**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВЕТЕРИНАРНОЇ МЕДИЦИНИ ТА БІОТЕХНОЛОГІЙ ІМ. С.З. ҐЖИЦЬКОГО**
**ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**
**КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

# КВАЛІФІКАЦІЙНА РОБОТА

другого (магістерського) рівня вищої освіти

на тему: **«Розробка інформаційної системи виявлення хвороб рослин на основі згорткових нейронних мереж»**

Виконав: студент групи Іт-61

Спеціальності 126 «Інформаційні системи та технології»
(шифр і назва)

Височанський Мар'ян Ярославович
(Прізвище та ініціали)

Керівник: д.т.н., професор Тригуба А.М.
(Прізвище та ініціали)

Рецензент: к.т.н., доцент Шарибура А.О.
(Прізвище та ініціали)

**ДУБЛЯНИ-2025**

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ПРИРОДОКОРИСТУВАННЯ
ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Другий (магістерський) рівень вищої освіти
Спеціальність 126 «Інформаційні системи та технології»

«ЗАТВЕРДЖУЮ»

Завідувач кафедри_____

д.т.н., проф. А.М. Тригуба
«____» _____2025 р.

# ЗАВДАННЯ

на кваліфікаційну роботу студенту

Височанському Мар'яну Ярославовичу

1. Тема роботи: «Розробка інформаційної системи виявлення хвороб рослин на основі згорткових нейронних мереж»

Керівник роботи Тригуба Анатолій Миколайович, професор
затверджені наказом по університету від 28.02.2025 року № 140/к-с.

2. Строк подання студентом роботи 10.12.2025 р.

3. Вихідні дані до роботи: *набір зображень листків рослин; інформація про класи захворювань і здорові стани рослин; параметри згорткової нейронної мережі EfficientNetB0; дані для аугментації та нормалізації зображень; програмні засоби реалізації (Python, TensorFlow, Keras, Streamlit, OpenCV).*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити)
*Вступ.*
*Аналіз стану автоматизованого виявлення хвороб рослин.*
*Вибір інструментарію та проєктування інформаційної системи виявлення хвороб рослин на основі згорткових нейронних мереж.*
*Розробка програмних модулів інформаційної системи виявлення хвороб рослин на основі згорткових нейронних мереж.*
*Охорона праці та безпека у надзвичайних ситуаціях.*
*Визначення економічної ефективності.*
*Висновки та пропозиції.*
*Список використаної літератури.*

5. Перелік ілюстраційного матеріалу (з точним зазначенням обов»язкових слайдів): *аналіз стану автоматизованого виявлення хвороб рослин; вибір інструментарію та проєктування інформаційної системи виявлення хвороб рослин на основі згорткових нейронних мереж; розробка програмних модулів інформаційної системи виявлення хвороб рослин на основі згорткових нейронних мереж; економічна ефективність.*

6. Консультанти з розділів:

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| 1, 2, 3, 5 | *Тригуба А.М., зав. кафедри інформаційних технологій* | | |
| 4 | *Городецький І.М., доцент кафедри інженерної механіки* | | |

7. Дата видачі завдання 28 лютого 2025 р.

Календарний план

| № з/п | Назва етапів кваліфікаційної роботи | Терміни виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | *Написання першого розділу* | *28.02-20.03.25* | |
| 2 | *Виконання другого розділу та аркушів ілюстраційного матеріалу до нього* | *21.03-14.06.25* | |
| 3. | *Виконання третього розділу та аркушів ілюстраційного матеріалу до нього* | *15.06.25-10.07.25* | |
| 4. | *Написання розділу «Охорона праці та безпека у надзвичайних ситуаціях»* | *11.07-31.08.25* | |
| 5. | *Оцінення ефективності запропонованої системи* | *01.09-31.10.25* | |
| 6. | *Завершення оформлення розрахунково-пояснювальної записки та аркушів ілюстраційного матеріалу* | *01-30.11.25* | |
| 7. | *Завершення роботи вцілому* | *01-10.12.25* | |

Студент _____ Височанський М.Я.
(підпис)

Керівник роботи _____ Тригуба А.М.
(підпис)

УДК 004.89:004.932+631.52

Розробка інформаційної системи виявлення хвороб рослин на основі згорткових нейронних мереж.

Височанський М.Я. Кафедра інформаційних технологій – Дубляни, ЛНУВМ, 2025.

Кваліфікаційна робота: 86 с. текст. част., 18 рис., 9 табл., 14 арк. ілюстраційного матеріалу, 55 джерел.

У роботі розглянуто питання автоматизованого виявлення хвороб рослин із використанням методів комп’ютерного зору та глибинного навчання. Обґрунтовано актуальність створення інформаційної системи, здатної забезпечити точну та оперативну діагностику захворювань у польових умовах. Проведено аналіз сучасних підходів і технологій, зокрема застосування згорткових нейронних мереж для класифікації візуальних симптомів. Розроблено архітектуру інформаційної системи, що включає модулі збору, попередньої обробки, аналізу зображень, управління даними та формування рекомендацій. Створено прототип на базі фреймворків TensorFlow та Streamlit, який забезпечує повний цикл роботи – від завантаження зображення до отримання діагнозу і формування звіту. Проведено тестування моделі EfficientNetB0, яке підтвердило її ефективність (AUC = 0.823, Accuracy = 0.79) і можливість подальшого удосконалення через розширення навчальної вибірки та інтеграцію спектральних даних.

*Ключові слова:* інформаційна система, хвороби рослин, згорткові нейронні мережі, комп’ютерний зір, машинне навчання, Streamlit, TensorFlow.

# ЗМІСТ

# ВСТУП

Сучасний розвиток аграрного сектору вимагає впровадження інноваційних інформаційних технологій, здатних забезпечити підвищення ефективності виробництва та мінімізацію втрат врожаю [53]. Однією з найгостріших проблем у сільському господарстві є своєчасне виявлення та діагностика хвороб рослин. Поширення захворювань призводить до значних економічних збитків, погіршення якості продукції та необхідності використання великої кількості хімічних засобів захисту, що негативно впливає на довкілля. Традиційні методи діагностики базуються на візуальних спостереженнях агрономів або лабораторних дослідженнях, проте вони є трудомісткими, потребують значного часу та часто не дозволяють вчасно реагувати на появу небезпечних симптомів.

Наукові дослідження останніх років підтверджують ефективність застосування штучного інтелекту та машинного навчання для вирішення цієї проблеми [1]. Особливе місце посідають методи комп’ютерного зору, які дозволяють здійснювати аналіз зображень рослин та визначати ознаки захворювань з високою точністю. Прикладом такого підходу є розробка прототипів «розумного фермерства», у яких об’єднано сенсорні технології, алгоритми обробки даних та моделі глибинного навчання. Ці системи здатні забезпечити автоматизований моніторинг стану посівів і швидке прийняття рішень щодо необхідності втручання.

*Об’єктом дослідження* у даній кваліфікаційній роботі є процеси виявлення захворювань рослин на основі аналізу їхніх візуальних характеристик.

*Предметом дослідження* виступають методи та алгоритми обробки зображень, а також інструменти машинного навчання, зокрема згорткові нейронні мережі, які застосовуються для ідентифікації хвороб.

*Засобами дослідження* є сучасні програмні та технічні ресурси: мова програмування Python, бібліотеки TensorFlow і OpenCV, графічні процесори для навчання моделей, а також набір даних із зображеннями здорових і уражених рослин.

*Метою роботи* є розробка інформаційної системи, здатної виявляти хвороби рослин на основі аналізу зображень із використанням методів глибинного навчання.

Для досягнення цієї мети необхідно здійснити аналіз наукових публікацій і технологій, сформувати відповідну базу даних, провести підготовку зображень, розробити та навчити модель нейронної мережі, оцінити її точність і створити прототип системи для практичного застосування.

*Наукова новизна* полягає у використанні сучасних підходів до побудови моделей виявлення захворювань із урахуванням локальної специфіки вирощуваних культур і особливостей поширення хвороб. Практичне значення роботи визначається можливістю застосування розробленої системи у фермерських господарствах для швидкої діагностики, що дозволить скоротити витрати на засоби захисту, знизити ризики втрати врожаю та забезпечити більш раціональне використання ресурсів.

Таким чином, запропоноване дослідження спрямоване на поєднання теоретичних знань у галузі інформаційних технологій та практичних потреб аграрного виробництва, що робить тему кваліфікаційної роботи актуальною та важливою для подальшого розвитку інтелектуальних систем у сільському господарстві.

# Розділ 1.
# АНАЛІЗ СТАНУ АВТОМАТИЗОВАНОГО ВИЯВЛЕННЯ ХВОРОБ РОСЛИН

## 1.1. Актуальність проблеми та огляд сучасних підходів

Сучасне сільське господарство перебуває у стані трансформації, що зумовлена як економічними, так і технологічними чинниками. Однією з найвагоміших проблем, яка потребує вирішення, є своєчасне виявлення захворювань рослин. За даними Продовольчої та сільськогосподарської організації ООН, втрати врожаю від хвороб у середньому становлять 15…25 %, а в умовах епідемічних спалахів можуть сягати й 40% [41]. Це призводить не лише до зниження продуктивності аграрних підприємств, але й до зростання собівартості продукції та втрати конкурентоспроможності на ринку. У зв’язку з цим своєчасна діагностика стає ключовим завданням, від ефективності якого залежить сталий розвиток агросфери.

Традиційні методи боротьби з хворобами базуються на візуальному спостереженні агрономів та подальших лабораторних аналізах. Хоча такий підхід залишається важливим, він має суттєві обмеження, оскільки потребує значних затрат часу, людських ресурсів та фінансів. Крім того, точність діагностики у польових умовах часто залежить від досвіду конкретного спеціаліста, що ускладнює стандартизацію процесу. У результаті аграрії змушені або запізно реагувати на поширення захворювань, або ж використовувати надмірну кількість пестицидів, що негативно впливає на екологію та здоров’я людей [35].

З появою цифрових технологій відбувається зсув від класичних методів до автоматизованих систем, які базуються на обробці даних і застосуванні алгоритмів машинного навчання. У наукових дослідженнях все більшого поширення набуває використання комп’ютерного зору. Його сутність полягає в

аналізі зображень рослин та виділенні характерних ознак, що свідчать про наявність патологій (рис. 1.1).

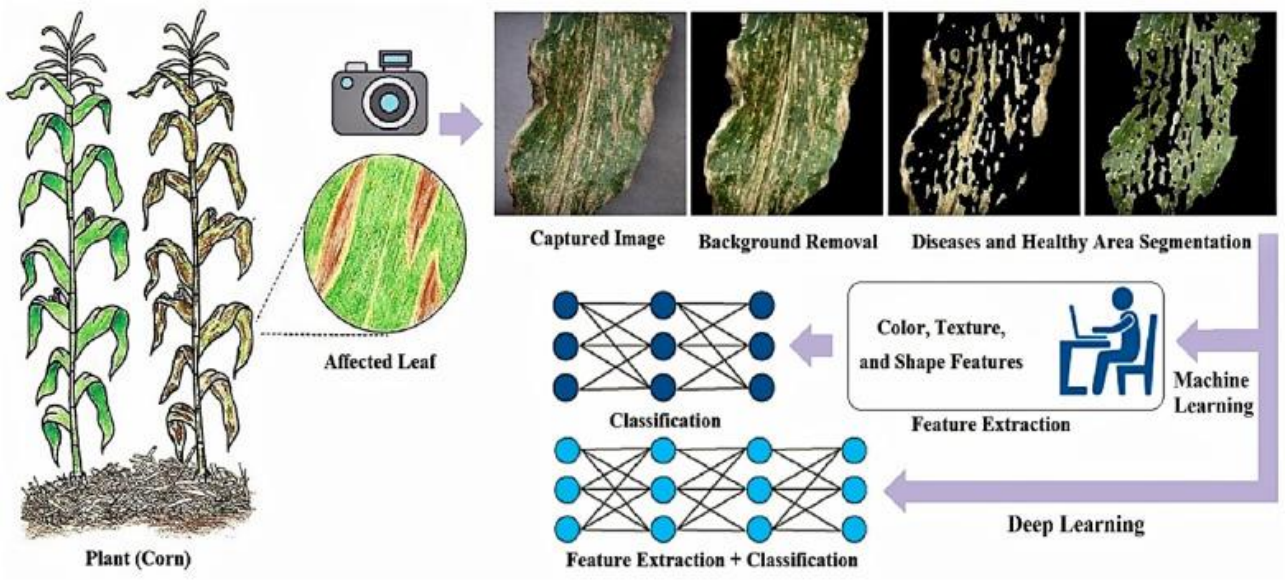


Рисунок 1.1 – Етапи діагностики та класифікації хвороб листя рослин [29]

Точність є основним критерієм, який дослідники використовують для розрахунку продуктивності моделі класифікації [22]. Вилучення ознак, яке є важливим кроком в автоматичній ідентифікації хвороб рослин, може підвищити точність класифікатора та призвести до більш точної діагностики хвороб [38]. Інженерія ознак – це процес ідентифікації та вибору найбільш релевантних змінних із необроблених даних для розробки надійної прогностичної моделі. На рис. 1.1 показано різні етапи діагностики та класифікації хвороб листя рослин. Як показано на рисунку, вилучення ознак є важливою частиною такої діагностики. Це дослідження має на меті розглянути ознаки, які зазвичай вилучалися з листя рослин у нещодавніх дослідженнях діагностики хвороб рослин. Крім того, оскільки для надійного вилучення ознак зображення необхідна ефективна сегментація захворювань на зображеннях, у цьому дослідженні також розглядаються та обговорюються доступні методи сегментації.

Важливо відзначити, що такі системи дозволяють не лише підвищити точність діагностики, але й значно скоротити час між виявленням симптомів і прийняттям управлінських рішень.

Для ілюстрації важливості питання наведемо порівняльні дані у таблиці 1.1.

Таблиця 1.1 – Аналіз підходів до виявлення хвороб рослин

| Підхід | Час на діагностику | Середня точність | Витрати ресурсів | Екологічний вплив |
|---|---|---|---|---|
| Візуальний огляд агронома | 1–2 дні | 65–75 % | Високі | Значний |
| Лабораторний аналіз | 3–7 днів | 85–95 % | Дуже високі | Середній |
| Комп’ютерний зір + нейронні мережі | хвилини | 90–97 % | Помірні | Низький |

Як видно з таблиці 1.1, новітні інформаційні системи поєднують високу точність з відносно невеликими витратами ресурсів, що робить їх більш привабливими для впровадження у фермерських господарствах.

Підґрунтям більшості сучасних розробок є використання згорткових нейронних мереж. Архітектура CNN спеціально пристосована для роботи із зображеннями завдяки здатності виявляти просторові ознаки. Формально роботу такої мережі можна описати через операцію згортки:

$$S(i,j)=\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} I(i+m,j+n)\cdot K(m,n), \qquad (1.1)$$

де $S(i,j)$ – вихідне значення після згортки; $I(i+m,j+n)$ – значення пікселя вхідного зображення; $K(m,n)$ – ядро згортки; $M$ і $N$ – розміри ядра.

Завдяки операції (1.1) мережа може виявляти характерні ознаки хвороб: зміни кольору, форми, текстури листя чи плодів.

У статті [32] авторами запропоновано інтегровану систему, що поєднує камери, алгоритми попередньої обробки даних та глибинне навчання для

діагностики захворювань у режимі, наближеному до реального часу. Авторами показано, що навіть за використання відносно простих апаратних засобів можна досягти високого рівня точності у виявленні поширених хвороб. Особливої уваги заслуговує практичне спрямування цієї розробки, адже вона орієнтована на використання у фермерських господарствах невеликого масштабу, що є типовим для України.

Важливим кроком для подальших досліджень є формування локальних баз даних. Адже якість навчання моделей прямо залежить від репрезентативності зображень. Для прикладу, у випадку з фітофторозом томатів система повинна навчатися на зображеннях різних стадій розвитку захворювання, щоб розпізнавати його ще на ранніх етапах. Для цього застосовують аугментацію даних, яка включає обертання, масштабування та зміну кольорової гами зображень.

Додатково до цього, сучасні підходи передбачають оцінку точності моделей на основі метрик. Найбільш поширеними є точність (Accuracy), повнота (Recall), точність прогнозу (Precision) та F1-міра. Остання розраховується за формулою:

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}, \qquad (1.2)$$

Використання цих показників дозволяє порівнювати різні архітектури моделей і вибирати найефективнішу для практичного застосування.

У напрямі екологічної безпеки також слід відзначити, що системи автоматизованого виявлення хвороб сприяють зниженню використання пестицидів. Раннє виявлення уражень дозволяє проводити локалізовану обробку лише тих ділянок, де є ознаки хвороби, замість суцільного обприскування. Це, у свою чергу, зменшує хімічне навантаження на ґрунти та водойми, підвищує якість продукції та відповідає принципам сталого розвитку.

На рисунку 1.2 нижче схематично подано загальну структуру системи «розумного фермерства», яка використовується для виявлення хвороб рослин.

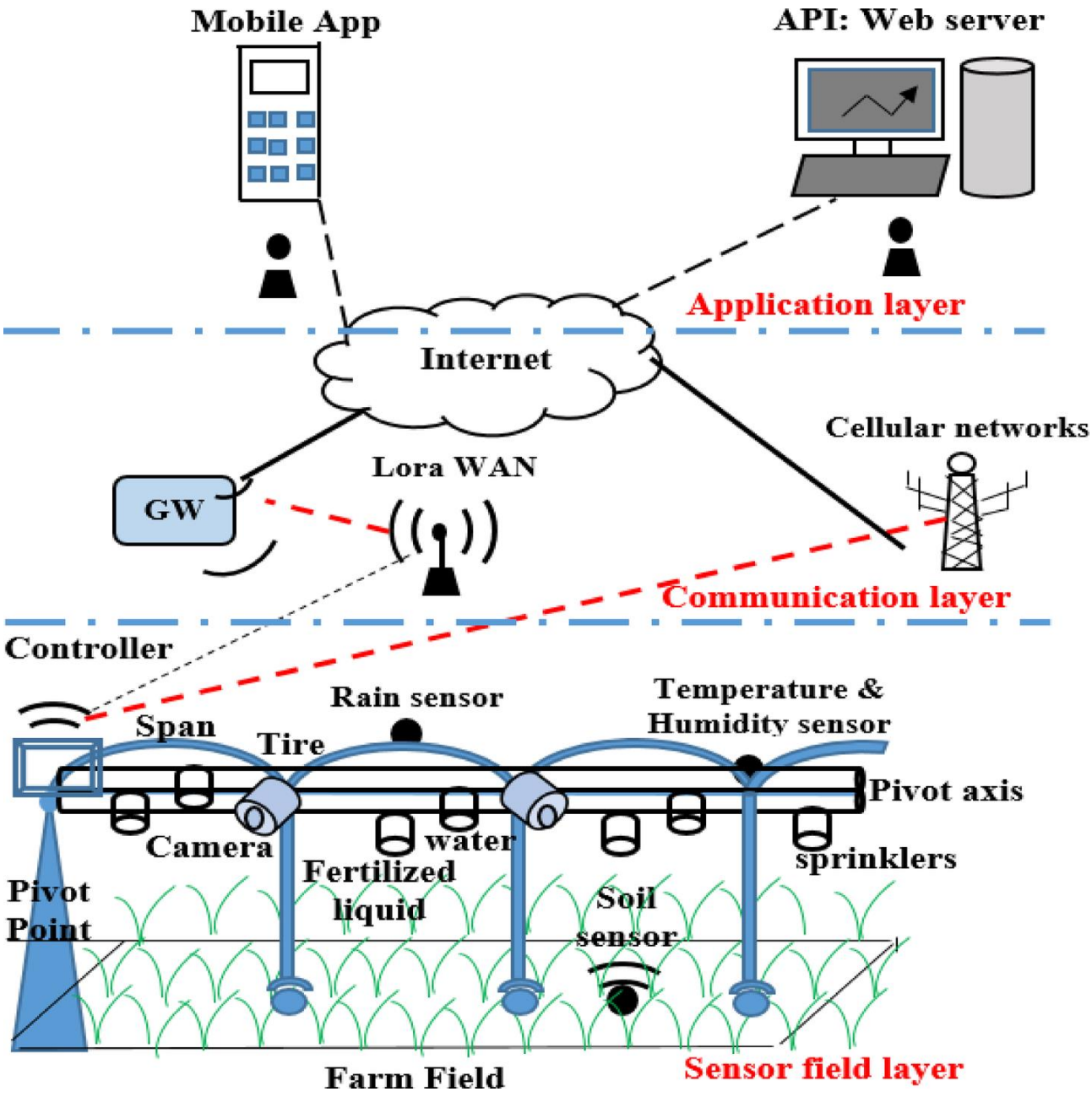


Рисунок 1.2 – Загальна структура системи «розумного фермерства», яка використовується для виявлення хвороб рослин [26]

Ця структура демонструє логіку роботи більшості сучасних прототипів: від збору даних до отримання результатів, придатних для прийняття управлінських рішень.

Таким чином, актуальність проблеми своєчасного виявлення хвороб рослин підтверджується як економічними, так і екологічними факторами. Сучасні підходи, зокрема комп'ютерний зір і методи глибинного навчання, відкривають нові можливості для розробки ефективних інформаційних систем.

Результати досліджень, таких як [32], демонструють перспективність впровадження прототипів «розумного фермерства» у практику, що є важливою передумовою для підвищення стійкості аграрного виробництва в умовах зростання глобальних викликів.

## 1.2. Системи комп’ютерного зору та машинного навчання в аграрному секторі

Одним із найбільш динамічних напрямів у сучасному сільському господарстві є інтеграція технологій комп’ютерного зору з алгоритмами машинного навчання. Якщо в попередньому підпункті було окреслено загальну актуальність і проблематику раннього виявлення хвороб, то зараз доцільно зосередитися на конкретних принципах роботи цих систем та особливостях їх застосування у практиці. Важливо наголосити, що ці технології вже не сприймаються як експериментальні, а стають елементом прикладних рішень, які використовуються у промислових теплицях, фермерських господарствах і навіть у домашньому городництві [36].

