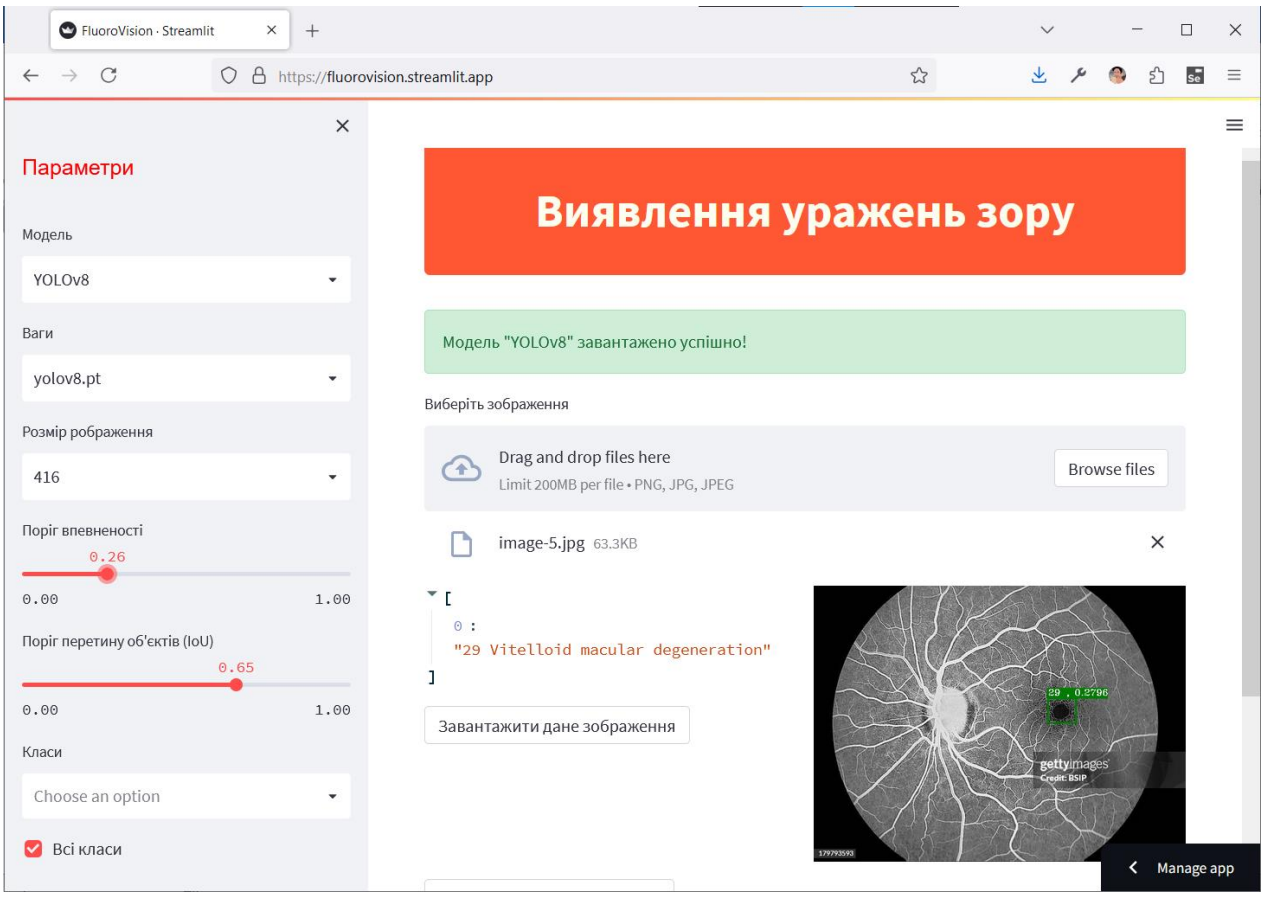

Рисунок 3.13 – Використання розгорнутого веб-додатку

Таким чином, розгортання веб-додатку з використанням Streamlit Community Cloud є швидким та зручним процесом, який не потребує глибоких знань з веб-програмування. Завдяки використанню Streamlit Community Cloud можна ефективно розгорнути веб-додаток, забезпечивши його доступність для кінцевих користувачів та спростивши процес оновлення та підтримки.

