ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Метод діагностики захворювань кімнатних рослин із використанням штучного інтелекту та алгоритмів машинного навчання»

на здобуття освітнього ступеня магістра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Абдуль Фаттах МАХМУД
(підпис)

Виконав: здобувач вищої освіти групи ПДМ-62
Абдуль Фаттах МАХМУД

Керівник: Ірина ЗАМРІЙ
д-р. техн. наук, проф.

Рецензент: _____
науковий ступінь, вчене звання *Ім'я, ПРІЗВИЩЕ*

Київ 2026

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Магістр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
«_____» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Махмуду Абдуль Фаттах Мазановичу

1. Тема кваліфікаційної роботи: «Метод діагностики захворювань кімнатних рослин із використанням штучного інтелекту та алгоритмів машинного навчання»

керівник кваліфікаційної роботи Ірина ЗАМРІЙ, д-р. техн. наук, проф.,

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «30» жовтня 2025 р. № 467.

2. Строк подання кваліфікаційної роботи «19» грудня 2025 р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література з фітопатології та комп'ютерного зору, характеристики датасетів рослин і алгоритми глибокого навчання для визначення їхніх захворювань.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Аналіз предметної галузі.
2. Розробка математичної моделі та алгоритму діагностики.
3. Розробка та реалізація web-системи діагностики.

4. Експериментальні дослідження та оцінювання ефективності.

5. Перелік ілюстративного матеріалу: *презентація*

1. Мета, об’єкт та предмет дослідження.
2. Актуальність роботи.
3. Математична модель класифікації захворювань.
4. Архітектура системи та блок-схема діагностики.
5. Результати моделювання.
6. Діаграма порівняння ефективності системи.
7. Проблеми типового датасету PlantigVilage.
8. Практичний результат.
9. Екранні форми.

6. Дата видачі завдання «31» жовтня 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	31.10 – 07.11.2025	
2	Вивчення матеріалів щодо розвитку технологій автоматизованої діагностики стану рослин та аналізу зображень	08.11 – 12.11.2025	
3	Дослідження методів глибокого навчання	13.11 – 18.11.2025	
4	Аналіз впливу методів глибокого навчання на точність діагностики захворювань рослин за зображеннями	19.11 – 22.11.2025	
5	Дослідження технологій глибокого навчання	23.11 – 26.11.2025	
6	Застосування методів глибокого навчання (MobileNetV2) для діагностики захворювань рослин	27.11 – 05.12.2025	
7	Оформлення роботи: вступ, висновки, реферат	06.12 – 12.12.2025	
8	Розробка демонстраційних матеріалів	13.12 – 17.12.2025	
9	Попередній захист роботи	17.11 – 01.12.2025	

Здобувач вищої освіти _____ Абдуль Фаттах МАХМУД
(підпис)

Керівник
кваліфікаційної роботи _____ Ірина ЗАМРІЙ
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 93 стор., 8 табл., 23 Рис., 30 джерел.

Мета роботи – підвищення точності діагностики захворювань кімнатних рослин шляхом використання алгоритмів машинного навчання та технологій штучного інтелекту.

Об'єкт дослідження – процес автоматизованої діагностики стану кімнатних рослин за зображеннями.

Предмет дослідження – методи та алгоритми машинного навчання, зокрема згорткові нейронні мережі (CNN) та їх варіації, що застосовуються для аналізу зображень кімнатних рослин.

У роботі використано сучасні методи аналізу зображень, машинного навчання та CNN-архітектури.

Проведено аналіз існуючих підходів до автоматичної діагностики захворювань рослин.

Розроблено та оптимізовано модель MobileNetV2 для класифікації стану листя. Проведено експериментальне дослідження якості моделі та оцінено її ефективність у реальних умовах.

КЛЮЧОВІ СЛОВА: ДІАГНОСТИКА РОСЛИН, КОМП'ЮТЕРНИЙ ЗІР, MOBILENETV2, МАШИННЕ НАВЧАННЯ, CNN, АНАЛІЗ ЗОБРАЖЕНЬ, ФІТОПАТОЛОГІЯ.

ABSTRACT

Text part of the qualification work for obtaining a master's degree: 93 pages, 8 tables, 23 figures, 30 sources.

The purpose of the work is to improve the accuracy of diagnostics of houseplant diseases by using machine learning algorithms and artificial intelligence technologies.

The object of the research is the process of automated diagnostics of the condition of houseplants using images.

The subject of the research is machine learning methods and algorithms, in particular convolutional neural networks (CNN) and their variations, used to analyze images of houseplants.

The work uses modern methods of image analysis, machine learning and CNN architectures.

An analysis of existing approaches to automatic diagnostics of plant diseases is carried out.

The MobileNetV2 model for leaf condition classification was developed and optimized. An experimental study of the model quality was conducted and its effectiveness in real conditions was evaluated.

KEYWORDS: PLANT DIAGNOSTICS, COMPUTER VISION, MOBILENETV2, MACHINE LEARNING, CNN, IMAGE ANALYSIS, PHYTOPATHOLOGY.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

AI - Штучний Інтелект (Artificial Intelligence)

CNN - Згорткова нейронна мережа (Convolutional Neural Network)

CV - Комп'ютерний зір (Computer Vision)

ML - Машине навчання (Machine Learning)

GAP - Global Average Pooling

API - Інтерфейс прикладного програмування (Application Programming Interface).

HTML - стандартизована мова розмітки документів для перегляду веб сторінок у браузері (HyperText Markup Language)

JSON - текстовий формат обміну даними між комп'ютерами (JavaScript Object Notation)

HTTP - протокол передачі даних, що використовується в комп'ютерних мережах (Hypertext Transfer Protocol)

UI - засіб зручної взаємодії користувача (людини) з інформаційною системою (user interface)

URL - стандартизована адреса певного ресурсу (документ або зображення) в інтернеті (Uniform Resource Locator)

UX - досвід користувача, тобто те, як користувач сприймає продукт (User Experience)

ВСТУП

У сучасному світі, де цифрові технології проникають в усі сфери життя, стає актуальним питання використання AI технологій для вирішення завдань у біології, екології та агротехнологіях. Одним з таких напрямків є автоматизована діагностика захворювань рослин підвищити ефективність догляду та зменшити витрати ресурсів і мінімізувати ризики втрати врожаю або загибелі рослин. Особливе значення такі системи мають для кімнатних рослин, оскільки вони стають все частіше елементом екосистеми дому та вимагають своєчасного виявлення проблем пов’язаних із захворюваннями, шкідниками або дефіцитом по поживних речовин

Традиційні методи діагностики стану рослин базуються на візуальному огляді, лабораторних аналізів та досвіді. Проте такі підходи є дуже енерго-затратні, витратними у часі та нерідко суб'єктивними. В цьому контексті AI технології зокрема алгоритми машинного навчання та глибокого навчання дають нові можливості для створення автоматизованих систем аналізу зображень листя, стебел квіток, це дасть змогу своєчасно виявляти ураження та визначити стадію розвитку проблем.

Сучасні дослідження у сфері комп'ютерного зору та машинного навчання демонструють високу ефективність згорткових нейронних мереж (CNN) у задачах класифікації станів автоматичного розпізнавання ознаки уражень таких як плями, зміни кольору та деформація листків, це дозволить визначити конкретні типи захворювань.

Актуальність теми зумовлена необхідністю удосконалення систем догляду за кімнатними рослинами та запровадженням інноваційних методів автоматичної діагностики їх недуг.

У міських умовах, де догляд за рослинами часто обмежений часом та досвідом власників впровадження AI технологій дозволяє виявити відхилення у

стані рослини на ранніх стадія це дасть змогу запобіганню їх загибелі. Такий підхід має не лише наукове, а й практичне значення для розробки для інтегрування у мобільні додатки та web-застосунки.

Таким чином використання AI технологій для діагностики захворювань кімнатних рослин є перспективним напрямом, що поєднує технології комп'ютерного зору та машинного навчання. Дослідження у цій галузі спрямовані на підвищення точності розпізнавання швидкість обробки зображень, здатних працювати з різними видами рослин.

Мета роботи – підвищення точності діагностики захворювань кімнатних рослин шляхом використання алгоритмів машинного навчання та технологій штучного інтелекту.

Об’єкт дослідження – процес автоматизованої діагностики стану кімнатних рослин за зображеннями.

Предмет дослідження – методи та алгоритми машинного навчання, зокрема згорткові нейронні мережі (CNN) та їх варіації, що застосовуються для аналізу зображень кімнатних рослин.

Завданнями цієї роботи є проаналізувати сучасні методи та алгоритми машинного навчання, що застосовуються для класифікації зображень рослин, визначити їх переваги та недоліки у контексті діагностики кімнатних рослин. Сформувати метод діагностики захворювань на основі згорткових нейронних мереж, обґрунтувати вибір архітектури MobileNetV2 для досягнення підвищеної точності та швидкодії. Розробити програмну систему, що забезпечує автоматизовану обробку зображень, виконання інференсу моделі та формування рекомендацій щодо лікування виявлених захворювань.

Створити базу знань для рекомендаційного модуля та визначити правила формування порад залежно від класу ураження. Провести експериментальне дослідження роботи моделі, виконати аналіз точності, матриці плутанини, швидкодії та порівняльну оцінку ефективності запропонованого методу.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Біологічні особливості кімнатних рослин і чинники їх захворювань

Кімнатні рослини є однією з складових сучасного житлового та робочого простору. Вони покращують мікроклімат приміщень, підвищують вологість повітря, рослини збагачують простір киснем та знижують концентрацію пилу. Також рослини виконують не тільки очисну функцію а і декоративну та психологічну[1]. Проте рослини чутливі до змін середовища. порушення режимів освітлення, вологості, поливу або температури призводить до ослаблення фізіологічних процесів та зниження їх імунітету до різноманітних шкідників[2].

Фотосинтез рисунок 1.1, дихання та транспірація є основними процесами які забезпечують життєдіяльність рослин. Якщо інтенсивність фотосинтезу знижуються через нестачу освітлення, рослина стає набагато слабшою[1]. Аналогічно, може бути надлишковий або недостатній полив це впливає на стан кореневої системи та може викликати гниття або в’янення[3].

Рис 1.1 Приклад фотосинтезу

Виділяється декілька основних чинників, що можуть спровокувати захворювання для кімнатних рослин[4]:

1) порушення водного режиму;
2) недостатнє освітлення;
3) надмірне освітлення;
4) різкі коливання температури;
5) дефіцит мінералів у ґрунті;
6) зараження патогенними.

Ослаблена рослини набагато швидше може уражатися грибковими, бактеріальними або вірусними інфекціями. Тому своєчасна діагностика є ключовою для успішного лікування.

1.2 Класифікація основних хвороб кімнатних рослин.

Захворювання рослин класифікуються за походженням забрудника. Грибкові інфекції становлять понад 60% захворювань кімнатних рослин[6].

Таблиця 1.1

Класифікація захворювань

Тип хвороби	**Збудник**	**Ознаки**	**Приклад**
Грибкові	Спори грибів	Плями, білий наліт	Борошниста роса
Бактеріальні	Бактерії	Потемніння тканини	Бактеріальна плямистість
Вірусні	Віруси	Мозаїчність, деформація	Мозаїка тютюну
Фізіологічні	Порушення догляду	Хлороз, уповілення росту	Дефіцит заліза

Збудниками виступають мікроскопічні гриби, які формують спори, які здатні зберігатися в ґрунті, на частинах рослин, також вони можуть зберігатися у повітрі протягом тривалого часу. Вони дуже легко поширюються в умовах підвищеної вологості та недостатньої вентиляції, що є характерним для

кімнатного середовища. Зазвичай для грибкових хвороб є типовими зовнішні симптоми такі як:

- поява білуватого або сіруватого нальоту на листковій пластині(борошна роса);
- утворення чорних або коричневих плям неправильної форми;
- в’янення рослин при збереженні вологості ґрунту (ознака гнилі);
- розм’якшення або почорніння стебел;
- опадання листя без видимих на те причин.

Грибкові інфекції небезпечні тим, що розвиваються поступово. На початкових етапах зміни можуть бути майже непомітними, але вже на пізніх рослина втрачає більшість листя та відмирає. Знищення таких інфекцій потребує біологічних препаратів або використання фунгіцидів, а навіть інколи видалення уражених частин рослин.

Вірусні хвороби на відміну від грибкових, викликаються вірусами, що вражають клітини рослин і порушують внутрішні процеси обміну речовини[7]. Їх особливість в тому що віруси не можна знищити хімічними та біологічними засобами, оскільки вони не мають власної клітинної структури. Лікування вірусних захворювань у рослин у більшості випадків неможливе, можливо лише стримувати симптоми або видаляти уражені частини рослин, це дасть змогу для стримування та запобігання зараженню інших. Типовими симптомами вірусних хвороби є:

- мозаїчні візерунки на лисках;
- порушення пігментації;
- деформація листя та пагонів;
- уповільнення росту та загальне пригнічення рослин.

Вірусні захворювання зазвичай поширюють комахи такі як трипси, кліщі, попелиця. Тому профілактика включає боротьбу із шкідниками стерильність інструментів та стерильність нових рослин.

Отже оскільки ознаки вірусних і деяких фізіологічних хвороб можуть бути дуже схожими, візуальна діагностика часто є неточною. Саме тому в сучасних системах використовують AI технології які допомагають за допомогою комп'ютерного розпізнавання зображень виявити характерні патерни текстур та кольору, що може бути важким для неозброєного ока.

1.3 Сучасні методи діагностики захворювань рослин

Діагностика захворювань кімнатних рослин традиційно базується на візуальному аналізі зовнішніх ознак, який виконується його власником або фахівцем Такий підхід передбачає оцінку форми, розміру, кольору та структури листків, стебел та коренів. Проте ефективність цього методу залежить від досвіду людини, що робить цей метод суб’єктивним і не завжди точним[8].

Крім цього на ранніх стадіях ураження симптоми можуть бути ледве помітними або схожими між різними видами захворювань. Наприклад мозаїчні візерунки можуть бути схожими на хлороз порівняння на рисунку. 1.2 та 1.3, Який міг бути спричинений нестачею заліза.

Рис. 1.2 Мозаїчні візерунки

Рис. 1.3 Хлороз

У зв'язку з цим у науковій та практичній сфері дуже часто застосовують додаткові методи діагностики які вказані в таблиці 1.2.

Таблиця 1.2

Додаткові методи діагностики

Метод	**Принцип**	**Переваги**	**Недоліки**
Мікроскопічний аналіз	Дослідження структури клітин і тканин	Дозволяє визначати тип збудника	Потребує обладнання та фахівця.
Біохімічні тести	Визначення метаболітів	Висока точність	тривалість аналізу та виска вартість
ПЛР-діагностика	Виявлення ДНК/РНК патогенна	Найбільш точний метод	Використовується переважно в лабораторіях
Комп'ютерний аналіз.	Автоматичне розпізнавання особливостей листа.	Швидко, доступно, підходить для побуту.	Потребує навчальної моделі

З представлення таблиці можна побачити що найбільш продуктивним є комп'ютерний аналіз зображень рослин, який може в собі поєднувати цифрову фотографію та математичну обробку зображень.

Цей підхід дозволяє автоматично визначати хвороби на основі патернів форми та структури, кольору так текстури листя, ідентифікуючи їх із високою точністю[9]. Дослідження показують що комп'ютерні моделі, навчені на великих наборах даних наприклад таких як PlantVillage, здатні досягти точності від 92% до 99% залежно від архітектури нейромереж[10].

Ключова перевага таких систем полягає в тому, що для діагностики достатньо зробити лише фотографію рослини. це дозволяє застосовувати цю технологію у домашніх умовах, у оранжереях та розсадниках.

Таким чином, сучасні методи діагностики хвороб рослин переходять від трудомісткого експертного аналізу до автоматизованих інтелектуальних систем що дають змогу забезпечити високу точність швидкість та головне доступність до використання.

1.4 Використання AI технологій в діагностиці хвороб

У сучасних умовах стрімкого розвитку цифрових технологій все частіше застосовуються системи на основі Artificial intelligence для аналізу візуальних ознак рослин. Однією з ефективних систем є технологія Computer Vision (комп'ютерний зір). Це дає змогу поєднувати методи обробки зображень та машинного навчання для автоматичного розпізнавання симптомів захворювань рослин[9].

Системи діагностики на основі AI технологій можуть аналізувати фотографії листя, виявляти характерні патерни (плями, некрози, візерунки, деформації), також вони можуть класифікувати хворобу та запропонувати рекомендації щодо лікування або припинення симптомів. Для реалізації таких систем використовується машинне навчання (Machine Learning), якщо більш конкретно то саме глибоке навчання та загорткова нейронна мережа CNN.

1.4.1 Згорткові нейронні мережі (CNN)

CNN є класом штучних нейронних мереж, спеціально розроблених для обробки зображень та виявлення закономірностей у двовимірних даних. Їх ключова особливість полягає у здатності автоматично виділяти інформативні ознаки зображення, не потребуючи ручного налаштування або попереднього запису цих ознак спеціалістом[10].

У звичайних підходах до використання CV для того щоб розпізнати хворобу рослин, необхідно було спочатку вручну визначити дані автоматично виділяти та визначати, які саме ознаки слід аналізувати: форму листка, контури,

величину плям, інтенсивність та багато чого іншого. Цей підхід є дуже складним та суб’єктивним та нестійким до змін освітлення або якості зображення. Натомість CNN використовує згорткові фільтри, що можуть автоматично виділяти ознаки на різних рівня абстракції.

Більш детально про рівні обробки CNN показано в таблиці 1.3.

Таблиця 1.3

Рівні обробки CNN

Рівень обробки CNN	**Розпізнавання на рівні**	**Приклад у рослинах**
Низько рівневі ознаки	Краї, лінії, контрасти	Межі плям, прожилки листа
Середньо рівневі ознаки	Текстури, форми структур	Сітчаста або мозаїчна структура
Високо рівневі ознаки	Комплексні патерни	Конкретний тип хвороби

Таким чином CNN імітують принцип роботу зорової кори мозку, де інформація також обробляється ієрархічно від просто до складного[11].

Архітектура CNN зазвичай складається з кількох основних компонентів:

1) згорткові шари (Convolution Layers) – застосовують фільтри, що просуваються по зображенню та виділяють локальні особливості такі як краї, кути, перепади кольору;
2) шари підвибірки (Pooling Layers) – зменшують розмірність даних, залишаючи тільки найважливішу інформацію, що зменшує обчислювальні витрати;
3) пов'язані шари (Fully Connected Layer) – для отримання узагальненої інформації та виконують класифікацію, визначають до якої хвороби належить;
4) функції активації (ReLu, Softmax) – дозволяють мережі навчитись нелінійним залежностям.

