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: **«Моніторинг стану рослин у теплицях на основі алгоритмів глибокого навчання для сегментації»**

Виконав: студент групи Іт-62

Спеціальності 126 «Інформаційні системи та технології»
(шифр і назва)
Голинський Руслан-Богдан Васильович
(Прізвище та ініціали)

Керівник: к.т.н., доцент Боярчук О.В.
(Прізвище та ініціали)

Рецензент: к.т.н., доцент Тимочко В.О.
(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

Голинському Руслану-Богдану Васильовичу

1. Тема роботи: «Моніторинг стану рослин у теплицях на основі алгоритмів глибокого навчання для сегментації»

Керівник роботи Боярчук Олег Віталійович, доцент
затверджені наказом по університету від 12.09.2024 року № 616/к-с.

2. Строк подання студентом роботи 10.01.2024 р.

3. Вихідні дані до роботи: *дані для моніторингу стану рослин у теплицях на основі алгоритмів глибокого навчання; методика глибокого навчання.*
4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити)
Вступ.
1. Аналіз стану теплиць та обґрунтування доцільності використання алгоритмів глибокого навчання для сегментації.
2. Характеристика об'єкту дослідження та вибір засобів.
3. Результати налаштування моделей та їх використання для моніторингу стану рослин у теплицях на основі алгоритмів глибокого навчання.
4. Охорона праці та безпека у надзвичайних ситуаціях.
5. Економічна ефективність від моніторингу стану рослин у теплицях на основі алгоритмів глибокого навчання для сегментації.
Висновки та пропозиції.
Список використаної літератури.

5. Перелік ілюстраційного матеріалу (з точним зазначенням обов'язкових слайдів): *аналіз стану теплиць та обґрунтування доцільності використання алгоритмів глибокого навчання для сегментації; характеристика об'єкту дослідження та вибір засобів; результати налаштування моделей та їх використання для моніторингу стану рослин у теплицях на основі алгоритмів глибокого навчання; визначення економічної ефективності.*

6. Консультанти з розділів:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1, 2, 3, 5	*Боярчук О.В., доцент 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 _____ Голинський Р.-Б.В.
(підпис)

Керівник роботи _____ Боярчук О.В.
(підпис)

УДК 004.9:631.52

Моніторинг стану рослин у теплицях на основі алгоритмів глибокого навчання для сегментації.

Голинський Р.-Б.В. Кафедра інформаційних технологій – Дубляни, ЛНУП, 2024.

Кваліфікаційна робота: 78 с. текст. част., 22 рис., 8 табл., 15 арк. ілюстраційного матеріалу, 41 джерел0.

Обґрунтована доцільність моніторингу стану рослин у теплицях на основі машинного навчання. Подано особливості моніторингу стану рослин у теплицях на основі глибокого навчання. Виконано аналіз існуючих систем моніторингу стану рослин у теплицях. Обґрунтовано доцільність досліджень щодо моніторингу стану рослин у теплицях на основі алгоритмів глибокого навчання для сегментації.

Здійснено формулювання задачі дослідження. Наведено особливості виконання сегментації рослин, виявлення стану здоров’я рослин. Здійснено вибір моделей для сегментації рослин.

Проведено налаштування моделей YOLOv9, SAM, DINOv2 для семантичної сегментації. Здійснено налаштування на виявлення хвороб рослин, підрахунків листів, обчислення висоти рослин та сегментації фону. Подано результати сегментації листя однієї рослини, сегментації листя в умовах теплиці.

Ключові слова: моніторинг стану рослин, теплиці, алгоритми глибокого навчання, сегментація зображення, YOLOv9e, SAM, DINOv2, виявлення хвороб, семантична сегментація.

ЗМІСТ

ВСТУП

У сучасному аграрному виробництві важливу роль відіграє ефективне управління станом рослин, що впливає на їх здоров’я, продуктивність і врожайність. Одним із ключових напрямків, що сприяють підвищенню ефективності процесів в аграрному виробництві є автоматизація моніторингу стану рослин, що дозволяє значно знизити час і ресурси, забезпечивши вирішення проблеми [17]. Теплиці, як інтенсивно використовувані аграрні об’єкти, потребують постійного контролю умов навколишнього середовища та здоров’я рослин, що вимагають застосування новітніх технологій для своєчасного реагування на поточні загрози.

Однією з найбільш перспективних технологій для автоматизації процесу моніторингу є застосування алгоритмів глибокого навчання, зокрема методів глибинних нейронних мереж, для обробки та сегментації зображення [26]. Завдяки здатності цих алгоритмів до аналізу великих обсягів даних та виявлення складних закономірностей можна ефективно виявити захворювання, стресові фактори та інші проблеми, які можуть виникати у рослин, на різних стадіях їх розвитку.

Сегментація зображення, як частина комп’ютерного зору, дає можливість відображати зображення окремих частин рослин, що дозволяє точно оцінити їх стан і провести аналіз. Використання методів глибокого навчання для цієї задачі є кроком до реалізації інтелектуальних систем моніторингу, які можуть оперативно й точно реагувати на зміни в стані рослин, знижуючи ризики втрати врожаю та підвищуючи ефективність використання ресурсів.

Метою цієї кваліфікаційної роботи є розробка моделей глибокого навчання для сегментації зображення рослин у теплицях, що дозволяють дозволити автоматичний моніторинг їх стану, виявити хвороби та дефекти, а також прогнозувати подальший розвиток рослин. Важливою складовою роботи є порівняння різних підходів до сегментації та вибір найбільш ефективного рішення для практичного використання в аграрних умовах.

Об’єктом дослідження є процес моніторингу стану рослин у теплицях, зокрема застосування технології обробки зображень для автоматичного виявлення проблем у їхньому стані, таких як хвороби, дефекти або стресові фактори, які можуть вплинути на їхній розвиток.

Предметом дослідження є алгоритми глибокого навчання для сегментації зображень рослин, які виключають автоматичну обробку та аналіз даних, отриманих від камер чи інших сенсорних систем у теплицях, для визначення стану рослин та прогнозування їх розвитку.

Таким чином, робота має на меті лише розробку інноваційних алгоритмів для моніторингу стану рослин, а й значне покращення якості управління станом рослин у теплицях, що є важливим кроком у напрямі цифровізації та автоматизації аграрного виробництва.

РОЗДІЛ 1.
АНАЛІЗ СТАНУ ТЕПЛИЦЬ ТА ОБҐРУНТУВАННЯ ДОЦІЛЬНОСТІ ВИКОРИСТАННЯ АЛГОРИТМІВ ГЛИБОКОГО НАВЧАННЯ ДЛЯ СЕГМЕНТАЦІЇ

1.1. Доцільність моніторингу стану рослин у теплицях на основі машинного навчання

Теплиці еволюціонували з основною метою захисту культур від несприятливих зовнішніх умов навколишнього середовища, одночасно створюючи середовище, сприятливе для росту культур, тим самим покращуючи розвиток культур, якість і врожайність [1]. Сучасні теплиці в основному використовують зовнішнє світло, але оснащені конструкціями (рис. 1.1), які дозволяють контролювати внутрішню температуру, включаючи виконавчі механізми, такі як циркуляційні вентилятори, туманоутворювачі та генератори CO_2, а також датчики та сервери даних, контролери для зовнішнього та внутрішнього моніторингу. екологічні умови [2].

Рисунок 1.1 – Сучасні теплиці із інтелектуальними системами управління

Внутрішнє середовище теплиці знаходиться в стані постійної зміни, під впливом зовнішніх умов навколишнього середовища, роботи приводів і фізіологічної діяльності культур, що вирощуються. Культиватори можуть відстежувати мінливе внутрішнє середовище теплиці та ріст культур, щоб приймати обґрунтовані управлінські рішення, керуючи приводами для оптимального налаштування середовища теплиці для росту культур. Ключові фактори навколишнього середовища, які вважаються критичними для регулювання тепличного середовища, включають внутрішню температуру, відносну вологість і концентрацію CO_2.

У 21 столітті завдяки Інтернету речей і великим даним штучний інтелект швидко розвивається, що призвело до досліджень парникового середовища та моделей прогнозування росту врожаю, які є більш універсальними, ніж традиційні методи моделювання. Зокрема, було оцінено, що моделі, засновані на штучному інтелекті, перевершують звичайні статистичні моделі при вирішенні складних нелінійних проблем і застосовуються до складних середовищ теплиць і прогнозів росту врожаю.

Наприклад, модель множинної лінійної регресії (MLR), статистичний метод, який моделює зв’язок між залежною змінною та двома чи більше незалежними змінними, може передбачити внутрішнє середовище теплиць, використовуючи зовнішні та внутрішні фактори середовища як змінні [27].

Модель MLR, здатна навчатися з відносно меншою кількістю даних, може похвалитися обчислювальною ефективністю та простотою інтерпретації, що робить її ефективно застосовною в різних прогнозних середовищах. Машини підтримки векторів (SVM) – це моделі машинного навчання, розроблені для розпізнавання образів, пошуку оптимальної межі рішення для класифікації або прогнозування даних. SVM можна застосовувати до нелінійних прогнозів навколишнього тепличного середовища, оскільки це спрощує нелінійні обчислення шляхом відображення нелінійних вхідних даних у лінійний простір вищої розмірності [7]. Штучні нейронні мережі (ШНМ) складаються з кількох вузлів (або штучних нейронів), з’єднаних між собою в мережеву структуру,

здатних автоматично вивчати характеристики даних і виконувати передбачення або класифікацію [1].

Дослідження із застосуванням ШНМ для прогнозування навколишнього середовища в теплицях використовували 13 різних моделей ШНМ для прогнозування внутрішньої температури повітря, ґрунту та рослин у теплицях [28].

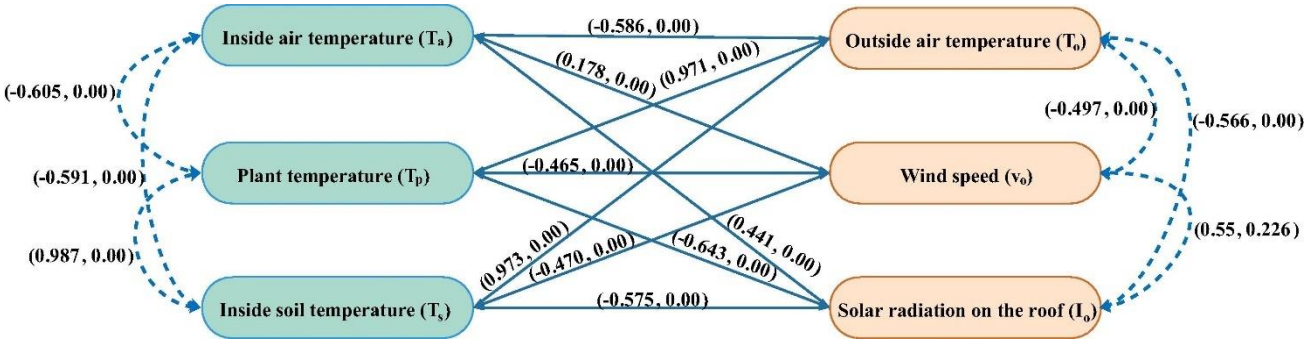

Рисунок 1.2 – Коефіцієнт кореляції між усіма входами та виходами (перше число в дужках показує значення кореляції, а друге число показує значення p-value) [28]

Як інший приклад застосування технології глибокого навчання, Alhnaity, B. et al. (2020) [3] використовували RNN для прогнозування врожайності, швидкості росту та товщини стебла тепличних томатів.

Крім того, Юнг, Д.-Х. та ін. (2020) [11] використовували ANN, NARX і RNN-LSTM для прогнозування температури, вологості та CO_2 умов у теплицях для помідорів, досягаючи значень R^2 до 0,97 для прогнозів у діапазоні від 5 до 30 хв.

