МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра інтелектуальних технологій

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

НА ТЕМУ:

Нейромережний застосунок розпізнавання захворювань рослин

Галузь знань **12 «Інформаційні технології»**

Спеціальність **122 «Комп'ютерні науки»**

Освітня програма «**Комп'ютерні науки»**

Освітній рівень: бакалавр

Виконав: студент 4 курсу, групи КН-42

Каліта В.І._____
(прізвище та ініціали)

Керівник _Гнатієнко Г.М._____
(прізвище та ініціали)

к.т.н., заступник декана факультету_____
інформаційних технологій з наукової роботи
(науковий ступінь, звання)

Випускна кваліфікаційна робота бакалавра допущена до захисту
рішенням кафедри *інтелектуальних технологій*
Протокол № 13 від 05.06.2023 р.
зав. кафедри_____ доц. Іларіонов О.Є.

Київ – 2023

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій
Кафедра інтелектуальних технологій
Спеціальність 122 «Комп'ютерні науки»

ЗАТВЕРДЖУЮ
Завідувач кафедри
інтелектуальних технологій
Іларіонов О.Є.

_____

"___"_____2023 р.

ЗАВДАННЯ
НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Каліті Вячеславу Ігоровичу

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи)

Нейромережний застосунок розпізнавання захворювань рослин

затверджена протоколом засідання кафедри від «11» жовтня 2022 р. №4

2. Термін здачі студентом закінченого проєкту (роботи)

3. Вихідні дані до проєкту (роботи)

Нейромережний застосунок, натренований на відповідному наборі даних, який може розпізнати та оцінити стан рослини за зображенням

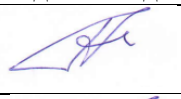
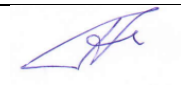
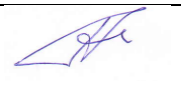
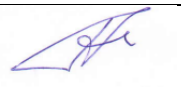
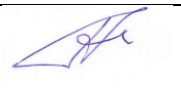
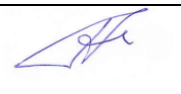
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)

Вибір та аналіз предметної області розпізнавання захворювань рослин, постановка задачі з розробки нейромережного застосунку для розпізнавання захворювань рослин, порівняння існуючих систем, що використовують машинен навчання в агротехнологіях, проектування архітектури системи: опис структури та основних компонентів нейромережного застосунку, огляд обраних засобів для розробки системи та обгрунтування вибору конкретних технологій та інструментів для розробки системи, розробка інтерфейсу користувача для застосунку, що створюється, вибір тестових задач для перевірки ефективності системи, опис вибраних сценаріїв та вхідних даних для тестування, тестування роботи системи, оцінка працездатності та ефективності системи на основі результатів тестування

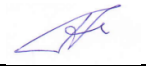
5. Перелік презентаційного матеріалу (з точним зазначенням обов'язкових презентацій)

Вступ, аналіз проблем предметної області, опис навчального набору даних, вибір моделі нейронної мережі, аналіз варіантів використання додатку, структурно-функціональний аналіз процесів в інформаційній системі, опис інформаційної архітектури застосунку, реалізація інтерфейсу та тестування застосунку, висновки

6. Консультанти з випускної кваліфікаційної роботи із зазначенням розділів випускної кваліфікаційної роботи, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Вибір та аналіз предметної області, проведення початкового дослідження та вивчення літератури.		
1	Аналітичний огляд функціонування та постановка задачі створення нейромережного застосунку розпізнавання захворювань рослин		
2	Проектування архітектури нейромережного застосунку		
2	Вибір інструментів для реалізації застосунку		
3	Розробка інтерфейсу користувача та програмна реалізація застосунку розпізнавання захворювань рослин		
3	Вибір тестових прикладів, тестування роботи системи		

7. Дата видачі завдання 20 лютого 2023 року

Керівник  / Гнатієнко Г.М. /
(підпис) (ПІБ)

Завдання прийняв до виконання  / Каліта В.І. /
(підпис) (ПІБ)

КАЛЕНДАРНИЙ ПЛАН

Пор. №	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів випускної кваліфікаційної роботи	Примітка
1	Вибір та аналіз предметної області, проведення початкового дослідження та вивчення літератури.	20.02 – 03.03	
2	Аналітичний огляд функціонування та постановка задачі створення нейромережного застосунку розпізнавання захворювань рослин	04.03 – 18.03	
3	Проектування архітектури нейромережного застосунку	19.03 – 10.04	
4	Вибір інструментів для реалізації застосунку	11.04 – 15.04	
5	Розробка інтерфейсу користувача та програмна реалізація застосунку розпізнавання захворювань рослин	16.04 – 8.05	
6	Вибір тестових прикладів, тестування роботи системи	9.05 – 17.05	
7	Підготовка та оформлення пояснювальної записки	18.05 –25.05	

Студент-дипломник  / Каліта В.І. /
(підпис) (ПІБ)

Керівник випускної кваліфікаційної роботи  / Гнатієнко Г.М. /
(підпис) (ПІБ)

Анотація

Каліта Вячеслав Ігорович виконав випускну кваліфікаційну роботу на тему "Нейромережний застосунок розпізнавання захворювань рослин" за спеціальністю 122 – "Комп'ютерні науки".

У випускній кваліфікаційній роботі проведено аналіз сучасних методів розпізнавання зображень і діагностики захворювань рослин, використовуючи глибинні нейронні мережі. Розглянуто процес збору та підготовки даних для тренування моделі. Розроблено архітектуру моделі регіональної згорткової нейронної мережі для визначення стану здоров'я рослин. Розроблено веб-додаток, який інтегрує модель глибокого навчання для діагностики захворювань рослин користувачами.

Ключові слова: глибоке навчання, розпізнавання об’єктів, діагностика захворювань рослин, згорткові нейронні мережі.

Summary

Kalita Vyacheslav Ihorovich completed the graduation qualification work on the topic "Neural network application for recognition of plant diseases" in the specialty 122 - "Computer Sciences".

In the graduation qualification work, an analysis of modern methods of image recognition and diagnostics of plant diseases using deep neural networks was carried out. The process of collecting and preparing data for training the model was considered. The architecture of the regional convolutional neural network model for determining the state of plant health was developed. A web application has been developed that integrates a deep learning model for diagnosing plant diseases by users.

Keywords: deep learning, image recognition, plant disease diagnosis, convolutional neural networks.

Перелік умовних позначень і скорочень

БД – база даних

CNN (convolutional neural network) – згорткова нейронна мережа

R-CNN (regional convolutional neural network) – регіональна згорткова нейронна мережа

ROI (region of interest) – регіон інтересу

RPN (region proposal network) – регіональна мережа пропозицій

FPN (feature pyramid network) – мережа піраміди ознак

IoU (intersection over union) – перетин через об’єднання

R (recall) – повнота

P (precision) – точність

AP (average precision) – середня точність

ЗМІСТ

ВСТУП

Виявлення та ідентифікація захворювань і шкідників рослин є надзвичайно важливою областю досліджень в сфері машинного зору. Ця технологія впроваджує використання спеціалізованого обладнання для збору зображень і визначення наявності хвороб та шкідників на зібраних фотографіях рослин. Сучасне обладнання для виявлення захворювань і шкідників рослин, яке використовує машинний зір, вже активно використовується в аграрному секторі, де воно поступово замінює традиційні методи ідентифікації, засновані на спостереженнях людського ока. Це актуальна проблема, оскільки затримка в виявленні захворювань рослин може привести до суттєвих втрат урожаю, що може негативно позначитися на економіці країни. Імплементація глибоких нейронних мереж для ідентифікації захворювань рослин значно спрощує та пришвидшує цей процес. Висока точність виявлення захворювань забезпечує точну ідентифікацію, що допомагає мінімізувати витрати на обрання оптимального методу лікування та боротьби зі шкідниками.

Метою цієї роботи є створення нейромережного застосунку, що здатен автоматично розпізнавати та класифікувати захворювання на листях агрокультур в межах обраного набору даних, а також надавати рекомендації щодо подальшого вирішення виявлених проблем.

Об'єктом дослідження є процеси виявлення, ідентифікації та класифікації захворювань з використанням машинного зору та глибоких нейронних мереж.

Предметом дослідження є методи та технології машинного зору, глибокого навчання, зокрема застосування регіональних згорткових нейронних мереж для виявлення та ідентифікації захворювань на зображеннях рослин.

Завданнями цього дослідження є:

- Аналіз проблеми виявлення захворювань в листях аграрних культур.

- Дослідження та вивчення сучасних методів машинного зору та глибокого навчання, які застосовуються для виявлення та ідентифікації захворювань рослин.
- Проведення функціонального аналізу та визначення основних вимог до створюваної системи.
- Обгрунтування вибору інструментальних засобів для програмної реалізації застосунку.
- Вибір і побудова оптимальної моделі глибокого навчання для розпізнавання захворювань та виведення рекомендацій щодо їх усунення.
- Розробка веб-додатку, який використовує вибрану модель для автоматичного виявлення та класифікації захворювань на листях рослин.
- Тестування та оцінка розробленого додатку на реальних даних та його вдосконалення.

Ініціалізація цього проекту була обумовлена потребою в сучасних рішеннях для швидкого та точного виявлення захворювань рослин, що допоможе мінімізувати втрати врожаю та зменшити витрати на лікування.

В результаті проекту очікується створення нейромережного застосунку, що здатний автоматично розпізнавати та класифікувати захворювання на листях агрокультур, а також надавати рекомендації щодо подальшого вирішення виявлених проблем. Це покращить ефективність сільського господарства, зменшивши втрати врожаю внаслідок захворювань рослин.

РОЗДІЛ 1. АНАЛІТИЧНИЙ ОГЛЯД ФУНКЦІОНУВАННЯ НЕЙРОМЕРЕЖНОГО ЗАСТОСУНКУ РОЗПІЗНАВАННЯ ЗАХВОРЮВАНЬ РОСЛИН

1.1. Аналіз проблем предметної області застосування додатку розпізнавання захворювань рослин.

Сільське господарство відіграє ключову роль у світовій економіці та продовольчій безпеці, проте постійно зіштовхується з рядом проблем. Центральне місце серед них займають захворювання рослин, які можуть призвести до значних втрат врожаю. Захворювання рослин - це відхилення від нормального стану рослини, яке порушує або змінює її життєві функції. За оцінками дослідників, втрати врожаю внаслідок захворювань рослин сягають від 20 до 40% на глобальному рівні. Як наслідок, вивчення захворювань рослин і розробка методів їх діагностики та боротьби з ними є важливим напрямком досліджень у галузі патології рослин [1].

Правильна ідентифікація захворювань рослин має вирішальне значення для ефективних заходів боротьби, оскільки без неї зусилля можуть бути неефективними і марною тратою ресурсів. Щоб зрозуміти взаємозв'язок між хворобами рослин і втратою врожаю, необхідно враховувати епідеміологію хвороби, фізіологію культури, розвиток врожаю, механізми пошкодження та вплив методів управління. Інтегруючи цю інформацію, можна покращити наше розуміння взаємозв'язку між хворобами рослин і втратами врожаю. Однак важливо зазначити, що дослідження втрат врожаю є ресурсомісткими і можуть бути складними для інтерпретації, оскільки культури рідко уражаються лише одним шкідником або патогеном одночасно.

Традиційно процес виявлення та діагностики захворювань рослин залежав від знань та досвіду людей, зайнятих в аграрній сфері. Цей процес включає визначення фізичних ознак на листках, стеблах або плодах рослин, таких як плями, деформація, зміна кольору або неправильний ріст [2]. Однак, цей підхід виявився обмеженим у своїй здатності виявляти захворювання

вчасно та ефективно, особливо в умовах великих агрокультурних полів. На відміну від традиційних методів, що включають в себе візуальний огляд рослин фахівцем або використання спеціалізованих хімічних аналізів, системи автоматичного виявлення захворювань рослин дозволяють проводити аналіз великої кількості рослин у короткі строки та із високою точністю [3].

Штучний інтелект може виявляти тонкі зміни у рослинах, що свідчать про захворювання, значно швидше та точніше, ніж людина. Сучасні технології, зокрема глибоке навчання, можуть сприяти вирішенню цієї проблеми шляхом автоматичного виявлення захворювань рослин. Комп'ютерний зір, що використовує алгоритми глибокого навчання, дозволяє обробляти зображення листків рослин і визначати ознаки, що свідчать про наявність захворювання [4]. Проте, ефективність використання глибокого навчання в боротьбі з захворюваннями рослин сильно залежить від доступності великих об'ємів даних для тренування моделей. Більш того, моделі глибокого навчання мають бути здатні виявляти різноманітність захворювань у різних умовах. Тому важливо забезпечити високу якість і різноманітність тренувальних даних, що використовуються для навчання цих моделей.

Отже, виявлення захворювань рослин є важливим аспектом сільського господарства, оскільки воно необхідне для ефективного контролю за хворобами. Алгоритми обробки зображень показали багатообіцяючі результати у виявленні захворювань. Розробка та впровадження автоматичних систем виявлення захворювань в агробізнес може допомогти в забезпеченні високих врожаїв, виявленні захворювань на ранніх стадіях та своєчасному втручанні для запобігання їх поширенню [5].

1.2. Аналіз методів виявлення захворювань рослин.

В останні роки автоматичне виявлення захворювань рослин шляхом обробки цифрових зображень та застосування методів машинного навчання набуває все більшої популярності. Цей підхід дозволяє ідентифікувати

захворювання сільськогосподарських культур з високою точністю та ефективністю.

У даному розділі проводиться детальний аналіз різних методів, що використовуються для виявлення захворювань рослин. Порівняння методів проводиться з урахуванням їхньої ефективності, точності та можливості автоматичного виявлення захворювань на основі обробки зображень.

1.2.1 Аналіз методів машинного навчання для виявлення захворювань рослин.

Машинне навчання відіграє ключову роль у виявленні та класифікації захворювань рослин. Це процес, за допомогою якого моделі навчаються ідентифікувати та класифікувати об'єкти (в нашому випадку - симптоми захворювання рослин) на основі набору тренувальних даних. Існують різні методи та алгоритми машинного навчання, які можуть бути використані в цьому контексті.

Методи машинного навчання.

Дерева рішень - метод машинного навчання, що використовує структуру дерева для прийняття рішень. Кожен вузол в дереві представляє певну ознаку (або атрибут), а кожна гілка представляє рішення, яке приймається на основі цієї ознаки. Цей метод добре підходить для класифікації захворювань рослин, тому що він може легко обробляти багато ознак і знаходити важливі відмінності між різними захворюваннями [6].

Метод опорних векторів (SVM) - це алгоритм машинного навчання, який використовується для класифікації та регресії. Він працює, знаходячи гіперплощину в багатовимірному просторі, яка найкраще розділяє різні класи даних. В контексті виявлення захворювань рослин, SVM може бути використаний для виявлення важливих шаблонів у даних, які відповідають різним захворюванням.

Метод k-найближчих сусідів (k-Nearest Neighbor) - це простий алгоритм машинного навчання, який використовується для класифікації та регресії. Він працює, порівнюючи новий об'єкт з уже відомими об'єктами в наборі даних і призначає йому той самий клас, що й найближчим до нього об'єктам. У

контексті виявлення захворювань рослин, цей метод може бути використаний для виявлення схожих симптомів захворювань у різних рослин.

Переваги методів машинного навчання:

- Обчислювальна ефективність: більшість алгоритмів машинного навчання вимагають менше обчислювальної потужності та пам'яті порівняно з глибоким навчанням, що робить їх більш доступними для малих та середніх об'ємів даних.
- Легше інтерпретувати: деякі алгоритми машинного навчання, такі як дерева рішень або лінійна регресія, надають моделі, що легко інтерпретуються, що може бути корисним при поясненні результатів.
- Ефективність з малими даними: методи машинного навчання можуть бути ефективними з меншими наборами даних, в той час як глибоке навчання зазвичай вимагає великих наборів даних для навчання.

Недоліки:

- Обмежена здатність вивчати складні шаблони: методи машинного навчання можуть мати труднощі з ідентифікацією складних шаблонів або нелинійних відносин, особливо у великих наборах даних.
- Попередня обробка даних: більшість методів машинного навчання вимагає попередньої обробки даних, такої як масштабування, нормалізація або вибір ознак, що може бути часомістким.

1.2.2. Аналіз методів глибокого навчання для виявлення захворювань рослин.

Якщо ми ставимо перед собою завдання не тільки класифікувати об'єкти, а також виявити та точно визначити їх місцезнаходження, або ж провести розпізнавання об'єктів в режимі реального часу, то традиційні методи машинного навчання вже не будуть ефективними в цьому випадку. В таких ситуаціях необхідно звернутися до використання глибоких нейронних мереж,

що включають операцію згортки. В даній дипломній роботі як раз будуть розглядатися такі методи.

Концепція глибокого навчання (Deep Learning, DL) виникла з статті, опублікованої в журналі Science Хінтоном та ін. у 2006 році [7]. Основна ідея глибокого навчання полягає в тому, що, використовуючи нейронну мережу для аналізу даних і вивчення ознак, ознаки даних витягуються за допомогою декількох прихованих шарів, кожен прихований шар можна розглядати як перцептрон, перцептрон використовується для вилучення низькорівневих ознак, а потім об'єднує низькорівневі ознаки для отримання абстрактних високорівневих ознак, що може значно полегшити проблему локального мінімуму. Глибоке навчання долає той недолік, що традиційні алгоритми покладаються на штучно створені ознаки, і привертає все більше уваги дослідників. Зараз цей метод успішно застосовується в комп'ютерному зорі, розпізнаванні образів, розпізнаванні мови, обробці природної мови та рекомендаційних системах [8]. Традиційні методи класифікації зображень та виявлення ознак розроблені для виявлення лише базових характеристик, і вони можуть зіткнутися з труднощами при спробах виявити більш глибокі та складні аспекти зображення. Тут методи глибокого навчання можуть стати вирішальними. Вони дозволяють проводити неконтрольоване навчання безпосередньо з оригінального зображення, екстрагуючи багаторівневу інформацію про характеристики зображення, таку як низькорівневі, проміжні та високорівневі семантичні характеристики.

Традиційні алгоритми виявлення захворювань рослин зазвичай використовують методи розпізнавання зображень, засновані на вручну створених ознаках, що є трудомістким процесом, що вимагає значного досвіду та інтуїції, і не здатні автоматично вчитись та витягувати ознаки з оригінального зображення. Навпроти, глибоке навчання дає змогу автоматично навчатися з великих обсягів даних без потреби в ручних маніпуляціях. Моделі глибокого навчання складаються з декількох шарів, вони володіють високою здатністю до самонавчання та представлення ознак, і

можуть автоматично екстрагувати ознаки зображення для класифікації та розпізнавання зображень. Таким чином, глибоке навчання може відіграти значну роль в області розпізнавання зображень захворювань рослин [9].

Перелічимо етапи завдання класифікації, виявлення та сегментації об'єктів у комп'ютерному зорі. На першому етапі, який відповідає завданню класифікації в комп'ютерному зорі, ставиться мета визначити, до якої категорії належить зображення (рис. 1.1). На цьому етапі ми надаємо мітку категорії, до якої належить зображення, щоб отримати інформацію про його класифікацію. Другий етап, який пов'язаний з завданням локалізації в комп'ютерному зорі, включає в себе більш точне виявлення. На цьому етапі ми не тільки визначаємо, які захворювання рослин або об'єкти присутні на зображенні, але й точно визначаємо їх місцезнаходження. Наприклад, ми можемо використовувати прямокутну рамку, щоб показати область пошкодження або зараження грибком. Третій етап, який відповідає завданню сегментації в комп'ютерному зорі, дозволяє нам розділити уражені ділянки рослин від фону піксель за пікселем. Це дозволяє отримати додаткову інформацію, таку як розміри, площа та розташування уражень, що сприяє більш детальній оцінці тяжкості захворювань рослин.

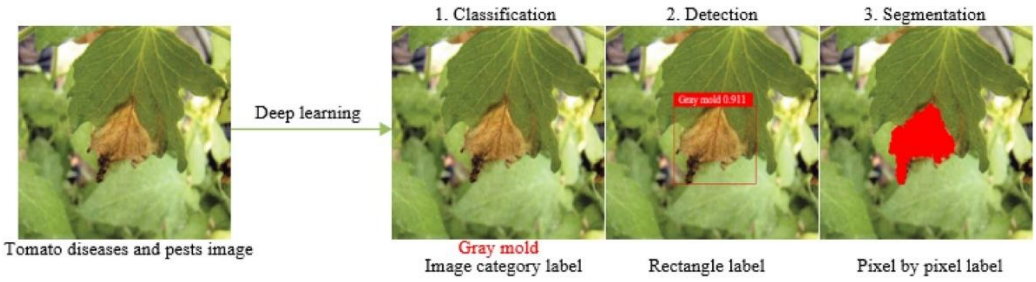


Рисунок 1.1 – Етапи виявлення захворювань за допомогою методів комп’ютерного зору

Загалом, класифікація описує зображення на загальному рівні шляхом визначення його основних ознак і вказує, чи присутній на зображенні певний тип захворювання рослин. З іншого боку, виявлення об'єктів у комп'ютерному зорі фокусується на локальному описі, вказуючи, який об'єкт зображений і

його позицію на зображенні. Таким чином, основними напрямками дослідження є вивчення ознак для класифікації об'єктів та аналіз структури для виявлення об'єктів [10, 11].

У порівнянні з іншими методами розпізнавання зображень, технологія розпізнавання зображень, заснована на глибокому навчанні, не потребує вилучення конкретних ознак, і лише за допомогою ітеративного навчання може знайти відповідні ознаки, які можуть набути глобальних і контекстних ознак зображень, а також має високу надійність і вищу точність розпізнавання.

Класифікація об’єктів.

Згорткові нейронні мережі (Convolutional Neural Networks), скорочено CNN, мають складну мережеву структуру і можуть виконувати операції згортки. CNN є популярною моделлю в галузі глибокого навчання. Причина полягає у величезній ємності моделі та складній інформації, яку несуть основні структурні характеристики CNN, що дозволяє CNN відігравати перевагу в розпізнаванні зображень. У той же час, успіхи CNN в задачах комп'ютерного зору сприяли зростанню популярності глибокого навчання. Першим шаром в CNN є вхідний шар, далі в шарі згортки (convolution layer) вибирається згорткове ядро (рис. 1.2).

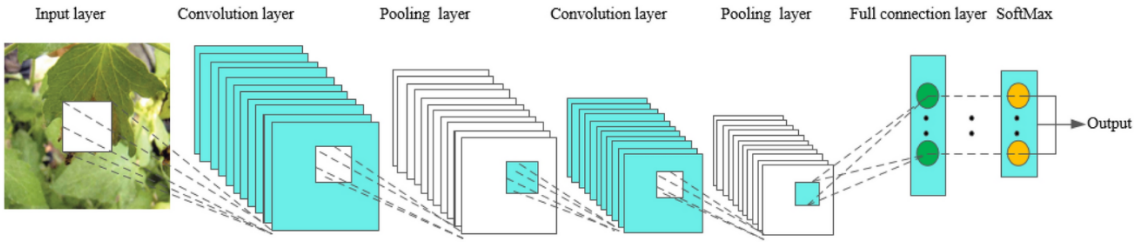


Рисунок 1.2 – Структура згорткової нейронної мережі

Це ядро функціонує як локальне сприйнятливе поле, яке є однією з ключових переваг згорткових нейронних мереж. Воно сканує мапу об'єктів, виділяючи специфічні ділянки інформації. Після цього процесу вилучення ознак, нейрони зі шару згортки передаються до шару об'єднання (pooling layer) для подальшого аналізу. Серед популярних методів об'єднання варто

відзначити обчислення середнього, максимального та випадкового значень серед всіх значень у локальному рецептивному полі. Коли дані пройшли кілька таких шарів згортки та об'єднання, вони входять в шар повного з'єднання (full connection layer), де всі нейрони повністю зв'язані з верхнім шаром. В кінцевому результаті, дані в шарі повного з'єднання можуть бути класифіковані за допомогою методу softmax і передані до вихідного шару для визначення кінцевих результатів. В реальному природному середовищі значна різноманітність у формі, розмірі, текстурі, кольорі, контексті, місцезнаходженні та освітленості зображень захворювань рослин робить процес розпізнавання вкрай складним. Згорткові нейронні мережі (CNN) з їх високою здатністю до вилучення ознак стали найбільш часто використовуваним рішенням для класифікації захворювань рослин. Сучасні мережі для класифікації захворювань рослин в основному використовують структури мережі комп'ютерного зору, такі як: AlexNet, GoogleLeNet, VGGNet, ResNet, Inception V4, DenseNet, MobileNet і SqueezeNet [12,13].

AlexNet: Ця архітектура була представлена в 2012 році на змаганні ImageNet і стала визначальною в області глибокого навчання. AlexNet має вісім шарів - п'ять згорткових і три повністю з'єднаних. Перевагою AlexNet є використання ReLU (Rectified Linear Units) як активаційних функцій, що прискорює процес навчання. Але водночас, його недоліком є велика кількість параметрів, що ускладнює процес навчання і збільшує вимоги до обчислювальних ресурсів [14].

GoogleLeNet: GoogleLeNet ввела нову архітектуру, відому як Inception, яка містить "модулі Inception" з одночасними 1x1, 3x3 та 5x5 згортками. Це зменшує кількість параметрів і дозволяє мережі бути глибшею без перенавчання. Основним недоліком GoogleLeNet є його складність в реалізації і налаштуванні [15].

VGGNet: VGGNet, представлена Oxford's Visual Geometry Group, включає в себе серію моделей, найпопулярнішими з яких є VGG16 і VGG19. Вони використовують багато шарів 3x3 згорток, що дозволяє мережі "бачити"

більше контексту. Це робить мережу глибшею при тому ж обсязі обчислень. Основним недоліком VGGNet є велика кількість параметрів, яка збільшує обсяг використовуваної пам'яті і часу навчання [16].

Inception v4: Inception v4, як і GoogleLeNet, використовує модулі Inception, але також включає в себе ідеї ResNet для покращення процесу навчання. Ця архітектура поєднує переваги обох мереж. Недоліком є складність архітектури і вимоги до обчислювальних ресурсів [17].