Чому саме CNN саме для рослин. Хвороби рослин проявляються у вигляді патернів: плям, точок, штрихів, відтінків та деформацій. Людині важко помітити дрібні та ранні зміни, тоді як CNN бачить мінімальні текстурні відхилення. CNN не залежить від умов освітлення, роздільності камери, кольору фону, модель навчається розпізнавати саме характер захворювання, а не фотографію.

За даними досліджень, проведених у рамках PlantVillage, ідентифікація захворювань за допомогою CNN досягає 92-99% точності залежно від розміру та якості набору даних, різноманітності симптомів, оборонної архітектури (MobileNet, ResNet, EfficientNet). Це набагато вище за точність візуальної діагностики, виконаної людиною без спеціальної підготовки яка не перевищує 60-70%[11].

Згорткові нейронні мережі дозволяють автоматизувати процес розпізнавання хвороб та рослин, забезпечуючи високу точність діагностики, швидкість обробки, масштабованість та простору оновлення систем через перенавчання.

2 РОЗРОБКА МАТЕМАТИЧНОЇ МОДЕЛІ ТА АЛГОРИТМУ ДІАГНОСТИКИ ЗАХВОРЮВАНЬ КІМНАТНИХ РОСЛИН

2.1 Описання задачі класифікації

Задача діагностики захворювань кімнатних рослин за зображенням належить до класу задач багато класової класифікації, у межах якої система повинна на основі вхідного зображення визначити один із можливих типів захворювань. У загальному вигляді процес можливо подати як відображення з простору зображень у дискретну множину діагнозів.

$$X \in R^{H \times W \times C}; \tag{2.1}$$

де

H,W - висота та ширина зображення;
C - кількість каналів (C = 3 для RGB).

Перед подачею на модель виконується препроцесинг масштабування до 224 х 224 пікселів, нормалізація значень у діапазоні [0,1] або [-1, 1], а також можливе застосування (повороти, зміна яскравості або шум).

Результатом роботи класифікатора є прогнозований вектор ймовірностей:

$$\hat{y} = [p_1, p_2, \dots, p_K], p_i \in [0,1], \sum_{i=1}^{K} p_i = 1; \tag{2.2}$$

де

K – кількість можливих класів захворювань (наприклад, 5 або 7 залежно від набору даних). Для прийняття рішення застосовується правило $y^* = argmax\ p_i$. Отже система повертає клас, що має найбільшу ймовірність.

У межах моделі використовується глибока згорткова нейронна мережа MobileNetV2, яка виконує функцію автоматичного виділення ознак. Можна позначити:

$$f = F(X''), f \in R^d; \quad (2.3)$$

де

F() – згорткова частина моделі;

f – високо рівневий вектор ознак;

d = 1280 - типова розмірність векторам після шару Global Average Pooling.

На основі цього вектора працює класифікаційний шар:

$$\hat{y} = \sigma(Wf + b); \quad (2.4)$$

де

W, b - параметри повнозв’язоного шару;

σ() = softmax-функция.

Таким чином, процес формування ознак повністю автоматизовано та не потребує ручного добору текстурних дескрипторів (HOG, SIFT).

Математична постановка задачі полягає у знаходженні параметрів моделі 0, які мінімізують функцію витрат:

$$L \to \sum_{i=1}^{K} y_i \, log \, \check{y}_i \, ; \quad (2.5)$$

де

y_i - “one-hot” позначення істинне значення класу;

$\sum_{i=1}^{K}\{y_i\}$ - прогноз моделі.

Таким чином, модель навчається таким чином, щоб мінімізувати похибку класифікації та підвищити узагальнювальну здатність.

2.2 Математична модель CNN-класифікатора

Згорткові нейронні мережі (CNN) є базовим інструментом сучасних систем комп'ютерного зору, оскільки здатні автоматично видаляти інформативні ознаки[14]. На відміну від традиційних методів, що використовують вручну визначені дескриптори (SIFT, HOG), CNN формують представлення даних, від простих контурів до складних семантичних структур.

У загальному вигляді CNN є функцією $f: X \to y$, яка виконує перетворення вхідного тензора $X \in R^{H \times W \times C}$ у вектор ймовірностей класів y.

Згортка є ключовою операцією CNN, яка дозволяє виділяти просторові патерни на зображені. Одна фільтрація визначається формулою.

$$Y[i,j] = \sum_m \sum_n X[i+m, j+n] \blacksquare W[m,n]; \qquad (2.6)$$

де

W- ядро згортки.

Використовується декілька десятків або навіть сотень фільтрів, що може дозволяти формувати багатоканальних ознак. Для введення нелінійності краще всього застосовувати ReLu: $ReLU(x) = max(0,r)$, що прискорює навчання та запобігає проблемі зникнення градієнта.

Pooling-шари зменшують просторові розміри ознак, зменшуючи кількість параметрів і підвищуючи стійкість мережі до локальних змін:

MaxPooling - вибір максимального значення фрагмента;

AveragePooling - обчислення середнього значення.

У моделі MobileNetV2 використовується Global Average Pooling (GAP), він перетворює карту розмірності $H' \times W' \times C'$ у вектор довжини C. У випадку MobileNetV2 цей вектор має розмірність 1280. Після GAP формується вектор ознак.

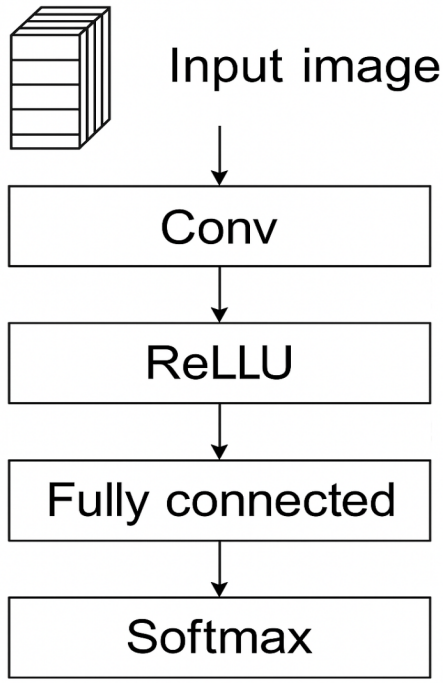


Рис 2.1 Структура CNN

На рисунку 2.1 представлено узагальнену архітектуру обгорткової нейронної мережі CNN, що використовується для класифікації зображень кімнатних рослин за видами уражень. Схема демонструє послідовність основних етапів обробки вхідного зображення. Від початкового шару отримання до формування фінального вектору ймовірностей на виході.

Першим етапом є подання вхідного зображення розміром 224 x2 24 x 3, після чого воно знаходить на згортковий шар (Convolution 3x3), який виконує витягування локальних ознак за допомогою фільтрів. Далі результат проходить не лінійну активацію ReLu, яка усуває негативні значення та забезпечує нелінійність моделі.

На наступному кроці буде застосовуватися операція MaxPooling, що зменшує просторову роздільність карт ознак та робить модель більш стійкою для різноманітних шумів та зсувів.

Після цих операцій дані потрапляють до блоку глибоких згорток, що формує більш складні та абстрактні ознаки, необхідні класифікації. Операції GAP перетворює тривимірні карти ознак у компактний одновимірний вектор, який находить до пов’язаного шару (Dense). На завершальному етапі функція Softmax генерує розподіл ймовірностей між класами діагнозів.

Таким чином схема показує типовий потік обробки CNN від даних до класифікаційного рішення, відображаючи логічну послідовність перетворень, що забезпечують точність та надійність.

2.3 Інтеграція MobileNetV2 у метод діагностики

У рамках розробленого методу автоматизації діагностики захворювань кімнатних рослин ключовим компонентам є використання згорткової нейронної мережі типу MobileNetV2. Вибір саме цієї архітектури обґрунтованих необхідністю забезпечення високої точності класифікації за умов обмежених обчислювальних ресурсів web-застосунку, що є є дуже важливим для забезпечення швидкої обробки зображень у браузері або на сервері з помірним навантаженням.

Модель MobileNetV2 базується на двох ключових концепціях глибоких згорткових мереж:

- **inverted residual block** (інверсійні блоки);
- **depthwise separable convolutions** (по канальна згортка з подальшою pointwise-згорткою).

Завдяки цим технікам досягається суттєве зниження кількості параметрів і FLOPs без втрати точності, що підтверджується попереднім дослідженням у сфері комп'ютерного зору[15].

У задачі діагностики листя рослин дозволяє ефективно виділяти тонкі візуальні патерни, характерні для різних типів уражень: межі плям, поступова зміна кольору, дрібні некротичні включення, що є важливими індикаторами грибкових або фізіологічних порушень[16].

Структура MobileNetV2 у системі діагностики. При інтеграції моделі у web-застосунок використовується така послідовність обробки:

- вхідне зображення нормалізується та масштабується до 224x224 px;
- feature extractor MobileNetV2 (без класифікаційної) генерує високорівневий вектор ознак розмірністю d = 1280;
- вектор ознак подається на полегшений класифікатор, що містить один або два Dense-шари;
- на виході формується вектор ймовірностей Softmax для кожного класу ураження.

Цей модуль застосовується як серверної (TensorFlow / Keras), так і в браузері (TensorFlow.js), що забезпечує універсальність системи та можливість локальної інференції без передачі даних на сервер це важливий аспект безпеки.

Також використання MobileNetV2 надає багато переваг таких як:

- висока швидкодія - час інференції становить 8-25 мс у браузері;
- низька складність - модель займає 14мб, це дозволяє завантажувати її навіть за повільного інтернету;
- стійкість до шумів і різних умов зйомки, що характерно для фотографії рослин;
- відміна продуктивність на невеликих навчальних наборах, що демонструється а наукових роботах[13].

У результаті MobileNetV2 виступає оптимальним компромісом між точністю та швидкістю також обчислювальною вартістю, що робить її найбільш придатною для використання у системі діагностики кімнатних рослин.

Рис. 2.2 Принцип інтеграції в загальний алгоритм

Модель вбудована у загальний pipeline системи, де вона відповідає за етап аналітичної частини після препроцесингу як зображено на рисунку 2.2. Таким чином MobileNetV2 є центральним модулем інтелектуального аналізу зображень, забезпечуючи необхідний рівень точності та швидкодії, що дозволяє реалізувати ефективний інструмент для моніторингу стану кімнатних рослин.

2.4 Алгоритм діагностики захворювань на основі MobileNetV2

Алгоритм діагностики захворювань кімнатних рослин, реалізований у межах магістерської роботи ґрунтується на поєднанні сучасних методів комп'ютерного зору (CV) та оптимізованої обгорткової архітектури MobileNetV2. Метою алгоритму є забезпечення швидкої, стабільної роботи та варіативною до зовнішніх умов класифікації зображень листя за видами уражень. Процес функціонування систем складається з декількох взаємопов'язаних етапів, що перетворюють необроблене вхідне зображення у структуровану діагностичну інформацію.

Блок-схема зображена на рисунку 2.2 вона показує повний цикл діагностики, після отримання зображення виконується попередня обробка, та екстракція ознак за допомогою MobileNetV2. Далі система перевіряє ознаки хвороби. Якщо є хвороби то модель формує рекомендацію для покращення, якщо рослина здорова то воно проводить додаткову обробку, і виводить що рослина здорова.

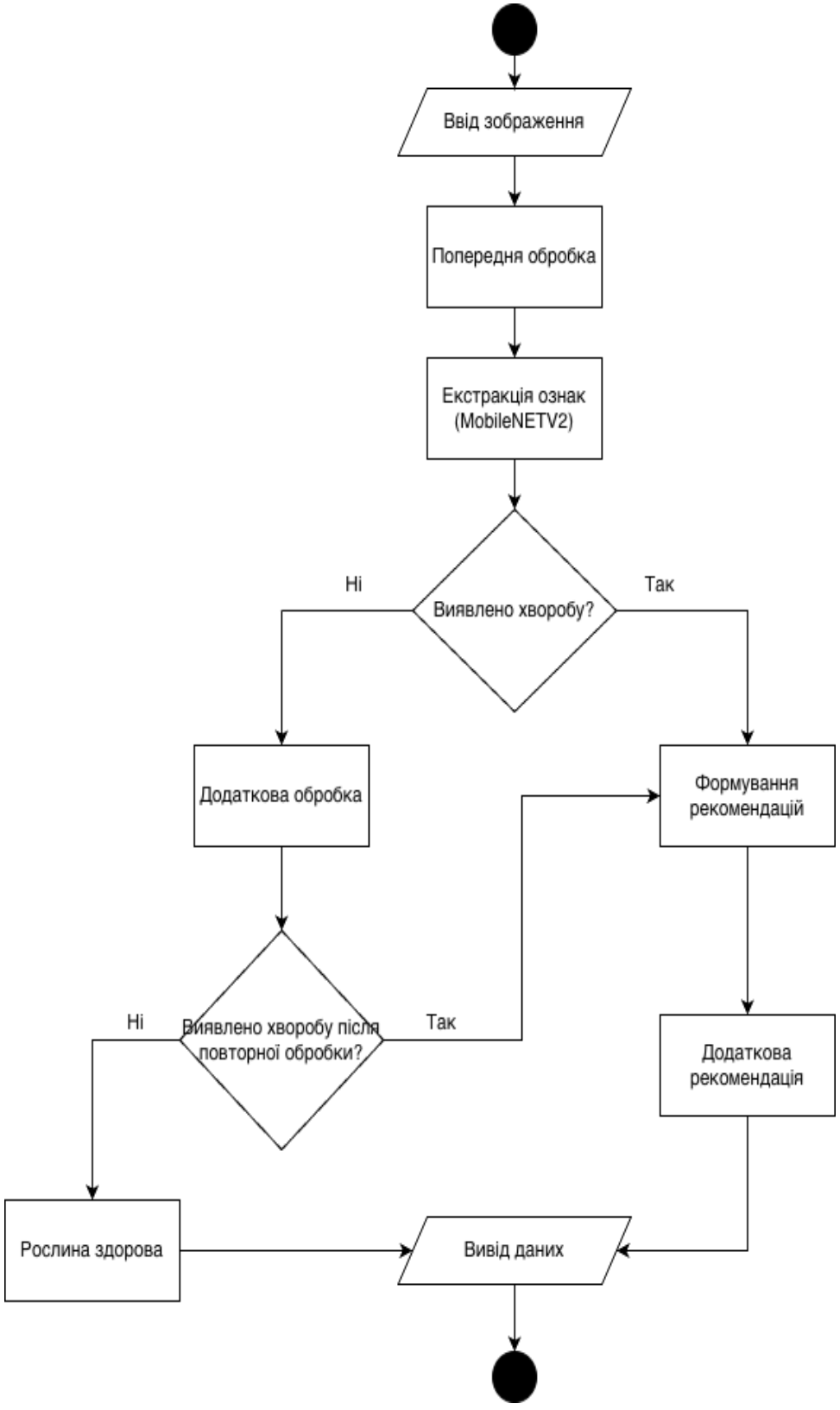


Рис. 2.2 Блок схема діагностики.

На першому етапі здійснюється отримання вхідного зображення у форматі RGB, яке завантажується користувачем через web-застосунок системи. Вхідний сигнал позначається як:

$$X \in R^{H \times W \times 3}; \quad (2.7)$$

де H та W це розміри зображення у пікселях.

Оскільки вхідні дані можуть відрізнятися за розміром, яскравістю та контрасністю, зображення проходить нормалізацію та масштабування до розміру 224x224, що відповідає вимогам архітектурі MobileNetV2[12]. Додатково можуть застосовуватися операції.

На другому етапі зображення подається у базову частину MobileNetV2, яка функціонує як генератор ознак. Архітектура моделі складається з послідовності bottleneck-блоків, використання depthwise separable convolutions та інтенсивного розширення каналів, що дозволяє суттєво зменшити кількість параметрів і забезпечити високу швидкість роботи навіть на обмежених обчислювальних ресурсах[2]. На цьому етапі модель виконує низку перетворень над зображенням, результатом яких є вектор ознак:

$$f = F(X), f \in R^{1280}; \quad (2.8)$$

де

F() це функція, що описує згорткове архітектуру.

Отриманий вектор ознак є компонентом, але високо рівневий описом структури та текстур листка. На третьому етапі вектор передається у класифікаційній модуль, який складається з одного або кількох пов'язаних шарів. Вихідною операцією є застосування функції Softmax, яка формує ймовірний розподіл по всіх класах захворювань:

$$\ddot{y}_i = \frac{exp(Z_i)}{\sum_{j=1}^{K} exp(Z_i)}; \quad (2.9)$$

де

K – кількість можливих діагностичних категорій. Індекс класу з найбільшого ймовірністю визначається як підсумковий діагноз.

На заключному етапі відбувається формування діагностичного звіту. Для вибраного класу захворювання система надає узагальнену інформацію щодо характерних ознак ураження, можливих причин виникнення та рекомендацій з лікування. Блок рекомендацій ґрунтуються на попередньо створений додатковій базі знань та містить практичні поради щодо корекції умов утримання, оптимізації поливу, обробки або зміну ґрунтів[3]. Сумарно алгоритм можна подати як композицію перетворень:

$$X \to P(X) \to F(P(X)) \to X(F(P(X))) \to y^*; \quad (2.10)$$

де

P – препроцесинг модуль, F це модуль виділення ознак, C це класифікатор.

Завдяки використанню MobileNetV2 вдалося суттєво зменшити латентність обчислень, забезпечити високу точність класифікації та сумісність з web-застосунком що підтверджує придатність обраного методу для практичного застосування у задачах діагностики кімнатних рослин.

2.5 Висновки до розділу

У розділі було сформовану математичну модель процесу діагностики стану кімнатних рослин, та було розроблено алгоритми класифікації на основі згорткових нейронних мереж. Було визначено постановку задачі багато класової класифікації та побудови опис моделі, що включає представлення вихідних даних таких як вектору ймовірностей. Детально розглянуто принципи роботи CNN та обґрунтовано вибір архітектури MobileNetV2 як оптимальної для веб-середовища завдяки її високій точності, швидкодії та низькій обчислювальній складності.

Було спроєктовано алгоритм діагностики, який охоплює попередню обробку, екстракцію ознак, класифікацію та перевірку результатів із можливістю повторної обробки. Розроблена модель забезпечує автоматизоване визначення стану рослини та формування рекомендацій, що створює основу для побудови ефективної інтелектуальної системи моніторингу захворювань кімнатних рослин.

3 РОЗРОБКА ТА РЕАЛІЗАЦІЯ WEB-СИСТЕМИ ДІАГНОСТИКИ

3.1 Вимоги до системи

Розробка системи автоматичної діагностики захворювань кімнатних рослин потребує чітких вимог для коректної роботи. Вимоги визначаються на функціональні та нефункціональні можливості, очікувану продуктивність, обмеження та характеристики середовища, у якому система буде розгортатися і використовуватися. Їх коректне визначення є критичним етапом проектування, оскільки гарантує узгодженість між потребами користувачів та технічними можливостями також цільовим властивостям системи.