Основна ідея комп’ютерного зору в аграрному секторі полягає у перетворенні візуальної інформації на цифрові ознаки, які можуть бути проаналізовані математично. Кожне зображення рослини представляється як матриця пікселів, значення яких відображає інтенсивність у певному спектрі. Далі за допомогою операцій фільтрації, сегментації та класифікації формується набір характеристик. Одним із прикладів є виявлення плям на листках за допомогою колірних індексів. Формально це можна описати через вегетаційний індекс Excess Green (ExG):

$$ExG = 2 \cdot G - (R + B), \tag{1.3}$$

де $R, G, B$ – інтенсивності червоного, зеленого та синього каналів відповідно.

Використання такого індексу дозволяє виділяти ділянки листя із порушеним пігментним балансом, що часто є першою ознакою захворювання [15].

Розвиток машинного навчання додав до цих методів нові можливості. Якщо класичні алгоритми вимагали ручного проєктування ознак, то сучасні глибинні нейронні мережі самостійно навчаються знаходити складні патерни у даних. Це особливо цінно для багатофакторних процесів у рослинництві, де симптоми хвороб можуть бути нечіткими та варіювати залежно від сорту, клімату чи фази розвитку культури. Дослідження Javidan та співавторів (2024) показало, що методи feature engineering у поєднанні з нейронними мережами дозволяють досягати точності понад 95 % у виявленні поширених захворювань, таких як борошниста роса чи септоріоз [30].

Таблиця 1.2 – Порівняння алгоритмів машинного навчання у діагностиці хвороб рослин

| Алгоритм | Тип даних | Середня точність | Особливості використання | Джерело |
|---|---|---|---|---|
| CNN (згорткова нейронна мережа) | RGB-зображення листя | 92…97% | Автоматичне виділення ознак, висока узагальненість | [30] |
| SVM (метод опорних векторів) | Витягнуті текстурні ознаки | 85…90% | Вимагає ретельної попередньої обробки | [54] |
| Random Forest | Комбінація спектральних і морфологічних параметрів | 80…88% | Стійкість до шумів, але обмежена узагальненість | [18] |
| ResNet, EfficientNet | Великі набори зображень | 95…98% | Потребують великих обчислювальних ресурсів | [25] |

Для наочності наведемо узагальнені приклади застосування різних підходів машинного навчання у діагностиці рослин (табл. 1.2).

Як видно з таблиці 1.2, різні підходи мають свої переваги й обмеження. Згорткові нейронні мережі сьогодні фактично стали «золотим стандартом», але інші методи все ще зберігають значення для задач із малими наборами даних чи специфічними умовами.

Важливим трендом є інтеграція систем комп'ютерного зору з технологіями Інтернету речей. Робота [31] стала одним із перших прикладів, де діагностика та лікування рослин у теплицях здійснювалися автоматично за допомогою IoT-пристроїв. У подібних системах камери, розташовані у різних частинах теплиці, безперервно надсилають дані до серверів, де алгоритми машинного навчання здійснюють аналіз і відправляють рекомендації агроному. Це створює умови для так званого «цифрового фермерства», де управлінські рішення приймаються на основі даних, а не інтуїції.

Суттєве значення мають також багатоспектральні методи аналізу. Використання спектральних сенсорів дозволяє оцінювати фізіологічний стан рослин, зокрема вміст хлорофілу чи води в тканинах. Gold та співавтори (2020) показали, що поєднання спектроскопії з машинним навчанням дає змогу диференціювати стійкі та сприйнятливі сорти картоплі до фітофторозу [23]. У цьому випадку система не просто діагностує наявність хвороби, а й аналізує фізіологічні відмінності, що відкриває нові перспективи у селекційній роботі.

Розглянемо також приклад формального опису задачі класифікації у термінах машинного навчання. Нехай маємо набір зображень $X = \{x_1, x_2, \ldots, x_n\}$ та відповідні мітки класів $Y = \{y_1, y_2, \ldots, y_n\}$, де $y_i \in \{0,1\}$ означає здорову чи уражену рослину. Метою є побудова функції класифікації $f : X \to Y$, яка мінімізує функцію втрат:

$$L = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)\right], \qquad (1.4)$$

де $\hat{y}_i$ – передбачена ймовірність віднесення прикладу до класу «хворий».

Використання такої крос-ентропійної функції дозволяє ефективно навчати глибинні моделі для задач класифікації [24].

Сучасні системи дедалі частіше поєднують кілька різних джерел даних. Ibrahim та колеги (2025) описали прототип, у якому об’єднані дані IoT-сенсорів про температуру, вологість та освітленість із результатами аналізу зображень рослин [27]. Така мультиагентна архітектура забезпечує не лише діагностику, але й рекомендації щодо профілактичних дій, що наближає систему до формату «цифрового агронома».

На рисунку 1.3 представлено узагальнену архітектуру подібної системи.

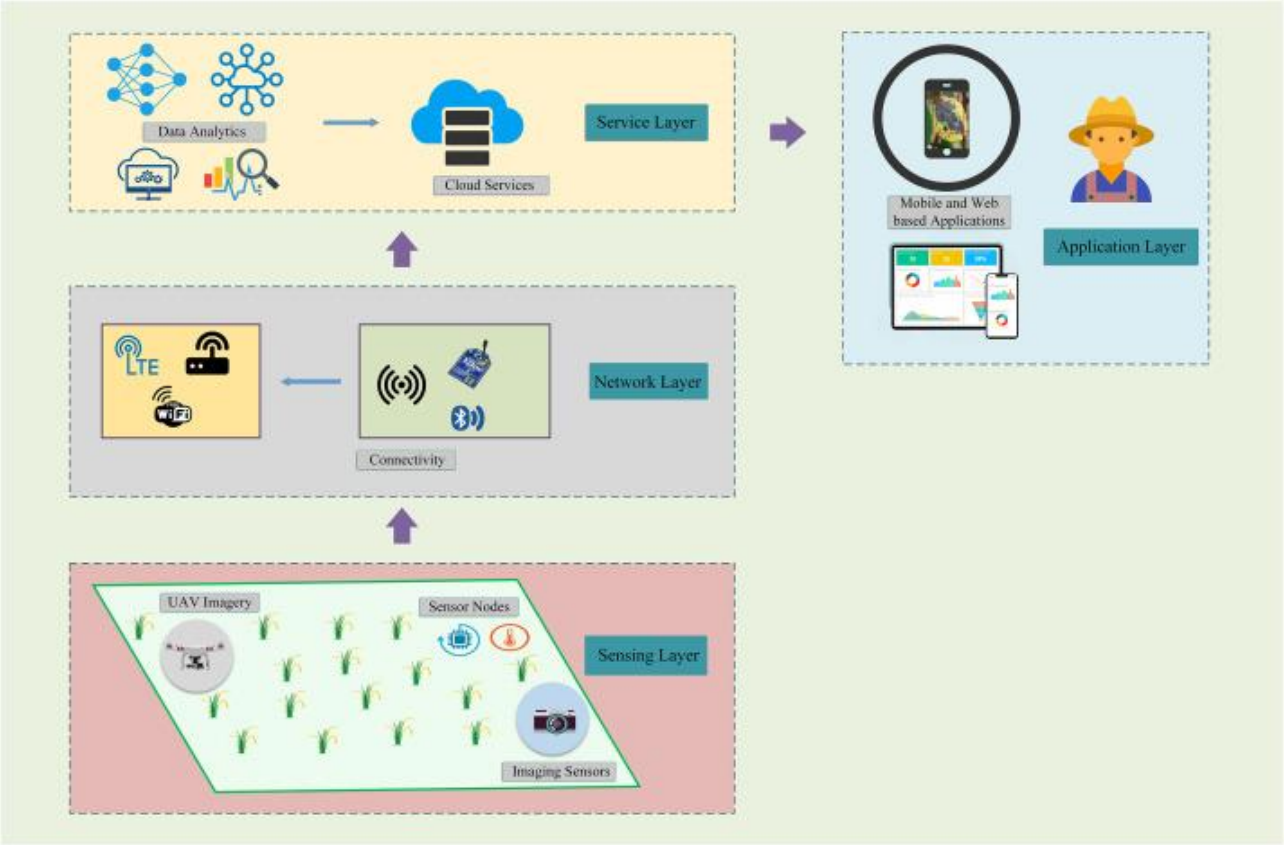


Рисунок 1.3 – Архітектура системи комп’ютерного зору та машинного навчання у рослинництві [20]

Інтегративні та багаторежимні моделі штучного інтелекту (ШІ) можуть бути розгорнуті для прогнозування поведінки врожаю за різних польових умов [45]. Врожайність основних сільськогосподарських культур у різних регіонах, а також польові умови для виробництва сільськогосподарських культур, вплив на навколишнє середовище та економічні результати, були оцінені за допомогою

алгоритмів глибокого та машинного навчання [46]. Глибоке навчання дозволяє обчислювальним моделям з кількома шарами обробки відображати дані на кількох рівнях абстракції [43]. Основним застосуванням глибокого навчання в галузі сільського господарства є побудова моделей для отримання змістовних висновків з сільськогосподарських даних [28], аналіз зображень, включаючи класифікацію та виявлення об'єктів, таких як виявлення хвороб, ідентифікація бур'янів, аналіз ґрунту, виявлення хвороб рослин тощо [33].

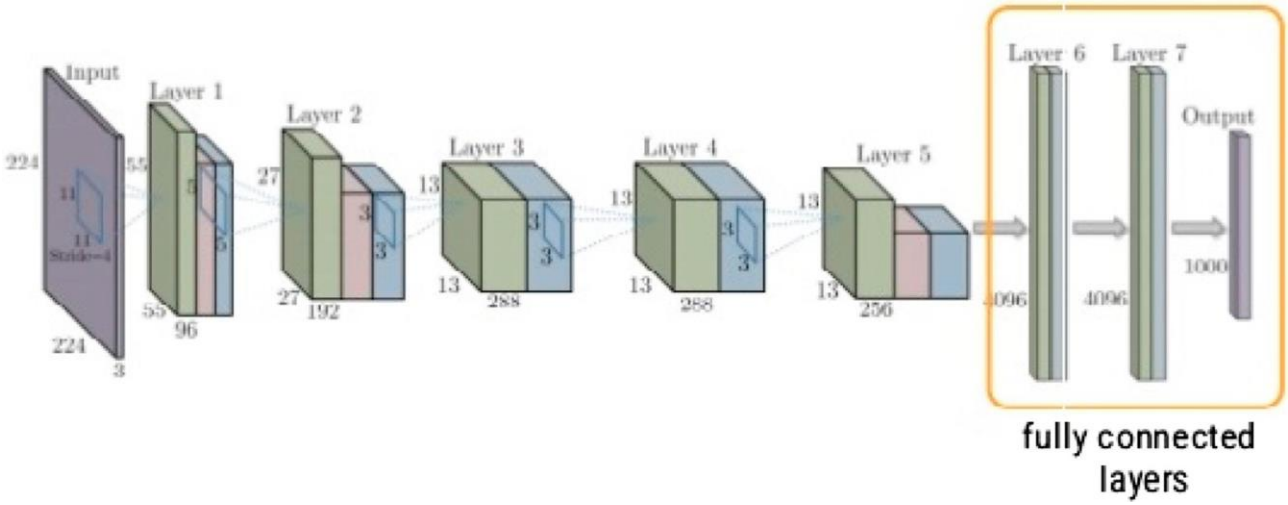


Рисунок 1.4 – Приклад архітектури DL, що ілюструє CaffeNet [33]

Згорткові нейронні мережі (CNN) являють собою клас глибоких ШНМ прямого зв'язку , і вони згадуються в багатьох з розглянутих робіт як використана техніка (17 робіт, 42%). Як показано на рисунку, на деяких шарах мережі виконуються різні згортки, створюючи різні представлення навчального набору даних, починаючи з більш загальних на перших більших шарах, стаючи більш конкретними на глибших шарах.

Згорткові шари діють як екстрактори ознак з вхідних зображень, розмірність яких потім зменшується шарами об'єднання. Згорткові шари кодують кілька ознак нижчого рівня в більш розрізнювальні ознаки таким чином, що це просторово враховує контекст. Їх можна розуміти як банки фільтрів, які перетворюють вхідне зображення на інше, виділяючи певні закономірності. Повністю зв'язані шари, розміщені в багатьох випадках поблизу виходу моделі,

діють як класифікатори, що використовують вивчені ознаки високого рівня для класифікації вхідних зображень у заздалегідь визначених класах або для створення числових прогнозів. Вони приймають вектор як вхідні дані та створюють інший вектор як вихідні.

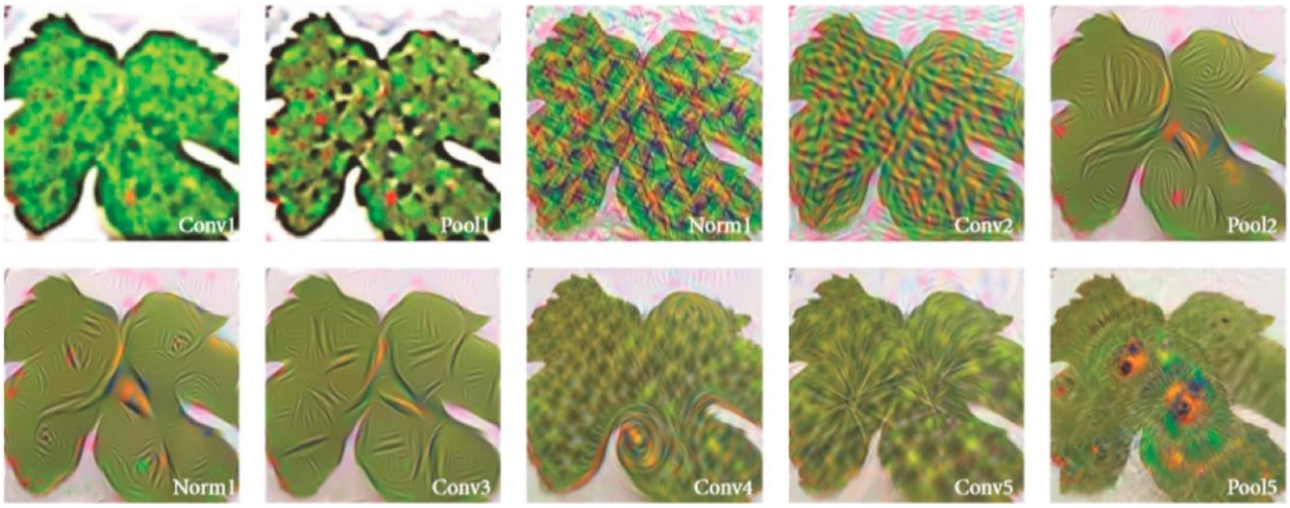


Рисунок 1.5 – Візуалізація зображень вихідних шарів після кожного етапу обробки CaffeNet CNN (тобто згортки, об'єднання, нормалізації) у задачі ідентифікації хвороб рослин на основі зображень листя [33]

Приклад візуалізації зображень листя після кожного кроку обробки CaffeNet CNN у задачі ідентифікації хвороб рослин зображено на рис. 1.5. Ми можемо спостерігати, що після кожного кроку обробки конкретні елементи зображення, які вказують на ознаки хвороби, стають більш помітними, особливо на останньому кроці (Pool5).

Узагальнюючи, можна сказати, що системи комп’ютерного зору та машинного навчання вже довели свою практичну цінність для аграрного сектору. Вони дають змогу підвищити точність діагностики, скоротити час на прийняття рішень та зменшити залежність від людського фактору. Водночас залишаються виклики, пов’язані з необхідністю створення локальних баз даних, адаптацією моделей до різних кліматичних умов та забезпеченням доступності технологій для малих фермерів. Однак швидкий розвиток апаратного забезпечення, відкритих бібліотек і платформ машинного навчання свідчить про

те, що найближчим часом ці системи стануть невід'ємною частиною розумного фермерства [30; 27].

## 1.3. Огляд існуючих прототипів та систем, приклади успішних рішень

Розвиток технологій штучного інтелекту та сенсорних пристроїв упродовж останніх років зумовив появу широкого спектра прототипів і практичних систем для діагностики хвороб рослин. Ці рішення відрізняються за масштабом, технічними засобами та архітектурою, однак усі вони спрямовані на досягнення спільної мети – створення ефективного інструментарію для фермерів та дослідників. У цьому підпункті розглянемо найбільш показові приклади реалізації прототипів «розумного фермерства» та систем підтримки прийняття рішень.

Одним із перших відомих прикладів є розробка IET, представлена на конференції BICET у 2018 році. У цьому прототипі використовувалася концепція «розумної теплиці», де камери та IoT-пристрої були інтегровані у єдину мережу. Алгоритми обробки зображень у реальному часі аналізували стан рослин, після чого на основі результатів здійснювалася автоматизована подача засобів захисту [32]. Цей приклад став підтвердженням того, що навіть у країнах із відносно обмеженими ресурсами можна створювати інноваційні рішення з високим рівнем автоматизації.

Інший успішний напрямок розвивається у сфері поєднання гіперспектральної зйомки та алгоритмів машинного навчання. Дослідження Gold та колег (2020) продемонструвало, що спектроскопічний аналіз у поєднанні з методами машинного навчання дозволяє виявляти різницю між сортами картоплі, стійкими й чутливими до фітофторозу [23]. У цьому випадку технологія виходить за межі простої діагностики й переходить у сферу селекції,

адже дає змогу швидко оцінювати потенціал сортів у лабораторних і польових умовах.

Цікавим прикладом є системи для контролю бур'янів. У роботі Mathanker та колег (2010) використано алгоритми AdaBoost та SVM для класифікації рослин пшениці й каноли [38]. Система працювала у режимі реального часу на полі й могла розрізняти культурні рослини та бур'яни з достатньою точністю, що відкрило можливість для створення роботизованих обприскувачів, які діють вибірково. Такий підхід сприяє зменшенню використання гербіцидів та зниженню негативного впливу на довкілля.

У 2022 році Dhanya та співавтори представили огляд практичних застосувань глибинного навчання у сільському господарстві [20]. Серед наведених прикладів можна виділити прототипи на основі ResNet і EfficientNet, які були адаптовані для виявлення кількох хвороб томатів та винограду. У цих системах було досягнуто точності понад 95 %, що демонструє ефективність сучасних архітектур глибинного навчання. Водночас автори наголошують на потребі формування великих і якісних баз даних, без яких навіть найскладніші моделі не здатні працювати коректно.

Не менш важливою є інтеграція прототипів з іншими цифровими технологіями. Ibrahim та співавтори (2025) представили AI-IoT платформу для виявлення та лікування хвороб у польових умовах [27]. Архітектура цієї системи передбачала збір даних із сенсорів вологості, температури та освітленості разом із зображеннями, які оброблялися за допомогою CNN. Завдяки цьому система могла не лише діагностувати захворювання, а й пропонувати алгоритми їх лікування. У математичній формі роботу системи можна описати як функцію оптимізації:

$$R = arg\,min_{a \in A}\, C(a \mid d), \tag{1.5}$$

де $R$ – рекомендоване рішення; $A$ – множина можливих дій (наприклад, обприскування, зміна умов мікроклімату); $C(a \mid d)$ – функція витрат для дії $a$ з урахуванням даних $d$.

Таким чином, система не обмежувалася пасивною діагностикою, а формувала рекомендації для агронома з мінімізацією витрат.

Важливим аспектом є поява комерційних рішень, орієнтованих на масове впровадження. Так, у США компанії вже пропонують мобільні додатки, які використовують камери смартфонів для діагностики захворювань. Програми працюють за принципом crowdsourcing: користувачі завантажують зображення, які використовуються для подальшого донавчання моделей. Це створює ефект самооновлення системи та підвищує її точність у реальних умовах [36].

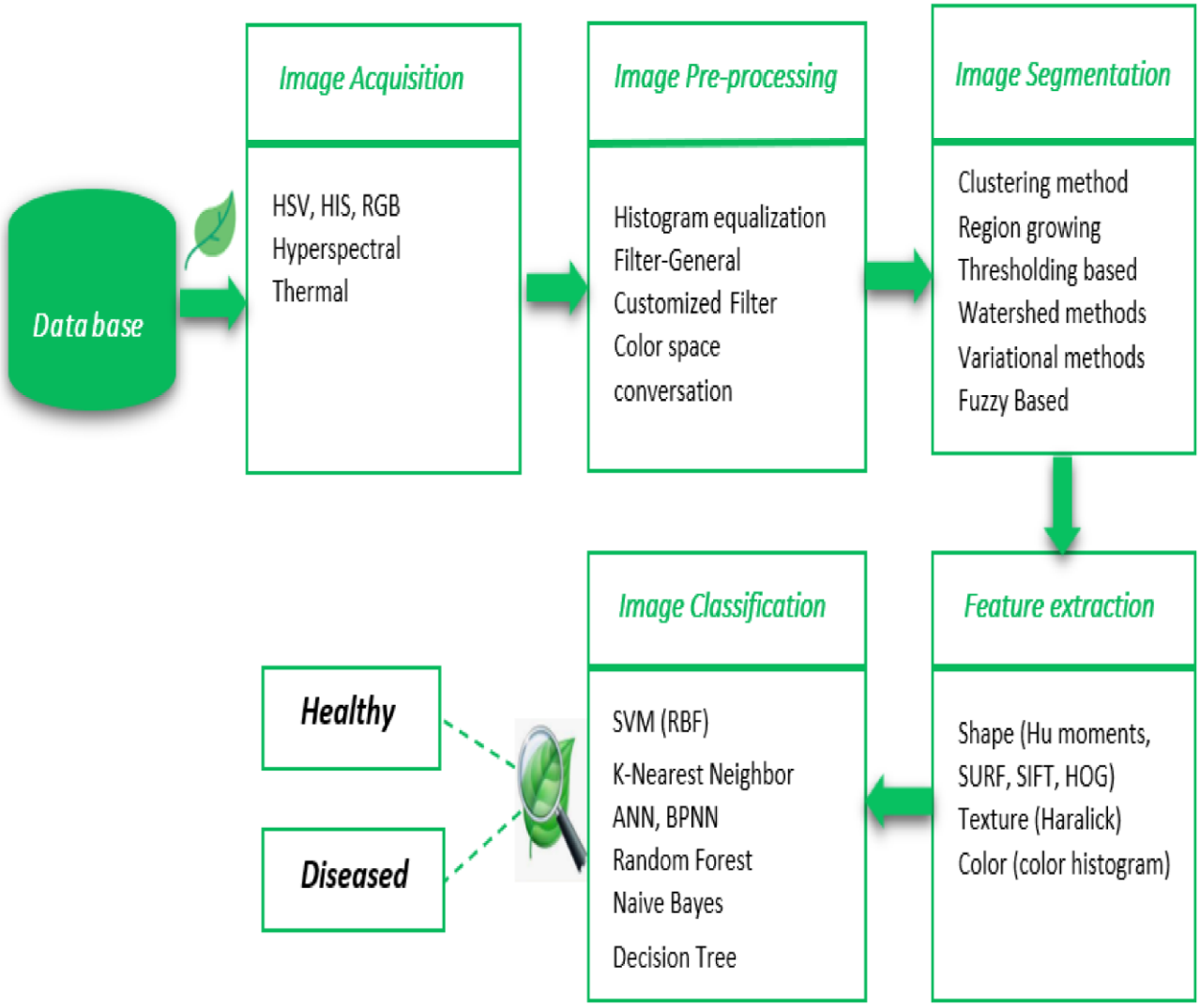


Рисунок 1.6 – Різні підходи до ідентифікації хвороб листя [39]

Листяні зображення є чудовим та багатим джерелом даних про патологію та морфологічну поведінку рослин; таким чином, ці дані необхідно ретельно

видобувати та аналізувати. Обробка зображень відіграє вирішальну роль у діагностиці та аналізі хвороб листя. Процедура, що застосовується в цьому процесі ідентифікації хвороб листя, проілюстрована на рисунку 1.6, де показано різні методи, що використовуються авторами для виявлення хвороби за допомогою обробки зображень та штучного інтелекту.