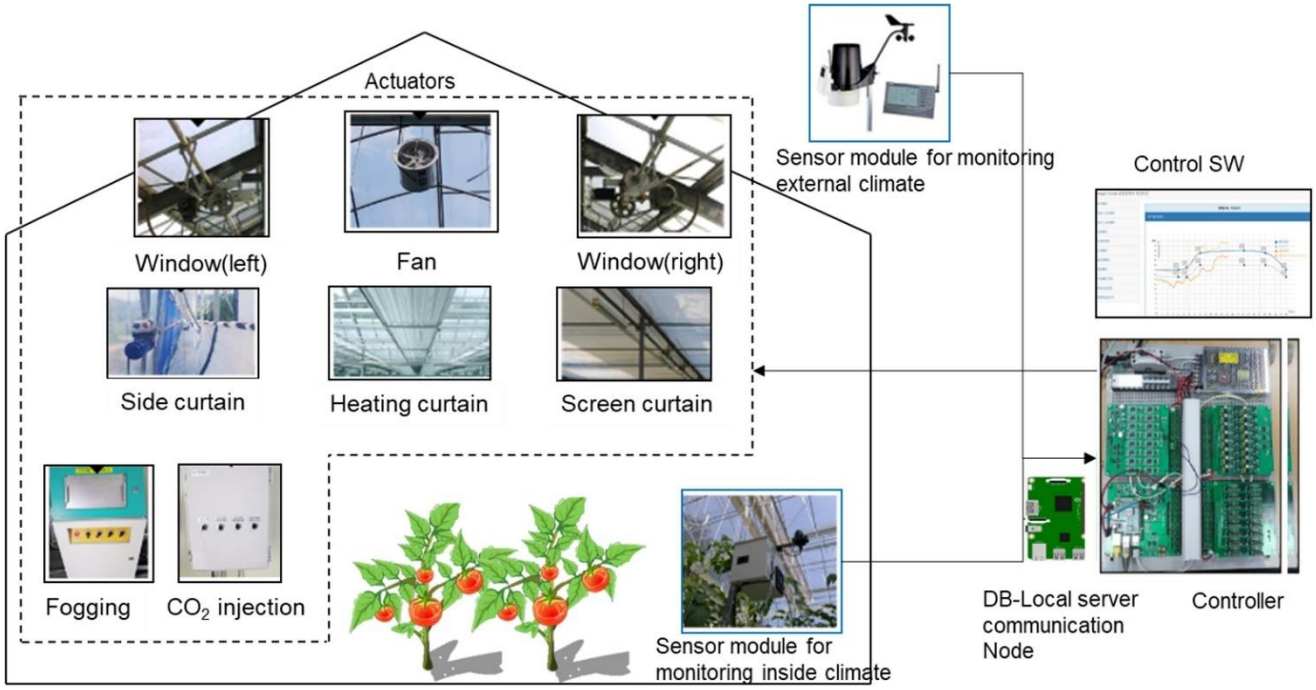

Рисунок 1.3 – Датчики моніторингу та виконавчі механізми контролю середовища , що використовуються для збору даних у теплиці [11]

Це дослідження підкреслило, що менші інтервали прогнозування призвели до вищої точності, але також збільшили споживання обчислювальних ресурсів і пов’язані з цим витрати.

Таким чином, моделі глибокого навчання, які показують хорошу прогностичну продуктивність, можуть вимагати високопродуктивних графічних процесорів або паралельної обробки з кількома графічним процесорами, що збільшує енергоспоживання через необхідність обробки великих обсягів даних і складних математичних операцій. Таким чином, практичне застосування моделей глибокого навчання в більшості реальних теплиць може бути обмеженим.

1.2. Моніторинг стану рослин у теплицях на основі глибокого навчання

Відомі дослідження застосування моделей машинного навчання на основі посилення в різних сферах, оскільки ці моделі можна навчати та аналізувати значно швидше та з меншою обчислювальною потужністю порівняно з моделями глибокого навчання (рис. 1.4).

Рисунок 1.4 – Взаємозв’язки між машинним навчанням та глибоким навчанням

На відміну від «чорної скриньки» моделей глибокого навчання мають структуру засновану на посиленні, легше інтерпретуються, що полегшує розуміння важливості змінних і їх впливу на прогнози.

Нейронні мережі зазвичай застосовуються для прогнозування нелінійних систем, де традиційні методи часових рядів можуть бути не в змозі охопити нелінійні моделі даних. Таким чином, нейронні мережі можна застосовувати для моделювання часових рядів без припущення апріорних функціональних форм моделей. Було запропоновано та успішно застосовано для прогнозування часових рядів багато різноманітних методів нейронних мереж (включаючи ANN, RNN та NARX). Наприклад, у роботі [16] порівняли структурні відмінності в цих алгоритмах навчання на основі часу (рис. 1.5).

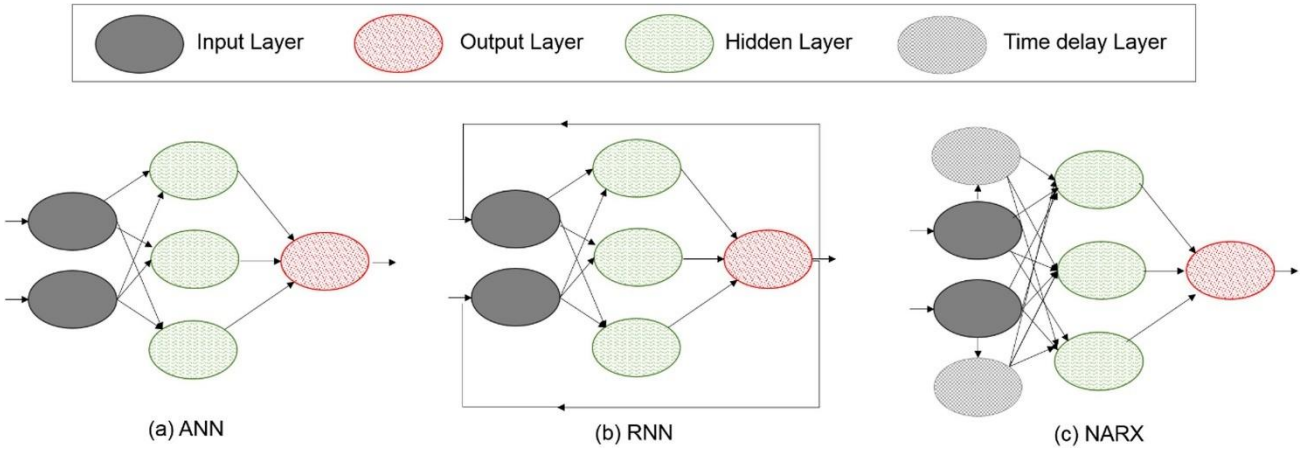

Рисунок 1.5 – Структурні порівняння нейронних мереж на основі часових рядів і традиційних нейронних мереж [16]

RNN може доставити вихідний сигнал на вхідний рівень на наступному часовому кроці, тоді як у NARX сигнал для всіх часових кроків існує на вхідному рівні як шар затримки часу.

Точні та надійні прогнози цих кліматичних умов можуть забезпечити більш ефективне керування приводом, підтримуючи оптимальні кліматичні діапазони та покращуючи якість та врожайність дині. Повідомляється, що ефективні інтервали контролю та спостереження за змінами навколишнього середовища внаслідок роботи приводу в теплицях проводяться щонайменше кожні 30 хвилин [18].

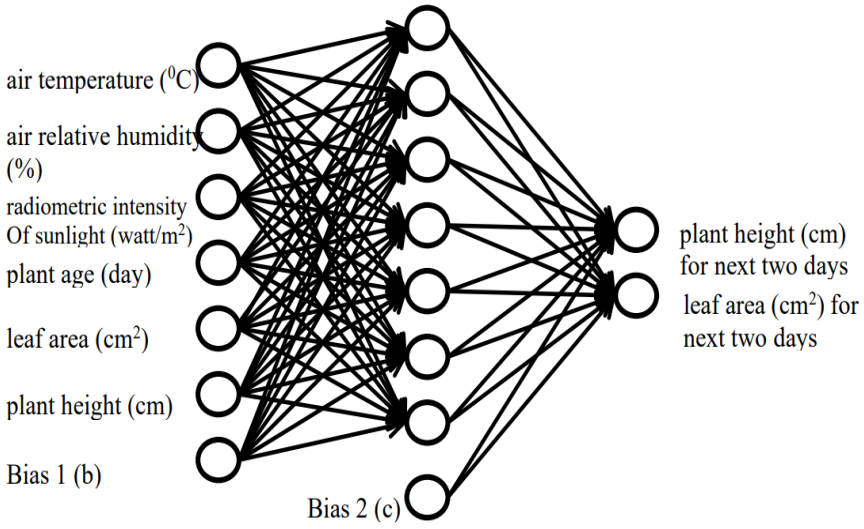

Рисунок 1.6 – Архітектура ШНМ для контролю росту рослин у теплиці [25]

Однак попередні дослідження науковців [26] більше зосереджувалися на прогнозах на більш тривалий період часу або в основному були спрямовані на прогнозування солодкості та врожайності плодів на пізніх стадіях росту культури. Нещодавно Suhardiyanto, H. та Hasbullah, R. (2022) [25] розробили модель ШНМ з точністю R^2=0,9, використовуючи середню температуру, відносну вологість, інтенсивність світла, вік рослин, площу листя та висоту рослин як вхідні параметри для прогнозування висоти рослин дині та площі листя через два дні.

1.3. Аналіз існуючих систем моніторингу стану рослин у теплицях

У контексті розумних теплиць використання нових технології має значні перспективи. Так, у [1] автори запропонували сільськогосподарську систему, яка використовує сервіси на основі вбудованих систем, підключених до мереж датчиків і виконавчих механізмів, розроблених для полегшення моніторингу та контролю. Ці мережі покладаються на перевірені технології (Wi-Fi, BLE або LoRa) і встановлені протоколи (MQTT, HTTP і LoRaWAN), які можуть працювати локально або в хмарі. Різні інтерфейси HMI та M2M створюються відповідно до потреб фермера.

Цю систему використовували для відстеження торгівлі марихуаною в медичних установах. У цій роботі використовується блокчейн Hyperledger Blockchain. Це повністю приватний блокчейн, щоб зберегти конфіденційність різних представлених даних. (рис. 1.7).

У роботі [19] автори представили сільськогосподарську систему, яка базується на ШІ та блокчейні. По-перше, вони працювали над WSN для впровадження розумної системи зрошення . Ця система була використана з іншими датчиками в [6] для відстеження та відстеження оливкової олії.

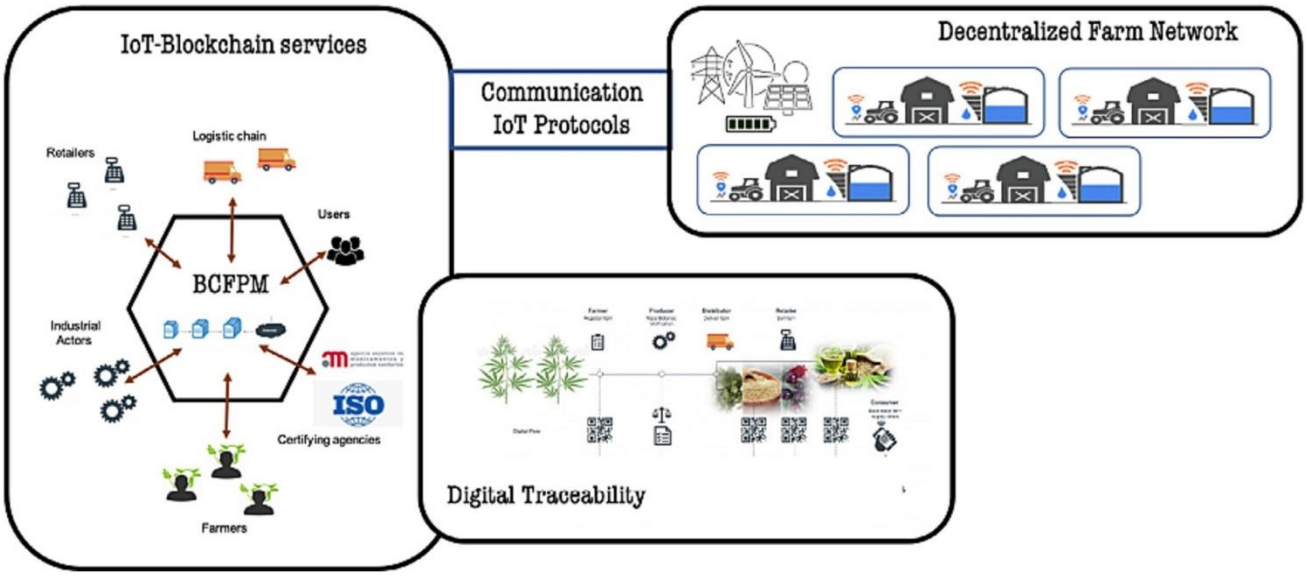

Рисунок 1.7 – Модельний сценарій рішень щодо відстеження стану рослин [1]

Потенційним обмеженням цієї роботи є те, що підхід не стосується відеопотоків для вивчення еволюції та росту рослин.

Ця програма була заснована на блокчейні Ethereum, який є дозволеним блокчейном. Результатом цієї роботи є надання клієнтам доступу до всієї історії виробництва споживаної оливкової олії. Крім того, потенційним обмеженням цієї роботи є те, що підхід не стосується відеопотоків для вивчення еволюції та росту рослин. Але відмінність від роботи, запропонованої в [26], полягає у використанні Ethereum. Це забезпечує кращу видимість платформи, що дозволяє створити публічну систему, доступну для всіх. Таким чином, в рамках цієї платформи продукт можна буде легше експортувати, і це відкриє більше можливостей для країни-виробника оливкової олії .

У дослідженні [32] запропоновано використання технології блокчейну для децентралізованого та безпечного зберігання даних датчиків і камер спостереження . Такий підхід забезпечує незмінність і синхронізацію даних у глобальній базі даних. Блокчейн-мережа об’єднує різні зацікавлені сторони в системі розумного землеробства , включаючи фермерів, постачальників продуктів харчування та клієнтів. Щоб зберегти автентичність зображень, зроблених кількома камерами, у дослідженні використовуються методи шифрування кількох зображень (MIE). Потім ці зашифровані зображення

зберігаються в базі даних блокчейну з унікальними ідентифікаторами. Дослідження представляє імітовану модель технології блокчейн, придатну для впровадження в розумному фермерському середовищі, використовуючи Ganache як тестову мережу. Розумні контракти використовуються для цифрового призначення ролей кожному учаснику мережі блокчейн і полегшення транзакцій. Дослідження показує, що шляхом підключення зашифрованих вихідних даних створення розумної системи ведення сільського господарства можна розглядати як перший крок до підвищення стійкості сільського господарства (рис. 1.8).