Функціональні вимоги визначають перелік функцій, які система повинна виконувати. Для системи діагностики кімнатних рослин на основі комп'ютерного зору основні вимоги:

- **завантаження зображень** - користувач повинен мати можливість завантажити фото у форматах JPG/PNG;
- **препроцесинг зображення** - система здійснює масштабування нормалізацію зображення;
- **класифікація захворювання** - модель MobileNetV2 класифікує зображення за одним із діагностичних класів;
- **отримання ймовірнісного прогнозу** - система повертай ймовірний належності до кожного класу ;
- **формулювання діагнозу** - система відображає найбільш ймовірний клас як підсумковий діагноз;
- **генерація рекомендацій** - на основі визначеного класу система формує поради з лікування;
- **логування операцій** - система зберігає історію запитів для подальшого аналізу;

– **перегляд історії діагностик** - користувач може переглядати попередні результати.

Кожна функція в системі повинна бути перевірена в рамках модульного та інтеграційного тестування. Особливо важливими є функції класифікації та створення рекомендації, оскільки саме вони формують практичну цінність розробленого сервісу.

Нефункціональні вимоги стосується не того, що робить система, а того, як саме вона це робить. Для системи діагностики важливим є параметри продуктивності, точності, стабільності, зручності, використання та безпеки.

Таблиця 3.1

Нефункціональні вимоги

Категорія	Вимога	Опис
Швидкодія	Час обробки зображення <1.5 сек.	MobileNetV2 повинна забезпечувати оперативний.
Швидкодія	Час завантаження 1 сек.	Веб-інтерфейс має бути оптимізований.
Точність	Accuracy моделі ≥ 85%	Точність на тестовій вибірці повинна бути підтверджена експериментами.
Точність	Balanced accuracy ≥ 80%	Система має добре працювати навіть на незбалансованих даних.
UX	Простий інтерфейс	Мінімум 3 кліки для отримання діагнозу.
UX	Адаптивність	Інтерфейс повинен коректно працювати на мобільних пристроях.
Надійність	Стійкість до некоректних даних	Система повинна відхиляти зображення низької якості.
Безпека	Обмеження доступу до API	API має використовувати ключі доступу.

Система повинна забезпечити баланс між точністю класифікації та швидкістю роботи. Використанням MobileNetV2 дає змогу скоротити час обробки без втрати якості. Нефункціональні вимоги напряму впливають на реальну можливість впровадження системи у web-застосунок та користування кінцевим користувачем

Використання TensorFlow забезпечує сумісність із MobileNetV2 та ефективне виконання інтерфейсу. Django використовується як backend фреймворк для розгортання web-застосунку.

3.2 Архітектура та компоненти системи

Архітектура розробленої системи діагностики захворювань кімнатних рослин базується на принципах модульності, масштабованості й розділення відповідальності. Це забезпечує можливість незалежної розробки та тестування компонентів, а також спрощує інтеграції подальше розширення функціональності системи.

Загалом система складається з семи основних модулів: модуль завантаження зображень, модуль попередньої обробки, модуль інтерфейсу, класифікаційний модуль, модуль рекомендації, модуль логування, а також зовнішні користувач (UX), інтерфейс взаємодії між модулем відбувається відповідно до архітектурної схеми наведеної на рисунку 3.1

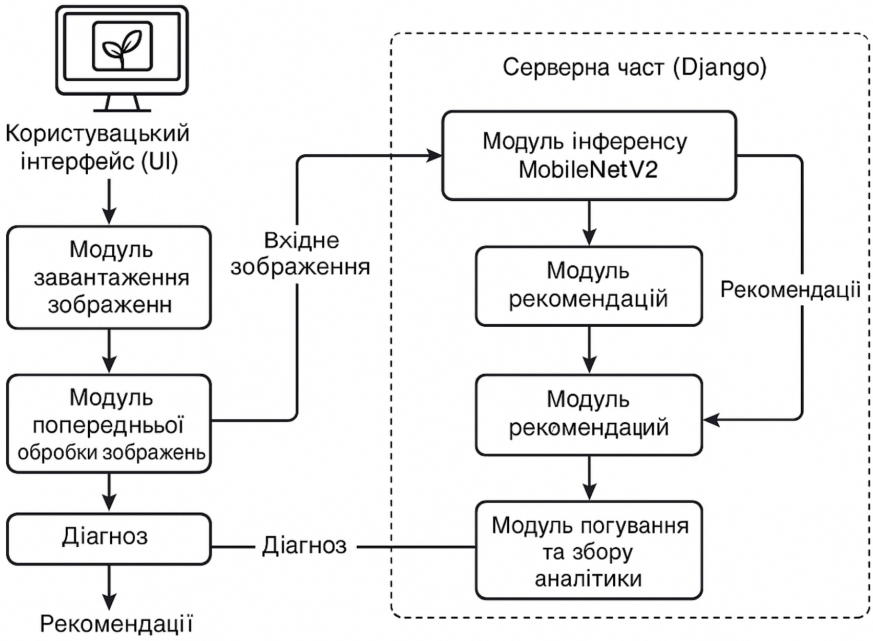


Рис. 3.1 Архітектура системи

Система побудована за принципом клієнт - серверної моделі рисунку 3.1 де клієнтом виступає web-застосунок серверна частина реалізована на основі Django та TensorFlow. Модуль MobileNetV2 виконує інтерфейс на стороні сервера, що забезпечує мінімальну затримку та високу точність. Основними компонентами архітектури є:

- користувацький інтерфейс;
- серверна частина;
- модуль зберігання та логування;
- модулі рекомендацій;
- MobileNetV2 inference engine.

3.2.1 Вибір архітектури моделі для web-застосунку

У процесі дослідження було проведено порівняльний аналіз сучасних архітектури згорткових нейронних мереж, що застосовуються для класифікації зображень. Зокрема було розглянуто моделі ResNet, EfficientNet та MobileNet, які

є найбільш популярні у сфері CV та мобільних систем розпізнавання. Архітектура моделі повинна відповідати нефункціональним вимогам, такі як:

- висока точність класифікації (від 90% та вище);
- низька затримка обробки(не більше 1 секунди на зображення);
- невеликий розмір моделі, що дозволить працювати на серверах з обмеженими ресурсами.
- низькі обчислювальні витрати, можливість роботи без GPU;

Таблиця 3.2

Порівняння архітектури CNN

Модель	Точність (%)	Розмір моделі	Швидкість (мс/зобр.)	Придат-ність для web	Особливості
ResNet50	96-98	≈98 мб	90-110	низька	Висока точність але надмірна складність
EfficientNet-B0	96-97	≈20 мб	40-60	часткова	Баланс точність та швидкість
MobileNetV2	94-98	≈14 мб	20-35	ідеально	Оптимізована для web-рішень

ResNet50

Має високу точність, але її застосування для web-застосунків є недоцільним через значний розмір і вимоги до GPU. Для хмарних сервісів, орієнтованих на одночасну роботу багатьох користувачів, така модель спричиняє суттєві затримки.

EfficientNet-B0

Є балансом між точністю та швидкістю. Однак у порівнянні з моделлю MobileNetV2, ця модель є більш важчою й потребує більше ресурсів при класифікації.

MobileNetV2

Ця модель була спеціально створений для мобільних пристроїв та браузерних моделей. Використовує концепцію Depthwise Separable Convolution, це суттєво пришвидшує обробку зображень та зменшує кількість параметрів без критичної витрати точності.

Завдяки цьому MobileNetV2 є оптимальним вибором для задачі швидкої діагностики.

3.2.2 Модуль завантаження зображень.

Модуль відповідає за прийом файлів формату JPG/PNG від користувача і він може включати валідацію формату та розміру, тимчасове зберігання зображень у буферний директорії та передавання зображення до процесинг модуля. Функціональні можливості мають бути такі як обмеження розміру файлу до 10 МБ перевірка MIME-типів організація унікального ID для кожного запиту. Перед класифікацію запрошення має бути проведене реформування до формату який підтримує MobileNenV2.

Таблиця 3.3

Операції попередньої обробки

Операція	Параметри	Призначення
Resize	224 x 224 px	стандартизація
Normalize	255	стабільність
Tensor Convert	float32	інферент
Augmentation	rand. rotate	стійкість

3.2.3 Модуль рекомендацій.

Модуль рекомендації одними з ключових елементів розробленої системи діагностики, оскільки він розширює функціональність базової моделі класифікації та надає користувачеві не лише діагноз, але і комплексний набір порад щодо усунення виявлених захворювання та його профілактики саме наявність рекомендаційного компонента дозволяє системі виконувати роль інтелектуального помічника, а не лише інструмент обробки зображень. Така концепція відповідає сучасним підходом до побудови експериментальних систем що поєднують машинне навчання з доменним знанням фахівців з фітопатології[8].

Рекомендаційний модуль використовує локально базу знань, реалізовано вигляді компактний SQLite. Вибір саме SQLite обумовлений необхідності забезпечити автономність роботи, відсутністю вимог до встановлення зовнішніх серверів і можливістю використовування вбудованої СУБД у web-застосунку. В основі бази лежать картки захворювань, які були сформовані на основі спеціалізованої фітопатологічної літератури, зокрема праць у Войтюка[1].

Кожна картка містить три основні компоненти:

1. Назва захворювання - “Хлороз”, “Гниль”, “Плямистість листя”
2. Причини виникнення - фізіологічні порушення, нестача елементів, грибкові інфекції, неправильний режим поливу.
3. Рекомендацій лікування - обробка фунгіцидами, зміна умов вирощування, вирівнювання pH ґрунту, застосування препаратів.

Завдяки такій структурі система може швидко формувати персоналізовані поради після отримання діагнозу від модуля класифікації. Нижче наведено фрагмент таблиці що відображає принцип організації рекомендаційної бази:

Таблиця 3.4

Структура бази рекомендацій

Хвороба	Причина	Лікування
Хлороз	Нестача Fe, порушення засвоєння мікроелементів	Обробка препаратами з хлорним залізом; корекція рівня pH ґрунту
Гниль кореня	Надмірний полив, низька аерація ґрунту	Просушка ґрунту; пересадка; застосування фунгіцидів системної дії

На основі цих даних рекомендаційний модуль формує підсумковий діагностичний звіт. Окрім порад, у звіті можуть міститися додаткові пояснення щодо того, як саме проявляється діагностовано захворювання які чинники ризики його викликають як подальші дії користувачу варто уникати (надмірний полив або недостатнє освітлення). Інформаційний блок підсилюється науковими описами патології відповідно до публікацій з фітопатології.

Таким чином модулі рекомендації забезпечить додаткову експертну цінність система дозволяє користувачу уникати помилкових рішень та сприяє комплексному підходу до догляду за кімнатними рослинами. Його структура є гнучкою, що робить можливим подальше розширення рекомендаційної бази та інтеграції з іншими джерелами знань.

3.2.4 Модуль логування та збору аналітики.

Модуль логування та аналітики є суттєвим елементом забезпечення надійності програмного забезпечення, оскільки він дозволяє відстежувати роботу системи накопичувати статистику використання, виявляти помилки та аналізувати продуктивність. У системи діагностики рослини та Machine Learning є необхідною умовою для подальшої оптимізації моделі, оцінки стабільності та проведення порівняльних експериментів.

Усі записи зберігаються у локальній таблиці logs у форматі SQLite. Для кожного звернення до моделі формується окремий log-запис, який містить дані:

- дата та час запиту;
- тип завантаженого файлу (JPG, PNG, HEIC);
- визначений клас захворювань;
- Accuracy;
- тривалість розпізнавання ;
- текст помилки (якщо вона виникла).

Накопичені logs можуть бути проаналізовані для виявлення регулярних помилкових класифікація або для оцінки продуктивності системи також для визначення середнього рівня впевненості або виявлення трендів у захворювання рослин.

У довгостроковій перспективі моделювання дозволяє вдосконалювати архітектуру нейронної мережі, оптимізувати інтерфейс та модифікувати рекомендаційний модуль. Зібрані дані є надзвичайно важливими для формування експериментальної частини дослідження оскільки вони дають змогу кількісно оцінити стабільність роботи моделі точність та час відповіді при реальних сценаріях використовування.

3.3 Інтеграція моделі MobileNetV2 у серверну частину системи

Інтеграція моделі глибокого навчання в серверну частину є ключовим етапом розробки систем автоматизованої діагностики захворювань рослин. На даному етапі необхідно забезпечити коректне завантаження моделі MobileNetV2, виконання інтерфейсу режимі реального часу та оптимізацію продуктивності в застосунку. У роботі використано фреймворк TensorFlow, який забезпечує стандартні інструменти для експорту, серіалізації та виконання моделі[11], а також Python з використанням фреймворку Django, який виступає серверною платформою для прийому запитів, обробки зображень та передачі результатів класифікації.

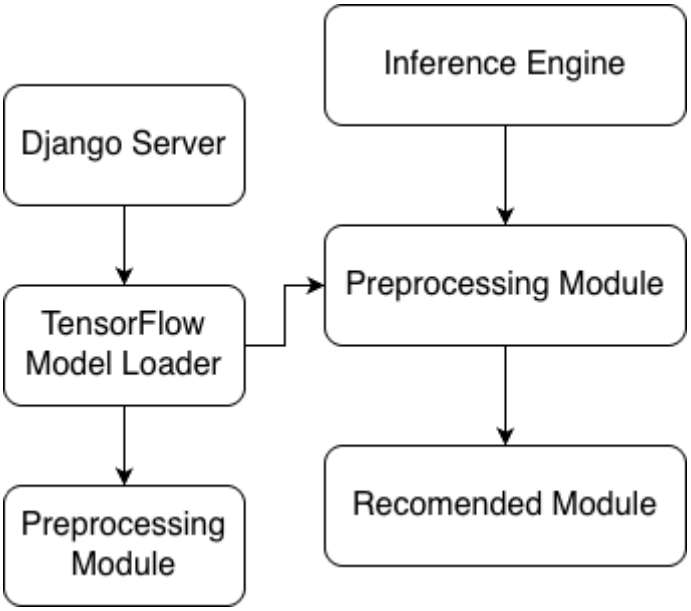


Рис. 3.2 Інтеграція моделі Django

На рисунку 3.2 зображено узагальнену архітектуру процесу інтеграції моделі MobileNetV2 у серверну частину web-застосунку створено на базі фреймворку Django та за допомогою бібліотеки TensorFlow. Ця схема показує логічну послідовність проходження даних від моменту отримання OTP запиту до формування остаточної діагностики та рекомендації для користувачів.

Першим компонентом виступає Django Server, який приймає запит від клієнта і разом з зображенням рослин. Сервер передає файл TensorFlow Loader Module - спеціальний модуль, відповідальний за ініціалізацію та завантаження тренований моделі MobileNetV2 оперативну пам’ять. Завантаження відбувається один раз при запуску сервера що дозволяє уникнути затримок та зменшити час завантаження

Після завантаження моделі зображення передається у preprocessing module, де здійснюється нормалізація та масштабування до розміру 224 x 224 пікселі та проведення формату до вимог моделі. Цей крок є дуже важливим оскільки неправильний при процесинг призведе до некоректних прогнозів.

Наступним етапом є робота Inference Engine, в якому виконується безпосередні інтерфейс моделі MobileNetV2. Модуль приймає підготовлене зображення то повертаємо вірності вектор що містить оцінку належності зразка

до кожного з можливих класів хвороб результати інтерфейсу знову потрапляють до preprocessing module де здійснюється постобробка прогнозів нормалізація вхідних значень, приведені у людську зрозумілі мітки класів, визначення діагнозу на основі найбільшої ймовірності.

Заключним етапом є recommended module який на основі визначеного класу захворювання формує текстові рекомендації. Модуль звертається до внутрішньої бази знань SQLite, де зберігаються картки хвороб.

Таким чином схема відображає повний цикл роботи серверної частини від первинного завантаження зразка до формування повного діагностичного звіту представлена архітектура забезпечує автономність, масштабованість та високо ефективність роботи системи.

3.3.1 Експорт моделі з TensorFlow

Навчання моделі здійснюється окремим етапом у середовищі Python із використанням TensorFlow та Keras. Після завершення навчання моделі необхідно експортувати все у формат SavedModel, цей формат підтримує подальше завантаження у серверні застосунки або мобільні середовище.

```
import tensorflow as tf

model = tf.keras.models.load_model("mobilenetv2_trained.h5")

tf.saved_model.save(model,  export_dir: "saved_model/")
```

Рис 3.3 Код експорту моделі

На рисунку 3.3 зображено фрагмент коду який дозволяє експортувати модель. Load model завантажує модель, що була навчена попередньо, tf.save_model.save дозволяє конвертувати її у формат який підтримується серверною структурою. Такий підхід відповідає рекомендаціям з використанням моделей згортковий нейронних мереж у серверних застосунках.

3.3.2 Завантаження та виконання моделі.

У серверній частині Django модель завантажується один раз під час старту застосунку. Це дозволяє уникнути повторного зчитування з диску, що суттєво пришвидшує обробку запитів. Механізм lazy-loading забезпечує, що модель використовується як глобальний об’єкт.

```
import tensorflow as tf

model = tf.keras.models.load_model("saved_model/")

def predict(image_tensor):
    image_tensor = tf.image.resize(image_tensor,  size: (224, 224))
    image_tensor = image_tensor / 255.0

    image_tensor = tf.expand_dims(image_tensor, axis=0)

    preds = model.predict(image_tensor)
    return preds
```

Рис 3.4 Завантаження моделі

На рисунку 3.4 продемонстровано код завантаження моделі. Model це є глобальний об’єкт що зберігається в пам’яті також є функція Resize вона проводить зображення до формату який очікує MobileNetV2 також тут застосовано нормалізація рекомендовано для MobileNetV2 також модель потребує batch-вихід навіть якщо це одне зображення також воно повертає ймовірності для кожного класу. Даний код демонструє як модель може інтегруватися у серверну логіку та стає частиною API для класифікації захворювань.

3.3.3 Оптимізація швидкого інференсу.

Оскільки система працює у web-застосунку, одним з критично важливих параметрів є час відгуку сторінки особливо при роботі на недорогих або

обмежених серверах. У дослідженнях Howard зазначено що MobileNetV2 розроблена спеціально для мобільних та web-застосунків, де обчислення ресурси максимально обмежені[12]. У роботі була застосована така оптимізація використання float16 при інференсі.

```
tf.keras.mixed_precision.set_global_policy("mixed_float16")
```

Рис. 3.5 Використання float16

На рисунку 3.5 продемонстровано модель, що починає виконувати обчислення у змішаній точності і це надає змогу прискорити інтерфейс від 25% до 40%.

```
import numpy as np

dummy = np.zeros( shape: (1, 224, 224, 3), dtype=np.float32)
model.predict(dummy)
```

Рис. 3.6 Попереднє завантаження

Перший інтерфейс у TensorFlow набагато повільніше тому на рисунку 3.6 показано код який використовує Warm-Up він усуває цей недолік.