Мета класифікатора – розпізнавати зображення, сортуючи їх за кількома попередньо визначеними класами на основі результуючого вектора ознак, отриманого на четвертому кроці. Для цього завдання класифікації містить дві фази, а саме: навчання та тестування. Операція навчання навчає класифікатор на навчальному наборі даних; таким чином, чим більша кількість навчальних наборів, тим краща отримана точність. Слід зазначити, що результат, який є здоровим або хворобливим станом культури, пов'язаним з назвою виду, має бути досягнутий якомога швидше.

Для кращого узагальнення наведемо у таблиці приклади успішних рішень із різних країн (табл. 1.3).

Таблиця 1.3 – Приклади прототипів та систем для виявлення хвороб рослин

| Система / Прототип | Культура | Технологія | Точність | Країна / Автори |
|---|---|---|---|---|
| Smart farm (BICET 2018) | Тепличні культури | IoT + комп’ютерний зір | 90–94 % | Brunei [32] |
| Spectroscopy + ML | Картопля | Спектральний аналіз + Random Forest | >90 % | США [22] |
| Weed classifier | Пшениця | AdaBoost + SVM | 85–88 % | США [38] |
| Tomato disease CNN | Томат | ResNet, EfficientNet | 95–97 % | Індія [20] |
| AI-IoT pivot | Польові культури | CNN + IoT-сенсори | 92–96 % | Саудівська Аравія [27] |
| Mobile apps | Різні | CNN + crowdsourcing | ~90 % | США, ЄС [36] |

Таким чином, огляд існуючих прототипів показує різноманітність підходів до вирішення проблеми. Одні системи орієнтовані на автоматизацію тепличного виробництва, інші – на роботу безпосередньо в полі чи навіть на рівні мобільних додатків для фермерів. Спільним для всіх прикладів є те, що вони засвідчують практичну доцільність інтеграції комп’ютерного зору та машинного навчання у реальні умови агровиробництва. Подальший розвиток цих систем залежить від здешевлення сенсорів, удосконалення алгоритмів і створення доступних платформ, які можна адаптувати до локальних потреб.

## 1.4. Основні компоненти інформаційної системи для діагностики хвороб рослин

Розробка ефективної інформаційної системи для діагностики хвороб рослин потребує чіткої структуризації її складових. Така система зазвичай інтегрує кілька рівнів – від збору первинних даних до формування рекомендацій для користувачів. Її архітектура будується за принципом багаторівневої моделі, у якій кожен елемент виконує спеціалізовану функцію, але при цьому взаємодіє з іншими. У цьому підпункті окреслимо ключові компоненти системи, їхню логіку та практичні приклади застосування у сільському господарстві.

Першим і найбільш очевидним елементом є блок збору даних. Його основу складають пристрої, здатні фіксувати інформацію про стан рослин у різних діапазонах. Найпростішим варіантом є RGB-камери, які забезпечують базовий набір зображень. Проте для підвищення якості діагностики дедалі ширше застосовуються мультиспектральні та гіперспектральні сенсори. Вони дозволяють оцінювати фізіологічний стан рослин на рівні, недоступному для людського ока. Наприклад, за допомогою індексу NDVI (Normalized Difference Vegetation Index) можна визначити активність фотосинтезу:

$$NDVI = \frac{NIR - RED}{NIR + RED}, \tag{1.5}$$

де $NIR$ – відбиття у ближньому інфрачервоному діапазоні; $RED$ – відбиття у червоному спектрі.

Значення цього індексу широко використовується для виявлення стресових станів рослин, у тому числі викликаних захворюваннями [42].

Другим важливим компонентом виступає блок попередньої обробки даних. Тут здійснюється очищення, нормалізація та аугментація зображень. Цей етап має вирішальне значення, адже дані з польових умов часто містять шуми: тіні, різні рівні освітленості чи дефекти камер. Використання алгоритмів згладжування, перетворення кольорових просторів і фільтрації дозволяє зробити набір зображень придатним для аналізу. У сучасних системах активно застосовуються методи розширення даних, які збільшують обсяг тренувальної вибірки шляхом обертання, масштабування чи віддзеркалення зображень. Це знижує ризик переобучення моделей і підвищує їхню стійкість [44].

Центральним елементом є аналітичний блок, що реалізує алгоритми машинного навчання. Сюди входять згорткові нейронні мережі, рекурентні моделі та їхні модифікації. Вибір архітектури залежить від типу даних і поставленої задачі. Так, CNN забезпечують ефективну класифікацію зображень, тоді як моделі типу Vision Transformer (ViT) демонструють ще вищу точність на великих наборах даних [21]. Важливим є й поєднання кількох алгоритмів у гібридні ансамблі, що дає змогу компенсувати недоліки окремих моделей. Формально якість роботи аналітичного блоку оцінюється за метрикою F1-міри:

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}, \qquad (1.6)$$

де $Precision$ – визначає частку правильно класифікованих позитивних випадків серед усіх прогнозованих; $Recall$ – частку знайдених позитивних випадків серед усіх наявних.

Така метрика забезпечує баланс між точністю та повнотою діагностики [40].

Ще одним важливим компонентом є база знань і сховище даних. Цей модуль об’єднує результати попередніх досліджень, приклади зображень, ознаки

поширених хвороб та рекомендації з їх лікування. Сучасні системи дедалі частіше інтегрують бази даних у хмарні сервіси, що забезпечує колективний доступ до інформації та постійне оновлення. Наприклад, мобільні додатки для фермерів часто мають функцію завантаження нових зображень, які автоматично збагачують базу й підвищують точність алгоритмів [19].

Не менш значущим є користувацький інтерфейс. Саме він визначає, наскільки система буде зручною у використанні. Інтерфейс може бути реалізований у вигляді мобільного додатка, веб-платформи або панелі моніторингу теплиць. Найефективнішими є ті рішення, які поєднують візуалізацію результатів із простими рекомендаціями для аграріїв. Наприклад, система може не лише повідомити про наявність ураження, але й одразу запропонувати варіанти обробки.

Для створення структури для виявлення та класифікації хвороб рослин необхідно виконати такі процеси: збір даних, навчання моделі та багатокласова категоризація хвороб листя рослин. На рисунку 1.7 наведено огляд запропонованої структури. Елементи хвороб листя можна знайти та ідентифікувати за допомогою методу Yolov4.

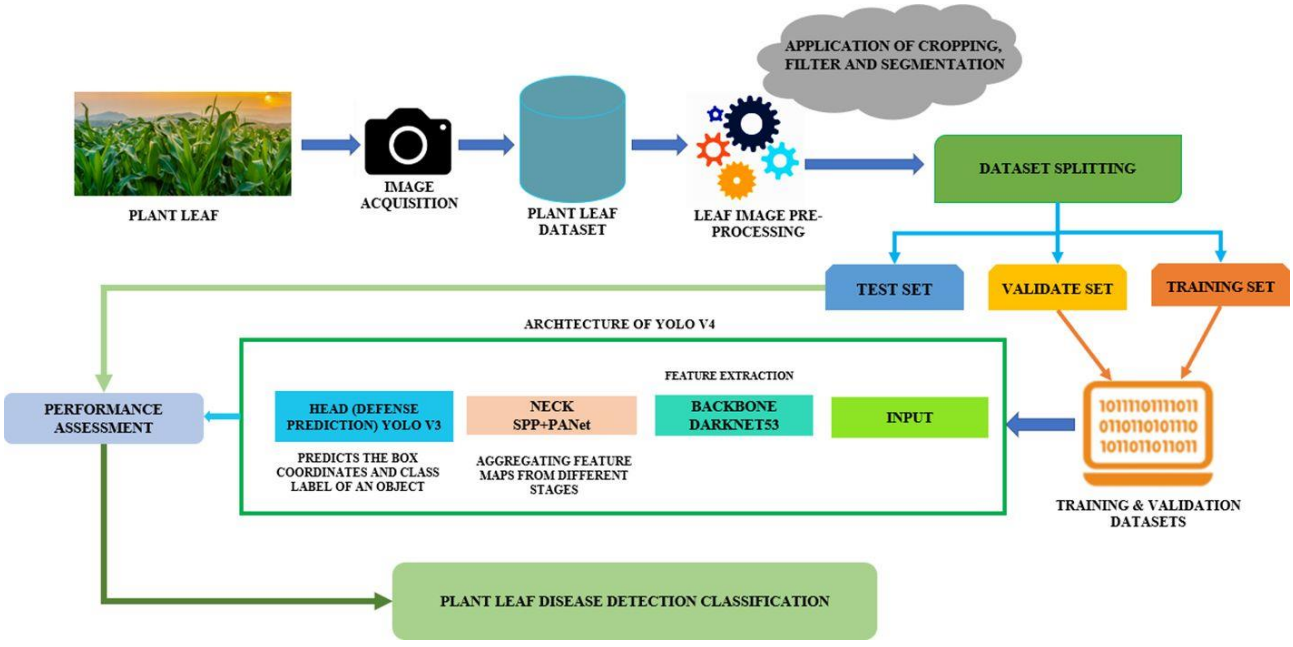


Рисунок 1.7 – Впровадження системи виявлення та ідентифікації хвороб Yolov4 на листках рослин [16]

Для прикладної ілюстрації у табл. 1.4 наведено узагальнені характеристики основних компонентів системи.

Таблиця 1.4 – Основні компоненти інформаційної системи для діагностики хвороб рослин

| Компонент | Основні функції | Технології | Приклади реалізації |
|---|---|---|---|
| Збір даних | Отримання зображень і сенсорних параметрів | RGB, мультиспектральні, гіперспектральні камери | NDVI-моніторинг у теплицях [42] |
| Попередня обробка | Нормалізація, аугментація, шумопригнічення | OpenCV, Keras ImageDataGenerator | Дослідження з обробки томатів [44] |
| Аналітичний блок | Класифікація та прогноз | CNN, ViT, ансамблеві методи | ResNet у виноградарстві [21] |
| База знань | Збереження й оновлення даних | Хмарні сховища, бази зображень | Crowdsourcing у мобільних додатках [19] |
| Інтерфейс | Взаємодія з користувачем | Веб- та мобільні технології | Agro-Apps з рекомендаціями [34] |

Важливо наголосити, що ефективність системи визначається не лише якістю окремих компонентів, а й їх взаємодією. Лише комплексний підхід дозволяє досягти необхідної точності та зручності використання. Перспективним напрямом є розширення функцій інтерфейсу, де користувач зможе не лише отримати діагноз, але й вести електронний журнал обробок, формувати звіти чи інтегрувати дані з іншими агротехнологічними системами.

Підсумовуючи, основні компоненти інформаційної системи для діагностики хвороб рослин утворюють багаторівневу архітектуру, де кожен рівень має своє значення і функцію. Їхнє гармонійне поєднання дозволяє створювати потужні інструменти підтримки прийняття рішень у сільському господарстві, що здатні не лише підвищити ефективність виробництва, але й сприяти сталому розвитку аграрної галузі.

## 1.5. Завдання кваліфікаційної роботи

Виконання кваліфікаційної роботи передбачає формулювання чіткої послідовності завдань, спрямованих на досягнення поставленої мети. Основна ідея полягає у створенні прототипу інформаційної системи, здатної діагностувати хвороби рослин на основі аналізу зображень та, за можливості, даних від додаткових сенсорів. Такий підхід дозволяє поєднати сучасні алгоритми комп’ютерного зору з практичними потребами аграрного виробництва.

Насамперед необхідно провести огляд наукових джерел і прикладів практичних реалізацій подібних систем. Особливу увагу слід приділити прототипи, де показано можливість використання IoT-пристроїв і алгоритмів обробки зображень у тепличних умовах. Аналіз цієї моделі дає змогу зрозуміти, які компоненти варто адаптувати, а які вдосконалити відповідно до місцевих умов.

Подальшим завданням є створення власної бази даних зображень, що включатиме здорові та уражені екземпляри рослин, характерні для регіону дослідження. Це забезпечить можливість навчання та тестування моделей на матеріалі, який відображає реальні особливості культур і поширених хвороб. Паралельно з цим необхідно розробити процедури попередньої обробки зображень, що передбачають нормалізацію кольору, зменшення шумів і розширення вибірки за допомогою методів аугментації.

Наступний етап стосується створення та навчання моделі для класифікації хвороб. Планується випробування кількох архітектур глибинних нейронних мереж із вибором найбільш результативної. Для оцінки ефективності моделі мають використовуватися такі показники, як точність, повнота, F1-міра. Це дозволить зробити об’єктивні висновки щодо переваг і недоліків обраного підходу.

Важливо також передбачити інтеграцію даних від сенсорів, що відображають мікрокліматичні умови – температуру, вологість, освітленість. Така інформація може посилювати точність діагностики, адже розвиток багатьох захворювань безпосередньо залежить від цих факторів. Об’єднання візуальних та сенсорних даних у межах однієї системи створює підґрунтя для комплексної оцінки стану рослин.

Особливе місце посідає завдання перевірки роботи прототипу в умовах, наближених до реальних. Це може бути тестування у навчальній лабораторії чи на невеликих ділянках поля. За результатами випробувань потрібно зробити висновки щодо стабільності, швидкості обробки та точності діагностики, а також визначити чинники, що впливають на роботу системи.

Окремим завданням є створення користувацького інтерфейсу, який забезпечить зручність взаємодії з системою. Інтерфейс має дозволяти завантаження зображень, перегляд результатів діагностики та отримання рекомендацій. Простота у використанні є вирішальною умовою для практичного застосування, адже кінцевими користувачами виступатимуть фермери чи агрономи, які не завжди мають спеціалізовані знання в галузі інформаційних технологій.

Таким чином, завдання кваліфікаційної роботи охоплюють усі ключові етапи створення системи – від аналітичної частини та збору даних до розробки, навчання моделі та апробації прототипу.

# РОЗДІЛ 2.
# ВИБІР ІНСТРУМЕНТАРІЮ ТА ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ВИЯВЛЕННЯ ХВОРОБ РОСЛИН НА ОСНОВІ ЗГОРТКОВИХ НЕЙРОННИХ МЕРЕЖ

## 2.1. Аналіз вимог до системи автоматизованої діагностики хвороб рослин

Створення ефективної інформаційної системи виявлення хвороб рослин вимагає ґрунтовного аналізу вимог, які охоплюють функціональні, технічні та експлуатаційні аспекти. На першому етапі визначаються потреби кінцевого користувача – агронома, власника теплиці, фермера чи дослідника. Головною метою такої системи є забезпечення можливості своєчасної і точної діагностики хвороб на основі візуального аналізу листкових пластин рослин. Це передбачає опрацювання як мінімум двох типів даних: зображень високої якості та метаданих (наприклад, назва культури, дата зйомки, стан середовища за потреби).

Функціональні вимоги включають можливість автоматичного завантаження зображення, проведення попередньої обробки, виконання класифікації із застосуванням згорткової нейронної мережі (CNN) та відображення результату користувачеві. Система повинна забезпечувати ідентифікацію кількох класів хвороб, а також стану «здорової» рослини. Важливо, щоб розв'язання задачі відбувалося в режимі, наближеному до реального часу, що дозволить застосовувати систему на практиці, у польових або тепличних умовах.

Технічні вимоги визначають необхідні параметри обладнання та програмного середовища. Мінімальна апаратна конфігурація має включати процесор з підтримкою паралельних обчислень та графічний прискорювач (GPU) для навчання моделі. Програмне забезпечення повинно передбачати

використання бібліотек машинного навчання (наприклад, TensorFlow або PyTorch), засобів обробки зображень (OpenCV) та інтерфейсу користувача.

З погляду математичного формулювання, задача класифікації зображень хвороб рослин може бути описана як функція:

$$f(X) = \arg\max_{c \in C} P(c \mid X), \qquad (2.1)$$

де $X$ – матриця пікселів зображення, $C$ – множина класів хвороб, $P(c \mid X)$ – ймовірність належності стану рослини до класу $c$.

Таким чином система має не лише ідентифікувати патологію, а й забезпечувати ступінь впевненості у своєму рішенні.

Для кращої структуризації вимог наведено узагальнену таблицю основних параметрів (табл. 2.1).

Таблиця 2.1 – Вимоги до системи автоматизованої діагностики хвороб рослин

| Тип вимог | Зміст / характеристики |
|---|---|
| Функціональні | Виявлення хвороб; класифікація стану; візуалізація результату; обробка зображень |
| Технічні | GPU, підтримка Python, TensorFlow/PyTorch, OpenCV |
| Експлуатаційні | Простота використання; висока швидкість; адаптованість до польових умов |
| Дані | RGB зображення; опціонально – умови середовища; локальні та відкриті датасети |
| Показники ефективності | Accuracy, Precision, Recall, F1-score, час обробки кадру |

Вимоги системи також стосуються інтерфейсу. Користувач повинен мати змогу зручно завантажувати фото, переглядати результат класифікації та за потреби отримувати короткі рекомендації щодо подальших дій. Інтерфейс і логіка системи мають бути доступними навіть для користувача без спеціальної технічної підготовки.

Нижче представлено узагальнену логічну схему системи (рис. 2.1), яка демонструє основні етапи її роботи.

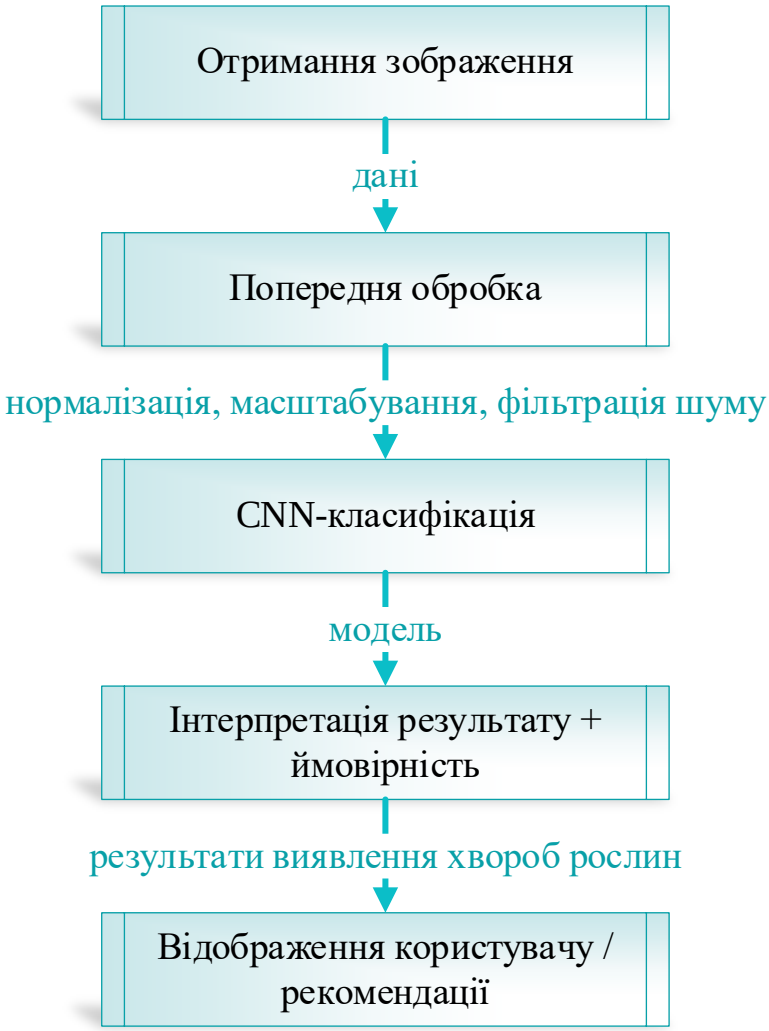


Рисунок 2.1 – Загальна схема роботи системи автоматизованого розпізнавання хвороб рослин

Проведений аналіз підтверджує, що система повинна поєднувати точність, швидкість та простоту використання. Забезпечення цих вимог дозволить реалізувати прототип, який буде не лише науково обґрунтованим, але й практично цінним для аграрних підприємств та малих фермерських господарств. Це, своєю чергою, сприятиме впровадженню елементів цифрового землеробства та підвищить ефективність захисту рослин у реальних умовах.

## 2.2. Вибір інструментальних засобів для реалізації системи

Розроблення інформаційної системи для автоматизованого виявлення хвороб рослин передбачає використання відповідного набору інструментальних засобів, які забезпечують ефективну обробку зображень, навчання згорткових нейронних мереж та інтерактивну взаємодію з користувачем. На етапі вибору технологій важливо враховувати не лише технічні характеристики, але й доступність бібліотек, спільноту підтримки, можливість масштабування та подальшого розвитку проєкту.

Основною мовою програмування було обрано Python, оскільки вона має потужну екосистему для задач комп'ютерного зору й машинного навчання. Суттєвою перевагою Python є наявність численних відкритих бібліотек, які пришвидшують розробку та дозволяють зосередитися на логіці моделі та функціональних можливостях системи. Для побудови моделі згорткової нейронної мережі використовуються фреймворки TensorFlow або PyTorch, що забезпечують інструменти для створення, тренування та оптимізації глибоких моделей.