Після розгортання важливо забезпечити постійний моніторинг та підтримку додатку. Streamlit Community Cloud надає інструменти для моніторингу продуктивності та логів, що дозволяє вчасно реагувати на можливі проблеми та забезпечувати стабільну роботу системи.

4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Комплексний аналіз життєдіяльності людини

Безпека життєдіяльності людини вивчається, використовуючи системний підхід, шляхом аналізу прямих і зворотних зв'язків в системі "людина – життєве середовище". Ця система складається з двох компонентів: людина і життєве середовище, які взаємопов'язані між собою. Людина реагує на дію чинників зовнішнього середовища відповідними фізіологічними та психологічними реакціями, адекватність яких важлива для безпеки життєдіяльності сучасної людини [12].

Дуже часто в системі життєдіяльності людини з'являються шкідливі та небезпечні чинники, які діють на людину. У систему вводиться захист людини – система охорони праці. В наш час актуальним є питання не лише захисту людини від шкідливих та небезпечних чинників виробництва та навколишнього середовища, а й захисту навколишнього середовища від впливу людини та виробництва. На цю систему діють у визначених умовах чинники навколишнього середовища. Система мусить в цих умовах стійко функціонувати та забезпечувати захист людини.

Ситуація, в котрій з'являється можливість виникнення нещасного випадку, вважається небезпечною, або аварійною. Аварія – подія в технічній системі, що не супроводжується загибеллю людей, при якій відновлення технічних засобів неможливе або економічно недоцільне. Коли аварія призводить до пошкоджень техніки, вона завдає збитків лише в економічному та моральному плані. Якщо ж аварія супроводжується тілесними ушкодженнями людей, то в цьому випадку йдеться про пов'язані з нею нещасні випадки. Якщо ж порушення в роботі техніки призводять до нещасних випадків, які спричинюють загибель хоча б однієї людини, то така ситуація класифікується вже як катастрофа.

Людина, її життя та здоров'я – найбільша цінність суспільства. Проте з кожним роком збільшується кількість небезпечних і шкідливих чинників, які

негативно впливають на організм людини, погіршують стан її здоров'я, зменшують тривалість життя.

Середовище в якому живе людина складається з трьох компонентів: природного, техногенного та соціального, або соціально-політичного середовищ. Природне середовище – це компонент життєвого середовища, утворений об'єктами природного походження і створеними ними екологічними системами. До природного середовища належать передусім біосфера (сфера існування живих організмів), що охоплює частину атмосфери, гідросфери і верхню частину літосфери, які взаємопов'язані складними біогеохімічними циклами міграції речовин і енергії. У природному середовищі діють переважно закони природного розвитку: фізичні, хімічні, геологічні та біологічні. До природного середовища належать також космічні об'єкти: Сонце, Місяць, комети, астероїди [12, 13].

Структура життєдіяльності – взаємозв'язок життєдіяльності, яка становить систему захисту, з навколишнім середовищем (див. рисунок 4.1).

Рисунок 4.1 – Структура життєдіяльності

Навколишнє середовище – це вся сукупність природних, антропогенних та соціокультурних факторів, у середовищі яких живе людина. Біосфера – це сфера

існування (функціонування) живих організмів на Землі, що включає (охоплює, містить, вміщує, має) частину атмосфери, літосфери та гідросфери. Верхня межа біосфери сягає 85 км, нижня – 2 км у літосфері та до 11 км у гідросфері. Техносфера – це частина біосфери в минулому, перетворена людиною за допомогою прямого або непрямого впливу технічних засобів з метою найкращої відповідності своїм (певним, власним) матеріальним і соціально-економічним потребам. Ноосфера (сфера розуму, буквально "оболонка, яка мислить") – фаза розвитку біосфери, де розумна діяльність людства стає визначальним фактором її функціонування [13].