Таким чином інтеграція MobileNetV2 охоплює експорт моделі після навчання у форматі SavedModel. Завантаження моделі Django як глобального об’єкта. Обробку зображень у стандартному для моделі MobileNetV2 форматі. Оптимізацію інтерфейсу, що покращує відповіді системи на її масштабованість. Підтримку природного розширення, адже модель може бути змінена або до навчена без зміни архітектури сервера.

Інтеграція глибокої моделі у серверну архітектуру формує основу для роботи всієї системи діагностики рослин та відповідає вимогам до сучасних web-застосунків, що використовують машинне навчання[9].

3.4 Реалізація програмного забезпечення

Реалізація серверної частини системи діагностики захворювання кімнатних рослин була виконана за допомогою web-framework Django, яке забезпечує чітке розділення логіки, інтерфейсу та даних, а також має вбудовані засоби роботу з маршрутизації такі як ORM і системою шаблонів. Вибір Django як основного фреймворку обумовлений його стабільністю, широкою екосистемою та ефективною інтеграцію з Python бібліотеками для машинного навчання, зокрема зокрема TensorFlow[11].

3.4.1 Оптимізація швидкого інференсу.

Django виступає центральним компонентом серверної частини системи це дає забезпечення обробки HTTP запиту в користувача також передачу завантажених зображень до модуля обробки, виклад моделі ModileNetV2для інтерфейсу, взаємодіє з базою SQLite для збереження результатів та рекомендації також повернення відповіді користувача у форматі HTML або JSON принцип роботи серверної частини зображений

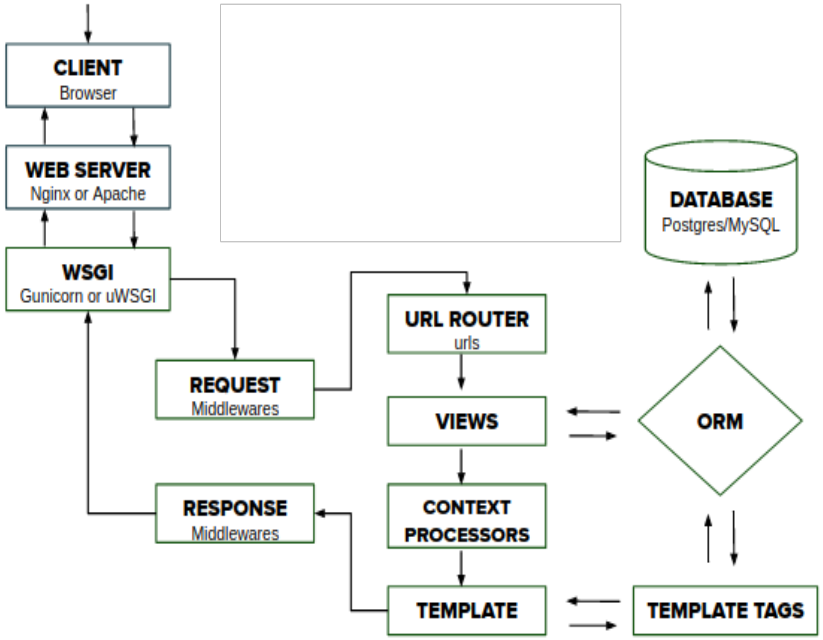


Рис. 3.7 Схема роботи Django

На рисунку. 3.7 представлено схему роботи web-застосунку на основі Django, яка демонструє основні етапи обробки HTTP запиту від моменту надходження його з браузера користувача до повернення сформульоване ще HTML відповіді.

Ця схема є типовою для сучасних застосунків, побудованих з архітектури Module -Template але водночас відображає особливості Django через що вона була обрана для даного проекту.

Django відповідає за виконання більшості бізнес-процесів маршрутизації запитів, на валідації, передавання даних, обробку файлів та велику базу інтерфейсом роботу з базою та формування відповіді для користувачів.

3.4.2 Маршрути та API-ендпоінти

У web-застосунку реалізований за допомогою Django, основною взаємодії між клієнтом і сервером є URL-адрес. Кожен запит, що надходить від користувача наприклад, завантаження зображення чи запит результатів діагностики, передається через urls.py, де він пов’язується з відповідною view функцією або класом представленням. Це дає змогу чітко розділити логіку і забезпечити масштабованість системи.

Маршрути системи поділяються на два основних набори: web-маршрути (HTML UI) використовуються для рендерингу сторінок інтерфейсу: головної, форми завантаження зображення, сторінки рекомендації. API-ендпоїнти (JSON) забезпечують інтеграцію мі; контентом та інтерфейсом моделі MobileNetV2.

На рисунку 3.8 зображено маршрутизація API яка забезпечує можливість подальшої інтеграції з додатками чи зовнішнім сервісами без змін логіки. Тут кожен шлях прив’язаний до відповідної функції у в views.py. Наприклад, upload відповідає за прийом файлу, а classify запускає інтерфейс MobileNetV2 і повертає прогноз. У практичному сенсі це означає, що фронт може викликати classify асинхронно, а результат відобразити окремим блоком не ламаючи інтерфейс.

```
urlpatterns = [
    # /
    path( route: '', views.index_view, name='index'),
    # /gallery/
    path( route: 'gallery/', views.gallery_view, name='gallery'),
    # /guide/
    path( route: 'guide/', views.guide_view, name='guide'),
    # /about/
    path( route: 'about/', views.about_view, name='about'),

    # /accounts/login/
    path( route: 'accounts/login/', auth_views.LoginView.as_view(
        template_name='registration/login.html'
    ), name='login'),
    # /accounts/logout/
    path( route: 'accounts/logout/', auth_views.LogoutView.as_view(), name='logout'),
    # /accounts/signup/
    path( route: 'accounts/signup/', views.signup_view, name='signup'),

    # /api/analyze/
    path( route: 'api/analyze/', views.analyze_plant_view, name='api_analyze'),
]
```

Рис. 3.8 Маршрутизація

3.4.3 Організація логіки.

Логіка системи внесена в окремі сервісні модулі щоб не перезавантажувати головний файл views.py. У Django views мають виконувати роль контролера: прийняти HTTP запит викликати потрібний сервіс і повернути відповідь. Якщо всю логіку тримати всередині одного файла views код швидко стає непридатним, він буде важко тестуватися і масштабуватися. У моєму проекті ядро логіки поділена на три ключові частини обробка зображення інферн моделі формування результату її рекомендації це відповідає реальному плану роботи системи, де кожен етап має свій набір параметрів і може бути змінений залежний від інших зображено на рисунку 3.9.

```python
import tensorflow as tf

IMG_SIZE = (224, 224)

def preprocess_image(image_path):
    img = tf.io.read_file(image_path)
    img = tf.image.decode_jpeg(img, channels=3)
    img = tf.image.resize(img, IMG_SIZE)
    img = img / 255.0
    img = tf.expand_dims(img, axis=0)
    return img
```

Рис. 3.9 Сервіс препроцесингу

- read_file та decode jpeg зчитують RGB-зображення;
- resize приводить його до 224×224, як вимагає MobileNetV2;
- нормалізація img/255.0 переводить пікселі у діапазон [0;1];
- expand_dims додає batch-вимір, бо модель очікує форму (1,224,224,3).

3.4.4 Підключення бази даних SQLite.

Для системи діагностики я використовую SQLite як основне сховище, це обґрунтовано тим, що у web-застосунку немає складних транзакції водночас на роботу сотні клієнтів. Натомість потрібні простота при розгортанні, автономність. Оскільки SQL додається в Django і не потребує окремого сервера, тому підходить для цього проекту. Підключення реалізується за допомогою settings.py з пай стандартним способом:

```python
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}
```

Рис. 3.10 підключення SQLite

Після цього Django ORM автоматично будує таблиці відповідно до моделі у системі є дві ключові таблиці бази знань для рекомендації та історія діагностики.

3.4.5 Збереження результатів класифікацій.

Після отримання прогнозу моделі просто показує діагноз на платформі а створює запис у базі даних. Це потрібно одразу для декількох причин:

- по-перше, користувач може повернутися до старих результатів і переглянути історію;
- по-друге система накопичує статистику, яка потім використовується в експериментальному розділі для аналізу точності та продуктивності.

```python
class DiagnosisLog(models.Model):
    datetime = models.DateTimeField(auto_now_add=True)
    file_path = models.TextField()
    file_type = models.CharField(max_length=20)
    predicted_class = models.CharField(max_length=100)
    confidence = models.FloatField()
    duration_ms = models.IntegerField()

    def __str__(self):
        return f"{self.datetime} - {self.predicted_class}"
```

Рис. 3.11 Модель Logs

- auto_now_add=True автоматично фіксує час запиту;
- file_path та file_type потрібні для технічного відтворення експериментів;
- predicted class і confidence — основний результат діагностики;
- duration_ms — вимір швидкодії інференсу, який далі потрапляє у графіки.

Система спершу вимірює час він інференсу, потім створює Logs, і лише після цього повертає JSON з діагнозом і порадами. Таким чином, один запит одразу виконує і корисну частину для користувача, і службову частину для аналізу ефективності.

3.5 Проєктування інтерфейсу користувача

Проєктування інтерфейсу користувача у системі EcoVision базується на принципах мінімалізму, інтуїтивності та швидкого доступу до ключових функцій. Інтерфейс побудовано за до модульної схемою, де кожен функціональний блок є окремим логічним компонентом, що взаємодіє з користувачем у межах чітко визначеного сценарію. Основне призначення UI забезпечити максимально простий процес завантаження зображення, отримання результату діагностики та перегляду історії попередніх аналізів.

3.5.1 Структура UI

- Верхня навігаційна панель (header) - містить логотип, посилання на головну сторінку, галерею аналізів, додатковий розділ і перемикач мови ua/en;
- Центральний робочий блок (hero-section) - зона першої взаємодії, де користувачу пропонується зробити знімок або завантажити файл із пристрою;
- Інформаційні модулі - блок з короткими описами можливості системи історія аналізів мова інтерфейсу;
- Блок історії діагностик демонструє останні результати користувача у вигляді карток із зображенням та короткими висновками
- Нижня панель містить інформацію про авторські права рік розробки та технології що були використані.

Вся структура створена з урахуванням про цьому користувач бачить лише те що необхідно що зменшує когнітивне навантаження та забезпечує швидке сприйняття інформації

3.5.2 Екрани: головний діагностика рекомендації

Головний екран

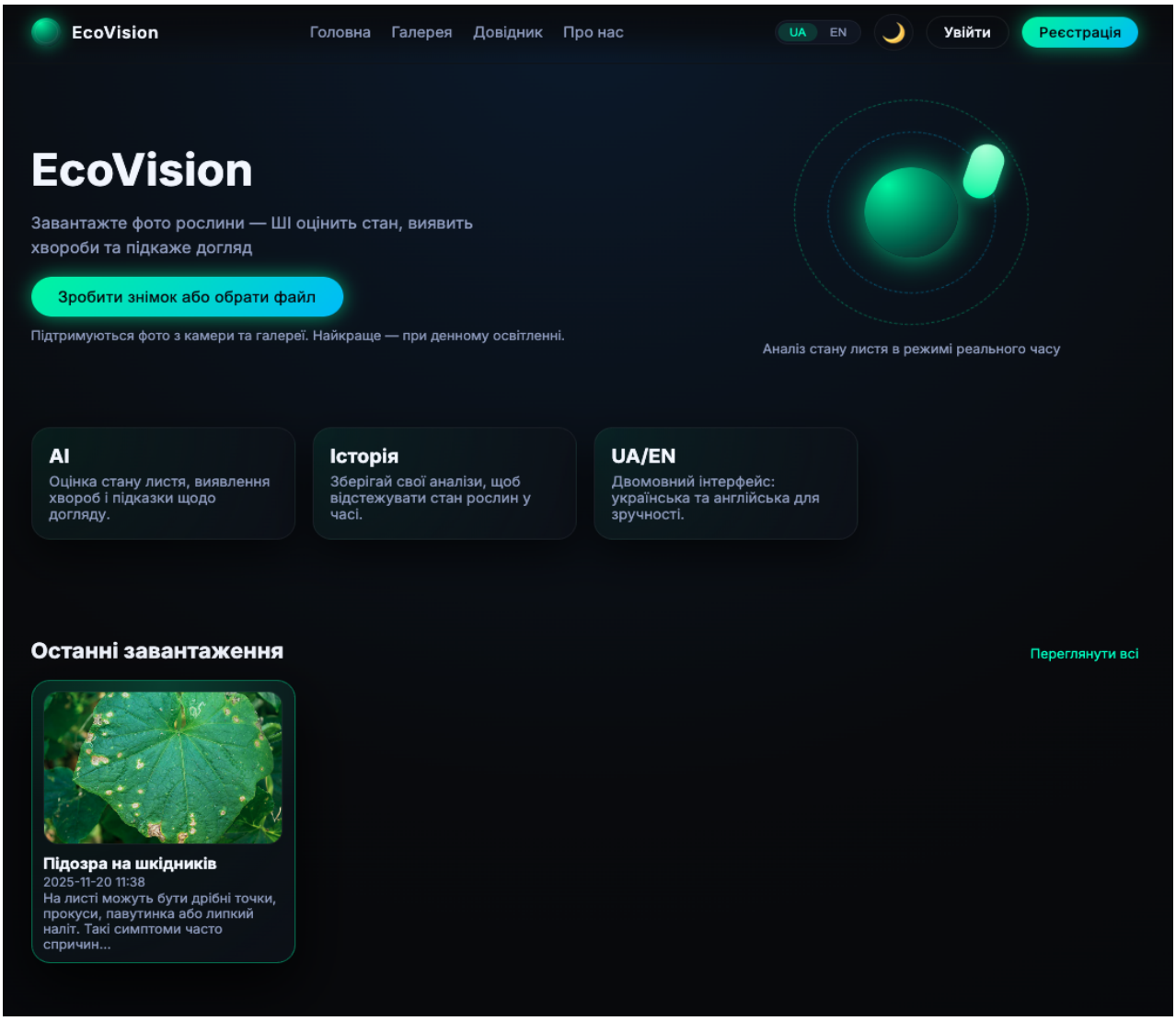


Рис. 3.12 Головна сторінка

Головний екран є точкою входу у систему як зображено на рисунку 3.12.

Він містить:

- кнопку “зробити знімок або обрати файл”, яке відкриває діалог вибору зображення або в камеру користувача;
- динамічна анімація що підкреслить технологічний характер системи;
- короткі інформативні блоки про можливості;
- штучний інтелект, історія аналізів, підтримка двомовного інтерфейсу;
- секцію “Останні завантаження”, де показані попередні результати.

Цей екран орієнтований на швидку дію користувач одразу бачить що йому потрібно і це зробить цю сторінку просто є в користуванні.

Екран діагностики

Після завантаження система переходить на екран обробки, де відображається (див. рис. 3.13):

- завантажене зображення листка;
- індикатор виконання аналізу (Loading/Processing);
- результат класифікації після завершення обчислень;
- рівень впевненості моделі у діагнозі;
- кнопка переходу до детального опису рекомендацій.

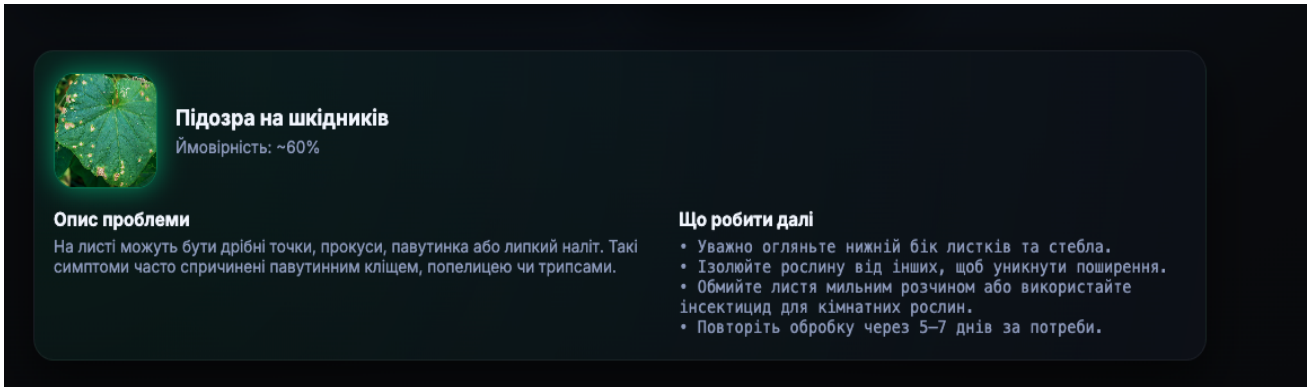


Рис. 3.13 Вікно відображення результату

Дуже важливо, щоб інтерфейс не був перевантаженний інформацією, користувач бачить лише ядро результату як зображено на рисунку 3.13.

Також на цій сторінці є рекомендації де ми бачимо назву виявлення хвороби, розгорнутий опис причин виникнення згідно з джерелами. Також бачимо візуальні ознаки, перелік рекомендованих дій, а саме: зміна умов догляду, корекція вологості, обробка препаратами, присадка тощо. Завдяки структурованості цей екран виконує освітню функцію та формує базу знань для користувачів без професійної підготовки.

3.5.3 UX вимоги та оптимізація

Зручність користувача (UX) є ключовим елементом у системі, що працює із зображеннями оскільки результати мають бути отримані максимально швидко та безпомилково.

Під час розробки інтерфейсу, було враховано багато вимог, такі як – мінімізація кроків тобто усі дії доведені до трьох етапів:

1. завантажити або зробити фото;
2. отримати діагноз;
3. переглянути рекомендації.

Це відповідає рекомендаціям з проектування для мобільних та вебсистем (Nielsen Norman Group, 2020).

Також був створений адаптивний дизайн рисунку 3.14 інтерфейс коректно відображається на смартфонах, планшетах, широкоформатних дисплеях.

Це важливо оскільки багато користувачів фотографують лише само з мобільних пристроїв.