DenseNet: DenseNet, або Densely Connected Convolutional Networks, відрізняється тим, що кожен шар з'єднується з кожним наступним шаром. Це забезпечує високу ефективність навчання і зменшує кількість параметрів. Однак, DenseNet може бути викликом для обчислювальних ресурсів через велику кількість з'єднань [18].

MobileNet: MobileNet використовує роздільні згортки для зменшення кількості параметрів, що робить його ідеальним для мобільних застосунків. Його перевага полягає в тому, що він легший і швидший, порівняно з іншими мережами, але зберігає достатній рівень точності для багатьох завдань. Його головний недолік - він не є найкращим варіантом для дуже складних завдань [19].

SqueezeNet: SqueezeNet створена з метою надання максимальної точності з мінімальною кількістю параметрів. Це досягається за допомогою стратегії "віджимання" (зменшення глибини вхідних каналів) і "розширення" (збільшення глибини вихідних каналів). Перевагою є мала кількість параметрів, але недоліком є нижча точність в порівнянні з більш великими мережами [20].

Розпізнавання об’єктів.

Розпізнавання об'єктів є одним з завдань у галузі комп'ютерного зору, а також є найбільш прямим аналогом до виявлення захворювань рослин і шкідників у традиційному розумінні. Основна мета виявлення полягає у точному визначенні місцезнаходження та категорії об'єкта.

Регіональні згорткові нейронні мережі (R-CNN) є спеціалізованими варіаціями згорткових нейронних мереж (CNN), які спеціально розроблені для розпізнавання та локалізації об'єктів на зображеннях. Вони розширюють здатності CNN, додаючи додатковий компонент для ідентифікації регіонів, в яких можуть знаходитися об'єкти [21].

R-CNN, або Regions with Convolutional Neural Networks, це метод для визначення та локалізації об'єктів на зображеннях (рис. 1.3). Спочатку, R-CNN генерує велику кількість потенційних областей об'єкта на зображенні, використовуючи техніку селективного пошуку. Ця техніка допомагає зменшити область пошуку об'єктів. Кожен із цих регіонів інтересу (ROI - Region of Interest) потім перетворюється до розміру, придатного для обробки згортковою мережею, і використовується для вилучення ознак. За допомогою цих ознак, використовуючи повністю зв'язані шари, здійснюється класифікація та локалізація об'єктів. Класифікація визначає, який об'єкт присутній в даному ROI, а локалізація забезпечує більш точне визначення положення об'єкта в межах ROI. Варто відзначити, що існують і більш сучасні версії R-CNN, такі як Fast R-CNN та Faster R-CNN, які пропонують додаткові покращення для збільшення швидкості та ефективності.

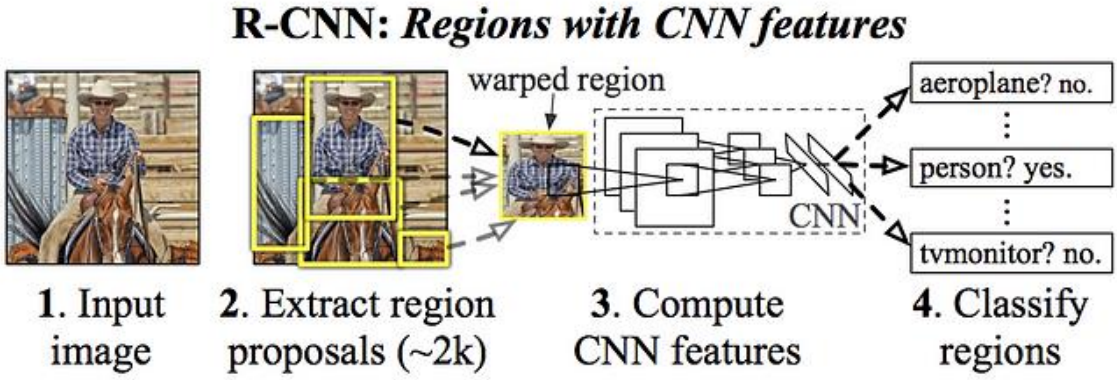


Рисунок 1.3 – Алгоритм роботи R-CNN

Fast R-CNN вносить важливе поліпшення, передавши усе зображення через CNN один раз замість проведення цього процесу для кожного ROI окремо (рис. 1.4). При роботі з Fast R-CNN, загальна ідея залишається схожою

на R-CNN, але процес оптимізовано. Натомість використовувати окремі області пропозицій, щоб пройти через CNN, у Fast R-CNN усе вхідне зображення проходить через CNN, результатом чого є згорнута карта ознак. Потім в цій карті ознак визначаються пропозиції регіонів.

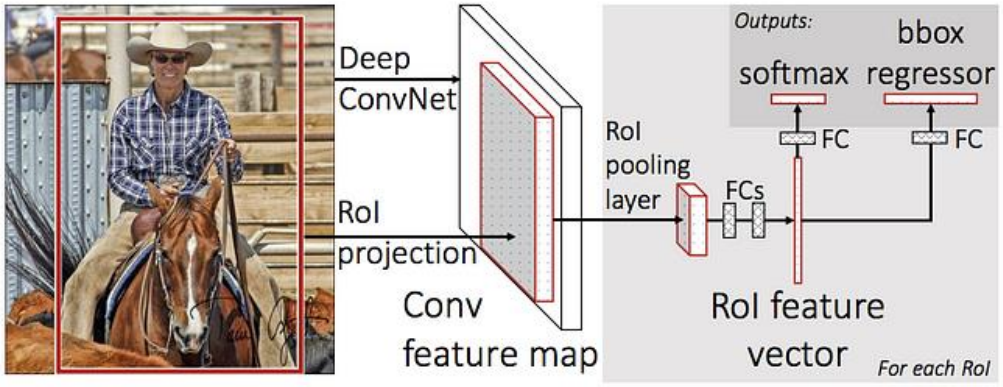


Рисунок 1.4 – Алгоритм роботи Fast R-CNN

Ці регіони потім перетворюються на квадратні області, а використовуючи шар об'єднання, ці квадратні області приводяться до фіксованого розміру, який може бути поданий на вхід повністю зв'язаного шару. По суті, Fast R-CNN використовує вектори ознак ROI для класифікації пропонованих областей за допомогою шару softmax, а також для визначення параметрів зсуву для обмежувальної рамки. Перевага Fast R-CNN порівняно з R-CNN полягає в тому, що нам не потрібно подавати 2000 областей пропозицій через CNN окремо для кожного зображення. Замість цього, ми виконуємо операцію згортки лише один раз на зображенні, згенерувавши на цій основі карту ознак.

Faster R-CNN впроваджує ще одне ключове поліпшення – регіональну мережу пропозицій (RPN - Region Proposal Network) (рис. 1.5). Замість використання алгоритму селективного пошуку для визначення ROI, Faster R-CNN використовує RPN, яка навчається визначати ці області на основі зразків зображень.

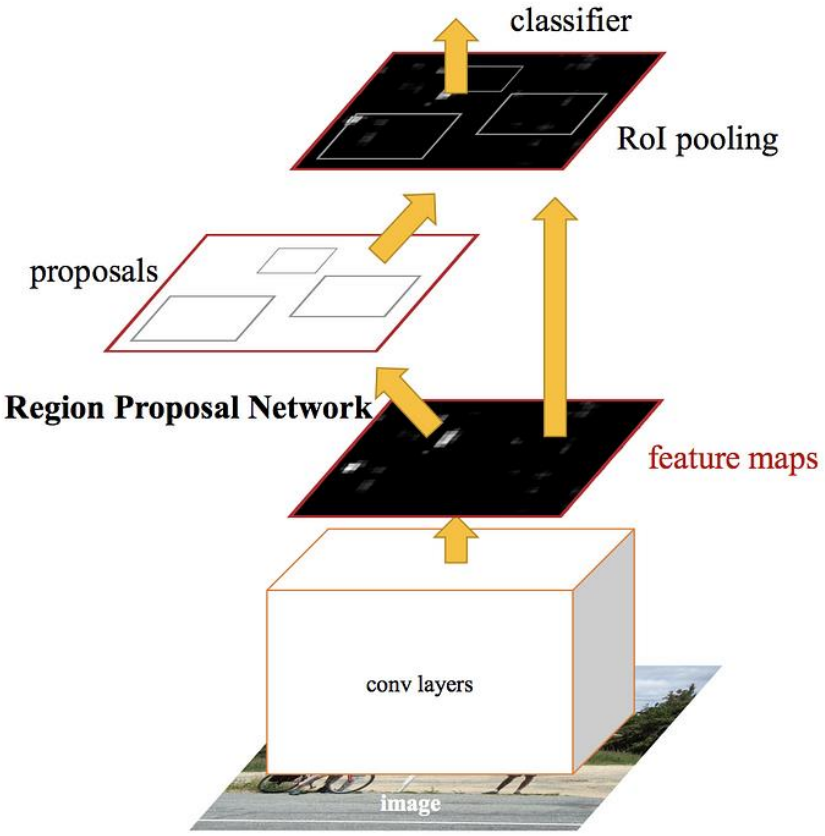


Рисунок 1.5 – Алгоритм роботи Faster R-CNN

Спрогнозовані регіони пропозицій змінюють форму за допомогою шару об'єднання ROI, який потім використовується для класифікації зображення в межах запропонованого регіону і прогнозування значень зміщення для обмежувальних рамок.Це робить процес визначення ROI більш адаптивним та ефективним.

Mask R-CNN - це метод виявлення об'єктів, що розширює архітектуру Faster R-CNN, додавши до неї третій вихідний шар для сегментації об'єктів. Цей підхід є дуже ефективним для завдань, що вимагають як виявлення об'єктів, так і їх піксельного розміщення на зображенні.

Mask R-CNN, варіант архітектури R-CNN, використовує техніку регіональної мережі пропозицій (Region Proposal Network, RPN) для визначення потенційних областей або об'єктних пропозицій на зображенні, де можуть знаходитися об'єкти (рис. 1.6).

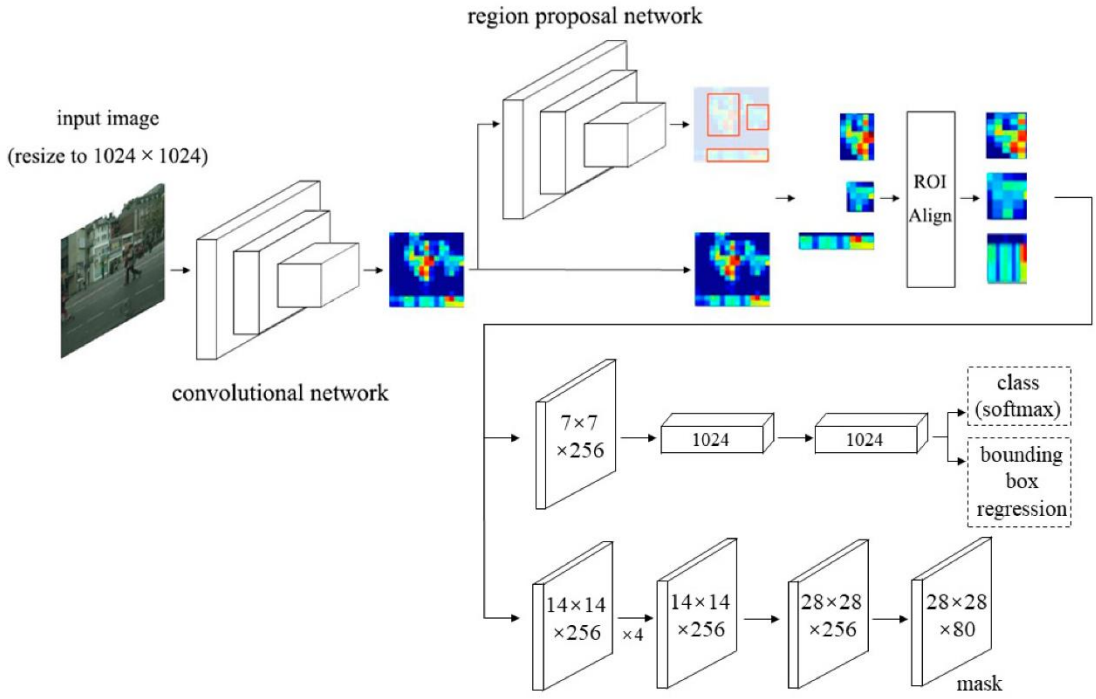


Рисунок 1.6. – Алгоритм роботи Mask R-CNN

Після того, як області визначені, згорткова нейронна мережа (CNN), часто ResNet, використовується для вираження ознак з кожного визначеного регіону. Що важливо, вихідні дані з CNN направляються на три паралельних шари. Один з шарів здійснює класифікацію об'єктів, інший коригує обмежувальні рамки об'єкта, а третій шар здійснює сегментацію об'єкта на рівні пікселів. Саме цей третій шар сегментації робить Mask R-CNN унікальним порівняно з Faster R-CNN. Нарешті, шар сегментації генерує бінарну маску для кожного об'єкта. Ці маски дають детальне представлення про місцеположення об'єктів на зображенні на рівні пікселів.

YOLO (You Only Look Once) - це популярний алгоритм виявлення об'єктів, який відрізняється від традиційних методів виявлення об'єктів. Він використовує єдину нейронну мережу для всього зображення. Ця мережа ділить зображення на сітку SxS, в кожній з яких цих сіток виділяється m обмежувальних рамок. Для кожного з них мережа виводить ймовірність класу та значення зсуву для нього. Обмеження, ймовірність класу яких перевищує порогове значення, обираються і використовуються для визначення місцезнаходження об'єкта на зображенні. Ці прогнози здійснюються

одночасно для всього зображення (рис. 1.7). Отже, в якості назви алгоритму - YOLO ви "дивитеся" на зображення лише один раз, що робить його неймовірно швидким.

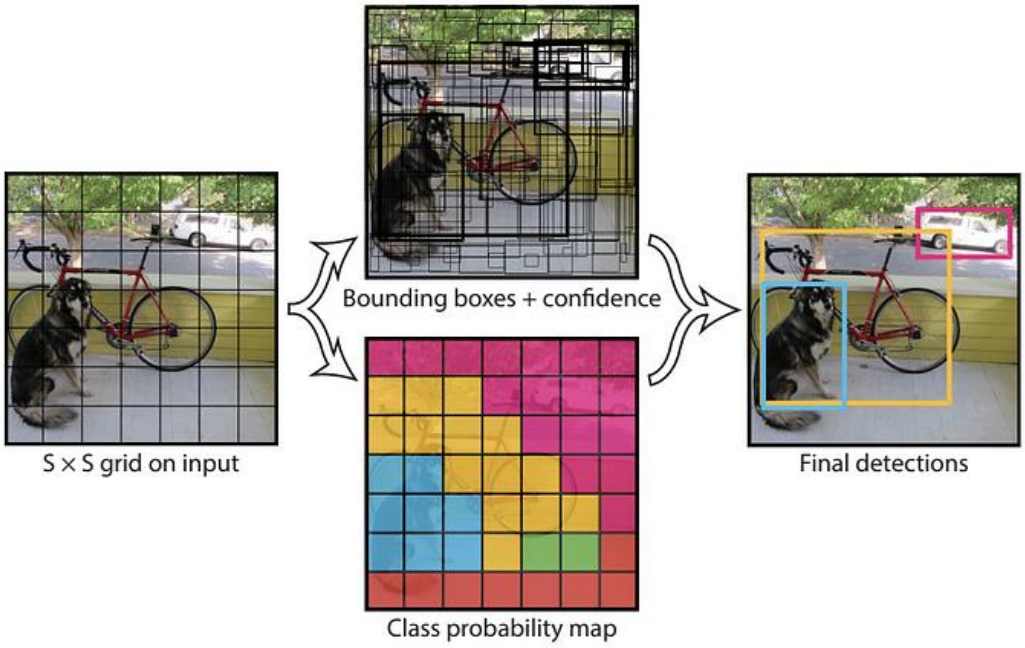


Рисунок 1.7. - Алгоритм роботи YOLO

Незважаючи на швидкість та здатність виявляти об'єкти на зображенні в режимі реального часу, YOLO має деякі обмеження. Він має труднощі з виявленням маленьких об'єктів, що згруповані разом, і він часто прогавляє об'єкти, які не є центром у відповідних областях. Однак, попри ці обмеження, YOLO залишається одним з найшвидших та найбільш широко використовуваних алгоритмів виявлення об'єктів

Для реалізації автоматичної системи розпізнавання захворювань рослин, що розглядається в цій роботі було обрано архітектуру Faster R-CNN на базі ResNet-50, зокрема її нову версію: Faster R-CNN ResNet-50 FPN v2. Faster R-CNN - це третє покоління R-CNN, яке забезпечує відмінну точність виявлення об'єктів і швидкість обробки. Faster R-CNN, на відміну від Fast R-CNN, включає в себе модуль визначення областей пропозицій (RPN), який використовується для генерації областей, які містять об'єкти. Це забезпечує

більшу ефективність порівняно з попередніми методами, які вимагали зовнішніх алгоритмів для генерації областей пропозицій.

Алгоритм Faster R-CNN був обраний на відміну від YOLO через його високу точність виявлення об'єктів. YOLO, хоча і демонструє високу швидкість виявлення об'єктів, має певні обмеження, як-от виявлення невеликих або групових об'єктів. Mask R-CNN, хоча і є розширенням Faster R-CNN, її було розроблено з метою вирішення завдань семантичної сегментації, а не просто виявлення об'єктів, що може бути занадто складним для потреб цієї конкретної системи.

Архітектуру ResNet-50 було обрано в якості основної (backbone) архітектури, через її глибину та високу точність. ResNet-50 включає 50 шарів глибокого навчання, що дозволяє виявляти і витягувати дуже складні шаблони з даних. Вона також використовує технологію "skip connections" для вирішення проблеми затухання градієнтів, що зазвичай виникає при глибокому навчанні.

Мережа піраміди ознак або Feature Pyramid Network (FPN) – це метод, який використовується для виявлення об'єктів різного масштабу. Він генерує піраміду ознак, яка використовується для виявлення об'єктів різного розміру. Версія Faster R-CNN ResNet-50 FPN v2 включає оновлення, які забезпечують кращу точність і швидкість.

Отже, модель Faster R-CNN з ResNet-50 та FPN була обрана для розроблюваної в даній роботі системи через наступні причини: вона забезпечує високу точність, здатна обробляти об'єкти різного масштабу та забезпечує високу швидкість обробки, що є критично важливим для реального застосування.

1.3. Порівняльний аналіз існуючих програмних систем.

У рамках виконання випускної кваліфікаційної роботи очікується розробити веб-додаток для розпізнавання захворювань на листях агрокультур. Метою є не лише виявлення захворювання, але й локалізація заражених

осередків та точна класифікація відповідних захворювань. Додаток має надавати користувачам корисні рекомендації щодо вирішення виявлених проблем, зокрема методів усунення цих захворювань.

У контексті цього проекту необхідно провести ґрунтовний аналіз сучасного ринку програмних продуктів, призначених для інтелектуальної обробки зображень. На даний момент вже існує кілька високоякісних програмних систем, що використовують методи машинного навчання для виявлення та класифікації захворювань і шкідників на листях рослин. Оцінка таких систем допоможе визначити найефективніші технології та підходи, які могли б знайти застосування в нашому веб-додатку, а також з'ясувати можливі прогалини в сучасних рішеннях, що могли б бути заповнені в ході нашого проекту.

FarmAI представляє собою високотехнологічне рішення в галузі сільського господарства, яке використовує моделі машинного та глибокого навчання для виявлення захворювань рослин. Ця система в основному зосереджена на аналізі зображень для точного виявлення захворювань. Основний інструмент, який використовується FarmAI - це згорткові нейронні мережі (CNN). CNN є особливо підходящим для обробки зображень завдяки своїй здатності "бачити" високоабстрактні характеристики зображення через використання багатьох згорткових та підсіткових шарів. Окрім CNN, FarmAI також використовує інші моделі глибокого навчання для підвищення точності визначення захворювань. Ці моделі можуть включати рекурентні нейронні мережі (RNN) для обробки послідовних даних або GAN (генеративно-суперечливі мережі) для генерації нових зразків зображень, які можуть бути використані для тренування. FarmAI розроблений з метою оптимізації процесу виявлення захворювань рослин, зменшуючи потребу в трудомісткому та дороговартісному ручному аналізі. Завдяки можливостям штучного інтелекту, FarmAI може надавати швидкі та точні діагностики, що дозволяє фермерам вчасно реагувати на проблеми та зменшувати втрати врожаю.

Plantix – це мобільний додаток для фермерів, консультантів та садівників. Це один з найпопулярніших додатків для виявлення захворювань рослин. Додаток діагностує пошкодження шкідниками, хвороби рослин та дефіцит поживних речовин, що впливають на врожай, і пропонує відповідні заходи лікування. Він використовує методи машинного навчання та глибокого навчання для аналізу зображень рослин і визначення різних типів захворювань. Хоча конкретні моделі, які використовуються в Plantix, є комерційною таємницею, схожі системи часто використовують згорткові нейронні мережі (CNN), які виявилися ефективними для завдань розпізнавання зображень.

Agrio є сучасним рішенням для землеробства, яке за допомогою штучного інтелекту підтримує агрономів, фермерів і садівників у віддаленому виявленні та лікуванні захворювань рослин та шкідників на полях, фермах та в садах. Ця платформа розроблена з використанням передових алгоритмів комп'ютерного зору та штучного інтелекту, що постійно удосконалюються і накопичують знання численних експертів у сфері агрономії. Агріо слугує як цифровий лікар для рослин, який виявляє захворювання та проблеми на основі фотографій, зроблених через смартфон. Завдяки його можливостям, користувачі мають змогу одержати діагностику та рекомендації щодо вирішення проблеми в дуже короткі терміни. Пропонуючи деталізовані стратегії інтегрованого управління шкідниками (IPM), ця платформа допомагає оптимізувати продуктивність врожаю, поліпшувати якість врожаю та знизити витрати на лікування.Порівняємо розроблений мною додаток для розпізнавання захворювань рослин із популярним додатком з магазину AppStore “ Plant Disease Identifier”.

Plant Disease Identifier – це мобільний додаток, доступний в App Store, який використовує штучний інтелект для ідентифікації різних захворювань рослин. Цей додаток розроблений з метою допомоги фермерам, агрономам та ентузіастам садівництва у виявленні і контролі захворювань рослин. Він пропонує зручний і доступний спосіб діагностики стану рослин безпосередньо

зі смартфона. Додаток "Plant Disease Identifier" працює шляхом сканування та аналізу зображень рослин, взятих через камеру смартфона. Він використовує досить високоточні моделі машинного навчання для визначення можливих проблем з рослинами, таких як хвороби та шкідники. Після визначення проблеми додаток надає детальну інформацію про захворювання та рекомендації щодо способів лікування. Інтерфейс додатку зображено на рис. 1.8.

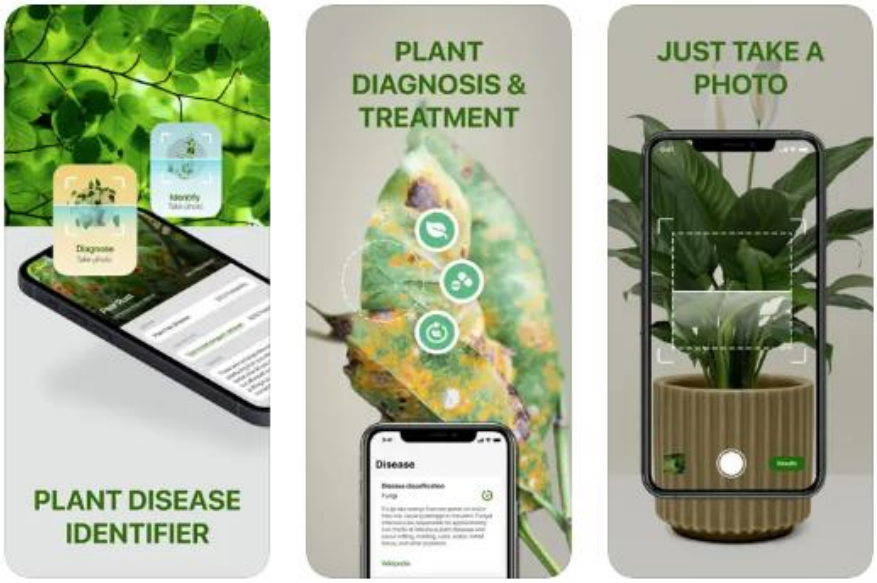


Рис 1.8 – Інтерфейс мобільного додатку “Plant Disease Identifier”

В контексті цієї кваліфікаційної роботи, ми проведемо порівняльний аналіз двох споживчих додатків: "Plant Disease Identifier", доступного на платформі App Store, та додатка, який в даний момент розробляється в рамках нашого проекту. Оскільки обидва додатки розраховані на використання не великими корпораціями, а кінцевими споживачами, зокрема фермерами та садівниками, то ці два програмні продукти стануть об'єктами нашого порівняльного аналізу. Оцінювати і порівнювати додатки ми будемо за рядом критеріїв, що включають перелік основних функцій, які вони виконують. Якщо певна функція присутня в додатку, або вона виконується в ньому краще, ніж в аналога, то відповідний рядок в таблиці порівняння буде позначено сірим кольором і додатку буде присуджено один "плюс". Порівняльний аналіз наведено у таблиці 1.1.

