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

КВАЛІФІКАЦІЙНА РОБОТА

другого (магістерського) рівня вищої освіти

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

Виконав: студент групи Іт-62

Спеціальності 126 «Інформаційні системи та технології»
(шифр і назва)
Кобрин Володимир Володимирович
(Прізвище та ініціали)

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

Рецензент: к.т.н., доцент Кригуль Р.Є.
(Прізвище та ініціали)

ДУБЛЯНИ-2024

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
«____» _____2024 р.

ЗАВДАННЯ

на кваліфікаційну роботу студенту

Кобрину Володимиру Володимировичу

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

Керівник роботи Тригуба Анатолій Миколайович, професор
затверджені наказом по університету від 12.09.2024 року № 616/к-с.

2. Строк подання студентом роботи 10.12.2024 р.

3. Вихідні дані до роботи: *дані для виявлення хвороб рослин; методика використання технологій Deep Learning.*
4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити)
Вступ.
1. Аналіз стану виявлення хвороб рослин та завдання кваліфікаційної роботи.
2. Особливості виявлення хвороб рослин, вибір методів Deep Learning та підготовка даних.
3. Результати навчання моделі та розробки інтелектуальної інформаційної системи виявлення хвороб рослин.
4. Охорона праці та безпека у надзвичайних ситуаціях.
5. Визначення показників ефективності.
Висновки та пропозиції.
Список використаної літератури.

5. Перелік ілюстраційного матеріалу (з точним зазначенням обов'язкових слайдів): *аналіз стану виявлення хвороб рослин та завдання кваліфікаційної роботи; особливості виявлення хвороб рослин, вибір методів Deep Learning та підготовка даних; результати навчання моделі та розробки інтелектуальної інформаційної системи виявлення хвороб рослин; результати створення вікна користувачів інтелектуальної інформаційної системи виявлення хвороб рослин; результати створення інтелектуальної інформаційної системи виявлення хвороб рослин; економічна ефективність.*

6. Консультанти з розділів:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1, 2, 3, 5	*Татомир А.В., в.о. доцента кафедри інформаційних технологій*		
4	*Городецький І.М., доцент кафедри фізики, інженерної графіки та безпеки виробництва*		

7. Дата видачі завдання 12 вересня 2024 р.

Календарний план

№ з/п	Назва етапів кваліфікаційної роботи	Терміни виконання етапів роботи	Примітка
1	*Написання першого розділу*	*12.09-20.09.24*	
2	*Виконання другого розділу та аркушів ілюстраційного матеріалу до нього*	*21.09-14.10.24*	
3.	*Виконання третього розділу та аркушів ілюстраційного матеріалу до нього*	*15.010-10.11.24*	
4.	*Написання розділу «Охорона праці та безпека у надзвичайних ситуаціях»*	*11.11-20.11.24*	
5.	*Оцінення ефективності запропонованої системи*	*21.11-30.30.24*	
6.	*Завершення оформлення розрахунково-пояснювальної записки та аркушів ілюстраційного матеріалу*	*01-04.12.24*	
7.	*Завершення роботи вцілому*	*05-10.12.24*	

Студент _____ Кобрин В.В.
(підпис)

Керівник роботи _____ Тригуба А.М.
(підпис)

УДК 004.896.2:631.53

Проектування інтелектуальної інформаційної системи виявлення хвороб рослин на основі згорткових нейронних мереж.

Кобрин В.В. Кафедра інформаційних технологій – Дубляни, ЛНУП, 2024.

Кваліфікаційна робота: 74 с. текст. част., 18 рис., 3 табл., 12 арк. ілюстраційного матеріалу, 43 джерела.

Проаналізовано сучасний стан та виявлення хвороб у сільськогосподарських рослин. Наведено передумови та еволюція глибокого навчання. Виконано аналіз технологій машинного навчання для виявлення хвороб у рослин. Проведено аналіз інтелектуальних систем для виявлення хвороб рослин. Обґрунтовано доцільність розробки інтелектуальної інформаційної системи виявлення хвороб рослин із використанням технології Deep Learning.

Подано особливості виявлення хвороб рослин. Здійснено вибір інструментарію Deep Learning. Проведено збір та підготовка даних для створення моделі виявлення хвороб рослин.

Створено модель для виявлення хвороб рослин. Подано результати навчання моделі для виявлення хвороб рослин. Наведено результати створення вікна користувачів інтелектуальної інформаційної системи виявлення хвороб рослин. Подано результати створення інтелектуальної інформаційної системи виявлення хвороб рослин.

Розроблено заходи щодо охорони праці. Розрахована економічна ефективність від розробки інтелектуальної інформаційної системи виявлення хвороб рослин із використанням технології Deep Learning.

ЗМІСТ

ВСТУП

Сучасний аграрний сектор стоїть перед складними викликами, пов’язаними з підтримкою стійкості та продуктивності рослинництва, яке є критичним для забезпечення світових продуктів харчування та ресурсів. Однак, на жаль, рослини можуть стати жертвами різноманітних хвороб, паразитів і стресів, які впливають на їхнє здоров'я та врожайність. Вчасне виявлення та діагностика цих проблем є критичними завданнями для забезпечення стабільного та високого вирощування сільськогосподарських культур.

У цьому контексті актуальним завданням стає розвиток інтелектуальних інформаційних систем, які базуються на передових технологіях. Технологія Deep Learning, що є підгалуззю машинного навчання, створила широку популярність завдяки своїй здатності до автоматичного вивчення складних патернів та роботи з великими обсягами даних. Використання Deep Learning для хвороб рослин відкриває нові можливості для покращення діагностики та контролю над захворюваннями рослин.

Метою даної кваліфікаційної роботи є розробка та дослідження інтелектуальної інформаційної системи, яка здатна автоматично виявляти та класифікувати рослини з використанням технології Deep Learning. Ця система дає можливість сільськогосподарським виробникам та агрономам швидше та ефективніше реагувати на захворювання рослин, що може позитивно вплинути на врожайність та якість сільськогосподарської продукції.

У цій роботі буде детально розглянуто методи та підходи до розробки системи виявлення хвороб рослин з використанням Deep Learning, а також проведено аналіз результатів експериментів і перспективи подальших досліджень у цій області.

Отже, розробка інтелектуальної інформаційної системи виявлення хвороб рослин із використанням технології Deep Learning має практичне значення та забезпечує покращення моніторингу стану посівів із сільськогосподарськими

культурами та забезпечує зменшення втрат від негативної дії шкідників та хвороб.

Об’єктом дослідження алгоритми та моделі Deep Learning, а також посіви із сільськогосподарськми культурами.

Предмет дослідження є вплив стану рослин, поява на них шкідників на параметри моделей Deep Learning для виявлення хвороб рослин.

РОЗДІЛ 1.
АНАЛІЗ СТАНУ ВИЯВЛЕННЯ ХВОРОБ РОСЛИН ТА ЗАВДАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

1.1. Сучасний стан та виявлення хвороб у сільськогосподарських рослин

Глобальна продовольча безпека, яка визначається балансом світового виробництва продуктів харчування та попиту, стала важливою міжнародною проблемою в останні роки [16]. Зростання цін на продукти харчування спричинило глобальну кризу, яка спричинила політичну та економічну нестабільність у деяких країнах, що розвиваються [17]. Було підраховано, що попит на їжу буде продовжувати зростати ще 40 років через постійне збільшення населення. Прогнози також показують, що до 2050 року для задоволення потреб потрібно ще 70% виробництва продуктів харчування [18].

В даний час більше одного мільярда людей страждають від різних ситуацій недоїдання через брак їжі, і приблизно вдвічі більше населення не має доступу до достатньої кількості поживних речовин або вітамінів для задоволення своїх щоденних потреб у харчуванні [19].

Ситуацію можна пояснити постійним скороченням площ сільськогосподарських угідь, що спричиняє зниження продуктивності. Хоча зниження продуктивності сільського господарства можна пояснити різними причинами, пошкодження, спричинені шкідниками та патогенами, відіграють значну роль у втратах врожаю в усьому світі.

Втрати врожаю через патогенні інфекції коливаються від 20% до 40% [20]. У середньому спричинені патогенами втрати кукурудзи, ячменю, рису та сої становлять приблизно 12%, арахісу та картоплі – приблизно 24%, а пшениці та бавовни – приблизно 50% та 80% відповідно [21]. Стан захисту рослин представлений в Україні на рис. 1.1-1.2.

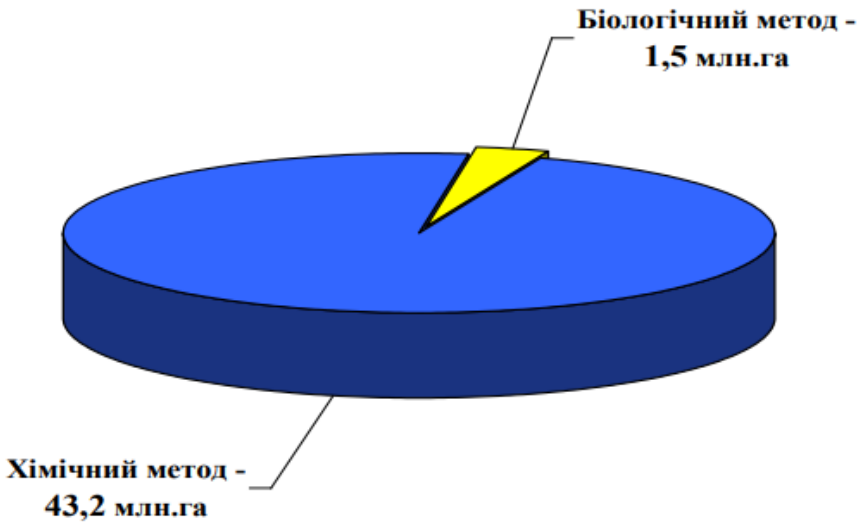

Рисунок 1.1 – Використовувані методи боротьби із хворобами сільськогосподарських культур [23]

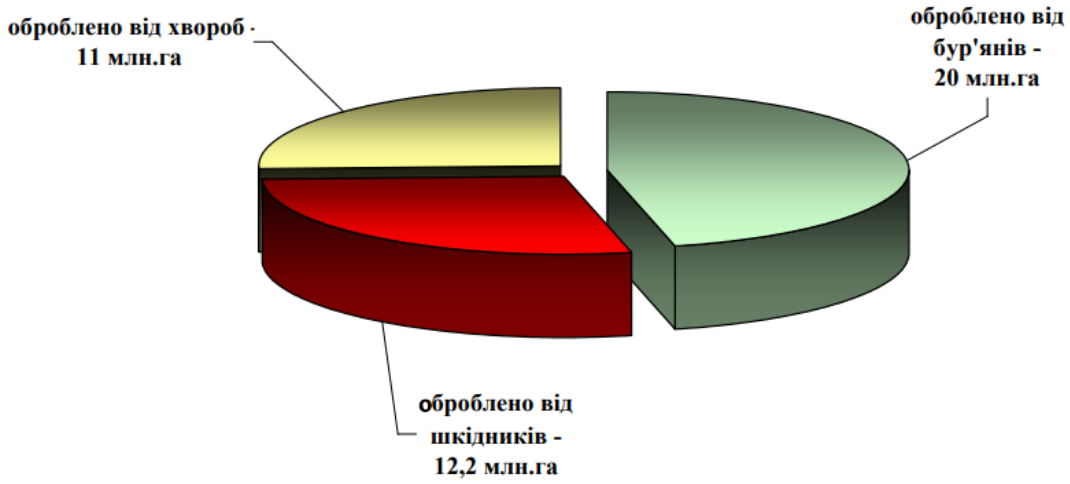

Рисунок 1.2 – Стан виконання робіт щодо боротьби із хворобами сільськогосподарських культур [23]

Втрати через хвороби та неякісну якість робіт із захисту рослин оцінюють у 30–40%. Загалом лише в США економічні втрати від інфекцій оцінюють у 40 мільярдів доларів США [22]. Щоб звести до мінімуму пошкодження культур, спричинені захворюваннями, під час росту, збору врожаю та післязбиральної обробки, а також для максимізації продуктивності та забезпечення стійкості сільського господарства, передове виявлення хвороб і профілактика культур є дуже важливими.

1.2. Передумови та еволюція глибокого навчання

Глибоке навчання – це набір методів машинного навчання , які використовуються для автоматичної побудови моделі з різними рівнями представлення шляхом відображення вхідних даних високої розмірності в вихідні дані низької розмірності. З 2006 року глибоке структурне навчання прийнято називати ієрархічним навчанням або глибоким навчанням, яке з'явилося як новий напрямок у дослідженнях галузі машинного навчання.

В останні десятиліття методи, запроваджені в результаті дослідження концепцій глибокого навчання, впливають на широкий спектр завдань обробки інформації та сигналів. Глибоке навчання є ключовим аспектом машинного навчання та штучного інтелекту (ШІ). Історично концепція глибокого навчання походить від досліджень ШНМ (рис. 1.3).

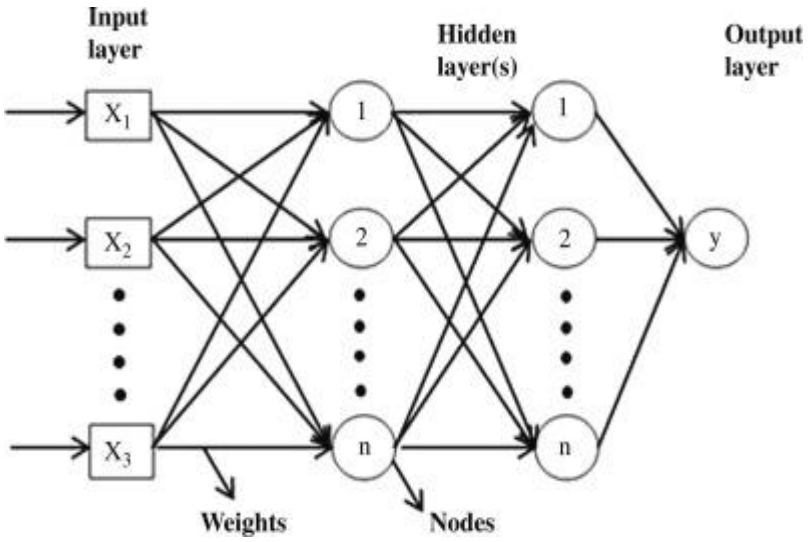

Рисунок 1.3 – Загальна структура ШНМ

Загалом ШНМ використовує алгоритм зворотного поширення як навчальний алгоритм для вивчення наборів даних. Загальну структуру ШНМ показано на рис. 1.4.

Перш ніж детально описувати концепцію глибокого навчання, давайте коротко обговоримо визначення глибокого навчання.

- Методи машинного навчання використовують різні рівні обробки інформації для трансформації та неконтрольованого або контрольованого виділення ознак, а також для класифікації, а також аналізу шаблонів.

- Ієрархія функцій низького та високого рівня в моделі неконтрольованого навчання називається глибокою архітектурою.
- Глибоке навчання – це набір підходів, які використовуються в методі машинного навчання, які допомагають вивчати різні рівні, які відповідають різним рівням абстракції, що допомагає відчувати дані, такі як звук, текст і зображення. Глибоке навчання використовує концепцію ШНМ.

Модель глибокого навчання містить кілька нейронів, так що кожен нейрон організовано в блок шарів ієрархічно. Кожен нейрон отримує вхідні дані від набору нейронів, а зв'язок, що використовується для з'єднання нейронів, має параметр, який відповідає вазі. Кожен нейрон виконує операції, отримуючи вхідний сигнал від попереднього нейрона та перетворюючи його на вихідне значення. Однак для кожного нового з'єднання вага нейрона множиться на вхідне значення, отримане від нейрона попереднього рівня, і агрегує значення за допомогою функції активації, яка обчислює вихід нейрона. Параметри оптимізовані за допомогою алгоритму градієнтного спуску (рис. 1.5), що зменшує функцію втрат.

Крім того, параметр у глибокому навчанні оновлюється після розповсюдження градієнтів функції втрат через мережу. Однак ієрархічні моделі в глибокому навчанні мають можливість вивчати різні рівні представлення даних, що відповідають різним рівням абстракції, що дозволяє використовувати концепцію щільного представлення [15]. Методи глибокого навчання широко використовуються в останні десятиліття в різних процесах автоматичної класифікації.

Градієнтний спуск (GD) – це метод оптимізації, який широко використовується в машинному навчанні для оптимізації параметрів моделі. Це метод оптимізації, спрямований виключно на опуклі цільові функції, який ітеративно пропонує, як оновити значення параметрів.