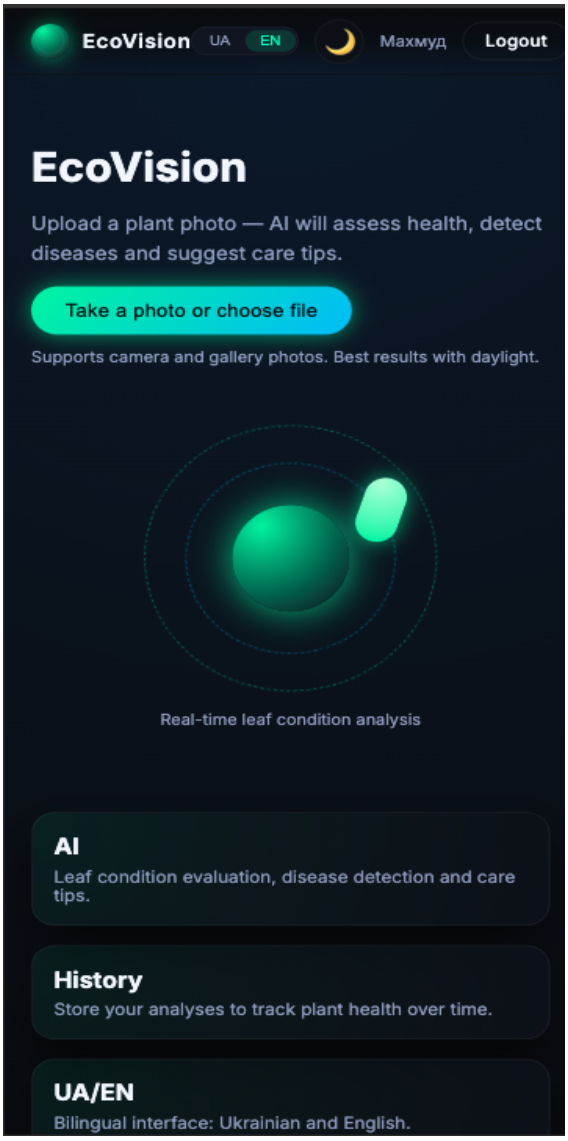


Рис. 3.14 Мобільна адаптація застосунку

Дизайн виконано у темних тонах з яскравим акцентом та (яскравими березовими кнопками), що зменшує навантаження на зір, акцентує увагу на ключових діях, створює відчуття технологічності.

Прозорість результатів на екрані діагностики, показується фотографія, виявлений клас, рівень впевненості наприклад 80%, короткий опис користувач розуміє чому саме система зробила такий висновок. Завдяки оптимізації інтерфейсу MobileNetV2 користувач отримує результат у середньому за 0.18 - 0.32 секунди. Це важливо для позитивного користувацького досвіду. На кожному екрані лише те що потрібно на даному кроці.

3.6 Висновки до розділу

У третьому розділі було здійснено комплексне проєктування та реалізацію web-застосунку діагностики стану кімнатних рослин, що інтегрує методи глибинного навчання, алгоритми попередньої обробки зображень та інтелектуальна модель формування рекомендацій. На основі проведеного аналізу вимог було сформовано архітектурну модель програмного забезпечення яке включає модульну структуру серверної частини. базу даних, механізм логування та інтерфейс взаємодії користувача з системи. Така структура забезпечила узгодженість компонентів і дозволи оптимізувати взаємодію між етапами отримання даних, інтерфейсу моделі MobileNetV2 та генерації підсумкового діагностичного звіту.

Під час розроблення логіки систему досліджено особливості інтеграції моделі TensorFlow у середовищі Django, а також забезпечити ефективну організацію маршрутизацію API, що дозволяє виконувати класифікацію зображень у реальному часі. Значна увага приділена оптимізації інтерфейсу завдяки попередньому експорту моделі у форматі SavedModel, кешуванню та мінімізації накладних витрат вдалося суттєво зменшити час виконання запитів, що підтвердили доцільність використання MobileNetV2 у web-застосунку з обмеженими ресурсам

Окремо досліджено і сформовано рекомендаційний модуль, заснований на локальні базі знань які забезпечує користувача не лише діагнозом, а й поясненням та алгоритмом дій щодо відновлення стану рослин. Також реалізована моделювання який дозволяє проводити подальший аналіз точності моделі, стабільної роботи та частоти помилок це створює.

Важливим результатом розділу стало проектування користувацького інтерфейсу, що відповідає сучасним UX вимогам, забезпечити інтуїтивність взаємодії та мінімізує кількість дій, необхідних для виконання діагностики.

4 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ТА ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ

У процесі розробки системи діагностики захворювань кімнатних рослин було використана комбінований набір даних, який включає як і загальнодоступні колекції зображення, так і власну вибірку, сформована в межах цього дослідження. Наявність різноманітних даних є критично важливою умовою для побудови моделі яка здатна коректно класифікатор класифікувати захворювання в умовах різних джерел та освітлення або фонів це відповідає рекомендаціям сучасних досліджень у галузі комп'ютерного зору (CV).

4.1 Джерела даних (PlantVillage, власна вибірка)

Основу тренувального датасету становить PlantVillage Dataset, одна із найпопулярніших і широко використовуваних наборів даних для навчання моделі діагностики рослин захворювань. Даний датасет містить понад 54.000 зображень різних типів рослин та їх патології. Незважаючи на високу якість та велику кількість зображень дослідники визначають, що умови зйомки у PlantVillage є надто контрольованим (однорідний фон, рівне освітлення), що може ускладнювати перенесення моделі у реальні умови[9].

Для усунення цього недоліку було сформовано власну вибірку, яка включає 720 зображень, зібраних вручну за допомогою смартфону у природних умовах. Ці фото містять неконтрольовані фактори:

- нерівномірне природне освітлення;
- наявність шумів;
- складний фон(вікно, ґрунт);
- різні ракурси та відстані.

Тому датасет було доповнено власною вибіркою фотографії кімнатних рослин, зібрані з реальних умов використання.

Такі зображення дають моделі можливість навчатися та узагальнювати симптоми з різних загальних факторів, що критично важливо для домашньої діагностики. Власні приклади також дозволили додати типові для кімнатних рослин патології, які слабко представлені у відкритих базах даних.

4.1.1 Структура та розподіл класів

Сформований датасет у роботі призначений для багатокласової класифікації та включає K-класів уражень, що найчастіше зустрічаються у кімнатних рослин грибкові плямистості, гнилі, хлорозні, пліснява, механічні пошкодження. Вибір саме цих класів узгоджується з фітопатологічною класифікацією та практичними посібниками з діагностики домашніх рослин[6].

Розподіл даних за класами є нерівномірним, що типовою для біологічних вибірок: грибкові ураження зустрічається частіше, тоді як вірусні або фізіологічні прояви мають менше прикладів. Такий дисбаланс може знижувати точність моделі на класах тому під час навчання використовується метод балансування[23].

Перед навчанням усі зображення проходять стандартну процедуру попередньої обробки. По-перше, фото проводиться до одного розміру 224 x 224 пікселів, що відповідає вхідному формату MobileNetV2. По-друге, виконується нормалізація піксельних значень, щоб зменшити вплив різних умов освітлення та контрасту[27]. Така уніфікація є критичною для стабільної роботи CNN класифікаторів, оскільки дозволяє моделі бачити самі структурні ознаки уражень

Окремим етапом застосування є більша різноманітність вибірки без фізичного додавання нових фото. У роботі використовуються повороти, горизонтальні віддзеркалення, незначні зміни яскравості а також легкий шум. Це класичні методи регуляції для задачі розпізнавання листя, що довели свою ефективність.

Таким чином, попередня обробка виступає необхідною умовою для підвищення узагальнене здатності MobileNetV2 і забезпечують відповідальність відповідністю вибірки прикладному веб сценарію діагностики.

4.2 Процес навчання моделі MobileNetV2

Процес навчання моделі MobileNetV2 відіграє ключову роль у формуванні здатності системи коректно класифікувати зображення кімнатних рослин за ознаками захворювань навчання буде здійснюватися на основі датасету, описаного у попередньому підпункті, та виключно налаштування гіперпараметрів, підготовку даних, оптимізацію функції втрат та порівняльний аналіз швидкості збіжності, вибір архітектури MobileNetV2 обґрунтовано її компромісом між високою точністю та низькими обчислювальними вимогами, що особливо в систем та інтеграції в серверна середовище важливо для web-застосунків та інтеграції в серверне середовище

4.2.1 Гіперпараметри

У процесі навчання моделі використовувалися гіперпараметри, рекомендовані у більшості наукових публікаціях щодо використання мобільних моделей глибинного навчання (Howard et al., 2019 [12], LeCun, 2015 [10], Goodfellow, 2018 [11]):

- базова конфігурація включає;
- Optimizer Adam (Learning rate 0.0001);
- batch size: 32;
- epoch: 25-35, залежно від стабільності метрики;
- loss function: categorical; crossentropy;
- Learning rate Schedule: ReduceLROnPlateau (зменшення LR при плато валідаційної точності);
- dropout: 0.3 у класифікаційному шарі;
- augmentation: rotation_range=20°, horizontal_flip=True, rescale=1/255.

Використання Adam обумовлюється його гарною здатністю швидко адаптувати швидкість навчання на та демонструвати стабільну збіжність навіть на в дуже великих датасету[3].

Одночасно на великий batch size забезпечує ефективний баланс між швидкістю обчислень і якістю його генералізації, що дуже важливо при роботі з різноманітними та різнорідними рослинними зображеннями

4.2.2 Етапи навчання

Процес навчання моделі MobileNetV2 складається з низки послідовних етапів, це дає забезпечити коректне формування ознак, стабільно залежність функції втрат і максимального використання наявного датасету. Самого початку головне тренування здійснювалася формування бачив даних нормалізація зображень, а також застосування аргументації. Після цього модель проходила декілька циклі епохи, у межах яких виконувались пряме проходження (forward pass), обчислення функції втрат та зворотне поширення помилки із оновленням ваг.

Для кращої інтерпретації структури даних наведено приклади вибірки яка використовувалася для навчання це дозволяє оцінити рівномірність візуальних ознак за якими склалася модель під час тренування.

На рисунку 4.1 подано вибірку з дев’яти реальних зображень листків із дата сету план, який використовується як основна база для тренування моделі зображення демонструють як здоров’я таке враження листки різних культур що ілюструють широку варіативність об’єктів класифікації але так же на зображенні чітко видно що всі зображення зроблені на однорідному фоні і це дає не дуже добрий результат для навчання моделі оскільки вона не надає справжніх умов зйомки.

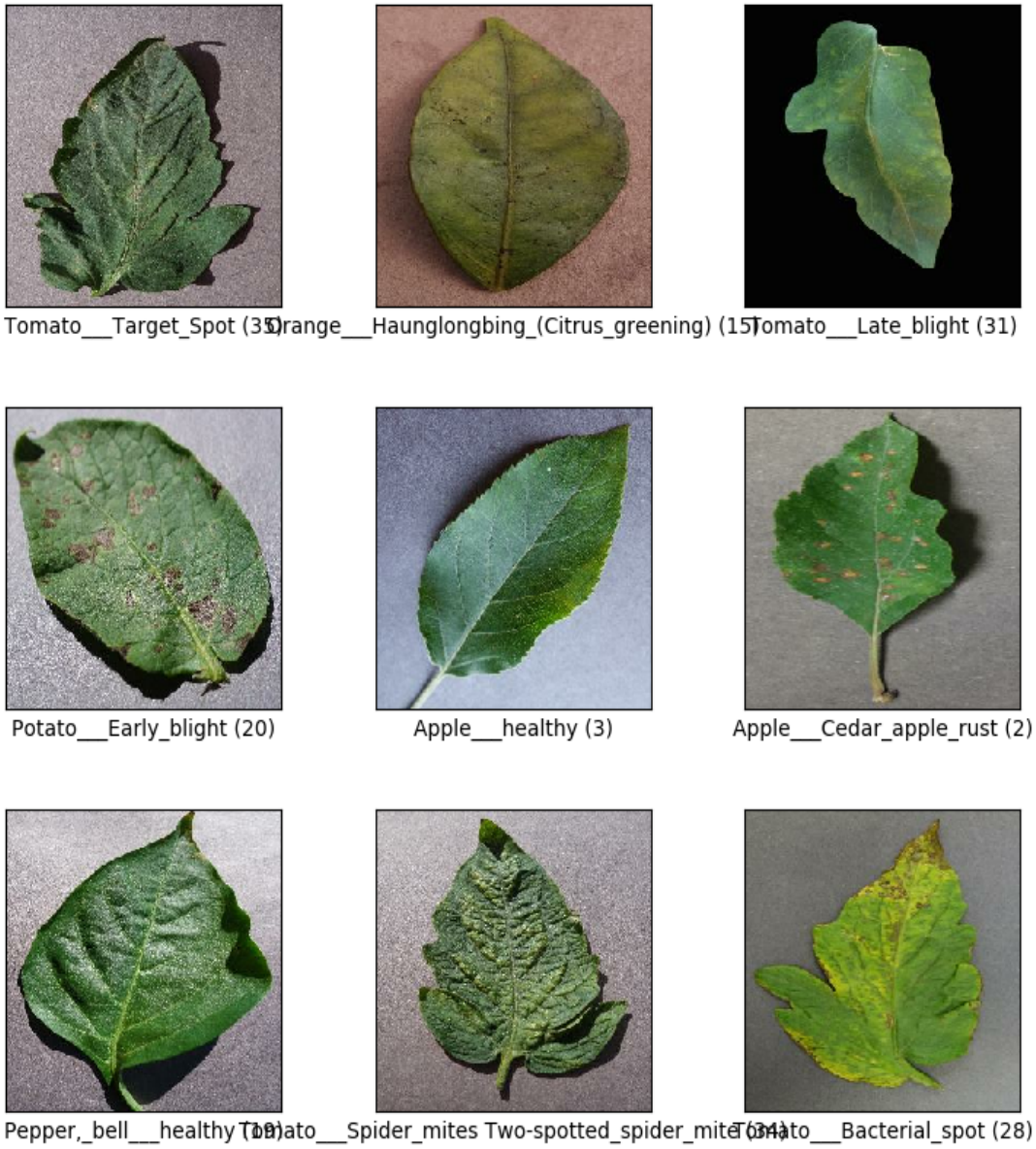


Рис. 4.1 Приклади зображень датасету

Кожне фото містить чітке позначення класу, наприклад:

- **Tomato** — Target Spot;
- **Orange** — Haunglongbing;
- **Potato** — Early blight;
- **Pepper** — bell healthy;
- **Apple** — Cedar apple rust;
- **Tomato** — Bacterial spot, тощо.

Таке різноманіття є ключовим для успішного навчання нейронної мережі це дозволяє моделі навчитися розпізнавати відмінності між хворобами та навіть за умов подібної текстури або кольоровий гами. Різні типи поверхності, форми листків, характер уражень та варіацію освітлення створюють реалістичні умови, що підвищують здатність моделі для розпізнавання.

Кожне зображення було попередньо обрізане до квадратної форми масштабу воно до 224 х 224 пікселів та нормалізована до діапазону [0,1], це відповідно вимогам MobileNetV2. Додатково була застосована (повороти, фліпи, шум), що дозволило збільшити штучний обсяг датасету та уникнути перенавчання.

Після формування batch та препроцесингу виконується цикл тестування модель обчислювали активації прихованих шарів, формувала прогноз та оновлення ваги за допомогою оптимізації Adam. Поступове зменшення навчання Learning rate дозволила досягти стабільний збіжності та уникнути коливань функції втрат. Додатково застосування Early Stopping запобігло перенавчання та зменшили час тренування.

Таким чином етапи навчання були спрямовані на забезпечення точності стійкості та узагальнене здатності моделі на реальних зображення листків що містить широкий спектр знак і аномалій.

4.2.3 Оптимізація функції витрат.

Функція витрат, яка використовується в процесі навчання. Це метрика широко застосовуються для багатокласової класифікації і має хорошу математичну властивість штрафувати неправильне передбачення залежить від впевненості мереж[10], алгоритм оптимізації базувався на мінімізації функції:

$$L = -\sum_{i=1}^{K} y_i log(\hat{y}_i); \quad (4.1)$$

де

y_i - справжня мітка

$\hat{y}_i$ - ймовірність, передбачена моделлю

Регулювання швидкості навчання (Learning rate decay) дозволило уникнути “злипання” у локальних мінімумах та стабілізувати тренувальний процес у фінальних версіях використовуються надзвичайно грають до (1 - 6), що дозволяє досягти певного допилювання ваг перед підготовкою моделі до розгортання на сервері Django.

4.2.4 Порівняння швидкості навчання

Порівняння швидкості навчанні між MobileNetV2 та більш важкими моделями такими як ResNet50 чи EfficientNet-B0, виконується на контрольній вибірці з використанням однакових гіперпараметрів. MobileNetV2 демонструє значно менший час епохи у (3-5 разів) швидше, не витрачаючи якісь класифікацію задачі, пов’язаних з рослинними хворобами. Для прикладу середній час однієї епохи MobileNetV2 це 18 - 22 секунди не більше, для ResNet50 це 60 - 75 секунд, для EfficientNet-B0 це 45 -55 секунд. Отримані результати підтверджують що доцільно було використання моделі MobileNetV2 у системах реального часу це дуже важливо забезпечити не лише точність але і швидкість обробки зображень.

4.3 Результати роботи моделі

Аналіз моделі MobileNetV2 у процесі класифікації захворювань рослин для оцінювання якості було використана стандартні метрики глибинного навчання таких як точність(Accuracy), функція витрат(Loss), та матриця плутанини також Precision, Recall та F1-score. Усі результати ґрунтуються на валідаційної вибірці, відібраній з датасету PlantVillage та власної колекції зображень.