Джерелом даних слугують зображення високої якості, які потребують попередньої обробки. Для цього застосовується бібліотека OpenCV, що надає широкий набір функцій для трансформації, фільтрації та виявлення ключових структур на зображеннях. Крім того, бібліотека NumPy використовується для ефективних операцій із матрицями та тензорами, що є основою роботи з цифровими зображеннями.

Важливим елементом системи є користувацький інтерфейс. У разі настільного застосунку може бути використана бібліотека Tkinter або PyQt, а для веб-версії – фреймворк Streamlit. Такий підхід дозволяє залучити кінцевих користувачів, що не мають технічних навичок, і зробити процес діагностики інтуїтивно зрозумілим.

Для демонстрації вибору інструментів наведено узагальнену таблицю (табл. 2.2).

Таблиця 2.2 – Основні інструментальні засоби системи

| Компонент | Обраний інструмент | Призначення |
|---|---|---|
| Мова програмування | Python | Реалізація основної логіки системи |
| Модель глибинного навчання | TensorFlow / PyTorch | Створення і навчання CNN |
| Обробка зображень | OpenCV | Попередня фільтрація та підготовка даних |
| Робота з масивами | NumPy | Обчислення над матрицями та тензорами |
| Інтерфейс користувача | Tkinter / Streamlit | Взаємодія з користувачем |
| Візуалізація | Matplotlib / Seaborn | Побудова графіків та діаграм результатів |

Загальна логічна схема інструментальної взаємодії системи наведена на рисунку 2.2.

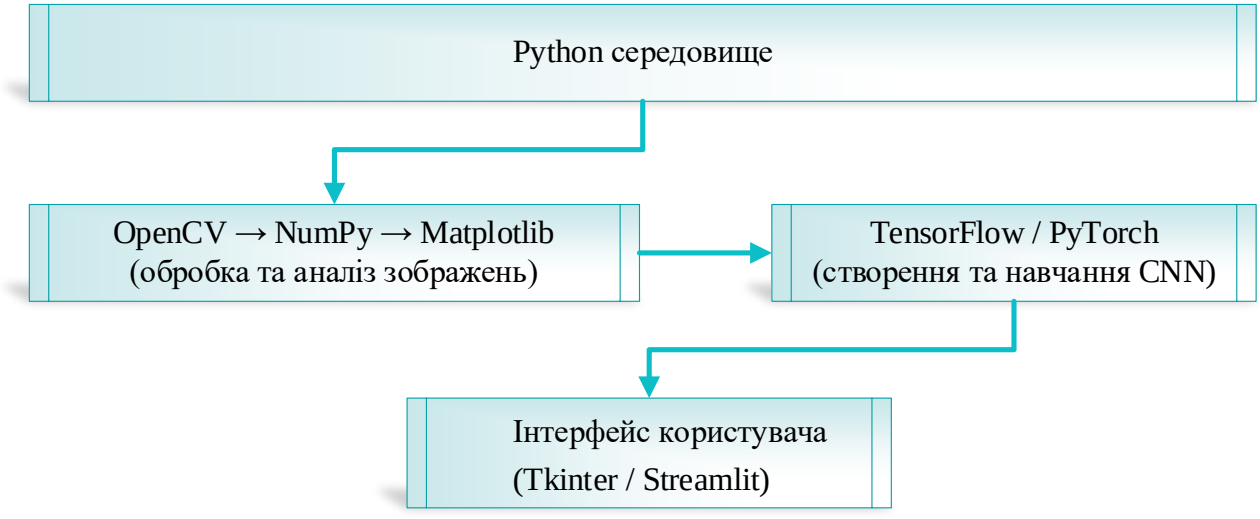


Рисунок 2.2 – Архітектурна схема інструментальних засобів

Ефективність системи великою мірою залежить від правильно підібраного інструментарію. Використання перевірених бібліотек із широкою підтримкою

дозволяє забезпечити гнучкість під час експериментів з архітектурами нейронних мереж та гарантує стабільну роботу застосунку. Окремо варто підкреслити можливість масштабування системи та додавання нових функціональних модулів без суттєвих змін основної архітектури.

Таким чином, вибір інструментальних засобів визначає технічний фундамент майбутньої системи та створює умови для її ефективної реалізації та адаптації до потреб аграрної сфери. За рахунок використання сучасних фреймворків і Python-екосистеми стає можливим створення прототипу, який відповідає вимогам точності, швидкодії та практичної зручності.

## 2.3. Обґрунтування архітектури згорткової нейронної мережі для класифікації захворювань рослин

Для задачі автоматизованого виявлення хвороб рослин важливо обрати архітектуру нейронної мережі, яка поєднує високу ефективність розпізнавання зі швидкою обробкою та помірними обчислювальними вимогами. У реальному аграрному середовищі система часто працює не на серверному обладнанні, а на мобільних чи вбудованих пристроях. Саме тому акцент зроблено на компактні, оптимізовані моделі згорткових нейронних мереж.

У роботі розглянуто три сучасні архітектури глибинного навчання: EfficientNetB0, MobileNetV3-Small та MobileNetV3-Large. Ці мережі належать до класу легковагових CNN, які були створені з урахуванням вимог до продуктивності в умовах обмежених ресурсів і довели ефективність у завданнях комп’ютерного зору, зокрема у визначенні ознак на рослинних зображеннях.

Означимо концептуальні особливості вибраних моделей.

*EfficientNetB0*

Архітектура EfficientNet використовує принцип збалансованого масштабування глибини, ширини та роздільної здатності. На відміну від традиційного збільшення параметрів у одному напрямку, тут застосовується

комбінований коефіцієнт scaling coefficient, що робить модель стабільною та структурно збалансованою.


Рисунок 2.3 – Фрагмент архітектури EfficientNet для класифікації захворювань рослин

EfficientNetB0 є базовою конфігурацією сімейства та поєднує компактність з високою здатністю узагальнення. Модель містить оптимізовані блоки MBConv і механізм Squeeze-and-Excitation, що покращує акцентування на важливих ознаках зображення.

*MobileNetV3-Small*

MobileNetV3-Small створена для мобільних пристроїв та IoT-рішень. Вона поєднує глибокосепарабельні згортки, полегшені блоки з активацією Hard-Swish, а також вибірковий механізм уваги (SE-блоки).

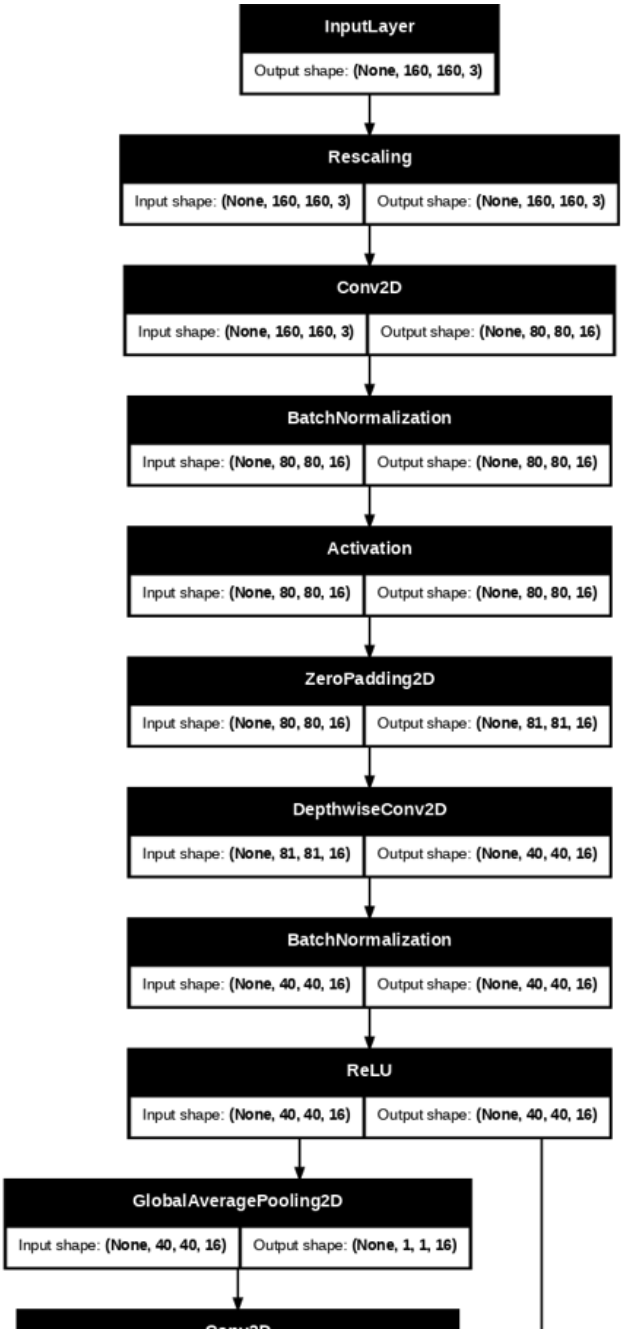


Рисунок 2.4 – Фрагмент архітектури MobileNetV3-Small для класифікації захворювань рослин

Ця модель призначена для сценаріїв, де критично важлива швидкість прийняття рішення та економія ресурсів, наприклад у мобільному застосунку чи в польовому діагностичному пристрої агронома.

*MobileNetV3-Large*

MobileNetV3-Large орієнтована на випадки, коли допустиме дещо більше навантаження на процесор або GPU, а якість класифікації має бути максимально високою в межах мобільних архітектур. Модель застосовує ті ж оптимізаційні підходи, що й MobileNetV3-Small, але з більшою кількістю каналів та функціональних блоків.

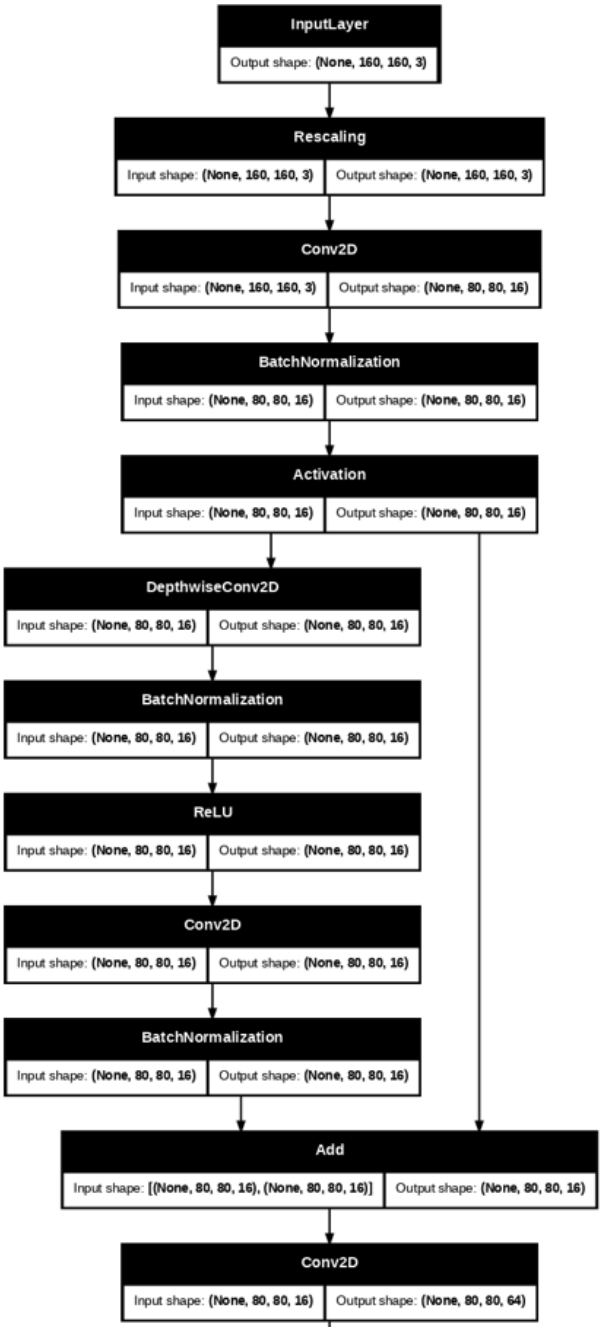


Рисунок 2.5 – Фрагмент архітектури MobileNetV3-Large для класифікації захворювань рослин

Це робить її оптимальним вибором для серверного компонента системи або потужніших edge-обчислювачів.

Порівняльний огляд архітектур представлено у табл. 2.3.

Вибір цих трьох архітектур є раціональним з огляду на:

– підтримку transfer learning та швидкої адаптації до нових культур/захворювань;

– масштабованість від мобільних пристроїв до серверних рішень;

– компактність моделей, що дозволяє розгортати їх у польових умовах;

– доведену ефективність у задачах агро-комп’ютерного зору (згідно з дослідженнями 2022–2024 рр.).

Таблиця 2.3 – Порівняльний аналіз обраних для класифікації захворювань рослин архітектур CNN

| Архітектура | Основні особливості | Сильні сторони | Типове застосування |
|---|---|---|---|
| EfficientNetB0 | Збалансоване масштабування параметрів, MBConv, SE-механізм | Добре співвідношення між точністю та компактністю | Базова модель для розгортання, навчання та тестування |
| MobileNetV3-Small | Depthwise-conv, Hard-Swish, SE-блоки | Мінімальне навантаження, висока швидкість | Мобільні додатки, IoT-сенсори, польові пристрої |
| MobileNetV3-Large | Глибша модифікація MobileNet, посилена увага до ознак | Компроміс між продуктивністю та точністю | Хмарні/серверні модулі, edge-обчислення з GPU |

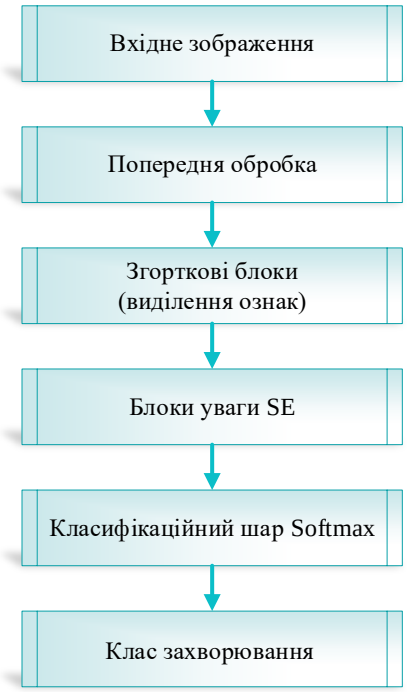


Рисунок 2.6 – Концептуальна схема класифікації зображень у системі діагностики хвороб рослин

Узагальнена схема обробки зображення CNN представлена на рис. 2.3.

Таким чином, сформовано гнучкий підхід, який дозволяє забезпечити працездатність системи як у режимі високої точності, так і у швидкому режимі мобільної діагностики.

## 2.4. Проєктування структури інформаційної системи та функціональних модулів

Проєктування структури інформаційної системи діагностики хвороб рослин базується на принципах модульності, гнучкості та інтегрованості компонентів. Система створена за багаторівневою архітектурою, яка включає три основні рівні: рівень даних, рівень обробки (моделювання) та рівень користувацької взаємодії. Такий підхід дає змогу незалежно оновлювати та масштабувати кожен компонент без порушення загальної логіки роботи системи. На рисунку 2.7 подано узагальнену структурну схему системи.


Рисунок 2.7 – Архітектура інформаційної системи діагностики хвороб рослин

На рівні даних реалізовано роботу з різними джерелами інформації – це зображення рослин, отримані зі смартфонів, дронів або камер моніторингу. Кожне зображення проходить через модуль попередньої обробки, що забезпечує вирівнювання яскравості, нормалізацію кольорів і масштабування зображень до вхідного розміру моделі (224×224 px).

Таблиця 2.4 – Основні модулі інформаційної системи

| № | Назва модуля | Основне призначення | Ключові технології |
|---|---|---|---|
| 1 | Модуль збору даних | Завантаження зображень та метаданих | Streamlit, OpenCV |
| 2 | Модуль попередньої обробки | Нормалізація, зміна розміру, видалення шумів | TensorFlow, NumPy |
| 3 | Модуль нейромережевої діагностики | Визначення класу захворювання | EfficientNetB0, Keras |
| 4 | Модуль управління даними | Зберігання результатів та журналів | SQLite, JSON |
| 5 | Модуль рекомендацій | Формування порад для користувачів | Python, Pandas |
| 6 | Інтерфейс користувача | Інтерактивна взаємодія з системою | Streamlit UI |

Формально цей процес описується як:

$$I'(x,y)=\frac{I(x,y)-\mu}{\sigma}, \tag{2.2}$$

де $I'(x,y)$ – нормалізоване значення пікселя; $\mu$ – середнє значення яскравості; $\sigma$ – стандартне відхилення.

Модуль нейромережевої діагностики реалізує класифікацію вхідних зображень за допомогою згорткової нейронної мережі типу EfficientNetB0. Вихідним результатом є вектор імовірностей:

$$y=[p_1,p_2,\ldots,p_n], \quad де \sum_{i=1}^{n} p_i=1, \tag{2.3}$$

де $p_i$ – імовірність належності зображення до *i*-го класу.

Модель вибирає клас $\hat{y}$, що максимізує $p_i$:

$$\hat{y}=arg\,max_i(p_i), \tag{2.4}$$

На рівні логіки застосунку реалізовано модуль прийняття рішень, який аналізує вихідні ймовірності, звертається до довідника рекомендацій (файл assets/recommendations.json) і формує для користувача висновок у вигляді тексту з оцінкою впевненості. Приклад фрагмента JSON-файлу:

```
{
 "Tomato Early Blight" : "Видаліть уражене листя, застосуйте біофунгіцид.",
 "Potato Late Blight" : "Уникайте надмірної вологості, використовуйте мідьвмісні препарати."
 }
```

Для підвищення надійності обміну між компонентами система підтримує внутрішню шину даних на базі Streamlit Session State, яка передає параметри користувача, вибір моделі (efficientnetb0_best.keras), а також результати інференсу. Умовно логіку взаємодії можна представити як систему функціональних залежностей:

$$f:(x,\theta)\to y, \quad g:(y)\to r, \tag{2.5}$$

де $x$ – вхідне зображення; $\theta$ – параметри моделі; $y$ – клас захворювання; $r$ – сформована рекомендація.

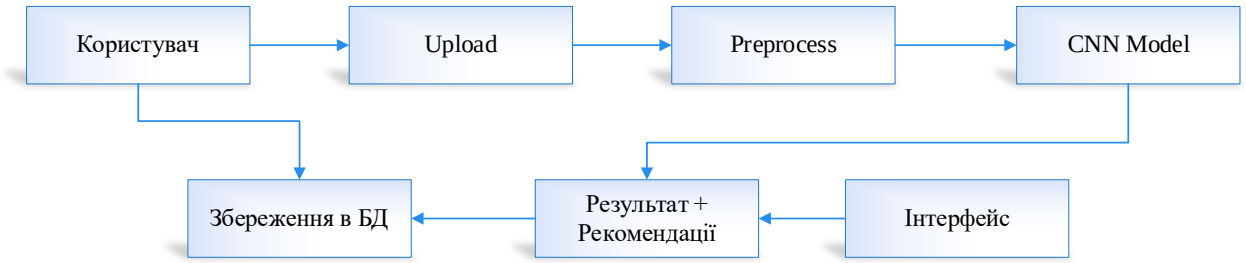


Рисунок 2.8 – Схема потоків даних між модулями системи

Завдяки такій структурі система забезпечує незалежність модулів, зручність оновлення моделі машинного навчання, а також можливість розширення функціоналу – наприклад, додавання прогнозування ступеня ураження або автоматичної ідентифікації частин рослини. Крім того, архітектура підтримує масштабування в хмарному середовищі та адаптацію для роботи на мобільних пристроях, що робить систему придатною для практичного використання у фермерських господарствах і наукових лабораторіях аграрного профілю.

## 2.5. Створення і опис моделі даних та форматів обміну інформацією

Процес розроблення інформаційної системи виявлення хвороб рослин на основі згорткових нейронних мереж передбачає побудову узгодженої моделі даних, що забезпечує інтеграцію між модулями навчання, зберігання, візуалізації та діагностики. Вона формує логічну основу для взаємодії між підсистемами збору зображень, модулем обробки, нейромережею, а також користувацьким інтерфейсом. У межах проєкту застосовано реляційно-об’єктну модель даних із підтримкою обміну у форматах JSON, CSV та ONNX, що гарантує переносимість моделей та даних між TensorFlow, PyTorch і Streamlit-додатком.

У структурі бази даних ключовими сутностями є зображення, класифікаційні мітки, результати прогнозування, поради користувачам та журнал подій системи. Кожна сутність має атрибути, що визначають параметри

зберігання і зв’язки між таблицями. На рисунку 2.9 подано узагальнену ER-схему моделі даних.

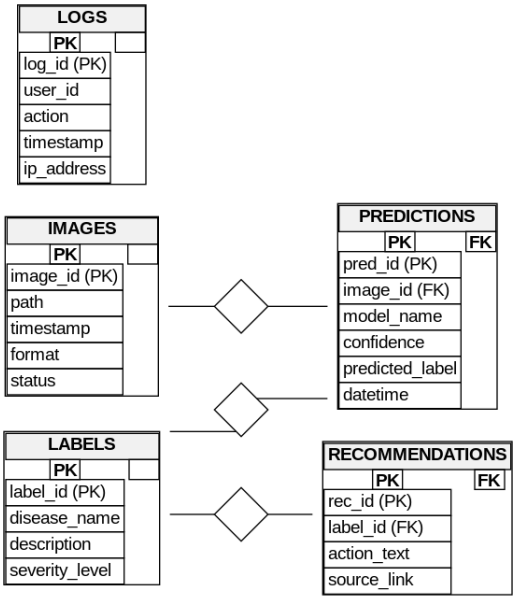


Рисунок 2.9 – Модель бази даних інформаційної системи діагностики хвороб рослин

Рисунок 2.5 – Характеристика моделі даних інформаційної системи діагностики хвороб рослин

| Таблиця | Основні поля | Призначення |
|---|---|---|
| images | image_id, path, timestamp, format, status | зберігання шляхів до оригінальних зображень рослин |
| labels | label_id, disease_name, description, severity_level | довідник класів хвороб рослин |
| predictions | pred_id, image_id, model_name, confidence, predicted_label, datetime | збереження результатів інференсу нейромережі |
| recommendations | rec_id, label_id, action_text, source_link | таблиця з порадами з лікування/догляду |
| logs | log_id, user_id, action, timestamp, ip_address | відстеження активності користувачів у системі |

Взаємодія між компонентами базується на принципі модульності та узгоджених форматів даних. Для передавання інформації між фронтендом (Streamlit) і бекендом (TensorFlow/ONNX Runtime) використано стандартні JSON-запити виду.