Системний аналіз у безпеці життєдіяльності – це методологічні засоби, що використовуються для визначення небезпек, які виникають у системі "людина – життєве середовище". Хоча основним об'єктом вивчення безпеки життєдіяльності людини є зв'язки в системі "людина – життєве середовище", у центрі уваги перебуває людина, як самоціль розвитку суспільства. Об'єктом системи «людина – життєве середовище» може бути як окрема людина, так і будь-яка спільнота, членом якої є ця людина. Є різні рівні цієї системи: мікроколектив студентів, макроколектив мікрорайону.

Для систем будь-якого рівня необхідно визначати, які люди чи спільнота є внутрішніми складовими системи "людина – життєве середовище", а які є елементами соціального середовища, що оточує цю спільноту. В основі системного аналізу лежить здатність системи мати такі властивості, яких немає і навіть не може бути у елементів, що складають її. Це необхідно усвідомлювати та пам'ятати при вирішенні конкретних завдань з безпеки життєдіяльності людини.

4.2 Вимоги до безпеки виробничого обладнання та технологічних процесів

Безпека виробничого процесу – здатність виробничого процесу відповідати вимогам безпеки праці під час його проведення в умовах, встановлених

нормативно-технічною документацією. Безпека виробничого устаткування – здатність устаткування зберігати безпечний стан при виконанні заданих функцій у певних умовах протягом встановленого часу [14].

Безпека виробничого обладнання забезпечується:

- вибором принципів дії, джерел енергії, параметрів робочих процесів;
- мінімізацією енергії, що споживається чи накопичується;
- застосуванням вмонтованих в конструкцію засобів захисту та інформації про можливі небезпечні ситуації;
- застосуванням засобів автоматизації, дистанційного керування та контролю;
- дотримання ергономічних, обмеженням фізичних і нервово психологічних навантажень працівників.

Виробниче обладнання повинно забезпечувати безпеку працюючих при монтажі (демонтажі), введенні в експлуатацію та експлуатації як у випадку автономного використання, так і у складі технологічних комплексів при дотриманні вимог (умов, правил), передбачених експлуатаційною документацією.

Згідно з Вимогами безпеки та захисту здоров’я під час використання виробничого обладнання працівниками (НПАОП 0.00-7.14-17) встановлені мінімальні вимоги щодо безпеки та захисту здоров’я під час використання виробничого обладнання працівниками у процесі трудової діяльності. Ці Вимоги поширюються на всіх суб’єктів господарювання, незалежно від форм власності, організаційно-правових форм і видів діяльності. Вказані Вимоги не обмежують право роботодавця встановлювати більш жорсткі вимоги щодо безпеки і захисту здоров’я та життя своїх працівників під час використання ними виробничого обладнання, якщо це не суперечить законодавству [14].

Безпека виробничих процесів та устаткування обумовлює рівень професійних захворювань та виробничого травматизму в галузі машинобудування. На підприємствах машинобудівного профілю використовуються різ- ні види технологічного обладнання, яке застосовується у певних технологічних процесах. Повністю безпечних виробництв не існує, тому важливо надати студентам

теоретичні знання та практичні навички щодо питань безпеки виробничого обладнання та технологічних процесів у галузі машинобудування [15].

Виробниче обладнання, робота якого супроводжується виділенням шкідливих речовин чи мікроорганізмів або пожежо- та вибухонебезпечних речовин, повинно включати вмонтовані пристрої для локалізації цих виділень. При відсутності таких пристроїв, в конструкції обладнання мають бути передбачені місця для підключення автономних пристроїв локалізації виділень. За необхідності згадані пристрої мають бути виконані з урахуванням чинних вимог щодо стану повітря робочої зони та захисту довкілля.

Якщо виробниче обладнання являється джерелом шуму, ультра та інфразвуку, вібрації, виробничих випромінювань (електромагнітних, лазерних тощо), то воно повинно бути виконано таким чином, щоб дія на працюючих перерахованих шкідливих виробничих факторів не перевищувала меж, встановлених відповідними чинними нормативами [16].