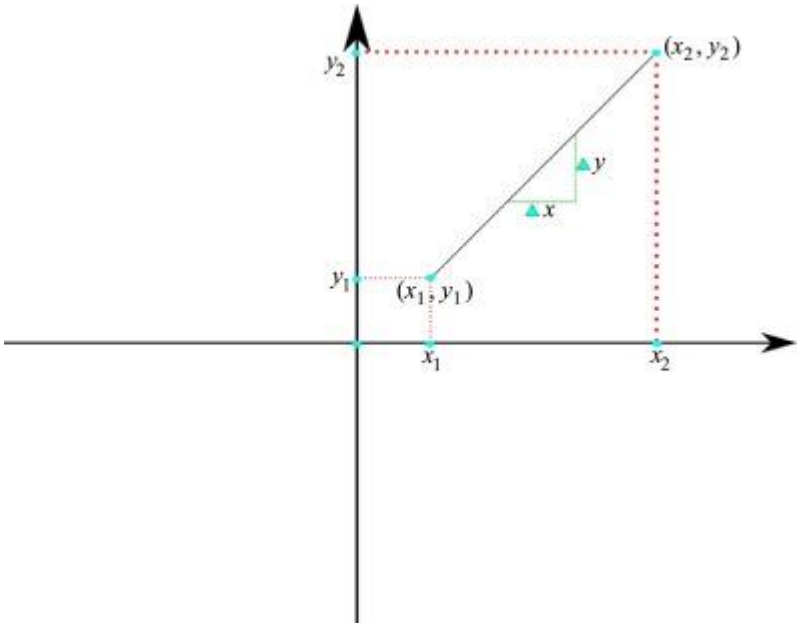

Рисунок 1.5 – Ілюстрація градієнта через 2-точкову форму лінії

Градієнт вимірює швидкість зміни виходу(ів) як функцію змін у вході(ах). У деяких налаштуваннях градієнт також називають нахилом. Наприклад, розглянемо рівняння прямої, що проходить через дві точки (x1,y1) nf (x2,y2):

$$y_2 - y_1 = m(x_2 - x_1). \qquad (1.1)$$

Тут *m* називається нахилом або градієнтом лінії, що показано на рис. 1.3.

У перспективі аналізу зображення звичайна процедура класифікації передбачає виділення ознак за допомогою набору згорткових шарів і виконує класифікацію через повністю зв’язані шари (рис. 1.6).

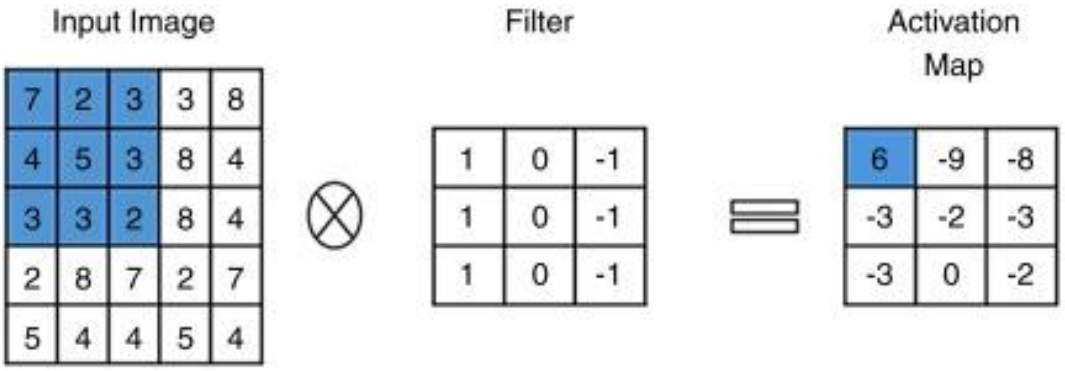

Рисунок 1.6 – Графічний приклад процесу згортки

Коли класифікатор навчений алгоритму оптимізації, якість прогнозованого результату перевіряється за допомогою справжніх значень, записаних у позначеному наборі даних. Отже, ці дані вважаються стандартними даними, виявленими на основі знань експертів-людей.

Глибоке навчання – це процес інтелектуального аналізу даних, який використовує структуру глибоких мережевих мереж, що є унікальним типом машинного навчання та методу штучного інтелекту, який надзвичайно розвинувся за останні десятиліття. Це дозволяє людям навчити машини виконувати завдання без втручання програміста. Таким чином ми увійшли в область ШІ та машинного навчання.

У майбутньому машина виконуватиме різні завдання, ніж люди сьогодні. В останні роки люди явно програмували комп’ютер у покроковому режимі для вирішення складних проблем шляхом надання даних. Завдяки експоненційному зростанню складних наборів даних останніми роками широко використовуються методи глибокого навчання, щоб запропонувати точну та надійну класифікацію даних.

Методи глибокого навчання досягли кращих результатів, ніж попередні методи машинного навчання, у таких завданнях, як обробка природної мови, класифікація зображень і розпізнавання обличчя. Отже, фактор успіху методів глибокого навчання значною мірою залежить від їх здатності формувати нелінійні та складні зв’язки з даними.

1.3. Аналіз технологій машинного навчання для виявлення хвороб у рослин

Алгоритми глибокого навчання не тільки продемонстрували надзвичайний успіх у різних сферах, але й показали вражаючі досягнення в точному діагностуванні різних захворювань рослин (рис. 1.7).

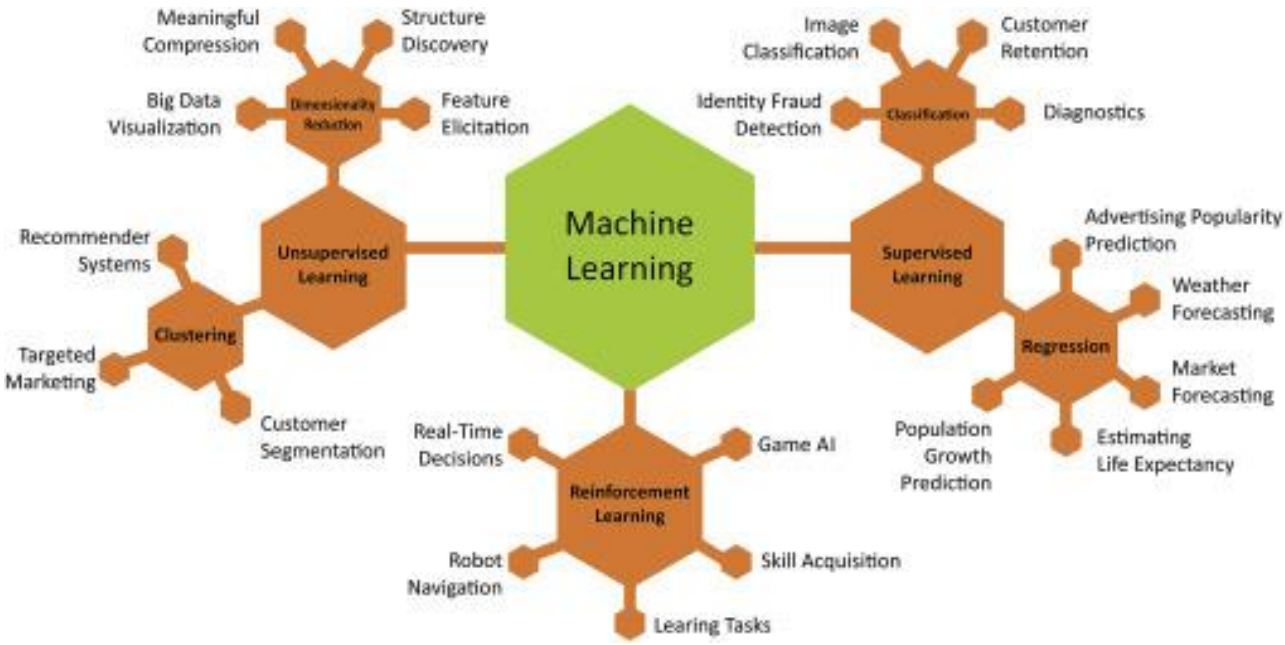

Рисунок 1.7 – Технологій машинного навчання для виявлення хвороб у рослин

Це зробило їх цінним інструментом, особливо для своєчасної діагностики серйозних захворювань, які можуть вразити такі високоцінні культури, як кукурудза. Підходи глибокого навчання пропонують ефективнішу та точнішу альтернативу традиційним методам машинного навчання, які покладаються на функції, створені вручну. Крім того, їхня гнучкість має потенціал для впровадження інноваційних застосувань у сільському господарстві. В останні роки кількість досліджень, присвячених хворобам листя сільськогосподарських рослин з використанням CNN і архітектур трансформатора зору, зросла, що робить ці підходи особливо привабливими та призводить до розробки більш ефективних методів діагностики хвороб листя рослин.

CNN зазвичай використовуються для завдань класифікації зображень і досягли найсучаснішої продуктивності в різних областях, включаючи класифікацію хвороб рослин (рис. 1.8). З іншого боку, Vision Transformers є новішою розробкою, яка демонструє багатообіцяючі результати для завдань класифікації зображень, особливо для великих наборів даних.

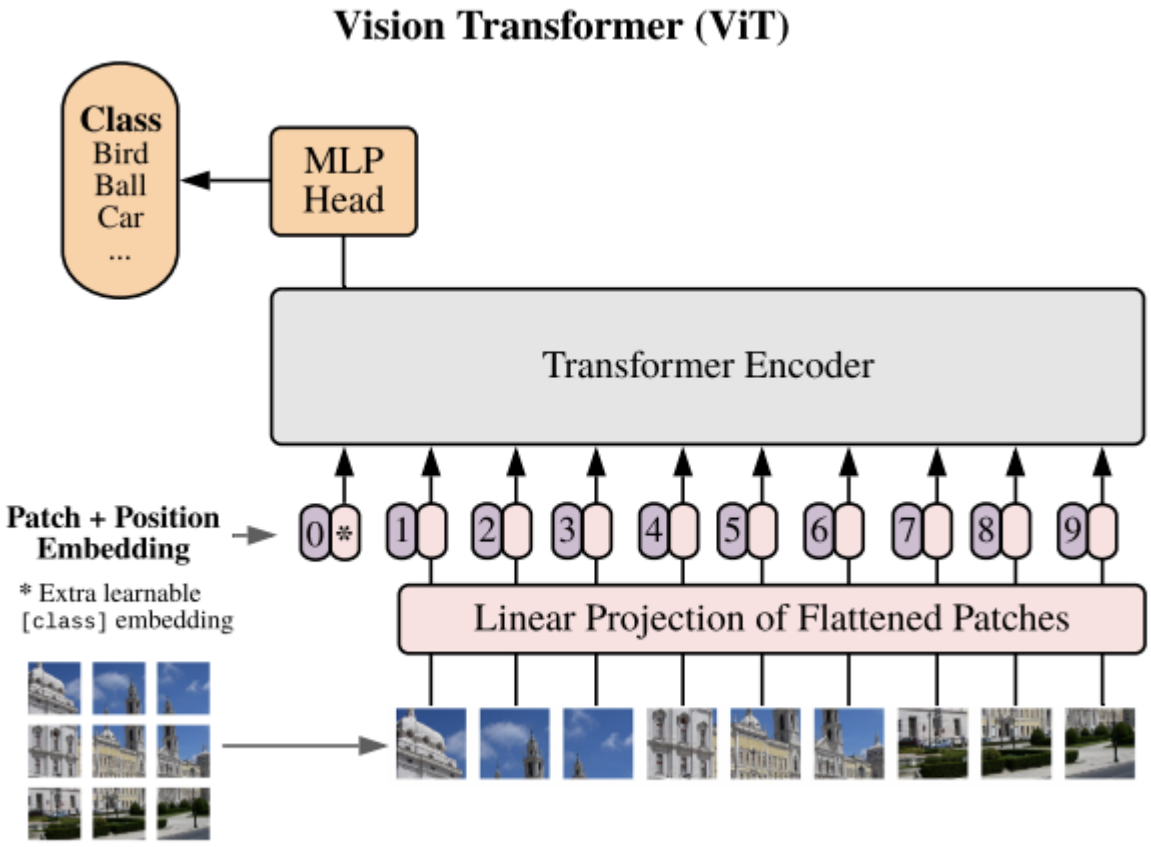

Рисунок 1.8 – модель для класифікації зображень Vision Transformer (ViT) [24]

Vision Transformer або ViT – це модель для класифікації зображень, яка використовує архітектуру, подібну до Transformer, на ділянках зображення. Зображення розбивається на патчі фіксованого розміру, кожен із яких потім лінійно вбудовується, додаються вбудовування позицій, і отримана послідовність векторів подається на стандартний кодер Transformer. Для виконання класифікації використовується стандартний підхід додавання додаткового «маркера класифікації» до послідовності.

Так як CNN, і трансформатори зору можна використовувати для аналізу зображень листя сільськогосподарських культур і точного визначення типу присутнього захворювання, що допомагає своєчасно ідентифікувати захворювання та запобігти пошкодженню посівів.

У літературі є багато досліджень, які проводилися щодо цих двох архітектур глибокого навчання. Однак обмежена доступність

загальнодоступних наборів даних призвела до досліджень, які постійно базуються або на цих наборах даних, або на спеціальних наборах даних [25]. Ось короткий виклад деяких із цих досліджень.

Хао та ін. представили більш ефективний підхід до діагностики хвороб листя рослин з іншої точки зору шляхом інтеграції ефективного дескриптора форми в архітектури CNN [26].

Розроблено методику глибокого навчання для виявлення хвороб кукурудзи. Вони використовують DenseNet121 як основну мережу для вилучення ознак і розробляють інноваційну модель під назвою MDCDenseNet, досягнуту шляхом інтеграції механізму уваги з розширеним модулем (рис. 1.9).

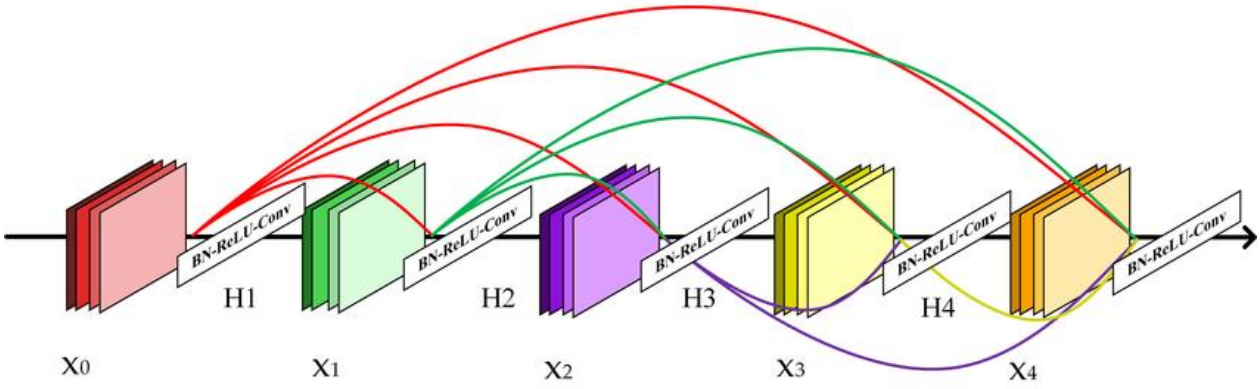

Рисунок 1.9 – Структра DenseNet121

Субраманян та ін. вивчити застосування трансферного навчання з глибокими CNN для виявлення захворювань листя кукурудзи. Вони досліджують використання попередньо навчених моделей (VGG16, ResNet50, InceptionV3 і Xception) і передачу знань у свій набір даних, який складається з 18 888 зображень. Крім того, була застосована байєсовська оптимізація для покращення можливостей узагальнення моделі [27].

Haque та ін., як і інші дослідження, представляють нову 15-шарову глибоку модель CNN, спеціально створену для виявлення хвороб кукурудзи. Вони зібрали набір даних із приблизно 3852 зображень посівів кукурудзи. Їхня нещодавно представлена модель продемонструвала вражаючі

можливості для точної ідентифікації раніше неспостережуваних зображень уражених культур кукурудзи [28].

Своєчасне виявлення захворювань у листі сільськогосподарських рослин дає кілька переваг, таких як підвищення продуктивності сільськогосподарських культур, зменшення залежності від шкідливих хімічних речовин і покращення виробництва здорових культур, що призводить до підвищення економічної прибутковості. Комп'ютерні системи (CAD) відіграють вирішальну роль у сільському господарстві, забезпечуючи своєчасну та ефективну ідентифікацію захворювань у листках рослин.

CAD-системи на основі глибокого навчання полегшують точну та швидку діагностику захворювань листя сільськогосподарських культур. Заслуговують на увагу моделі трансформатора зору, які забезпечують виняткову точність і швидкість висновків у виявленні захворювань у листі сільськогосподарських культур. Насамперед адаптується модель Multi-axis vision transformer (MaxViT) до набору даних 4 класів рослин [14], створюючи легку структуру, яка пропонує покращену точність і швидкість висновку (рис. 1.10).

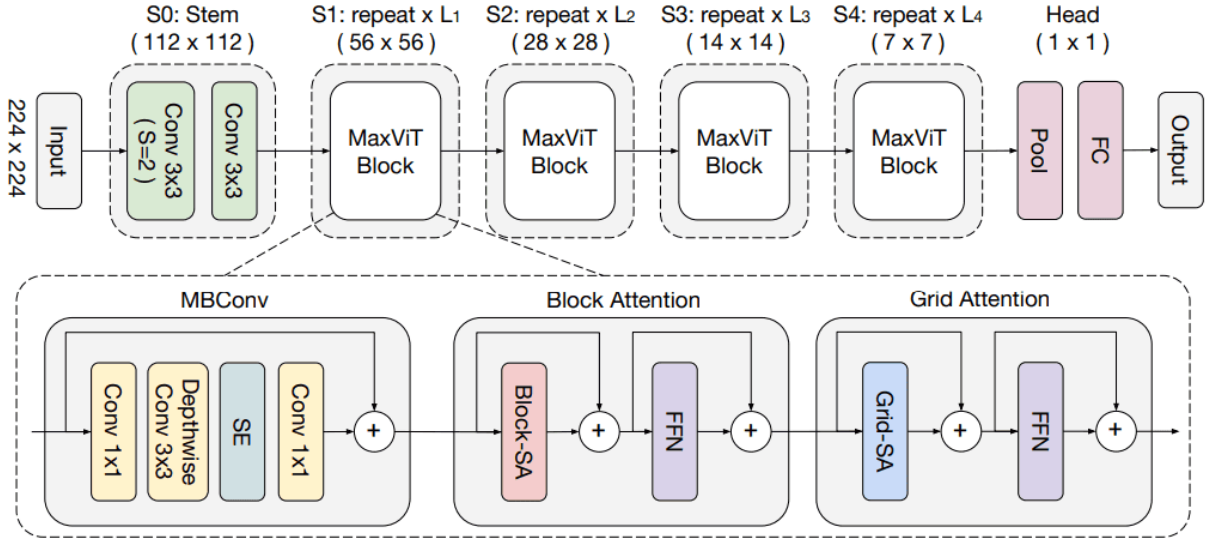

Рисунок 1.10 – Архітектура MaxViT [14]

На рис. 1.10 представлена модель архітектури MaxViT, яка передбачає дотримання типової ієрархічної структури практик Con-vNet (наприклад,

ResNet), але натомість передбачено створення нового типу базового будівельного блоку, який об’єднує рівні уваги MBConv, блоку та сітки. Для спрощення шари нормалізації та активації опущені.