Таблиця 1.1. – Порівняльний аналіз розробленого додатку та мобільного додатку “Plant Disease Identifier”

№	Опис	Додаток розпізнавання захворювань рослин	“Plant Disease Identifier”
1	Можливість відкрити додаток з будь-якого пристрою	Так	Лише мобільний додаток
2	Наявність зручного та зрозумілого інтерфейсу	Так	Так
3	Можливість переглянути підказку щодо використання додатку	Ні	Так
4	Можливість зробити фото за допомогою камери та завантажити його	Ні	Так
5	Можливість завантажити фото з диску	Так	Так
6	Кількість класів захворювань, що може розпізнати додаток	30	30
7	Локалізація об’єктів захворювання на зображенні	Так	Ні
8	Можливість переглянути історію ідентифікацій	Так	Так
9	Виведення інформації про захворювання та рекомендацій щодо його лікування	Так	Так
10	Точність виявлення захворювання	86%	92%
Всього		7	8

Після детального порівняння програмних продуктів, мобільний додаток для класифікації захворювань рослин, "Plant Disease Identifier", використовуючи методи машинного навчання, отримав 8 балів. Тоді як додаток, який ми розробляємо та який використовує методи регіональних згорткових нейронних мереж для визначення захворювань рослин, здобув 7 балів. На сьогоднішній день "Plant Disease Identifier" може похизуватися більшою базою з зображеннями захворювань, відповідно кращою точністю їх класифікації, більшою зручністю використання та більш виваженою візуальною презентацією для кінцевого користувача. Однак ключовим елементом, який надає перевагу нашому додатку є його здатність виявляти об'єкти та локалізувати їх за допомогою прямокутних рамок. Ця особливість дозволяє більш ефективно опрацьовувати зображення, на яких присутні більше одного об'єкта або декілька осередків захворювань. Підсумовуючи, можна стверджувати, що обидва додатки успішно виконують свої завдання і задовольняють потреби користувачів в контексті їх використання.

1.4. Постановка задачі створення нейромережного застосунку розпізнавання захворювань рослин.

У даній дипломній роботі буде вирішуватися задача з розробки застосунку для автоматичного розпізнавання та класифікації захворювань рослин за допомогою R-CNN. Даний додаток буде використовувати Faster R-CNN на базі архітектури ResNet-50 з використанням FPN для виявлення та класифікації захворювань на вхідних зображеннях, що завантажуються на сервер користувачем. Користувач зможе завантажити зображення з листям рослини та отримати у відповідь зображення з локалізованими захворюваннями за допомогою прямокутних рамок, класи відповідних захворювань та точність прогнозу для кожного об’єкта на цьому зображенні. На основі цього користувачу буде виводитися інформація про тип захворювання та рекомендації щодо його лікування.

Метою цієї роботи є створення нейромережного застосунку, що здатен автоматично розпізнавати та класифікувати захворювання на листях агрокультур в межах обраного набору даних, а також надавати рекомендації щодо подальшого вирішення виявлених проблем.

Об'єктом дослідження є процеси виявлення, ідентифікації та класифікації захворювань з використанням машинного зору та глибоких нейронних мереж.

Предметом дослідження є методи та технології машинного зору, глибокого навчання, зокрема застосування регіональних згорткових нейронних мереж для виявлення та ідентифікації захворювань на зображеннях рослин.

Алгоритм створення веб-додатку:

1. Розробити модуль для роботи з моделлю нейронної мережі. Цей модуль повинен включати:
    - збір та попередню обробку даних для навчання;
    - побудовану модель Faster R-CNN;
    - налаштування гіперпараметрів моделі;
    - навчання та тестування створеної моделі;
    - оцінка точності моделі;
    - збереження створеної моделі у файл.
2. Розробити базу даних, яка буде зберігати інформацію щодо розпізнавання зображень з захворюваннями завантажених користувачами.
3. Розробити модуль для обробки вхідних запитів, що включатиме:
    - логіку обробки вхідних запитів користувачів;
    - логіку формування прогнозу з використанням попередньо навченої моделі нейронної мережі;
    - логіку роботи збереження даних прогнозів у базі даних;
    - валідацію вхідних даних;

- логіку відображення відповіді, що буде передаватися користувачу.

4. Розробити модуль представлення клієнтської частини, цей модуль включатиме:
    - інтерфейс для завантаження обраного зображення;
    - валідацію вхідних зображень;
    - сторінку для виведення сформованої відповіді користувачу;
    - сторінку з історією запитів, здійснених користувачем.

Сформуємо основні вимоги для додатку, що розробляється.

Функціональні вимоги:

- можливість завантаження файлу шляхом перетягування в дроп-зону;
- можливість завантаження файлу через оглядач файлів;
- відображення назви обраного файлу;
- наявність валідації формату вхідних зображень;
- забезпечення точності ідентифікації захворювання на зображенні нейронною мережею та формування вірного прогнозу щодо виду захворювання та знаходження його на зображенні в межах обраного датасету;
- виведення користувачу на екран рекомендацій та можливих кроків щодо лікування хвороби;
- зберігання отриманих прогнозів у базі даних.
- для кожного користувача повинні зберігатися саме його прогнози;
- можливість користувачем переглядати історію створених прогнозів;
- можливість перегляду користувачем обраного з історії прогнозу;

Нефункціональні вимоги:

- веб-сайт має бути адаптивним та чутливим до масштабування;
- веб-сайт повинен добре функціонувати на мобільних пристроях;

- оформлення інтерфейсу має бути читабельним та зручним для розуміння;
- серверна частина сайту має бути розроблена на фреймворку django;
- функції валідації даних мають бути написані мовою javascript;
- після оновлення сторінки виведені дані мають зберігатися.

Системні вимоги:

- додаток повинен бути сумісний зі всіма основними веб-браузерами, такими як Google Chrome, Mozilla Firefox, Safari, і Microsoft Edge;
- додаток має працювати швидко, незалежно від об'єму даних або кількості користувачів, що одночасно користуються сервісом;
- всі дані, що обробляються та передаються, повинні бути захищені;
- додаток має бути доступний і зручний для користувачів мобільних пристроїв, забезпечуючи плавний робочий процес на смартфонах та планшетах.

Висновки до першого розділу

В першому розділі було проведено детальний аналіз предметної області вирішуваної задачі. З'ясовано, що діагностика захворювань рослин за допомогою глибокого навчання є актуальною і надзвичайно важливою проблемою сучасного сільського господарства. Визначено мету, об'єкт та предмет дослідження, а також завдання дослідження. Проаналізовано сучасні методи та підходи до вирішення проблеми, включаючи використання згорткових нейронних мереж та методів машинного навчання. Розглянуто існуючі програмні рішення в даній сфері, включаючи мобільний додаток "Plant Disease Identifier". Виконано порівняльний аналіз із розроблюваною системою, визначено їх переваги та недоліки. Сформовано чітку постановку задачі розвитку нейромережного застосунку для розпізнавання захворювань рослин. Вказано на основні вимоги до системи, які включають функціональні

та нефункціональні вимоги, а також вимоги до окремих видів забезпечення та їх компонентів. В результаті було зроблено висновок, що розробка веб-додатку для розпізнавання захворювань рослин за допомогою глибокого навчання є перспективною та актуальною задачею, яка може мати значний вплив на ефективність сільськогосподарської діяльності.

РОЗДІЛ 2. ПРОЕКТНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ НЕЙРОМЕРЕЖНОГО ЗАСТОСУНКУ РОЗПІЗНАВАННЯ ЗАХВОРЮВАНЬ РОСЛИН

2.1. Аналіз варіантів використання додатку.

На головній сторінці розроблюваного нами веб-додатку користувач матиме змогу перетягнути потрібне зображення з рослиною, на якій він хоче розпізнати захворювання в дроп-зону або натиснути лівою кнопкою миші у виокремлену область, щоб обрати зображення через оглядач файлів. Після натисканні кнопки «Завантажити файл» користувачу виведеться обране ним зображення разом із ідентифікованою хворобою на ньому та рекомендацією щодо усунення захворювання.

Створимо модель варіантів використання.

Дійові особи:

- користувач;
- адміністратор.

Варіанти використання:

Користувач:

- Завантаження файлу через перетягування в дроп-зону або через оглядач файлів на головній сторінці.
- Перегляд детальної інформації про розпізнане захворювання та рекомендації щодо його лікування на сторінці з відповіддю.
- Повторне завантаження файлу на сторінці з відповіддю.
- Перегляд історії своїх запитів на сторінці "Мої прогнози".
- Перегляд деталізованої інформації про попередні запити, натиснувши на запис в історії запитів.

Адміністратор:

- Вхід в панель адміністратора.
- Перегляд бази даних(БД) користувачів та їх запитів.

– Внесення змін у БД.
– Завантаження нових даних в БД.

Побудуємо діаграму варіантів використання для кращого представлення функціональних вимог до системи, що розробляється (рис. 2.1).

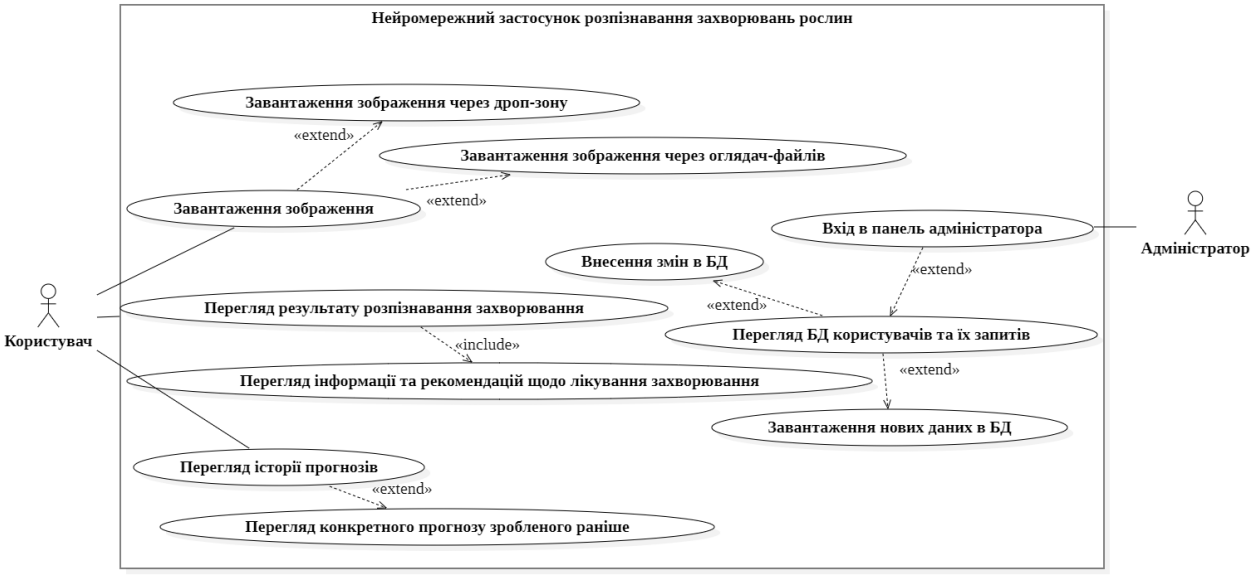


Рис. 2.1 – Діаграма варіантів використання додатку

Сценарії використання.

Основний сценарій розпізнавання зображення:

1. Користувач завантажує зображення одним із двох доступних методів.
2. Користувач натискає на кнопку «Завантажити файл».
3. Система обробляє зображення та виводить результат розпізнавання на сторінці з відповіддю: зображення з виокремленими рамками об'єктів розпізнавання, підписами класу захворювання та впевненості прогнозу, опис захворювання та рекомендації по лікуванню.

Альтернативний сценарій розпізнавання зображення:

1. Користувач натиснув кнопку завантаження файлу не обравши жоден файл, система виводить повідомлення «Будь ласка, оберіть файл».

2. Користувач обрав файл з недопустимим форматом, система виводить повідомлення «Невірний формат файлу».
3. Користувач обрав пошкоджене зображення. Система виводить повідомлення «Зображення пошкоджено або не може бути прочитано. Будь ласка, спробуйте інше зображення».

Основний сценарій перегляду історії прогнозів:

1. Користувач переходить на сторінку «Мої прогнози».
2. Система виводить на екран список з історією всіх прогнозів сформованих користувачу з зазначенням дати створення прогнозу та назви файлу. Прогнози зберігаються в базі даних за ключем поточної сесії.
3. Користувач натискає на потрібний йому прогноз з метою перегляду його змісту.
4. Система переадресовує користувача на сторінку з «id» відповідного прогнозу з відображенням змісту та рекомендацій.

Альтернативний сценарій перегляду історії прогнозів:

1. Користувач перейшов на сторінку «Мої прогнози», попередньо не виконавши жодного запиту в систему
2. Система виведе на сторінці повідомлення: «Жодного прогнозу немає».

Сценарій для адміністратора:

1. Адміністратор входить в панель адміністратора.
2. Адміністратор переглядає базу даних користувачів та їх запитів.
3. Адміністратор робить зміни в базі даних, якщо потрібно.
4. Адміністратор завантажує нові дані в базу, якщо потрібно.

Для кращого розуміння роботи застосунку наведемо також діаграми послідовності та діаграми діяльності роботи додатку (рис. 2.2, рис 2.3).

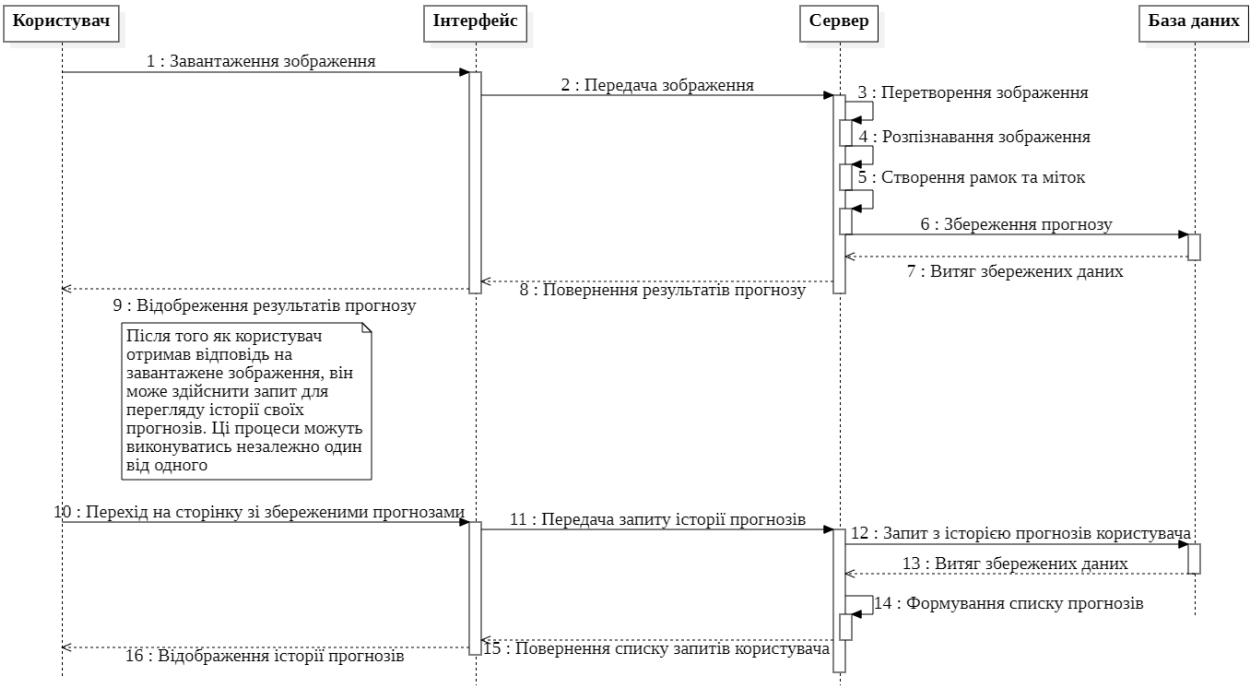


Рисунок 2.2 – Діаграма послідовності роботи додатку

Як показано на діаграмі послідовності: спочатку користувач має можливість обрати, чи бажає він переглянути історію своїх прогнозів, чи зробити новий прогноз. Якщо користувач вибирає зробити новий прогноз, він завантажує зображення рослини через інтерфейс додатку. Далі зображення передається на сервер, де за допомогою нейромережі відбувається процес розпізнавання захворювання. Після цього результат повертається користувачу через інтерфейс та зберігається в базі даних для подальшого доступу. У випадку, коли користувач обирає перегляд історії прогнозів, запит передається через сервер до бази даних, звідки витягуються дані про попередні прогнози користувача. Ці дані потім відображаються користувачу через інтерфейс додатку.

Діаграма діяльності детально ілюструє всі процеси, які відбуваються під час різних етапів валідації вхідного зображення у нейромережному застосунку для розпізнавання захворювань рослин. Вона допомагає візуалізувати, як система виконує валідацію та які дії системи викликаються при виявленні помилок валідації.

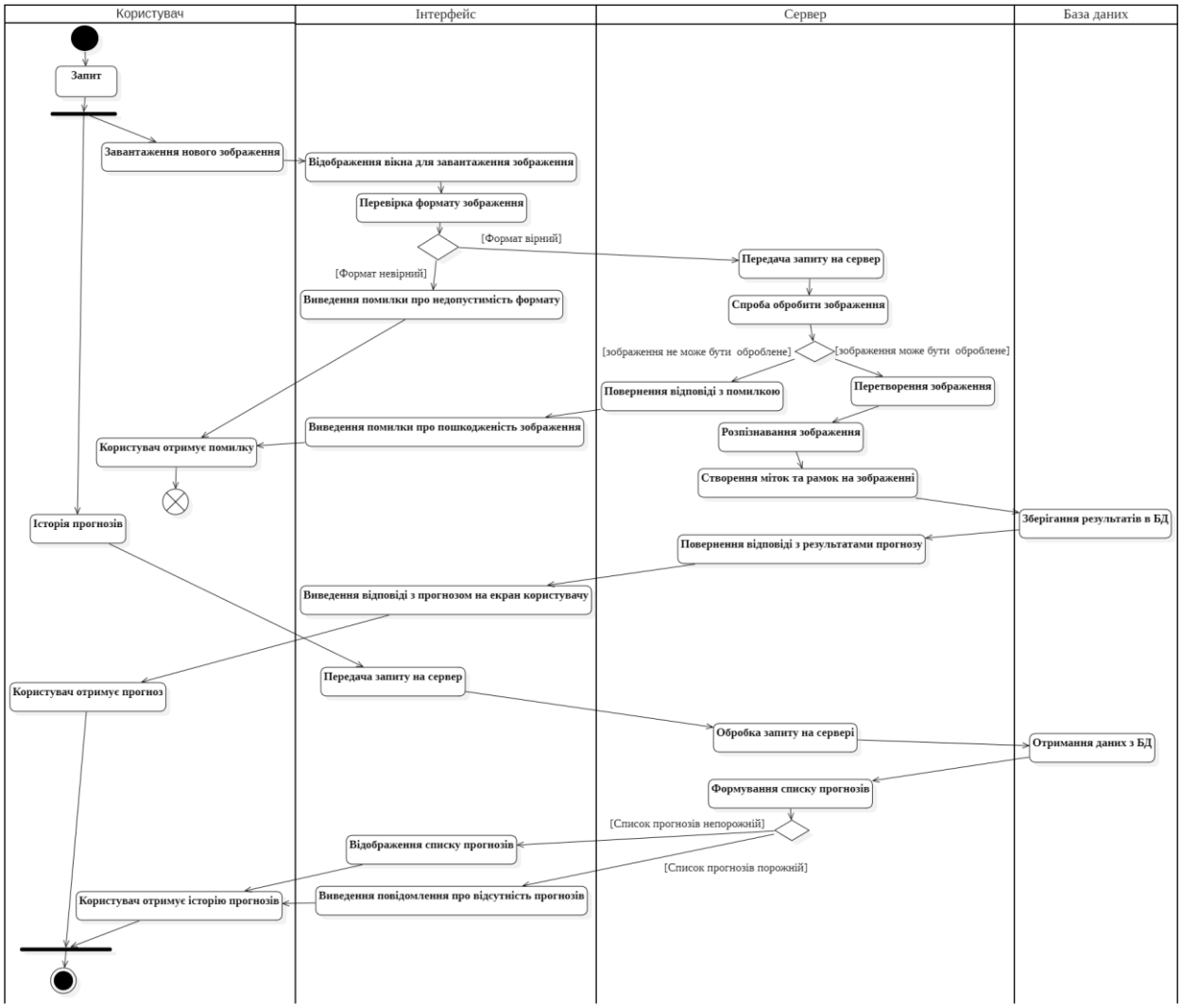


Рисунок 2.3 – Діаграма діяльності роботи додатку

2.2. Розробка моделі нейронної мережі.

2.2.1. Підготовка набору даних для навчання нейромережі.

Підготовка даних є важливим етапом будь-якого процесу машинного навчання. На цьому етапі ми забезпечуємо, що дані, які використовуються для навчання моделі, належним чином очищені, організовані і готові до обробки моделлю. Для тренування нейронних мереж, особливо моделей для об'єктного виявлення, цей процес часто включає такі кроки, як:

- Завантаження даних: Спочатку необхідно завантажити датасет, що використовуватиметься для навчання.
- Розділення даних: Далі, зазвичай розподіляють датасет на тренувальну, валідаційну та тестову підмножини.

– Анотація даних: Кожне зображення у датасеті повинне мати відповідні мітки, які вказують, де на зображенні знаходиться об'єкт і до якого класу він належить. Для завдання об'єктного виявлення це зазвичай включає координати обмежувальної рамки і мітку класу для кожного об'єкта на зображенні.

Для нашої кваліфікаційної роботи було обрано анотований датасет PlantDoc з Roboflow – платформи, що забезпечує засоби для роботи з датасетами для вирішення задач комп’ютерного зору. Вона допомагає у підготовці, аугментації, розмітці, тренуванні та виведенні моделей глибокого навчання для завдань розпізнавання образів [22].

PlantDoc - це обширний датасет, що містить 2569 зображень, які представляють 13 різних видів рослин і включають 30 унікальних класів, що відображають стан здоров'я рослин (хворі та здорові). Загалом, у цьому наборі даних є 8851 анотація, що допомагає в задачах класифікації зображень та виявлення об'єктів. В датасеті представлені наступні класи рослин: 'Apple Scab Leaf' (Листя яблуні із борошнистою паршою), 'Apple leaf' (Листя яблуні), 'Apple rust leaf' (Листя яблуні з іржею), 'Bell pepper leaf spot' (Плями на листі болгарського перцю), 'Bell pepper leaf' (Листя болгарського перцю), 'Blueberry leaf' (Листя чорниці), 'Cherry leaf' (Листя вишні), 'Corn Gray leaf spot' (Cipi плями на листі кукурудзи), 'Corn leaf blight' (Плямистість листя кукурудзи), 'Corn rust leaf' (Іржа на листі кукурудзи), 'Peach leaf' (Листя персика), 'Potato leaf early blight' (Рання плямистість листя картоплі), 'Potato leaf late blight' (Пізня плямистість листя картоплі), 'Potato leaf' (Листя картоплі), 'Raspberry leaf' (Листя малини), 'Soyabean leaf' (Листя сої), 'Squash Powdery mildew leaf' (Листz кабачка з пороховою паршою), 'Strawberry leaf' (Листz полуниці), 'Tomato Early blight leaf' (Рання плямистість листя помідора), 'Tomato Septoria leaf spot' (Плями на листі помідора, викликані Septoria), 'Tomato leaf bacterial spot' (Бактеріальні плями на листі помідора), 'Tomato leaf late blight' (Пізня плямистість листя помідора), 'Tomato leaf mosaic virus' (Мозаїчний вірус листя помідора), 'Tomato leaf yellow virus' (Жовтий вірус листя помідора), 'Tomato

leaf' (Листя помідора), 'Tomato mold leaf' (Пліснява на листі помідора), 'Tomato two spotted spider mites leaf' (Листя помідора із плямами від двоплямого павутинного кліща), 'grape leaf black rot' (Чорна гниль на листі винограду), 'grape leaf' (Листя винограду).

Приклад зображення з датасету зображено на рисунку 2.4.

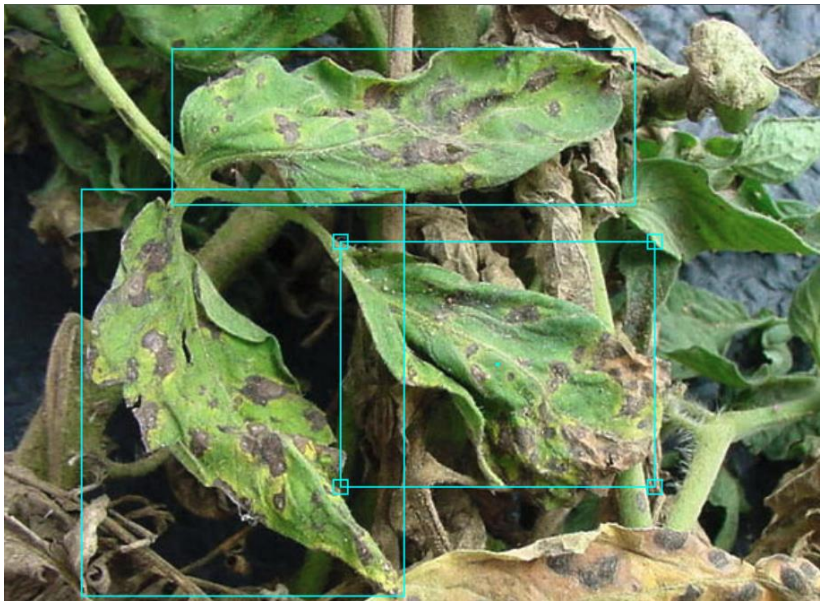


Рисунок 2.4 – Приклад зображення з датасету PlantDoc

Набір даних PlantDoc доступний у різних форматах анотацій, включаючи CoCo JSON, Pascal VOC XML, YOLO v4 PyTorch, Tensorflow Object Detection CSV та інші. Однак, для нашої вибраної моделі нейронної мережі, формат анотацій Pascal VOC XML є найбільш відповідним варіантом. VOC (Visual Object Classes) - це формат, розроблений для датасету Pascal VOC, який включає в себе зображення та відповідні анотації. XML (Extensible Markup Language) - це мова розмітки, яка дозволяє створювати семантичні структури даних.

В анотаціях VOC XML для кожного зображення створюється окремий XML-файл з однойменною назвою. Кожен файл містить такі поля:

- filename: Назва файла зображення.
- path: Шлях до файлу зображення.

- size: Інформація про розмір зображення, включаючи ширину, висоту та глибину (кількість каналів).
- segmented: Чи використовувався для анотації сегмент (набір пікселів, що формують об'єкт).
- object: Блок, що містить інформацію про об'єкти на зображенні, включаючи клас об'єкта (name) і опис об'єкта у форматі оголошення (boundarybox), що включає координати верхнього лівого та нижнього правого кутів об'єкта (xmin, ymin, xmax, ymax).