Основними вимогами безпеки до технологічних процесів є: усунення безпосереднього контакту працюючих з вихідними матеріалами, заготовками, напівфабрикатами, готовою продукцією та відходами виробництва, що є вірогідними чинниками небезпек; заміна технологічних процесів та операцій, що пов’язані з виникненням небезпечних та шкідливих виробничих факторів, процесами і операціями, за яких зазначені фактори відсутні або характеризуються меншою інтенсивністю; комплексна механізація та автоматизація виробництва, застосування дистанційного керування технологічними процесами і операціями при наявності небезпечних та шкідливих виробничих факторів; герметизація обладнання; застосування засобів колективного захисту працюючих.

При визначенні необхідних засобів захисту потрібно керуватися вказівками відповідних розділів стандартів по видах виробничих процесів та групах виробничого обладнання, що використовуються у цих процесах. Перелік діючих стандартів стосовно процесів дається у покажчиках Держстандарту, що видаються кожен рік.

ВИСНОВКИ

В даній кваліфікаційній роботі була розроблена система для автоматизованого виявлення уражень зору на зображеннях флюоресцентної ангіографії ока з використанням глибокого навчання та алгоритмів комп'ютерного зору. Основними етапами розробки системи були: підготовка та обробка даних, моделювання та тренування моделі YOLO, інтеграція моделі з веб-додатком та його подальше розгортання на платформі Streamlit Community Cloud. Для досягнення поставленої мети використовувались такі інструменти та технології: мова програмування Python, бібліотеки Ultralytics для роботи з YOLO, Streamlit для побудови графічного інтерфейсу та Selenium для автоматизованого тестування. Веб-додаток спрямований на надання необхідної допомоги у виявленні уражень на зображеннях флюоресцентної ангіографії, не потребуючи високих обчислювальних ресурсів, значних накопичень даних або поглиблених знань у галузі машинного навчання.

Розроблена система успішно виконує автоматизоване виявлення уражень зору на зображеннях флюоресцентної ангіографії, забезпечуючи точність та швидкість аналізу. Застосунок "FluoroVision" демонструє значний потенціал для раннього виявлення хронічних очних захворювань, що підтверджено емпіричними тестами, проведеними на публічно доступних наборах даних. Зокрема, веб-додаток пропонує надійність і простоту впровадження в реальних клінічних умовах, що є суттєвою перевагою. Важливо, що створена програма здатна виявляти дрібні ураження, які можуть бути пропущені при ручному огляді. Крім того розроблена програма сприяє зменшенню часу, необхідного для ручного виявлення, тим самим прискорюючи процес діагностики.

Було проведено навчання моделі YOLO на підготовлених даних, що дозволило досягти високої точності ідентифікації уражень. Інтерфейс веб-додатку, створений за допомогою бібліотеки Streamlit, забезпечує зручність взаємодії користувача з системою, дозволяючи легко налаштовувати параметри аналізу та

переглядати результати. Додаток був успішно розгорнутий на платформі Streamlit Community Cloud, що спрощує доступ до нього з будь-якого пристрою, підключеного до мережі Інтернет.

Розроблена система має значний потенціал для використання в офтальмологічній практиці, забезпечуючи автоматизовану та точну діагностику уражень зору. Це дозволяє значно зменшити час, необхідний для аналізу зображень флюоресцентної ангіографії, що є важливим фактором для оперативної діагностики та планування лікування пацієнтів. Система має потенціал для інтеграції у медичні інформаційні системи та використовуватись офтальмологами та іншими медичними спеціалістами для покращення якості діагностики.

Використання автоматизованої системи виявлення уражень зору може значно зменшити витрати на проведення діагностичних процедур, підвищуючи їх ефективність та точність. Це дозволить медичним установам скоротити час на обробку зображень та зосередитися на інших важливих аспектах лікування пацієнтів. Зменшення ручної праці та підвищення точності діагностики сприятиме покращенню якості медичних послуг та зменшенню ймовірності помилкових діагнозів.