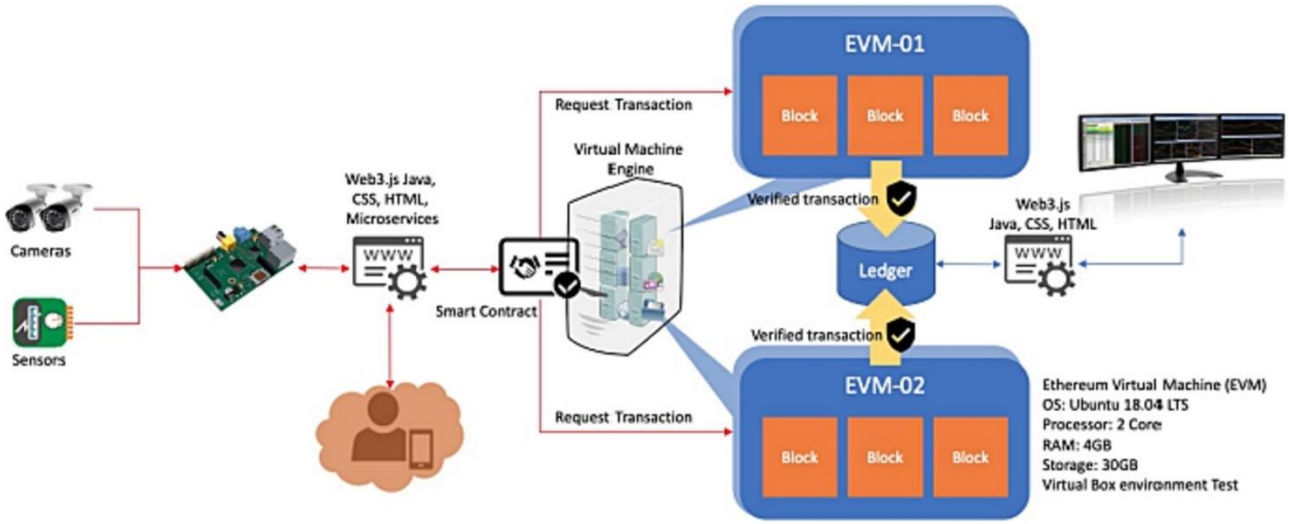

Рисунок 1.8 – Схема реалізації локальної мережі Ethereum [32]

Однак відсутність обробки зображень у цій роботі для подальшого аналізу обмежує дослідження візуальних даних і глибоке розуміння характеристик рослин . Такі методи аналізу зображень, як виявлення контурів, сегментація та виділення ознак, можуть надати цінну інформацію про ріст рослин, здоров’я культур і ідентифікацію хвороб. Крім того, відсутність реального інтерфейсу користувача може обмежити доступність і сумісність системи. Зручний інтерфейс дозволить користувачам, таким як фермери та дослідники, легко взаємодіяти з даними, візуалізувати результати та виконувати певні аналізи. Це сприятиме співпраці між зацікавленими сторонами та посилить обмін інформацією.

Робота [8] представляє систему відстеження фотографій на основі технології блокчейн. Система забезпечує автентичність як оригінальних, так і перетворених зображень, зокрема перевіряючи джерело камери, яка їх зафіксувала. Для цього система використовує інфраструктуру відкритих ключів (PKI), що складається з цифрових камер, і покладається на надійні сертифікати походження, підписані сертифікованими камерами. Запропоноване рішення добре підходить для систем відстеження фотографій в Інтернеті, оскільки в ньому використовується схема ланцюжка сертифікатів, яка може бути застосована до різноманітного програмного забезпечення для перетворення зображень , включаючи програми для редагування фотографій з відкритим кодом. Автори розробили прототип системи відстеження, використовуючи Ethereum як дозволений блокчейн. Тим не менш, використання сертифікованих камер є необхідною умовою для цього завдання. Зображення, зроблені цими камерами, залишаються необробленими, що призводить до того, що значний обсяг даних потрібно зберігати в блокчейні. Отже, транзакційні витрати, пов’язані зі зберіганням такої масивної інформації, є значно високими.

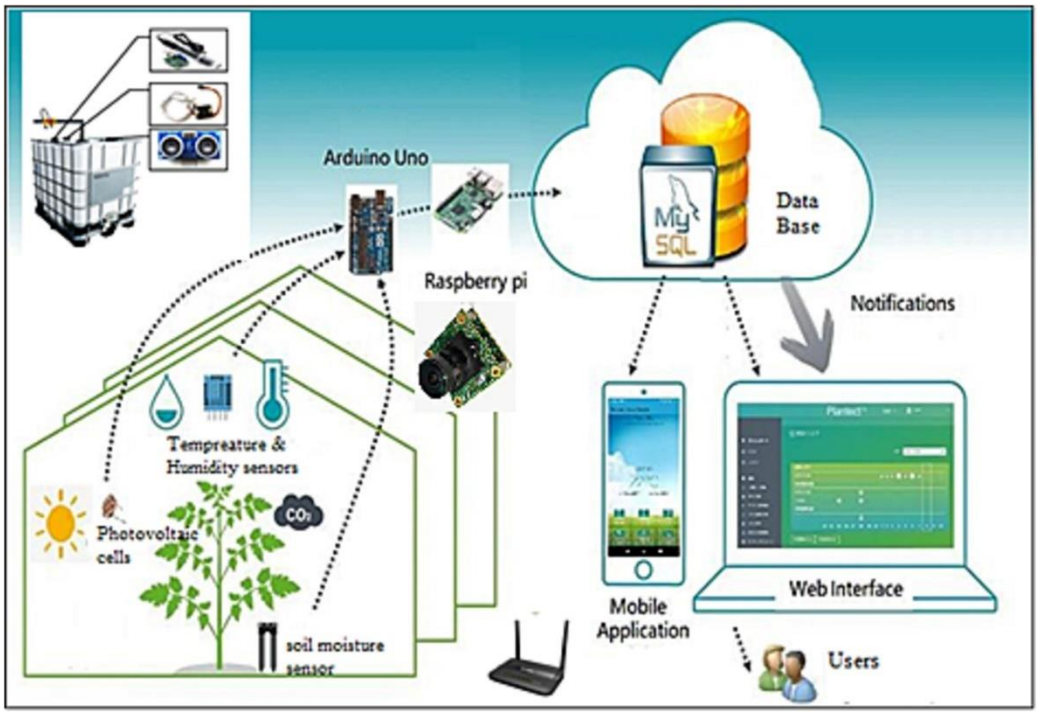

Рисунок 1.9 – Схема впровадженої системи розумної теплиці

Також відомий реалізований прототип теплиці (рис. 1.5). Ця теплиця дозволяє завантажувати дані з різних датчиків, таких як вологість, температура, вологість ґрунту тощо, і здатна відстежувати часові тенденції, раптові зміни та інші коливання цих факторів навколишнього середовища. Вона підключена до бази даних для запису зібраних даних. Користувач має прямий доступ до цих даних за допомогою веб-додатків або програм для мобільного телефону .

1.4. Обґрунтування доцільності досліджень щодо моніторингу стану рослин у теплицях на основі алгоритмів глибокого навчання для сегментації.

Аграрний сектор, особливо тепличне господарство, є важливою складовою економіки багатьох країн, включаючи Україну. Теплиці забезпечують стабільний врожай навіть у несприятливих кліматичних умовах, що їх необхідними для вирощування широкого спектру культури. Проте, на відміну від відкритих підстав, у теплицях рослини знаходяться під постійним впливом мінливих умов навколишнього середовища, таких як температура, вологість, освітленість, а також ризики зараження хворобами або атакою шкідників. У зв’язку з цим виникає необхідність у вільному моніторингу стану рослин для мінімізації втрат врожаю та оптимізації процесів вирощування.

Моніторинг стану рослин традиційно робляють агрономи, які використовують огляди та оцінюють здоров’я рослин на основі фізичних симптомів. Однак цей метод має суттєві обмеження, зокрема, суб’єктивність оцінок, залежність від досвіду фахівця, а також трудомісткість і часозатратність процесу. З огляду на ці недоліки, автоматизовані технології моніторингу рослин стають все більш актуальними. Останні досягнення в галузі обробки зображень і машинного навчання відкривають нові перспективи для вирішення цієї проблеми, зокрема застосування алгоритмів глибокого навчання для сегментації зображення рослин.

Автоматизація моніторингу рослин у теплицях дозволяє забезпечити безперервний, точний та об’єктивний контроль за станом рослин. Технології на основі глибокого навчання, зокрема методи сегментації зображень, дають змогу точний аналіз стану кожної рослини, виявити хвороби, дефекти, навіть незначні зміни у фізіології рослин, які можуть свідчити про проблеми, що потребують втручання. Це дозволяє не тільки зменшити втрати врожаю, але й підвищити загальну ефективність теплового виробництва.

Застосування глибокого навчання для сегментації зображення рослин у теплицях є доцільним з кількох причин. По-перше, сучасні алгоритми глибокого навчання, зокрема згорткові нейронні мережі (CNN), здатні обробляти великі обсяги даних з високою точністю та швидкістю. Вони можуть автоматично виділити зображення здорових і хворих частин рослин, визначити їх фізіологічний стан та передбачити розвиток можливих захворювань або стресових умов. По-друге, зображення з камери, які можуть бути встановлені в теплицях, дають змогу отримувати точну інформацію про шкіру рослини або її частину, що є місцем для прийняття офіційних рішень щодо корекції умов вирощування.

Таким чином, дослідження у напрямі моніторингу стану рослин у теплицях на основі алгоритмів глибокого навчання для сегментації зображень не тільки є доцільними, а й необхідними для підвищення ефективності аграрного виробництва. Вони можуть створити нові, більш точні та автоматизовані методи моніторингу, які значно підвищать якість управління тепличними господарствами, зменшать витрати на робочу силу та збільшать врожайність. Враховуючи швидкий розвиток технологій глибокого навчання та машинного зору, можна очікувати подальше вдосконалення таких методів та їх широке впровадження в аграрний сектор.

РОЗДІЛ 2.
ХАРАКТЕРИСТИКА ОБ'ЄКТУ ДОСЛІДЖЕННЯ ТА ВИБІР ЗАСОБІВ

2.1. Формулювання задачі дослідження

Рослини є основою життя різних біологічних об'єктів на Землі. Вони реалізують життєво важливі процеси абіогенезу, такі як виробництво кисню та забезпечення їжею для широкого спектру організмів. Контроль стану та здоров'я рослин покращує сільськогосподарську діяльність. У нашій роботі запропоновано розгортання найсучасніших моделей комп'ютерного зору в конвеєрі, який є основою для системи моніторингу здоров'я рослин, модифікованої та поєднаної з доданими попередньо навченими моделями машинного навчання для розширення функціональності та створення системи моніторингу здоров'я рослин для внутрішніх гідропонних умов тепличного рослинництва.

Робота зосереджена на виконанні цього завдання з обмеженими ресурсами. Зокрема, використано лише одну камеру RGB, яка генерує сповільнені відеодані як вхідні дані та відстежує зміни параметрів, що вказують на здоров'я рослин, зберігаючи безперервність у часі. Інше основне обмеження полягає в тому, що для подальшого навчання деяких моделей використовуються агностичні дані щодо рослин, тобто середовище у сценаріях використання теплиці, яке відрізняється від доданих навчальних даних.

Параметри, які потрібно перевірити, були вибрані з урахуванням вказівок із відповідної літератури та досвіду в галузі. Ці параметри включають зміни площі листя, зміни висоти рослини, підрахунок кількості листя, ознаки стресу, такі як опіки по краях, коричневі плями та сильні зміни кольору. Будь-який параметр, який є візуальним сигналом, що характеризує стан рослини, є важливим фактором для системи моніторингу здоров'я рослин.

У роботі вирішувалися наступні завдання:

✓ використання набору моделей машинного навчання як одного конвеєра для обробки візуальних даних уповільненої зйомки для розробки системи моніторингу здоров'я рослин;

✓ ефективна ідентифікація ознак здоров'я рослин за допомогою сповільнених відеоданих за допомогою сукупності моделей машинного навчання, навчених на наборах даних, не пов'язаних із рослинами, без спеціального обладнання для камер чи ручних вимірювань.

Мета цієї роботи стосується вивчення того, як можна створити систему моніторингу здоров'я рослин, використовуючи запропоновану реалізацію, яка може стати основою для такої системи. Очікуваним результатом є інтелектуальна система, яка виконує низку завдань у теплицях (рис. 2.1).