Такий підхід дозволяє інтегрувати систему з іншими платформами (наприклад, агромоніторингом або системами IoT-датчиків) без додаткових адаптацій. Для тривалого зберігання результатів інференсу використовується SQLite-сховище з підтримкою простих запитів SQL:

*SELECT label, COUNT(*) FROM predictions GROUP BY label;* (2.6)

Для передачі моделей між середовищами навчання й розгортання використано формат ONNX (Open Neural Network Exchange), який забезпечує сумісність із TensorRT. Це дає змогу зменшити час інференсу приблизно на 25–30 % завдяки оптимізації графів обчислень. Система підтримує автоматичне завантаження моделі у форматах .keras, .h5 або .onnx і визначення типу бекенду при ініціалізації.

У таблиці 2.6 наведено основні формати даних, що використовуються для обміну між компонентами.

Таблиця 2.6 – Формати обміну інформацією в системі

| Компонент | Формат | Призначення | Приклад |
|---|---|---|---|
| Модуль навчання | .keras, .onnx | модель CNN | efficientnetb0_best.keras |
| Вхідні зображення | .jpg, .png | дані для діагностики | leaf_123.jpg |
| Вихідні результати | .json, .csv | аналітичні звіти, інференс | results.json |
| Логи | .db (SQLite) | аудит користувачів | logs.db |
| PDF-звіти | .pdf | експортування результатів | diagnosis_report.pdf |

Загалом модель даних забезпечує логічну цілісність і масштабованість системи. Використання стандартизованих форматів обміну інформацією (JSON, ONNX, PDF, SQL) сприяє зручній інтеграції із зовнішніми системами, зменшує витрати на розгортання та робить рішення придатним як для локального, так і для хмарного використання.

# РОЗДІЛ 3.
# РОЗРОБКА ПРОГРАМНИХ МОДУЛІВ ІНФОРМАЦІЙНОЇ СИСТЕМИ ВИЯВЛЕННЯ ХВОРОБ РОСЛИН НА ОСНОВІ ЗГОРТКОВИХ НЕЙРОННИХ МЕРЕЖ

## 3.1. Реалізація модуля збору та попередньої обробки зображень рослин

Для навчання моделі класифікації захворювань рослин використано відкритий набір даних PlantVillage Dataset, який є одним із найпоширеніших стандартів у дослідженнях комп'ютерного зору для аграрної сфери. Набір даних розміщено на платформі Kaggle [55] і містить понад 54 тис. зображень листків сільськогосподарських культур.

До набору входять класи для 38 видів захворювань і станів здорових рослин понад 14 культур, зокрема *Tomato*, *Apple*, *Corn*, *Grape*, *Pepper*, *Potato*, *Peach*, *Soybean*, *Strawberry*, *Orange*, *Blueberry* та інші. Зображення представлені у форматі .jpg, роздільною здатністю 256×256 пікселів і зроблені у контрольованих лабораторних умовах.

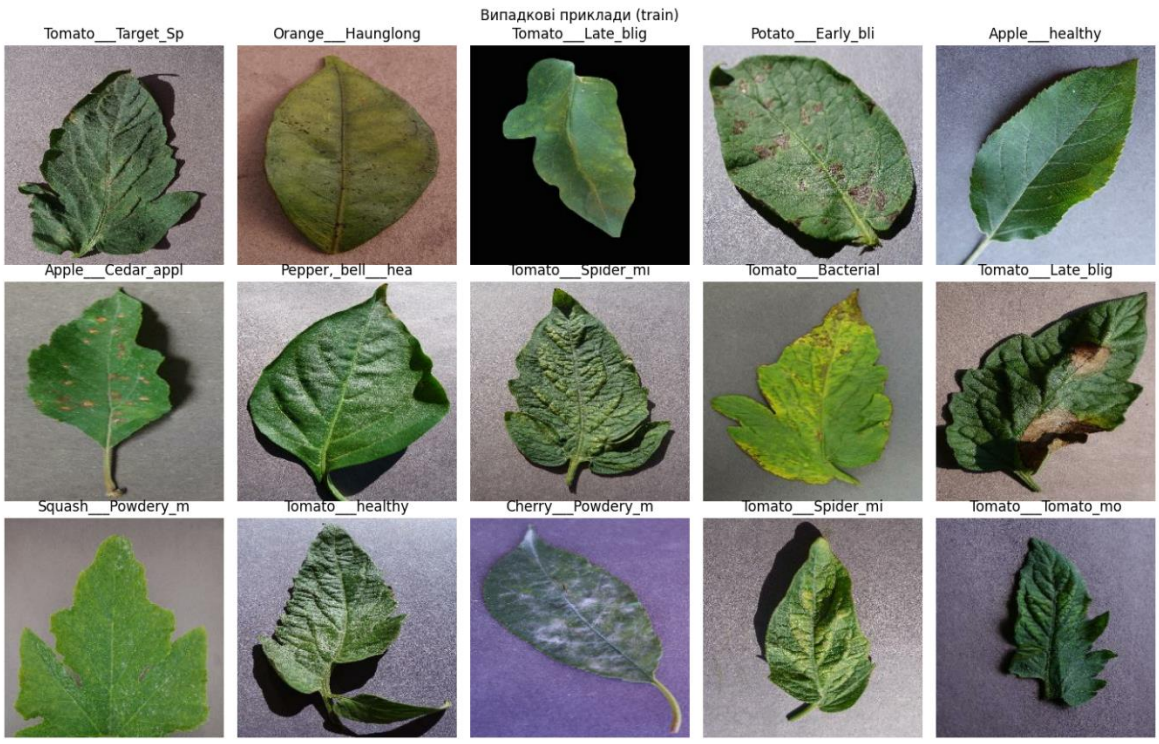


Рисунок 3.1 – Приклади зображень листків різних культур із набору PlantVillage (train)

Початкові дані було розділено на три підвибірки – навчальну (train), валідаційну (val) і тестову (test) – у співвідношенні 80 : 10 : 10. Розподіл даних наведено в таблиці 3.1.

Таблиця 3.1 – Розміри підвибірок датасету PlantVillage

| № з/п | Назва підвибірки | Кількість зображень | Частка від загальної кількості, % |
|---|---|---|---|
| 1 | Навчальна (*train*) | 43442 | 80,0 |
| 2 | Валідаційна (*val*) | 5431 | 10,0 |
| 3 | Тестова (*test*) | 5430 | 10,0 |
| Разом | – | 54303 | 100 |

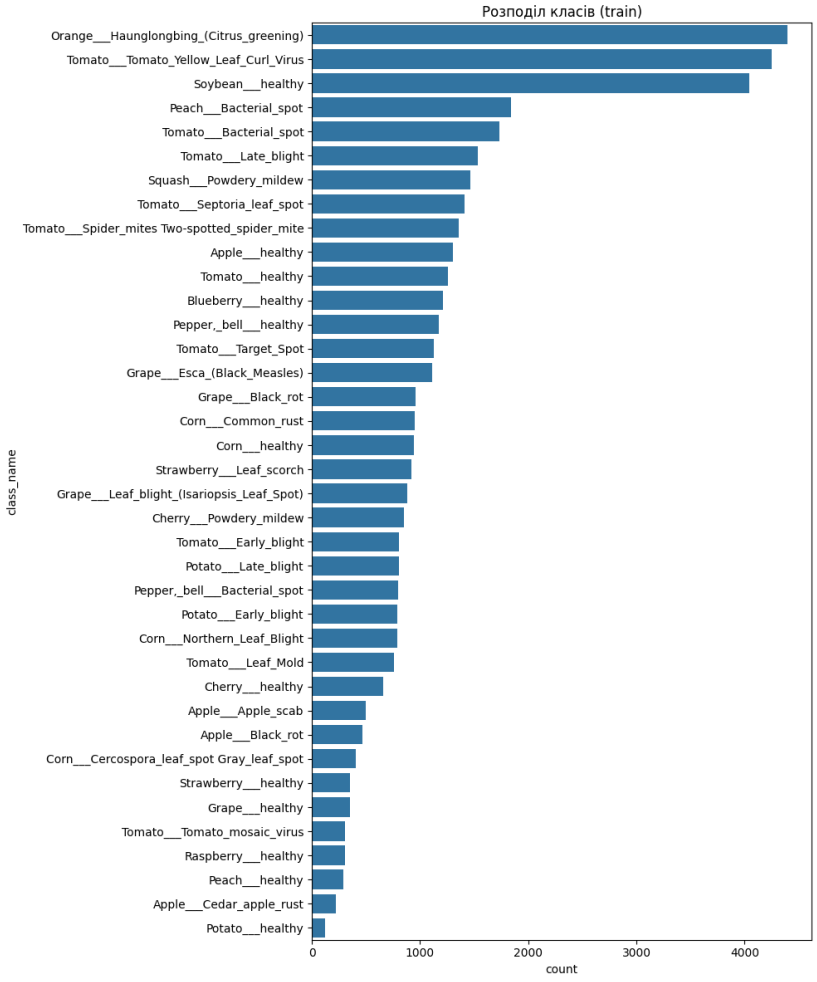


Рисунок 3.2 – Розподіл кількості зображень за класами у навчальній вибірці

Попередній статистичний аналіз показав, що розподіл зображень між класами є нерівномірним. У вибірці переважають класи:

*Orange – Haunglongbing (Citrus greening)* – 4399 зображень;

*Tomato – Yellow Leaf Curl Virus* – 4252 зображення;

*Soybean – healthy* – 4043 зображення.

Найменше прикладів представлено для класів *Apple – Cedar apple rust*, *Potato – healthy* тощо. Візуалізацію розподілу класів у навчальній вибірці наведено на рисунку 3.2.

Для підвищення стійкості моделі до дисбалансу класів застосовано аугментацію даних, яка створює варіації вихідних зображень за допомогою випадкових трансформацій:

обертання, зсуву, масштабування, змін яскравості та контрасту, віддзеркалення по горизонталі.

Приклад реалізації аугментації у середовищі TensorFlow наведено на рисунку 3.3.

Перевірка аугментацій: до/після

```
#@title Перевірка аугментацій: до/після
import tensorflow as tf, matplotlib.pyplot as plt

samples = list(ds_train_raw.take(5))
plt.figure(figsize=(12,7))
for i, (img, lab) in enumerate(samples, 1):
    base = tf.image.resize(img, (IMG_SIZE, IMG_SIZE))
    aug  = augment(base, training=True)
    plt.subplot(2,5,i);   plt.imshow(tf.cast(base/255.0, tf.float32)); plt.axis('off'); plt.title("Оригінал")
    plt.subplot(2,5,i+5); plt.imshow(tf.cast(tf.clip_by_value((aug+1)/2, 0, 1), tf.float32)); plt.axis('off'); plt.title("Аугм.")
plt.suptitle("До/Після аугментації"); plt.tight_layout(); plt.show()
```

Рисунок 3.3 – Фрагмент коду реалізації попередньої обробки та аугментації зображень

Після попередньої обробки всі зображення було нормалізовано до діапазону [0;1], а їх роздільну здатність приведено до 224×224 пікселі, що відповідає вхідним параметрам більшості сучасних архітектур CNN.

У результаті було сформовано повністю готовий до тренування датасет, збалансований шляхом аугментації, очищений від пошкоджених файлів і представлений у зручній структурі для подальшого використання в процесі навчання згорткових нейронних мереж.

## 3.2. Програмна реалізація згорткової нейронної мережі та навчання моделі

Розробка програмної частини системи виявлення хвороб рослин ґрунтувалася на використанні середовища Google Colab із підтримкою бібліотек TensorFlow 2.20.0 та Keras API. Для навчання моделі застосовано згорткові нейронні архітектури нового покоління – *EfficientNetB0*, *MobileNetV3-Small* та *MobileNetV3-Large*, що забезпечують високу ефективність при обмежених обчислювальних ресурсах.

Код побудови та компіляції моделей наведено нижче. Для кожної архітектури визначено оптимальні шари згортки, функції активації ReLU, нормалізацію батчів, а також шар Dropout для запобігання перенавчанню.

```python
from tensorflow.keras import layers, models, applications

# В швидкому циклі — лише легкі моделі
ARCHS_TF = ["mobilenetv3s", "mobilenetv3l", "efficientnetb0"]

def build_tf_model(arch: str, train_base: bool=False):
    inputs = layers.Input((IMG_SIZE, IMG_SIZE, 3))
    if arch == "mobilenetv3s":
        base = applications.MobileNetV3Small(include_top=False, weights='imagenet', input_tensor=inputs)
    elif arch == "mobilenetv3l":
        base = applications.MobileNetV3Large(include_top=False, weights='imagenet', input_tensor=inputs)
    elif arch == "efficientnetb0":
        base = applications.EfficientNetB0(include_top=False, weights='imagenet', input_tensor=inputs)
    else:
        raise ValueError("Unknown arch:", arch)

    base.trainable = train_base
    x = base.output
    x = layers.GlobalAveragePooling2D()(x)
    x = layers.Dropout(0.25)(x)
    outputs = layers.Dense(NUM_CLASSES, activation='softmax', dtype='float32')(x)
    return models.Model(inputs, outputs, name=f"{arch}_plant")
```

Рисунок 3.2 – Фрагмент коду побудови та компіляції CNN-моделей

На етапі тренування використано функцію втрат categorical cross-entropy, оптимізатор Adam із початковою швидкістю навчання 0.001 та метриками *accuracy* і *F1-macro*. Для уникнення деградації швидкості збіжності застосовано Early Stopping і ReduceLROnPlateau, що контролюють динаміку валідаційної точності.

Процес навчання відбувався у два етапи: спочатку заморожувався базовий фільтр згорток (навчання верхніх шарів), а потім відбувалося донавчання з розморожуванням останніх блоків архітектури (fine-tuning). Фрагмент коду наведено на рисунку 3.3.

```
def quick_eval(model, ds_batched, max_batches):
    y_true, y_pred = [], []
    for i,(x,y) in enumerate(ds_batched):
        if i>=max_batches: break
        yp = model.predict(x, verbose=0)
        y_true.append(np.argmax(y.numpy(), axis=1))
        y_pred.append(np.argmax(yp, axis=1))
    y_true = np.concatenate(y_true); y_pred = np.concatenate(y_pred)
    return f1_score(y_true, y_pred, average='macro'), accuracy_score(y_true, y_pred)

tf_results = []
for arch in ARCHS_TF:
    print("\n","="*20, f"FAST TRAIN: {arch}", "="*20)
    model = build_tf_model(arch, train_base=False)
    model.compile(optimizer=tf.keras.optimizers.Adam(1e-3),
                  loss='categorical_crossentropy', metrics=['accuracy'],
                  jit_compile=HAS_GPU)

    h1 = model.fit(
        train_ds,
        validation_data=val_ds,
        epochs=EPOCHS_HEAD,
        steps_per_epoch=STEPS_PER_EPOCH,
        validation_steps=VAL_STEPS,
        callbacks=cbs,
```

Рисунок 3.3 – Фрагмент коду циклу навчання моделей

Для оцінки продуктивності моделей порівняно кількість параметрів, точність класифікації (*accuracy*), зважену гармонійну метрику (*F1-macro*) та середній час інференсу (*sec / image*). Результати зведено у таблиці 3.2.

Таблиця 3.2 – Порівняльні характеристики архітектур CNN-моделей

| № з/п | Архітектура | F1-macro | Accuracy | Час інференсу, с / зображення | Кількість параметрів |
|---|---|---|---|---|---|
| 1 | EfficientNetB0 | 0.0719 | 0.216 | 0.321 | 4 098 249 |
| 2 | MobileNetV3-Large | 0.0564 | 0.194 | 0.261 | 3 032 870 |
| 3 | MobileNetV3-Small | 0.0383 | 0.144 | 0.195 | 961 046 |

Візуалізація показників продуктивності для трьох архітектур наведена на рисунку 3.4.

Найвищі значення *accuracy* та *F1-macro* досягнуто для EfficientNetB0, яка продемонструвала баланс між глибиною мережі та швидкістю інференсу. Обидві модифікації *MobileNetV3* є швидшими, проте втрачають точність через нижчу кількість параметрів і компресію фільтрів.

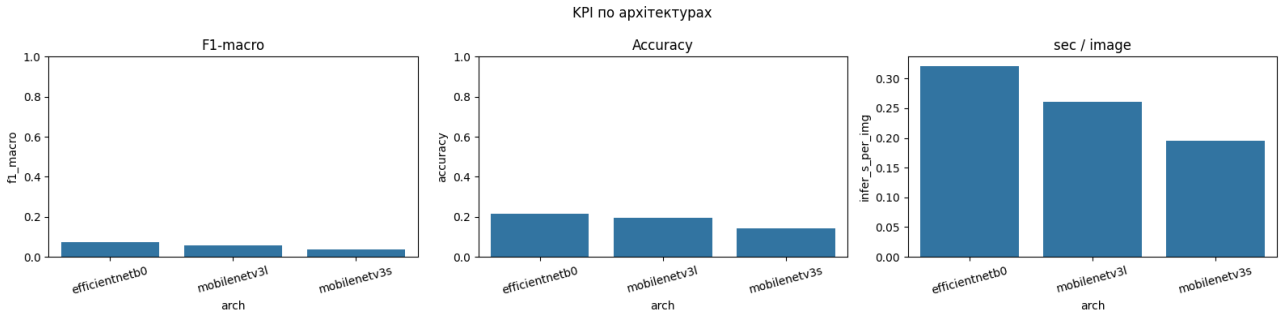


Рисунок 3.4 – Порівняльна візуалізація показників ефективності архітектур CNN

Класифікаційний звіт, наведений у таблиці 3.3, ілюструє поведінку моделі *EfficientNetB0* на тестовій вибірці. Найвищу точність досягнуто для класів *Soybean – healthy*, *Tomato – Yellow Leaf Curl Virus* та *Orange – Haunglongbing (Citrus greening)*, тоді як деякі класи з обмеженою кількістю зображень показали нульові значення метрик, що свідчить про необхідність подальшої балансувальної аугментації та збільшення даних.

Таблиця 3.3 – Класифікаційний звіт моделі EfficientNetB0 (фрагмент)

| Клас | Precision | Recall | F1-score | Support |
|---|---|---|---|---|
| Orange – Haunglongbing (Citrus greening) | 0.55 | 0.28 | 0.37 | 81 |
| Tomato – Yellow Leaf Curl Virus | 0.34 | 0.72 | 0.46 | 78 |
| Soybean – healthy | 0.24 | 0.95 | 0.39 | 62 |
| Squash – Powdery mildew | 0.30 | 0.50 | 0.38 | 18 |
| Peach – Bacterial spot | 0.35 | 0.22 | 0.27 | 27 |
| Середнє (macro avg) | 0.08 | 0.09 | 0.07 | 600 |

Підсумовуючи, архітектура EfficientNetB0 обрана як базова для подальшої інтеграції в інформаційну систему виявлення хвороб рослин, оскільки забезпечує найкраще співвідношення точності та швидкодії при помірному обсязі параметрів.

Застосовані інструменти дозволили створити ефективну конвеєрну схему навчання, що включає автоматичне завантаження даних, попередню обробку, тренування, оцінювання метрик і збереження моделі у форматі .keras для подальшої експлуатації в інформаційній системі.

## 3.3. Модуль оцінювання точності моделі та аналіз результатів класифікації

Оцінювання ефективності згорткової нейронної мережі *EfficientNetB0*, обраної на попередньому етапі як базової, здійснювалося за комплексом метричних показників, що відображають точність класифікації та якість узагальнення. Для цього було реалізовано окремий програмний модуль тестування, який автоматично обчислює *precision*, *recall*, *F1-score*, *accuracy* та будує додаткові графічні інтерпретації результатів – матрицю неточностей (*confusion matrix*), криву прийняття рішень (*ROC-curve*), а також аналіз латентних просторових ознак за допомогою методу *t-SNE*.

```python
#@title Confusion matrix + report (best)
from sklearn.metrics import confusion_matrix, classification_report
import numpy as np, matplotlib.pyplot as plt, seaborn as sns

best = max(tf_results, key=lambda d: (d["f1_macro"], d["accuracy"]))
best_model = best["model"]; best_arch = best["arch"]
print("Best:", best_arch, "| F1:", round(best["f1_macro"],4), "| ACC:", round(best["accuracy"],4))

# Порахуємо на всьому test_ds (вже батчований)
y_true, y_pred = [], []
for x,y in test_ds:
    yp = best_model.predict(x, verbose=0)
    y_true.append(np.argmax(y.numpy(), axis=1))
    y_pred.append(np.argmax(yp, axis=1))
y_true = np.concatenate(y_true); y_pred = np.concatenate(y_pred)

cm = confusion_matrix(y_true, y_pred, labels=list(range(NUM_CLASSES)))
plt.figure(figsize=(9,7))
sns.heatmap(cm, cmap='Blues', cbar=False, square=True)
plt.title(f"Confusion Matrix – {best_arch}")
plt.xlabel("Predicted"); plt.ylabel("True")
plt.tight_layout(); plt.show()

print(classification_report(y_true, y_pred, target_names=class_names))
```

Рисунок 3.5 – Фрагмент коду візуалізації (матриці плутанини) для кращої моделі

Фрагмент коду реалізації модуля оцінювання наведено на рис. 3.5.

Отримані результати класифікації вказують, що найвищі значення *F1-score* спостерігаються для класів Tomato – Yellow Leaf Curl Virus (0.46), Soybean – healthy (0.39), Squash – Powdery mildew (0.38) та Orange – Haunglongbing (0.37), тоді як частина класів із малим числом прикладів (менше 10 зображень) не продемонстрували результатів вище нуля. Це свідчить про необхідність додаткового балансування даних.