Під час підготовки датасету для навчання моделі нейронної мережі, важливим етапом є аналіз балансу класів. У вибраному датасеті PlantDoc, як показано на рисунку 2.5, не всі класи рівномірно представлені.

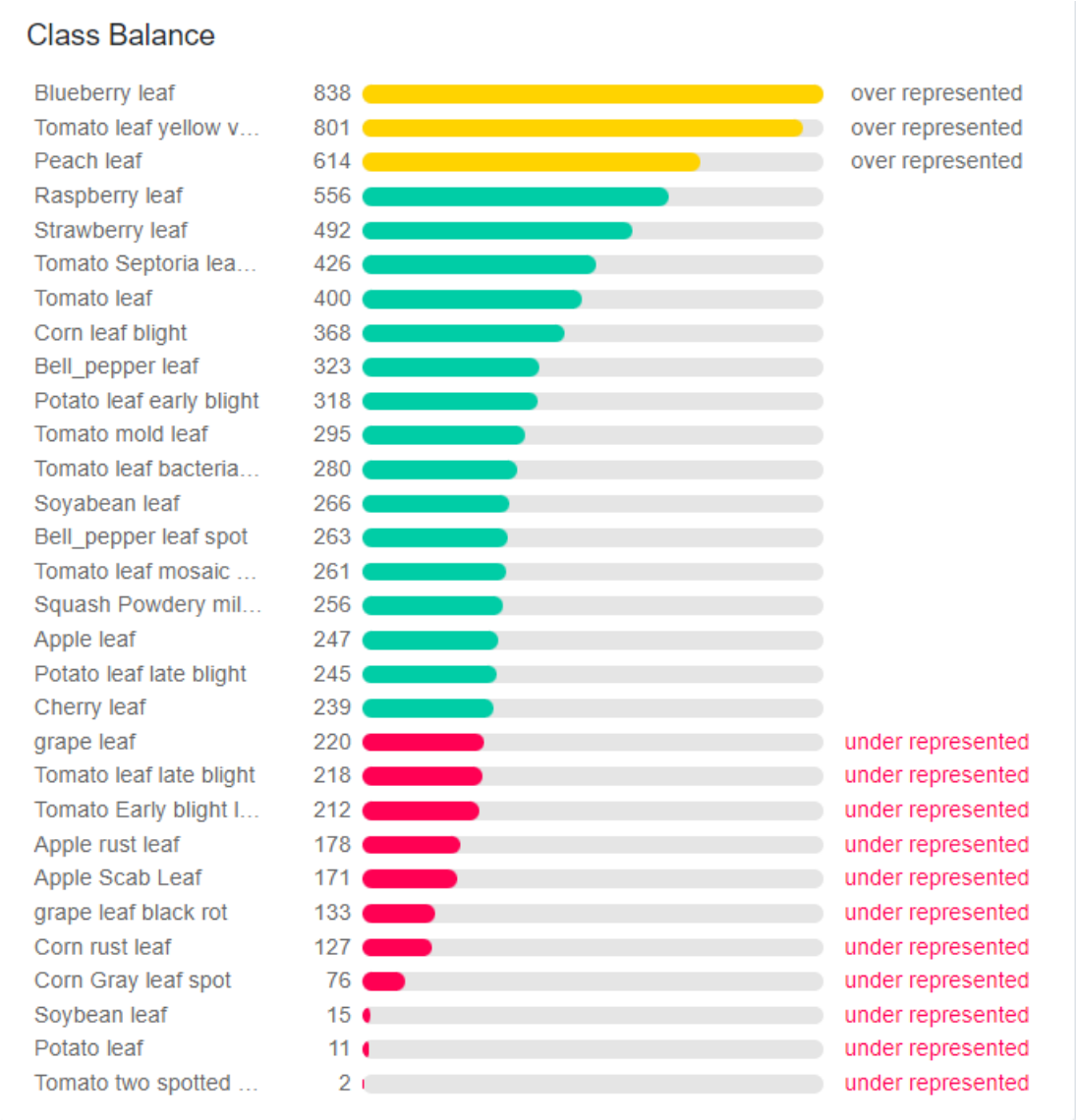


Рисунок 2.5 – Представлення балансу класів в датасеті PlantDoc

Надлишкова представленість деяких класів та недостатня представленість інших може негативно вплинути на точність передбачень моделі. Щоб вирішити цю проблему та збалансувати класи, використовують методи аугментації. Аугментація даних - це процес створення нових тренувальних прикладів шляхом внесення невеликих модифікацій до існуючих даних. Це допомагає моделі краще узагальнювати і протистояти перенавчанню, оскільки вона вивчає різноманітність даних. Типи аугментації, які ми застосуємо, включають горизонтальне та вертикальне відображення, поворот, масштабування, зсув, зміну яскравості, контрасту та насиченості кольору. Важливо вибрати аугментації, які відповідають природі нашого датасету та завданню, яке ми намагаємось вирішити. На рисунку 2.6 зображено приклад аугментації даних.

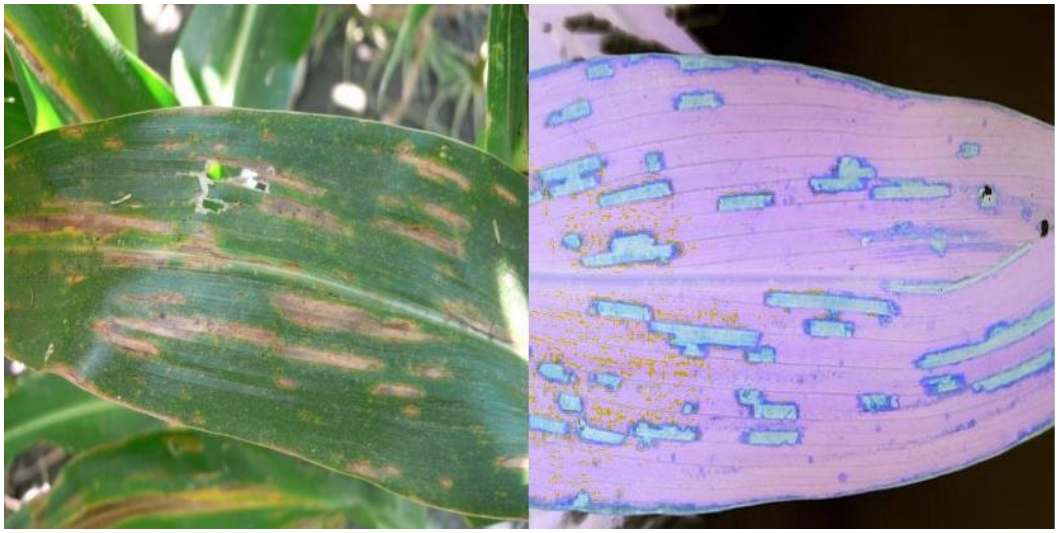


Рисунок 2.6 – Аугментація зображення для Corn gray leaf

2.2.1 Створення моделі нейронної мережі.

Для виконання поставленого завдання було обрано підхід до навчання transfer learning (TL). Transfer learning - це підхід до машинного навчання, при якому модель, навчена на одному завданні, використовується як початкова точка для навчання в іншому завданні. Цей метод широко використовується в глибокому навчанні, оскільки може значно скоротити час навчання і покращити точність моделі, особливо при невеликій кількості тренувальних даних. Моделлю нейронної мережі, що була вибрана для виконання цього

завдання, є Faster R-CNN FPN. Faster R-CNN FPN - це одна з передових моделей для виявлення об'єктів, яка використовує концепцію Feature Pyramid Networks (FPN) для ефективного виявлення об'єктів різних розмірів.

Feature Pyramid Network (FPN) є архітектурою, яка дозволяє ефективно виявляти об'єкти різних масштабів на зображеннях. Головною проблемою виявлення об'єктів є те, що об'єкти різних розмірів можуть бути представлені різними масштабами областей. FPN пропонує рішення для цієї проблеми шляхом створення пірамідальної структури ознак з різними розмірами. Архітектура FPN використовує нижні шари згорткових нейромереж для створення пірамідальної структури (рис. 2.7). Згорткові шари генерують ознаки з різних рівнів деталізації. Високорівневі ознаки мають широке поле зору, що допомагає виявляти об'єкти загального контексту, тоді як низькорівневі ознаки мають більшу локальну деталізацію, що корисно при виявленні дрібних об'єктів. Застосування згорткових шарів різних розмірів дозволяє побудувати ієрархічну піраміду ознак [23].

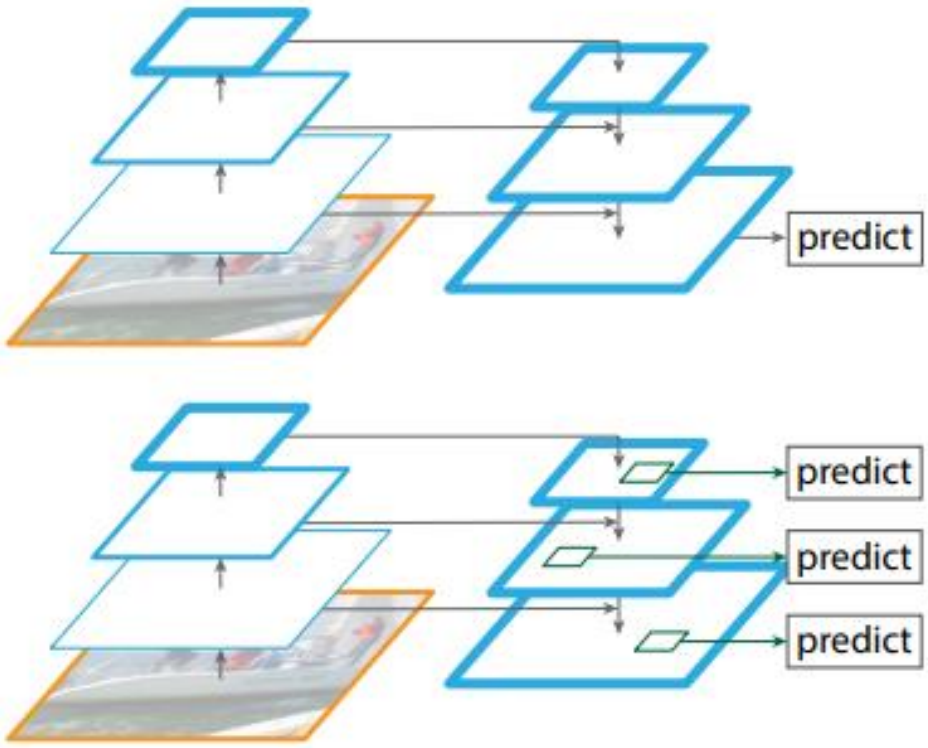


Рисунок 2.7 – Структура FPN

Faster R-CNN, в поєднанні з FPN (Feature Pyramid Network), використовує спеціалізований шар, названий RPN (Region Proposal Network), для визначення регіонів на зображенні, які, ймовірно, містять об'єкти. RPN працює на ознаках, отриманих з FPN. Потім RPN прогнозує ймовірність наявності об'єкту та координати граничних рамок для кожного об’єкту. Для правильного визначення та корегування граничних рамок використовуються якірні рамки (anchor boxes) з різними співвідношеннями сторін (1:1, 2:1, 1:2) і різними розмірами. Якірні рамки використовуються як початкові точки для генерації пропозицій регіонів. Як показано на рисунку 2.8, у структурі RPN для кожного рівня масштабу, до карт об'єктів застосовується згортковий шар 3 × 3 (sliding window), а потім застосовуються два окремих згорткових шари, розмірністю 1 × 1. Перший шар (cls 2k) використовується для прогнозування наявності об’єктів в пропозиції (класифікація з двома класами). Другий шар (cls 4k) необхідний для визначення вектору правок координат якорів.

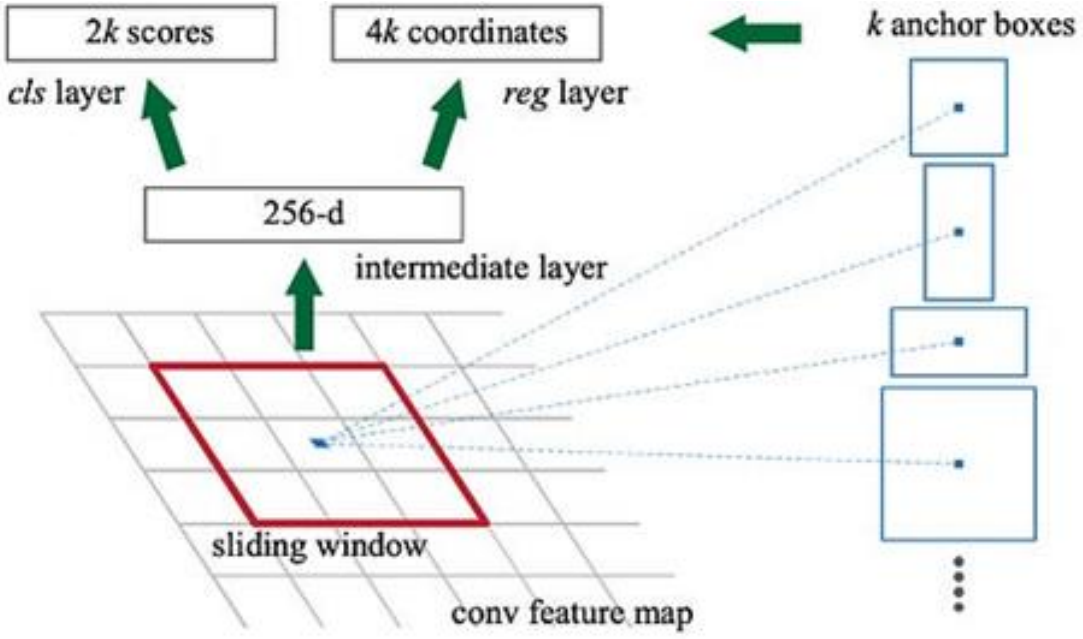


Рисунок 2.8 – Зображення ковзаючого вікна (згортки), що виявляє анкори

Загальну архітектуру мережі Faster R-CNN зображено на рисунку 2.9. Вона складається з 4 частин: згорткова нейронна мережа (convolutional network) в нашому випадку мережа Resnet-50, регіональна мережа пропозицій (region proposal network), шар об’єднання(вирівнювання) регіонів інтересу

(RoI Align) та повнозв'язний шар з подальшою класифікацією та регресією обмежуючих рамок.

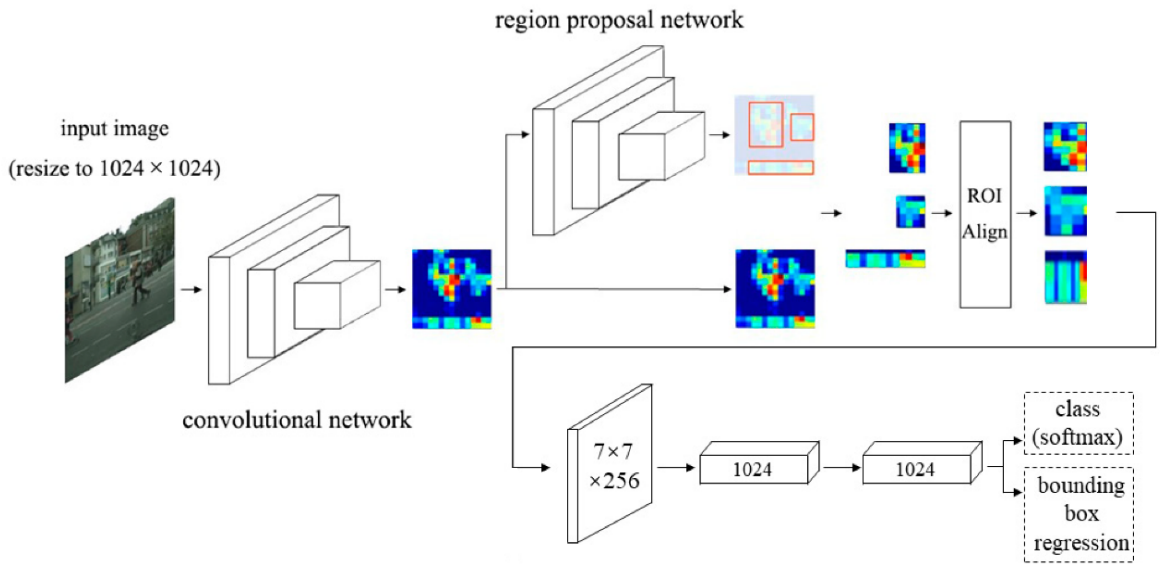


Рисунок 2.9 – Архітектура Faster R-CNN

Для підготовки моделі Faster R-CNN FPN до навчання було визначено кроки, наведені нижче.

Попередня обробка зображень. Перш за все, зображення завантажуються з вказаного датасету. Зображення змінюють розмір таким чином, щоб відповідати квадрату, в нашому випадку всі зображення з нашого датасету мають розмірність 416 на 416 пікселів. Такий розмір був вибраний згідно зі стандартними розмірами, які використовуються в моделі Faster R-CNN FPN. Крім того, зображення нормалізуються з використанням стандартних значень для кожного з трьох каналів (RGB). Також ми використовуємо аугментацію зображень, описану в цьому розділі раніше. Після завантаження і зміни розміру, зображення перетворюються в тензори, що є стандартною процедурою при підготовці даних для моделей глибокого навчання. Перетворення зображення в тензор означає, що ми перетворюємо зображення, яке представлено як двовимірний масив пікселів, в багатовимірний масив, який можна обробити в моделі глибокого навчання. Цей багатовимірний масив (тензор) має додаткові виміри для каналів кольору

(якщо ми працюємо з кольоровими зображеннями) і для пакетів зображень (якщо ми працюємо з більш ніж одним зображенням за раз).

Додавання анотацій. Для кожного зображення, анотації завантажуються у формі bounding boxes та відповідних міток. Координати bounding boxes і мітки класу перетворюються в тензори PyTorch для подальшого використання в процесі навчання.

Ініціалізація моделі. Обрану модель ми ініціалізуємо з вагами CoCo. Ваги CoCo - це ваги моделі, навченої на наборі даних CoCo (Common Objects in Context), який є одним з найбільших наборів даних для виявлення, сегментації та розпізнавання об'єктів. Наша мета - додатково навчити нашу модель на анотованому датасеті PlantDoc. Це навчання дозволить моделі визначати та класифікувати різні захворювання рослин. Більше того, модель зможе не тільки виявляти ці захворювання, але й локалізувати їх на зображенні, використовуючи прямокутні рамки, або bounding boxes, що значно полегшує процес ідентифікації та діагностики цих захворювань.

Визначення оптимізатора та коефіцієнта навчання (learning rate). В нашій моделі будемо використовувати оптимізатор Stochastic Gradient Descent (SGD) з моментом (momentum) рівним 0.9. Швидкість навчання (learning rate) встановлюється рівною 0.005, що є доволі типовим вибором для цього типу моделей. Ваговий розпад (weight decay) встановлюється рівним 0.0005. Stochastic Gradient Descent або стохастичний градієнтний спуск, є одним з найпопулярніших оптимізаторів в глибокому навчанні. Його ключова ідея полягає в оновленні ваг моделі за допомогою їхнього градієнта по відношенню до функції втрат з метою мінімізації цієї функції втрат. "Стохастичний" в назві означає, що оновлення виконуються на основі випадково обраного підмножини даних (або "міні-батча"), а не на всьому датасеті одразу. Learning rate вказує на величину кроку при оновленні ваг. Велика швидкість навчання може призвести до того, що процес навчання перестрибне мінімум функції втрат, тоді як занадто мала швидкість навчання може сповільнити процес навчання. Momentum в SGD допомагає прискорити SGD в релевантному

напрямку і пом'якшує коливання. Це досягається шляхом накопичення експоненційно згорнутого середнього градієнтів минулих кроків і використання цього середнього для оновлення ваг. Weight decay є технікою регуляризації, яка запобігає перенавчанню шляхом додавання додаткового члена до функції втрат, що штрафує великі ваги. Це допомагає зберігати ваги моделі малими і, отже, зменшує її складність, що, в свою чергу, знижує ризик перенавчання.

Визначення кількості епох. Ще одним важливим параметром є кількість епох (Epochs), яка визначає, скільки разів алгоритм буде проходити через весь набір даних. Зазвичай кількість епох вибирається таким чином, щоб балансувати між часом навчання та точністю моделі. В нашій моделі ми використовуємо кількість епох рівній 30.

Для валідації нашої моделі було обрано наступні метрики [24]:

Перетин через об’єднання або Intersection over Union (IoU) – це метрика, що використовується для виміру точності алгоритму виявлення об'єктів. IoU обчислюється за наступною формулою:

$$IoU = \frac{A_{overlap}}{A_{union}} \tag{2.1}$$

Де, $A_{overlap}$ (Area of overlap) – це площа перетину між правильною граничною рамкою (ground truth bounding box) і передбаченою граничною рамкою (predicted bounding box), а A_{union} (Area of union) – це площа об’єднання між правильною граничною рамкою і передбаченою граничною рамкою. Для бінарної класифікації, якщо IoU більше 0.5, клас класифікованого об'єкта можна визначити як true positive (TP). Для IoU нижче 0.5, відповідний клас можна позначити як false positive (FP). False negative (FN) виникає, коли модель помилково ідентифікує негативний випадок як позитивний. Навідміну від FP, коли модель визначає позитивний випадок як негативний, FN вважається гіршим варіантом.

Параметри ефективності моделі, точність (precision – P) та повнота (recall – R) можуть бути виражені як:

$$P = \frac{TP}{(TP+FP)}; \qquad R = \frac{TP}{(TP+FN)} \tag{2.2}$$

З рівняння (2.2) можна зробити висновок, що вища точність(P) означає кращу здатність моделей розрізняти негативні набори даних, тоді як вища повнота(R) означає кращу здатність виявляти позитивні набори даних. Середня точність (average precision – AP) дорівнює площі під PR-кривою, яку можна виразити як:

$$AP = \int_0^1 P(R)dR. \tag{2.3}$$

Вища AP відповідає більшій площі під кривою PR, що свідчить про кращу точність передбачення класу об'єкта, тоді як mean average presision(mAP) є середнім значенням усіх AP, яке можна виразити як:

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i. \tag{2.4}$$

На рисунку 2.10 зображено графіки метрики, отримані в ході тренування та валідації нашої моделі, де синім кольором позначено графік val_map_05_95 –mAP при IoU між 0.5:0.95, а оранжевим кольором: val_map_05 – mAP для IoU > 0.5.

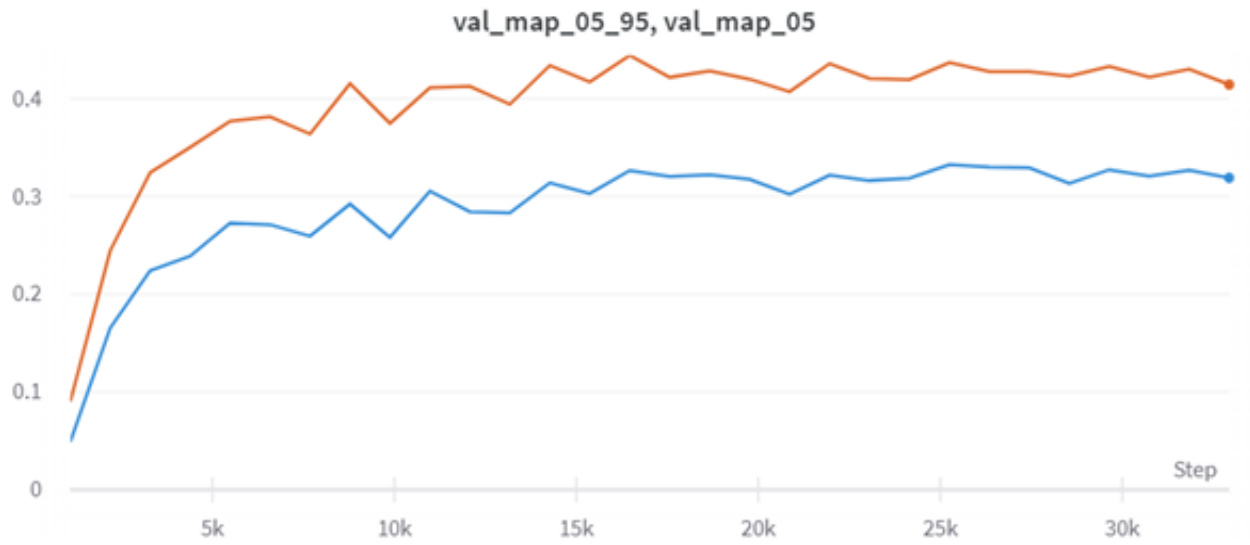


Рисунок 2.10 – Графік mean average precision(mAP)

На основі отриманих результатів можна зробити висновок, що наша модель нейронної мережі Faster R-CNN FPN показала себе досить ефективною у задачі виявлення захворювань на зображеннях рослин. Метрика mAP (середня точність по всіх категоріях) становила 0.34 для IoU 0.5:0.95 і 0.44 для IoU > 0.5, що свідчить про високий рівень точності виявлення. Ще більш важливо, що ці результати були досягнуті лише після 24 епох навчання, що свідчить про ефективність використання переднавчання (transfer learning) та велику взаємодію моделі з даними. Відповідно, зберігаючи ваги найкращої моделі, ми маємо хорошу вихідну точку для подальших вдосконалень та експериментів. Отже, в цілому, наші результати демонструють, що нейронна мережа Faster R-CNN FPN може бути ефективним інструментом для виявлення та класифікації захворювань рослин на основі зображень, і це відкриває широкі можливості для подальшого використання і вдосконалення даного підходу.

2.3. Обґрунтування вибору інструментальних засобів для програмної реалізації застосунку.

Наша система складається з трьох основних компонентів: частини для розробки та тренування нейронної мережі, серверної частини і клієнтської

частини. Для розробки та тренування моделі нейронної мережі ми використали ряд інструментів та технологій.