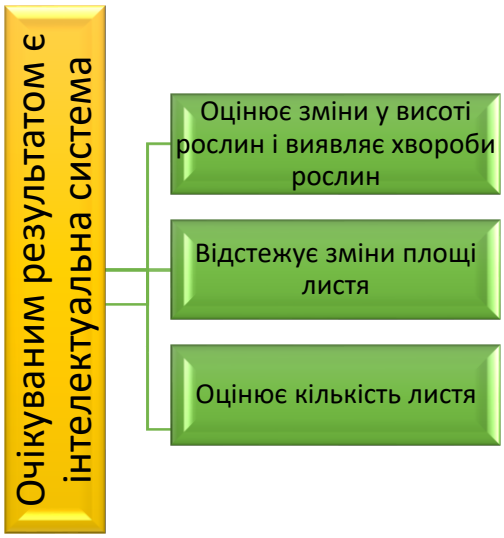

Рисунок 2.1 – Завдання які розв'язує створена інтелектуальна система

Створена на основі досліджень інтелектуальна система оцінює зміни у висоті рослин і виявляє хвороби рослин, відстежує зміни площі листя та оцінює кількість листя, присутнього в спостережуваному середовищі, роблячи це за допомогою вищезгаданого самоконтролю.

При цьому існують наступні обмеження, які приносять великі переваги, але потенційно можуть перешкоджати точності впровадження. Наприклад, використання лише однієї RGB-камери та даних навчання, не пов'язаних із умовами конкретної теплиці. Очікуваний результат має бути досягнутий за

допомогою поєднання моделей комп’ютерного зору на основі глибокого навчання.

2.2. Особливості виконання сегментації рослин

Незважаючи на постійний прогрес, найсучасніші моделі відеоспостереження приймають лише кадри з нижчою роздільною здатністю, такі як трансформатори для 224x224 або 384x384 пікселів [30]. У цих низьких роздільних здатностях, для яких відеовхід зменшується, може бути важче відстежувати невеликі, дрібні зміни в деталях об’єкта, що контролюється.

Наявність ресурсоефективної системи моніторингу теплиці, яка функціонує без спостереження, робить таку систему більш надійною для повторного створення в місцях, де теоретично неможливо виконувати спостереження за рослинами із проведенням ручних вимірювань або за допомогою доданого фізичного обладнання або обладнання, відмінне від однієї відеокамери RGB. Крім того, оскільки кадри уповільненої зйомки висвітлюють еволюцію рослини у стислій картинці відео, майбутні моделі, які мають завдання прогнозувати, можуть робити це більш ефективно. При цьому можна спостерігати за різними стадіями росту та унікальними характеристиками рослин у форматі сповільненої зйомки.

Метою початкового процесу сегментації є видалення об’єктів, які не мають відношення до системи моніторингу теплиці, наприклад фону. Бажано мати можливість виконувати сегментацію рослин, які не мають фону. Припущення полягає в тому, що далі в конвеєр моделі, де використовується панорамна сегментація, тобто поєднання семантичної та екземплярної сегментації, яка потребує потрібних обчислень [13], ця частина конвеєра матиме справу лише з відповідним об’єктом і, таким чином, підвищить точність виділення ознак і прогнозування стану здоров’я рослин.

Процес сегментації попередньої обробки даних використовує алгоритм кластеризації K-Means, широко поширену техніку неконтрольованого машинного навчання, яка використовується для багатьох різних цілей обробки зображень, а вибір кластеризації K-Means для початкової сегментації теплиці здійснений зп простотою в розрізненні рослин та їх фонові у пікселях. Цей метод не вимагає позначених навчальних даних, що пов’язано з цією роботою щодо застосування мінімального спостереження та уникнення непрактичності ручного анотування [9].

Прийнявши кластеризацію K-Means, виконано поділ пікселів кожного кадру на кластери на основі їхніх значень кольору та інтенсивності, припускаючи, що пікселі, які належать рослини та фону, утворюватимуть окремі кластери за різними колірними характеристиками.

Щоб фонова сегментація була успішною, до піксельних даних кожного кадру застосовується алгоритм K-Means. Кількість кластерів встановлюється на попередньо визначене значення (наприклад, 2), що представляє рослину та фон.

Рисунок 2.2 – Сегментація фону для ізольованої рослини з використанням кластеризації K-середніх

Алгоритм перебирає піксельні дані, призначаючи кожному пікселю найближчий центроїд кластера на основі евклідової відстані в колірному просторі. Після завершення кластеризації створюється зображення сегментації шляхом присвоєння всім пікселям, що належать кластеру рослин, значення 1 (білий), а всім іншим пікселям значення 0 (чорний). Це зображення ефективно

відокремлює рослину від фону. Сегментаціґ застосовується до оригінального кадру, ізолюючи рослинні пікселі від фону. Потім цей сегментований кадр використовується для подальшого аналізу.

Відомо, що алгоритм K-means (рисунок 2.2) успішно сегментує рослини у теплиці, хоча є певний фоновий витік з об’єктів, не пов’язаних з рослинами теплиці. Однак у більш складних сценаріях швидко стає очевидним, що групування пікселів на основі подібності або мінімального чи максимального простору, не вдається чітко виділити потрібні об’єкти (рослин), як це видно на рисунку 2.3. Зображення на рисунку 2.3 є сповільненою зйомкою. Відео з вертикальної теплиці, де кілька вирощуваних рослин складаються у одне зображення. Для обробки складнішого динамічного зображення потрібні кращі методи сегментації фону.

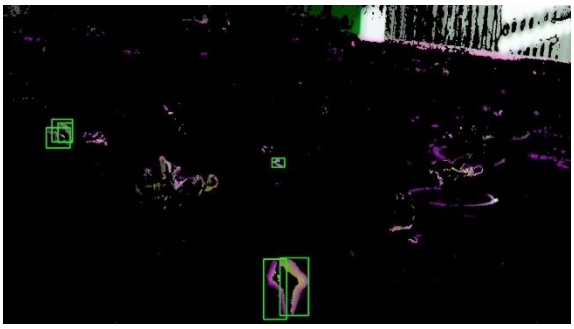

Рисунок 2.3 – Фонова сегментація для множини рослин у теплиці за допомогою кластеризації K-середніх

Сегментуючи фон шляхом групування подібних пікселів, також можна відстежувати зміни, наприклад зміни висоти для одного такого кластера. На рисунку 2.4 наведено графік змін висоти для зображення представленого на рисунку 2.2.

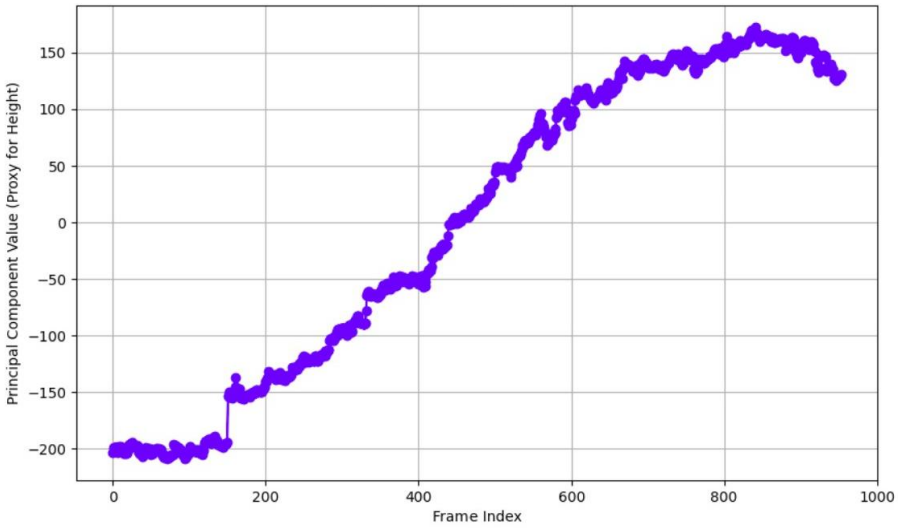

Рисунок 2.4 – Початкові результати відстеження зміни висоти за сповільненим вхідним відео сигналом

Нахил вниз у кінці є артефактом непослідовного відстеження, викликаного неоднозначністю в зображенні поводження з об’єктом і фоном, або іншими формами невідповідностей, які модель інтерпретує як продовження моделі росту рослини.

2.3. Особливості виявлення стану здоров’я рослин

Обмежувальні прямокутники використовуються для виявлення стану здоров’я рослин. Проблема з цим підходом полягає в тому, що навколишня область, де може бути дефект, не є дуже специфічною, як можна побачити нижче на рисунку 2.5. Точніше знати, де відбувається погіршення стану здоров’я рослин, є більш корисним. Отже, існує потреба звести до мінімуму будь-яку частину зображення, щоб більш конкретно націлитися на здорову чи нездорову область рослини. Для цього конкретного виявлення використовувався Roboflow Plant Leaf Dataset Version 2. Конкретні класи в цьому наборі даних, які представляють дефекти здоров’я рослин, такі як опіки наконечників і цвіль [10], були об’єднані в нову папку з класом під назвою Just Unhealthy.

На рисунку 2.5 нижче на рослинах немає видимих артефактів, пов’язаних із погіршенням здоров’я, тому модель не повинна класифікувати будь-який об’єкт у будь-якому відео кадрі як нездоровий.

Рисунок 2.5 – Початкові результати виявлення здорових і нездорових рослин

Як видно на рисунку 2.5, область рослин успішно позначена як здорова. Однак, проблема великої аналізу великої області теплиці все ще залишається, а результати просто надто розпливчасті. Оскільки використовується обмежувальна рамка, то отримуємо дуже велику область зображення. Не показано, яка рослина або навіть яка область рослини виявляється як здорова.

2.3. Вибір моделей для сегментації рослин

Декодер моделі Segment Anything (SAM) було налаштовано за допомогою набору даних DeepPlants AGM-HS [22], щоб дозволити йому автоматично сегментувати листя рослин. Це він може зробити лише за допомогою підказки, використовуючи так звану функцію наведення курсора та клацання. При цьому створюється лише одна межа для об’єкта, яку користувач вибирає, де клацання користувача діє як підказка. Крім того, використовуючи демонстрацію, надану Meta [23] можна встановити, що специфічна для листя сегментація не є чимось, що робить унікальною модель, як показано на рисунку 2.6.

Рисунок 2.6 – Сегментація екземпляра за допомогою SAM

Використовуючи DeepPlants AGM-HS, можна оцінити потенціал точно налаштованого SAM для виявлення та сегментування листка, де є дефект, як показано на рисунку 2.7.

Рисунок 2.7 – Сегментація дефектів рослин – невдала спроба

Як можна побачити, сегментація із зеленою маскою застосована до неправильного листка, і модель не розпізнає коричневу область ліворуч. Модель також непослідовна у своїй сегментації. Для того самого зображення, але в іншому циклі, вона виконує сегментацію для більшого листка на рисунку 2.8. Дефектною ділянкою є сіра частина на одному з листків. Сегментація виконується на неправильному листку.

Рисунок 2.8 – Сегментація дефектів рослин – невдала спроба

Заслуговує на увагу семантична сегментація Dinov2. На рисунку 2.9 нижче наведено результат базової моделі DINOv2 для семантичної сегментації, згенерованого з демонстраційного листка [15].

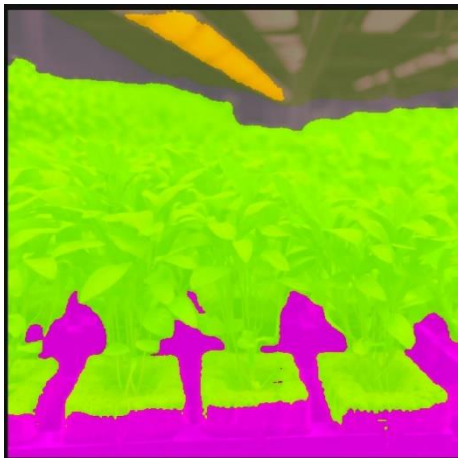

Рисунок 2.9 – Семантична сегментація за допомогою DINOv2

Як видно, початкових можливостей семантичної сегментації DINOv2 було б недостатньо для оцінки конкретної площі листа або зміни висоти чи розміру. При цьому модель у складних зображеннях має перекриття об’єктів і не може виконати належне розмежування. Модель з її можливостю сегментації кожної окремої рослини, не виконує специфічну сегментацію листя в його існуючому стані.

Таблиця 2.1 – Вибрані моделі та їх характеристики

Назва моделі	Технічні деталі
YOLOv9e	Виявлення об’єктів
YOLOv9e-seg	Виявляє об’єкти та окреслює їх межі
YOLOv9e-seg з SAHI	Використовує структуру SAHI для нарізки та обробки великих зображень; покращує продуктивність сегментації на детальних і складних зображеннях.
FastSAM-X із ByteTrack	Швидша версія моделі Segment Anything і найбільша версія з функцією відстеження об’єктів.

Наша робота зосереджена на розробці системи моніторингу рослин, яка може бути розгорнута в більшості тепличних господарств. Пропонована система не покладається на будь-яке спеціальне обладнання, окрім звичайної відеокамери для збору даних. Для навчання використовуються набори діагностичних даних із різними середовищами теплиці. Моделі, які використовуються описані в таблиці 2.1.