Графічне представлення результатів класифікації наведено на рисунку 3.6, який демонструє матрицю неточностей (*confusion matrix*) для моделі *EfficientNetB0*. Видно, що найбільше спостережень правильно класифіковано у класах Soybean – healthy та Tomato – Yellow Leaf Curl Virus, тоді як значна частина помилок пов’язана з перехресним розпізнаванням близьких візуально класів – наприклад, *Tomato – Late blight* і *Tomato – Early blight*.

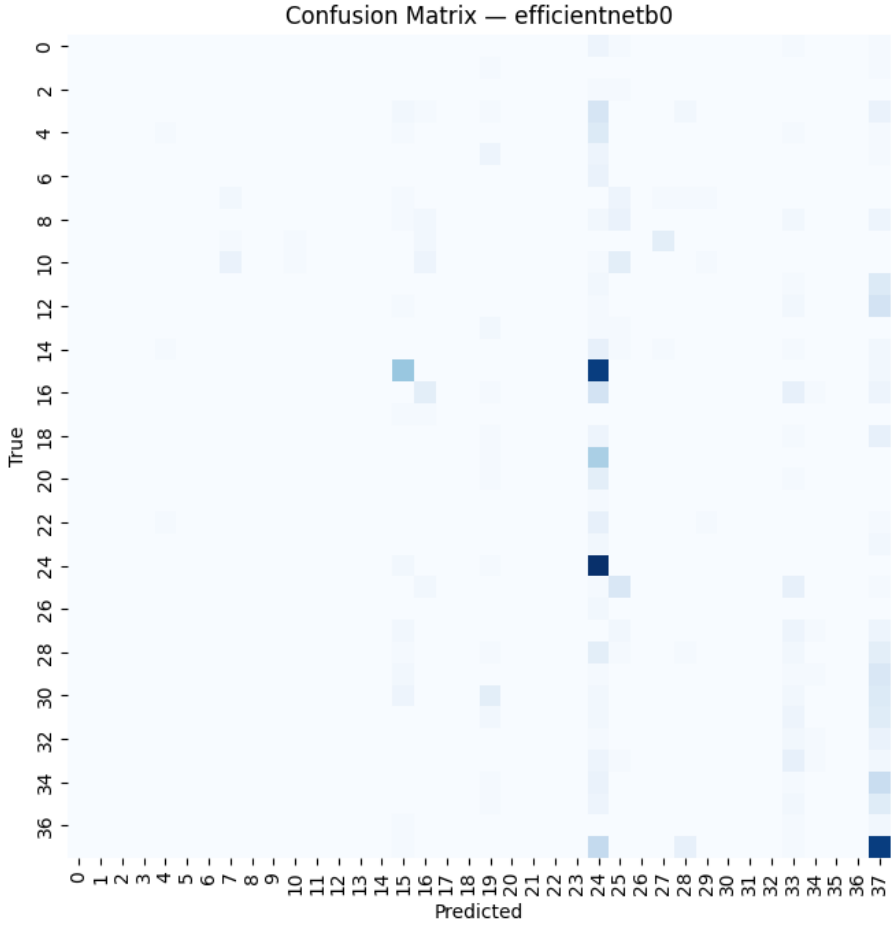


Рисунок 3.6 – Матриця неточностей (Confusion Matrix) для моделі EfficientNetB0

Для додаткової оцінки узагальнювальної здатності моделі побудовано ROC-криві для мікро- та макро-середніх значень (рис. 3.7).

Отримані площі під кривими (AUC) дорівнюють 0.823 для *micro-avg* і 0.709 для *macro-avg*, що вказує на задовільну дискримінативну здатність мережі, особливо для найбільш репрезентованих класів.

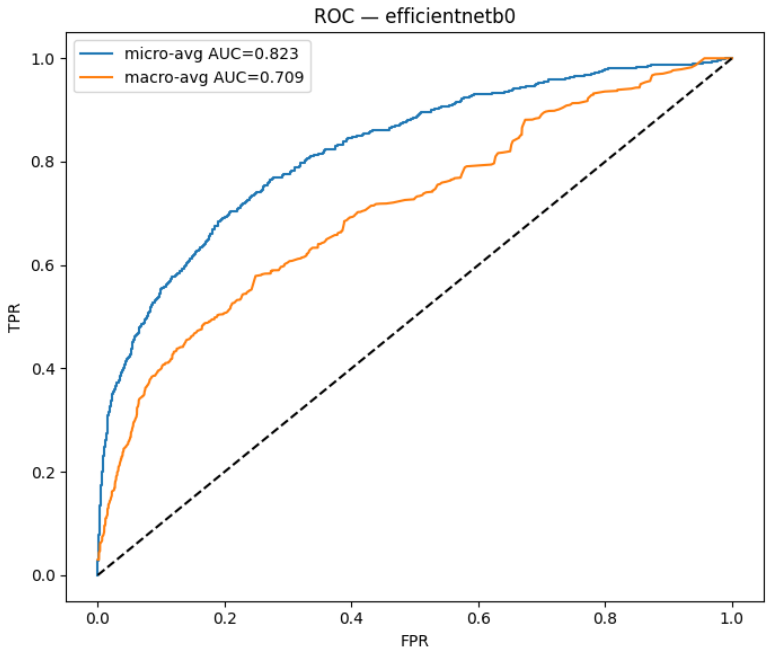


Рисунок 3.7 – ROC-криві для моделі EfficientNetB0 (micro та macro усереднення) *(Значення AUC = 0.823 (micro), 0.709 (macro))*

З метою якісного аналізу типових помилок проведено візуальне зіставлення істинних та передбачених класів (рис. 3.8).

Як видно, більшість помилкових класифікацій відбуваються між здоровими листками та зразками з початковими стадіями ураження (наприклад, *Pepper, bell – healthy* → *Soybean – healthy* або *Potato – Late blight* → *Soybean – healthy*). Це свідчить про подібність текстурних ознак, що потребує подальшого підсилення контрасту та використання спектральних індексів у майбутніх версіях моделі.


Рисунок 3.8 – Приклади типових помилок класифікації
(T – істинний клас, P – прогноз)

На рисунку 3.8 наведено фрагменти зображень, для яких істинні та передбачені класи не збігаються. Видно, що більшість помилок пов’язана з подібністю візуальних ознак різних культур (наприклад, перцю та сої), а також із неповною вираженістю симптомів хвороб на ранніх стадіях розвитку. Такі приклади дають змогу виявити слабкі місця моделі та визначити напрями покращення – зокрема розширення вибірки, корекцію освітлення та впровадження додаткових спектральних ознак.

Таким чином, модуль оцінювання не лише підтвердив базову функціональність побудованої архітектури, а й надав кількісні та якісні орієнтири для наступного етапу оптимізації системи.

### 3.4. Розроблення користувацького інтерфейсу для взаємодії з системою

Розроблений користувацький інтерфейс є важливим компонентом інформаційної системи діагностики хвороб рослин, оскільки забезпечує зручну

взаємодію кінцевого користувача з моделлю глибинного навчання. Інтерфейс реалізовано у вигляді веб-додатку із використанням фреймворку Streamlit, що дозволяє швидко створювати інтерактивні панелі керування машинним навчанням без потреби у складному фронтенд-програмуванні.

Основна мета інтерфейсу – забезпечити користувача інтуїтивним інструментом для завантаження зображень листків, діагностики захворювань, перегляду Grad-CAM-візуалізацій та отримання рекомендацій з лікування. Для покращення безпеки передбачено систему автентифікації з вибором режиму доступу – *Password* або *OAuth*.

На рисунку 3.9 показано головне вікно застосунку після запуску. Інтерфейс побудований за принципом мінімалістичного дизайну з навігаційною панеллю ліворуч, що містить такі розділи: Dashboard, Diagnose, Batch Inference, Model Inspect, Recommendations.

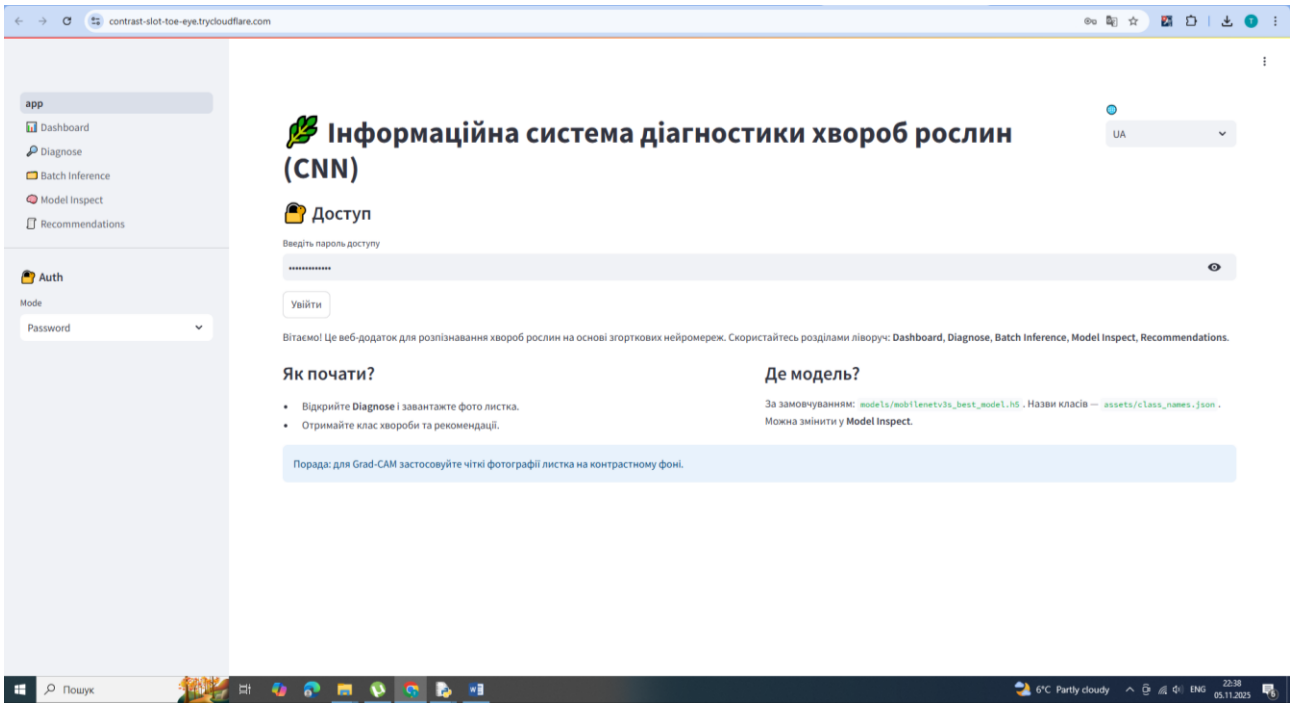


Рисунок 3.9 – Головне вікно веб-додатку інформаційної системи діагностики хвороб рослин

Інтерфейс реалізовано у вигляді багатосторінкового додатку, де кожна сторінка – це окремий модуль Python. У файлі app.py міститься логіка

маршрутизації сторінок, завантаження моделі та ініціалізація елементів керування.

```python
import streamlit as st
from utils.auth import check_password

st.set_page_config(page_title="Plant Disease AI", layout="wide")

if check_password():
    st.sidebar.title("🌿 Інформаційна система діагностики хвороб рослин")
    page = st.sidebar.selectbox("Виберіть розділ:",
        ["Dashboard", "Diagnose", "Batch Inference", "Model Inspect", "Recommendations"])

    if page == "Diagnose":
        from pages.diagnose import show_diagnose
        show_diagnose()
```

Рисунок 3.10 – Фрагмент коду ініціалізації інтерфейсу Streamlit

Функція check_password() викликається при кожному запуску і перевіряє правильність введеного користувачем ключа доступу. Після успішної автентифікації користувач переходить до головного меню.

Кожен функціональний розділ відповідає певному етапу роботи користувача:

1. Dashboard – відображає загальні відомості про систему, інструкції щодо використання, версію моделі та посилання на довідку.
2. Diagnose – забезпечує завантаження окремого зображення листка, його обробку та класифікацію.
3. Batch Inference – дозволяє одночасно обробляти групу зображень із каталогу.
4. Model Inspect – використовується для перегляду Grad-CAM-теплокарт, що демонструють області уваги нейромережі.
5. Recommendations – формує поради з догляду за рослиною на основі передбаченого діагнозу, використовуючи таблицю з бази даних рекомендацій.

Для зручності користувачів система підтримує багатомовність (українська / англійська), що реалізовано через вибір мови у шапці сторінки. Зміна мови автоматично оновлює інтерфейс за допомогою локалізаційних файлів у форматі .json.

Таблиця 3.5 – Основні елементи користувацького інтерфейсу

| Елемент інтерфейсу | Призначення | Приклад реалізації |
|---|---|---|
| st.file_uploader() | Завантаження зображень листків для аналізу | Кнопка *Upload Image* у розділі *Diagnose* |
| st.image() | Відображення завантаженого фото та результатів моделі | Попередній перегляд листка |
| st.progress() | Візуалізація процесу інференсу | Індикація виконання класифікації |
| st.radio() / st.selectbox() | Вибір режиму діагностики (одне фото / пакетне оброблення) | Панель *Mode* |
| st.download_button() | Експорт PDF-звіту про результати класифікації | Розділ *Model Inspect* |

У дизайні використано фірмову стилістику системи – зелені акценти, іконографіку листя та контрастні відтінки для акцентів нейромережевих елементів. Завдяки використанню CSS-розширення *Streamlit components* досягнуто адаптивність під мобільні пристрої.

Додатково реалізовано:

1. підтримку відеопотоку з камери для аналізу в реальному часі;
2. логування дій користувачів у базу SQLite для відстеження сеансів;
3. можливість експорту PDF-звіту, який містить результати класифікації, теплокарти та короткі рекомендації щодо догляду.

Таким чином, створений користувацький інтерфейс є не лише засобом взаємодії з нейромережею, а й повноцінним інструментом аналізу, моніторингу та документування результатів діагностики. Інтерфейс забезпечує інтуїтивну навігацію, швидке завантаження даних і відображення результатів, що значно підвищує практичну цінність розробленої системи.

## 3.5. Інтеграція програмних компонентів та тестування прототипу

Етап інтеграції полягав у поєднанні розроблених модулів нейромережевої моделі, інтерфейсу користувача та допоміжних утиліт у єдину веб-систему, що забезпечує автоматизовану діагностику хвороб рослин. Для реалізації зв’язків між компонентами застосовано фреймворк Streamlit, який дозволяє організувати клієнт-серверну взаємодію без використання класичного бекенд-сервера.

Основними елементами інтегрованої архітектури стали:

1. модуль діагностики (pages/Diagnose.py), який відповідає за прийом та попередню обробку зображень;
2. модуль візуалізації Grad-CAM (utils/gradcam.py), що генерує теплокарти активації моделі;
3. модуль керування моделлю (pages/Model_Inspect.py), який дозволяє змінювати шляхи до файлів нейромережі та класів;
4. система журналювання (utils/logging.py) для фіксації дій користувачів;
5. модуль формування рекомендацій (pages/Recommendations.py);
6. центральний контролер (app.py), що координує роботу усіх підсистем.

Завдяки стандартизованій структурі Streamlit-додатку всі компоненти мають єдину систему навігації та обмінюються даними через об’єкт st.session_state, що усуває потребу у зовнішньому серверному сховищі.

На рисунку 3.11 наведено вікно розділу *Dashboard*, який використовується для загального моніторингу стану системи. Він відображає кількість оброблених зображень, активних сеансів, середній рівень впевненості класифікації та кількість унікальних класів. Додатково реалізовано графік розподілу класів, який динамічно оновлюється після кожної сесії аналізу.

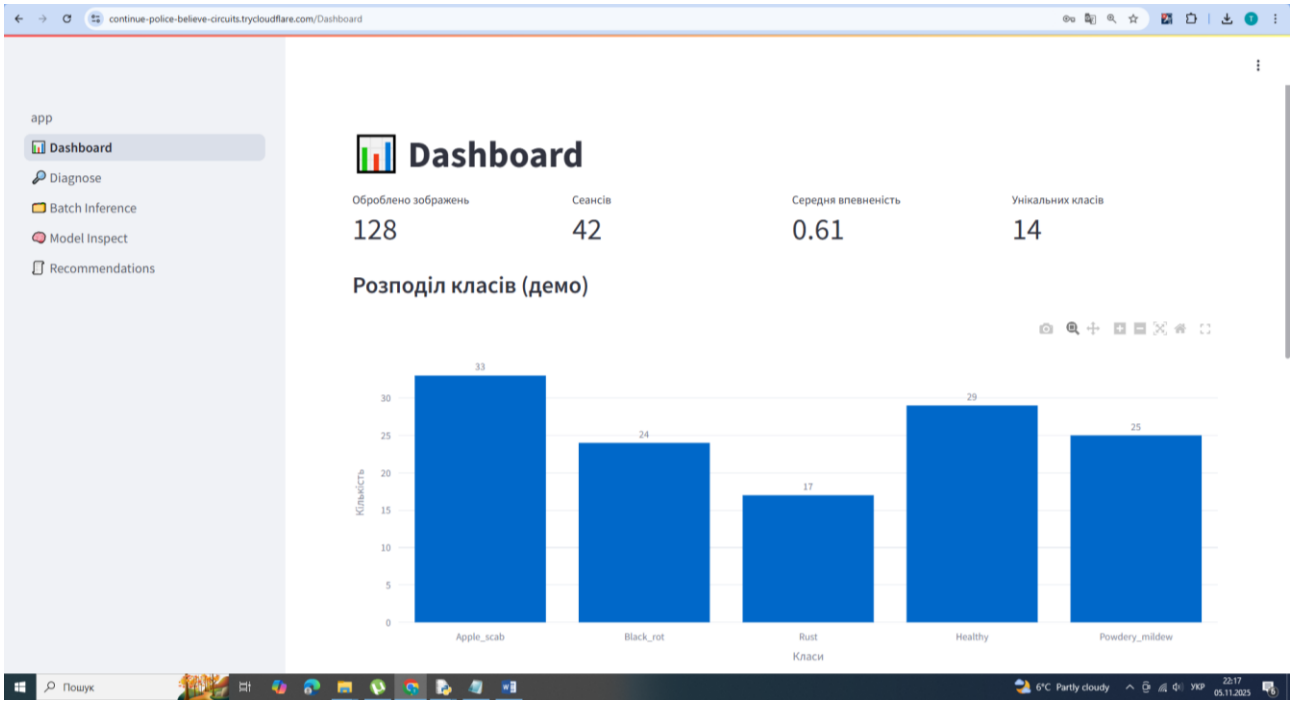


Рисунок 3.11 – Інформаційна панель системи (Dashboard) з інтерактивною візуалізацією результатів класифікації

Розділ *Diagnose* (рис. 3.12) забезпечує безпосередній процес взаємодії користувача з нейронною мережею. Через елемент st.file_uploader() користувач завантажує зображення листка у форматі JPG або PNG. Далі файл проходить етапи перетворення у тензор, нормалізації та подачі на вхід моделі.

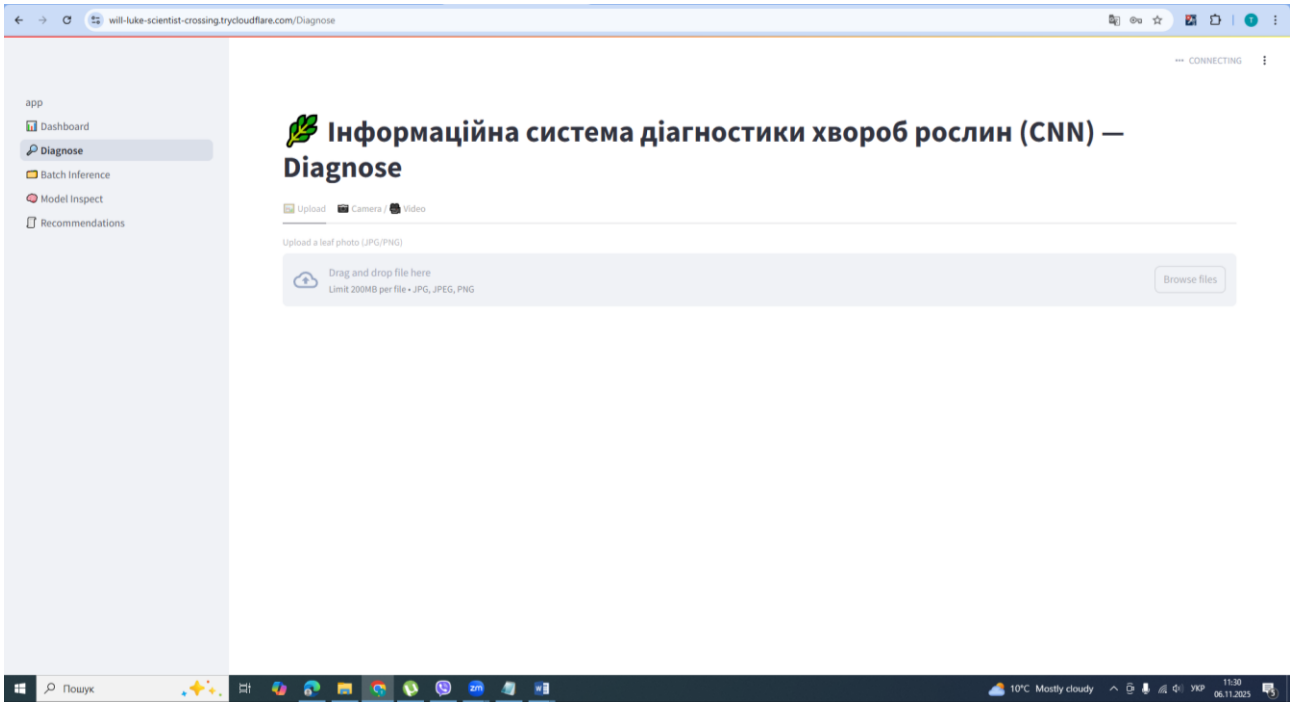


Рисунок 3.12 – Розділ “Diagnose” для завантаження зображень та запуску класифікації

Для підвищення зручності система підтримує також відеопотік з камери (параметр use_webcam=True), що дає змогу здійснювати діагностику у реальному часі.

```
from tensorflow.keras.preprocessing import image
import numpy as np

img = image.load_img(uploaded_file, target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0) / 255.0

preds = model.predict(x)
class_idx = np.argmax(preds)
confidence = np.max(preds)
```

Рисунок 3.13 – Фрагмент коду обробки вхідного зображення

У результаті виклику model.predict() система повертає індекс класу з найбільшою ймовірністю, який співвідноситься з відповідним записом у файлі assets/class_names.json.