Крім того, тори цієї моделі вважають, що вона підвищує точність шляхом заміни традиційної згорткової структури в стеблі архітектури MaxViT на блок стискання та збудження (SE). Для подальшого підвищення точності використано MLP на основі глобальної нормалізації відгуку (GRN) з архітектури ConvNexTv2 замість MLP в архітектурі MaxViT. При цьому об’єднано набори даних PlantVillage, PlantDoc і CD&S, що призводить до створення найширшого доступного набору даних. Потім цей набір даних розділяється на три набори – навчання, перевірка та тестування, що дозволяє оцінити здатність узагальнення моделей глибокого навчання.

Автори цієї роботи виконали комплексне порівняння продуктивності понад 28 моделей CNN і понад 36 моделей трансформаторів зору на новоствореному наборі даних. Досягнувши чудового рівня точності 99,24% і високої швидкості логічного висновку, запропонований метод перевершує всі існуючі моделі глибокого навчання в літературі. Таким чином, було продемонстровано, що ця передова модель трансформатора зору, заснована на MaxViT, є надзвичайно ефективною для практичного застосування в сільському господарстві.

Огляд літератури підкреслює трансформаційний вплив алгоритмів глибокого навчання, зокрема CNN і Vision Transformers, у сфері діагностики хвороб листя рослин. Ці передові методи не тільки перевершили традиційні методи машинного навчання, але й продемонстрували свою адаптивність і універсальність у застосуванні в сільському господарстві.

Незважаючи на те, що дослідники використовували різноманітні архітектури та інноваційні стратегії, включаючи механізми привернення уваги, перенесення навчання та збільшення даних, щоб досягти надзвичайної точності ідентифікації захворювань, проблеми, пов’язані з доступністю даних та інтерпретацією моделі, залишаються. Тим не менш, успіхи, досягнуті в цій

галузі, мають величезні перспективи для сільськогосподарської практики, пропонуючи потенціал для своєчасної ідентифікації хвороб, покращення управління виконанням робіт із захисту культур та, зрештою, підвищення продуктивності сільського господарства.

1.4. Аналіз інтелектуальних систем для виявлення хвороб рослин

У роботі [29] представлена загальна блок-схема запропонованої системи. У цій роботі було спроектовано та розроблено компактну систему підтримки прийняття рішень з використанням Raspberry PI (системне бортове обладнання) для класифікації різних рослин, рослин із хворобами та без них.

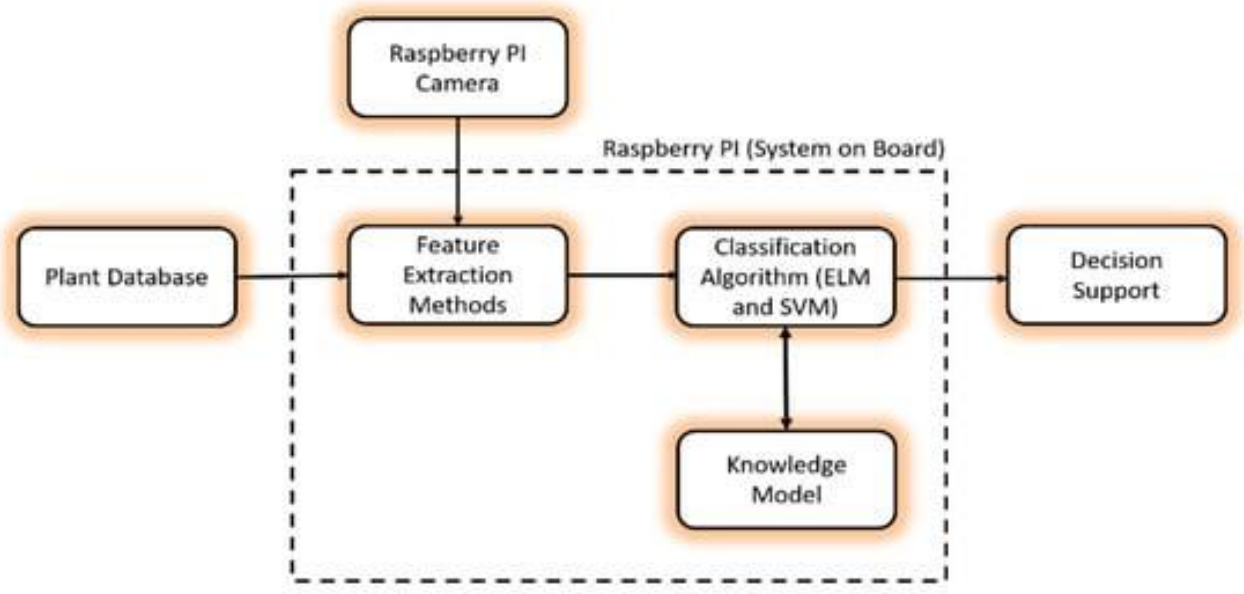

Рисунок 1.11 – Загальна блок-схема системи підтримки прийняття рішень у режимі реального часу

Для навчання моделі класифікатора були використані дані рослин для різних рослин із хворобами та без них. Під час фази навчання зображення рослин із бази даних були взяті як вхідні дані, а інформаційні характеристики з використанням моментів Ху та особливостей текстури Хараліка були вилучені із зображення. Витягнуті характеристики були представлені двом різним

алгоритмам класифікації, Extreme Learning Machine (ELM) і Support Vector Machine (SVM), з лінійним і поліноміальним ядром (порядок = 8).

Модель знань була створена в кінці етапу навчання. Під час етапу тестування зображення рослини в режимі реального часу було отримано за допомогою камери Raspberry PI та передано методам вилучення ознак. Алгоритм класифікації класифікує рослини з різними захворюваннями та без них, використовуючи попередньо отримані знання, які служать підтримкою прийняття рішення для користувача.

У роботі [30] зазначається, що однією з найважливіших проблем у сільському господарстві є боротьба зі шкідниками та хворобами у відкритому грунті (рільництва) і тепличних умовах. Існує кілька систем візуалізації, таких як гіперспектральне відображення, мультиспектральна флуоресценція та оптична когерентна томографія для зображення різних частин рослин у польових умовах. За допомогою цих систем можна отримати величезний пул зображень для заражених і нормальних рослин (рис. 1.12).

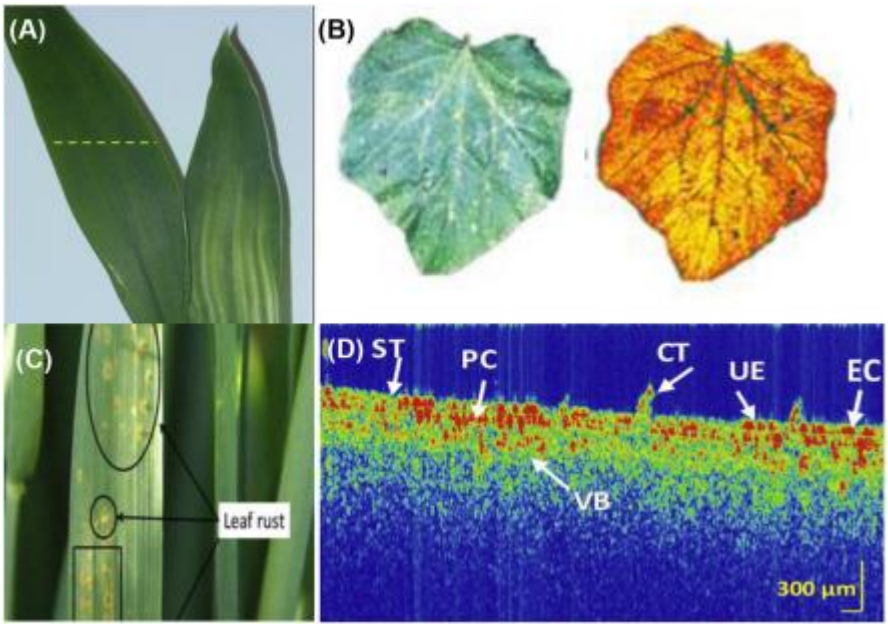

Рисунок 1.12 – Оцінення стану рослин:
(A) Неінфіковані листя; (B) гіперспектральне зображення зараженого листя; (C) заражене листя пшениці; та (D) ОКТ B-сканування в режимі реального часу in vivo . *UE* , верхній епідерміс; *PC* , клітина паренхіми; *VB* , судинний пучок; *EK* , епідермальна клітина; *ST* , продихи. Жовта пунктирна лінія (в A) показує положення ОКТ-сканування (D)

Дані можна класифікувати та класифікувати на основі методу «вилучення ознак» із отриманих зображень. Особливості кольору, текстури та форми зразків можна витягти та зберегти в базі даних. Ці функції використовуються для навчання системи ШІ. Автоматизоване, точне, швидке та в режимі реального часу виявлення заражених або здорових рослин можливе за допомогою обробки зображень за допомогою штучної нейронної мережі (ANN), згорткової нейронної мережі або моделей глибокого навчання та функцій спектрального відбиття.

Зараз широко використовується телефон з камерою, і фермери можуть зробити знімок зараженого регіону в полі та завантажити зображення через мобільний додаток. Коли фермер надає заражене листя/інші зображення, навчена система ШІ зможе виявити хворобу. Мобільний додаток надає рішення або експертні поради майже в режимі реального часу. Таке раннє та точне виявлення хвороби дозволить фермеру точно націлити пестициди на полі і тим самим захистити рослину від патогенів.

Смартфони не можуть замінити селянина в полі, але вони можуть допомогти йому виявити хвороби рослин і підказати, як з ними боротися. До переліку найкращих сервісів для діагностики хвороб рослин увійшли п'ять мобільних додатків.

1. ***Agrobase.*** Цей додаток розроблений спеціально для аграріїв. Він постійно оновлюється, щоб додавати нові дані про бур'яни, шкідників і хвороби рослин. У базі даних додатка міститься інформація про всі зареєстровані в обраній країні пестициди, інсектициди та гербіциди. Додаток можна використовувати для різних сільськогосподарських культур, овочів, фруктів, горіхів і садових рослин. Він доступний на більш ніж 80 мовах.

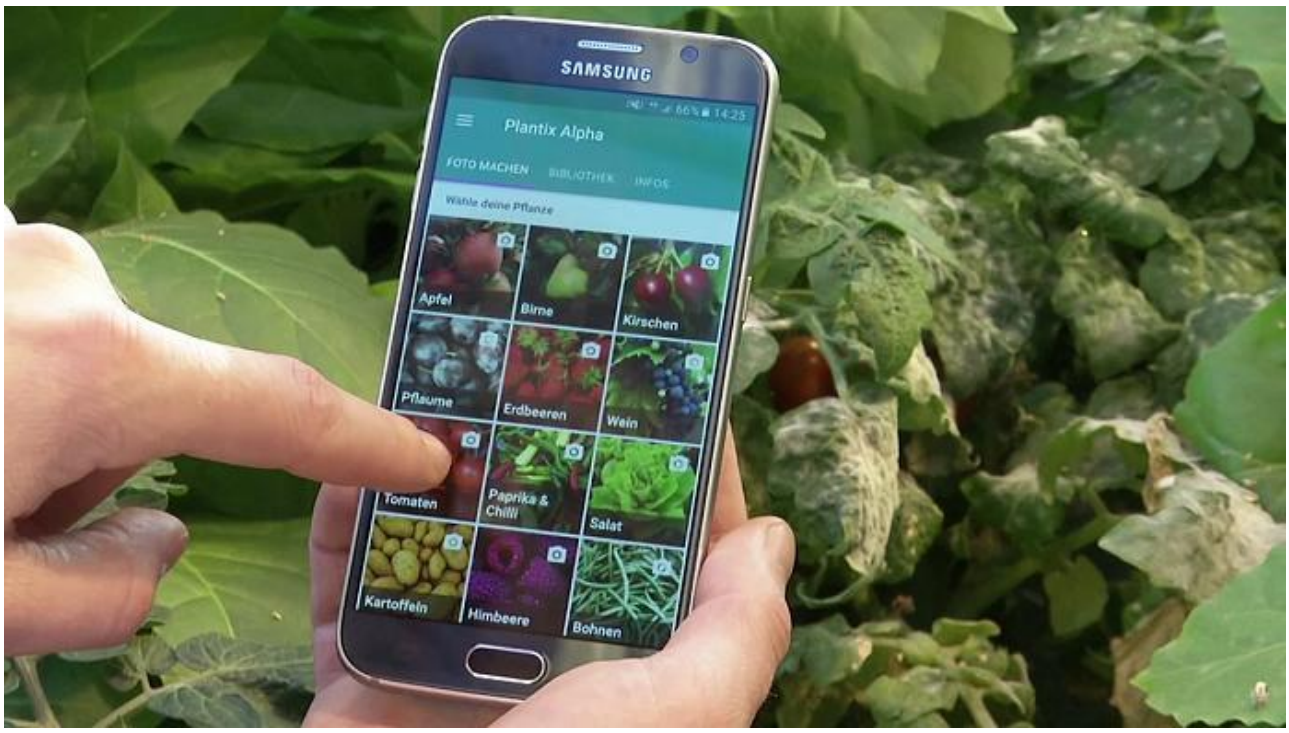

Рисунок 1.13 – Використання мобільних додатків для оцінення стану рослин

Однак у додатку є один недолік: він не містить посилань на джерела інформації про хвороби, шкідників і комах.

2. ***Plantix.*** Цей додаток розробила німецька компанія PEAT. Він використовує штучний інтелект для діагностики хвороб фруктів, овочів і польових рослин на основі фотографій. Додаток також пропонує поради щодо лікування виявлених захворювань.

Plantix є платформою, де фахівці з усього світу можуть ділитися знаннями про хвороби рослин. Додаток співпрацює з міжнародними науково-дослідними інститутами та міжурядовими організаціями, такими як ІКРІСАТ, CIMMYT і CABI.

Додаток Plant Doctor Android від Plantix можна безкоштовно завантажити через Google Play.

Рисунок 1.14 – Використання мобільних додатків Plantix для оцінення стану рослин

3. ***Pestoz.*** Цей додаток використовує штучний інтелект для діагностики захворювань рослин. Він може визначити хворобу протягом кількох секунд на основі фотографії хворої частини рослини. Додаток також пропонує поради щодо лікування.

4. ***Crop Diagnosis.*** Цей додаток використовується для діагностики захворювань рослин, виявлення шкідників і вибору необхідних хімікатів для боротьби зі шкідниками. Він також надає індивідуальні інструкції для кожного випадку. Користувачі можуть вводити дані про середовище і характеристики культури через анкету. Після аналізу анкети додаток ставить діагноз.

1.5. Обґрунтування доцільності розробки інтелектуальної інформаційної системи виявлення хвороб рослин із використанням технології Deep Learning

Отже на підставі вище сказаного можна зазначити, що хвороби рослин є однією з основних причин втрати врожаю. Вони можуть знижувати

урожайність на 20-30%, а в деяких випадках навіть до 100%. Втрати врожаю внаслідок хвороб рослин становлять близько 200 мільярдів доларів США на рік.

Традиційна діагностика хвороб рослин проводиться за допомогою візуального огляду рослин експертами. Цей метод є трудомістким і може бути ненадходящим, особливо при діагностиці незнайомих або рідкісних хвороб.

Технологія Deep Learning є одним із напрямків штучного інтелекту, який дозволяє навчати моделі на великих обсягах даних. Deep Learning може бути використаний для розробки інтелектуальних інформаційних систем виявлення хвороб рослин.

Інтелектуальна інформаційна система виявлення хвороб рослин із використанням технології Deep Learning має ряд переваг перед традиційними методами діагностики:

- ✓ Висока точність діагностики. Deep Learning-моделі можуть навчатися на великих обсягах даних, що дозволяє їм досягати високої точності діагностики.
- ✓ Швидкість діагностики. Deep Learning-моделі можуть діагностувати хвороби рослин за лічені секунди.
- ✓ Автоматизація процесу діагностики. Deep Learning-моделі можуть автоматизувати процес діагностики, що звільняє експертів для інших завдань.
- ✓ Розробка інтелектуальної інформаційної системи виявлення хвороб рослин із використанням технології Deep Learning може принести наступні вигоди:
- ✓ Зменшення втрат врожаю. Раннє виявлення хвороб рослин дозволяє вжити заходів щодо їх усунення, що може призвести до зменшення втрат врожаю.
- ✓ Підвищення ефективності виробництва. Автоматизація процесу діагностики хвороб рослин може призвести до підвищення ефективності виробництва.
- ✓ Зменшення витрат. Автоматизація процесу діагностики хвороб рослин може призвести до зниження витрат на діагностику.