РОЗДІЛ 3.
РЕЗУЛЬТАТИ НАЛАШТУВАННЯ МОДЕЛЕЙ ТА ЇХ ВИКОРИСТАННЯ ДЛЯ МОНІТОРИНГУ СТАНУ РОСЛИН У ТЕПЛИЦЯХ НА ОСНОВІ АЛГОРИТМІВ ГЛИБОКОГО НАВЧАННЯ

3.1. Налаштування моделей моніторингу стану рослин у теплицях

Процес тонкого налаштування починається з пошуку відповідного набору даних, який виконується за допомогою платформи Roboflow [5]. Перевага Roboflow для завантаження набору даних полягає в тому, що він надає функціональні можливості для автоматичного структурування набору даних відповідно до вимог використовуваної моделі, хоча підтримується лише обмежений набір моделей. Після завантаження набору даних для YOLOv9, у подальшому він структурований у наступному потрібному форматі YOLO:

– зображення – зберігаються в каталозі та розділені на підкаталоги train, val і test;

– анотації – відповідні кожному зображенню, збережені у файлах .txt із такою ж назвою, що й зображення. Кожен рядок у файлі анотації представляє одну обмежувальну рамку у форматі: <ідентифікатор класу> <x центр> <y центр> <ширина> <висота>. Координати також нормалізуються, приймаючи значення від нуля до одиниці.

3.1.1. Налаштування та навчання моделі YOLOv9

Для YOLOv9 також потрібен файл конфігурації у форматі YAML. Метою файлу YAML є визначення шляхів до папок тестування, навчання та перевірки в каталозі, а також імен для використовуваних класів. Також створюється додатковий файл YAML, який ми називаємо hyperparameters.yaml, де, наприклад, швидкість навчання вказується наступним чином:

```
l r : 0.0016
```

За замовчуванням платформа Ultralytics вибирає швидкість навчання, яка не завжди може бути найоптимальнішою для навчання, яке ми проводимо на наших конкретних даних.

Для реального навчання наших моделей YOLO використовується платформа Ultralytics [31]. Це робиться шляхом створення навчального сценарію, де спочатку імпортуємо YOLO з бібліотеки Ultralytics. Для навчання моделі на спеціальних даних Ultralytics вимагає, щоб спочатку створили виконуваний файл Windows. Це тому, що Ultralytics потрібен доступ до багатопроцесорної функції в Python. Через це імпортуємо torch.multiprocessing як mp і створюємо рядок, пишемо код (рис. 3.1).

```
import torch.multiprocessing as mp
mp.freeze_support()
```

Рисунок 3.1 – Фрагмент коду імпорту бібліотеки torch.multiprocessing

Модель YOLOv9 завантажується, спочатку використовуючи попередньо навчену версію, як показано в наступному рівнянні:

$$model\_x = YOLO(“yolov9.pt"), \quad (3.1)$$

Спеціальна конфігурація YAML і початок навчання:

За потреби можемо створити власний файл YAML для додаткового налаштування гіперпараметрів. Це продемонстровано за допомогою наступного фрагмента коду Python (рис. 3.2).

```
hyperparams_file = 'custom_hyperparameters.yaml'

with open(hyperparams_file, 'w') as f:
    f.write('lr0: 0.0016\n')
```

Рисунок 3.2 – Фрагмент коду створення власного файлу YAML для додаткового налаштування гіперпараметрів

Початок навчання ініціюється за допомогою змінної, як показано нижче на рис. 3.3.

```
results = model_x.train(
    data='path_that_points_to_head_data.yaml',
    epochs=85,
    imgsz=640,
    optimizer='AdamW',
    lr0=0.0016,
    momentum=0.937
)
```

Рисунок 3.3 – Фрагмент коду початку навчання, що ініціюється за допомогою змінної

Тут описано загальний процес використання Roboflow із Ultralytics і спеціальним сценарієм тонкого налаштування. Цей процес можна повторити для того самого підходу до тонкого налаштування для моделей YOLO.

3.1.2. Налаштування SAM

Для точного налаштування моделі SAM необхідні такі бібліотеки та інструменти:

- ✓ Numpy – для числових операцій;
- ✓ Pytorch – включає моделі комп'ютерного зору і torchvision для моделей і трансформацій;
- ✓ Обробка даних – Dataset і DataLoader від Pytorch для ефективного завантаження даних;
- ✓ Обробка зображень – зображення з бібліотеки PIL використовується для завдань обробки зображень;
- ✓ Процесор SAM – спеціальна бібліотека для обробки спеціальної попередньої обробки, необхідної для SAM;
- ✓ Набори даних Hugging Face – для доступу та використання наборів даних, доступних через Hugging Face;
- ✓ Monai – спеціально імпортований для розширених можливостей функції втрати.

Функція get_bounding_box призначена для обчислення обмежувальної рамки з карти сегментації. Клас SAMDataset – це клас є невід'ємною частиною подачі даних у попередньо підготовлену модель SAM для тонкого налаштування. Структура архітектури використовується в циклі навчання, який включає відповідну функцію втрат і оптимізатор для вдосконалення продуктивності моделі на нових даних.

3.1.3. Налаштування DINOv2 для семантичної сегментації

Нижче наведено основні частини процесу реалізації процесу тонкого налаштування моделі DINOv2 (додаток А).

Наступні бібліотеки мають вирішальне значення для успішного впровадження та роботи процесу тонкого налаштування:

- ✓ random – використовується для генерації випадкових чисел для розділення даних.
- ✓ numpy – необхідний для виконання числових операцій.
- ✓ torch – керує тензорами та пристроями.
- ✓ Pillow – бібліотека зображень Python, яка використовується для обробки та обробки зображень.
- ✓ Transformers – від Huggingface, використовується для отримання попередньо навченої моделі DINOv2.
- ✓ Evaluate – використовується для обчислення метрик оцінювання, таких як середнє перетину через об'єднання (meanIoU).

Клас DinoSegmentationModel – це клас побудовано навколо попередньо підготовленої моделі DINOv2. Він включає в себе кілька компонентів:

1. Ініціалізація – попередньо підготовлена модель передається як вхідний аргумент. Файл конфігурації завантажується в конструктор для налаштування параметрів моделі.
2. Основа сегментації – додано згортковий шар:

```
nn.Conv2d(config.hidden_size, config.num_labels, kernel_size=1)
```

Цей рівень перетворює вихід попередньо підготовленого DINOv2, перетворюючи приховані розміри на кількість класів, необхідних для семантичної сегментації.

3. Метод Forward – приймає два вхідні дані: self і pixel_values. Він виконує передачу вперед, використовуючи:

```
outputs = self . pretrained_model ( pixel_values )
```

Потім карти функцій обробляються основою сегментації, створюючи необхідні прогнози для завдання сегментації.

4. Вихід – метод повертає необроблені прогнози як логіти для кожного класу та кожного пікселя в піксельній сітці.

Навчальний цикл структурований для повторення набору даних для певної кількості епох.

Для завдання оцінки листової площі є використовується кілька бібліотек:

✓ OpenCV (cv2) – використовується для різних завдань обробки зображень;

✓ PyTorch (факел) – комплексна структура глибокого навчання, яка використовується для керування такими моделями, як YOLOv9, і моделями оцінки глибокого навчання;

✓ PIL (бібліотека зображень Python) – використовується для обробки зображень і завдань;

✓ ultralytics YOLO – структура виявлення об’єктів, яка ідентифікує об’єкти, зокрема листя на зображеннях;

✓ Transformers – використовується для оцінки глибини із зображень за допомогою таких моделей, як AutoModelForDepthEstimation і AutoImageProcessor.

Функція evaluation_area_change – ця функція оцінює площу листків за допомогою таких параметрів, як карта глибини, маска сегментації та координати обмежувальної рамки. Вона виконує вилучення значення глибини – функція змінює розмір маски сегментації відповідно до розмірів карти глибини. Потім

вона застосовує карту глибини як булеву маску для виділення значень глибини, що відповідають площі листа. У подальшому проводиться розрахунок середньої глибини – обчислюється середня глибина аркуша із середнього ізольованих значень глибини. Ця середня глибина служить мірою відстані до камери. Розрахунок площі листа підсумовує кількість пікселів у масці сегментації, де значення більше нуля, що представляє область листа. Кожен вхідний відеокадр фіксується та обробляється для виявлення листя. Оцінка глибини виконується одночасно, а координати обмежувальної рамки, що охоплює маску сегментації, подаються в YOLO за допомогою ByteTrack як вхідних даних.

3.1.4. Виявлення хвороб рослин

Для виявлення хвороби рослин використовуємо кілька ключових бібліотек:

- Torch – керує тензорами та пристроями, є центральний для операцій із моделями глибокого навчання;
- Cv2 – надає доступ до функцій OpenCV для обробки зображень;
- Numpy – необхідний для числових операцій і маніпуляцій в екосистемі Python;
- Ultralytics – полегшує завантаження та використання моделі YOLO для виявлення об’єктів;
- Logging – використовується для відстеження процесів і налагодження шляхом реєстрації даних виконання.

Ініціалізація моделі YOLO виконується завдяки його унікальному файлу «.pt» за допомогою бібліотеки ultralytics, а поріг достовірності встановлюється на 0,85 за допомогою вбудованого модуля «.conf».

Функція apply_masks_to_frame приймає два аргументи – рамку та зображення сегментації. Її основна мета застосувати ці зображення до вхідного кадру, ізолюючи об’єкти на передньому плані.

Функція process_video призначена для обробки відеоданих. Вона приймає відео і модель сегментації як вхідні аргументи:

1. Обробка кадрів – ініціалізується список для зберігання оброблених кадрів. Потім цикл обробляє кожен кадр, використовуючи точно налаштовану модель YOLOv9 для відстеження та сегментації кадру на основі вказаного порогу;

2. Застосування маски – для кожного кадру застосовується маска сегментації за допомогою функції apply_masks_to_frame;

3. Зберігання – усі оброблені кадри зберігаються в списку для візуалізації.

Основна функція Головна функція керувати загальним процесом. Вона викликає функцію ‘process_video’, вказуючи каталог шляху до вхідних даних. Ця функція діє як точка входу для обробки відеоданих для виявлення дефектів стану здоров’я рослин.

3.1.5. Підрахунок листів

Наведені нижче особливості, які висвітлюють важливі аспекти реалізації коду для підрахунку листів, при цьому ключові області підкреслюються шляхом демонстрації відповідних фрагментів коду.

Насамперед імпортуються усі потрібні бібліотеки (рис. 3.4).

```
import cv2
import numpy as np
import torch
import matplotlib.pyplot as plt
from PIL import Image
from ultralytics import YOLO, FastSAM
from ultralytics.models.fastsam import FastSAMPrompt
import supervision as sv
from ultralytics.trackers.byte_tracker import STrack
```

Рисунок 3.4 – Фрагмент коду для імпорту усіх потрібних бібліотек

У подальшому виконується завантаження моделі FastSAM і YOLO (рис. 3.5).

```python
from ultralytics import YOLO
from ultralytics.models.fastsam import FastSAM

# Завантаження моделей
fastsam_model = FastSAM('FastSAM-x.pt')  # Завантаження моделі FastSAM
yolo_model = YOLO('bestleafseg.pt')      # Завантаження моделі YOLO
```

Рисунок 3.5 – Фрагмент коду для завантаження моделі FastSAM і YOLO

Здійснюється вибір області інтересу (ROI)

```python
# Початкові значення для області інтересу (ROI) та прапора малювання
roi = [0, 0, 0, 0]  # Координати області інтересу (Region of Interest)
drawing = False      # Прапор для визначення, чи відбувається зараз малювання
```

Рисунок 3.6 – Фрагмент коду для вибору області інтересу (ROI)

roi – це список із чотирьох значень, що представляють область інтересу (Region of Interest) – [x_start, y_start, x_end, y_end]. Використовується для збереження координат прямокутної області, яку потрібно виділити, наприклад, у зображенні. Drawing – логічна змінна (False або True), яка використовується для визначення стану рисунка.

```python
def extract_fastsam_mask(frame, model, roi):
    """
    Виділяє сегментаційну маску із заданої області інтересу (ROI) у кадрі за допомогою FastSAM.

    Parameters:
    - frame: numpy.ndarray
        Вхідний кадр (зображення).
    - model: FastSAM
        Модель FastSAM для сегментації.
    - roi: list
        Область інтересу у форматі [x_start, y_start, x_end, y_end].

    Returns:
    - mask_full: numpy.ndarray
        Повнорозмірна сегментаційна маска.
    """
    # Вирізати область ROI із кадру
    roi_frame = frame[roi[1]:roi[3], roi[0]:roi[2]]

    # Застосувати модель для отримання результатів
    results = model(
        roi_frame,
        device="cuda",
        retina_masks=True,
        imgsz=1024,
        conf=0.3,
        iou=0.85
    )

    # Обробка результатів FastSAMPrompt
    prompt_process = FastSAMPrompt(roi_frame, results, device="cuda")
    annotations = prompt_process.everything_prompt()  # Отримати сегментаційні маски

    # Створити і повернути повнорозмірну маску
    if annotations and annotations[0].masks:
        mask = annotations[0].masks.data[0].cpu().numpy()
        mask_full = np.zeros(frame.shape[:2], dtype=np.uint8)
        mask_full[roi[1]:roi[3], roi[0]:roi[2]] = mask
        return mask_full
    else:
        # Якщо масок немає, повертається порожня маска
        return np.zeros(frame.shape[:2], dtype=np.uint8)
```