Для подальшого покращення системи можна розглянути можливість розширення набору даних для навчання моделі, включаючи різноманітні типи уражень та пацієнтів з різних регіонів. Це дозволить підвищити точність і ефективність моделі. Також варто дослідити можливість інтеграції системи з іншими медичними діагностичними інструментами, що дозволить створити комплексні рішення для діагностики та лікування офтальмологічних захворювань. Додатково, можна розглянути використання інших алгоритмів глибокого навчання та методів обробки зображень для покращення результатів аналізу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Fluorescein Angiography [Електронний ресурс] // The University of British Columbia – Режим доступу до ресурсу: https://shorturl.at/XNtVu.

2. Donald S. Fong. Retinopathy in Diabetes [Електронний ресурс] / Donald S. Fong // American Diabetes Association. – 2004. – Режим доступу до ресурсу: https://diabetesjournals.org/care/article/27/suppl_1/s84/24669/Retinopathy-in-Diabetes.

3. IDF Diabetes Atlas: Global estimates for the prevalence of diabetes for 2015 and 2040 [Електронний ресурс] / K. Ogurtsova, J.D. da Rocha Fernandes, Y. Huang та ін. // Diabetes Research and Clinical Practice. – 2017. – Режим доступу до ресурсу: https://www.sciencedirect.com/science/article/abs/pii/S0168822717303753.

4. Daniel Shu Wei Ting. Development and Validation of a Deep Learning System for Diabetic Retinopathy and Related Eye Diseases Using Retinal Images From Multiethnic Populations With Diabetes [Електронний ресурс] / Daniel Shu Wei Ting, Carol Yim-Lui Cheung, Gilbert Lim. – 2017. – Режим доступу до ресурсу: https://jamanetwork.com/journals/jama/fullarticle/2665775.

5. Automated Diabetic Retinopathy Image Assessment Software: Diagnostic Accuracy and Cost-Effectiveness Compared with Human Graders [Електронний ресурс] / Adnan Tufail FRCOphth, Caroline Rudisill PhD, Catherine Egan FRANZCO та ін.] // 2017 – Режим доступу до ресурсу: https://shorturl.at/Weoxo.

6. Optical Coherence Tomography, Fluorescein Angiography, And Diagnosis Of Choroidal Neovascularization In Age-Related Macular Degeneration [Електронний ресурс] / Vincent Gualino, Ramin Tadayoni, Salomon Yves Cohen та ін.] // PubMed Central. – 2018. – Режим доступу до ресурсу: https://shorturl.at/jhDfx.

7. Geethapriya. S. Real-Time Object Detection with Yolo [Електронний ресурс] / Geethapriya. S, N. Duraimurugan, S.P. Chokkalingam // International Journal of Engineering and Advanced Technology (IJEAT). – 2019. – Режим доступу до ресурсу: https://www.ijeat.org/wp-content/uploads/papers/v8i3S/C11240283S19.pdf.

8. Use cases | VPAF Project Management Office [Електронний ресурс] // University of Waterloo – Режим доступу до ресурсу: https://shorturl.at/M0hML.

9. Hussein Mohammed Ali. A Comprehensive Study on Automated Testing with the Software Lifecycle [Електронний ресурс] / Hussein Mohammed Ali, Mahmood Yashar Hamza, Tarik Ahmed Rashid – Режим доступу до ресурсу: https://arxiv.org/pdf/2405.01608.

10. Web Automated Testing using Selenium IDE [Електронний ресурс] // InterWorks. – 2017. – Режим доступу до ресурсу: https://iwconnect.com/wp-content/uploads/2021/01/Web-Automated-Testing-using-Selenium-IDE.pdf.

11. Huang, G.Q., Mak, K.L. (2003). Development and Deployment of Web Applications. In: Internet Applications in Product Design and Manufacturing. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-55778-1_4.