Виходячи з вищевикладеного, розробка інтелектуальної інформаційної системи виявлення хвороб рослин із використанням технології Deep Learning є доцільною і може принести значні вигоди.

Розробка інтелектуальної інформаційної системи виявлення хвороб рослин із використанням технології Deep Learning є перспективним напрямком досліджень, який має значний потенціал для підвищення ефективності сільського господарства.

РОЗДІЛ 2.
ОСОБЛИВОСТІ ВИЯВЛЕННЯ ХВОРОБ РОСЛИН, ВИБІР МЕТОДІВ DEEP LEARNING ТА ПІДГОТОВКА ДАНИХ

2.1. Особливості виявлення хвороб рослин

Технології штучного інтелекту (ШІ) нещодавно почали застосовувати в галузі патології рослин для виявлення патологій та заражень рослин. Ці технології можуть трансформувати методи виявлення, діагностики та управління хворобами рослин. У цій статті ми розглянемо різні технології штучного інтелекту, які були запропоновані для виявлення патологій і заражень рослин, їхні переваги та обмеження, а також вплив цих технологій на сферу патології рослин. Однією з найпоширеніших технологій ШІ в патології рослин є машинне навчання (ML). Алгоритми ML, такі як класифікатор C4.5, деревоподібний пакувальник і машини лінійних опорних векторів, застосовуються для класифікації хвороб рослин на основі цифрових зображень. Ці алгоритми можна навчити розпізнавати специфічні патерни і симптоми хвороб, що робить їх придатними для класифікації хвороб на початкових стадіях. Однак алгоритми ML вимагають значної кількості анотованих даних для навчання і можуть не підходити для хвороб, які раніше не зустрічалися.

Технології Deep Learning, такі як CNN і DBN, також були запропоновані для виявлення аномалій рослин і заражень їх хворобами. Ці технології демонструють багатообіцяючі результати у виявленні та ідентифікації уражень на цифрових зображеннях [26]. Моделі DL можуть автоматично вивчати особливості зображень і виявляти ледь помітні симптоми захворювань, які традиційні методи обробки зображень можуть бути не в змозі виявити. Однак моделі глибокого навчання потребують значного обсягу маркованих навчальних даних і залучають інтенсивні обчислювальні ресурси, що може бути обмеженням для деяких застосувань.

Ще одна технологія ШІ, яка застосовується для виявлення патологій рослин – це комп'ютерний зір (CV). Алгоритми CV, такі як виявлення об'єктів і семантична сегментація, можуть бути використані для ідентифікації та локалізації конкретних областей, що представляють інтерес на зображеннях, таких як листя рослин і симптоми захворювань [42]. Ці алгоритми можна використовувати для автоматичного перетворення зображень у розпізнавані патерни або характеристики, які можна інтегрувати з алгоритмами ML або DL для виявлення та класифікації захворювань. Однак алгоритми CV потребують величезної кількості мічених зображень для навчання моделі і можуть не підходити для захворювань, які раніше не спостерігалися.

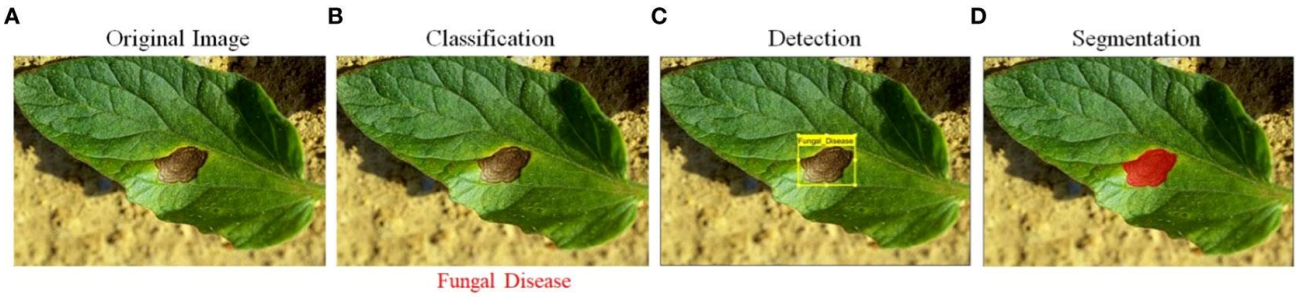

Рисунок 2.1 – Вхідне необроблене зображення(A), класифікація листків(B), виявлення пошкоджень (C) і сегментація пошкоджень (D)

На рисунку 2.1 представлено чотири зображення, кожне з яких відображає різний етап виявлення хвороби рослин. Перше зображення є вхідним, наступне зображення відображає результати ідентифікації хвороби. На третьому зображенні показано виявлення ураження, а на останньому – результати сегментації ураження рослин.

Технології штучного інтелекту показали багатообіцяючі результати у виявленні патологій та заражень рослин. Інтелектуальні системи на основі ML, DL та CV використовуються для класифікації та сегментації ураження рослин на цифрових зображеннях і можуть суттєво змінити методи виявлення, діагностики та управління хворобами рослин (табл. 2.1).

Таблиця 2.1 – Порівняння різних технологій обробки зображень

Технологія	Основні складові	Необхідні передумови	Використання
Традиційна обробка зображень	Ручна розробка ознак + класифікатори або правила	Значна диференціація між ураженими та здоровими ділянками, мінімальне втручання або занепокоєння.	Виявлення хвороб рослин і шкідників у контрольованому середовищі
DL	Автоматичне навчання ознак з використанням CNNs	Великі обсяги відповідних даних, високопродуктивні обчислювальні пристрої	Адаптація до змін у складних природних умовах

Означені технології потребують значної кількості анотованих навчальних даних і можуть не підходити для хвороб, які раніше не спостерігалися. Необхідні подальші дослідження для розробки узагальнюючих моделей, які можна застосувати до різних видів рослин і хвороб, а також для оприлюднення більшої кількості наборів даних для навчання та оцінки моделей. У таблиці 2.1 надано вичерпну інформацію про інструменти та технології, що використовуються для виявлення хвороб рослин. Вона містить детальну інформацію про різні методи вилучення ознак, в тому числі ті, що базуються на ручних і навчальних ознаках, а також відповідні методи обробки малих і великих наборів даних зображень рослин.

2.2. Вибір інструментарію Deep Learning

Для розробки моделі виявлення хвороб рослин із використанням технології Deep Learning використано наступні бібліотеки:

- ✓ Os – взаємодіє з операційною системою, ймовірно, для операцій з файлами та каталогами;
- ✓ Numpy – забезпечує ефективні чисельні обчислення та маніпуляції масивами;
- ✓ Pandas – пропонує високопродуктивні структури даних і інструменти аналізу для маніпулювання даними та дослідження;
- ✓ Torch – основна бібліотека PyTorch, яка використовується для завдань глибокого навчання, таких як побудова та навчання нейронних мереж;
- ✓ matplotlib.pyplot – дозволяє будувати графіки та візуалізації для візуалізації даних і результатів моделювання.

Окрім того, використовували спеціальні бібліотеки PyTorch, до яких належать:

- ✓ torch.nn – містить будівельні блоки для побудови нейронних мереж (шари, функції активації тощо);
- ✓ torch.utils.data – надає інструменти для створення та керування завантажувачами даних, які ефективно передають дані моделям під час навчання;
- ✓ torch.nn.functional – пропонує набір корисних функцій для операцій нейронної мережі, таких як обчислення втрат і активація;
- ✓ torchvision.transforms – містить перетворення зображень для попередньої обробки зображень перед подачею їх у моделі;
- ✓ torchvision.utils – надає службові функції для візуалізації та обробки зображень;
- ✓ torchvision.datasets – пропонує попередньо створені набори даних для типових завдань класифікації зображень.

✓ Torchsummary – підсумовує архітектуру та кількість параметрів моделей PyTorch, допомагаючи зрозуміти складність моделі.

Також нами використовувалася бібліотеки зображень, до якої належить – PIL (рис. 2.2).

Рисунок 2.2 – Бібліотека PIL

Бібліотека PIL використовується для відкриття, обробки та відображення зображень. Python Imaging Library – це потужний інструмент для роботи з растровою графікою в Python. Бібліотека підтримує різні типи зображень та формати файлів, що дозволяє виконувати різноманітні завдання, такі як редагування, обробка та перетворення зображень.

Створення моделі виконуємо у Jupyter Notebook. Jupyter Notebook – це веб-додаток, який дозволяє створювати та ділитися документами, що містять код, текст, рівняння, графіки та інші елементи. Він є популярним інструментом для розробки ПЗ, аналізу даних, машинного навчання та інших завдань, де необхідна інтерактивна взаємодія з кодом і даними.

Jupyter Notebook складається з двох основних компонентів: ядра (kernel) та інтерфейсу. Ядро відповідає за виконання коду, а інтерфейс забезпечує можливість взаємодії з ядром і відображення результатів виконання коду. Jupyter Notebook має ряд переваг, які роблять його популярним інструментом для різних завдань.

Окрім того, нами використано %matplotlib inline, який у Jupyter Notebook забезпечує відображення графіків.

Нами вирішується одне завдання глибокого навчання, що передбачає класифікацію зображень із рослинами та виявлення їх хвороб враховуючи бібліотеку PyTorch, яка пов’язана із зображеннями. Конкретним завданням є

класифікація зображень, виявлення хворих рослин та аналіз на основі зображень.

Використання Torchsummary вказує на складість нашої моделі. Torchsummary – це бібліотека для Python, яка дозволяє отримати резюме архітектури та параметрів моделей PyTorch. Цей резюме містить інформацію про кількість шарів, типи шарів, розміри шарів та кількість параметрів у кожному шарі.

Torchsummary можна використовувати для різних цілей, включаючи:

✓ розуміння архітектури моделі – Torchsummary може допомогти вам зрозуміти, як працює ваша модель і як вона обробляє дані.

✓ виявлення проблем із моделлю – Torchsummary може допомогти вам виявити проблеми з архітектурою моделі, такі як надмірна складність або недостатня кількість параметрів.

✓ порівняння моделей – Torchsummary можна використовувати для порівняння архітектур різних моделей, щоб визначити, яка модель найкраще підходить для вашого завдання.

2.3. Збір та підготовка даних для створення моделі виявлення хвороб рослин

Людство має збільшити виробництво продуктів харчування приблизно на 70% до 2050 року, щоб прогодувати очікувану чисельність населення, яка, за прогнозами, становитиме понад 9 мільярдів людей. Наразі інфекційні захворювання знижують потенційний врожай в середньому на 40%, а багато фермерів у країнах, що розвиваються, втрачають до 100% врожаю.

У нашій роботі використано набір даних PlantVillage [43]. Він складається з близько 87 000 зображень здорових і хворих листків, розділених на 38 категорій за видами і хворобами.

Збір та підготовка даних для створення моделі виявлення хвороб рослин є важливим етапом у процесі розробки моделі. Від якості та кількості даних залежить точність та ефективність моделі. Нами виконано дослідження цих даних. Зразки набору даних PlantVillage представлено на рис. 2.3.

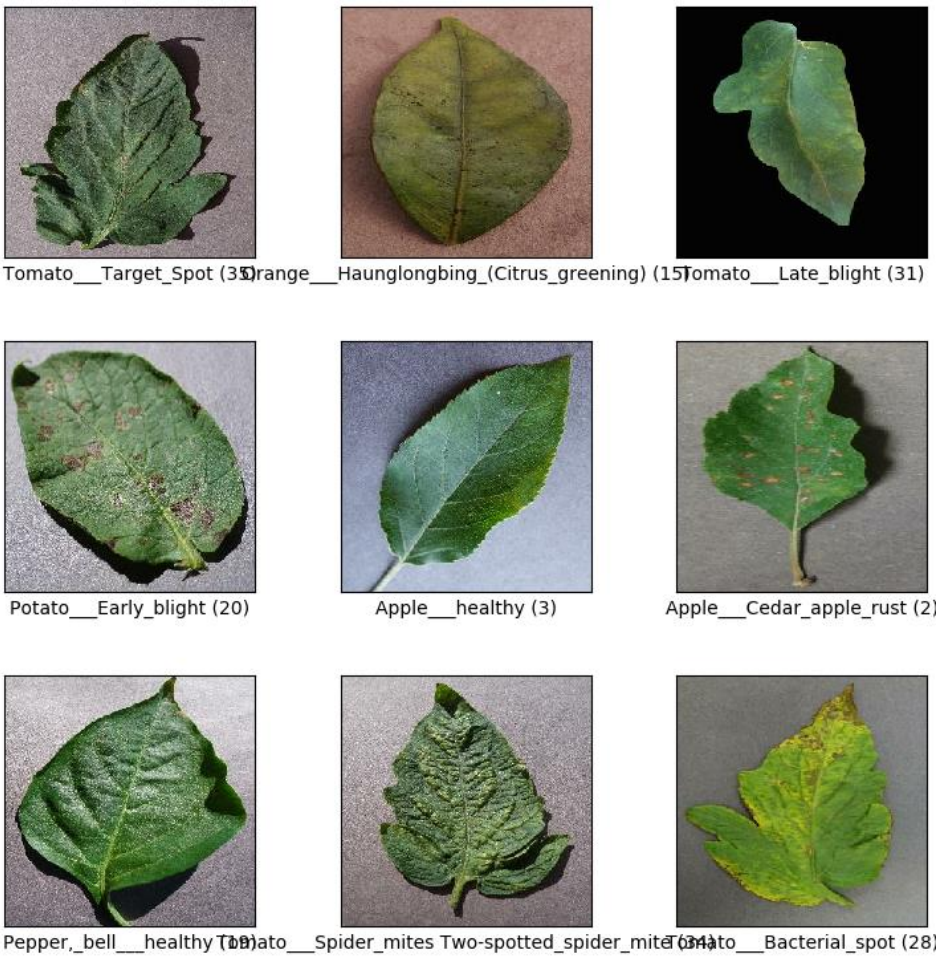

Рисунок 2.3 – Зразки набору даних PlantVillage

У наборі даних доступні наступні 38 класів зображень листків сільськогосподарських рослин:

1. Яблуня___Яблунева_парша;
2. Яблуня___Чорна_гниль;
3. Яблуня___Яблунева_іржа;
4. Яблуня___здорова;
5. Чорниця___здорова;
6. Вишня___Морошниста_роса;
7. Вишня___здорова;

8. Кукурудза___Церкоспорозна_плямистість_листка;
9. Кукурудза___Сіра_плямистість_листка;
10. Кукурудза_(кукурудзяна)___Звичайна_іржа;
11. Кукурудза_(кукурудза)___Північний_окис_листя;
12. Кукурудза_(кукурудза)___здорова;
13. Виноград___Чорна_гниль;
14. Виноград___Листяна_гнилизна_(Ізаріопсис_листяна_плямистість);
15. Виноград___здоровий;
16. Апельсин___Хаунглонгінг_(Цитрусова_позеленіння) ;
17. Персик___Бактеріальна_плямистість;
18. Персик___здоровий;
19. Перець,_дзвіночок___Бактеріальна_плямистість;
20. Перець,_болгарський___здоровий;
21. Картопля___Ранній_фітофтороз;
22. Картопля___пізня_фітофтороз;
23. Картопля___здорова;
24. Малина___здорова;
25. Соя___здорова;
26. Патисони___Морошниста_борошниста_роса;
27. Полуниця___Опік_листя;
28. Полуниця___здорова;
29. Томат___Бактеріальна_плямистість;
30. Томат___Фітофтороз;
31. Томат___Пліснявіння_листя;
32. Томат___Септоріоз_листяна_плямистість;
33. Томат___Павутинний_кліщ,
34. Томат___Двоплямистий_павутинний_кліщ;
35. Томат___Цільова_плямистість;
36. Томат___Вірус_жовтої_кучерявості_листя_томатів;
37. Томат___Вірус_мозаїки_томатів;

38. Томат___здоровий.

Нами виконана попередня обробка зображень (рис. 2.4).

```
In [2]: import tensorflow as tf
        from tensorflow import keras

        import matplotlib.pyplot as plt
        import numpy as np

        import os

In [3]: image_size = 224
        target_size = (image_size, image_size)
        input_shape = (image_size, image_size, 3)

        batch_size = 32
        epochs = 25
```

Рисунок 2.4 – Код для попередньої обробки зображень

Зокрема, нами встановлено цільовий розмір для зображень, розмір яких буде змінено до 224x224 пікселів – image_size = 224: Це звичайний розмір, який використовується для завдань класифікації зображень. Після цього створено кортеж – target_size = (image_size, image_size), що представляє потрібні розміри зображення, що використовується для зміни розміру зображень у наступному. Також нами визначено очікувану вхідну форму для моделі – input_shape = (image_size, image_size, 3), вказуючи, що очікуються зображення з висотою та шириною 224 пікселів і 3 кольоровими каналами (RGB).

Після цього нами задано параметри навчання. Зокрема, вказано кількість зображень – batch_size = 32, які потрібно обробляти разом на кожному кроці навчання. Розмір пакета 32 є звичайним вибором, який збалансовує ефективність і використання пам’яті. Встановлено кількість разів – epochs = 25, коли модель проходитиме весь навчальний набір даних. Кожна епоха передбачає пакетну обробку всіх доступних навчальних зображень. Значення епохи 25 свідчить про помірну тривалість навчання. Розмір партії та кількість епох є вирішальними факторами, що впливають на швидкість навчання та конвергенцію моделі.