Рисунок 3.7 – Фрагмент коду для вилучення масок сегментації

Змінна ROI використовується для зберігання списку значень координат, які вказуватимуть, де в першому кадрі відео буде намальовано прямокутник, усередині якого потрібно виконати виявлення/сегментацію об’єкта, а змінна рисунку як глобальна змінна вказує, чи ROI малюється або не малюється. Наведена нижче функція використовується для реєстрації дії ШІ в прямокутному полі, яке буде намальовано на екрані.

```python
import cv2
import supervision as sv

def process_video(video_path, fastsam_model, yolo_model, roi):
    """
    Обробляє відео, застосовуючи сегментацію FastSAM, детекцію YOLO,
    і мультиоб'єктне трекінг ByteTrack.

    Parameters:
        video_path (str): Шлях до відеофайлу.
        fastsam_model: Модель FastSAM для сегментації.
        yolo_model: Модель YOLO для детекції об'єктів.
        roi (list): Область інтересу [x_start, y_start, x_end, y_end].

    Returns:
        tuple: Списки frame_numbers і cumulative_counts, що зберігають
        дані для кожного кадру.
    """
    # Ініціалізація відеозахоплення і трекера
    cap = cv2.VideoCapture(video_path)
    mask_list = []
    tracker = sv.ByteTrack()
    cumulative_count = 0
    unique_ids = set()
    frame_numbers = []
    cumulative_counts = []
    frame_id = 0

    while True:
        ret, frame = cap.read()
        if not ret:
            break  # Завершення обробки, якщо кадри закінчилися

        # Отримання сегментаційної маски за допомогою FastSAM
        mask = extract_fastsam_mask(frame, fastsam_model, roi)
        mask_list.append(mask)

        # Вирізання кадру в межах ROI
        roi_frame = frame[roi[1]:roi[3], roi[0]:roi[2]]

        # Виконання детекції об'єктів у ROI за допомогою YOLO
        results = yolo_model(roi_frame)

        # Конвертація результатів у формат Detections (Supervision)
        detections = sv.Detections.from_ultralytics(results[0])

        # Виконання мультиоб'єктного трекінгу за допомогою ByteTrack
        sv.ByteTrack.multi_predict(tracker.tracked_tracks)
        tracker.update_with_detections(detections)

        # Оновлення кумулятивного підрахунку унікальних об'єктів
        for track in tracker.tracked_tracks:
            if track.track_id not in unique_ids:
                unique_ids.add(track.track_id)
                cumulative_count += 1

        cumulative_counts.append(cumulative_count)
        frame_numbers.append(frame_id)

        # Інкремент номера кадру
        frame_id += 1

    cap.release()
    return frame_numbers, cumulative_counts
```

Рисунок 3.7 – Фрагмент коду для обробки відео для виявлення об’єктів і виділення масок

У подальшому виконується вилучення масок сегментації (рис. 3.7). Після цього проводиться обробка відео для виявлення об’єктів і виділення масок (рис. 3.8).

Перегляд кадрів відео передбачає такі кроки:

- ✓ Важливо отримати маски сегментації за допомогою FastSAM;
- ✓ Виявляти об’єкти за допомогою YOLO;
- ✓ Перетворити результати виявлення у формат виявлення нагляду;
- ✓ Виконуйте відстеження кількох об’єктів за допомогою ByteTrack;
- ✓ Оновіть трекер виявлення;
- ✓ Оновіть сукупний підрахунок;
- ✓ Збільшити ідентифікатор кадру також можна, але це більш необхідно, якщо важливо відстежувати унікальні ідентифікатори.

Після оновлення сукупного підрахунку можна візуалізувати графік, щоб також продемонструвати зміни між відеокадра, або можна роздрукувати максимальне число, щоб швидко показати найвищий підрахунок.

3.1.6. Обчислення висоти рослин

Реалізація оцінки висоти подібна до того, як використовували область інтересу в попередньому підрозділі для підрахунку кількості листя, але замість використання YOLO для сегментації листя FastSAM знову використовується для сегментації всієї рослини та ядра цієї реалізації для використання карти глибини, які допомагають оцінити зміну висоти сегментованої області між кадрами за допомогою окремої функції. Щоб використовувати модель глибини будь-що для створення карт глибини, Huggingface можна використовувати для завантаження моделі в наш локальний скрипт.

Функція обчислення висоти рослин з інтеграцією карти глибини представлена на рис. 3.8.

```python
import numpy as np

def calculate_plant_height(mask_array, depth_array):
    """
    Обчислює висоту рослин на основі маски сегментації та глибинної карти.

    Parameters:
        mask_array (list of np.ndarray): Список масок сегментації.
        depth_array (list of np.ndarray): Список відповідних глибинних карт.

    Returns:
        list: Список згладжених висот рослин.
    """
    heights = []

    for mask, depth in zip(mask_array, depth_array):
        # Ідентифікуємо пікселі рослин
        plant_pixels = np.where(mask == 1)

        if len(plant_pixels[0]) > 0:
            # Визначаємо глибини верхньої та нижньої частин
            plant_depths = depth[plant_pixels]
            top_depth = np.min(plant_depths)
            bottom_depth = np.max(plant_depths)

            # Обчислюємо висоту
            height = bottom_depth - top_depth
            heights.append(height)
        else:
            # Якщо рослин немає, додаємо 0
            heights.append(0)

    # Згладжуємо дані висоти для зменшення шуму
    kernel = np.ones(5) / 5  # Ядро згладжування (усереднення)
    heights_smoothed = np.convolve(heights, kernel, mode='same')

    return heights_smoothed.tolist()
```

Рисунок 3.8 – Фрагмент коду для обчислення висоти рослин

Ця функція оцінює висоту рослин за допомогою сегментаційних масок і карт глибини – mask_array (масив масок сегментації) та deep_array (масив карт глибини, що відповідає маскам).

Для створення функції обчислення висоти рослин без інтеграції карти глибини виконувався б подібний процес, за винятком того, що замість обчислення висоти як різниці між нижнім і верхнім значення глибини обчислення полягало б у пошуку верхнього та нижнього положень пікселя та обчисленні різниці між двома.

3.1.7. Сегментація фону

Використання InSPyReNet для наших цілей здійснюється за допомогою адаптованої версії, спеціально запропонованої для видалення фону [12]. Для цього встановлюємо вимоги, зазначені у файлі requirements.txt, який міститься в

репозиторії. Після того як очистимо вимоги до імпорту та імпорт, необхідний для запуску інструменту. Наступним кроком буде використання коду, спеціально адаптованого для відео. Код успішно гарантує, що кожен кадр обробляється, використовуючи цикл для запису кожного кадру в змінну під назвою fps.

3.2. Результати сегментації листя однієї рослини

Одним із накладених обмежень цієї роботи є використання навчальних даних, які не є репрезентативними для середовища Swegreen X. Сегментацію листя рослин виконували навчаючись на даних зображення орхідей, де одна рослина спрямована фронтально до камери, і лише одна рослина знаходиться в сцені. Ці дані служать хорошою відправною точкою, оскільки характеристики середовища зображень різко контрастують із захараченим середовищем рослин, які закривають одна одну, перекриваючи один одного, як це відбувається у типовому середовищі теплиці.

Цей набір даних зосереджується лише на продуктивності сегментації листя, але важливо зберегти інші нерелевантні класи, щоб оцінити, наскільки добре виконується сегментація листя, коли інші компоненти присутні в сцені. Тому всі інші класи не були вилучені з графіків і можуть розглядатися як еталонні.

Набір даних, який було використано, це інший набір даних від Roboflow, який містить зображення рослин орхідей, одна рослина для кожного зображення, що містить загалом 758 зображень із 663 зображеннями в наборі поїздів, 63 зображеннями в наборі перевірки та 32 зображеннями в тестовому наборі [24].

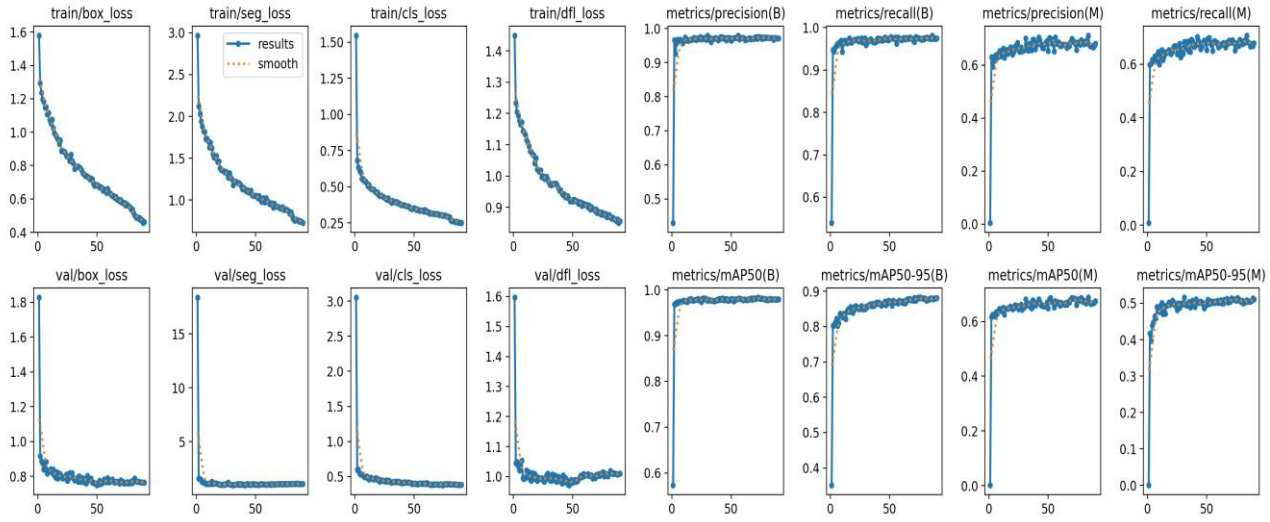

Рисунок 3.9 – Загальні результати навчання всіх показників

Як показано на рисунку 3.9, втрати сегментації при навчанні демонструють тенденцію до зменшення приблизно від трьох до одиниць і вказують на те, що втрати послідовності для сегментації починаються великими, але встигають значно зменшитися, тоді як втрати підтвердження для сегментації різко зменшуються, але майже відразу, на 3-й епосі вирівнюються і залишаються рівними протягом періоду навчання.

Це може свідчити як те, що модель швидко досягає конвергенції, але також може бути те, що модель не може покращити своє навчання на наборі перевірки лише через кілька епох. Це також може свідчити, що оскільки втрата перевірки постійна, але втрата навчання продовжує зменшуватися, модель добре працює на даних навчання, але не так добре на даних перевірки.

Рисунок 3.10 – Сегментація рослинних компонентів на одній рослині

Для рисунку 3.10 створюється кілька масок – одна для листя, одна для горщика та одна для рослини. Сама рослина сегментується разом із листям на рослині. Щоб уточнити, наскільки добре сегментовано листя, потрібно переглянути додаткові графіки ефективності. Але нас цікавить лише листковий клас.

На рисунку 3.11 сегментація листа демонструє високу точність (близьку до одиниці) у діапазоні відкликання від 0,9 до 1. Це показує, що для цих порогових значень модель є високоточною у своїх прогнозах для класу листів.

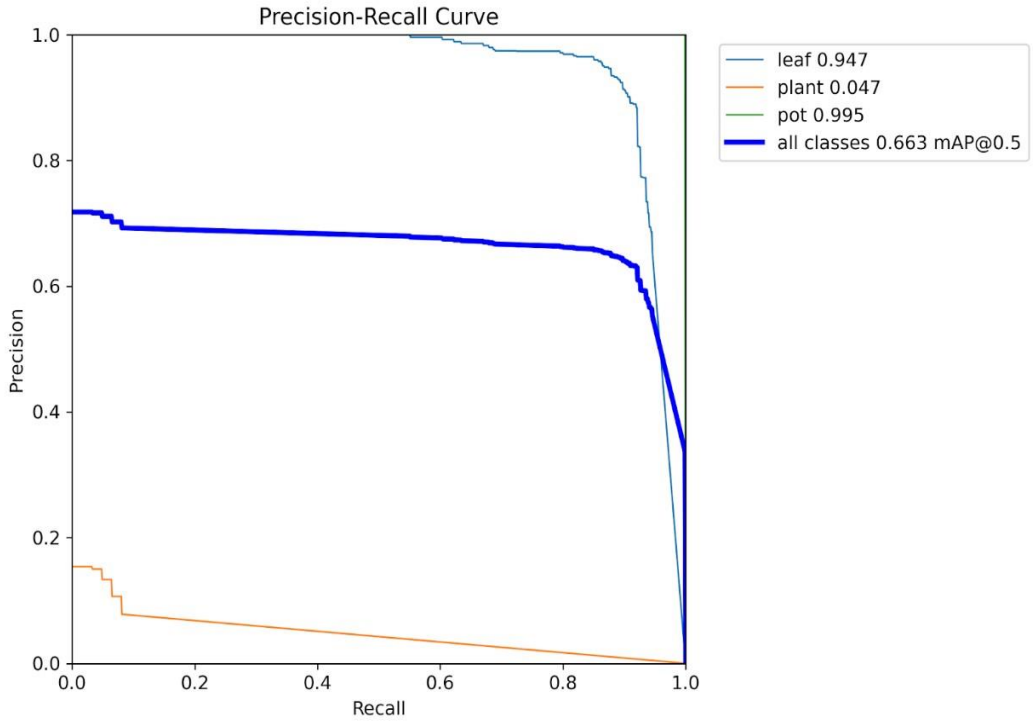

Рисунок 3.11 – Крива точного відкликання окремої установки

Щоб детальніше розглянути справжні позитиви, можна зробити візуалізацію за допомогою матриці плутанини, як показано на рисунку 3.12.