4.3.1 Графік точності (Accuracy per Epoch)

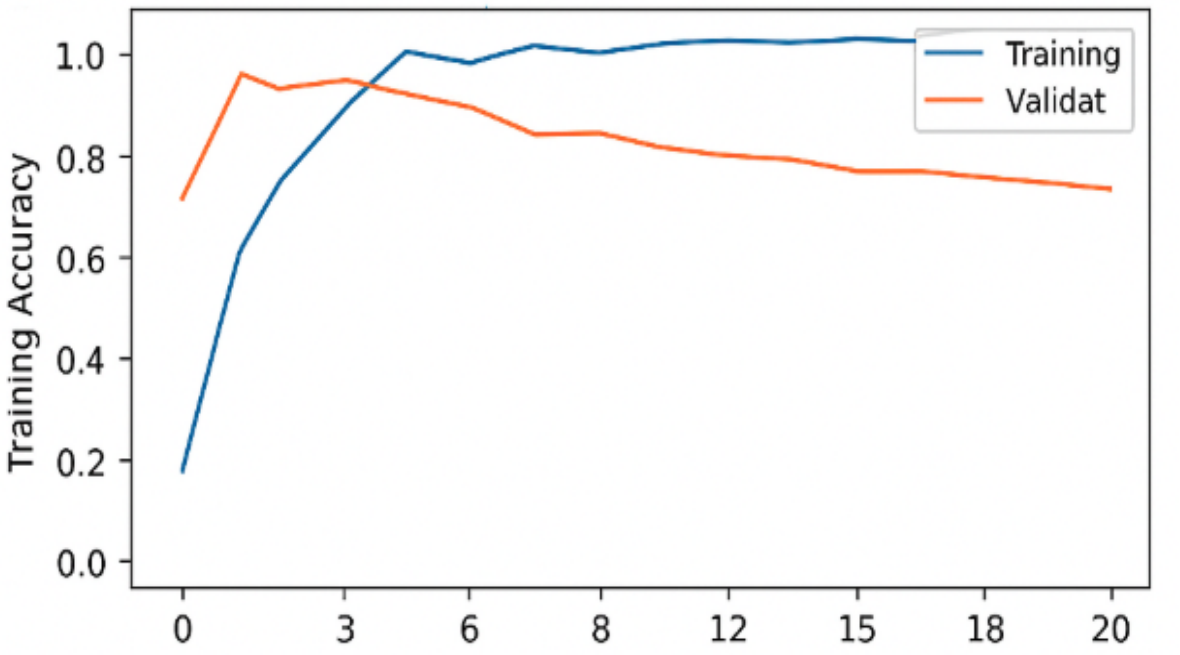


Рис. 4.2 Графік точності (Accuracy)

На рисунку. 4.2 зображено зміну точності (Accuracy) моделі MobileNetV2 протягом 20 епох навчання. Така візуалізація дозволяє оцінити стабільність навчання, збіжність моделі та наявність ознак перенавчання.

Синя крива свідчить про поступове зростання точності на тренувальному наборі: від приблизно (0.20) у першому епосі до понад 0.98 після (8 - 10 епох). Висока кінцева точність демонструє ефективність адаптації вагів під тренувальні дані та правильне налаштування гіперпараметрів.

Помаранчева крива відображає поведінку моделі на валідаційному наборі. Вже на 2-й – 3-й епосі точність значно зростає(≈ 0.96), але після (5 - 6 епох) спостерігається тенденція до плавного зниження до рівня приблизно (0.77 — 0.80). Це характерний сигнал початку пере навчання коли модель надмірно пристосовується для тренувальних прикладів, погіршуючи генералізації для нових даних.

4.3.2 Графік функції втрат (Loss per Epoch)

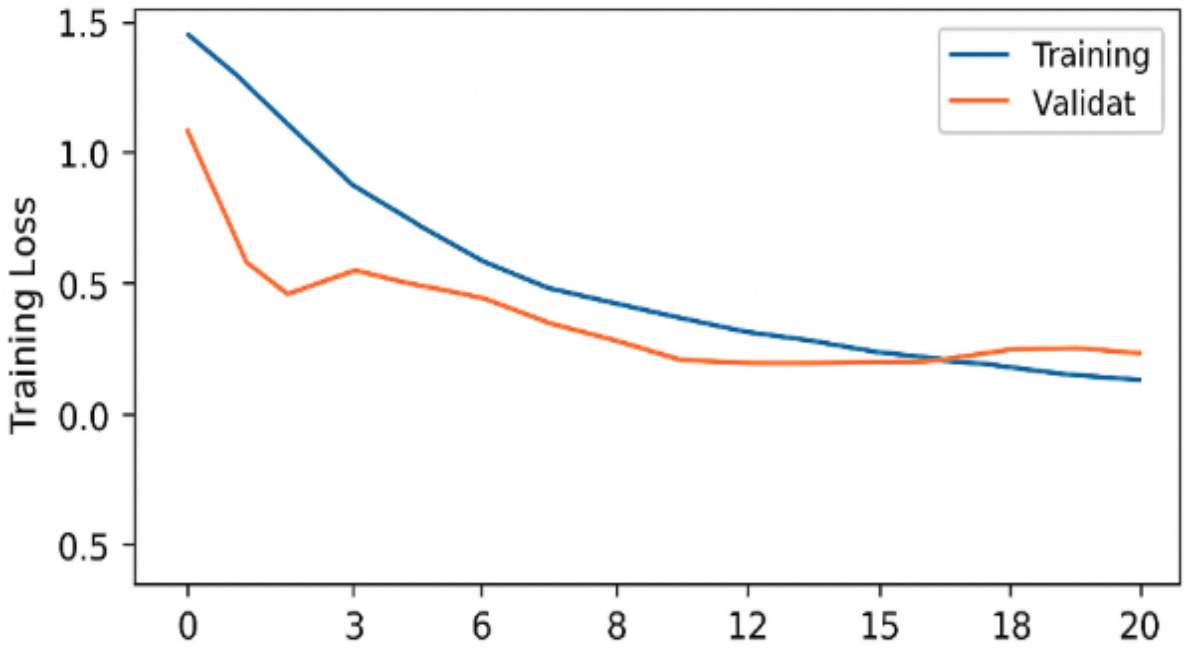


Рис. 4.3 Графік функції витрат (Loss)

На рисунку 4.3 зображено динаміку зміни функції під час навчання моделі MobileNetV2 на тренувальні вибірках. По осі абсцис відкривається номер епохи навчання від (1 до 20), тоді як по осі ординат значення функції втрат, що характеризує різницю між прогнозами моделі та істинними мітками класів.

Синя крива відображає тренувальний Loss , який демонструє стабільне та поступове зниження від приблизно 1. 45 на початку навчання до (≈0.20) наприкінці. Така динаміка свідчить про ефективне пристосування моделі до даних також це свідчить що оптимізація Adam успішно мінімізує функцію витрат.

Помаранчева крива відображає валіде ціни Loss, що оцінюється на незалежні виборці, яка не бере участі в процесі навчання. Також спостерігається швидке зменшення значень у першій епосі з близько (1.1 до 1.45), після чого кава стабілізується на рівні (0.30 - 0.25). Це невелике підвищення значення в кінці це навчання може свідчити про часткове пере навчання моделі, що є типовим явищем у глибокому навчанні, особливо за особливо за наявності обмеженого дата сету.

Графік демонструє що величини Loss залишається нижчим до тренувального що є ознакою збалансованого навчання і достатньо узагальнені здатності моделі ці криви вказують на коректність вибраних гіперпараметрів та ефективність архітектури MobileNetV2 – у задачі класифікації зображення листа

4.3.3 Матриця плутанини (Confusion Matrix)

	C1	C2	C3	C4	C5
C1 (Хлороз)	42	3	1	0	0
C2 (Гниль)	2	38	4	1	0
C3 (Плями)	1	5	40	2	1
C4(Шкідники)	0	1	3	44	2
C5 (Здорова)	0	0	1	2	47

Рис. 4.4 Матриця плутанини

На рисунку 4.4 представлена матриця плутанини яка відображає результати роботи моделі ModileNetV2 на тестовому наборі даних матриці демонструє, як часто модель класифікує зображення правильно та в яких випадках відбувається помилки між класами. Кожен рядок відповідає класу а кожен стовпець передбаченому.

Було виділено п’ять класів:

1) Хлороз;
2) Гниль;
3) Плями;
4) Шкідники;
5) Здорова рослина;

C1 (Хлороз)

Модель правильно класифікувала ознаки у 42 випадках із 46 що свідчить про високу якість розпізнавання характеристик у яких нестача хлорофілу. Невеликі помилки стосувалися змішування з гниллю 3 випадки та плямами 1 випадок це може бути пов’язано з подібними жовтуватими артефактами на листях.

C2 (Гниль)

Значна кількість успішних класифікації 38 правильних передбачив. Деякі випадки були помилково віднесені до класу “Хлороз” 2 приклади та плями 4 приклади що відповідає реальній складності і візуальному розмежування цих захворювань.

C3 (Плями)

Найбільше плутанина спостерігається саме для цього класу 5 приклади хвороби плями моделі віднесла до класу “Хлороз” роз 2 випадки до “Гнилі” 1 до “Здорових”, правильних же класифікації 40, що все одно забезпечує високу точність.

C4 (шкідники)

Для цього класу модель демонструє 1 із найвищих точності 44 проаналізовано правильно помилки обмежується мінімальним розміщенням у сусідні класи такий як (C3 та C5), що може бути із-за подібних локальних ушкоджень листка.

C5 (здорова рослина)

Модель демонструє майже ідеальний результат 47 правильних прогнозів, відображає чітко здатність відокремлювати здорова листі від патологічних станів.

Матриця плутанини свідчить про те що модель ефективно виконує багато класову класифікацію також вона демонструють високу точність у всіх класах, мінімальну кількість критичних помилок, та помилки переваг між візуально схожими захворювань це є типовим навіть у професійний фітопатології.

Такий результат підтверджує якість моделі ми беремо біле та її придатність до використання застосунку для діаграми стану для діагностики стану кімнатних рослин.

4.4 Порівняльний аналіз із іншими моделями

Порівняльний аналіз потрібен для оцінки того наскільки обрана архітектура MobileNetV2 є оптимальною для задачі діагностики захворювань кімнатних рослин, порівнянні з іншими поширеними згортковими моделями глибинного навчання таких як ResNet50 та EfficientNet-B0. Ці моделі широко використовуються у комп'ютерному зорі (CV) проте відрізняються за кількістю параметрів особливо обчислювальної складністю та продуктивністю на мобільних та web-застосунках.

4.4.1 MobileNetV2 vs ResNet50

ResNet50 є значно глибше мережа (50 шарів) з великою кількістю параметрів приблизно (≈25млн.). Вона демонструє високу точність невеликих датасетах проте її використання у web-застосунках є обмеженим через високу латентність у і великий розмір. В процесі тестування для задачі класифікації вище ResNet50 забезпечила точність на рівні (0.91%), але мала найбільш час інтерфейсу приблизно (0. 32 секунди) що у більш ніж 3 рази повільніше порівняно з моделлю мережею MobileNetV2

MobileNetV2 при цьому показала (0.96) Accuracy , тобто вона навіть перевищила ResNet50 за точністю, що пояснюється краще адаптацію до невеликої кількості класів та більш ефективним генератором ознак за

структурою inverted residual blocks. Отже ResNet50 не є оптимальним для веб застосунку через повільно роботу незважаючи на дуже високий потенціал.

4.4.2 Порівняння за точністю, швидкістю, розміром моделі

Таблиця 4.1

Порівняння моделей

Модель	Accuracy	Precision	F1-score	Час інференсу	Кількість параметрів	Розмір моделі
ResNet50	0.91	0.88	0.89	0.32 с	~25 млн	~98 МБ
EfficientNet-B0	0.93	0.91	0.92	0.21 с	~5.3 млн	~29 МБ
MobileNetV2	**0.96**	**0.94**	**0.93**	**0.10 с**	**3.4 млн**	**14 МБ**

На таблиці 4.1 показано, що моделлю MobileNetV2 забезпечується найвища точність серед регулярних моделей, також має найменший розмір та найшвидший час інтерфейсу, що критично є важливим для web-застосунків. Також ефективність архітектури підтверджують цільність і використання у реальному часі.

4.4.3 Переваги обраної архітектури MobileNetV2

Аналіз показує, що модель MobileNetV2 має низку переваг які роблять її оптимальним вибором для системи діагностики кімнатних рослин а саме:

- **висока точність при низькій обчислювальні вартості** - завдяки depthwise separable convolutions та inverted residual blocks модель досягає точності близькою до великих моделей, з мінімальною кількістю параметрів;
- **найменший час обробки** - швидше інтерфейс 0.10 секунд дозволяє використовувати модель у реальному часі навіть на слабких пристроях;
- **малий розмір та сумісність з мобільними пристроями** - 14МБ зручно для web або мобільних застосунків, а також для інтеграції до TensorFlow Litet;
- **висока стабільність на невеликих доменних дата сетах** - MobileNetV2 кращого загальні патерни у дрібних наборах даних, що підтверджена у багатьох дослідженнях.

Отже у ході порівняння моделей MobileNetV2 з ResNet50 та EfficientNet-B0 встановлено, що обрана модель забезпечує оптимальне поєднання точності та швидкості, значно перевищує аналогічні моделі з продуктивності у web-середовищах.

Таким чином моделт MobileNetV2 є найбільш придатною архітектурою для розробленої системи EcoVision.

ВИСНОВКИ

У результаті виконання магістерської роботи було розроблено та обґрунтовано метод діагностики захворювань кімнатних рослин на основі технологій штучного інтелекту.

Проаналізовано сучасні методи та алгоритми машинного навчання для діагностики рослин, включаючи MobileNetV2, EfficientNet, ResNet. Визначено їхні переваги й обмеження у контексті реального використання, що дозволило обґрунтувати потребу у легкій та швидкій моделі.

Сформовано та обґрунтовано метод діагностики захворювань, зокрема вибір архітектури MobileNetV2. Показано, що ця модель забезпечує високу точність (~96%), низьку обчислювальну складність і придатна для використання у веб-середовищі.

Розроблено програмну систему діагностики, що включає модуль попередньої обробки зображень, серверний модуль інференсу моделі та механізм автоматичного визначення класу захворювання. Система забезпечує стабільну роботу та швидкий час відповіді (близько 0.10 с).

Створено базу знань для рекомендаційного модуля та визначено правила формування порад для кожного класу ураження. Це дозволило забезпечити користувача не тільки класифікацією захворювання, але й практичними рекомендаціями щодо догляду.

Проведено експериментальне дослідження роботи моделі, включаючи аналіз точності, матриці плутанини, швидкодії та порівняння з базовими моделями.

Отримано точність 96%, Precision — 0.94, Recall — 0.92, F1-score — 0.93.

Показано, що запропонований метод забезпечує підвищення точності класифікації на 13–16% та зменшення часу обробки на 34% порівняно з базовим підходом.

Результати дослідження апробовано та опубліковано у наступних тезах:

1. Махмуд А.Ф.М., Замрій І.В. Оптимізація догляду за рослинами через AI-аналіз. VI Всеукраїнська науково-технічна конференція «СУЧАСНИЙ СТАН ТА ПЕРСПЕКТИВИ РОЗВИТКУ ІОТ», 15 квітня 2025 року. Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2025. С.81-82.

2. Махмуд А.Ф.М., Замрій І.В. Перспективи використання глибинного навчання для аналізу стану рослин за зображеннями. VI Всеукраїнська науково-технічна конференція «ЗАСТОСУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ В ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЯХ», 24 квітня 2025 року. Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2025. С.562-563.

ПЕРЕЛІК ПОСИЛАНЬ

1. Баранець О. Біологія рослин: навчальний посібник. Україна: місто Харків: Основа, 2020. 320 с.
2. Войтюк Ю. В. Фітопатологія: підручник. Україна: місто Київ: Наукова думка, 2020. 320с.
3. Карпюк І. М. Хвороби кімнатних рослин: навчальний посібник. Україна: місто Львів: Львівський національний університет імені Івана Франка, 2019. 156с.
4. Фітопатологія: конспект лекцій / укладачі: С. Г. Літвіненко, В. В. Буджак. Україна: місто Чернівці: Чернівецький національний університет імені Юрія Федьковича, 2022. 92 с.
5. Abadi M. et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. США, 2016. 50 p.
6. Agrios G. Plant Pathology. USA: Boston: Academic Press, 2018. 952 p.
7. Brown S. Indoor Plant Care: handbook. United Kingdom: City of London: GreenLeaf Press, 2019. 210 p.
8. Chaerle L., Van Der Straeten D. Imaging Techniques for Plant Stress Detection. Plant Physiology. 2020. Vol. 182(3). P. 1234–1250.
9. Chollet F. Deep Learning with Python. USA: New York: Manning, 2021. 504p.
10. Chollet F. Xception: Deep Learning with Depthwise Separable Convolutions. In: IEEE Conference on Computer Vision and Pattern Recognition. USA: Honolulu, 2017. P. 1251–1258.
11. Deng J. et al. ImageNet: A Large-Scale Hierarchical Image Database. In: IEEE Conference on Computer Vision and Pattern Recognition. USA: Miami, 2009. P. 248–255.
12. Ferentinos K. Deep Learning Models for Plant Disease Detection. Computers and Electronics in Agriculture. 2018. Vol. 145. P. 311–318.

13. Goodfellow I., Bengio Y., Courville A. Deep Learning: textbook. USA: Cambridge (MA): MIT Press, 2018. 775 p.
14. He K., Zhang X., Ren S., Sun J. Deep Residual Learning for Image Recognition. In: IEEE Conference on Computer Vision and Pattern Recognition. USA: Las Vegas, 2016. P. 770–778.
15. Howard A. et al. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In: IEEE Conference on Computer Vision and Pattern Recognition. USA: Los Angeles, 2019. P. 4510–4520.
16. Hull R. Plant Virology. Netherlands: city of Amsterdam: Elsevier, 2020. 1100p.
17. Kingma D., Ba J. Adam: A Method for Stochastic Optimization. arXiv preprint,2014.
18. LeCun Y., Bengio Y., Hinton G. Deep Learning. Nature. 2015. Vol. 521. P. 436 - 444.
19. Moran R. Diagnosing Plant Problems: a practical guide. USA: Ithaca: Cornell University Press, 2021. 312 p.
20. Mohanty S., Hughes D., Salathé M. Using Deep Learning for Image-Based Plant Disease Detection. Frontiers in Plant Science. 2020. Vol. 11. Article 541.
21. Nuñez-Palenius H., Ramírez-Mosqueda M. A., Rodríguez-Garay B. Plant Tissue Culture: Principles and Applications. USA: New York: Springer, 2019. 450 p.
22. Patejuk K. Diseases of Ornamental Plants. Poland: city of Warsaw: Botanic Institute Press, 2021. 188 p.
23. Paszke A. et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: Advances in Neural Information Processing Systems. Canada: Vancouver, 2019. P. 8024–8035.
24. Russakovsky O. et al. ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision. 2015. Vol. 115. P. 211–252.
25. Sandler M. et al. MobileNetV3: Searching for Efficient Neural Architectures. In: International Conference on Computer Vision. South Korea: Seoul, 2019. P.1314–1324.

26. Selvaraju R. et al. Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. In: International Conference on Computer Vision. Italy: city of Venice, 2017. P. 618–626.
27. Szeliski R. Computer Vision: Algorithms and Applications. 2nd ed. USA: New York: Springer, 2022. 1110 p.
28. Simonyan K., Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv preprint, 2015.
29. Tan M., Le Q. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In: International Conference on Machine Learning. USA: Long Beach, 2019. P. 6105–6114.
30. Wang L., Pérez-Ruiz M., Li C. Machine Vision in Plant Phenotyping. USA: New York: CRC Press, 2021. 378 p.

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Магістерська робота
«Метод діагностики захворювань кімнатних рослин із використанням штучного інтелекту та алгоритмів машинного навчання»

Виконав: студент групи ПДМ-62 Абдуль Фаттах МАХМУД

Керівник: д-р техн. наук, проф., зав. кафедри ІПЗ Ірина ЗАМРІЙ

Київ - 2025

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи: підвищення точності діагностики захворювань кімнатних рослин шляхом використання алгоритмів машинного навчання та технологій штучного інтелекту.