12. Іщейкіна Ю . О., Філатова В. Л., Нечепасва Л. В., Буря Л. В., Саргош О. Д., Бичко М. В. Основи безпеки життєдіяльності людини. Навчальний посібник. – Полтава: Вид-во ПО “ШвидкоДрук”, 2013. – 113 с.

13. Безпека життєдіяльності та цивільний захист [Електронний ресурс]: підручник для студ. спеціальностей з природничих, соціально-гуманітарних наук та інженерно-комунікаційних технологій / О. Г. Левченко, О. В. Землянська, Н. А. Праховнік, В. В. Зацарний; КПІ ім. Ігоря Сікорського. – Електронні текстові данні (1 файл: 10,2 Мбайт). – Київ: КПІ ім. Ігоря Сікорського, 2019. – 267 с.

14. Безпека виробничих процесів і устаткування : навчальний посібник для студентів спеціальності 263 – Цивільна безпека, освітня програма – Охорона праці. Частина І. Організаційні та технічні заходи безпеки трудового процесу / І. О. Мезенцева. – НТУ “ХПІ”, Харків, 2022. − 246 с.

15. Методичні вказівки до виконання самостійної роботи по дисципліні «Безпека виробничих процесів і устаткування» для студентів спеціальності 263 «Цивільна безпека» денної і заочної форми навчання / Уклад. І. О. Мезенцева, І.М. Любченко. – Харків : НТУ «ХПІ», 2018. – 21 с.

16. ДСТУ EN 981:2014. Безпечність машин. Системи звукових і візуальних сигналів небезпеки та попередження (EN 981:1996 + A1:2008, IDT). – Введ. 01.10.2019.