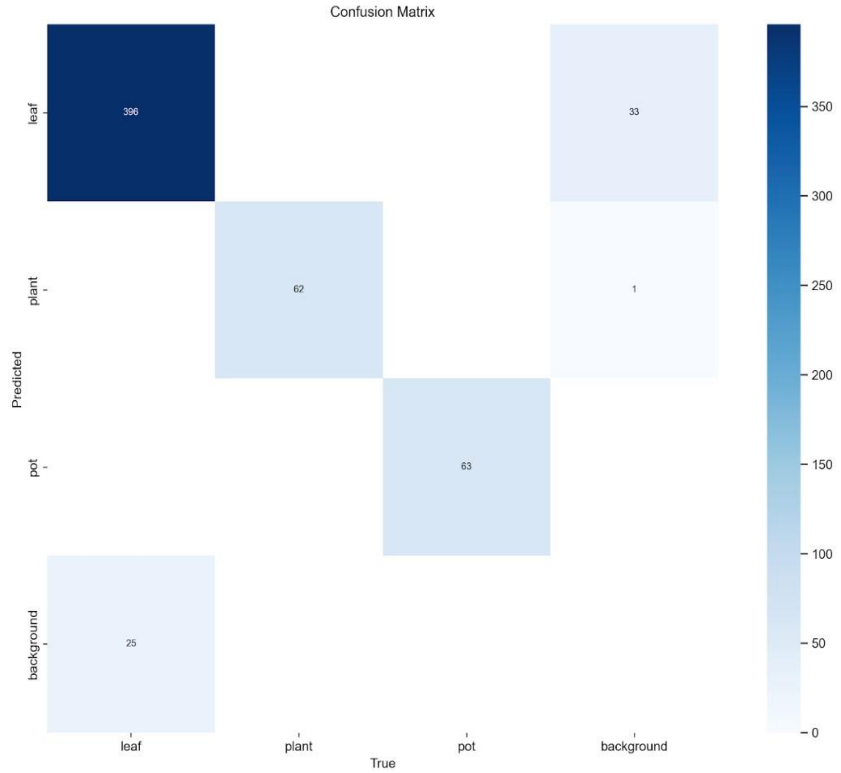

Рисунок 3.12 – Матриця плутанини

Рядки цієї матриці представляють фактичні класи (справжні мітки), а стовпці представляють прогнозовані класи. Для листового класу було позначено 396 екземплярів, що належать до листового класу. 33 екземпляри з класу листя були неправильно класифіковані, як приналежні до класу рослин (помилково негативні). 25 екземплярів були неправильно класифіковані, як приналежна до фону. Значення цих результатів показує, що модель дуже ефективна під час виявлення екземплярів листя. Однак їй важко розрізнити листки і рослину та вона має тенденцію неправильно класифікувати фон як листки.

Як видно на рисунку 3.13, оцінка F1 для листя залишається відносно високою порівняно з іншими класами, перевищуючи 0,8 у широкому спектрі порогів достовірності. Оцінка F1 є показником точності моделі в тому, як вона балансує між точністю та запам’ятовуванням. Оскільки точність вимірює частку справжніх позитивних прогнозів серед усіх позитивних прогнозів, зроблених моделлю, а відкликання є мірою частки справжніх позитивних прогнозів серед

усіх фактичних позитивних випадків у наборі даних, значення F1 становить 0,8 по відношенню до вищезазначеного.

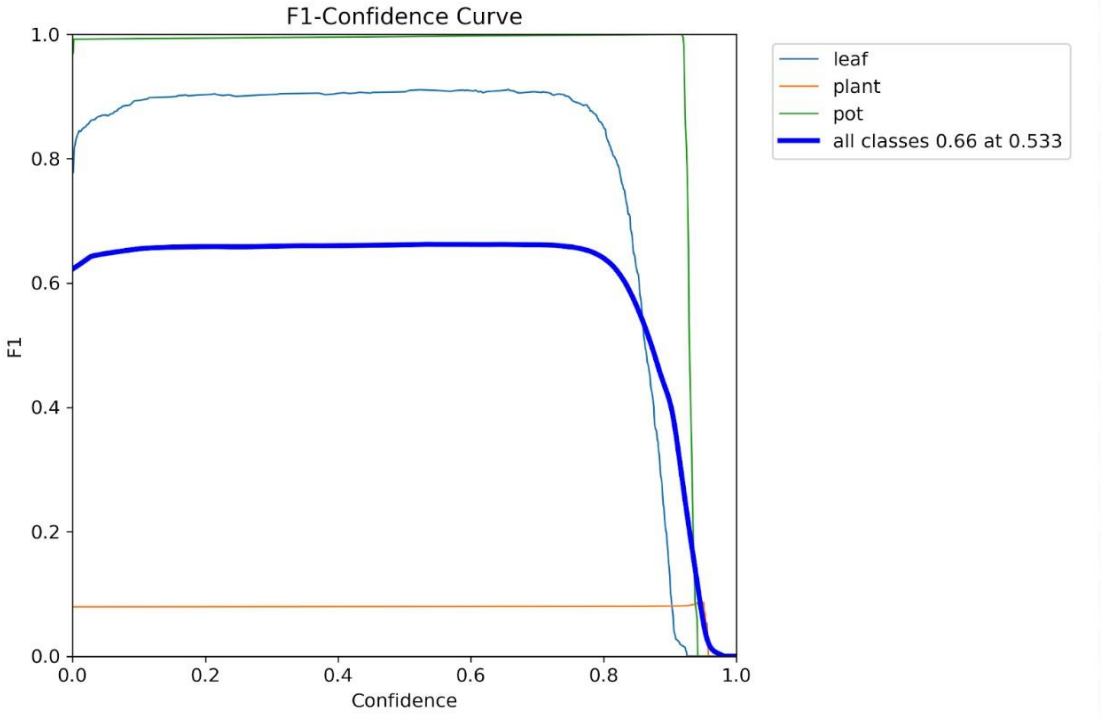

Рисунок 3.13 – Результати оцінки показника точності моделі F1 для відповідного класу листя

Згадані визначення точності та відкликання означають, що модель знаходить хороший баланс між виявленням справжніх позитивних результатів (пригадування) і уникненням помилкових позитивних результатів (точність), коли справа стосується клас листя. Близько 0,9 оцінка F1 значно знижується.

Причина цього зниження полягає в тому, що зі збільшенням порогу достовірності, що можна розглядати як збільшення кількості прогнозів, які можна зробити для заданого порогу достовірності, кількість прогнозованих випадків зменшується, що спричиняє зниження запам’ятовування, а це призводить до F1 значення для спадання.

Оцінюючи точно налаштований YOLOv9e на основі даних, отриманих від Swegreen, стає зрозуміло, що захоплення однієї рослини, як видно в наборі даних, надто відрізняється від реального середовища теплиці.

Рисунок 3.14 – Сегментація екземпляра після того, як YOLOv9e-seg було налаштовано на наборі даних Single plant

Як видно на рисунку 3.14, лише кілька листків сегментуються з дуже низькою достовірністю, де порогову змінну достовірності потрібно було зменшити до значення 0,3. Налаштована модель також реагує на відбиваючі поверхні та інші несуттєві об’єкти. Щоб покращити сегментацію листя, потрібно використовувати набір даних, де реальне тепличне середовище є дещо більшим.

3.3. Результати сегментації листя в умовах теплиці

Процес навчання моделі YOLOv9e-seg для сегментації листя рослин у середовищі, більш типовому для теплиці, з листям, що закривають одне одного та сцену, заповнену більшою щільністю об’єктів, ніж у прикладі однієї рослини, включали серію з 60 епох. Щоб оцінити продуктивність моделі використали набір даних сегментації листків рослин Roboflow [20]. Це включає запис втрат і вимірювання точності для етапів навчання та перевірки. Записані показники включають втрату блоку, втрату сегментації, втрату класифікації та втрату фокусу розподілу (dfl_loss). Також вимірюється точність, відкликання та середня точність (mAP). Назви класів Plant1 і Plant2 не стосуються конкретних рослин, і оскільки цей підхід до сегментації листя не залежить від виду рослини, іменування класів не має значення.

Розглянемо втрати під час навчання. Процес навчання моделі YOLOv9e-seg включає різні показники для оцінки її продуктивності протягом 60 епох:

1. Box Loss – цей показник показує, наскільки добре модель передбачає координати обмежувальної рамки для листя рослини. Він починається з 1,8164 і демонструє загальну тенденцію до зниження, закінчуючись на 0,60972 до епохи 60;

2. Втрата сегментації – починаючи з 4,1026, втрата також зменшується з епохами, закінчуючи на 0,91147;

3. Втрата класифікації – втрата вимірює здатність моделі правильно класифікувати сегменти як листя рослини. Починається з 3,8278 і знижується до 0,53133;

4. DFL Loss – починаючи з 1,8355, втрата зменшується до 0,94834 до кінця періоду навчання.

Кожна метрика представлена з її початковими та кінцевими значеннями, що демонструє поступове вдосконалення моделі в роботі зі складнощами сегментації листя в захараченому середовищі (рис. 3.15).

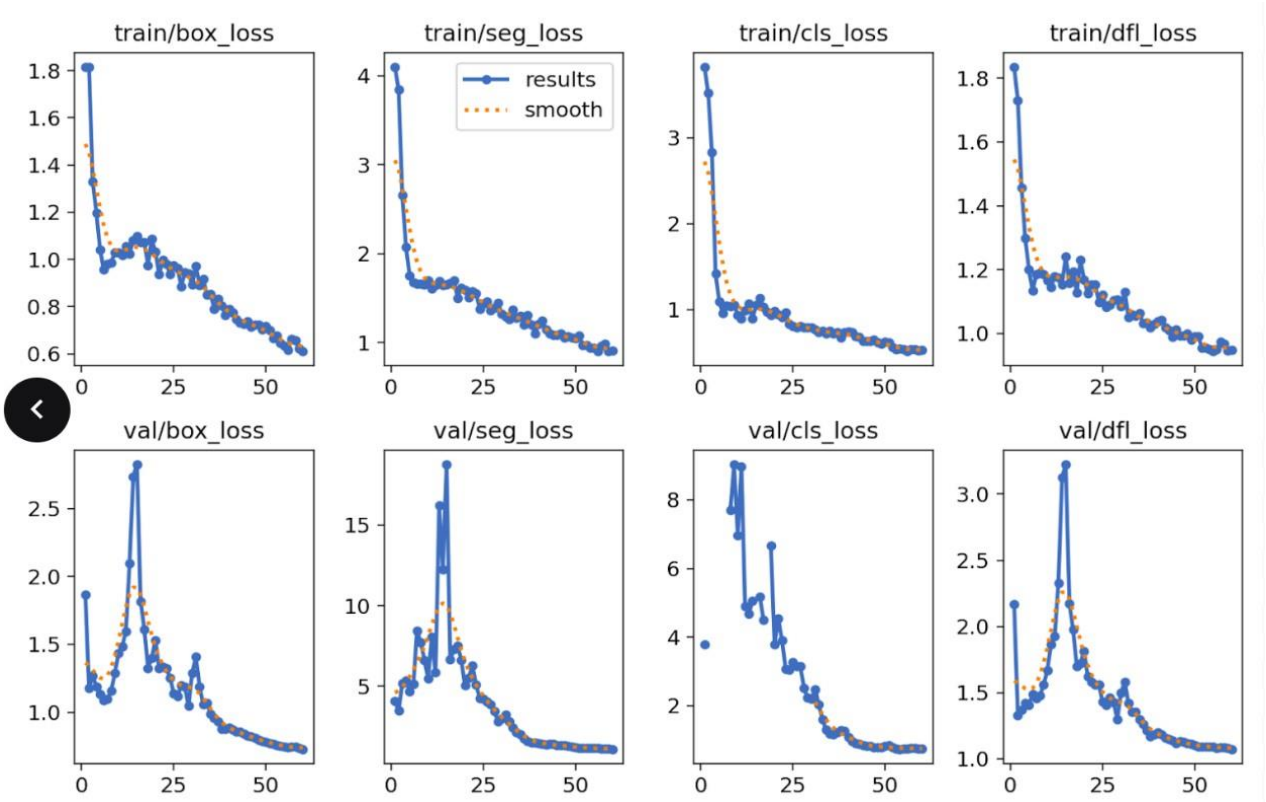

Рисунок 3.15 – Загальні результати YOLOv9e-seg – налаштування [20]

Як показано на рисунку 3.15, втрати мають подібну тенденцію та починають виявлятися приблизно через 50 епох, але зі значними сплесками втрат під час перевірки для епох 13, 15 і 24. Втрати DFL також сильно зростають, але починають мати повне послідовне зменшення втрат близько епохи 30. Ці тенденції можна на загальному рівні візуалізувати в прогнозах, зроблених на вибірці з набір даних:

Рисунок 3.16 – Сегментація листя, виконана на багатьох листках за допомогою налаштованого YOLOv9e-seg

Як видно на рисунку 3.16, сегментація досягає рівня точності, коли листя рослини окреслюються та відокремлюються від інших відповідно до координат обмежувальної рамки.