Для того, щоб отримати максимальну користь від навчальних прикладів, нами доповнено їх низкою випадкових перетворень, щоб модель не бачила двічі одне й теж саме зображення. Це допомагає запобігти надмірному пристосуванню і допомагає моделі краще узагальнювати результати.

У TensorFlow це зроблено за допомогою класу tf.keras.preprocessing.image.ImageDataGenerator. Цей клас дозволяє налаштовувати випадкові перетворення та операції нормалізації, які будуть виконуватися над вашими даними зображень під час навчання, а також створювати генератори пакетів доповнених зображень (та їхніх міток) за допомогою – flow(дані, мітки) або flow_from_directory (каталог). Ці генератори можна використовувати з методами моделі tf.keras, які приймають генератори даних як вхідні дані, підганяють, оцінюють і прогнозують.

```
In [4]: base_dir = "../input/new-plant-diseases-dataset/new plant diseases dataset(augmented)/New Plant Diseases Dataset(Augmented)"
        train_dir = os.path.join(base_dir,"train")
        test_dir = os.path.join(base_dir,"valid")
```

Рисунок 2.5 – Код для встановлення шляху до каталогів

Нами встановлено шлях до базового каталогу – base_dir = "../input/new-plant-diseases-dataset/new plant diseases dataset(augmented)/New Plant Diseases Dataset(Augmented)". Після цього виконано побудову шляху до навчального каталогу – train_dir = os.path.join(base_dir, "train").

Нами представлено гістограму кількості зображень для окремих класів зображень листків сільськогосподарських рослин (рис. 2.6).

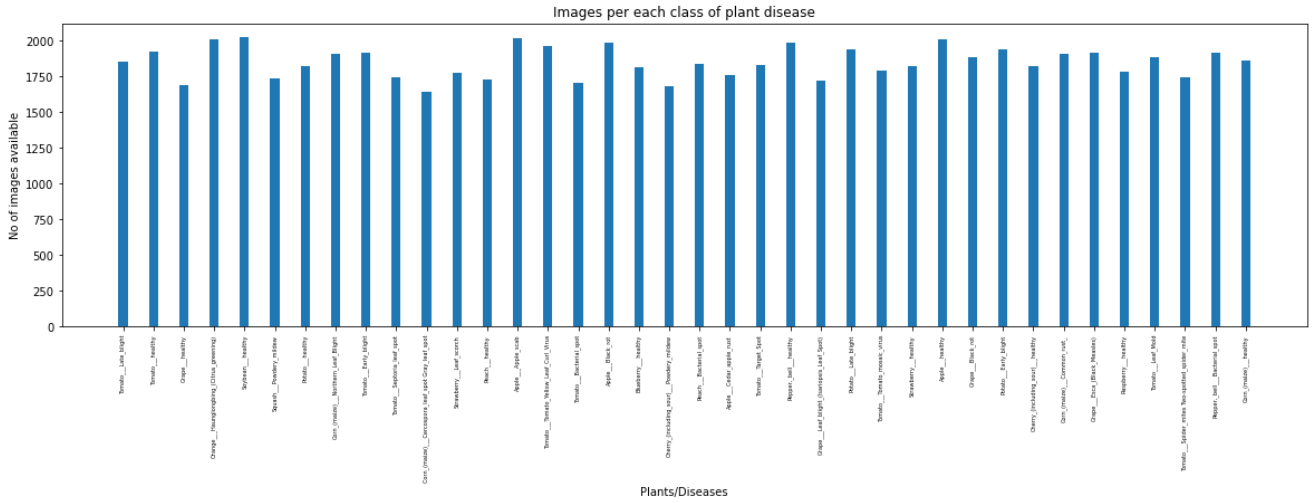

Рисунок 2.6 – Гістограма кількості зображень для окремих класів зображень листків сільськогосподарських рослин

Це забезпечує поєднання базового каталогу із «тренуванням», щоб створити шлях, що вказує на навчальні зображення в наборі даних. Наступник кроком є створення шляху до каталогу тестування – test_dir = os.path.join(base_dir, "valid"). Це створює шлях до зображень перевірки, позначених як «дійсні» в структурі набору даних.

```
In [6]: train_data = train_datagen.flow_from_directory(train_dir,
                                                       target_size = (image_size, image_size),
                                                       batch_size = batch_size,
                                                       class_mode = "categorical")

        test_data = test_datagen.flow_from_directory(test_dir,
                                                     target_size = (image_size, image_size),
                                                     batch_size = batch_size,
                                                     class_mode = "categorical")

        Found 70295 images belonging to 38 classes.
        Found 17572 images belonging to 38 classes.
```

Рисунок 2.7 – Код для розділення наборів зображень для навчання та перевірки

Нами використано flow_from_directory() для створення пакетів даних зображень (і їхніх міток) безпосередньо з наших зображень у відповідних теках. Розділення наборів для навчання та перевірки має вирішальне значення для оцінки продуктивності моделі на невидимих даних.

РОЗДІЛ 3.
РЕЗУЛЬТАТИ НАВЧАННЯ МОДЕЛІ ТА РОЗРОБКИ ІНТЕЛЕКТУАЛЬНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ВИЯВЛЕННЯ ХВОРОБ РОСЛИН

3.1. Створення моделі для виявлення хвороб рослин

Нами використано модель MobileNet, яка є архітектурою нейронної мережі, розробленою компанією Google для використання в мобільних пристроях. Вона є ефективною в обчисленні та споживає менше ресурсів, ніж інші архітектури, такі як ResNet або VGG. Модель MobileNet використовує техніку, відому як згорткові операції з роздільною здатністю, щоб зменшити кількість параметрів у мережі. Ці операції зводять до мінімуму кількість параметрів, необхідних для виявлення візуальних ознак, не втрачаючи при цьому точності. Модель MobileNet була успішно використана в різних завданнях машинного навчання, включаючи класифікацію зображень, розпізнавання об'єктів та виявлення осіб.

Спочатку нами створено базову модель MobileNet без верхніх шарів, оскільки хочемо використати її для 38 класів, а також попередньо навчені ваги для ImageNet (рис. 3.1).

```
In [9]: base_model = tf.keras.applications.MobileNet(weights = "imagenet",
                                                     include_top = False,
                                                     input_shape = input_shape)

        base_model.trainable = False

        Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet/mobilenet_1_0_224_tf_no_top.h5
        17227776/17225924 [==============================] - 0s 0us/step
```

Рисунок 3.1 – Код для створення базової модель MobileNet без верхніх шарів

Встановлено, що функція load_model() завантажує модель MobileNet з Google Cloud Storage. Модель має розмір 17227776 байтів, що становить

приблизно 17 МБ. Завантаження моделі займає 0 секунд, що означає, що вона завантажується досить швидко.

Після цього нами створено наступну модель поверх MobileNet, використовуючи функціональний API (рис. 3.2).

```
In [10]: inputs = keras.Input(shape = input_shape)

         x = base_model(inputs, training = False)
         x = tf.keras.layers.GlobalAveragePooling2D()(x)
         x = tf.keras.layers.Dropout(0.2)(x)
         x = tf.keras.layers.Dense(len(categories),
                                   activation="softmax")(x)

         model = keras.Model(inputs = inputs,
                             outputs = x,
                             name="LeafDisease_MobileNet")
```

Рисунок 3.2 – Код для створення моделі поверх MobileNet, використовуючи функціональний API

Насамперед створено вхідний шар – inputs = keras.Input(shape=input_shape), який приймає зображення з указаним input_shape. Ця форма відповідає очікуваному формату попередньо навченої моделі MobileNet.

Після цього виконується інтеграція базової моделі – x = base_model(inputs, training=False), що забезпечує подачу вхідних зображень через попередньо навчену базову модель, ймовірно, архітектуру MobileNet.

Виконується глобальне середнє об'єднання – x = tf.keras.layers.GlobalAveragePooling2D()(x), що забезпечує стискання просторових розмірів виходу з базової моделі в єдиний вектор ознак для кожного зображення. Це зменшує складність моделі та допомагає запобігти переобладнанню.

Також запропоновано випадково скидати 20% з’єднань у векторі ознак під час навчання – x = tf.keras.layers.Dropout(0.2)(x), що забезпечує зменшення переобладнання, змушуючи модель вивчати більш надійні функції.

Вихідний рівень – x = tf.keras.layers.Dense(len(categories), activation="softmax")(x), застосовує повний зв’язаний шар із len(categories) нейронами, де кожен нейрон представляє ймовірність певної категорії хвороби

листя. Функція активації «softmax» гарантує, що сума вихідних ймовірностей дорівнює 1, що представляє дійсний розподіл ймовірностей.

Виконуємо створення моделі – model = keras.Model(inputs=inputs, outputs=x, name="LeafDisease_MobileNet"), що дає можливість отримати остаточну модель із зазначеними вхідним і вихідним шарами, називаючи її «LeafDisease_MobileNet».

Вище означене створює багатокласову модель класифікації хвороб листя з використанням попередньо навченої бази MobileNet. Він використовує передачу навчання, заморожуючи ваги базової моделі та додаючи шари для виділення ознак і класифікації. Глобальне об’єднання середніх значень спрощує виділення ознак і зменшує переобладнання.

У наших численних експериментах було виявлено, що оптимізатор Adam дуже добре працює зі швидкістю навчання за замовчуванням із значеннями $\beta 1$, $\beta 2$ (рис. 3.3)

```
In [11]: optimizer = tf.keras.optimizers.Adam()

         model.compile(optimizer = optimizer,
                       loss = tf.keras.losses.CategoricalCrossentropy(from_logits = True),
                       metrics=[keras.metrics.CategoricalAccuracy(),
                                'accuracy'])
```

Рисунок 3.3 – Код для створення моделі поверх MobileNet, використовуючи функціональний API

Нами здійснено вибір оптимізатора – optimizer = tf.keras.optimizers.Adam(). Зокрема, вибрано оптимізатор Adam, популярний і ефективний алгоритм оптимізації, відомий своїми можливостями адаптивної швидкості навчання. Він часто є швидший, ніж інші оптимізатори, такі як стохастичний градієнтний спуск (SGD).

Після цього виконується складання моделі – model.compile(optimizer=optimizer, loss=tf.keras.losses.CategoricalCrossentropy(from_logits=True), metrics=[keras.metrics.CategoricalAccuracy(), 'accuracy']), що налаштовує модель

для навчання, визначаючи оптимізатор, функцію втрат і показники для відстеження:

Нами використовується функція втрат, як категоріальна перехресна ентропія, що є звичайним вибором для проблем багатокласової класифікації. Вона вимірює різницю між прогнозованими ймовірностями моделі та справжніми мітками. from_logits=True вказує, що вихідний рівень моделі надає необроблені логіти (ненормалізовані оцінки), які функція втрат внутрішньо перетворює на ймовірності.

Запропоновано використовувати метрики – keras.metrics.CategoricalAccuracy(), які спеціально розроблено для багатокласової класифікації, обчислює частку правильно класифікованих прикладів. При цьому використовується «accuracy» – загальний показник точності, який також вимірює правильні класифікації.

Код налаштовує модель для навчання з відповідними виборами для оптимізації та оцінки. Adam optimizer сприяє ефективному навчанню. Категоріальна перехресна втрата ентропії ефективно спрямовує навчання для багатокласової класифікації. Показники категорійної точності та загальної точності дають змогу зрозуміти ефективність моделі. Аргумент from_logits=True забезпечує правильну обробку вихідних даних моделі для розрахунку збитків.

3.2. Результати навчання моделі для виявлення хвороб рослин

Нами виконано навчання моделі на наборі даних для тренування з використанням набору даних для перевірки для оцінки точності із використанням функції fit() – history = model.fit(train_data, validation_data=test_data, epochs=epochs, steps_per_epoch=150, validation_steps=100).

Встановлено відповідні параметри: 1) train_data – набір даних для тренування, що містить пари зображень та міток; 2) validation_data – набір даних для перевірки, що також містить пари зображень та міток; 3) epochs – кількість епох навчання; 4) steps_per_epoch – кількість кроків за епоху на наборі даних для тренування; 5) validation_steps – кількість кроків за епоху на наборі даних для перевірки.

```
In [12]: history = model.fit(train_data,
                             validation_data=test_data,
                             epochs=epochs,
                             steps_per_epoch=150,
                             validation_steps=100)
```

Рисунок 3.4 – Код для навчання моделі

Набір даних для тренування train_data містить пари зображень та міток. Зображення повинні мати однаковий формат, що і input_shape моделі. Мітки повинні бути цілими числами від 0 до len(categories) - 1, де categories – список категорій листяних захворювань.

Набір даних для перевірки validation_data також містить пари зображень та міток. Він використовується для оцінки точності моделі під час навчання.

Епохи epochs визначають кількість разів, коли модель буде обробляти весь набір даних для тренування. Більше епох, як правило, призводить до більшої точності, але також може призвести до перенавчання.

Кроки за епоху steps_per_epoch визначають, скільки разів модель буде обробляти набір даних для тренування за епоху. Церозраховується, поділяючи кількість зображень у наборі даних для тренування на steps_per_epoch.

Код навчає модель на наборі даних для тренування з використанням набору даних для перевірки для оцінки точності. Параметри epochs, steps_per_epoch та validation_steps можна налаштувати для оптимізації процесу навчання.

Результати виконаного навчання моделі та визначення її оцінок представлено у табл. 3.1.

Таблиця 3.1 – Результати виконаного навчання моделі та визначення її оцінок

Epoch	Loss	Categorical Accuracy	Validation Loss	Validation Categorical Accuracy
1	2,97	0,26	0,85	0,79
2	0,85	0,78	0,50	0,87
3	0,58	0,83	0,39	0,89
4	0,52	0,84	0,33	0,90
5	0,42	0,88	0,29	0,92
6	0,38	0,88	0,28	0,91
7	0,33	0,91	0,25	0,92
8	0,30	0,91	0,24	0,93
9	0,31	0,90	0,21	0,94
10	0,29	0,91	0,22	0,93
11	0,27	0,91	0,24	0,92
12	0,29	0,90	0,21	0,93
13	0,26	0,92	0,17	0,94
14	0,25	0,92	0,18	0,94
15	0,24	0,92	0,17	0,95
16	0,24	0,92	0,19	0,93
17	0,21	0,93	0,17	0,95
18	0,23	0,93	0,15	0,95
19	0,26	0,91	0,15	0,95
20	0,24	0,92	0,17	0,95
21	0,21	0,93	0,18	0,94
22	0,22	0,93	0,15	0,95
23	0,21	0,93	0,16	0,94
24	0,22	0,94	0,16	0,94
25	0,22	0,93	0,15	0,95

На підставі отриманих результатів побудовано тенденції зміни показників точності моделі впродовж її навчання (рис. 3.5). Втрати на наборі даних для тренування швидко знижуються в перших кількох епох, а потім знижуються повільніше. Точність на наборі даних для тренування також швидко зростає в перших кількох епох, а потім зростає повільніше. Втрати на наборі даних для перевірки знижуються повільніше, ніж на наборі даних для тренування, і досягають 0,15 у 25-й епосі. Точність на наборі даних для перевірки зростає повільніше, ніж на наборі даних для тренування, і досягає 95% у 25-й епосі.

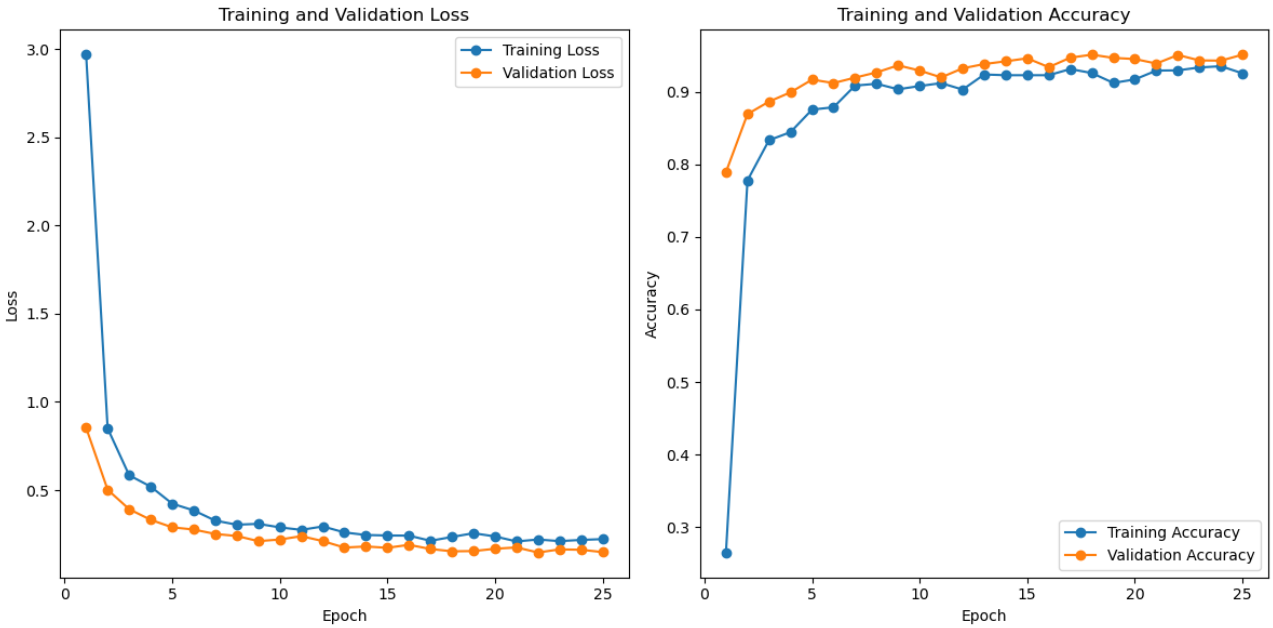

Рисунок 3.5 – Тенденції зміни показників точності моделі впродовж її навчання

Встановлено, модель досягла високої точності на наборі даних для перевірки, досягнувши 95% точності в 25-й епосі навчання. Втрати на наборі даних для перевірки також були низькими, досягнувши 0,15 у 25-й епосі. Модель демонструє хорошу стабільність у своїх результатах, оскільки втрати та точність на наборі даних для перевірки не змінюються значно після 20-ї епохи.