Python - це високорівнева, інтерпретована мова програмування, яка відома своєю простотою, зрозумілістю та широким спектром функцій. Вона надає багато утиліт та бібліотек, які значно спрощують розробку та реалізацію нейронних мереж. Python має велику підтримку спільноти розробників, що дозволяє швидко отримати допомогу, розробляти швидше та збільшувати продуктивність роботи. Також, він має багато бібліотек для машинного навчання та глибокого навчання, таких як TensorFlow, Keras, PyTorch, що дозволяють ефективно використовувати готові моделі та алгоритми для тренування наших нейронних мереж. Використання Python у нашій системі дозволяє нам розробляти та тренувати нейронну мережу зручним і ефективним способом, забезпечуючи високу якість та швидкість розробки [25].

PyTorch - це відкрите програмне забезпечення з відкритим вихідним кодом, яке є одним з найпопулярніших фреймворків глибокого навчання. Він базується на мові програмування Python і надає зручні інструменти для розробки та тренування нейронних мереж. PyTorch пропонує динамічну обчислювальну графіку, що дозволяє розробникам створювати та модифікувати моделі на льоту, що робить його особливо підходящим для дослідницьких проектів. Він також надає багато практичних інструментів для ефективного тренування нейронних мереж, включаючи автоматичне диференціювання, оптимізацію та розподілене навчання на багатоядерних системах або кластерах. У нашій системі ми використовуємо PyTorch для розробки та тренування нашої регіональної згорткової нейромережі для виявлення та класифікації захворювань рослин. PyTorch надає нам потужні інструменти для побудови та налаштування нейронних мереж, а також розширені можливості для оптимізації та навчання моделей на великих обсягах даних. Завдяки простоті використання та гнучкості PyTorch ми можемо ефективно використовувати його для реалізації складних алгоритмів,

проводити експерименти з різними архітектурами нейронних мереж та швидко приступати до пошуку оптимального рішення для виявлення та класифікації захворювань рослин у нашій системі [26].

OpenCV — (Open Source Computer Vision Library) - це бібліотека відкритого програмного забезпечення, призначена для обробки зображень і комп'ютерного зору. Вона надає розширений набір функцій і алгоритмів для роботи зі зображеннями та відео, що робить її популярною в галузі комп'ютерного зору та розпізнавання образів. OpenCV підтримує різноманітні завдання, такі як зчитування та запис зображень та відео, обробка зображень, детекція об'єктів, вимірювання, сегментація, відстеження об'єктів, калібрування камер, реконструкція 3D-сцен, а також багато інших. OpenCV має широкий спектр підтримуваних мов програмування, включаючи C++, Python, Java і MATLAB, що робить її доступною для розробників з різними уподобаннями. У нашій системі ми використовуємо OpenCV для реалізації функціоналу, пов'язаного з обробкою та аналізом зображень рослин. За допомогою OpenCV ми можемо виконувати операції над зображеннями, які включають фільтрацію, виокремлення особливостей, виявлення контурів, зміну розміру та багато інших. Вона допомагає нам підготувати дані перед використанням їх у нейронній мережі та забезпечує потрібний функціонал для підтримки процесу виявлення та аналізу захворювань рослин у нашому програмному продукті [27].

Scikit-learn — це бібліотека відкритого програмного забезпечення для машинного навчання, яка надає широкий спектр алгоритмів і інструментів для розв'язання завдань класифікації, регресії, кластеризації, а також для виконання вимірювань, валідації та підготовки даних. Scikit-learn побудована на основі інших популярних бібліотек Python, таких як NumPy, SciPy і matplotlib, і надає високорівневий інтерфейс для розробки та виконання різних алгоритмів машинного навчання. Вона підтримує різні методи навчання, такі як метод опорних векторів (SVM), дерева рішень, наївний Баєс, ансамблеві методи (наприклад, random forest), нейронні мережі та багато інших. Scikit-

learn також має багато корисних функцій для обробки та підготовки даних, таких як вимірювання, валідація, вибір моделей, векторизація тексту, видалення шуму, шкалювання даних та інші. Крім того, вона надає зручні інструменти для оцінки результатів моделей та визначення оптимальних параметрів моделі шляхом пошуку по сітці (grid search). У нашій системі ми використовуємо scikit-learn для реалізації деяких алгоритмів машинного навчання, які допомагають у класифікації та аналізі захворювань рослин. Scikit-learn дозволяє нам ефективно застосовувати різні алгоритми класифікації, регресії та кластеризації, а також забезпечує зручні функції для оцінки та налаштування моделей машинного навчання. Використання scikit-learn дозволяє нам швидко та ефективно розробляти та валідувати нашу модель машинного навчання для виявлення та класифікації захворювань рослин у нашому програмному продукті [28].

Pillow - це бібліотека Python для обробки зображень. Вона надає широкий спектр функцій для завантаження, маніпуляції, обрізки, розмірів, збереження та відображення зображень у різних форматах. Pillow є розширенням бібліотеки Python Imaging Library (PIL), але з простішим інтерфейсом та покращеною підтримкою для сучасних форматів зображень. За допомогою Pillow, ми можемо виконувати різні операції зображення, такі як зміна розміру, обрізка, поворот, зміна кольору, прозорості та контрастності, застосування фільтрів та ефектів, об'єднання та розділення зображень, роботу з пікселями та багато іншого. Pillow також підтримує роботу з різними форматами зображень, включаючи JPEG, PNG, BMP, TIFF, GIF та інші. Вона надає можливість завантажувати зображення з файлів або URL, а також зберігати зображення в різних форматах. У нашій системі ми використовуємо Pillow для обробки та маніпуляції зображень рослин, зокрема для підготовки та обрізки зображень перед подачею їх на вхід у модель нейронної мережі. Завдяки зручному та потужному інтерфейсу Pillow, ми можемо легко виконувати різноманітні операції зображень, що допомагає нам забезпечити якісну та точну обробку зображень у нашій системі [29].

NumPy (Numerical Python) - це основна бібліотека для наукових обчислень у мові програмування Python. Вона надає підтримку для роботи з багатовимірними масивами, а також функції для виконання різноманітних математичних операцій на цих масивах. Одним з ключових об'єктів у NumPy є масив NumPy (NumPy array), який представляє собою таблицю елементів одного типу даних, індексовану за допомогою неузагальненого цілого числа. Масиви NumPy є ефективними структурами даних, що дозволяють здійснювати швидкі та ефективні обчислення над великими об'ємами даних. За допомогою NumPy ми можемо виконувати операції лінійної алгебри, статистики, випадкових чисел, перетворень Фур'є, операцій з векторами та матрицями, індексацію та вибірку даних, обробку зображень та багато іншого. Вона також надає потужні функції для роботи з файлами, вводу-виводу та інтеграції з іншими бібліотеками наукових обчислень. У нашій системі ми використовуємо NumPy для ефективного представлення та обробки даних, зокрема числових даних, таких як числові ознаки, використовувані для навчання моделей. Ми можемо виконувати різні операції над цими даними, включаючи математичні операції, статистичний аналіз, масштабування та перетворення даних, що допомагає нам підготувати дані для подальшої обробки та аналізу. Завдяки широкому функціоналу NumPy, ми можемо реалізувати ефективні та потужні операції обчислення в нашій системі [30].

Розробка серверної частини здійснюється за допомогою фреймворку Django та бази даних SQLite.

Django є високорівневим веб-фреймворком, написаним на мові програмування Python, який дозволяє швидко та ефективно розробляти складні веб-додатки. Він надає потужні інструменти для створення масштабованих веб-сайтів та веб-додатків з високим рівнем безпеки. Одна з ключових особливостей Django - це його модульність та зручна архітектура. Фреймворк пропонує розбити функціональність веб-додатка на окремі компоненти, такі як моделі бази даних, представлення (views) та шаблони (templates). Це дозволяє розробникам працювати над окремими частинами

проекту ізольовано, спрощує розширення та підтримку додатків. Django також надає вбудований ORM (Object-Relational Mapping), що дозволяє взаємодіяти з базою даних, використовуючи об'єктно-орієнтований підхід замість написання складних SQL-запитів. Це спрощує роботу з даними та забезпечує переносимість між різними системами управління базами даних. Фреймворк також має багатий набір вбудованих функціональних можливостей, таких як аутентифікація та авторизація користувачів, кешування, обробка форм, міжнародна підтримка, адміністративний інтерфейс та багато іншого. У нашій системі ми використовуємо Django для розробки серверної частини. Він надає потужні інструменти для створення API, обробки запитів, маршрутизації URL, роботи з базою даних та багато іншого. Django допомагає нам швидко та ефективно створювати стабільні та масштабовані веб-додатки, забезпечуючи нам потрібну функціональність та безпеку [31].

SQLite – це легка вбудовувана реляційна база даних, яка зберігає всю свою інформацію у локальних файлових системах. Вона використовується для зберігання та керування структурованою інформацією, такою як дані користувачів, конфігураційні параметри, журнали тощо. SQLite приваблює своєю простотою використання та невимогливістю до ресурсів. Вона не вимагає окремого сервера або інфраструктури, оскільки база даних знаходиться в одному файлі на локальній машині. Це робить її ідеальним вибором для малих проєктів або додатків, які не потребують великої потужності бази даних. SQLite підтримує стандартні SQL-запити та операції, що дозволяє взаємодіяти з базою даних через різні мови програмування. Це дозволяє розробникам легко виконувати запити до бази даних, створювати, змінювати та видаляти дані. Будучи легкою та швидкою базою даних, SQLite є популярним вибором для мобільних додатків та веб-сайтів з невеликим обсягом даних. Вона дозволяє ефективно керувати інформацією без необхідності складних налаштувань або великої інфраструктури бази даних [32].

Функціональна частина клієнтської системи виконана мовою JavaScript.

JavaScript — це високорівнева, інтерпретована мова програмування, що використовується для розробки веб-додатків. Вона дозволяє додавати динамічність та взаємодію на стороні клієнта веб-сторінок. JavaScript може бути вбудований безпосередньо в HTML-код сторінки або використовуватися зовнішніми файлами скриптів. Вона надає можливості для керування подіями, маніпулювання DOM-структурами, обробки форм, валідації даних та багатьох інших функцій, які забезпечують більш динамічні та інтерактивні веб-сторінки. У нашому проекті ми використовуємо JavaScript для інтерактивності та валідації вхідних зображень, а також для обробки подій натискання на кнопки [33].

2.4. Структурно-функціональний аналіз процесів в інформаційній системі.

З метою кращого розуміння процесів, пов'язаних з розробкою нашого програмного продукту, який використовує регіональну згорткову нейронну мережу для виявлення та класифікації захворювань рослин, було вирішено використовувати метод структурно-функціонального моделювання SADT, контекстну діаграму IDEF0. Це допоможе нам більш детально проаналізувати та уявити процеси та функції, які залучені у розробку нашого програмного продукту.

Методика SADT (Structured Analysis and Design Technique), яка також відома як IDEF0, це графічний інструмент, який дозволяє аналізувати та проектувати системи з багатьма взаємозалежними процесами. Він включає в себе набір засобів та методів для моделювання, аналізу, проектування та оптимізації систем. Модель IDEF0/SADT складається з блоків, які відображають процеси та створюють структуру "дерева". Кожен блок може бути розбитий на дрібніші процеси для подальшого аналізу. Цей процес рекурсивно повторюється до того моменту, поки не буде досягнуто потрібного рівня деталізації.

Кожний блок в моделі IDEF0/SADT має наступні компоненти:

– Вхід - дані, що використовуються у процесі.
– Вихід - результат процесу.
– Ресурси - ресурси, що використовуються в процесі.
– Контроль - обмеження або вимоги до процесу.

Основним процесом нашої інформаційної системи є «Розпізнавання захворювань рослин за зображенням листя».

Діаграму моделювання процесу "Розпізнавання захворювань рослин за зображенням листя" зображено на рисунку 2.11.

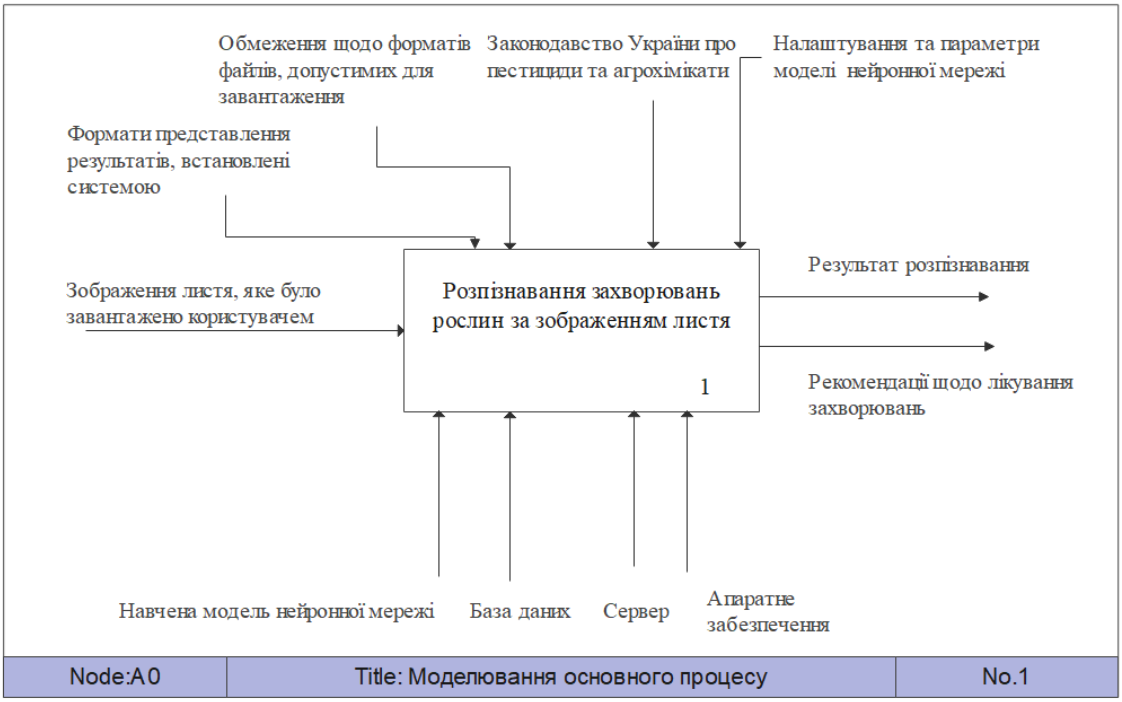


Рисунок 2.11 – Процес "Розпізнавання захворювань рослин за зображенням листя" у вигляді контекстної діаграми IDEF0

Вхідними даними є: зображення листя, яке завантажує користувач.

Вихідними даними є: результат розпізнавання та рекомендації щодо лікування захворювання.

Обмежуючими даними зверху є: законодавство України про пестициди та агрохімікати, обмеження щодо форматів файлів, формати представлення

результатів, встановлені системою, допустимих для завантаження, а також налаштування та параметри моделі нейронної мережі.

Ресурсами функціонування знизу являються: база даних, сервер на якому працює додаток, апаратне забезпечення та попередньо навчена модель нейронної мережі для виконання прогнозів.

З метою отримання більш детального розуміння внутрішніх процесів нашої системи, необхідно розкрити їх за допомогою декомпозиції IDEF0 на більш деталізовані діаграми нижчого рівня. Це дозволить нам докладніше розглянути та представити складові елементи та взаємозв'язки в межах цих процесів. Основний процес було розбито на декілька функціональних блоків: завантаження зображення, створення прогнозів, виведення результату (рис. 2.12).

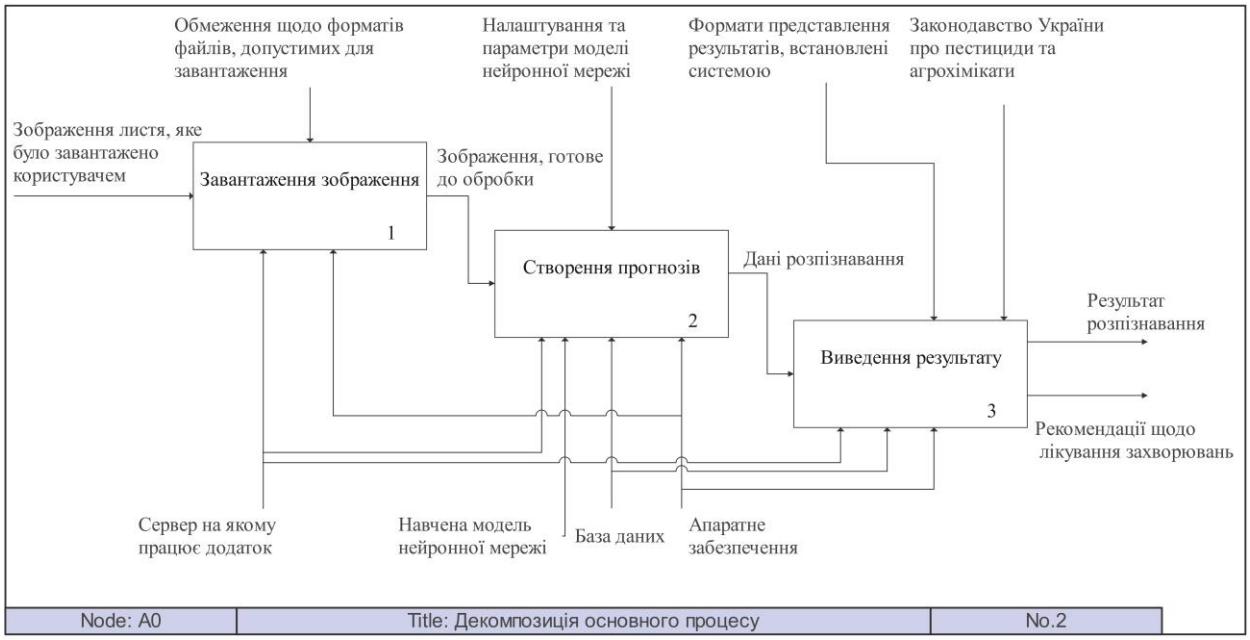


Рисунок 2.12 – Декомпозиція процесу "Розпізнавання захворювань рослин за зображенням листя" у вигляді контекстної діаграми IDEF0

Процес "Завантаження зображення" в контексті IDEF0 описує дії, пов'язані з перенесенням обраного користувачем файлу зображення у систему. Вхідним елементом цього процесу є сам файл зображення, який користувач вибирає для завантаження. Вихідним елементом є вже завантажене зображення, яке є готовим до подальшого аналізу. Щодо ресурсів, вони

включають сервер програмного додатку, що відповідає за обробку завантаження файлів, та апаратне забезпечення користувача (таке як комп'ютер або мобільний пристрій), що використовується для вибору та надсилання файлу. Механізми контролю для цього процесу включають системні обмеження на допустимі формати файлів та їх розміри. Цей процес може бути деталізований шляхом поділу на кілька підпроцесів, таких як вибір файлу користувачем, надсилання файлу на сервер, перевірка файлу на допустимість, і збереження файлу в системі для подальшого аналізу.

Процес "Створення прогнозів" в контексті IDEF0 описує аналіз завантаженого зображення з метою виявлення потенційних ознак хвороб. Вхідним елементом для цього процесу є зображення, що було завантажено в систему. Вихідним елементом є результат розпізнавання, що включає зображення із виявленими ознаками захворювань та відповідними прогнозами. Щодо ресурсів, вони включають навчену модель нейронної мережі, яка використовується для аналізу зображення, і сервер, на якому виконується процес аналізу. Механізми контролю для цього процесу включають налаштування та параметри моделі нейронної мережі, що визначають характеристики та обмеження її роботи. Цей процес може бути деталізований шляхом поділу на кілька підпроцесів, таких як препроцесинг зображення (наприклад, нормалізація, масштабування тощо), застосування моделі нейронної мережі до зображення, інтерпретація результатів, і формування прогнозу на основі отриманих результатів.

Процес "Виведення результату" у контексті IDEF0 описує подання результатів розпізнавання та прогнозу користувачу. Вхідними елементами для цього процесу є результати розпізнавання захворювань рослин та сформовані прогнози, що були отримані на основі аналізу завантаженого зображення. Вихідними елементами є результати, які відображаються користувачу, включаючи відображення зображення із виявленими ознаками захворювань та рекомендації щодо лікування можливих захворювань. Ресурсами для цього процесу є сервер програмного додатку, що відповідає за відображення

результатів, та інтерфейс користувача, через який ці результати подаються користувачу. Механізми контролю для цього процесу можуть включати вимоги до формату та стилю представлення результатів, встановлені системою, а також закон України про пестициди та агрохімікати, яке обмежує формування рекомендацій щодо лікування захворювань. Деталізація цього процесу може включати підпроцеси, такі як форматування результатів для відображення, передача результатів до інтерфейсу користувача, і відображення результатів користувачу через інтерфейс додатку.

Структурно-функціональний аналіз за допомогою методики SADT та нотації IDEF0 був проведений із метою глибокого розуміння процесів у розроблюваному веб-додатку для розпізнавання захворювань рослин. В результаті аналізу, ми виявили та детально описали основні функції системи, визначили їх взаємозв'язки та залежності. Використання даного інструментарію дозволило наочно відобразити структуру системи та розкрити процеси, що відбуваються на різних рівнях. Декомпозиція основних процесів сприяла зрозумінню їх складових та взаємодії з іншими елементами системи. Такий аналіз створює чітке розуміння того, як система має функціонувати та в який спосіб відбувається обробка інформації.

2.5. Опис інформаційної архітектури нейромережного застосунку розпізнавання захворювань рослин.

Наша система складається з трьох частин: серверної частини, клієнтської частини та частини навчання системи (рис. 2.13). Основний обмін даними між серверною та клієнтською частинами відбувається за допомогою REST API через протокол HTTP. REST API - це набір правил, що визначають, як клієнти можуть отримувати доступ до даних та взаємодіяти з ними. Він дозволяє створювати, читати, оновлювати та видаляти ресурси за допомогою стандартних HTTP-запитів, таких як GET, POST, PUT і DELETE. Клієнтська частина відправляє запит на сервер через REST API, сервер обробляє цей запит, виконує необхідні дії та відправляє відповідь назад до клієнта.

Взаємодія між серверною частиною та частиною навчання системи здійснюється через завантаження ваг навченої моделі.

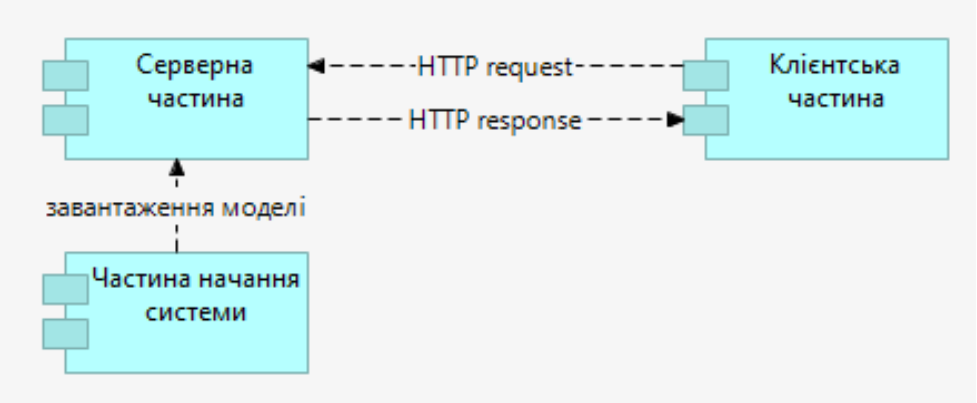


Рисунок 2.13 – Діаграма взаємодії додатків системи

Після того, як модель була навчена і збережена, серверна частина завантажує цю модель із локального сховища і використовує її для обробки запитів від користувачів. Така структура та взаємодія компонентів дозволяє створити ефективну, гнучку та масштабовану систему, яка може адаптуватися до різних потреб користувачів і виконувати велику кількість різноманітних завдань.

Наша клієнтська частина відповідає за представлення веб-інтерфейсу, проведення перевірки вхідних файлів та реагування на взаємодії користувача, такі як натискання кнопок. Відповідну діаграму зображено на рисунку 2.14.

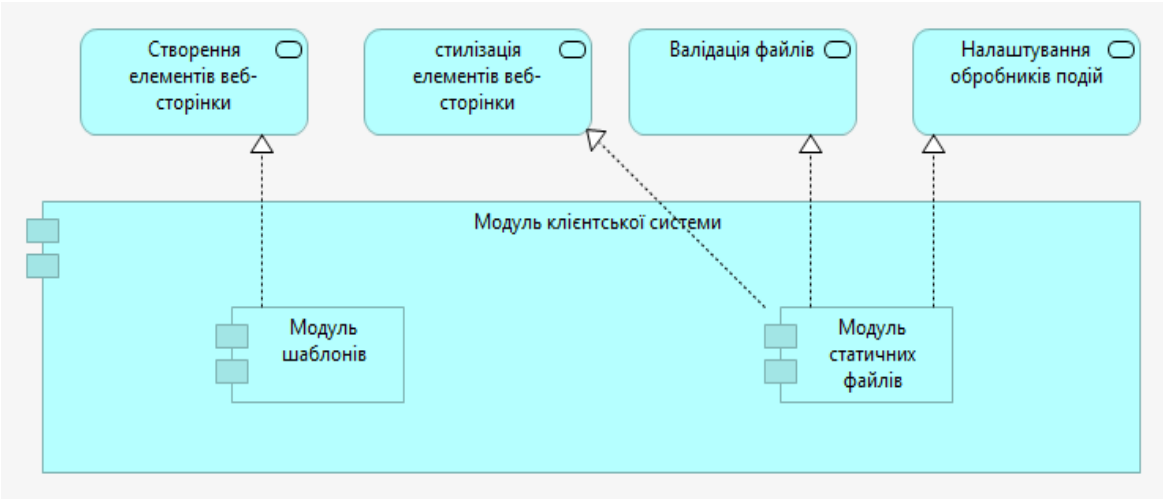


Рисунок 2.14 – Діаграма представлення структури клієнтської частини

Ця частина системи має структуровану організацію, яка складається з кількох модулів:

- Модуль шаблонів - це сховище html-шаблонів, що формують візуальне представлення веб-сторінок нашого сайту. Шаблони дозволяють створювати консистентний та привабливий дизайн, що підвищує зручність користування сервісом.
- Модуль статичних файлів - цей модуль включає в себе файли стилів, які визначають оформлення веб-сторінок, а також різні файли, необхідні для забезпечення функціонування додатку. Тут знаходяться файли зображень, скрипти JavaScript, файли CSS та інші ресурси, що підвищують функціональність та естетику нашого веб-сайту.