ДОДАТКИ

ДОДАТОК А – Лістинг коду розробленого додатку

Лістинг коду 1 – Код розробленої програми

```
import cv2
import numpy as np
import pandas as pd
import streamlit as st
import matplotlib.colors as mcolors
from PIL import Image
from config import CLASSES1
from config import CLASSES2
from ultralytics import YOLO
from pathlib import Path
from io import BytesIO
import zipfile

# HTML код для заголовка сторінки Streamlit
html = """
    <div
        style = "
            background-color:#FF5733;
            padding:18px;
            border-radius: 5px;"
    >
        <h1 style = "color:#FFFFF0; text-align:center">
            Виявлення уражень зору
        </h1>
    </div>
"""

# Налаштування сторінки Streamlit
st.set_page_config(
    page_title="FluoroVision",
)

# Відображення заголовка сторінки
st.markdown(html, unsafe_allow_html=True)

# Кешування моделі YOLO для повторного використання
@st.cache(max_entries=2, allow_output_mutation=True)
def get_yolo8(weights):
    return YOLO(str(weights))

# Кешування результатів прогнозування для зображень
@st.cache(max_entries=10)
def get_preds(img, imgsz):
    if weights == "yolov8.pt":
        if not all_classes:
            model.classes = classes
```

```
        model.conf = conf_thres
        model.iou = iou_thres
        results = model([img])
        boxes = results[0].boxes
        result = boxes.data  # повертає один бокс
        return result.numpy()
    print("hi", result.xyxy[0])
    return result.xyxy[0].numpy()

# Функція для отримання кольорів для відображення класів
def get_colors(indexes):
    to_255 = lambda c: int(c * 255)
    tab_colors = list(mcolors.TABLEAU_COLORS.values())
    tab_colors =
        [list(map(to_255, mcolors.to_rgb(name_color))) for
        name_color in tab_colors]
    base_colors = list(mcolors.BASE_COLORS.values())
    base_colors =
        [list(map(to_255, name_color)) for
        name_color in base_colors]
    rgb_colors = tab_colors + base_colors
    rgb_colors = rgb_colors * 5

    color_dict = {}
    for i, index in enumerate(indexes):
        if i < len(rgb_colors):
            color_dict[index] = rgb_colors[i]
        else:
            color_dict[index] = (255, 0, 0)

    return color_dict

# Функція для отримання кольору легенди
def get_legend_color(class_name):
    index = CLASSES.index(class_name)
    c = rgb_colors[index]
    return
        'background-color: rgb({c[0]},{c[1]},{c[2]})'.format(color=c)

# CSS для стилізації
st.markdown("""
        <style>
        .format-style{
            font-size: 20px;
            font-family: arial;
            color: red;
            margin-top: -50px;
        }
        </style>
        """,
        unsafe_allow_html=True
    )
```

```
st.markdown(
    """
    <style>.
    common-style{
        font-size:18px;
        font-family:arial;
        color:pink;
    }
    </style>
    """,
    unsafe_allow_html=True
)

# Сайдбар для налаштувань користувача
st.sidebar.markdown(
    '<p class="format-style">Параметри</p>',
    unsafe_allow_html=True
)

# Вибір моделі користувачем
modelSelect = st.sidebar.selectbox(
    'Модель',
    ('YOLOv8', 'YOLOv5', 'YOLOv3'),
    format_func=lambda a: a[:len(a)]
)

# Вибір ваг користувачем
weights = st.sidebar.selectbox(
    'Ваги',
    ('yolov8.pt',)
)

if weights == 'yolov8.pt':
    CLASSES = CLASSES1

# Вибір розміру зображення користувачем
imgsz = st.sidebar.selectbox(
    'Розмір рображення',
    (416, 512, 608, 896, 1024, 1280, 1408, 1536)
)

# Вибір порогу впевненості користувачем
conf_thres = st.sidebar.slider(
    'Поріг впевненості', 0.00, 1.00, 0.7
)

# Вибір порогу перетину об'єктів (IoU) користувачем
iou_thres = st.sidebar.slider(
    'Поріг перетину об\'єктів (IoU)', 0.00, 1.00, 0.45
)

# Вибір класів для обробки користувачем
classes = st.sidebar.multiselect(
```

```
    'Класи',
    [i for i in range(len(CLASSES1))],
    format_func=lambda index: CLASSES1[index]
)

# Вибір усіх класів для обробки
all_classes = st.sidebar.checkbox('Всі класи', value=True)

# Завантаження моделі з вибраними параметрами
with st.spinner('Завантаження моделі...'):
    if modelSelect == '':
        model = get_yolo8(weights)
    elif modelSelect == 'YOLOv8':
        model = get_yolo8(weights)

st.success('Модель "' + modelSelect + '" завантажено успішно!')

# Режим прогнозування
prediction_mode = 'Multiple images'

if all_classes:
    target_class_ids = list(range(len(CLASSES1)))
elif classes:
    target_class_ids = [class_name for class_name in classes]
else:
    target_class_ids = [0]

# Отримання кольорів для класів
rgb_colors = get_colors(target_class_ids)

detected_ids = None

# Функція для завантаження зображень
def load_images(file_uploader, imgsz):
    uploaded_files =
        file_uploader(
            "Виберіть зображення",
            type=['png', 'jpg', 'jpeg'],
            accept_multiple_files=True
        )
    image_filenames, img_collection = [], []
    for uploaded_file in uploaded_files:
        bytes_data = uploaded_file.getvalue()
        file_bytes = np.asarray(
            bytearray(bytes_data),
            dtype=np.uint8