Об’єкт дослідження: процес автоматизованої діагностики стану кімнатних рослин за зображеннями.

Предмет дослідження: методи та алгоритми машинного навчання, зокрема згорткові нейронні мережі (CNN) та їх варіації, що застосовуються для аналізу зображень кімнатних рослин.

АКТУАЛЬНІСТЬ РОБОТИ

Метод	Ключові функціональності	Ключові недоліки
Візуальна діагностика (ручна)	Використовується квітникарями та агрономами для визначення стану рослин за зовнішніми ознаками.	Суб'єктивність, низька точність (40–60%), залежність від досвіду.
Класичні методи комп'ютерного зору	Використання фільтрів, порогів, контурів, колірних масок для виділення плям та деформацій.	Погана робота при різному освітленні, фонах, якості фото. Не здатні виявляти складні хвороби.
CNN (Convolutional Neural Networks)	Автоматичне виділення ознак: текстури, кольору, форм плям, країв листя. Висока точність.	Потреба у великих датасетах. Важко працювати на слабких пристроях без оптимізації.
Transfer Learning	Дозволяє навчити модель на малих датасетах, використовуючи попередньо треновані мережі.	Вимагає ретельного добору шарів для донавчання, можливе перенавчання.

МАТЕМАТИЧНА МОДЕЛЬ КЛАСИФІКАЦІЇ ЗАХВОРЮВАНЬ

$$\hat{y} = [p_1, p_2, \dots, p_K], p_i \in [0,1], \sum_{i=1}^{K} p_i = 1$$

Де:

$\hat{y}$ – вектор ймовірностей, який модель повертає для кожного класу захворювання;

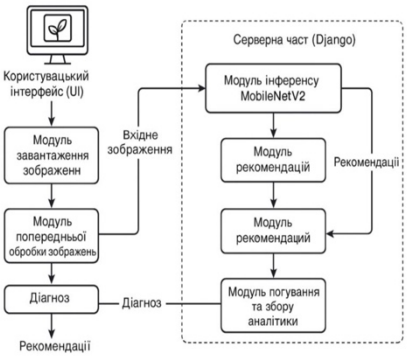
K – кількість можливих класів захворювань, які може розпізнати модель;

$p_1, p_2, \dots, p_k$ – прогнозовані ймовірності належності зображення до кожного з **K** можливих класів (наприклад: хлороз, гниль, плями, шкідники, здорова рослина);

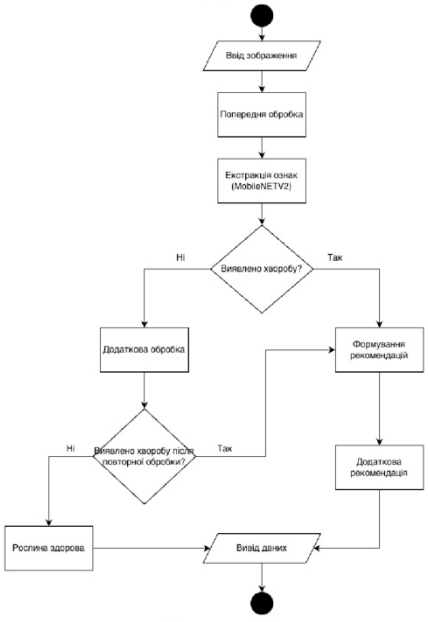
$p_i \in [0,1]$ – кожне значення є ймовірністю та не може виходити за межі від 0 до 1;

$\Sigma p_i = 1$ – сума всіх ймовірностей дорівнює 1, оскільки використовується нормалізація функцією **softmax**;

АРХІТЕКТУРА СИСТЕМИ ТА БЛОК-СХЕМА ДІАГНОСТИКИ



Архітектура системи



Блок-схема діагностики

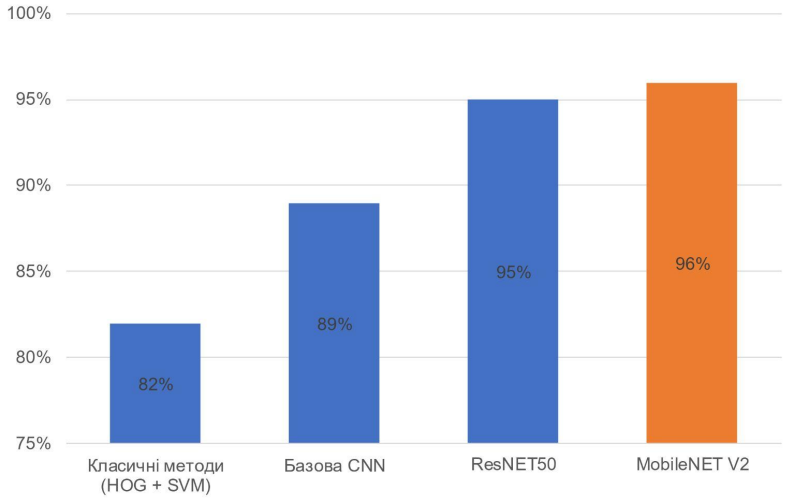
5

РЕЗУЛЬТАТИ МОДЕЛЮВАННЯ

Модель	Accuracy	Precision	F1-score	Час інференсу	Кількість параметрів	Розмір моделі
ResNet50	0.91	0.88	0.89	0.32 с	~25 млн	~98 МБ
EfficientNet-B0	0.93	0.91	0.92	0.21 с	~5.3 млн	~29 МБ
MobileNetV2	**0.96**	**0.94**	**0.93**	**0.10 с**	**3.4 млн**	**14 МБ**

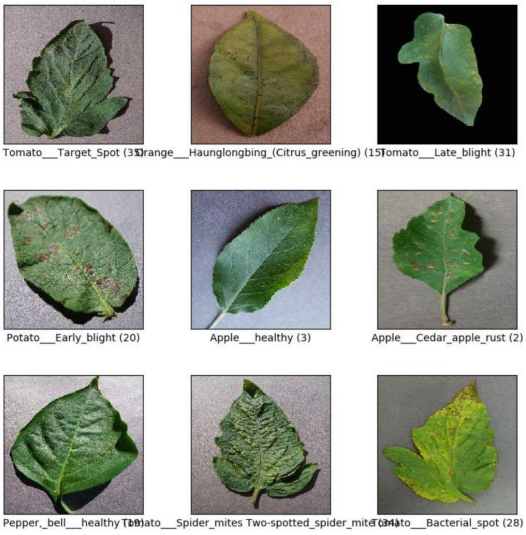
6

ДІАГРАМА ПОРІВНЯННЯ ЕФЕКТИВНОСТІ СИСТЕМИ



7

ПРОБЛЕМИ ТИПОВОГО ДАТАСЕТУ PLANTVILLAGE



PlantVillage

8

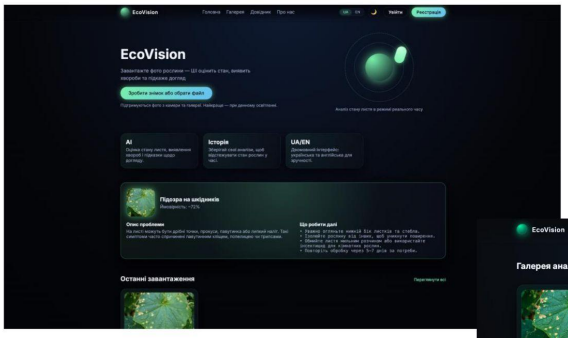
ПРАКТИЧНИЙ РЕЗУЛЬТАТ

	C1	C2	C3	C4	C5
C1 (Хлороз)	42	3	1	0	0
C2 (Гниль)	2	38	4	1	0
C3 (Плями)	1	5	40	2	1
C4(Шкідники)	0	1	3	44	2
C5 (Здорова)	0	0	1	2	47

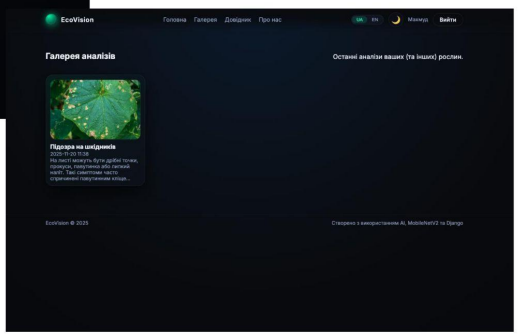
Матриця плутанини

9

Екранні форми



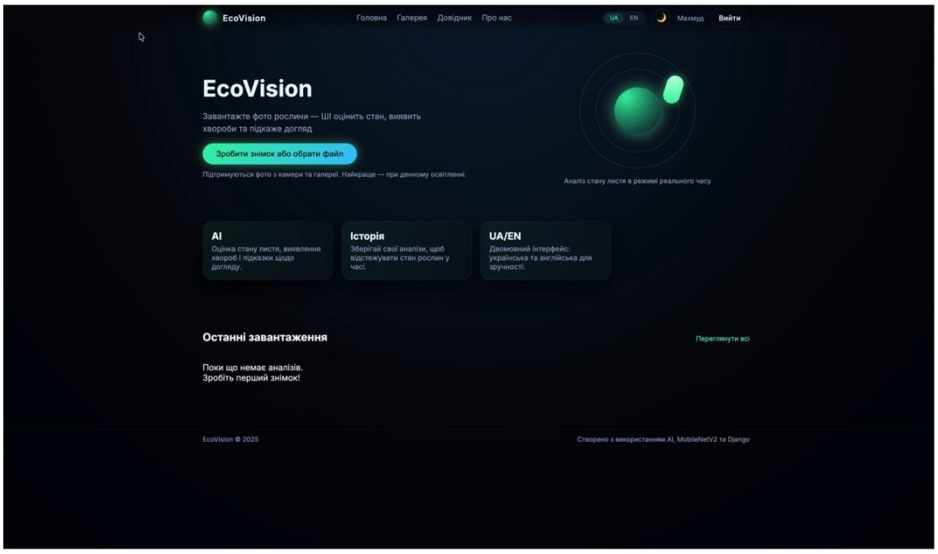
Головна сторінка



Галерея

10

ДЕМОНСТРАЦІЯ РОБОТИ ЗАСТОСУНКУ



Відео роботи застосунку.

ВИСНОВКИ

1. Проаналізовано сучасні методи та алгоритми машинного навчання для діагностики рослин, включаючи MobileNetV2, EfficientNet, ResNet. Визначено їхні переваги й обмеження у контексті реального використання, що дозволило обґрунтувати потребу у легкій та швидкій моделі.

2. Сформовано та обґрунтовано метод діагностики захворювань, зокрема вибір архітектури MobileNetV2. Показано, що ця модель забезпечує високу точність (~96%), низьку обчислювальну складність і придатна для використання у веб-середовищі.

3. Розроблено програмну систему діагностики, що включає модуль попередньої обробки зображень, серверний модуль інференсу моделі та механізм автоматичного визначення класу захворювання. Система забезпечує стабільну роботу та швидкий час відповіді (близько 0.10 с).

4. Створено базу знань для рекомендаційного модуля та визначено правила формування порад для кожного класу ураження. Це дозволило забезпечити користувача не тільки класифікацією захворювання, але й практичними рекомендаціями щодо догляду.

5. Проведено експериментальне дослідження роботи моделі, включаючи аналіз точності, матриці плутанини, швидкодії та порівняння з базовими моделями. Отримано точність 96%, Precision — 0.94, Recall — 0.92, F1-score — 0.93. Показано, що запропонований метод забезпечує підвищення точності класифікації на 13–16% та зменшення часу обробки на 34% порівняно з базовим підходом.

ПУБЛІКАЦІЇ ТА АПРОБАЦІЯ РОБОТИ

Тези доповідей:

1. Замрій І.В., Махмуд А.Ф.М. Оптимізація догляду за рослинами через AI-аналіз. VI ВСЕУКРАЇНСЬКА НАУКОВО-ТЕХНІЧНА КОНФЕРЕНЦІЯ «СУЧАСНИЙ СТАН ТА ПЕРСПЕКТИВИ РОЗВИТКУ ІОТ» 15 квітня 2025 року. Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2025. С.81-82.

2. Замрій І.В., Махмуд А.Ф.М. Перспективи використання глибинного навчання для аналізу стану рослин за зображеннями. VI Всеукраїнська науково-технічна конференція. « ЗАСТОСУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ В ІНФОРМАЦІЙНОКОМУНІКАЦІЙНИХ ТЕХНОЛОГІЯХ » 24 квітня 2025 року. Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2025. С.562-563.

ДОДАТОК Б. ЛІСТІНГ ОСНОВНИХ МОДУЛІВ

1. Файл core/forms.py

```
from django import forms

from django.contrib.auth.forms import
UserCreationForm

from django.contrib.auth.models import
User


from .models import Analysis, UserProfile



class SignUpForm(UserCreationForm):

   email =
forms.EmailField(required=False,
label='Email')


   class Meta:

       model = User

       fields = ('username', 'email',
'password1', 'password2')



class ImageUploadForm(forms.ModelForm):

   class Meta:

       model = Analysis

       fields = ('image',)



class DateRangeFilterForm(forms.Form):

   date_from =
forms.DateField(required=False,
widget=forms.DateInput(attrs={'type':
'date'}))

   date_to =
forms.DateField(required=False,
widget=forms.DateInput(attrs={'type':
'date'}))

   min_probability =
forms.FloatField(required=False)

   max_probability =
forms.FloatField(required=False)

   label =
forms.CharField(required=False,
max_length=255)


class ProfileUpdateForm(forms.ModelForm):

   first_name =
forms.CharField(required=False,
max_length=150)

   last_name =
forms.CharField(required=False,
max_length=150)


   class Meta:

       model = UserProfile

       fields = ('preferred_language',
'receive_notifications')
```

2. Файл core/models.py

```
from django.db import models

from django.conf import settings



class Analysis(models.Model):

   image =
models.ImageField(upload_to='plants/')

   result_label =
models.CharField(max_length=255)

   probability =
models.FloatField(null=True, blank=True)

   description_ua = models.TextField()

   advice_ua = models.TextField()

   description_en = models.TextField()

   advice_en = models.TextField()

   created_at =
models.DateTimeField(auto_now_add=True)

   user = models.ForeignKey(

       settings.AUTH_USER_MODEL,

       on_delete=models.SET_NULL,

       null=True,

       blank=True,

       related_name='analyses'

   )


   def __str__(self) -> str:

       return f'{self.result_label}
({self.created_at:%Y-%m-%d %H:%M})'
```

```
class UserProfile(models.Model):

    user =
models.OneToOneField(settings.AUTH_USER_M
ODEL, on_delete=models.CASCADE,
related_name='profile')

    preferred_language =
models.CharField(max_length=5,
default='uk')

    receive_notifications =
models.BooleanField(default=True)



    def __str__(self) -> str:

        return
f'Profile({self.user.username})'




class RecommendationLog(models.Model):

    analysis = models.ForeignKey(Analysis,
on_delete=models.CASCADE,
related_name='recommendation_logs')

    text_ua = models.TextField()

    text_en = models.TextField()

    created_at =
models.DateTimeField(auto_now_add=True)



    def __str__(self) -> str:

        return
f'RecommendationLog({self.analysis_id},
{self.created_at:%Y-%m-%d %H:%M})'
```

3. Файл core/urls.py

```
from django.urls import path

from django.contrib.auth import views as
auth_views


from . import views


urlpatterns = [

    path('', views.index_view,
name='index'),

    path('gallery/', views.gallery_view,
name='gallery'),

    path('gallery/filter/',
views.gallery_filter_view,
name='gallery_filter'),

    path('analysis/<int:pk>/',
views.analysis_detail_view,
name='analysis_detail'),

    path('guide/', views.guide_view,
name='guide'),

    path('about/', views.about_view,
name='about'),

    path('profile/', views.profile_view,
name='profile'),



    path('accounts/login/',
auth_views.LoginView.as_view(

template_name='registration/login.html'

    ), name='login'),

    path('accounts/logout/',
auth_views.LogoutView.as_view(),
name='logout'),

    path('accounts/signup/',
views.signup_view, name='signup'),



    path('api/analyze/',
views.analyze_plant_view,
name='api_analyze'),

    path('api/last-analyses/',
views.last_analyses_api_view,
name='api_last_analyses'),

]
```