Серверна частина нашої системи відповідає за обробку вхідних запитів, управління взаємодією з модулем розпізнавання та обробку зображень для подальшого розпізнавання. Вона також відповідає за роботу з базою даних та надсилання відповідей користувачам. Відповідну діаграму зображено на рис. 2.15.

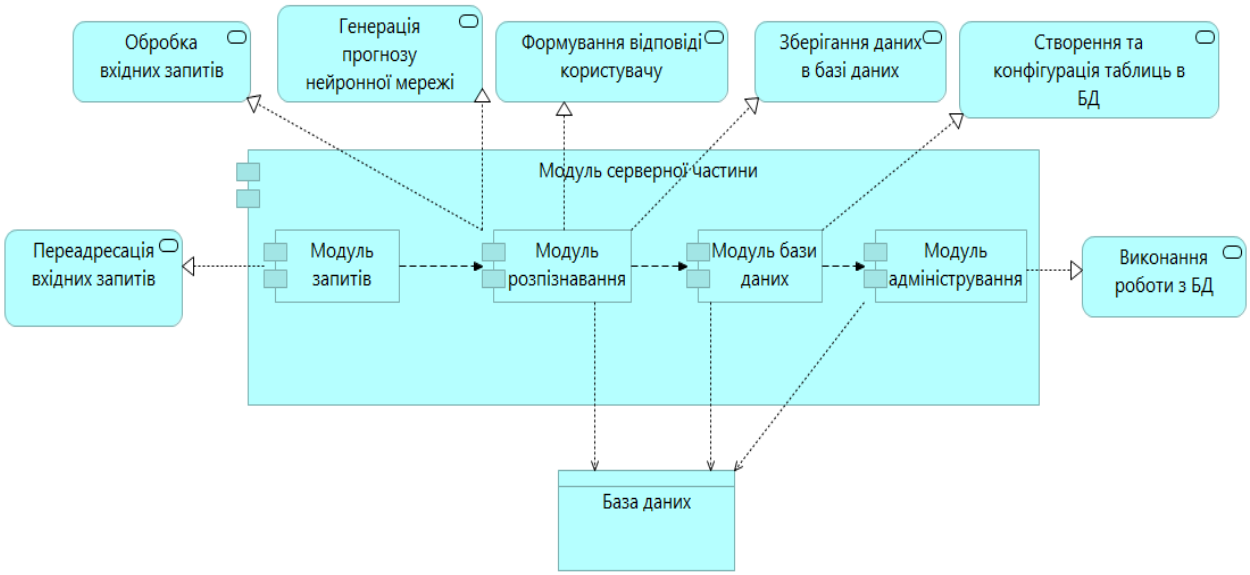


Рисунок 2.15 – Діаграма представлення структури сереверної частини

Структура серверної частини включає в себе наступні модулі:

- Модуль запитів - цей модуль приймає вхідні URL-запити, обробляє їх, переадресовує до відповідних служб та передає для подальшої обробки у модуль розпізнавання.
- Модуль розпізнавання - в цьому модулі відбувається завантаження попередньо натренованої моделі нейронної мережі, виконання прогнозування на основі вхідних зображень, створення анотацій та обмежуючих рамок, що виділяють об'єкти на зображенні. Зберігання результатів розпізнавання в базі даних та формування відповіді зі встановленим діагнозом та рекомендаціями щодо лікування.
- Модуль бази даних - цей модуль відповідає за створення та конфігурацію таблиць у нашій базі даних, що дозволяє систематизувати та ефективно зберігати всю необхідну інформацію.
- Модуль адміністрування - в цьому модулі відбувається управління існуючою базою даних, що забезпечує гнучкість та високу продуктивність системи.

Частина навчання системи відповідає за завантаження та попередню обробку даних для навчання системи, створення моделі нейронної мережі, навчання, налаштування гіперпараметрів та валідацію нейронної мережі. Також тестування моделі та збереження її в локальному сховищі.

Розробимо структуру частини навчання, вона буде складатися з наступних модулів:

- Модуль тренування моделі, в якому буде здійснюватися завантаження та попередня обробка даних для навчання, створюватися модель нейронної мережі, виконуватися її навчання, валідація та збереження моделі в локальному сховищі.
- Модуль тестування, де буде здійснюватися тестування створеної моделі.

Відповідну діаграму представлено на рис. 2.16.

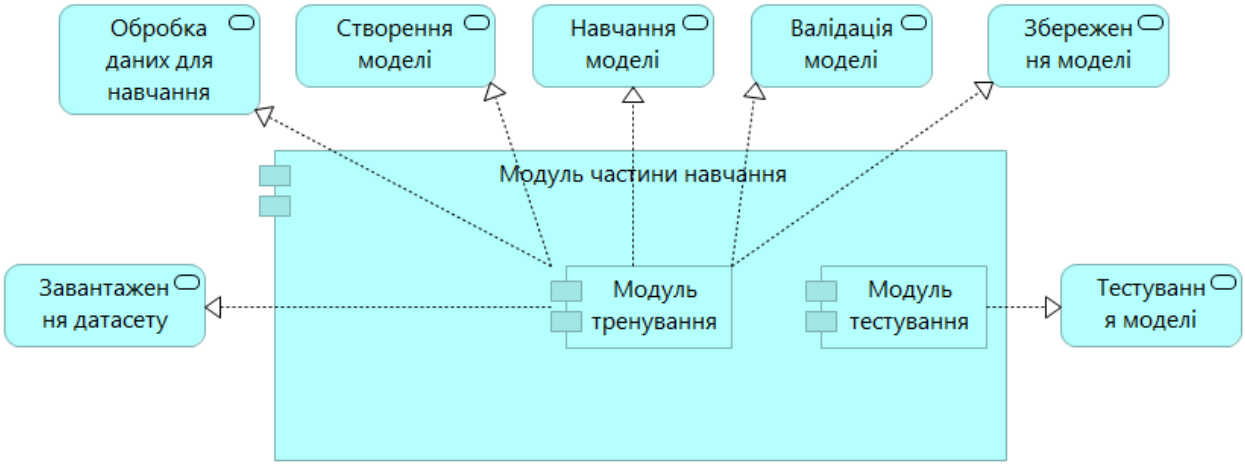


Рисунок 2.16 - Діаграма представлення структури частини навчання

Таблиця "PredictionModel" є ключовою структурою даних в базі даних нашої системи (рис. 2.17). Її призначення полягає в зберіганні запитів користувачів та пов'язаних з ними даних. Вона складається з наступних полів:

- "id" - це унікальний ідентифікатор запиту, представлений цілочисельним типом даних. Воно є первинним ключем в таблиці.
- "name" - це текстове поле, в якому зберігається назва вхідного файлу з зображенням.
- "image" - це спеціалізоване поле для зберігання зображень. Воно містить зображення з локалізованими областями захворювань, визначеними класами цих захворювань та відповідними впевненостями прогнозів.
- "class_indexes" - це текстове поле, в якому зберігається перелік унікальних індексів класів захворювань. Воно потрібне для виведення відповідних рекомендацій для кожного виявленого захворювання.
- "timestamp" - це поле дата-час, в якому фіксується дата та час створення прогнозу.

– "owner" - це цілочисельне поле з зовнішнім ключем, яке посилається на запис користувача в вбудованій таблиці "User" Django. Воно вказує на те, хто створив конкретний запит.
– "session_id" - це текстове поле, в якому зберігається унікальний ідентифікатор сесії. Воно дозволяє відслідковувати запити, зроблені анонімними користувачами під час поточної сесії.

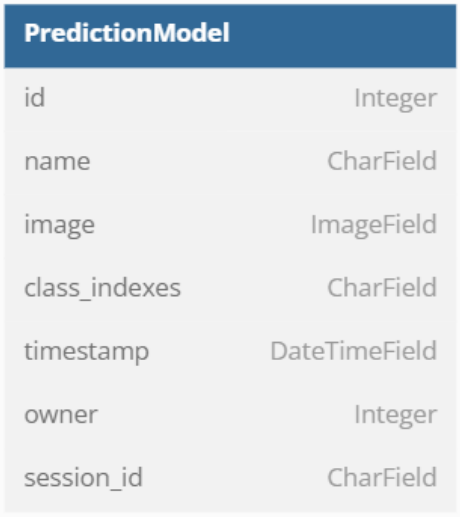


Рисунок 2.17 – Таблиця PredictionModel в БД

Висновки до другого розділу

У другому розділі роботи було проведено комплексний аналіз системи. було проаналізовано варіанти використання системи та побудовано діаграму варіантів використання системи. Також було проведено структурно-функціональний аналіз SADT. Це дало можливість детально проаналізувати процеси, які відбуваються в системі, визначити їх взаємозв'язок та вплив на результативність системи. Було вибрано й обґрунтовано використання інструментальних засобів для розробки програми. Важливо відмітити, що обраний інструментарій дозволив нам створити високопродуктивну систему, яка може легко масштабуватись та адаптуватись до змінюваних умов. Було описано створення моделі нейронної мережі. Створена нейронна мережа Faster R-CNN FPN виявила себе як ефективне рішення для визначення захворювань на рослинах на основі зображень. Її метрики показують гарну

точність. Було розроблено інформаційну архітектуру системи та детально пояснено взаємодію компонентів в системі. Також було спроектовано базу даних для зберігання прогнозів користувачів. Завдяки цьому можна зберігати історію діагностованих захворювань та використовувати цю інформацію для виведення історії прогнозів користувачам або для подальшого аналізу та вдосконалення системи. Розроблений нами додаток відповідає всім заздалегідь встановленим функціональним вимогам, включаючи здатність завантажувати файли двома шляхами, валідувати розширення вхідних файлів, перевіряти цілісність вхідних файлів та переглядати історію створених прогнозів. Логіка взаємодії користувача з додатком відповідає нашим початковим вимогам і забезпечує зручне та ефективне використання системи. Користувач може використовувати додаток для виявлення захворювань рослин на основі зображень, отримуючи точні й швидкі результати.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ НЕЙРОМЕРЕЖНОГО ЗАСТОСУНКУ РОЗПІЗНАВАННЯ ЗАХВОРЮВАНЬ РОСЛИН.

3.1. Розробка і реалізація інтерфейсу користувача.

Створений веб-застосунок складається з трьох основних сторінок: "Головна", "Прогноз" та "Мої прогнози". Кожна сторінка містить навігаційну панель, що включає посилання "Головна" та "Мої прогнози" для зручного переходу між різними частинами сайту. При першому відвідуванні сайту користувача вітає головна сторінка (див. рис. 3.1). Ця сторінка містить вікно для завантаження файлів. У цьому вікні присутній спеціально виділений прямокутник (так звана "дроп-зона"), де користувач може перетягнути зображення для його завантаження. Альтернативно, користувач може використати кнопку "Завантажити файл", яка відкриває стандартне діалогове вікно для вибору файлу в системі. Обидва варіанти завантаження зображень передбачають процедуру валідації вхідних даних, яка перевіряє, чи відповідає файл дозволеним форматам зображень. Список дозволених форматів включає розширення: .jpg, .jpeg, .png, .svg, .jfif та .bmp.

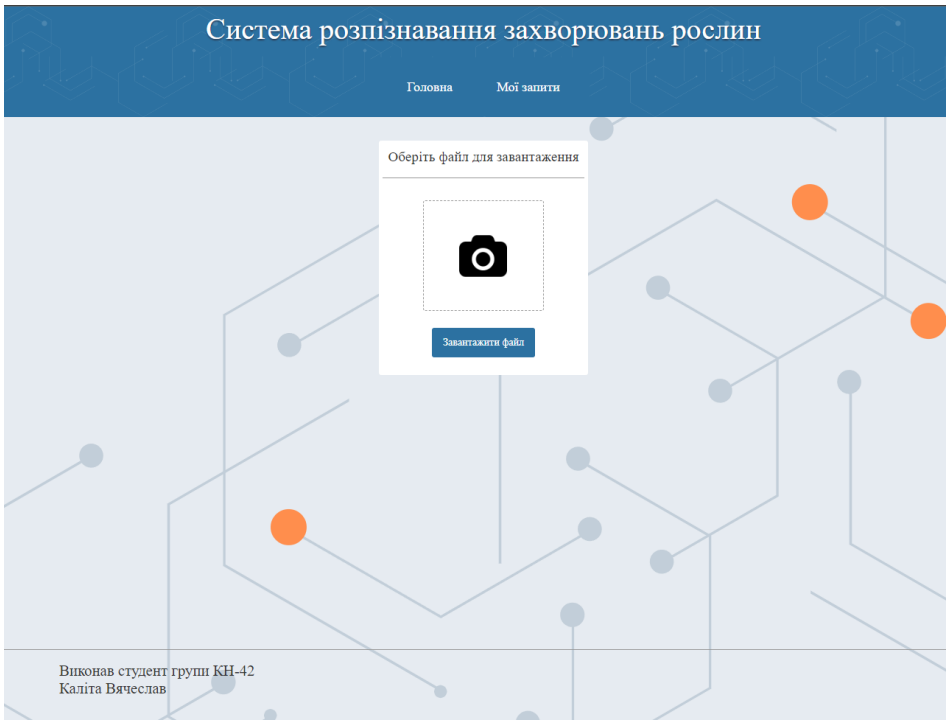


Рисунок 3.1 – Головна сторінка додатку

З моменту, коли користувач завантажує файл, система автоматично перенаправляє його до сторінки прогнозування. Ця сторінка продовжує відображати вікно для завантаження файлу, надаючи можливість користувачу відправити інше зображення для аналізу нейронною мережею, якщо він цього бажає. Прямо під вікном завантаження файлу розміщено вікно, що відображає результати аналізу зображення (див. рис. 3.2).

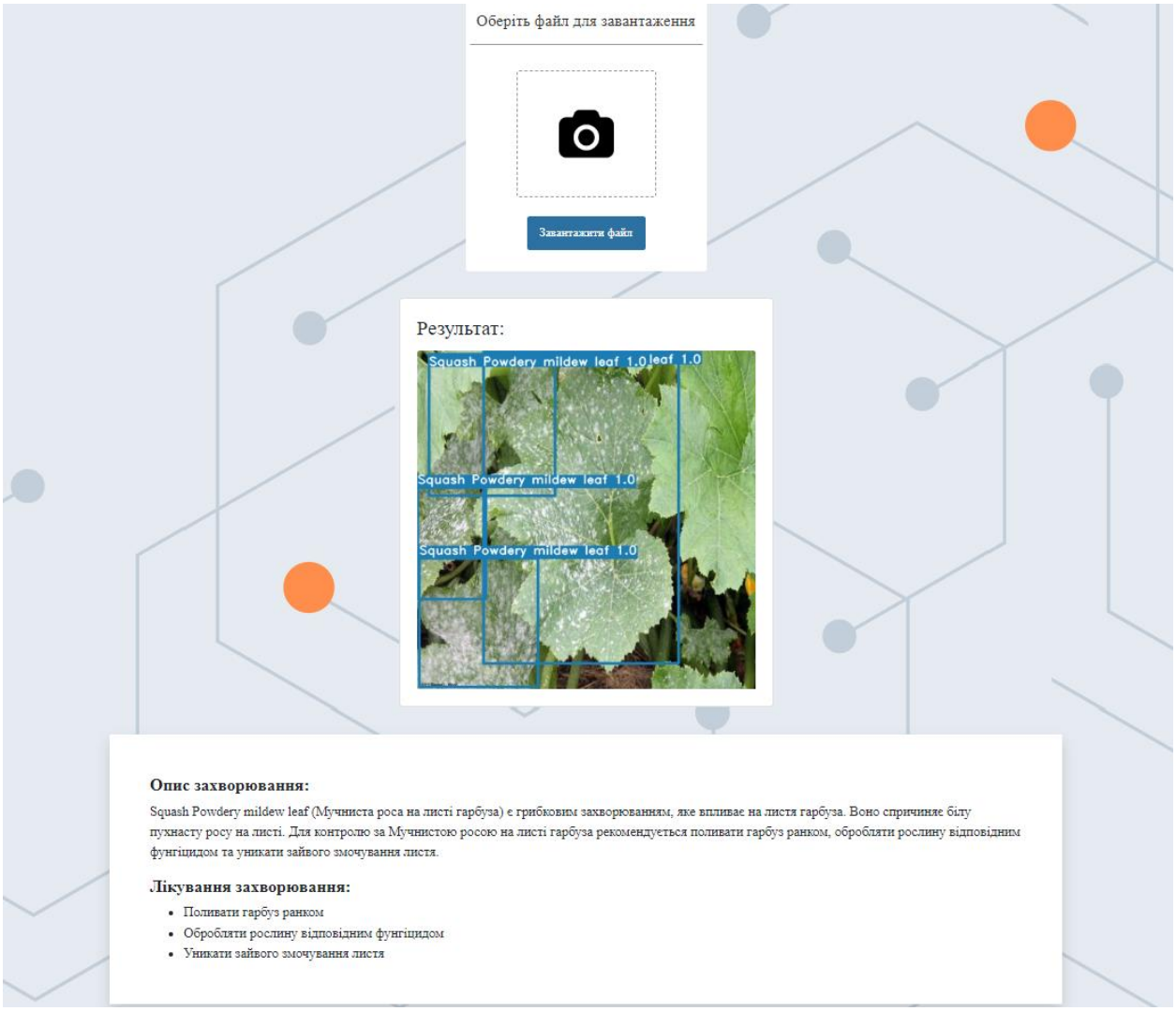


Рисунок 3.2 – Відображення сторінки з прогнозом

Тут відображається завантажене зображення, на якому спрогнозовані об'єкти виділені прямокутниками. Біля кожного прямокутника виводиться відповідний клас об'єкта та ступінь впевненості в результатах прогнозу. У динамічному порядку генеруються додаткові вікна, число яких відповідає кількості унікальних класів об'єктів, виявлених на зображенні. Для кожного

визначеного класу рослин, відображається відповідний опис стану рослини, або загальні відомості, якщо рослина виявилася здоровою. Наведені також можливі дії для лікування рослини, якщо виявлено хворобу. Описи станів рослин і рекомендації щодо їх усунення містяться в файлі info.csv, який включає наступні колонки:

- index – це індекси для всіх можливих станів рослин;
- disease_name – назва захворювання англійською мовою;
- description – це детальний опис стану рослини, або загальна інформація, якщо рослина здорова;
- possible_steps – це перелік можливих дій для лікування захворювання.

При натисканні кнопки «Мої прогнози» в навігаційному меню, користувач потрапляє на сторінку з історією прогнозів. На цій сторінці представлений перелік усіх отриманих користувачем прогнозів в рамках поточної сесії. Кожний елемент цього списку відображається у вигляді окремого вікна, де вказані назва завантаженого файлу та дата його завантаження. Додатково, в цьому вікні показано зображення з належними до нього прогнозами. Відображення прогнозів відбувається в порядку від нового до старого, де найсвіжіший прогноз розміщений на початку. Вікна з прогнозами генеруються динамічно на сторінці, а вся інформація про них зберігається в базі даних системи у таблиці користувацьких прогнозів. Вони зберігаються за унікальним ключем сесії, створеним для кожного користувача в поточній сесії. Така структура забезпечує, що прогнози відображаються для відповідного користувача і не навантажують систему. У кінці сторінки розташована кнопка "Назад", що дозволяє користувачу легко повернутися на головну сторінку, уникаючи прокручування сторінки вгору (див. рис. 3.3).

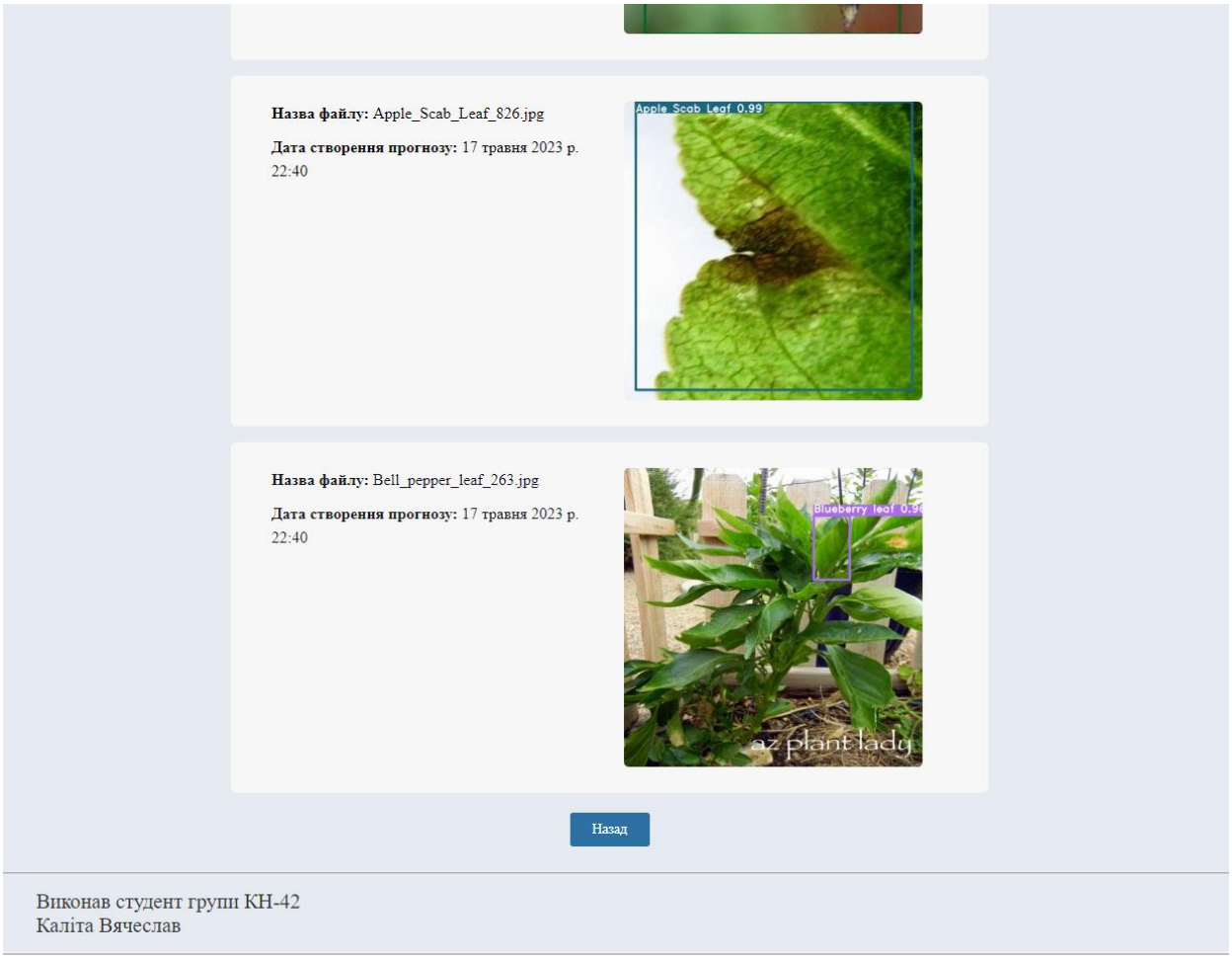


Рисунок 3.3 – Сторінка з відображенням історії прогнозів

Користувач має змогу переглянути деталізовану інформацію про будь-який зі створених раніше прогнозів. Для цього потрібно вибрати відповідний елемент у списку прогнозів, натиснувши на нього. Після цього система перенаправить користувача на сторінку, що містить конкретний прогноз. Доступ до цього прогнозу реалізований через унікальний ідентифікатор "id", який присвоєно кожному прогнозу в базі даних системи. На відповідній сторінці відображається вікно, яке містить зображення рослин із спрогнозованими станами здоров'я. Окрім цього, в нижній частині сторінки представлені описи виявлених станів і рекомендації щодо їх лікування. (див. рис. 3.4).

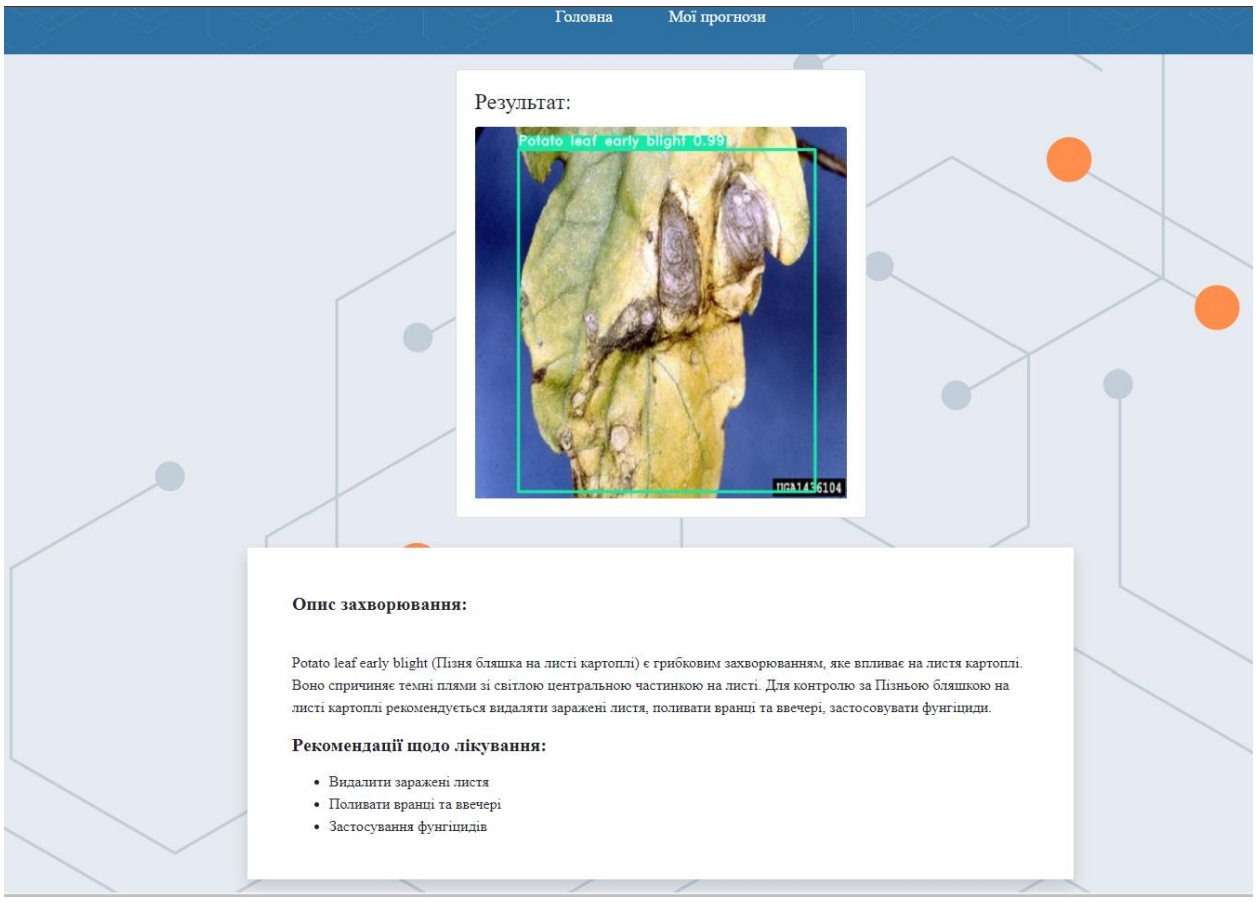


Рисунок 3.4 – Сторінка з відповідним прогнозом

3.2. Тестування працездатності додатку в процесі розв’язку поставленої задачі.