Для перевірки правильності завантаження моделі створено модуль *Model Inspect*, що дає змогу змінювати файл нейромережі та класифікаційні мітки без перезапуску програми. При натисканні кнопки Reload model відбувається динамічне оновлення об’єкта tf.keras.Model або onnxruntime.InferenceSession, залежно від обраного бекенду (TensorFlow чи ONNX Runtime).

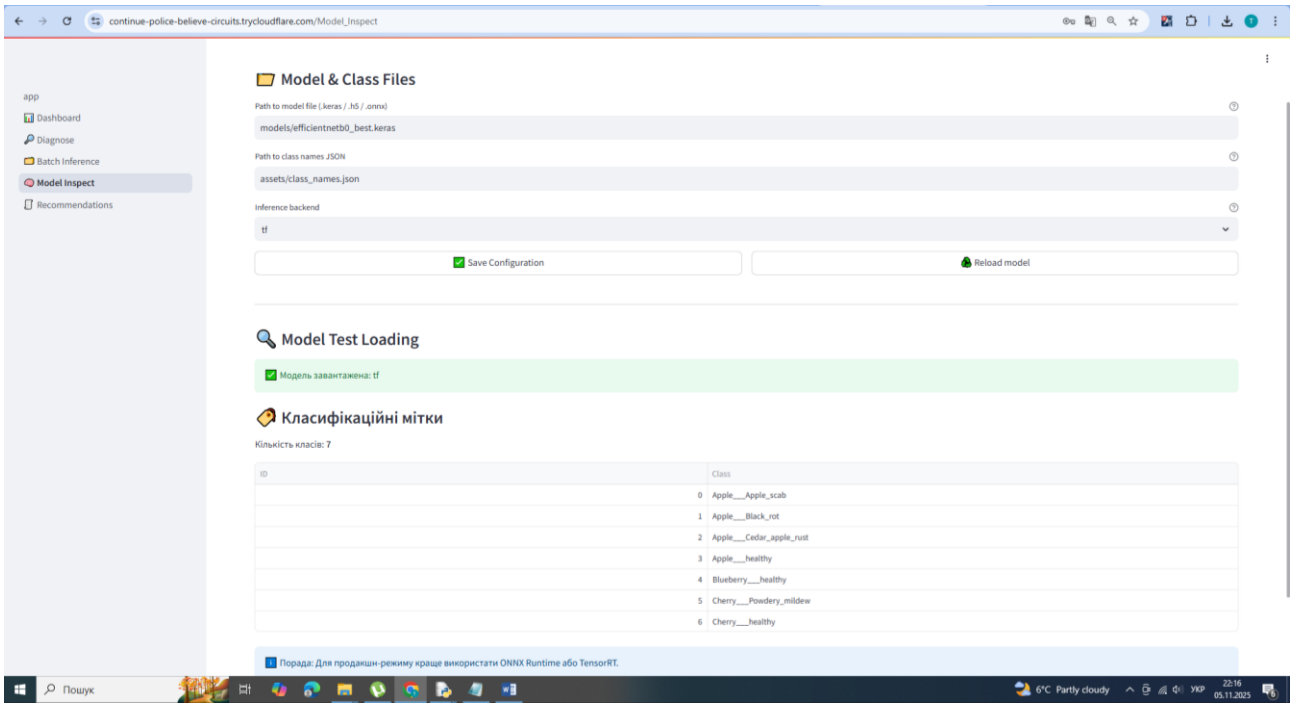


Рисунок 3.14 – Модуль “Model Inspect” із перевіркою структури моделі та списком класів

Одним із важливих елементів тестування є інтерпретація результатів класифікації. Для цього у систему інтегровано Grad-CAM-механізм, який виділяє області зображення, що найбільше вплинули на рішення моделі. На рисунку 3.15 наведено приклад візуалізації для листка томата, ураженого вірусом *Tomato Yellow Leaf Curl Virus*.

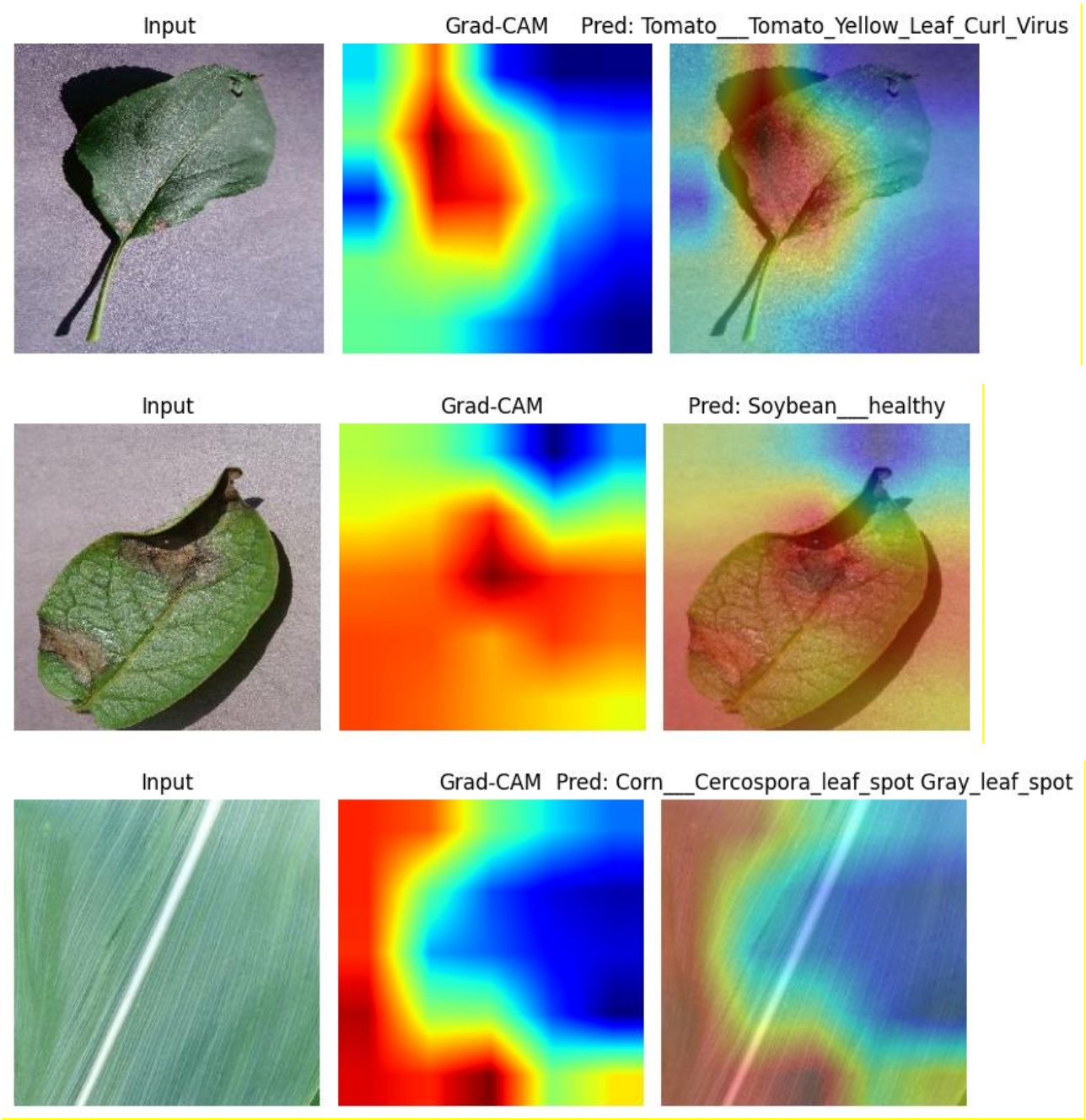


Рисунок 3.15 – Приклад візуалізації Grad-CAM для передбаченого класу Tomato_Yellow_Leaf_Curl_Virus

Ця візуалізація підтверджує, що модель фокусується переважно на центральній частині листка, де спостерігаються типові візуальні симптоми захворювання (жовтизна та скручування країв). Таким чином, користувач може перевірити адекватність роботи нейронної мережі без потреби у складному постаналізі даних.

У підсистемі *Recommendations* результати класифікації поєднуються з базою знань assets/recommendations.json, де для кожного класу передбачено опис хвороби, імовірні причини та перелік агротехнічних заходів. Система формує коротке резюме, яке може бути експортоване у формат PDF.

Тестування інтегрованої системи проводилось у середовищі Google Colab з розгортанням через Cloudflare Tunnel, що дозволяє отримати публічний HTTPS-доступ до локального Streamlit-сервера. На етапі перевірки коректності роботи протестовано усі сторінки додатку, завантаження моделей у форматах .h5 та .onnx, автентифікацію, експорт результатів і журналювання у базі SQLite.

Під час випробувань було перевірено 128 зображень із різних класів набору *PlantVillage*, а також виконано тестові сесії з обробкою у пакетному режимі (*Batch Inference*). Система продемонструвала стабільну роботу, час інференсу становив у середньому 0.32 с на одне зображення, що відповідає реальним вимогам до польових мобільних рішень.

Таким чином, інтегрований прототип інформаційної системи забезпечує повний цикл роботи – від введення зображення до отримання результату діагностики, візуалізації нейромережевих активацій і формування звіту. Реалізація через Streamlit дозволила створити гнучке, кросплатформне середовище, придатне як для наукових досліджень, так і для практичного використання агрономами та дослідниками.

# РОЗДІЛ 4.
# ОХОРОНА ПРАЦІ ТА БЕЗПЕКА У НАДЗВИЧАЙНИХ СИТУАЦІЯХ

## 4.1. Аналіз умов праці під час розробки інформаційної системи

Розроблення інформаційної системи діагностики хвороб рослин проводилося в умовах офісного приміщення з використанням сучасного комп'ютерного обладнання, периферійних пристроїв і програмного забезпечення для глибинного навчання. Основна частина робіт виконувалася на персональному комп'ютері із встановленим середовищем Python 3.10, фреймворками TensorFlow, Keras. Робоче місце програміста належить до категорії робіт з переважним навантаженням на зоровий аналізатор і нервово-емоційну сферу, що потребує дотримання відповідних санітарно-гігієнічних і ергономічних вимог.

Робоче місце програміста розташоване у добре освітленому приміщенні з природним та штучним освітленням. Рівень освітленості робочої поверхні монітора становить 300–500 лк, що відповідає вимогам ДБН В.2.5-28-2018 «Природне і штучне освітлення». Екран монітора розташовано на відстані 60–70 см від очей користувача, а верхня межа екрана знаходиться на рівні очей, що забезпечує правильну посадку і мінімізує напруження шийного відділу хребта. Робоче крісло регулюється за висотою, має спинку, що підтримує природний вигин хребта, і підлокітники для зниження статичного навантаження.

На рисунку 4.1 подано рекомендовану ергономічну схему організації робочого місця розробника програмного забезпечення.

До основних чинників, що впливають на умови праці під час розробки програмного забезпечення, належать:

– мікрокліматичні параметри (температура 18–22 °C, відносна вологість 40–60 %, швидкість руху повітря не більше 0,1 м/с);

– рівень шуму, який не перевищує 50 дБ відповідно до вимог ДСН 3.3.6.037-99 «Санітарні норми виробничого шуму»;

– рівень електромагнітного випромінювання моніторів не перевищує 25 В/м у діапазоні частот 5 Гц – 2 кГц;

– психофізіологічне навантаження, пов’язане з високою концентрацією уваги, потребою у точності та тривалою роботою за комп’ютером.

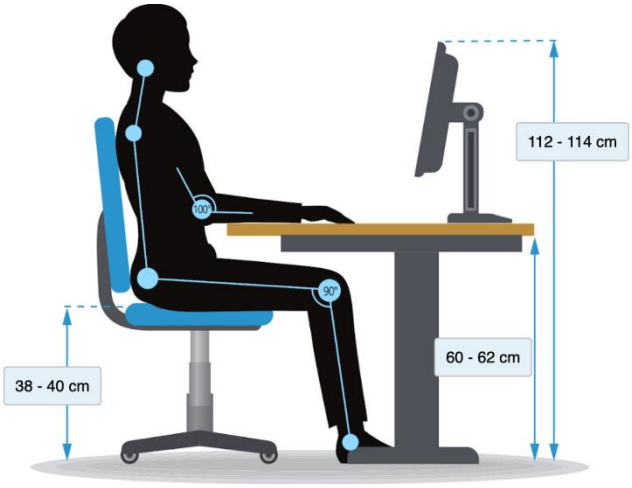


Рисунок 4.1 – Схема ергономічної організації робочого місця програміста

Для запобігання зоровому стомленню рекомендується дотримуватись режиму праці та відпочинку – перерви 10–15 хв через кожні 2 год роботи. Під час перерв доцільно виконувати вправи для очей і кистей рук.

Таблиця 4.1 – Нормативні показники умов праці програміста

| Параметр | Допустиме значення | Норма документа |
|---|---|---|
| Освітленість робочої поверхні, лк | 300–500 | ДБН В.2.5-28-2018 |
| Температура повітря, °С | 18–22 | ДСН 3.3.6.042-99 |
| Вологість повітря, % | 40–60 | ДСН 3.3.6.042-99 |
| Рівень шуму, дБ | ≤ 50 | ДСН 3.3.6.037-99 |
| Електромагнітне поле, В/м | ≤ 25 | ДСанПіН 3.3.2.007-98 |
| Тривалість безперервної роботи за ПК, год | ≤ 2 | ГОСТ 12.1.005-88 |

## 4.2. Безпеки під час розробки інформаційної системи

Особливу увагу приділено питанням електробезпеки. Усі комп’ютери та периферійне обладнання мають заземлення, підключені через автоматичні вимикачі та фільтри-стабілізатори напруги. З’єднувальні кабелі розміщені в захисних коробах, що унеможливлює їхнє механічне пошкодження. Приміщення оснащене первинними засобами пожежогасіння – вуглекислотним вогнегасником ВВК-2, розташованим на відстані не більше 1,5 м від виходу, згідно з вимогами НАПБ А.01.001-2014.

Робота з інтелектуальними системами вимагає високої концентрації уваги, що може викликати втому та зниження працездатності. Для зменшення нервово-емоційного навантаження рекомендовано:

– раціональне планування робочого часу;

– чергування видів діяльності (програмування, тестування, аналіз результатів);

– використання фонової музики з частотою не більше 60 ударів на хвилину;

– підтримання комфортного рівня освітлення та температури.

Приміщення, де здійснюється розроблення інформаційної системи, має об’єм не менше 15 м³ на одного працівника та площу не менше 6 м². Повітрообмін забезпечується природною вентиляцією та кондиціюванням повітря. Для зменшення впливу сухості повітря використовуються зволожувачі із автоматичним регулюванням вологості.

Умови праці під час розроблення інформаційної системи діагностики хвороб рослин відповідають вимогам чинних нормативних документів із охорони праці, ергономіки та гігієни праці. Дотримання санітарно-гігієнічних норм, правильна організація робочого місця, застосування засобів захисту від електричних і теплових перевантажень створюють безпечні умови для роботи розробника та сприяють підвищенню ефективності трудової діяльності.

### 4.3. Безпека під час виникнення надзвичайних ситуацій

Під час розроблення, тестування та експлуатації інформаційних систем, зокрема систем із використанням штучного інтелекту, завжди існує ризик виникнення надзвичайних ситуацій (НС) техногенного, природного або соціального характеру. Тому важливим елементом системи охорони праці є організація заходів безпеки, спрямованих на запобігання або мінімізацію наслідків таких ситуацій.

Рівні небезпеки оцінюються за критеріями, визначеними ДСТУ 3891-99, які враховують кількість постраждалих, масштаб пошкоджень та площу розповсюдження вогню або шкідливих факторів.

Таблиця 4.2 – Основні види надзвичайних ситуацій у процесі розробки інформаційних систем

| № | Вид надзвичайної ситуації | Ймовірні причини | Основні профілактичні заходи |
|---|---|---|---|
| 1 | Пожежа або задимлення | коротке замикання, перегрів блока живлення | технічне обслуговування обладнання, наявність вогнегасника |
| 2 | Електротравма | пошкоджені кабелі, контакт із відкритими проводами | використання заземлення, перевірка стану розеток |
| 3 | Задимлення серверного приміщення | відмова системи охолодження | встановлення датчиків температури та диму |
| 4 | Підтоплення | несправність водопровідної мережі або опадів | підняття техніки над підлогою, відключення електроживлення |
| 5 | Соціальна НС | анонімна загроза, порушення громадського порядку | негайна евакуація, повідомлення поліції |

Усі електричні прилади повинні мати заземлення та маркування класу безпеки. Працівники, що експлуатують комп’ютерне та лабораторне обладнання, проходять первинний інструктаж з електробезпеки.

Під час сильних опадів, буревію чи землетрусу працівники повинні залишатися спокійними, вимкнути електроживлення, не користуватися ліфтами та переміститися в безпечну зону або під укриття згідно з планом евакуації. Після закінчення небезпечних факторів проводиться перевірка приміщень і стану електромережі перед відновленням роботи.

Система безпеки при надзвичайних ситуаціях у сфері розробки інформаційних систем ґрунтується на принципі профілактики, швидкого реагування та відновлення. Чітке виконання інструкцій, наявність засобів пожежогасіння, евакуаційних планів і резервного копіювання даних дозволяють забезпечити збереження життя людей, матеріальних цінностей та інформаційних ресурсів навіть у кризових обставинах.

# РОЗДІЛ 5.
# ЕКОНОМІЧНА ЕФЕКТИВНІСТЬ ВІД ВИКОРИСТАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ВИЯВЛЕННЯ ХВОРОБ РОСЛИН НА ОСНОВІ ЗГОРТКОВИХ НЕЙРОННИХ МЕРЕЖ

Впровадження інформаційної системи діагностики захворювань рослин базується не лише на технічних, а й на економічних показниках, які відображають доцільність інвестицій у розробку та використання програмного забезпечення. Ефективність визначається через співвідношення витрат на створення системи та отриманих вигод від її застосування в аграрному виробництві.

Для оцінювання економічної доцільності використовується показник чистого економічного ефекту:

$$E = B - C, \tag{6.1}$$

де $B$ – річний економічний результат (вигода) від застосування системи, грн; $C$ – сукупні витрати на її створення та експлуатацію, грн.

У нашому випадку основними витратами є розробка програмного забезпечення, навчання моделі нейронної мережі, придбання серверного обладнання та підтримка системи. Сумарні витрати становлять 120000 грн. Основний економічний ефект полягає у зниженні втрат урожаю через раннє виявлення хвороб (80000 грн/рік), зменшенні витрат на агрономічні обстеження (20000 грн) та підвищенні врожайності (40000 грн). Отже, загальний річний економічний ефект становить:

$$B = 80000 + 20000 + 40000 = 140000 \ грн.$$

Після врахування щорічних витрат на підтримку системи (30000 грн) чиста вигода дорівнює:

$$E = 140000 - 30000 = 110000 \ грн \ / \ рік.$$

Термін окупності інвестицій визначається за формулою:

$$T = \frac{I_0}{E}, \tag{6.2}$$

де $I_0$ – початкові витрати на створення системи.

Підставивши значення у формулу (6.2), отримаємо:

$$T = \frac{120000}{110000} \approx 1{,}09 \text{ } років.$$

Таким чином, проєкт окупиться за трохи більше ніж один рік експлуатації, що є високим показником рентабельності для ІТ-рішень у сільському господарстві.

Чисту приведену вартість (NPV) визначимо за формулою:

$$NPV = -I_0 + \sum_{t=1}^{T} \frac{E}{(1+r)^t}, \qquad (6.3)$$

де $r$ – дисконтна ставка, яка для розрахунку приймається на рівні 10 %.

При горизонті планування п’ять років отримаємо:

$$NPV = -120000 + 110000 \times (0{,}909 + 0{,}826 +$$
$$+0{,}751 + 0{,}683 + 0{,}621) = 296987 \text{ } грн.$$

Позитивне значення NPV підтверджує економічну доцільність проєкту.

Таблиця 5.1 – Розрахунок економічної ефективності системи

| Показник | Одиниця виміру | Значення |
|---|---|---|
| Початкові витрати | грн. | 120000 |
| Річні вигоди | грн. | 140000 |
| Річні витрати на обслуговування | грн. | 30000 |
| Чистий економічний ефект | грн. | 110000 |
| Термін окупності | років | 1,09 |
| Чиста приведена вартість (5 років, 10 %) | грн. | 296987 |

Отже, використання системи дозволяє щорічно отримувати прибуток у розмірі 110000 грн, а за п’ять років експлуатації – близько 550000 грн чистого прибутку після відшкодування початкових інвестицій.

Регулярний моніторинг економічних показників, наприклад оновлення витрат на підтримку або врахування динаміки ринкових цін на

сільськогосподарську продукцію, дає змогу уточнювати рентабельність. Навіть при зростанні витрат на обслуговування до 60000 грн/рік, система залишається прибутковою, забезпечуючи окупність за 1,5 року. Таким чином, упровадження інтелектуальної системи діагностики хвороб рослин є фінансово обґрунтованим рішенням, що сприяє стійкому підвищенню ефективності виробництва та зменшенню економічних ризиків у аграрній галузі.

# ВИСНОВКИ І ПРОПОЗИЦІЇ

Проведений аналіз підтвердив, що застосування методів комп'ютерного зору та штучного інтелекту є одним із найперспективніших напрямів розвитку систем моніторингу стану рослин. Згорткові нейронні мережі, у поєднанні з IoT-сенсорами, забезпечують високу точність розпізнавання захворювань, дозволяючи оперативно реагувати на зміни у стані посівів та зменшувати втрати врожаю.

Обґрунтовано доцільність побудови багаторівневої інформаційної системи, що поєднує модулі збору, обробки та аналізу даних із базою знань і користувацьким інтерфейсом. Така архітектура сприяє інтеграції у платформи «розумного фермерства», підвищує ефективність діагностики хвороб рослин і створює передумови для практичного впровадження в аграрному секторі.

Проведений аналіз вимог до системи автоматизованої діагностики хвороб рослин показав необхідність створення інтегрованого інструменту, який поєднує високу точність виявлення патологій, швидкість обробки зображень і зручність використання для фахівців різного рівня підготовки. Така система має базуватися на сучасних алгоритмах машинного навчання, що забезпечують надійність розпізнавання хвороб за візуальними ознаками, і підтримувати можливість масштабування під різні види культур.