На основі отриманих результатів можна зробити висновок, що модель досягла високої точності і може бути використана для класифікації листяних захворювань. Для досягнення ще більшої точності можна продовжити навчання моделі до більшої кількості епох. Однак це може призвести до перенавчання, тому слід уважно стежити за результатами навчання.

Отриману модель зберігаємо model.save('plant_disease') у стандартному форматі TensorFlow 2 SavedModel. У подальшому її використовуємо у інтелектуальній інформаційній системі виявлення хвороб рослин.

3.3. Результати створення вікна користувачів інтелектуальної інформаційної системи виявлення хвороб рослин

Нами написано код, який представлено у додатку А, який створює інтерфейс користувача для виявлення хвороб рослин. Він написаний на Python за допомогою бібліотеки Tkinter.

Інтерфейс користувачів інтелектуальної інформаційної системи виявлення хвороб рослин представлено на рис. 3.6 і він складається із декількох елементів.

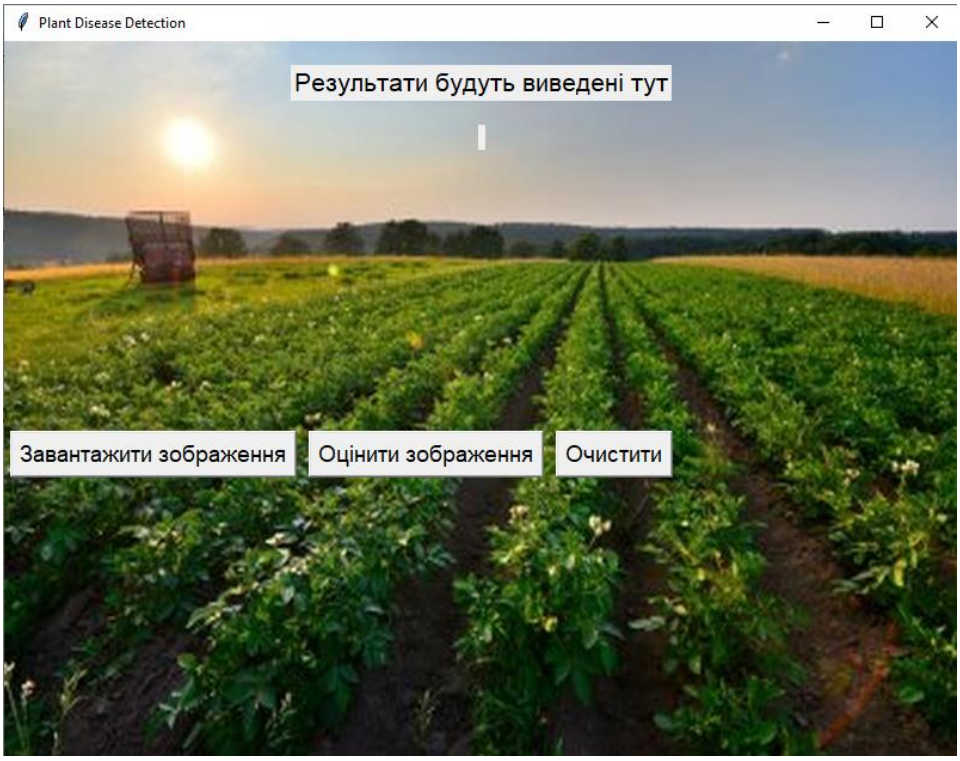

Рисунок 3.6 – Результати створення вікна користувача інтелектуальної інформаційної системи виявлення хвороб рослин

Насамперед передбачено фонову картинку у вигляді зображення, яке відображається в фоні вікна. Виконано етикетки – текстові поля, які відображають інформацію про зображення та результати оцінки. Передбачено кнопки, які дозволяють користувачеві завантажувати зображення, оцінювати його та очищати інтерфейс.

Функція __init__() класу PlantDiseaseUI() ініціалізує всі елементи інтерфейсу. Вона також встановлює розмір вікна та додає фонову картинку. Функція load_image() дозволяє користувачеві вибрати зображення з файлу. Після того, як зображення вибрано, воно відображається в інтерфейсі. Функція predict_image() оцінює зображення за допомогою моделі машинного навчання. Результати оцінки виводяться в інтерфейс. Функція clear() очищає інтерфейс. Вона видаляє зображення та результати оцінки.

Основний код програми запускається у функції if __name__ == "__main__": Він створює вікно та ініціалізує об'єкт класу PlantDiseaseUI(). Потім він запускає цикл основного потоку, який продовжується доти, доки користувач не закриє вікно.

Модель отримує результат оцінки стану рослини. Результат оцінки виводиться в інтерфейс. Функція clear() очищає інтерфейс. Вона виконує наступні дії – видаляє зображення з інтерфейсу та видаляє результати оцінки з інтерфейсу.

3.4. Результати створення інтелектуальної інформаційної системи виявлення хвороб рослин

Нами написано код на Python, який представлено у додатку Б. Він забезпечує за допомогою бібліотеки TensorFlow функціонування інтелектуальної інформаційної системи виявлення хвороб рослин. Інтерфейс аналогічний до попереднього коду, але в цьому випадку він використовує попередньо навчену модель машинного навчання для виявлення хвороб рослин (рис. 3.7). Модель створена на основі архітектури MobileNet і навчена на наборі даних зображень рослин із хворобами.

In [5]:
```
import tkinter as tk
from tkinter import filedialog
from PIL import Image, ImageTk
import tensorflow as tf
import numpy as np

# Завантаження попередньо навченої моделі MobileNet
base_model = tf.keras.applications.MobileNet(weights='imagenet', include_top=False)

# Додаємо шар GlobalAveragePooling2D та Dense шар для класифікації
global_average_layer = tf.keras.layers.GlobalAveragePooling2D()
prediction_layer = tf.keras.layers.Dense(38, activation='softmax')

# Збираємо модель
model = tf.keras.Sequential([
    base_model,
    global_average_layer,
    prediction_layer
])

model.load_weights('plant_disease')
```

Рисунок 3.7 – Код для імпорту потрібних бібліотек та використання попередньо навченої моделі виявлення хвороб рослин

Функція __init__() класу PlantDiseaseDetectionApp() ініціалізує всі елементи інтерфейсу, а також завантажує модель машинного навчання (рис. 3.8).

```
class PlantDiseaseDetectionApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Plant Disease Detection")

        # Ініціалізація елементів інтерфейсу
        self.image_label = tk.Label(self.root)
        self.result_label = tk.Label(self.root, text="Результати будуть виведені тут")

        # Кнопки
        self.load_button = tk.Button(self.root, text="Завантажити зображення", command=self.load_image)
        self.predict_button = tk.Button(self.root, text="Оцінити зображення", command=self.predict_image)
        self.clear_button = tk.Button(self.root, text="Очистити", command=self.clear)

        # Розміщення елементів у вікні
        self.image_label.pack()
        self.result_label.pack()
        self.load_button.pack()
        self.predict_button.pack()
        self.clear_button.pack()

    def load_image(self):
        file_path = filedialog.askopenfilename(title="Виберіть зображення", filetypes=[("Зображення", "*.png;*.jpg;*.jpeg")])
        if file_path:
            # Відображення завантаженого зображення
            image = Image.open(file_path)
            image = image.resize((230, 180))
            self.tk_image = ImageTk.PhotoImage(image)
            self.image_label.config(image=self.tk_image)

            # Збереження зображення для подальшого використання
            self.loaded_image = image

            # Очистка попередніх результатів
            self.result_label.config(text="Результати будуть виведені тут")
```

Рисунок 3.8 – Код функцій __init__() для ініціалізації всіх елементів інтерфейсу та load_image() для вибору користувачем зображення з файлу

Функція load_image() дозволяє користувачеві вибрати зображення з файлу. Після того, як зображення вибрано, воно піддається обробці та передається в модель машинного навчання.

Функція predict_image() отримує результат оцінки від моделі машинного навчання (рис. 3.9). Результат оцінки виводиться в інтерфейсі.

```
def predict_image(self):
    if hasattr(self, 'loaded_image'):
        # Підготовка зображення для передачі в модель
        img_array = tf.keras.preprocessing.image.img_to_array(self.loaded_image)
        img_array = tf.expand_dims(img_array, 0)
        img_array = tf.keras.applications.mobilenet.preprocess_input(img_array)

        # Передача зображення в модель і отримання результатів
        predictions = model.predict(img_array)

        # Визначення класу та ймовірності
        class_index = np.argmax(predictions)
        confidence = predictions[0][class_index]

        # Отримання імені класу з файлу з класами (наприклад, classes.txt)
        classes = [line.strip() for line in open('classes.txt')]
        class_name = classes[class_index]

        # Виведення результатів на інтерфейс
        result_text = f"Клас: {class_name}\nЙмовірність: {confidence:.2%}"
        self.result_label.config(text=result_text)
    else:
        self.result_label.config(text="Спочатку завантажте зображення")

def clear(self):
    # Очистка зображення та результатів
    self.image_label.config(image="")
    self.result_label.config(text="Результати будуть виведені тут")
    delattr(self, 'loaded_image')
```

Рисунок 3.9 – Код функцій predict_image() для оцінки результату та clear() для очищення інтерфейсу

Функція clear() очищає інтерфейс. Вона виконує наступні дії, які стосуються видалення зображення з інтерфейсу, видалення результатів оцінки із інтерфейсу. Ось приклад роботи програми – насамперед слід вибрати зображення:

[Image file (*.png, *.jpg, *.jpeg)]

...

Результати будуть виглядати наступним чином:

Клас: іржа листя

Ймовірність: 99.9%

Якщо користувач не завантажить зображення, то в інтерфейсі буде виведено повідомлення «Спочатку завантажте зображення».

РОЗДІЛ 4.
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА У НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1. Аналіз небезпечних і шкідливих виробничих чинників під час створення інтелектуальної інформаційної системи виявлення хвороб рослин

Створення інтелектуальної інформаційної системи виявлення хвороб рослин (ІІС ВХР) є комплексним завданням, яке вимагає врахування багатьох факторів, у тому числі небезпечних і шкідливих виробничих чинників (НШВЧ). НШВЧ можуть негативно впливати на здоров'я працівників, які задіяні у створенні та експлуатації ІІС ВХР, а також на якість продукції, що виробляється.

До основних НШВЧ, які можуть виникати під час створення ІІС ВХР, належать:

1. Фізичні фактори –
   - електромагнітні поля;
   - шум;
   - вібрація;
   - підвищена або знижена температура;
   - підвищена або знижена вологість;
   - нерівномірне освітлення.
2. Хімічні фактори –
   - шкідливі речовини, що містяться у матеріалах та обладнаннях, які використовуються для створення ІІС ВХР;
   - шкідливі речовини, що утворюються в процесі експлуатації ІІС ВХР.
3. Біологічні фактори –
   - мікроорганізми, які можуть викликати захворювання у працівників.

4.2. Система заходів безпеки під час створення інтелектуальної інформаційної системи виявлення хвороб рослин

Для попередження впливу НШВЧ на працівників під час створення та експлуатації ІІС ВХР необхідно розробити та впровадити систему заходів безпеки, яка включатиме:

1. Технічні заходи –
   - використання обладнання та матеріалів, які відповідають вимогам безпеки;
   - організація робочих місць відповідно до вимог безпеки;
   - забезпечення працівників засобами індивідуального захисту.
2. Санітарно-гігієнічні заходи –
   - проведення систематичних санітарно-гігієнічних обстежень робочих місць;
   - дотримання правил особистої гігієни працівниками.
3. Медико-профілактичні заходи –
   - проведення медичних оглядів працівників;
   - проведення профілактичних заходів щодо попередження захворювань.

Під час розробки та впровадження системи заходів безпеки необхідно враховувати конкретні умови створення та експлуатації ІІС ВХР.

4.3. Особливості оцінки небезпечних і шкідливих виробничих чинників під час створення інтелектуальної інформаційної системи виявлення хвороб рослин

Оцінка небезпечних і шкідливих виробничих чинників під час створення ІІС ВХР має певні особливості. Зокрема, необхідно враховувати наступні фактори. Насамперед це динамічний характер процесу створення ІІС ВХР.

Протягом процесу створення ІІС ВХР можуть змінюватися види та рівні впливу НШВЧ. Необхідно проводити періодичну оцінку небезпечних і шкідливих виробничих чинників.

Слід оцінювати вплив НШВЧ на різні етапи створення ІІС ВХР. НШВЧ можуть впливати на різні етапи створення ІІС ВХР, у тому числі на етапи проектування, виготовлення, монтажу та експлуатації.

Також слід передбачити врахування впливу НШВЧ на різні види робіт, що виконуються при створенні ІІС ВХР. НШВЧ можуть впливати на різні види робіт, що виконуються при створенні ІІС ВХР, у тому числі на роботи з монтажу та обслуговування обладнання, роботи з використанням комп'ютерної техніки, роботи в лабораторіях.

Аналіз небезпечних і шкідливих виробничих чинників під час створення ІІС ВХР є важливим етапом у розробці та впровадженні системи заходів безпеки. Врахування особливостей впливу НШВЧ на різні етапи створення ІІС ВХР дозволяє розробити ефективні заходи безпеки, які забезпечать захист працівників від негативного впливу НШВЧ.

4.4. Розробка логічно-імітаційної моделі процесу виникнення травм під час монтажу мережі

Для побудови логіко-імітаційної моделі процесу, формування і виникнення аварії та травми під час створення інтелектуальної інформаційної системи виявлення хвороб рослин складемо список базових подій. Вони лежатимуть у основі даної моделі. Кожному пункту списку присвоюємо певне значення ймовірності виникнення. Нижче подано сам список:

1. Стан контролю з охорони праці ..... **$P_1 = 0,2$;**
2. Несерйозне відношення до проходження ТО інструменту **$P_2 = 0,1$;**
3. Відсутність комплектуючих установки..... **$P_3 = 0,2$;**
4. Невисока міцність ..... **$P_4 = 0,03$;**

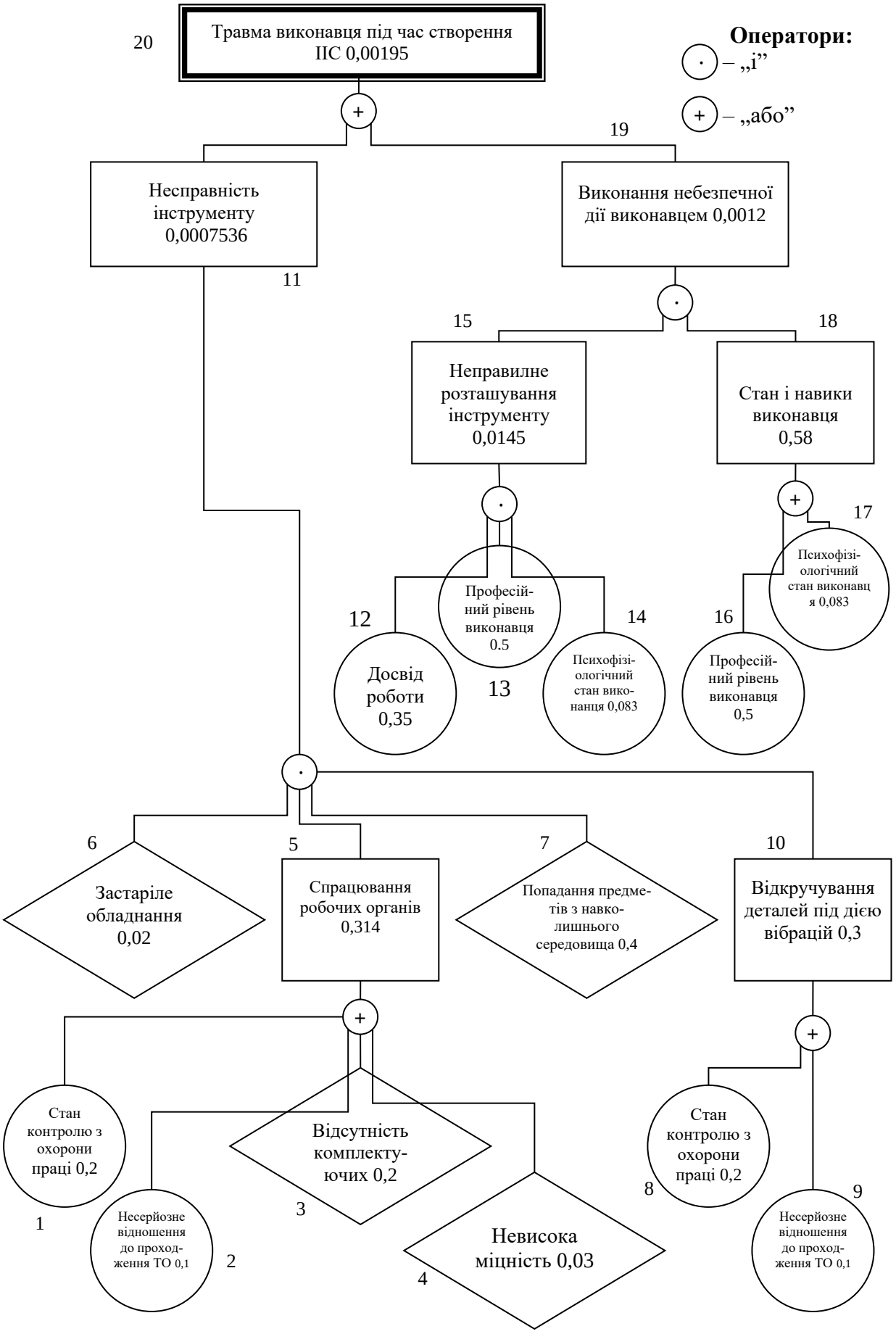

Рис. 4.1. Логіко-імітаційна модель процесу формування та виникнення аварії та травми під час створення інтелектуальної інформаційної системи виявлення хвороб рослин

1. Використання застарілого обладнання..... **P_6 = 0,02;**
2. Попадання сторонніх предметів ..... **P_7 = 0,4;**
3. Досвід роботи виконавця ..... **P_{12} = 0,35.**
4. Професійний рівень виконавця ..... **P_{13} = 0,5;**
5. Психофізіологічний стан виконавця..... **P_{14} = 0,083;**

На основі даного списку будуємо матрицю логічних взаємозв'язків між окремими пунктами, графічне представлення якої зображено на рис. 4.1.