Проведемо тестування функціональних вимог методом чорного ящика. Тестування методом чорного ящика — це метод тестування програмного забезпечення, в якому аналізується реакція системи на вхідні дані, не вдаючись в аналіз внутрішнього механізму роботи програми. Суть цього методу в тому, що тестер розглядає програму як "чорний ящик", не маючи доступу до її внутрішнього коду, структури даних або алгоритмів роботи.

В контексті нашого додатку, тестування функціональних вимог методом чорного ящика буде включати перевірку таких елементів:

1. Здатність завантажувати файли зображень і валідувати їх формати. Перевірка цієї функції може включати спробу завантажити файли різних форматів і перевірку реакції системи на недопустимі формати.

2. Відображення прогнозів і історії прогнозів. Можна перевірити, чи відображає система вірно створені прогнози, чи належать ці прогнози відповідним користувачам і чи відсортовані вони відповідно до дати.
3. Функціональність інтерфейсу користувача. Це може включати перевірку роботи кнопок, меню та інших елементів інтерфейсу.

Усі ці тести будуть виконуватися без розгляду внутрішньої реалізації програми, а лише з орієнтацією на результат роботи та коректність реагування на дії користувача. Нижче будуть представлені відповідні тест-кейси.

Перевірка завантаження файлів, їх валідації та виведення прогнозів і рекомендацій:

Таблиця 3.1 – Тест-кейс №1 перевірка коректного завантаження файлів через оглядач

Тест кейс №1	Передумова	веб-додаток відкритий в браузері, відкрита головна сторінка
1	Крок 1	Натиснути лівою кнопкою миші на область завантаження файлу
2	Крок 2	Обрати зображення рослини та натиснути «Відкрити»
3	Крок 3	Натиснути кнопку «Завантажити файл»
4	Очікуваний результат тест кейсу	Під областю завантаження виводиться назва обраного файлу, на екран виводиться зображення з результатом прогнозу, опис стану рослини та рекомендації щодо лікування

Таблиця 3.2 – Тест-кейс №2 перевірка коректного завантаження файлів через дроп-зону

Тест кейс №2	Передумова	Веб-додаток відкритий в браузері, відкрита головна сторінка, відкритий провідник на комп'ютері
1	Крок 1	Перетягнути зображення рослини з провідника у дроп-зону
2	Крок 2	Натиснути кнопку «Завантажити файл»
3	Очікуваний результат тест кейсу	Під областю завантаження виводиться назва обраного файлу, на екран виводиться зображення з результатом прогнозу, опис стану рослини та рекомендації щодо лікування

Таблиця 3.3 – Тест-кейс №3 перевірка наявності файлу

Тест кейс №3	Передумова	веб-додаток відкритий в браузері, відкрита головна сторінка, файл не обраний
1	Крок 1	Натиснути кнопку «Завантажити файл»
2	Очікуваний результат тест кейсу	В полі вибору файлу виводиться повідомлення «Оберіть файл»

Таблиця 3.4 – Тест-кейс №4 перевірка валідації формату файлу

Тест кейс №4	Передумова	веб-додаток відкритий в браузері, відкрита головна сторінка
1	Крок 1	Обрати файл недопустимого формату
2	Крок 3	Натиснути кнопку «Завантажити файл»
3	Очікуваний результат тест кейсу	На екран виводиться повідомлення: «Невірний формат файлу»

Таблиця 3.5 – Тест-кейс №5 перевірка пошкодженого зображення

Тест кейс №5	Передумова	веб-додаток відкритий в браузері, відкрита головна сторінка
1	Крок 1	Обрати файл з пошкодженим зображенням
2	Крок 3	Натиснути кнопку «Завантажити файл»
3	Очікуваний результат тест кейсу	Помилка. Неможливо обробити зображення

Перевірка відображення прогнозів і історії прогнозів:

Таблиця 3.6– Тест-кейс №6 перевірка історії прогнозів

Тест кейс №6	Передумова	веб-додаток відкритий в браузері, відкрита сторінка «Мої прогнози», користувач не отримав жодного прогнозу
1	Очікуваний результат тест кейсу	На сторінці виводиться повідомлення «Жодного прогнозу немає»

Таблиця 3.7– Тест-кейс №7 перевірка відсортовування прогнозів за датою створення

Тест кейс №7	Передумова	веб-додаток відкритий в браузері, відкрита сторінка «Мої прогнози»
1	Крок 1	Перейти на сторінку «Головна» та завантажити файл
2	Крок 2	Перейти на сторінку «Мої прогнози»
3	Очікуваний результат тест кейсу	На сторінці відображається список прогнозів, щойно створений прогноз відображається першим у списку

Таблиця 3.8– Тест-кейс №8 перевірка зберігання прогнозів в поточній сесії

Тест кейс №8	Передумова	веб-додаток відкритий в браузері
1	Крок 1	Відкрити вікно браузера в анонімному режимі та перейти на сайт
2	Крок 2	Перейти на сторінку «Мої прогнози»
3	Очікуваний результат тест кейсу	На сторінці виводиться повідомлення «Жодного прогнозу немає»

Таблиця 3.9– Тест-кейс №9 перевірка доступу до необхідного прогнозу

Тест кейс №9	Передумова	веб-додаток відкритий в браузері, відкрита сторінка «Мої прогнози»
1	Крок 1	Обрати необхідний прогноз та натиснути лівою кнопкою миші по ньому
2	Очікуваний результат тест кейсу	Буде відображена сторінка з бажаним прогнозом, на якій виведене зображення з спрогнозованими станами рослин, опис та рекомендації лікування

3.3. Опис та аналіз результатів тестових прикладів роботи застосунку.

3.3.1. Опис та аналіз тестових прикладів.

Проведемо аналіз тестових прикладів роботи розробленої нами системи у вигляді веб-додатку розпізнавання захворювань рослин.

Приклад 1:

На головній сторінці вибираємо і завантажуємо зображення з класом «Tomato Septoria leaf spot» - це показує плями на листі помідора, викликані Septoria. В отриманій відповіді маємо вірний прогноз стану рослини, вірно визначене уражене листя, а також відповідний опис та рекомендації щодо лікування (рис. 3.5).

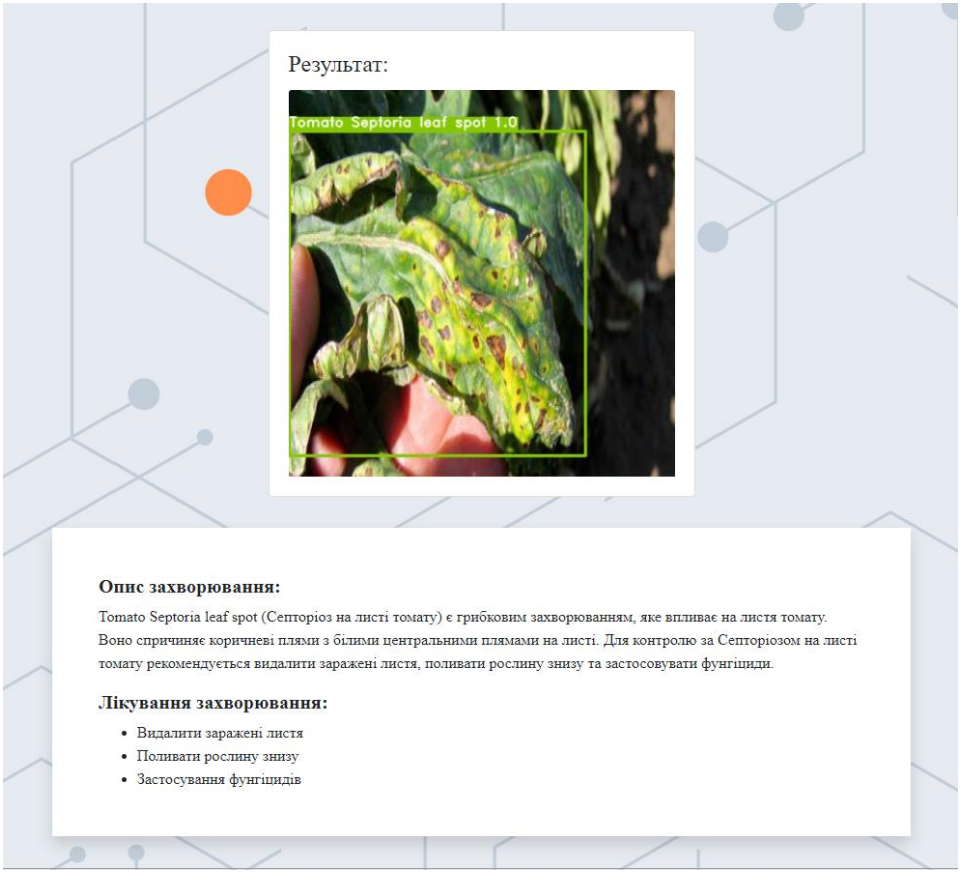


Рисунок 3.5 – Результат прогнозу «Tomato Septoria leaf spot»

Приклад 2:

Тепер перевіримо реакцію системи при використанні зображення здорової рослини. Завантажуємо зображення з категорією «Cherry leaf» (листя вишні). Отримуємо вірний прогноз стану рослини, вірно ідентифіковане здорове листя, а також інформацію, що рослина здорова і не вимагає особливого лікування (рис. 3.6).

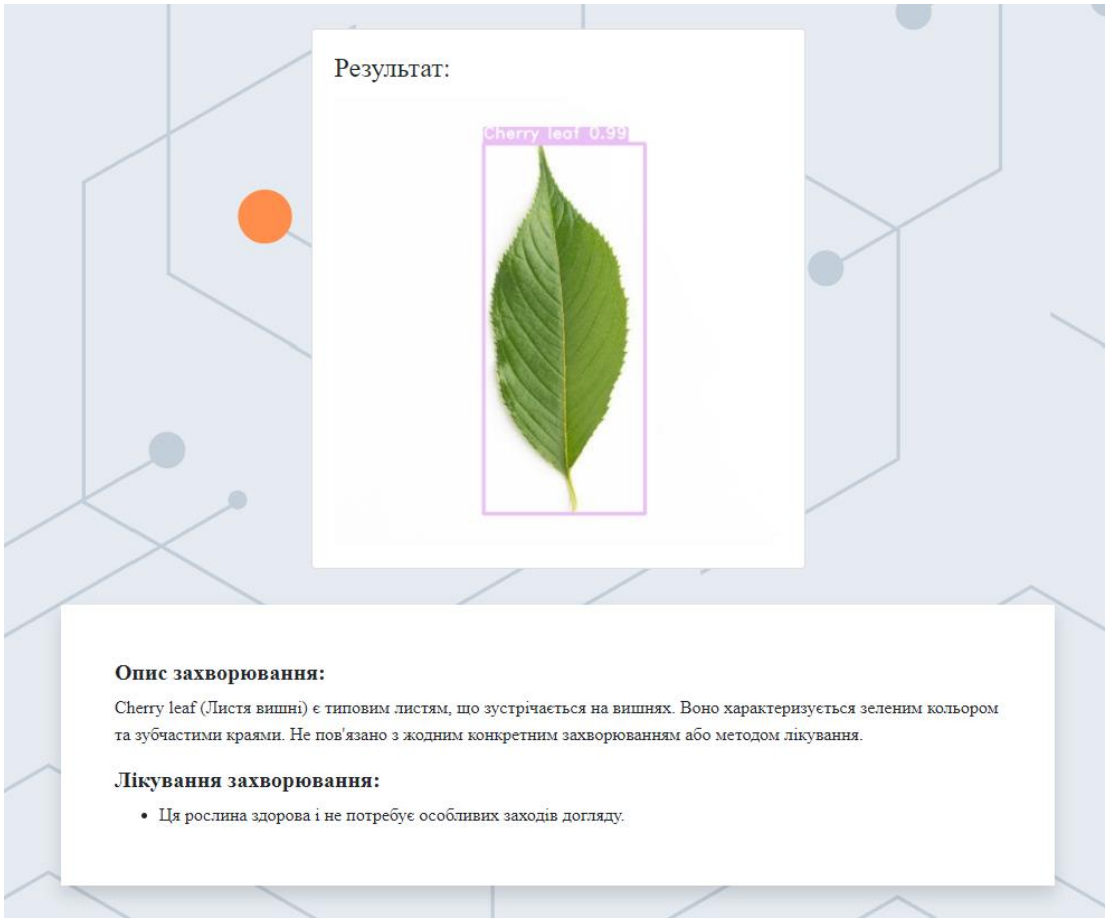


Рисунок 3.6 – Результат прогнозу «Cherry leaf»

Приклад 3:

Наступним кроком перевіримо правильне представлення збережених прогнозів. Для цього переходимо на сторінку «Мої прогнози». Відображається список всіх наших створених прогнозів, при цьому останній створений прогноз «Cherry leaf» з'являється першим у списку (рис. 3.7).

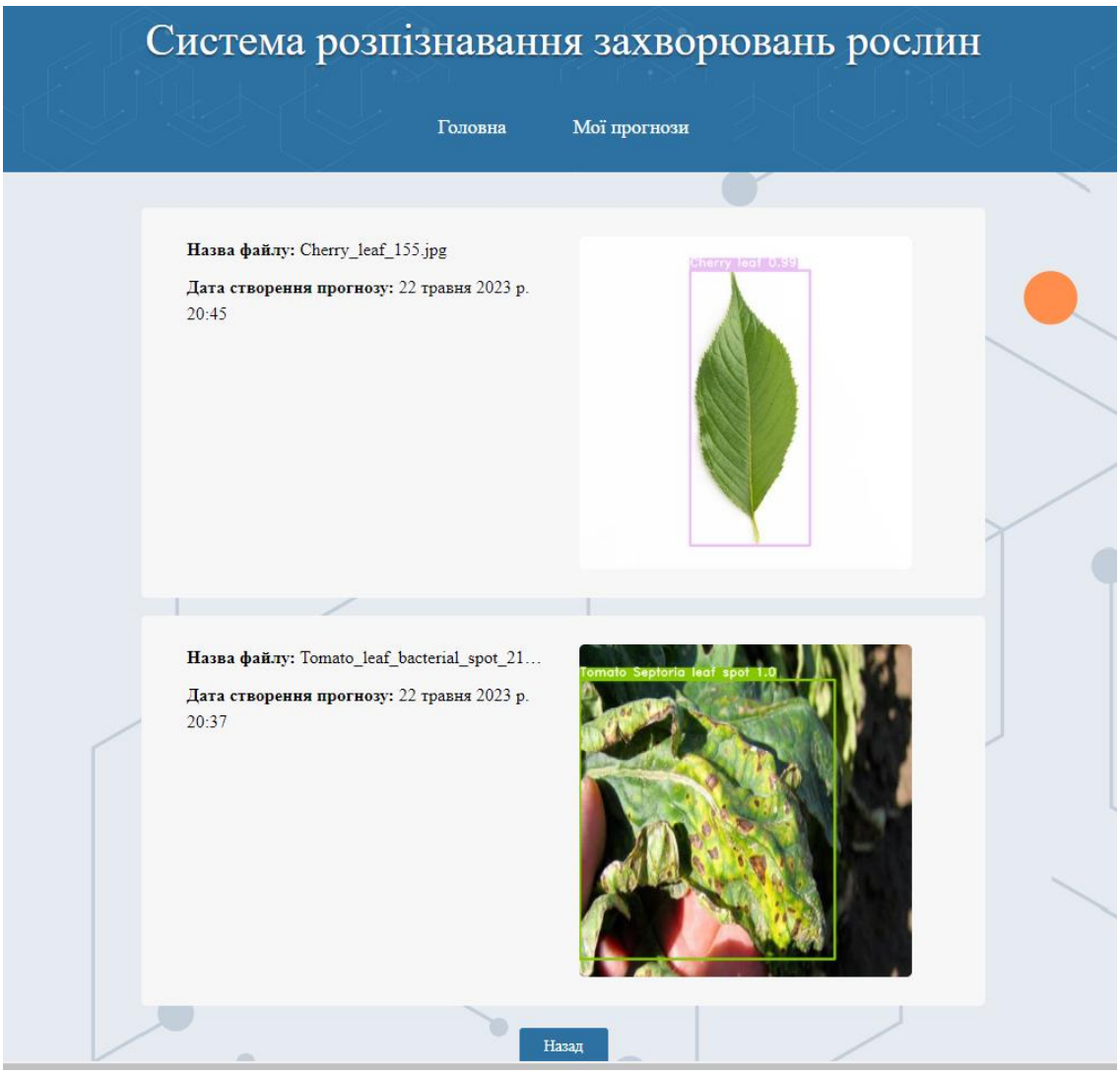


Рисунок 3.7 – Відображення історії прогнозів

Приклад 4:

Далі перевіримо доступ до деталізованої інформації по конкретному прогнозу. Вибираємо наш останній прогноз із списку. Ми отримуємо деталізовану інформацію з описом стану рослини та рекомендацій щодо лікування. В адресному полі браузера вказується відповідний ID прогнозу (рис. 3.8).

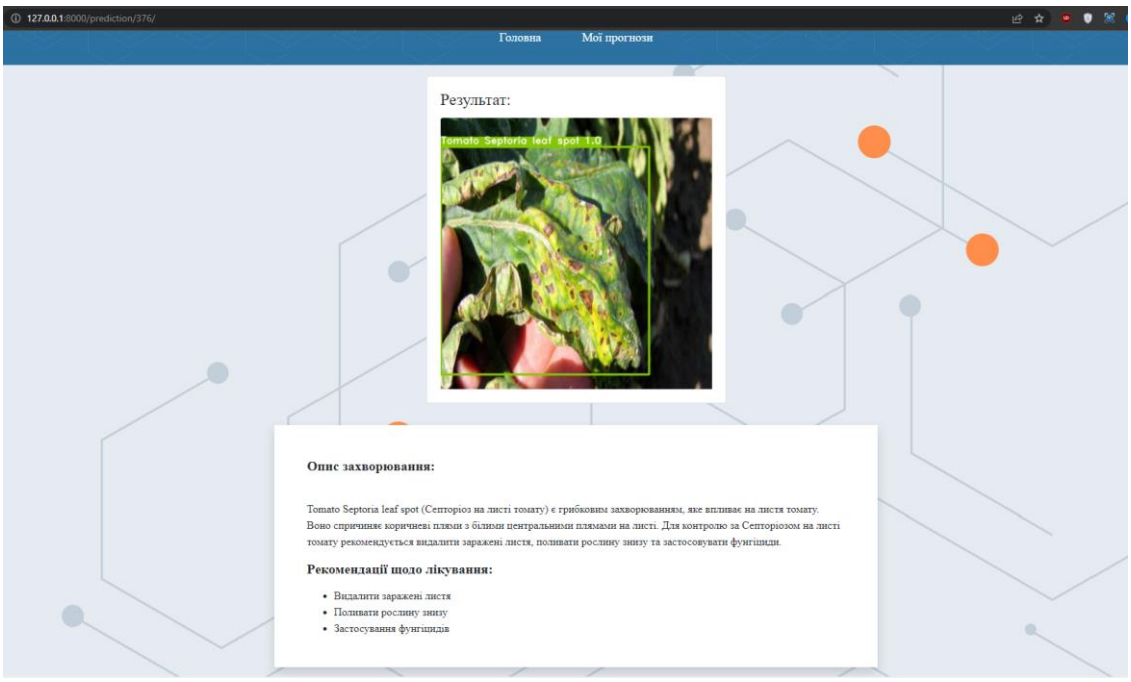


Рисунок 3.8 – Відображення детальної інформації про конкретний прогноз

Приклад 5:

Перевіримо, чи коректно прогнози зберігаються окремо для кожного анонімного користувача за ключем поточної сесії (рис. 3.10).

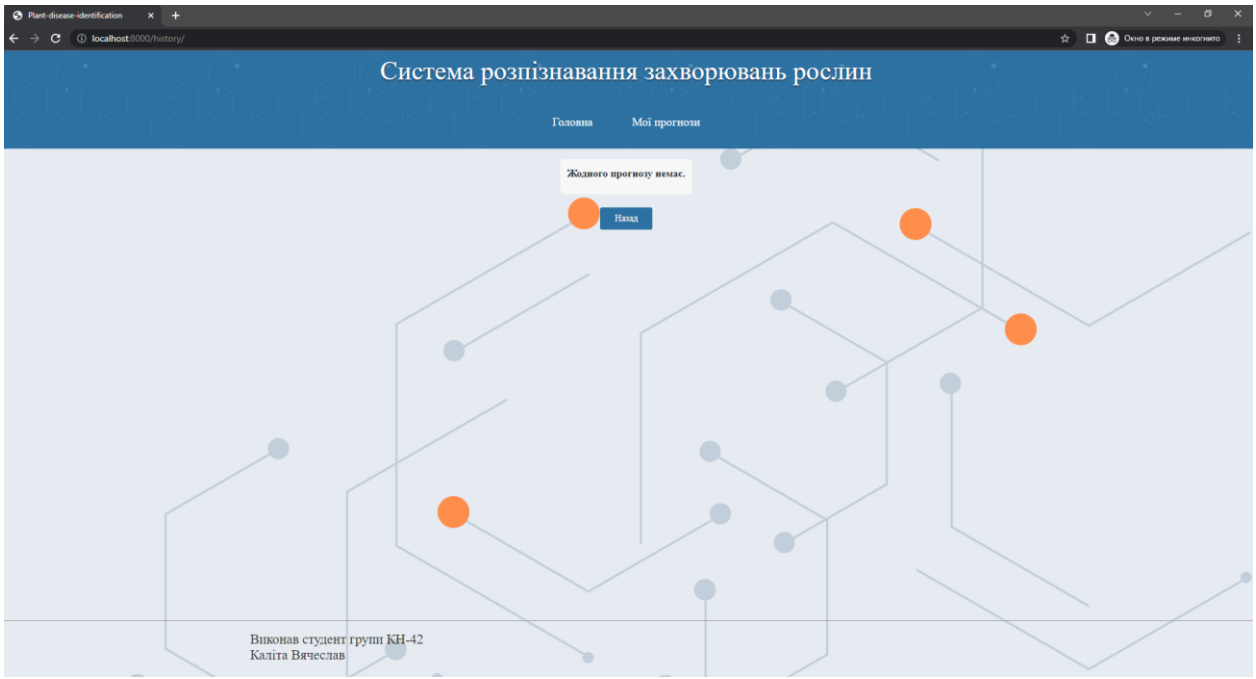


Рисунок 3.10 – Відображення історії прогнозів з анонімного вікна браузера

Для цього переходимо на адресу нашого локального сервера, будучи в анонімному режимі браузера, щоб очистити кукі . Ми отримуємо повідомлення «Жодного прогнозу немає», що свідчить про коректне зберігання прогнозів для кожного користувача.

Приклад 6:

Виконаємо тестування системи на випадок наявності кількох видів захворювань рослин на одному зображенні. Тестування виявило позитивний результат. Використовуючи об'єднане фото, що містить рослину з двома захворюваннями - "Corn gray leaf spot" та "Corn rust leaf", отримали вдале розпізнавання обох захворювань. Система відобразила два окремих вікна з описом і рекомендаціями для кожного із виявлених захворювань (рис. 3.11).