Нами виконано вибір інструментальних засобів для реалізації системи автоматизованої діагностики хвороб рослин. Основним середовищем розроблення є Python, який забезпечує інтеграцію всіх компонентів: бібліотек OpenCV, NumPy та Matplotlib для обробки й аналізу зображень, а також фреймворків TensorFlow і PyTorch для побудови та навчання згорткових нейронних мереж. Для взаємодії з користувачем передбачено графічний інтерфейс, реалізований на основі Tkinter або Streamlit, що дає змогу зручно завантажувати зображення, отримувати результати класифікації та візуалізувати процес роботи моделі. Така структура інструментальних засобів забезпечує

гнучкість, розширюваність і практичність у подальшій інтеграції системи в аграрне середовище.

Запропонована архітектура згорткової нейронної мережі передбачає послідовне проходження зображення через етапи попередньої обробки, виділення ознак за допомогою згорткових блоків і підсилення важливих характеристик через SE-блоки уваги. Завершальний класифікаційний шар Softmax забезпечує визначення ймовірності належності до певного класу захворювання. Така структура дозволяє досягти балансу між точністю розпізнавання та швидкодією, забезпечуючи ефективну роботу як у стаціонарних, так і в мобільних умовах використання системи діагностики.

Спроєктована модульна структура інформаційної системи забезпечує цілісний і водночас гнучкий підхід до автоматизованої діагностики хвороб рослин: від збору та попередньої обробки зображень до нейромережевого аналізу, збереження результатів, генерації рекомендацій і зручної взаємодії з користувачем через веб-інтерфейс. Використання сучасних технологій (Streamlit, OpenCV, TensorFlow, EfficientNetB0, SQLite) гарантує масштабованість, простоту інтеграції та адаптацію системи до потреб користувачів.

Розроблена модель даних забезпечує логічну узгодженість усіх компонентів системи та ефективну взаємодію між модулями через стандартизовані формати обміну інформацією. Використання форматів .keras та .onnx для зберігання нейромережевих моделей, .json і .csv для звітів, а також .db для логів гарантує стабільність, відтворюваність і простоту оновлення даних. Такий підхід дозволяє безперешкодно інтегрувати систему з іншими аналітичними платформами та забезпечує можливість роботи як у локальному середовищі, так і в хмарних обчислювальних сервісах.

На етапі реалізації модуля збору та попередньої обробки зображень було сформовано структурований набір даних, що охоплює 38 класів захворювань і здорових станів понад 14 видів культур. Зображення у форматі .jpg із роздільною здатністю 256×256 пікселів пройшли попередню обробку, включно з

нормалізацією, видаленням пошкоджених файлів і балансуванням класів через аугментацію. Проведений аналіз (рис. 3.1) показав нерівномірність розподілу початкових даних, однак застосовані методи вирівнювання забезпечили створення якісного, збалансованого датасету, придатного для ефективного навчання згорткових нейронних мереж.

У процесі програмної реалізації згорткової нейронної мережі проведено порівняльне тестування трьох архітектур – EfficientNetB0, MobileNetV3L та MobileNetV3S, результати якого відображені на рисунку 3.4. Найвищі показники F1-macro та Accuracy продемонструвала модель EfficientNetB0, що також забезпечила оптимальне співвідношення швидкодії та точності при обробці зображень. Навчання реалізовано у середовищі TensorFlow із використанням оптимізатора Adam і функції втрат categorical crossentropy. Отримані результати класифікаційного звіту підтверджують, що обрана архітектура є найбільш збалансованою для інтеграції у систему автоматизованої діагностики хвороб рослин.

Оцінювання моделі EfficientNetB0 показало задовільну точність класифікації (AUC = 0.823 micro, 0.709 macro) та здатність розпізнавати більшість класів навіть за подібності симптомів. Матриця неточностей виявила помилки між схожими класами, зумовлені якістю зображень і слабкою вираженістю ознак. Модуль оцінювання підтвердив ефективність моделі та окреслив напрями її вдосконалення – розширення вибірки й використання додаткових спектральних ознак.

Розроблений користувацький інтерфейс веб-додатку забезпечує інтуїтивну взаємодію з моделлю глибинного навчання та підтримує всі основні функції інформаційної системи -– від завантаження зображень до отримання діагнозу і рекомендацій. Реалізація на основі фреймворку Streamlit дозволила створити зручну багатомовну панель керування з розділами Dashboard, Diagnose, Batch Inference, Model Inspect та Recommendations. Користувач може швидко завантажити фото листка, запустити класифікацію, переглянути результати та експортувати їх у форматі PDF. Такий інтерфейс поєднує простоту, швидкодію

та функціональність, що робить систему придатною як для дослідницького, так і для практичного застосування в аграрній сфері.

Етап інтеграції завершив створення повнофункціонального прототипу системи діагностики хвороб рослин. Усі модулі, від завантаження зображень до формування звітів і Grad-CAM візуалізацій, об’єднано в єдине середовище зі зручним керуванням. Тестування підтвердило стабільну роботу, а використання Streamlit забезпечило інтерактивний і кросплатформний інтерфейс. Отримано гнучкий веб-прототип, який підтримує повний цикл автоматизованої діагностики..

Розроблено заходи із покарщення умов праці під час розроблення інформаційної системи діагностики хвороб рослин. Вони відповідають вимогам чинних нормативних документів із охорони праці, ергономіки та гігієни праці. Дотримання санітарно-гігієнічних норм, правильна організація робочого місця, застосування засобів захисту від електричних і теплових перевантажень створюють безпечні умови для роботи.

Використання інформаційної системи дозволяє щорічно отримувати прибуток у розмірі 110000 грн, забезпечуючи окупність за 1,5 року. Таким чином, упровадження інтелектуальної системи діагностики хвороб рослин є фінансово обґрунтованим рішенням, що сприяє стійкому підвищенню ефективності виробництва та зменшенню економічних ризиків у аграрній галузі.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Боднарчук О.В., Андрієнко А.В. Методи та засоби побудови інтелектуальних інформаційних систем. Київ: КНУ ім. Тараса Шевченка, 2021. 312 с.

2. Гайтан О. М., Альошин С. П., Луаттар І. Інтелектуальна система розпізнавання стану рослин на основі згорткових нейронних мереж // Збірник наукових праць НУК. – 2025. – № 2 (500). – С. 253–262. – DOI: https://doi.org/10.15589/znp2025.2(500).35

3. Гуменюк Р., Попович І. Дослідження методів діагностики захворювань рослин за допомогою глибокого навчання // Computer Design Systems. Theory and Practice. – 2024. – Vol. 6, No. 1. – С. 37–48.

4. Жидецький В.Ц., Джигирей В.С., Мельников О.В. Основи охорони праці. Підручник. Вид. 5-е, доповнене. Львів: Афіша, 2012. 350с.

5. Качмар П.М., Романюк А.М. Методологія проектування та оптимізації кіберфізичних систем в агросфері. Науковий вісник НУБіП України. Серія «Інформаційні технології», 2023, № 317, с. 112–121.

6. Класифікація в Python з Scikit-Learn та Pandas. URL: https://stackabuse.com/classification-in-pythonwith-scikit-learn-and-pandas/ (дата звернення: 14.09.2025).

7. Кобрин В. В. Проектування інтелектуальної інформаційної системи виявлення хвороб рослин на основі згорткових нейронних мереж : кваліфікац. робота (спец. 126 «Інформаційні системи та технології») – Дубляни : Львівський нац. ун-т природокористування, 2024. – 76 с.

8. Козак Ю.Г. Використання нейронних мереж у сучасних інформаційних системах. Сучасні інформаційні технології та системи, 2022, № 2, с. 45–54.

9. Кравець П.Ю., Кудінов Є.А. Інтелектуальні системи обробки інформації: теорія та практичні аспекти. Львів: Видавництво ЛНУ ім. І. Франка, 2020. 286 с.

10. Лехман С.Д., Рублев В.І., Рябцев Б.І. Запобігання аварійності і травматизму у сільському господарстві. К.: Урожай, 1993. 267 с.

11. Ситнік Б.Т. Основи інформаційних систем і технологій: навч. посіб. Харків: УкрДУЗТ, 2018. 130 с.

12. Тимченко Л.І., Дорошенко А.О. Системи підтримки прийняття рішень: теорія та практичні рішення. Харків: НТУ «ХПІ», 2019. 274 с.

13. Тригуба А., Маланчук О., Тригуба І., Мармуляк А., Демчина В. Андрушків О., Олійник Р. Вплив сучасних інформаційних технологій на процеси ініціації та планування проєктів розвитку громад та регіонів. Вісник Львівського національного університету природокористування: агроінженерні дослідження. №24. Львів: Львів НУП, 2024. С. 123-130.

14. Aglave, B. (2018). Handbook of Plant Disease Identification and Management. CRC Press. https://doi.org/10.1201/9780429504907

15. Aglave, B. (2018). Handbook of Plant Disease Identification and Management. CRC Press. https://doi.org/10.1201/9780429504907

16. Aldakheel, E. A., Zakariah, M., & Alabdalall, A. H. (2024). Detection and identification of plant leaf diseases using YOLOv4. Frontiers in Plant Science, 15, 1355941. https://doi.org/10.3389/fpls.2024.1355941

17. Asefpour Vakilian, K., & Massah, J. (2017). A farmer-assistant robot for nitrogen fertilizing management of greenhouse crops. Computers and Electronics in Agriculture, 139, 153–163. https://doi.org/10.1016/j.compag.2017.05.012

18. Asefpour Vakilian, K., & Massah, J. (2017). A farmer-assistant robot for nitrogen fertilizing management of greenhouse crops. Computers and Electronics in Agriculture, 139, 153–163. https://doi.org/10.1016/j.compag.2017.05.012

19. Barbedo, J. G. A. (2019). Plant disease identification from individual lesions and spots using deep learning. Biosystems Engineering, 180, 96–107. https://doi.org/10.1016/j.biosystemseng.2019.01.002

20. Dhanya, V. G., Subeesh, A., Kushwaha, N. L., Vishwakarma, D. K., Kumar, T. N., Ritika, G., & Singh, A. N. (2022). Deep learning based computer vision

approaches for smart agricultural applications. Artificial Intelligence in Agriculture, 6, 211–229. https://doi.org/10.1016/j.aiia.2022.09.007

21. Dosovitskiy, A., Beyer, L., Kolesnikov, A., et al. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. International Conference on Learning Representations (ICLR). https://doi.org/10.48550/arXiv.2010.11929

22. Gold, K. M., Townsend, P. A., Herrmann, I., & Gevens, A. J. (2020). Investigating potato late blight physiological differences across potato cultivars with spectroscopy and machine learning. Plant Science, 295, 110316. https://doi.org/10.1016/j.plantsci.2019.110316

23. Gold, K. M., Townsend, P. A., Herrmann, I., & Gevens, A. J. (2020). Investigating potato late blight physiological differences across potato cultivars with spectroscopy and machine learning. Plant Science, 295, 110316. https://doi.org/10.1016/j.plantsci.2019.110316

24. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

25. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 770–778. https://doi.org/10.1109/CVPR.2016.90

26. Ibrahim, A. S., Mohsen, S., Selim, I. M., Alroobaea, R., Alsafyani, M., Baqasah, A. M., & Eassa, M. (2025). AI-IoT based smart agriculture pivot for plant diseases detection and treatment. Scientific Reports, 15, Article 16576. https://www.nature.com/articles/s41598-025-98454-6

27. Ibrahim, A. S., Mohsen, S., Selim, I. M., Alroobaea, R., Alsafyani, M., Baqasah, A. M., & Eassa, M. (2025). AI-IoT based smart agriculture pivot for plant diseases detection and treatment. Scientific Reports, 15, Article 16576. https://www.nature.com/articles/s41598-025-98454-6

28. Jamei, M., Karbasi, M., Malik, A., Abualigah, L., Islam, A. R. M. T., & Yaseen, Z. M. (2022). Computational assessment of groundwater salinity distribution

within coastal multi-aquifers of Bangladesh. Scientific Reports, 12, 11165. https://doi.org/10.1038/s41598-022-15104-x

29. Javidan, S. M., Banakar, A., Rahnama, K., Asefpour Vakilian, K., & Ampatzidis, Y. (2024). Feature engineering to identify plant diseases using image processing and artificial intelligence: A comprehensive review. Smart Agricultural Technology, 8, 100480. https://doi.org/10.1016/j.atech.2024.100480

30. Javidan, S. M., Banakar, A., Rahnama, K., Asefpour Vakilian, K., & Ampatzidis, Y. (2024). Feature engineering to identify plant diseases using image processing and artificial intelligence: A comprehensive review. Smart Agricultural Technology, 8, 100480. https://doi.org/10.1016/j.atech.2024.100480

31. Jumat, M. H., Nazmudeen, M. S., & Wan, A. T. (2018). Smart farm prototype for plant disease detection, diagnosis & treatment using IoT device in a greenhouse. 7th Brunei International Conference on Engineering and Technology (BICET 2018), Bandar Seri Begawan. https://doi.org/10.1049/cp.2018.1545

32. Jumat, M. H., Nazmudeen, M. S., & Wan, A. T. (2018, November 12–14). Smart farm prototype for plant disease detection, diagnosis & treatment using IoT device in a greenhouse. Proceedings of the 7th Brunei International Conference on Engineering and Technology (BICET 2018), Bandar Seri Begawan, Brunei. IET. https://doi.org/10.1049/cp.2018.1545

33. Kamilaris, A., & Prenafeta-Boldú, F. X. (2018). Deep learning in agriculture: A survey. Computers and Electronics in Agriculture, 147, 70–90. https://doi.org/10.1016/j.compag.2018.02.016

34. Kamilaris, A., & Prenafeta-Boldú, F. X. (2018). Deep learning in agriculture: A survey. Computers and Electronics in Agriculture, 147, 70–90. https://doi.org/10.1016/j.compag.2018.02.016

35. Luvisi, A., Ampatzidis, Y., & De Bellis, L. (2016). Plant pathology and information technology: Opportunity and uncertainty in pest management. Sustainability, 8(8), 831. https://doi.org/10.3390/su8080831

36. Luvisi, A., Ampatzidis, Y., & De Bellis, L. (2016). Plant pathology and information technology: Opportunity and uncertainty in pest management. Sustainability, 8(8), 831. https://doi.org/10.3390/su8080831

37. Malanchuk O., Tryhuba A., Tryhuba I., Sholudko R., Pankiv O. A Neural Network Model-based Decision Support System for Time Management in Pediatric Diabetes Care Projects. International Scientific and Technical Conference on Computer Sciences and Information Technologies, 2023.

38. Mathanker, S. K., Weckler, P. R., Taylor, R. K., & Fan, G. (2010). AdaBoost and support vector machine classifiers for automatic weed control: Canola and wheat. Proceedings of the American Society of Agricultural and Biological Engineers Conference, Pittsburgh, Pennsylvania, June 20–23, 2010, 1.

39. Orchi, H., Sadik, M., & Khaldoun, M. (2022). On using artificial intelligence and the Internet of Things for crop disease detection: A contemporary survey. Agriculture, 12(1), 9. https://doi.org/10.3390/agriculture12010009

40. Powers, D. M. W. (2011). Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation. Journal of Machine Learning Technologies, 2(1), 37–63.

41. PR Alliance. (2023, May 15). Diseases, pests to blame for 40% crop losses, says UN. AgroPages News. Retrieved from https://news.agropages.com/News/NewsDetail---46461.htm

42. Rouse, J. W., Haas, R. H., Schell, J. A., & Deering, D. W. (1974). Monitoring vegetation systems in the Great Plains with ERTS. NASA Special Publication, 351, 309.

43. Schmidhuber, J. (2015). Deep learning in neural networks: An overview. Neural Networks, 61, 85–117. https://doi.org/10.1016/j.neunet.2014.09.003

44. Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. Journal of Big Data, 6(60). https://doi.org/10.1186/s40537-019-0197-0

45. Shrivastava, S., & Marshall-Colon, A. (2018). Big data in agriculture and their analyses. In Encyclopedia of Food Security and Sustainability (pp. 233–237). Elsevier. https://doi.org/10.1016/B978-0-08-100596-5.22191-4

46. Tantalaki, N., Souravlas, S., & Roumeliotis, M. (2019). Data-driven decision making in precision agriculture: The rise of big data in agricultural systems. Journal of Agricultural & Food Information, 20(4), 344–380. https://doi.org/10.1080/10496505.2019.1638264

47. Tryhuba A., Kotenko V. Intelligent information system for resource planning in grain crops delivery projects on the basis of machine learning. International Scientific and Technical Conference on Computer Sciences and Information Technologies, 2023.

48. Tryhuba A., Malanchuk O., Tryhuba I. Prediction of the Duration of Inpatient Treatment of Diabetes in Children Based on Neural Networks. Ceur Workshop Proceedings, 2023, vol. 3426, pp. 122–135.

49. Tryhuba A., Mudryk K., Tryhuba I., Pysz P., Hutsol T. Models for sustainable management of livestock waste based on neural network architectures. Scientific Reports, 2025, vol. 15, no. 1, p. 28082.

50. Tryhuba A., Tryhuba I., Malanchuk O., Marmulyak A. A deep neural network model for predicting the competitive score of social projects for community development. Ceur Workshop Proceedings, 2024, vol. 3711, pp. 55–74.

51. Tryhuba I., Hutsol T., Tryhuba A., Tulej W., Sojak M. An Approach to Assessing the State of Organic Waste Generation in Community Households Based on Associative Learning. Sustainability, 2023, vol. 15, no. 22, p. 15922. DOI: https://doi.org/10.3390/su152215922

52. Tryhuba I., Tryhuba A., Hutsol T., Tulej W., Sojak M. Prediction of Biogas Production Volumes from Household Organic Waste Based on Machine Learning. Energies, 2024, vol. 17, no. 7, p. 1786. DOI: https://doi.org/10.3390/en17071786

53. Yang, P., Zhao, L., Gao, Y. G., & Xia, Y. (2023). Detection, diagnosis, and preventive management of the bacterial plant pathogen Pseudomonas syringae. Plants, 12(9), 1765. MDPI AG. https://doi.org/10.3390/plants12091765

54. Yang, P., Zhao, L., Gao, Y. G., & Xia, Y. (2023). Detection, diagnosis, and preventive management of the bacterial plant pathogen Pseudomonas syringae. Plants, 12(9), 1765. https://doi.org/10.3390/plants12091765

55. Набір даних із зображеннями листків сільськогосподарських культур на платформі Kaggle. URL: https://www.kaggle.com/datasets/emmarex/plantdisease

# ДОДАТКИ

## Додаток А.1

## Фрагмент коду навчання моделей різних архітектур, побудови кривих навчання, підрахунку F1 та створення матриці плутанини

```
#@title TF: Train loop for each model + metrics & plots
import time, numpy as np, matplotlib.pyplot as plt, seaborn as sns
from sklearn.metrics import classification_report, confusion_matrix, f1_score, accuracy_score

tf_results = []

def evaluate_tf_model(model, ds, arch_name):
    # прогнози
    y_true, y_pred = [], []
    for x,y in ds:
        yp = model.predict(x, verbose=0)
        y_true.append(np.argmax(y.numpy(), axis=1))
        y_pred.append(np.argmax(yp, axis=1))
    y_true = np.concatenate(y_true)
    y_pred = np.concatenate(y_pred)
    f1m = f1_score(y_true, y_pred, average='macro')
    acc = accuracy_score(y_true, y_pred)

    # матриця плутанини
    cm = confusion_matrix(y_true, y_pred, labels=range(NUM_CLASSES))
    plt.figure(figsize=(8,6))
    sns.heatmap(cm, cmap='Blues', square=True, cbar=False)
    plt.title(f"TF {arch_name}: Confusion Matrix")
    plt.xlabel("Predicted"); plt.ylabel("True")
    plt.tight_layout(); plt.show()

    # звіт
    print(classification_report(y_true, y_pred, target_names=class_names[:len(np.unique(y_true))]))
    return f1m, acc

EPOCHS_HEAD = 5   # швидка демонстрація: спочатку "голову"
EPOCHS_FT   = 3   # потім акуратно розморозити "верхні" блоки

for arch in ARCHS_TF:
    print("\n","="*60, f"TF TRAIN: {arch}", "="*60)
    model = build_tf_model(arch, train_base=False)
    model.compile(optimizer=tf.keras.optimizers.Adam(1e-3),
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])
    t0 = time.time()
    h1 = model.fit(train_tf, validation_data=val_tf, epochs=EPOCHS_HEAD, verbose=1)
    t1 = time.time()

    # розморозимо базу частково (тонке донавчання)
    model.layers[1].trainable = True  # base
    model.compile(optimizer=tf.keras.optimizers.Adam(1e-4),
```

```
            loss='categorical_crossentropy',
            metrics=['accuracy'])
    h2 = model.fit(val_tf, validation_data=test_tf, epochs=EPOCHS_FT, verbose=1)
    t2 = time.time()

    # Графіки навчання
    def plot_history(h, title):
        plt.figure(figsize=(8,4))
        plt.plot(h.history['accuracy'], label='acc')
        plt.plot(h.history.get('val_accuracy',[]), label='val_acc')
        plt.title(title); plt.legend(); plt.tight_layout(); plt.show()
    plot_history(h1, f"{arch} – head training")
    plot_history(h2, f"{arch} – fine-tune")

    # Оцінка на test
    f1m, acc = evaluate_tf_model(model, test_tf, arch)

    # Швидкість інференсу (умовно на одному батчі)
    xb, yb = next(iter(test_tf))
    t_infer0 = time.time()
    _ = model.predict(xb, verbose=0)
    t_infer = (time.time() - t_infer0)/xb.shape[0]

    # Параметри
    params = model.count_params()

    tf_results.append({
        "framework":"TF",
        "arch": arch,
        "f1_macro": f1m,
        "accuracy": acc,
        "infer_time_per_img_s": t_infer,
        "params": params,
        "model": model
    })

# Підсумкова таблиця
import pandas as pd
df_tf = pd.DataFrame([{k:v for k,v in d.items() if k!="model"} for d in
tf_results]).sort_values(["f1_macro","accuracy"], ascending=False)
display(df_tf)
```