Розрахуємо ймовірності виникнення подій, що входять у дану логіко-імітаційну модель процесу створення інтелектуальної інформаційної системи виявлення хвороб рослин (на прикладі ймовірності отримання травми виконавця).

Ймовірність виникнення події P_5 визначаємо наступним чином:

$$P_5 = 02 + 0{,}1 + 0{,}2 + 0{,}003 - 0{,}2 \cdot 0{,}1 - 0{,}2 \cdot 0{,}03 - 0{,}2 \cdot 0{,}03 - 0{,}1 \cdot 0{,}2 - 0{,}1 \cdot 0{,}03 -$$
$$- 0{,}2 \cdot 0{,}03 + 0{,}2 \cdot 0{,}1 \cdot 0{,}2 + 0{,}1 \cdot 0{,}2 \cdot 0{,}03 + 0{,}2 \cdot 0{,}1 \cdot 0{,}2 + 0{,}2 \cdot 0{,}1 \cdot 0{,}03 -$$
$$- 0{,}2 \cdot 0{,}1 \cdot 0{,}2 \cdot 0{,}03 = 0{,}314$$

Ймовірність виникнення події P_{10} визначаємо так:

$$P_{10} = 0{,}2 + 0{,}1 = 0{,}3.$$

Ймовірність виникнення події P_{11} визначаємо:

$$P_{11} = 0{,}02 \cdot 0{,}314 \cdot 0{,}4 \cdot 0{,}3 = 0{,}00075.$$

Ймовірність виникнення події P_{15} визначаємо наступним чином:

$$P_{15} = 0{,}35 \cdot 0{,}5 \cdot 0{,}083 = 0{,}0145.$$

Ймовірність події P_{18}:

$$P_{18} = 0{,}5 + 0{,}083 = 0{,}58.$$

Ймовірність події P_{19}:

$$P_{19} = 0{,}0145 \cdot 0{,}083 = 0{,}0012.$$

Ймовірність події P_{20}:

$$P_{20} = 0{,}00075 + 0{,}0012 = 0{,}00195.$$

4.5. Розробка заходів із забезпечення безпеки під час надзвичайних ситуацій

Розробка заходів із забезпечення безпеки під час надзвичайних ситуацій є важливим завданням, яке має бути виконано для захисту людей і майна від шкоди. Ці заходи повинні бути розроблені з урахуванням конкретних умов, у яких може виникнути надзвичайна ситуація, а також потенційних загроз, які можуть виникнути.

Заходи із забезпечення безпеки під час надзвичайних ситуацій можна розділити на кілька основних категорій. Профілактичні заходи – ці заходи спрямовані на запобігання виникненню надзвичайних ситуацій. До них належать, наприклад, заходи з технічної безпеки, заходи з охорони праці, заходи з пожежної безпеки та інші.

До профілактичних заходів безпеки під час надзвичайних ситуацій належать заходи реагування, які спрямовані на надання допомоги людям і ліквідацію наслідків надзвичайних ситуацій. До них належать, наприклад, заходи з евакуації людей, заходи з надання першої медичної допомоги, заходи з ліквідації пожеж, заходів з ліквідації наслідків аварій та інших.

Заходи відновлення спрямовані на відновлення життя і діяльності людей після надзвичайної ситуації. До них належать, наприклад, заходи з відновлення житла, заходів з відновлення інфраструктури, заходів з відновлення виробництва та інших.

Розробка заходів із забезпечення безпеки під час надзвичайних ситуацій повинна бути комплексним процесом, який включає в себе наступні етапи:

1. Аналіз загроз – на цьому етапі необхідно визначити потенційні загрози, які можуть призвести до надзвичайної ситуації. Цей аналіз повинен враховувати всі можливі джерела загроз, у тому числі природні, техногенні та соціальні;

2. Оцінка ризиків – на цьому етапі необхідно оцінити ймовірність і наслідки виникнення надзвичайної ситуації. Ця оцінка повинна бути заснована на результатах аналізу загроз;

3. Розробка заходів безпеки – на цьому етапі необхідно розробити конкретні заходи, які будуть спрямовані на запобігання, реагування та відновлення після надзвичайної ситуації. Ці заходи повинні бути ефективними та економічно доцільними.

4. Впровадження заходів безпеки – на цьому етапі необхідно забезпечити впровадження розроблених заходів. Це включає в себе навчання персоналу, забезпечення ресурсами та інші заходи.

Ефективна розробка заходів із забезпечення безпеки під час надзвичайних ситуацій є важливим фактором, який може допомогти захистити людей і майно від шкоди.

.

РОЗДІЛ 5.
ЕКОНОМІЧНА ЕФЕКТИВНІСТЬ ВІД РОЗРОБКИ ІНТЕЛЕКТУАЛЬНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ВИЯВЛЕННЯ ХВОРОБ РОСЛИН ІЗ ВИКОРИСТАННЯМ ТЕХНОЛОГІЇ DEEP LEARNING

В сучасних умовах розробка і впровадження інтелектуальних інформаційних систем на основі технології глибокого навчання (Deep Learning) стає важливим інструментом для сільськогосподарського сектору. Одним з ключових напрямків є виявлення хвороб рослин за допомогою таких систем. У цьому розділі ми розглянемо економічну ефективність такого підходу та проведемо розрахунок показників ефективності.

Задача розробки інтелектуальної інформаційної системи полягає в автоматизованому виявленні хвороб рослин на ранніх стадіях за допомогою аналізу зображень. Модель глибокого навчання буде тренуватися на наборі даних, який включає в себе зображення рослин з різними хворобами та їхніми анотаціями.

Для оцінки економічної ефективності розробки системи виявлення хвороб рослин розглянемо наступні показники:

1) Вартість розробки системи (CR) – сума коштів, витрачених на розробку інтелектуальної системи, включаючи витрати на програмування, обладнання та навчання моделей. Формула для визначення вартості розробки системи CR може мати вигляд:

$$CR = C_{пр} + C_{об} + C_{нав}, \qquad (5.1)$$

де $C_{пр}$ – витрати на програмування, грн.; $C_{об}$ – витрати на придбання необхідного обладнання, грн.; $C_{нав}$ – витрати на проведення навчання моделей машинного навчання, грн.

Підставивши відповідні значення у формулу (5.1) отримаємо:

$$CR = 45000 + 38000 + 15000 = 98000 \text{ грн.}$$

2) Прибуток від використання системи (PR) – очікуваний прибуток від впровадження системи в сільському господарстві, зменшення витрат на лікування рослин та підвищення врожайності. Формула має вигляд:

$$PR = (B_{зв} + B_{пв}) \cdot T, \qquad (5.2)$$

де $B_{зв}$ – економія коштів від захисту рослин, що досягається завдяки використанню системи, грн.; $B_{пв}$ – додатковий дохід від збільшення врожайності завдяки вчасному виявленню хвороб та їхньому ефективному контролю, грн.; Т – тривалість часу, на який розраховується прибуток, років.

Підставивши відповідні значення у формулу (5.2) отримаємо:

$$PR = (82000 + 94000) \cdot 1 = 176000 \text{ грн.}$$

Ця формула дозволяє оцінити очікуваний прибуток від впровадження інтелектуальної системи для виявлення хвороб рослин в сільському господарстві.

3) Показник віддачі інвестицій (ROI) – відношення прибутку до витрат, що визначає ефективність вкладених коштів. Формула для обчислення ROI має вигляд:

$$ROI = \frac{PR - CR}{CR} \cdot 100, \qquad (5.3)$$

де PR – прибуток, отриманий від використання системи (наприклад, прибуток від підвищення врожайності та економія витрат на лікування рослин), грн.; CR – сума коштів, витрачених на розробку та впровадження системи, грн.

Ця формула дозволяє визначити, наскільки ефективно були вкладені кошти в розробку і використання інтелектуальної системи для виявлення хвороб рослин в сільському господарстві. Високий показник ROI свідчить про ефективність інвестицій, тоді як низький може вказувати на потребу вдосконалення стратегії впровадження системи або зменшення витрат на її розробку.

Виконаємо розрахунок показників економічної ефективності від розробки інтелектуальної інформаційної системи виявлення хвороб рослин із

використанням технології Deep Learning. Показник віддачі інвестицій (ROI) становитиме:

$$ROI = \frac{176000 - 98000}{98000} \cdot 100 = 79,6\%.$$

Таблиця 5.1 – Результати оцінки економічної ефективності розробки системи виявлення хвороб рослин

№ з/п	Показник	Одиниця виміру	Значення
1	Витрати на програмування	грн.	45000
2	Витрати на придбання необхідного обладнання	грн.	38000
3	Витрати на проведення навчання моделей машинного навчання	грн.	15000
4	Витрати на розробку та впровадження системи	грн.	98000
5	Економія коштів від захисту рослин, що досягається завдяки використанню системи	грн.	82000
6	Додатковий дохід від збільшення врожайності завдяки вчасному виявленню хвороб та їхньому ефективному контролю	грн.	94000
7	Прибуток, отриманий від використання системи	грн.	176000
8	Показник віддачі інвестицій (ROI)	%	79,6
9	Термін окупності капіталовкладень	років	0,56

4) Термін окупності капіталовкладень (PP) визначає час, протягом якого інвестиції повертаються. Формула для розрахунку терміну окупності має вигляд:

$$PP = \frac{CR}{PR}, \tag{5.4}$$

Отримаємо:

$$PP = \frac{98000}{176000} = 0,56 \text{ року}.$$

ROI у даному прикладі складає 79,6%, що означає, що кожна витрачену гривню буде отримано приносить 79,6 грн. прибутку. Термін окупності капіталовкладень становить 0,56 року. Це є позитивним показником ефективності інвестицій.

Розробка і впровадження інтелектуальної інформаційної системи для виявлення хвороб рослин з використанням технології Deep Learning може бути економічно ефективною для сільськогосподарського сектору. Розрахунки показників ефективності допомагають оцінити вигоди від впровадження такої системи та прийняти обґрунтовані рішення щодо інвестицій в її розробку.

ВИСНОВКИ І ПРОПОЗИЦІЇ

Сучасний стан виявлення хвороб у сільськогосподарських рослин потребує використання точних та ефективних підходів. Традиційні методи мають обмежену точність та швидкість реакції на захворювання, що може призвести до великих втрат врожаю та ресурсів. Глибке навчання (Deep Learning) є потужною технологією, яка показала вражаючі результати в різних сферах, включаючи виявлення хвороб рослин. Використання нейронних мереж зі здатністю до автоматичного вивчення патернів дозволяє підвищити точність та швидкість діагностики.

На підставі аналізу встановлено, що методи глибокого навчання досягли кращих результатів, ніж попередні методи машинного навчання, у таких завданнях, як обробка природної мови, класифікація зображень і розпізнавання обличчя. Отже, фактор успіху методів глибокого навчання значною мірою залежить від їх здатності формувати нелінійні та складні зв’язки з даними.

Нами виконано аналіз технологій машинного навчання для виявлення хвороб у рослин. Незважаючи на те, що дослідники використовували різноманітні архітектури та інноваційні стратегії, включаючи механізми привернення уваги, перенесення навчання та збільшення даних, щоб досягти надзвичайної точності ідентифікації захворювань, задачі, пов’язані з доступністю даних та інтерпретацією моделі, залишаються ще невирішеними. Успіхи, досягнуті в цій галузі, мають величезні перспективи для сільськогосподарської практики, пропонуючи потенціал для своєчасної ідентифікації хвороб, покращення управління виконанням робіт із захисту культур та, зрештою, підвищення продуктивності сільського господарства.

Нами виконано аналіз інтелектуальних систем для виявлення хвороб рослин. Встановлено, що існують системи підтримки прийняття рішень у режимі реального часу (рис. 1.11), забезпечують отримання зображень для заражених і нормальних рослин (рис. 1.12), передбачають використання мобільних додатків для оцінення стану рослин (рис. 1.13-1.14). Усі вони мають

вагоме значення для практики. Однак, їх алгоритми пристосовані під вирішення конкретних практичних задач і є дорого вартісними.

Виходячи з вищевикладеного, слід зазначити що розробка інтелектуальної інформаційної системи виявлення хвороб рослин із використанням технології Deep Learning є доцільною і може принести значні вигоди для сільськогосподарських виробників.

Встановлено, що технології Deep Learning, такі як CNN і DBN, сьогодні використовуються для виявлення аномалій рослин і заражень їх хворобами. Ці технології демонструють точні результати у виявленні та ідентифікації уражень на цифрових зображеннях. У таблиці 2.1 надано вичерпну інформацію про інструменти та технології, що використовуються для виявлення хвороб рослин. Вона містить детальну інформацію про різні методи вилучення ознак, в тому числі ті, що базуються на ручних і навчальних ознаках, а також відповідні методи обробки малих і великих наборів даних зображень рослин.

Нами виконано вибір інструментарію Deep Learning. Для розробки моделі виявлення хвороб рослин із використанням технології Deep Learning використано наступні бібліотеки: Os; Numpy; Pandas; Torch; matplotlib.pyplot. Окрім того, використовували спеціальні бібліотеки PyTorch, до яких належать: torch.nn; torch.utils.data; torch.nn.functional; torchvision.transforms; torchvision.utils; torchvision.datasets. Створення моделі виконуємо у Jupyter Notebook.

Нами виконано збір та підготовка даних для створення моделі виявлення хвороб рослин. Використано набір даних PlantVillage. Він складається з близько 87 000 зображень здорових і хворих листків, розділених на 38 категорій за видами і хворобами. У наборі даних доступні наступні 38 класів зображень листків сільськогосподарських рослин.

Нами виконана попередня обробка зображень (рис. 2.4). Зокрема, нами встановлено цільовий розмір для зображень, розмір яких буде змінено до 224x224 пікселів – image_size = 224: Це звичайний розмір, який використовується для завдань класифікації зображень.

Після цього нами задано параметри навчання. Зокрема, вказано кількість зображень – batch_size = 32, які потрібно обробляти разом на кожному кроці навчання. Розмір пакета 32 є звичайним вибором, який збалансовує ефективність і використання пам’яті. Встановлено кількість разів – epochs = 25, коли модель проходитиме весь навчальний набір даних. Кожна епоха передбачає пакетну обробку всіх доступних навчальних зображень. Значення епохи 25 свідчить про помірну тривалість навчання.

Нами використано flow_from_directory() для створення пакетів даних зображень (і їхніх міток) безпосередньо з наших зображень у відповідних теках. Розділення наборів для навчання та перевірки має вирішальне значення для оцінки продуктивності моделі на невидимих даних.

Нами використано модель MobileNet, яка є архітектурою нейронної мережі, розробленою компанією Google для використання в мобільних пристроях. Спочатку нами створено базову модель MobileNet без верхніх шарів, оскільки хочемо використати її для 38 класів, а також попередньо навчені ваги для ImageNet (рис. 3.1). Після цього нами створено наступну модель поверх MobileNet, використовуючи функціональний API (рис. 3.2).

Результати виконаного навчання моделі та визначення її оцінок представлено у табл. 3.1. Встановлено, модель досягла високої точності на наборі даних для перевірки, досягнувши 95% точності в 25-й епосі навчання. Втрати на наборі даних для перевірки також були низькими, досягнувши 0,15 у 25-й епосі. Модель демонструє хорошу стабільність у своїх результатах, оскільки втрати та точність на наборі даних для перевірки не змінюються значно після 20-ї епохи. Отриману модель зберігаємо model.save('plant_disease') у стандартному форматі TensorFlow 2 SavedModel. У подальшому її використовуємо у інтелектуальній інформаційній системі виявлення хвороб рослин.

Нами написано код, який представлено у додатку А, який створює інтерфейс користувача для виявлення хвороб рослин. Він написаний на Python за допомогою бібліотеки Tkinter. Інтерфейс користувачів інтелектуальної

інформаційної системи виявлення хвороб рослин представлено на рис. 3.6 і він складається із декількох елементів.

Нами написано код на Python, який представлено у додатку Б. Він забезпечує за допомогою бібліотеки TensorFlow функціонування інтелектуальної інформаційної системи виявлення хвороб рослин. Інтерфейс аналогічний до попереднього коду, але в цьому випадку він використовує попередньо навчену модель машинного навчання для виявлення хвороб рослин (рис. 3.7). Модель створена на основі архітектури MobileNet і навчена на наборі даних зображень рослин із хворобами.