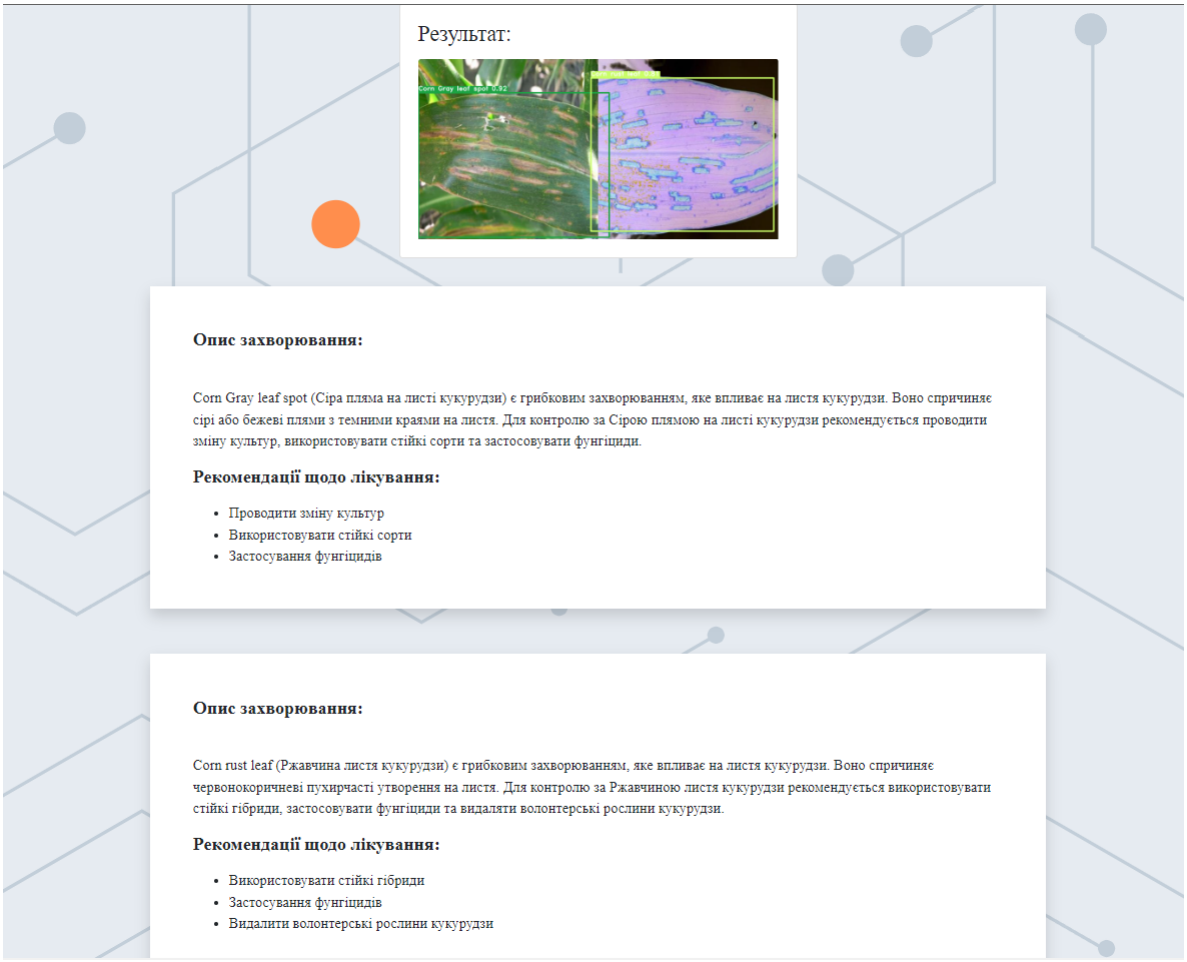


Рисунок 3.11 – Результат прогнозу для двох різних класів захворювання

Приклад 7:

Також було протестовано роботу системи з пошкодженим зображенням. На завантаження зіпсованого файлу система реагувала коректно, відображаючи повідомлення про помилку з текстом: "Помилка. Неможливо обробити зображення" (рис. 3.12).

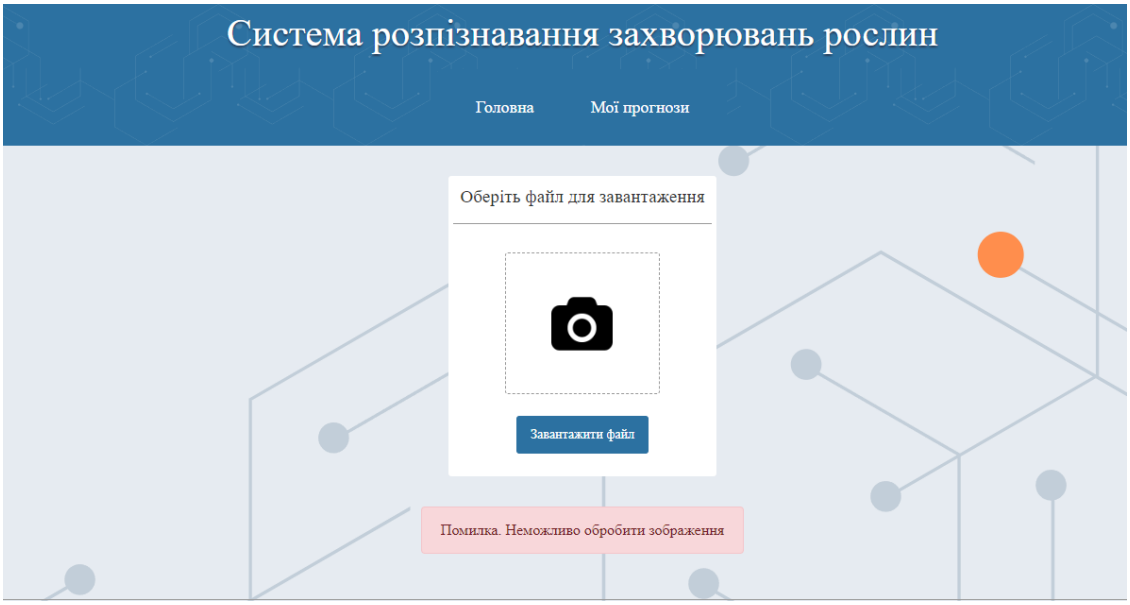


Рисунок 3.12 – Виведення повідомлення про помилку

Приклад 8:

Окремим кроком було випробування системи на зображенні складного характеру. При завантаженні складного фото яблука, знайденого в Інтернеті, система не змогла вірно визначити клас рослини, що свідчить про потребу в додатковому вдосконаленні алгоритмів розпізнавання (рис. 3.13).

Рисунок 3.13 – Невірна ідентифікація класу рослини системою

2.4.1. Опис інструктивних матеріалів для користувача додатку.

Опис інструктивних матеріалів для користувача додатку, який визначає стан рослин по зображеннях.

Передпочаткові вимоги до використання додатку: користувач повинен мати доступ до інтернету та веб-браузера для доступу до веб-додатку. Також користувач повинен мати зображення рослин для аналізу.

Підготовка до використання додатку: користувачу рекомендується завантажити наявні зображення рослин, що будуть аналізуватися. Це можуть бути зображення з власної колекції або з джерел в Інтернеті.

Використання додатку: користувач повинен завантажити зображення для аналізу, натиснувши на область завантаження файлу або перетягнувши файл до цієї області. Після того, як зображення завантажено, користувач повинен натиснути кнопку «Завантажити файл» для аналізу. Додаток здійснить аналіз зображення і надасть прогноз стану рослини, включаючи вірогідність прогнозу.

Реагування на виняткові ситуації: у разі виникнення помилок під час завантаження або аналізу зображення, додаток надасть відповідне повідомлення. Користувач повинен виконати вказівки, надані в повідомленні, для вирішення проблеми.

Збереження результатів: результати аналізу можуть бути збережені у вкладці «Мої прогнози» для подальшого використання.

Висновки до третього розділу

В цьому розділі було розроблено та описано інтерфейс програмного додатку. Також було проведено тестування функціональних вимог методом чорного ящика. Результати тестування показали, що додаток в цілому відповідає поставленим специфікаціям та вимогам. Він виконує задані функції коректно та ефективно. Впродовж тестування було виявлено декілька мінорних помилок та недоліків, які не впливають на основну функціональність застосунку, але вимагають усунення для покращення якості продукту. Веб-додаток проявив себе як надійне та продуктивне рішення. Завантаження і обробка даних проводяться швидко та без помилок. Крім того, система показала себе відмінно при взаємодії з користувачами, надаючи високий рівень сервісу та зручності використання. Загалом, результати тестування доводять, що додаток готовий до використання в реальних умовах. Проте, для забезпечення найкращої якості та відповідності очікуванням користувачів, слід продовжити працювати над вдосконаленням додатку, включаючи додаткове тестування та усунення знайдених помилок.

ВИСНОВКИ

У ході виконання даної роботи було проведено детальний аналіз предметної області діагностики захворювань рослин за допомогою методів глибокого навчання. В результаті було виділено ключові проблеми, з якими стикаються спеціалісти у даній сфері, а також потенційні шляхи вирішення цих проблем за допомогою впровадження новітніх технологій. На основі проведеного аналізу було створено веб-застосунок, який за допомогою нейронної мережі Faster R-CNN FPN здатен розпізнавати та класифікувати захворювання рослин. Вибір саме цієї моделі був обумовлений її високою точністю та ефективністю в задачах розпізнавання об'єктів на зображеннях. У роботі було розроблено інформаційну архітектуру системи, що дозволило детально розібрати взаємодію компонентів в системі, в тому числі базу даних для зберігання прогнозів користувачів. Це забезпечило можливість зберігати історію діагностованих захворювань та використовувати цю інформацію для виведення історії прогнозів користувачам або для подальшого аналізу та вдосконалення системи. Розроблена система успішно пройшла тестування, продемонструвавши високий рівень точності та продуктивності, а також зручність використання для користувачів. Попри декілька мінорних недоліків, які були виявлені під час тестування, система в цілому відповідає всім поставленим вимогам та специфікаціям. В майбутньому, для підвищення ефективності системи, рекомендується її розгортання на сервері з подальшим виконанням обчислень нейронною мережею в Google Colab через API. Це дозволить використовувати хмарні ресурси для обробки даних, зменшити вимоги до апаратного забезпечення користувачів та зробити додаток доступним для використання на різних пристроях. Загалом, проведена робота продемонструвала великий потенціал використання глибокого навчання в сільському господарстві, особливо в контексті діагностики захворювань рослин.

Список використаних джерел

1. Savary, S., Willocquet, L., Pethybridge, S. J., Esker, P., McRoberts, N., & Nelson, A. (2019). The global burden of pathogens and pests on major food crops. Nature Ecology & Evolution, 3(3), 430–439. [Електронний ресурс]. - Режим доступу: https://doi.org/10.1038/s41559-018-0793-y
2. Barbedo, J.G.A. (2018). Impact of dataset size and variety on the effectiveness of deep learning and transfer learning for plant disease classification. Computers and Electronics in Agriculture, 153, 46–53.
3. Boulent, J., Foucher, S., Théau, J., & St-Charles, P. L. (2019). Convolutional Neural Networks for the Automatic Identification of Plant Diseases. Frontiers in Plant Science, 10, 941. DOI=10.3389/fpls.2019.00941.
4. Mokhtar, U., Ali, M. A., Hassanien, A. E., & Hefny, H. (2020). Deep learning approach for plant leaf disease identification. Computers, Materials & Continua, 62(1), 75–88.
5. Sladojevic, S., Arsenovic, M., Anderla, A., Culibrk, D., & Stefanovic, D. (2016). Deep Neural Networks Based Recognition of Plant Diseases by Leaf Image Classification. Computational Intelligence and Neuroscience, 2016, 3289801. DOI=10.1155/2016/3289801.
6. Kelleher, J. D., Namee, B. M., & D'Arcy, A. (2015). Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies. MIT Press.
7. Hinton GE, Salakhutdinov R. (2006). Reducing the dimensionality of data with neural networks. Science, 313(5786), 504–7.
8. Shekhawat RS, Sinha A. (2020). Review of image processing approaches for detecting plant diseases. IET Image Process, 14(8), 1427–39.
9. Liu W, Wang Z, Liu X, et al. (2017). A survey of deep neural network architectures and their applications. Neurocomputing, 234, 11–26
10. Kumar S, Kaur R. (2015). Plant disease detection using image processing—a review. Int J Comput Appl, 124(2), 6–9.

11. Bengio Y, Courville A, Vincent P. (2013). Representation learning: a review and new perspectives. IEEE Trans Pattern Anal Mach Intell, 35(8), 1798–828
12. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).
13. Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. [Електронний ресурс]. - Режим доступу: https://arxiv.org/abs/1409.1556
14. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).
15. Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. [Електронний ресурс]. - Режим доступу: https://arxiv.org/abs/1409.1556
16. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).
17. Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. A. (2017). Inception-v4, inception-resnet and the impact of residual connections on learning. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 31, No. 1).
18. Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4700-4708).
19. Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. [Електронний ресурс]. - Режим доступу: https://arxiv.org/abs/1704.04861
20. Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., & Keutzer, K. (2016). Squeezenet: Alexnet-level accuracy with 50x fewer

parameters and< 0.5 mb model size. [Електронний ресурс]. - Режим доступу: https://arxiv.org/abs/1602.07360

21. R-CNN, Fast R-CNN, Faster R-CNN, YOLO. [Електронний ресурс]. - Режим доступу: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e
22. PlantDoc Object Detection Dataset. [Електронний ресурс]. - Режим доступу: https://public.roboflow.com/object-detection/plantdoc
23. Lin, T., Dollar, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2016). Feature pyramid networks for object detection. [Електронний ресурс]. - Режим доступу: https://arxiv.org/pdf/1612.03144.pdf
24. Plants: From Cells to Systems. [Електронний ресурс]. - Режим доступу: https://www.mdpi.com/2673-2688/2/3/26
25. Python documentation. [Електронний ресурс]. - Режим доступу: https://docs.python.org/
26. PyTorch 2.0 documentation. [Електронний ресурс]. - Режим доступу: https://pytorch.org/docs/stable/index.html
27. OpenCV-Python Tutorials. [Електронний ресурс]. - Режим доступу: https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html
28. Scikit-learn: machine learning in Python. [Електронний ресурс]. - Режим доступу: https://scikit-learn.org/stable/
29. Pillow (PIL Fork) 9.5.0 documentation. [Електронний ресурс]. - Режим доступу: https://pillow.readthedocs.io/en/stable/
30. NumPy documentation. [Електронний ресурс]. - Режим доступу: https://numpy.org/doc/stable/
31. About the Django Software Foundation. [Електронний ресурс]. - Режим доступу: https://www.djangoproject.com/foundation/
32. About SQLite. [Електронний ресурс]. - Режим доступу: https://www.sqlite.org/index.html
33. JavaScript – MDN Web Docs - Mozilla. [Електронний ресурс]. - Режим доступу: https://developer.mozilla.org/uk/docs/Web/JavaScript

Додаток А

Лістинг програмного коду модулю train.py:

```
import os
import torch
import torchvision
from PIL import Image
from torchvision.models.detection.faster_rcnn import FastRCNNPredictor
from torch.utils.data import DataLoader
from torchvision import transforms
from torch.utils.data import Dataset
from xml.etree import ElementTree as ET


def initialize_model(num_classes):
    model = torchvision.models.detection.fasterrcnn_resnet50_fpn(pretrained=True)
    features = model.roi_heads.box_predictor.cls_score.features
    model.roi_heads.box_predictor = FastRCNNPredictor(features, num_classes)
    return model


class MyDataset(Dataset):
    def __init__(self, root, transforms=None):
        self.root = root
        self.transforms = transforms
        self.imgs = list(sorted(os.listdir(root)))
        self.classes = ['__background__',
                        'Apple Scab Leaf', 'Apple leaf', 'Apple rust leaf',
                        'Bell_pepper leaf', 'Bell_pepper leaf spot', 'Blueberry leaf',
                        'Cherry leaf', 'Corn Gray leaf spot', 'Corn leaf blight',
                        'Corn rust leaf', 'Peach leaf', 'Potato leaf',
                        'Potato leaf early blight', 'Potato leaf late blight',
                        'Raspberry leaf', 'Soyabean leaf', 'Soybean leaf',
                        'Squash Powdery mildew leaf', 'Strawberry leaf',
                        'Tomato Early blight leaf', 'Tomato Septoria leaf spot',
                        'Tomato leaf', 'Tomato leaf bacterial spot',
                        'Tomato leaf late blight', 'Tomato leaf mosaic virus',
                        'Tomato leaf yellow virus', 'Tomato mold leaf',
                        'Tomato two spotted spider mites leaf', 'Grape leaf',
                        'Grape leaf black rot']

    def __getitem__(self, idx):
        img_path = os.path.join(self.root, self.imgs[idx])
        box_path = os.path.join(self.root, self.imgs[idx].replace('.jpg',
                                                    '.xml'))

        img = Image.open(img_path).convert("RGB")

        tree = ET.parse(box_path)
        root = tree.getroot()

        boxes = []
        labels = []
        for object in root.findall('object'):
            name = object.find('name').text
            labels.append(self.classes.index(name))
            for box in object.findall('bndbox'):
                xmin = int(box.find('xmin').text)
                ymin = int(box.find('ymin').text)
                xmax = int(box.find('xmax').text)
                ymax = int(box.find('ymax').text)
                boxes.append([xmin, ymin, xmax, ymax])
```

```
        boxes = torch.as_tensor(boxes, dtype=torch.float32)
        labels = torch.as_tensor(labels, dtype=torch.int64)

        image_id = torch.tensor([idx])
        target = {}
        target["boxes"] = boxes
        target["labels"] = labels
        target["image_id"] = image_id

        if self.transforms is not None:
            img = self.transforms(img)

        return img

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


dataset_train = MyDataset('../input/train/', transforms=transforms.ToTensor())
dataset_val = MyDataset('../input/val/', transforms=transforms.ToTensor())

dataloader_train = DataLoader(dataset_train, batch_size=2, shuffle=True)
dataloader_val = DataLoader(dataset_val, batch_size=2, shuffle=False)

num_classes = 31
model = initialize_model(num_classes)

device = torch.device('cuda') if torch.cuda.is_available() else torch.device('cpu')
model.to(device)

params = [p for p in model.parameters() if p.requires_grad]
optimizer = torch.optim.SGD(params, lr=0.005, momentum=0.9, weight_decay=0.0005)

best_score = float('inf')
num_epochs = 30
for epoch in range(num_epochs):
    model.train()
    for images, targets in dataloader_train:
        images = list(image.to(device) for image in images)
        targets = [{k: v.to(device) for k, v in t.items()} for t in targets]

        loss_dict = model(images, targets)
        losses = sum(loss for loss in loss_dict.values())

        optimizer.zero_grad()
        losses.backward()
        optimizer.step()

    model.eval()
    with torch.no_grad():
        for images, targets in dataloader_val:
            images = list(image.to(device) for image in images)
            targets = [{k: v.to(device) for k, v in t.items()} for t in targets]

            loss_dict = model(images, targets)
            losses = sum(loss for loss in loss_dict.values())

            if losses < best_score:
                best_score = losses
                torch.save(model.state_dict(), '../logs/bestmodel.pth')
```

Лістинг програмного коду модулю views.py:

```
import os
import ast
import uuid

import cv2
import pandas as pd
from django.shortcuts import render, get_object_or_404
from django.conf import settings
import numpy as np
import torchvision
import torch
from torch.testing._internal.codegen.random_topo_test import DEVICE
from torchvision.models.detection.faster_rcnn import FastRCNNPredictor
from torchvision import transforms as transforms
from application.forms import PredictionModelForm
from application.models import PredictionModel
from django.db.models import Q


def initialize_model(num_classes):
    model = torchvision.models.detection.fasterrcnn_resnet50_fpn_v2(weights='DEFAULT')
    in_features = model.roi_heads.box_predictor.cls_score.in_features
    model.roi_heads.box_predictor = FastRCNNPredictor(in_features, num_classes)
    return model


def load_model():
    checkpoint = torch.load("C:/Users/gedre/Desktop/PlantDiseaseDetector/logs/best_model.pth",
map_location=DEVICE)
    num_classes = checkpoint['data']['NC']
    classes = checkpoint['data']['CLASSES']
    model = initialize_model(num_classes)
    model.load_state_dict(checkpoint['model_state_dict'])
    model.to(DEVICE).eval()
    return model, classes


def set_annotations(outputs, detection_threshold, classes, original_image, resized_image):
    height, width, _ = original_image.shape
    boxes = outputs[0]['boxes'].data.numpy()
    scores = outputs[0]['scores'].data.numpy()
    filtered_boxes = boxes[scores >= detection_threshold].astype(np.int32)
    draw_boxes = filtered_boxes.copy()
    predicted_classes = [classes[i] for i in outputs[0]['labels'].cpu().numpy()]
    line_width = max(round(sum(original_image.shape) / 2 * 0.003), 2)
    text_font_scale = max(line_width - 1, 1)
    indexes = []
    colors = np.random.uniform(0, 255, size=(len(classes), 3))

    for j, box in enumerate(draw_boxes):
        point1 = (int(box[0] / resized_image.shape[1] * width), int(box[1] / resized_image.shape[0] * height))
        point2 = (int(box[2] / resized_image.shape[1] * width), int(box[3] / resized_image.shape[0] * height))
        class_name = predicted_classes[j]
        color = colors[classes.index(class_name)]
        cv2.rectangle(original_image, point1, point2, color=color, thickness=line_width,
lineType=cv2.LINE_AA)
        final_label = class_name + ' ' + str(round(scores[j], 2))
        text_width, text_height = \
            cv2.getTextSize(final_label, cv2.FONT_HERSHEY_DUPLEX, fontScale=line_width / 4,
thickness=text_font_scale)[
                0]
```

```
        outside = point1[1] - text_height >= 3
        point2 = point1[0] + text_width, point1[1] - text_height - 3 if outside else point1[1] + text_height + 3
        cv2.rectangle(original_image, point1, point2, color=color, thickness=-1, lineType=cv2.LINE_AA)
        cv2.putText(original_image, final_label, (point1[0], point1[1] - 5 if outside else point1[1] + text_height
+ 2),
                    cv2.FONT_HERSHEY_DUPLEX, fontScale=line_width / 4, color=(255, 255, 255),
thickness=text_font_scale,
                    lineType=cv2.LINE_AA)
        idx = classes.index(class_name)
        if idx not in indexes:
            indexes.append(idx)
    return original_image, indexes


def predict_image(image_path, user, session_key):
    model, classes = load_model()
    image = cv2.imread(image_path)
    frame_height, frame_width, _ = image.shape
    if frame_height != frame_width:
        img_size = min(frame_width, frame_height)
        h, w = image.shape[:2]
        r = img_size / max(h, w)
        if r != 1:
            image = cv2.resize(image, (int(w * r), int(h * r)))
    image_tensor = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    image_tensor = transforms.ToTensor()(image_tensor)
    image_tensor = torch.unsqueeze(image_tensor, 0)
    with torch.no_grad():
        predictions = model(image_tensor.to(torch.device('cuda:0' if torch.cuda.is_available() else 'cpu')))
    predictions = [{k: v.to('cpu') for k, v in t.items()} for t in predictions]
    class_indexes = []
    threshold = 0.8
    if len(predictions[0]['boxes']) != 0:
        output_image, class_indexes = set_annotations(predictions, threshold, classes, image, image)
    output_path = f"{image_path}_output.jpg"
    cv2.imwrite(output_path, output_image)
    image_name = os.path.basename(image_path)
    prediction_model = PredictionModel.objects.create(
        name=image_name,
        image=output_path,
        class_indexes=class_indexes,
        owner=user,
        session_id=session_key
    )
    return prediction_model, class_indexes


def convert_string_to_list(string):
    try:
        list_data = ast.literal_eval(string)
        if isinstance(list_data, list):
            int_list = [int(item) for item in list_data]
            return int_list
    except (SyntaxError, ValueError):
        pass
    return None


def index(request):
    if request.method == 'POST':
        form = PredictionModelForm(request.POST, request.FILES)
        if form.is_valid():
            user = request.user if request.user.is_authenticated else None
```

```
            if not request.session.session_key:
                request.session.create()
            session_key = request.session.session_key
            image = form.cleaned_data['image']
            filepath = os.path.join(settings.MEDIA_ROOT, image.name)
            with open(filepath, 'wb') as f:
                f.write(image.read())
            info = pd.read_csv('info.csv', encoding='cp1251')
            descriptions = info['description'].tolist()
            possible_steps = info['possible_steps'].tolist()
            prediction, class_idx = predict_image(filepath, user, session_key)
            prediction.save()
            context = {
                'form': form,
                'prediction': prediction,
                'class_indexes': class_idx,
                'descriptions': descriptions,
                'possible_steps': possible_steps,
            }
            return render(request, 'prediction.html', context)
        else:
            return render(request, 'prediction.html', {'form': form, 'error': 'Помилка. Неможливо обробити зображення'})
    else:
        prediction_form = PredictionModelForm()
        return render(request, 'index.html',
                      {'prediction_form': prediction_form})


def prediction_view(request, prediction_id):
    prediction = get_object_or_404(PredictionModel, id=prediction_id)
    info = pd.read_csv('info.csv', encoding='cp1251')
    descriptions = info['description'].tolist()
    possible_steps = info['possible_steps'].tolist()
    class_indexes = convert_string_to_list(prediction.class_indexes)
    context = {
        'prediction': prediction,
        'class_indexes': class_indexes,
        'descriptions': descriptions,
        'possible_steps': possible_steps,
    }
    return render(request, 'prediction_detail.html', context)


def predictions_history(request):
    user = request.user
    session_key = request.session.session_key

    if user.is_authenticated:
        prediction_list = PredictionModel.objects.filter(owner=user).order_by('-timestamp')
    else:
        if session_key is not None:
            prediction_list = PredictionModel.objects.filter(Q(owner__isnull=True) & Q(session_id=session_key)).order_by('-timestamp')
        else:
            prediction_list = PredictionModel.objects.none()

    return render(request, 'history.html', {'prediction_list': prediction_list})
```

Лістинг програмного модулю models.py:

```
from django.contrib.postgres.fields import ArrayField
```

```
from django.core.validators import FileExtensionValidator
from django.db import models
from django.contrib.auth.models import User


class PredictionModel(models.Model):
    name = models.CharField('Назва файлу', max_length=100)
    image = models.ImageField('Зображення', validators=[FileExtensionValidator(['jpg', 'jpeg', 'png', 'svg'])])
    class_indexes = models.CharField('Індекси прогнозів', max_length=100, null=True, blank=True)
    owner = models.ForeignKey(User, on_delete=models.CASCADE, null=True)
    timestamp = models.DateTimeField(auto_now_add=True)
    session_id = models.CharField(max_length=255, null=True, blank=True)

    def delete(self, *args, **kwargs):
        self.image.delete()
        super().delete(*args, **kwargs)

    def __str__(self):
        return f"{self.name} - {self.timestamp}"
```

Лістинг програмного модулю index.js:

```
function getFileName() {
    let file = document.getElementById('filePath');
    let filePath = file.value;
    let allowed = /(\.jpg|\.jpeg|\.png|\.svg|\.jfif|\.bmp|\.tiff|\.webp)$/i;
    if (file.files.length) {
        if (!allowed.exec(filePath)) {
            alert('Невірний формат файлу');
            file.value = '';
            return false;
        } else {
            for (var i = 0; i <= file.files.length - 1; i++) {
                document.getElementById('showName').innerText =
                    file.files.item(i).name;
            }
        }
    }
}
```