        )
        img = cv2.imdecode(file_bytes, cv2.IMREAD_COLOR)
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        img_collection.append(img)
        image_filenames.append(uploaded_file.name)

    return [img_collection, image_filenames]
```

```
# Функція для обробки зображень
def process_images(images, target_class_ids, all_classes, weights):
    results = []
    for img in images:
        result = get_preds(img, imgsz)

        result_copy = result.copy()
        result_copy = result_copy[
            np.isin(result_copy[:, -1], target_class_ids)
        ]

        results.append(result_copy)

    return results

# Функція для малювання рамок на зображеннях
def draw_bounding_boxes(images, results, rgb_colors, CLASSES2,
conf_thres, iou_thres):
    drawn_images = []
    for img, result_copy in zip(images, results):
        img_draw = img.copy().astype(np.uint8)
        font = cv2.FONT_HERSHEY_TRIPLEX

        res = []
        detected_ids = []

        text = "Some text in a box!"
        (text_width, text_height) = cv2.getTextSize(
            text,
            font,
            fontScale=0.5,
            thickness=2
        )[0]

        for bbox_data in result_copy:
            xmin, ymin, xmax, ymax, con, label = bbox_data

            if con >= conf_thres:
                con = round(con, 4)
                p0, p1, label =
                    (int(xmin), int(ymin)),
                    (int(xmax), int(ymax)),
                    int(label)

                img_draw = cv2.rectangle(
                    img_draw,
                    p0,
                    p1,
                    rgb_colors[label],
                    2
                )
                label2 = CLASSES1.index(str(label))
                print(str(label), CLASSES2[label2])
```

```
                res.append(str(label) + ' ' + CLASSES2[label2])

                box_coords = ((int(xmin) - 1, int(ymin)),
                              (int(xmin) + text_width - 80,
                             (int(ymin) - 5) - text_height))
                img_draw = cv2.rectangle(
                    img_draw,
                    box_coords[0],
                    box_coords[1],
                    rgb_colors[label],
                    cv2.FILLED
                )

                img_draw = cv2.putText(
                    img_draw,
                    str(label),
                    (int(xmin), int(ymin) - 5),
                    font, 0.5,
                    (255, 255, 255), 1
                )
                img_draw = cv2.putText(
                    img_draw,
                    ', ' + str(con),
                    (int(xmin) + 30, int(ymin) - 5),
                    font, 0.5,
                    (255, 255, 255), 1)

                detected_ids.append(label)

        drawn_images.append((res, img_draw))

    return drawn_images

# Функція для відображення результатів
def display_results(drawn_images):
    for i, (res, img_draw) in enumerate(drawn_images):
        col1, col2 = st.columns(2)
        with col1:
            st.write(res)
            download_single_image(drawn_images[i], file_names[i])
        with col2:
            st.image(img_draw, use_column_width=True)

    download_zip(drawn_images, file_names)

# Функція для завантаження зображень у zip-архіві
def download_zip(drawn_images, file_names):
    with BytesIO() as zip_buffer:
        with zipfile.ZipFile(zip_buffer, 'a') as zip_file:
            for i, (res, img_draw) in enumerate(drawn_images):
                image_bytes = image_to_bytes(img_draw)
                file_name = f"{file_names[i].rsplit('.', 1)[0]}.png"
                zip_file.writestr(file_name, image_bytes)
```

```
        zip_buffer.seek(0)
        st.download_button(
            label="Завантажити всі зображення",
            data=zip_buffer,
            file_name='images.zip',
            key='download_all'
        )

# Функція для завантаження окремого зображення
def download_single_image(drawn_image, file_name):
    _, img_draw = drawn_image
    image_bytes = image_to_bytes(img_draw)
    file_name = f"{file_name.rsplit('.', 1)[0]}.png"
    st.download_button(
        label="Завантажити дане зображення",
        data=image_bytes,
        file_name=file_name,
        key='download_single'
    )

# Функція для перетворення зображення у байти
def image_to_bytes(image):
    pil_image = Image.fromarray(image)
    buffer = BytesIO()
    pil_image.save(buffer, format='PNG')
    return buffer.getvalue()

# Виконання завантаження та обробки зображень
if prediction_mode == 'Multiple images':
    images, file_names = load_images(st.file_uploader, imgsz)

    if images:
        results = process_images(
            images,
            target_class_ids,
            all_classes,
            weights
        )
        drawn_images = draw_bounding_boxes(
            images,
            results,
            rgb_colors,
            CLASSES2,
            conf_thres,
            iou_thres
        )

        display_results(drawn_images)
```

ДОДАТОК Б – Диск із кваліфікаційною роботою бакалавра