Запропоновано заходи щодо охорони праці, які дають можливість створити належні умови праці під час створення та використання інтелектуальної інформаційної системи для виявлення хвороб рослин.

Розробка і впровадження інтелектуальної інформаційної системи для виявлення хвороб рослин з використанням технології Deep Learning може бути економічно ефективною для сільськогосподарського сектору. ROI у даному прикладі складає 79,6%, що означає, що кожна витрачену гривню буде отримано приносить 0,79 грн. прибутку. Термін окупності капіталовкладень становить 0,56 року. Це є позитивним показником ефективності інвестицій.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Введення в машинне навчання за допомогою Python и Scikit-Learn. URL: https://habr.com/ua/company/mlclass/blog/247751/ (дата звернення: 11.10.2024).

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

3. Класифікація в Python з Scikit-Learn та Pandas. URL: https://stackabuse.com/classification-in-pythonwith-scikit-learn-and-pandas/ (дата звернення: 17.08.2023).

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

5. Навчання нейромережі з учителем, без вчителя, з підкріпленням – у чому відмінність? URL: https://neurohive.io/ru/osnovy-data-science/obuchenie-s-uchitelem-bez-uchitelja-s-podkrepleniem/(дата звернення: 08.10.2024).

6. Плескач В.Л., Рогушина Ю.В., Кустова Н.П. Інформаційні технології та системи. К.: Книга, 2004. 519 с.

7. Tryhuba A., Ivanyshyn V., Chaban V., Mushenyk I., Zharikova O. Computer model of resource demand planning for dairy farms. Independent Journal of Management & Production (Special Edition ISE, S&P). 2021. 12(3), pp. 138-149. URL: http://www.ijmp.jor.br/index.php/ijmp/article/view/1531/1971 (Last accessed: 21.10.2024).

8. Tryhuba, A., Boyarchuk, V., Tryhuba, I., Ftoma, O., Padyuka, R., Rudynets, M. Forecasting the Risk of the Resource Demand for Dairy Farms Basing on Machine Learning. Proceedings of the 2nd International Workshop on Modern Machine Learning Technologies and Data Science (MoMLeT+DS 2020). 2020. I. P. 327-340.

9. Koval N., Tryhuba A., Kondysiuk I., Tryhuba I., Boiarchuk O., Rudynets M., Grabovets V., Onyshchuk V., Forecasting the Fund of Time for Performance of Works in Hybrid Projects Using Machine Training Technologies.

Proceedings of the 3nd International Workshop on Modern Machine Learning Technologies and Data Science Workshop. Proc. 3rd International Workshop (MoMLeT&DS 2021). Volume I: Main Conference. Lviv-Shatsk, Ukraine, June 5-6, 2021. pp.196-206.

10. Hutsol T., Glowacki S., Tryhuba A. Current Trends of Biohydrogen Production from Biomass – Green Hydrogen. Monograph. Warsaw: 2021. 102 p.

11. Tryhuba A., Zachko O., Grabovets V., Berladyn O., Pavlova I., Rudynets M. Examining the effect of production conditions at territorial logistic systems of milk harvesting on the parameters of a fleet of specialized road tanks. Eastern-European Journal of Enterprise Technologies. 2018. 5(3). P. 59-70. URL: http://nbuv.gov.ua/UJRN/Vejpte_2018_5(3)__7. (Last accessed: 17.10.2024).

12. Tryhuba, A., Kondysiuk, I., Tryhuba, I., Boiarchuk, O., Tatomyr, A., Intellectual information system for formation of portfolio projects of motor transport enterprises. CEUR Workshop Proceedings, 2022, 3109, pp. 44–52.

13. 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, 3426, pp. 122–135.

14. Pacal I. Enhancing crop productivity and sustainability through disease identification in maize leaves: Exploiting a large dataset with an advanced vision transformer model. Expert Systems with Applications. Vol. 238, Part D, 15 March 2024, 122099. https://doi.org/10.1016/j.eswa.2023.122099

15. La Torre J. de, Puig D., Valls A. Weighted kappa loss function for multi-class classification of ordinal data in deep learning. Pattern Recognit Lett, 105 (2018), pp. 144-154.

16. Senauer B., Vaclav S. Feeding the World: A Challenge for the Twenty-First Century. [(accessed on 5 January 2015)]. Available online: http://onlinelibrary.wiley.com/doi/10.1111/j.1728-4457.2000.00827.x/pdf

17. Rosset P. Food sovereignty and the contemporary food crisis. Development. 2008;51:460–463. doi: 10.1057/dev.2008.48.

18. Godfray H.C.J., Beddington J.R., Crute I.R., Haddad L., Lawrence D., Muir J.F., Pretty J., Robinson S., Thomas S.M., Toulmin C. Food security: The challenge of feeding 9 billion people. Science. 2010;327:812–818. doi: 10.1126/science.1185383.

19. Conway G. One Billion Hungry: Can We Feed the World? Cornell University Press; Ithaca, NY, USA: 2012.

20. Savary S., Ficke A., Aubertot J., Hollier C. Crop losses due to diseases and their implications for global food production losses and food security. Food Secur. 2012;4:519–537. doi: 10.1007/s12571-012-0200-5

21. Oerke E.-C. Crop losses to pests. J. Agric. Sci. 2006;144:31–43. doi: 10.1017/S0021859605005708.

22. Pimentel D., Zuniga R., Morrison D. Update on the environmental and economic costs associated with alien-invasive species in the United States. Ecol. Econ. 2005;52:273–288. doi: 10.1016/j.ecolecon.2004.10.002.

23. Актуальна інформація про хід робіт із захисту рослин. URL: https://infoindustria.com.ua/aktualna-informacziya-pro-hid-robit-iz-zahistu-roslin/

24. Vision Transformer. URL: https://paperswithcode.com/method/vision-transformer

25. Chen J. , Zhang D. , Suzauddola M. , Zeb A. Identifying crop diseases using attention embedded MobileNet-V2 model. Applied Soft Computing, 113 (2021), 10.1016/j.asoc.2021.107901

26. Wu H. , Fang L. , Yu Q. , Yuan J. , Yang C. Plant leaf identification based on shape and convolutional features. Expert Systems with Applications, 219 (2023), 10.1016/j.eswa.2023.119626

27. Subramanian M. , Shanmugavadivel K. , Nandhini P.S. On fine-tuning deep learning models using transfer learning and hyper-parameters optimization for disease identification in maize leaves. Neural Computing and Applications, 34 (16) (2022), pp. 13951-13968, 10.1007/s00521-022-07246-w

28. Haque M.A. , Marwaha S. , Deb C.K. , Nigam S. , Arora A. Recognition of diseases of maize crop using deep learning models. Neural

Computing and Applications, 35 (10) (2023), pp. 7407-7421, 10.1007/s00521-022-08003-9

29. Alagumariappan P., Dewan N.J. Intelligent Plant Disease Identification System Using Machine Learning. Presented at the 7th International Electronic Conference on Sensors and Applications, 15–30 November 2020. *Eng. Proc.* 2020, *2*(1), 49; https://doi.org/10.3390/ecsa-7-08160

30. Tripathy S.S., Poddar R., Satapathy L., Mukhopadhyay K. Chapter 35 - Image processing–based artificial intelligence system for rapid detection of plant diseases. Bioinformatics in Agriculture. 2022, P. 619-624. https://doi.org/10.1016/B978-0-323-89778-5.00023-4

31. Tryhuba A., Tryhuba I., Bashynsky O., et al., Conceptual model of management of technologically integrated industry development projects, Proceedings of the 15th International Scientific and Technical Conference on Computer Sciences and Information Technologies, Lviv Ukraine, 2020, pp. 155-158. doi: 10.1109/CSIT49958.2020.9321903

32. Tryhuba A., Boyarchuk V., Tryhuba I., Boyarchuk O., Ftoma O., Evaluation of Risk Value of Investors of Projects for the Creation of Crop Protection of Family Daily Farms. Acta universitatis agriculturae et silviculturae mendelianae brunensis 67(5) (2019) 1357-1367. URL: https://doi:10.11118/actaun201967051357

33. Tryhuba A., Rudynets M., Pavlikha N., Kytsyuk I., Komeliuk O., Fedorchuk-Moroz V., Androshchuk I., Skorokhod I., Seleznov D., Establishing patterns of change in the indicators of using milk processing shops at a community territory. Eastern-European Journal of Enterprise Technologies: Control processes 6(3 (102) (2019) 57–65. URL: https://doi.org/10.15587/1729-4061.2019.184508

34. Tryhuba A., Boyarchuk V., Tryhuba I., Boiarchuk O., Pavlikha N., Kovalchuk N., Study of the impact of the volume of investments in agrarian projects on the risk of their value. CEUR Workshop Proceedings 2851 (2021) 303–313.

35. Tryhuba A., Boyarchuk V., Tryhuba I., et al., Method and Software of Planning of the Substantial Risks in the Projects of Production of raw Material for Biofuel. CEUR Workshop Proceedings, 2020, pp. 116-129.

36. Tryhuba, A.; Ivanyshyn, V.; Chaban, V. Influence of agrometeorological component of the project environment on the duration of works in chemical protection projects of agricultural crops. Independent Journal of Management & Production (Special Edition ISE, S&P), v. 12, n. 3, p. 138-149. DOI: https://doi.org/10.14807/ijmp.v12i3.1531.

37. Tryhuba, A.; Bashynsky, O. (2019) Coordination of dairy workshops projects on the community territory and their project environment. In: 14-th International Scientific and Technical Conference on Computer Sciences and Information Technologies. Lviv Polytechnic National University, 17–20 September. Lviv, pp. 51–54.

38. Tryhuba, A.; Hridin, O.; Slavina, N.; Mushenyk, I. & Dobrovolska, E. (2020) Managerial decisions in logistic systems of milk provision on variable production conditions. Independent Journal of Management & Production, Vol 11. No 8, pp. 783-800.

39. Tryhuba, A.; Bashynskyi, O.; Medvediev, Y.; Slobodian, S. & Skorobogatov, D. (2019) Justification of models of changing project environment for harvesting grain, oilseed and legume crops. Independent Journal of Management & Production, Vol 10. No 7, pp. 658-672.

40. Boyarchuk, V. & Ftoma, O. (2019) Forecasting of a lifecycle of the projects of production of biofuel raw materials with consideration of risks. International Conference on Advanced Trends in Information Theory (ATIT), pp. 420-425.

41. Boyarchuk, V., & Ftoma, O. 2019. Evaluation of risk value of investors of projects for the creation of crop protection of family dairy farms. Acta Universitatis Agriculturae et Silviculturae Mendelianae Brunensis, 67(5): 1357–1367.

42. Kurmi, Y., Gangwar, S. (2022). A leaf image localization based algorithm for different crops disease classification. Inf. Process. Agric. 9 (3), 456–474.

43. Hughes, David P., and Marcel Salathe. An Open Access Repository of Images on Plant Health to Enable the Development of Mobile Disease Diagnostics. ArXiv:1511.08060 [Cs], Apr. 2016. arXiv.org, http://arxiv.org/abs/1511.08060.

ДОДАТКИ

Додаток А

Код створення вікна інтелектуальної інформаційної системи виявлення хвороб рослин із використанням технології Deep Learning

```
import tkinter as tk
from tkinter import filedialog
from PIL import Image, ImageTk

class PlantDiseaseUI:
def __init__(self, root):
self.root = root
self.root.title("Plant Disease Detection")
self.root.geometry("800x600") # Вікно у два рази менше

# Додавання фонової картинки
background_image = Image.open("background.jpg") # Замініть "background.jpg" на шлях до вашого файлу
background_image = background_image.resize((800, 600), Image.LANCZOS)

self.background_photo = ImageTk.PhotoImage(background_image)
background_label = tk.Label(self.root, image=self.background_photo)
background_label.place(x=0, y=0, relwidth=1, relheight=1)

# Ініціалізація елементів інтерфейсу
self.image_label = tk.Label(self.root)
self.result_label = tk.Label(self.root, text="Результати будуть виведені тут", font=("Helvetica", 16))

# Кнопки
self.load_button = tk.Button(self.root, text="Завантажити зображення", command=self.load_image, font=("Helvetica", 14))
self.predict_button = tk.Button(self.root, text="Оцінити зображення", command=self.predict_image, font=("Helvetica", 14))
self.clear_button = tk.Button(self.root, text="Очистити", command=self.clear, font=("Helvetica", 14))

# Розміщення елементів у вікні
self.result_label.pack(side=tk.TOP, pady=20)
self.image_label.pack()

# Розміщення кнопок у ряд горизонтально знизу
self.load_button.pack(side=tk.LEFT, padx=5, pady=5)
self.predict_button.pack(side=tk.LEFT, padx=5, pady=5)
self.clear_button.pack(side=tk.LEFT, padx=5, pady=5)


def load_image(self):
file_path = filedialog.askopenfilename(title="Виберіть зображення", filetypes=[("Зображення", "*.png;*.jpg;*.jpeg")])
if file_path:
```

```
# Відображення завантаженого зображення
image = Image.open(file_path)
image = image.resize((int(self.root.winfo_screenwidth() * 0.8), int(self.root.winfo_screenheight() *
0.6)))
self.tk_image = ImageTk.PhotoImage(image)
self.image_label.config(image=self.tk_image)

# Збереження зображення для подальшого використання
self.loaded_image = image

# Очистка попередніх результатів
self.result_label.config(text="Результати будуть виведені тут", font=("Helvetica", 16))

def predict_image(self):
if hasattr(self, 'loaded_image'):
# Тут слід додати код для передачі зображення в модель та отримання результатів
# Наприклад, викликати функцію, яка обробляє зображення

# Тимчасова імітація результатів
class_name = "Примітка"
confidence = 0.85

# Виведення результатів на інтерфейс
result_text = f"Клас: {class_name}\nЙмовірність: {confidence:.2%}"
self.result_label.config(text=result_text, font=("Helvetica", 16))
else:
self.result_label.config(text="Спочатку завантажте зображення", font=("Helvetica", 16))

def clear(self):
# Очистка зображення та результатів
self.image_label.config(image="")
self.result_label.config(text="Результати будуть виведені тут", font=("Helvetica", 16))
delattr(self, 'loaded_image')

if __name__ == "__main__":
root = tk.Tk()
app = PlantDiseaseUI(root)
root.mainloop()
```

Додаток Б

Код інтелектуальної інформаційної системи виявлення хвороб рослин із використанням технології Deep Learning

```
import tkinter as tk
from tkinter import filedialog
from PIL import Image, ImageTk
import tensorflow as tf
import numpy as np
# Завантаження попередньо навченої моделі MobileNet
base_model = tf.keras.applications.MobileNet(weights='imagenet', include_top=False)
# Додаємо шар GlobalAveragePooling2D та Dense шар для класифікації
global_average_layer = tf.keras.layers.GlobalAveragePooling2D()
prediction_layer = tf.keras.layers.Dense(38, activation='softmax')
# Збираємо модель
model = tf.keras.Sequential([
base_model,
global_average_layer,
prediction_layer
])
model.load_weights('plant_disease')

class PlantDiseaseDetectionApp:
def __init__(self, root):
self.root = root
self.root.title("Plant Disease Detection")

# Ініціалізація елементів інтерфейсу
self.image_label = tk.Label(self.root)
self.result_label = tk.Label(self.root, text="Результати будуть виведені тут")

# Кнопки
self.load_button = tk.Button(self.root, text="Завантажити зображення",
command=self.load_image)
self.predict_button = tk.Button(self.root, text="Оцінити зображення",
command=self.predict_image)
self.clear_button = tk.Button(self.root, text="Очистити", command=self.clear)

# Розміщення елементів у вікні
self.image_label.pack()
self.result_label.pack()
self.load_button.pack()
self.predict_button.pack()
self.clear_button.pack()

def load_image(self):
file_path = filedialog.askopenfilename(title="Виберіть зображення", filetypes=[("Зображення",
"*.png;*.jpg;*.jpeg")])
if file_path:
```

```
# Відображення завантаженого зображення
image = Image.open(file_path)
image = image.resize((230, 180))
self.tk_image = ImageTk.PhotoImage(image)
self.image_label.config(image=self.tk_image)

# Збереження зображення для подальшого використання
self.loaded_image = image

# Очистка попередніх результатів
self.result_label.config(text="Результати будуть виведені тут")

def predict_image(self):
if hasattr(self, 'loaded_image'):
# Підготовка зображення для передачі в модель
img_array = tf.keras.preprocessing.image.img_to_array(self.loaded_image)
img_array = tf.expand_dims(img_array, 0)
img_array = tf.keras.applications.mobilenet.preprocess_input(img_array)

# Передача зображення в модель і отримання результатів
predictions = model.predict(img_array)

# Визначення класу та ймовірності
class_index = np.argmax(predictions)
confidence = predictions[0][class_index]

# Отримання імені класу з файлу з класами (наприклад, classes.txt)
classes = [line.strip() for line in open('classes.txt')]
class_name = classes[class_index]

# Виведення результатів на інтерфейс
result_text = f"Клас: {class_name}\nЙмовірність: {confidence:.2%}"
self.result_label.config(text=result_text)
else:
self.result_label.config(text="Спочатку завантажте зображення")

def clear(self):
# Очистка зображення та результатів
self.image_label.config(image="")
self.result_label.config(text="Результати будуть виведені тут")
delattr(self, 'loaded_image')


if __name__ == "__main__":
root = tk.Tk()
app = PlantDiseaseDetectionApp(root)
root.mainloop()
```