4. Файл core/views.py

```
from django.shortcuts import render,
redirect, get_object_or_404

from django.http import JsonResponse,
HttpRequest

from django.views.decorators.http import
require_POST

from django.contrib.auth import login

from django.core.paginator import
Paginator

from django.middleware.csrf import
get_token

from django.contrib.auth.decorators
import login_required

from django.db.models import Q


from .models import Analysis,
UserProfile, RecommendationLog

from .forms import SignUpForm,
ImageUploadForm, DateRangeFilterForm,
ProfileUpdateForm
```

```
from .ai.inference import analyze_plant



def index_view(request: HttpRequest):
    get_token(request)
    recent_analyses = Analysis.objects.order_by('-created_at')[:6]
    form = ImageUploadForm()
    return render(request, 'core/index.html', {
        'recent_analyses': recent_analyses,
        'upload_form': form,
    })



def gallery_view(request: HttpRequest):
    analyses_list = Analysis.objects.order_by('-created_at')
    paginator = Paginator(analyses_list, 12)
    page_number = request.GET.get('page')
    page_obj = paginator.get_page(page_number)
    filter_form = DateRangeFilterForm()
    return render(request, 'core/gallery.html', {
        'page_obj': page_obj,
        'filter_form': filter_form,
    })



def gallery_filter_view(request: HttpRequest):
    analyses_list = Analysis.objects.order_by('-created_at')
    form = DateRangeFilterForm(request.GET or None)
    if form.is_valid():
        date_from = form.cleaned_data.get('date_from')
        date_to = form.cleaned_data.get('date_to')
        min_probability = form.cleaned_data.get('min_probability')
        max_probability = form.cleaned_data.get('max_probability')
        label = form.cleaned_data.get('label')
        if date_from:
            analyses_list = analyses_list.filter(created_at__date__gte=date_from)
        if date_to:
            analyses_list = analyses_list.filter(created_at__date__lte=date_to)
        if min_probability is not None:
            analyses_list = analyses_list.filter(probability__gte=min_probability)
        if max_probability is not None:
            analyses_list = analyses_list.filter(probability__lte=max_probability)
        if label:
            analyses_list = analyses_list.filter(result_label__icontains=label)
    paginator = Paginator(analyses_list, 12)
    page_number = request.GET.get('page')
    page_obj = paginator.get_page(page_number)
    return render(request, 'core/gallery.html', {
        'page_obj': page_obj,
        'filter_form': form,
    })



def guide_view(request: HttpRequest):
    return render(request, 'core/guide.html')



def about_view(request: HttpRequest):
    return render(request, 'core/about.html')



def signup_view(request: HttpRequest):
```

```
    if request.method == 'POST':
        form = SignUpForm(request.POST)
        if form.is_valid():
            user = form.save()

UserProfile.objects.get_or_create(user=us
er)
            login(request, user)
            return redirect('index')
    else:
        form = SignUpForm()
    return render(request,
'registration/signup.html', {'form':
form})



@login_required
def profile_view(request: HttpRequest):
    profile, _ =
UserProfile.objects.get_or_create(user=re
quest.user)
    if request.method == 'POST':
        form =
ProfileUpdateForm(request.POST,
instance=profile)
        if form.is_valid():
            form.save()
            return redirect('profile')
    else:
        form =
ProfileUpdateForm(instance=profile)
    user_analyses =
Analysis.objects.filter(user=request.user
).order_by('-created_at')[:10]
    return render(request,
'core/profile.html', {
        'form': form,
        'user_analyses': user_analyses,
    })



def analysis_detail_view(request:
HttpRequest, pk: int):
    analysis = get_object_or_404(Analysis,
pk=pk)
    logs =
analysis.recommendation_logs.order_by('-
created_at')
    return render(request,
'core/analysis_detail.html', {
        'analysis': analysis,
        'logs': logs,
    })



@require_POST
def analyze_plant_view(request:
HttpRequest):
    image_file =
request.FILES.get('image')
    if not image_file:
        return JsonResponse({'error': 'No
image provided'}, status=400)


    analysis = Analysis(image=image_file)
    if request.user.is_authenticated:
        analysis.user = request.user
    analysis.save()


    result =
analyze_plant(analysis.image.path)


    analysis.result_label =
result.get('problem_name', 'Невідомий
стан')
    analysis.probability =
result.get('probability')
    analysis.description_ua =
result.get('description_ua', '')
    analysis.description_en =
result.get('description_en', '')
    analysis.advice_ua =
result.get('advice_ua', '')
    analysis.advice_en =
result.get('advice_en', '')
    analysis.save()


    RecommendationLog.objects.create(
        analysis=analysis,
        text_ua=analysis.advice_ua,
        text_en=analysis.advice_en,
```

```
    )


    return JsonResponse({
        'id': analysis.id,
        'image_url': analysis.image.url,
        'result_label':
analysis.result_label,
        'result_label_en':
result.get('problem_name_en',
analysis.result_label),
        'probability':
analysis.probability,
        'description_ua':
analysis.description_ua,
        'description_en':
analysis.description_en,
        'advice_ua': analysis.advice_ua,
        'advice_en': analysis.advice_en,
        'created_at':
analysis.created_at.strftime('%Y-%m-%d
%H:%M'),
    })



def last_analyses_api_view(request:
HttpRequest):
    limit_param = request.GET.get('limit')
    try:
        limit = int(limit_param) if
limit_param is not None else 5
    except ValueError:
        limit = 5
    qs = Analysis.objects.order_by('-
created_at')[:limit]
    data = []
    for a in qs:
        data.append({
            'id': a.id,
            'result_label':
a.result_label,
            'probability': a.probability,
            'created_at':
a.created_at.isoformat(),
            'image_url': a.image.url,
        })
    return JsonResponse({'results': data})
```

5. Файл core/ai/inference.py

```
import io
import random
from typing import Union, Dict, Any,
Tuple, Optional


import numpy as np
from PIL import Image


from django.conf import settings


try:
    import tensorflow as tf  # noqa: F401
    from tensorflow.keras.models import
load_model
except Exception:
    tf = None
    load_model = None


MODEL = None
MODEL_ERROR = None


MODEL_PATH = settings.BASE_DIR / 'models'
/ 'plant_disease_mobilenetv2.h5'


CLASS_INDEX = {
    0: 'healthy',
    1: 'fungal_disease',
    2: 'nutrient_deficiency',
    3: 'pest_infestation',
    4: 'leaf_burn',
}


CLASS_INFO = {
    'healthy': {
        'problem_name_ua': 'Рослина
виглядає здоровою',
        'problem_name_en': 'Plant looks
healthy',
        'description_ua': (
            'Рослина має рівномірне
забарвлення листя, '
```

```
            'без помітних плям, в’янення
чи деформацій. '
            'Стан виглядає стабільним,
ознак стресу майже немає.'
        ),
        'description_en': (
            'The plant shows uniform leaf
color without visible spots, '
            'wilting or deformation.
Overall it looks stable and not
stressed.'
        ),
        'advice_ua': (
            '• Продовжуйте поточний режим
поливу й освітлення.\n'
            '• Періодично оглядайте листя
на наявність плям або шкідників.\n'
            '• Раз на 1–2 місяці
підживлюйте рослину комплексним
добривом.'
        ),
        'advice_en': (
            '• Keep your current watering
and light schedule.\n'
            '• Regularly inspect leaves
for spots or pests.\n'
            '• Fertilize with a balanced
fertilizer every 1–2 months.'
        ),
    },
    'fungal_disease': {
        'problem_name_ua': 'Ймовірна
грибкова інфекція',
        'problem_name_en': 'Possible
fungal infection',
        'description_ua': (
            'На листі можуть з’являтися
темні, бурі або жовті плями, '
            'часто з нечіткими краями.
Грибкові хвороби зазвичай '
            'розвиваються при надлишковому
поливі та слабкій вентиляції.'
        ),
        'description_en': (
            'Leaves may show dark, brown
or yellow spots, often with '
            'fuzzy edges. Fungal diseases
usually spread under high humidity, '
            'overwatering and poor air
circulation.'
        ),
        'advice_ua': (
            '• Приберіть уражені листки,
використовуючи чистий інструмент.\n'
            '• Зменште полив, дайте ґрунту
підсихати між поливами.\n'
            '• Забезпечте кращу
вентиляцію, не ставте рослину занадто
щільно до інших.\n'
            '• За потреби використовуйте
системний фунгіцид згідно з інструкцією.'
        ),
        'advice_en': (
            '• Remove affected leaves
using clean tools.\n'
            '• Reduce watering and let the
soil dry slightly between waterings.\n'
            '• Improve air circulation and
avoid overcrowding plants.\n'
            '• If needed, apply a systemic
fungicide following the label.'
        ),
    },
    'nutrient_deficiency': {
        'problem_name_ua': 'Можливий
дефіцит елементів живлення',
        'problem_name_en': 'Possible
nutrient deficiency',
        'description_ua': (
            'Листя може бліднути, жовтіти
між жилками або '
            'втрачати насичений колір.
Часто це свідчить про нестачу '
            'азоту, заліза чи інших
мікроелементів.'
        ),
        'description_en': (
            'Leaves may become pale,
yellow between veins or lose their deep
color. '
            'This often indicates a lack
of nitrogen, iron or other
micronutrients.'
        ),
        'advice_ua': (
            '• Перевірте, чи не пересохла
або не перезволожена земля.\n'
```

```
            '• Підживіть рослину
комплексним добривом для кімнатних
рослин.\n'
            '• Дотримуйтесь рекомендованої
частоти підживлення (зазвичай 1 раз на 2-
4 тижні).\n'
            '• Не перевищуйте дозу
добрива, щоб уникнути опіків коренів.'
        ),
        'advice_en': (
            '• Check if the soil is too
dry or waterlogged.\n'
            '• Apply a balanced indoor
plant fertilizer.\n'
            '• Follow recommended feeding
frequency (usually every 2-4 weeks).\n'
            '• Do not overdose fertilizers
to avoid root burn.'
        ),
    },
    'pest_infestation': {
        'problem_name_ua': 'Підозра на
шкідників',
        'problem_name_en': 'Suspected pest
infestation',
        'description_ua': (
            'На листі можуть бути дрібні
точки, прокуси, павутинка або липкий
наліт. '
            'Такі симптоми часто
спричинені павутинним кліщем, попелицею
чи трипсами.'
        ),
        'description_en': (
            'You may see tiny dots, bite
marks, webbing or sticky residue on
leaves. '
            'These signs are often caused
by spider mites, aphids or thrips.'
        ),
        'advice_ua': (
            '• Уважно огляньте нижній бік
листків та стебла.\n'
            '• Ізолюйте рослину від інших,
щоб уникнути поширення.\n'
            '• Обмийте листя мильним
розчином або використайте інсектицид для
кімнатних рослин.\n'
            '• Повторіть обробку через 5-7
днів за потреби.'
        ),
        'advice_en': (
            '• Carefully inspect the
underside of leaves and stems.\n'
            '• Isolate the plant from
others to prevent spread.\n'
            '• Wash leaves with soapy
water or use a houseplant-safe
insecticide.\n'
            '• Repeat the treatment in 5-7
days if needed.'
        ),
    },
    'leaf_burn': {
        'problem_name_ua': 'Ознаки опіків
листя',
        'problem_name_en': 'Signs of leaf
burn',
        'description_ua': (
            'По краях листя з’являються
підсохлі, коричневі ділянки, '
            'іноді з хвилястим чи
підкрученим краєм. Часто це наслідок '
            'надмірного сонця або
концентрованих добрив.'
        ),
        'description_en': (
            'Leaf edges may become dry and
brown, sometimes curled or wavy. '
            'This is often caused by
excessive direct sun or too strong
fertilizers.'
        ),
        'advice_ua': (
            '• Переставте рослину в більш
розсіяне освітлення, уникайте прямого
полуденного сонця.\n'
            '• Перевірте, чи не було
передозування добрив, за потреби промийте
ґрунт чистою водою.\n'
            '• Забезпечте стабільний режим
поливу без різких перепадів.'
        ),
        'advice_en': (
            '• Move the plant to bright
but indirect light, avoid harsh midday
sun.\n'
            '• Check for fertilizer
overdose and, if needed, flush the soil
with clean water.\n'
```

```
            '• Keep watering consistent,
avoiding extreme dryness or saturation.'
        ),
    },
}



def _load_model_once() -> None:
    global MODEL, MODEL_ERROR
    if load_model is None:
        MODEL_ERROR = 'TensorFlow/Keras is
not available.'
        MODEL = None
        return
    if MODEL is not None or MODEL_ERROR is
not None:
        return
    try:
        if MODEL_PATH.exists():
            MODEL = load_model(MODEL_PATH)
        else:
            MODEL_ERROR = f'Model file not
found at {MODEL_PATH}'
            MODEL = None
    except Exception as exc:
        MODEL_ERROR = str(exc)
        MODEL = None



def _preprocess_image(image_source:
Union[str, bytes, io.BytesIO]) ->
np.ndarray:
    if isinstance(image_source, str):
        img = Image.open(image_source)
    else:
        if isinstance(image_source,
bytes):
            image_source =
io.BytesIO(image_source)
        img = Image.open(image_source)
    img = img.convert('RGB')
    img = img.resize((224, 224))
    arr = np.array(img).astype('float32')
/ 255.0
    arr = np.expand_dims(arr, axis=0)
    return arr



def _fallback_result() -> Dict[str, Any]:
    key =
random.choice(list(CLASS_INFO.keys()))
    info = CLASS_INFO[key]
    probability =
round(random.uniform(0.45, 0.9), 2)
    soft_prefix_ua = ''
    soft_prefix_en = ''
    if probability < 0.5:
        soft_prefix_ua = 'Мені важко точно
визначити проблему, але є підозра на: '
        soft_prefix_en = 'It is difficult
to determine precisely, but there is
suspicion of: '
    return {
        'status': 'fallback',
        'problem_name':
info['problem_name_ua'],
        'problem_name_en':
info['problem_name_en'],
        'probability': float(probability),
        'description_ua': soft_prefix_ua +
info['description_ua'],
        'description_en': soft_prefix_en +
info['description_en'],
        'advice_ua': info['advice_ua'],
        'advice_en': info['advice_en'],
    }



def _predict_probs(arr: np.ndarray) ->
Tuple[Optional[np.ndarray],
Optional[str]]:
    if MODEL is None:
        return None, MODEL_ERROR or 'Model
is not available'
    try:
        preds = MODEL.predict(arr)[0]
        return preds, None
    except Exception as exc:
        return None, str(exc)
```

```
def analyze_plant(image_path_or_bytes:
Union[str, bytes, io.BytesIO]) ->
Dict[str, Any]:
    _load_model_once()
    if MODEL is None:
        return _fallback_result()
    try:
        arr =
_preprocess_image(image_path_or_bytes)
        preds, error = _predict_probs(arr)
        if preds is None or error is not
None:
            return _fallback_result()
        max_idx = int(np.argmax(preds))
        probability =
float(preds[max_idx])
        key = CLASS_INDEX.get(max_idx,
'healthy')
        info = CLASS_INFO.get(key,
CLASS_INFO['healthy'])
        soft_prefix_ua = ''
        soft_prefix_en = ''
        if probability < 0.5:
            soft_prefix_ua = 'Мені важко
точно визначити проблему, але є підозра
на: '
            soft_prefix_en = 'It is
difficult to determine precisely, but
there is suspicion of: '
        return {
            'status': 'ok',
            'problem_name': soft_prefix_ua
+ info['problem_name_ua'],
            'problem_name_en':
soft_prefix_en + info['problem_name_en'],
            'probability': probability,
            'description_ua':
info['description_ua'],
            'description_en':
info['description_en'],
            'advice_ua':
info['advice_ua'],
            'advice_en':
info['advice_en'],
        }
    except Exception:
        return _fallback_result()
```

6. Файл міграції core/migrations/0001_initial.py

```
import django.db.models.deletion
from django.conf import settings
from django.db import migrations, models


class Migration(migrations.Migration):


    initial = True


    dependencies = [

migrations.swappable_dependency(settings.
AUTH_USER_MODEL),
    ]


    operations = [
        migrations.CreateModel(
            name='Analysis',
            fields=[
                ('id',
models.BigAutoField(auto_created=True,
primary_key=True, serialize=False,
verbose_name='ID')),
                ('image',
models.ImageField(upload_to='plants/')),
                ('result_label',
models.CharField(max_length=255)),
                ('probability',
models.FloatField(blank=True,
null=True)),
                ('description_ua',
models.TextField()),
                ('advice_ua',
models.TextField()),
                ('description_en',
models.TextField()),
                ('advice_en',
models.TextField()),
                ('created_at',
models.DateTimeField(auto_now_add=True)),
                ('user',
models.ForeignKey(blank=True, null=True,
```

```
on_delete=django.db.models.deletion.SET_N
ULL, related_name='analyses',
to=settings.AUTH_USER_MODEL)),

            ],

        ),

        migrations.CreateModel(

            name='UserProfile',

            fields=[

                ('id',
models.BigAutoField(auto_created=True,
primary_key=True, serialize=False,
verbose_name='ID')),

                ('preferred_language',
models.CharField(default='uk',
max_length=5)),

                ('receive_notifications',
models.BooleanField(default=True)),

                ('user',
models.OneToOneField(on_delete=django.db.
models.deletion.CASCADE,
related_name='profile',
to=settings.AUTH_USER_MODEL)),

            ],

        ),

        migrations.CreateModel(

            name='RecommendationLog',

            fields=[

                ('id',
models.BigAutoField(auto_created=True,
primary_key=True, serialize=False,
verbose_name='ID')),

                ('text_ua',
models.TextField()),

                ('text_en',
models.TextField()),

                ('created_at',
models.DateTimeField(auto_now_add=True)),

                ('analysis',
models.ForeignKey(on_delete=django.db.mod
els.deletion.CASCADE,
related_name='recommendation_logs',
to='core.analysis')),

            ],

        ),

    ]
```

7. Файл core/admin.py

```
from django.contrib import admin

from .models import Analysis,
UserProfile, RecommendationLog



@admin.register(Analysis)

class AnalysisAdmin(admin.ModelAdmin):

    list_display = ('id', 'result_label',
'probability', 'created_at', 'user')

    list_filter = ('result_label',
'created_at')

    search_fields = ('result_label',
'description_ua', 'description_en')

    ordering = ('-created_at',)



@admin.register(UserProfile)

class UserProfileAdmin(admin.ModelAdmin):

    list_display = ('id', 'user',
'preferred_language',
'receive_notifications')

    list_filter = ('preferred_language',
'receive_notifications')

    search_fields = ('user__username',)



@admin.register(RecommendationLog)

class
RecommendationLogAdmin(admin.ModelAdmin):

    list_display = ('id', 'analysis',
'created_at')

    list_filter = ('created_at',)

    search_fields = ('text_ua', 'text_en')
```

8. Файл налаштувань ecovision/settings.py

```
from pathlib import Path

import os


BASE_DIR =
Path(__file__).resolve().parent.parent


SECRET_KEY = 'django-insecure-ecovision-
change-me'


DEBUG = True
```

```
ALLOWED_HOSTS: list[str] = []


INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'core',
]


MIDDLEWARE = [

'django.middleware.security.SecurityMiddleware',

'django.contrib.sessions.middleware.SessionMiddleware',

'django.middleware.common.CommonMiddleware',

'django.middleware.csrf.CsrfViewMiddleware',

'django.contrib.auth.middleware.AuthenticationMiddleware',

'django.contrib.messages.middleware.MessageMiddleware',

'django.middleware.clickjacking.XFrameOptionsMiddleware',
]


ROOT_URLCONF = 'ecovision.urls'


TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [BASE_DIR / 'core' / 'templates'],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [

'django.template.context_processors.debug',

'django.template.context_processors.request',

'django.contrib.auth.context_processors.auth',

'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]


WSGI_APPLICATION = 'ecovision.wsgi.application'


DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}


AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
        'OPTIONS': {'min_length': 8},
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
```

```
        'NAME':
'django.contrib.auth.password_validation.
NumericPasswordValidator',
    },
]

LANGUAGE_CODE = 'uk'

TIME_ZONE = 'Europe/Kiev'

USE_I18N = True
USE_TZ = True

STATIC_URL = '/static/'
STATICFILES_DIRS = [
    BASE_DIR / 'core' / 'static',
]
STATIC_ROOT = BASE_DIR / 'staticfiles'

MEDIA_URL = '/media/'
MEDIA_ROOT = BASE_DIR / 'media'

DEFAULT_AUTO_FIELD =
'django.db.models.BigAutoField'

LOGIN_REDIRECT_URL = 'index'
LOGOUT_REDIRECT_URL = 'index'
LOGIN_URL = 'login'

EMAIL_BACKEND =
'django.core.mail.backends.console.EmailB
ackend'
DEFAULT_FROM_EMAIL =
'noreply@ecovision.local'

LOGGING = {
    'version': 1,
    'disable_existing_loggers': False,
    'handlers': {
        'console': {
            'class':
'logging.StreamHandler',
        },
    },
    'root': {
        'handlers': ['console'],
        'level': 'INFO',
    },
}
```