Показники точності прогнозів як для обмежувальної рамки (B), так і для маски сегментації (M) починаються з низького рівня, але спостерігаються суттєві покращення, досягаючи піку в пізніші епохи (наприклад, 0,8363 для обмежувальної рамки та 0,83847 для маски в епоху 60). Значення запам’ятовування також значно покращуються, як можна побачити на рисунку 3.17.

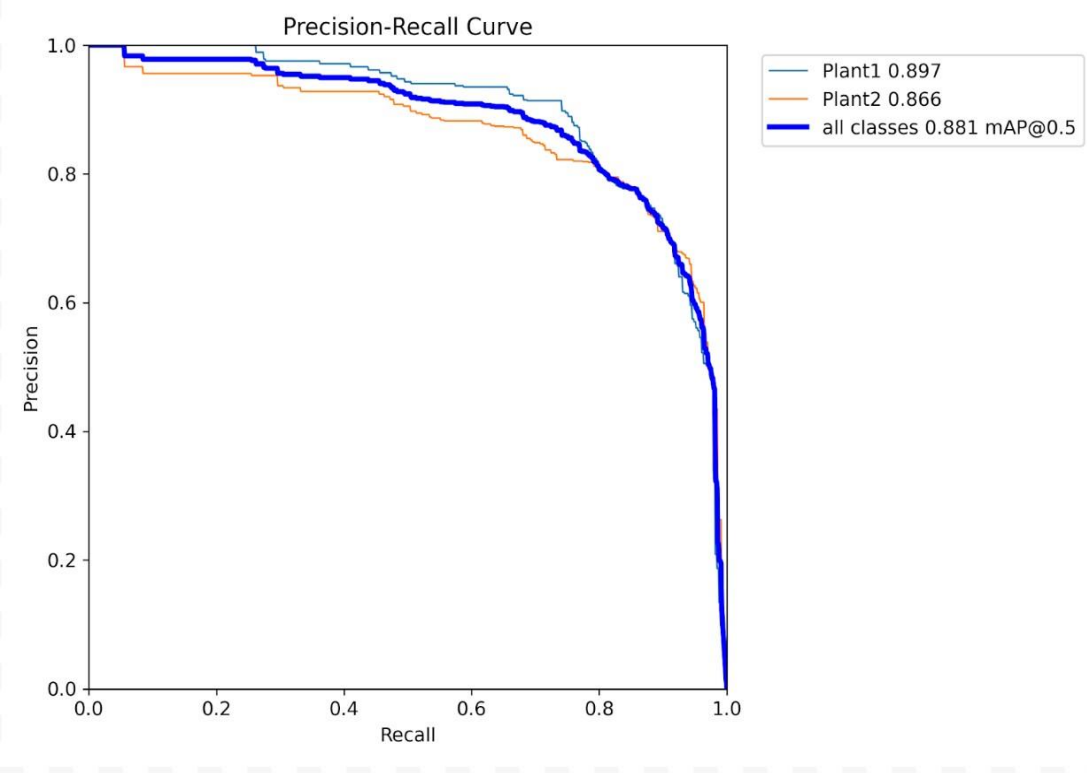

Рисунок 3.17 – Крива точності-відклику

Середня точність (mAP) mAP50 і mAP50-95 – mAP50 враховує 50-відсоткове порогове значення IoU, а mAP50-95 усереднює порогові значення IoU від 50 до 95 відсотків. Обидва показники демонструють покращення за зростання епох, що вказує на надійність моделі у виявленні та точному сегментуванні листів на різних рівнях IoU. Для mAP50 (B) він почався з 0,013 і покращувався протягом епох, а закінчився на 0,879 для 50-відсоткового порогового значення IoU. Для mAP50-95(B) значення почалося з 0,009 для першої епохи і згодом

покращилося протягом епох до 0,713 в епоху 60. mAP50 (M) – почалося з 0,012 і збільшилося до 0,881 до останньої епохи. Для mAP50-95 він починався з 0,008 для першої епохи і закінчувався значенням 0,705 до епохи 60.

Оцінка F1 також використовується як метрика ефективності. Оцінка F1 починається приблизно з 0,21 у першу епоху, і з часом стає кращою в сегментації, за винятком деяких коливань на ранніх і середніх етапах навчання. До останньої епохи оцінка F1 досягла 0,8220 для зображень, як можна побачити на рис. 3.18.

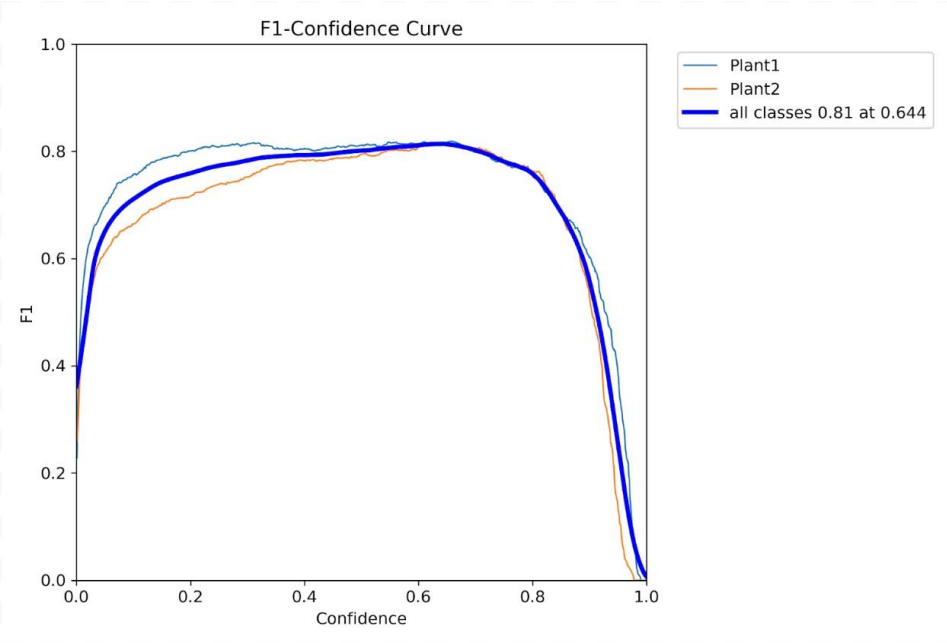

Рисунок 3.18 – Результати оцінки показника точності моделі F1 для умов теплиці

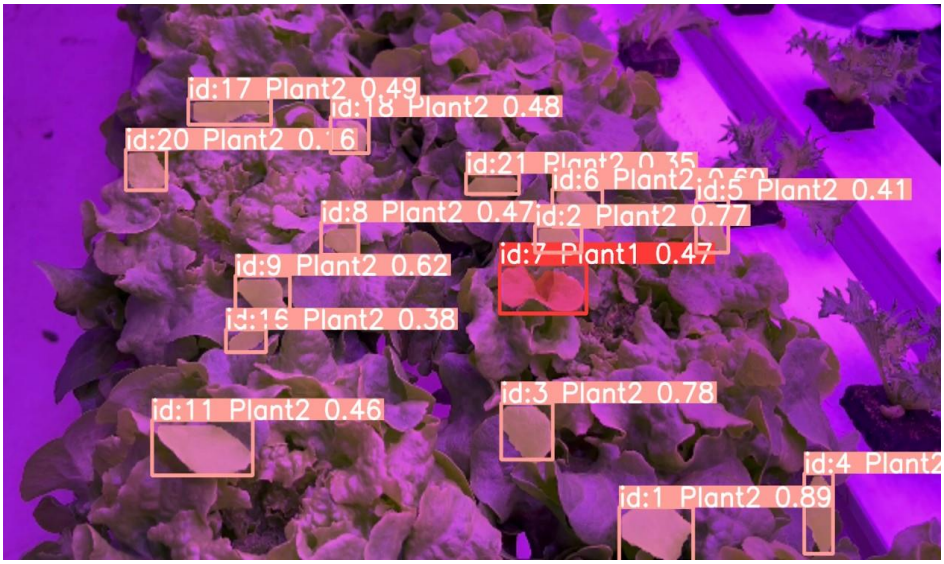

Рисунок 3.19 – Налаштована модель YOLOv9e-seg, використана для салату у теплиці з алгоритмом ByteTrack для відстеження об’єктів

На рисунку 3.19 подана оцінка результатів, отриманих за допомогою точно налаштованої моделі сегментації листя. Це стосується найбільш релевантних даних – всередині теплиць Swegreen X.

На рисунку 3.19 вище було випробувано сегментацію листя на ряді із салатом у теплиці, що є значно складнішим і менш точним. Це пов’язано із тим, що рослина салату зазвичай не має таких же характеристик фенотипу листя порівняно з іншими зеленими листками, такими як рослини базиліка.

Нові дані з теплиці суттєво відрізняються від тренувальних даних, на яких було виконано тонке налаштування, що також може бути причиною будь-яких неточностей у сегментації листя на рисунку 3.19. Стосовно ByteTrack, відповідального за відстеження кожного об’єкта, на рисунку 3.19 видно, що розподіл ідентифікаторів не є повністю точним, і він стає більш неточним, оскільки кадрова прогресія триває протягом більших періодів.

Отже, очевидно, що послідовне відстеження не досягається за допомогою ByteTrack для цього прикладу, як показано на рисунку 3.19 вище. Механізм повторної ідентифікації потрібен, якщо ті самі ідентифікатори перерозподіляються, навіть якщо відстеження ідентифікатора втрачено. Такі механізми можна знайти в таких роботах, як Deep OC Sort [14].

РОЗДІЛ 4.
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА У НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1. Аналіз стану процесів у теплицях та прогнозування травмонебезпечних ситуацій

Тепличні господарства є важливою галуззю сільського господарства, яка забезпечує населення свіжими овочами, фруктами та квітами незалежно від сезонності. Проте умови роботи в теплицях часто пов’язані з підвищеним ризиком травмування через вплив складних факторів, таких як висока температура, вологість, контакт із хімічними засобами, механічні травми тощо. У цьому підрозділі проведено аналіз основних небезпек у теплицях та запропоновано методи прогнозування травмонебезпечних ситуацій для зниження їх імовірності.

Теплиці характеризуються специфічними умовами, які можуть створити небезпечну ситуацію (табл. 4.1).

Таблиця 4.1 – Небезпечні складові у тепличних комплексах

Складова	Характеристика
Температурні ризики	Робота в умовах високої температури до теплового стресу, перегріву та зниження концентрації уваги
Вологість	Підвищена вологість сприяє розвитку грибкових захворювань у працівників і погіршує видимість, що закінчується ризиком травмування
Хімічні фактори	Контакт із добривами, пестицидами або гербіцидами може викликати хімічні опіки або отруєння
Механічні травми	Неправильна експлуатація обладнання, пошкодження обладнання або недостатня освітленість можуть стати причинами падіння або пошкодження
Біологічні фактори	Контакт із рослинами, що містять алергени, або збудниками хвороби

Для ефективного прогнозування травмонебезпечних ситуацій розроблено математичні моделі та методи машинного навчання. Це дозволяє аналізувати

історичні дані про нещасні випадки – виявлення найбільш небезпечних зон та умов роботи. Визначати параметри мікроклімату теплиці – температура, вологість, концентрація CO_2. Оцінювати фактори людського впливу – графіки роботи, рівень кваліфікації, дотримання правил безпеки.

Пропонується використання датчиків для моніторингу мікроклімату та автоматичного попередження про відхилення параметрів від допустимих норм. Окрім того, слід виконувати аналіз ризиків за допомогою кластеризації та виявлення аномалій у поведінці персоналу чи стані обладнання. Проводити імітаційне моделювання для тестування різних сценаріїв роботи в теплицях.

4.2. Заходи для забезпечення безпеки працівників у теплицях

На основі проведеного аналізу пропонуємо такі заходи безпеки працівників, які представлено у табл. 4.2.

Таблиця 4.2 –Заходи для забезпечення безпеки працівників у теплицях

Заходи	Характеристика
Технічні заходи	✓ Встановлення вентиляційних систем та датчиків для контролю мікроклімату. ✓ Обладнання теплитись нековзкими поверхнями та безпечним освітленням.
Організаційні заходи	✓ Навчання персоналу правилам безпеки. ✓ Ротація працівників для зниження впливу екстремальних умов.
Індивідуальний захист	✓ Забезпечення працівників відповідним одягом, рукавицями, респіраторами.

Забезпечення безпеки праці у теплицях вимагає комплексного підходу, який включає прогнозування ризиків, використання сучасних технологій моніторингу та впровадження ефективних заходів безпеки. Прогнозування

травмонебезпечних ситуацій зменшує зменшення кількості нещасних випадків і підвищує продуктивність праці.

4.3. Інструкція із охорони праці під час монтажу та обслуговування системи моніторингу стану рослин у теплицях

Загальні положення

Інструкція має основні вимоги з охорони праці під час монтажу та обслуговування системи моніторингу стану рослин у теплицях. Вона спрямована на забезпечення безпеки працівників, запобігання травмам та аварійним ситуаціям.

Вимоги цієї інструкції є обов'язковими для всіх працівників, які беруть участь у монтажі, обслуговуванні, ремонті чи перевірці роботи системи моніторингу.

1. Вимоги до організації робіт

Перед початком робіт необхідно провести інструкцію з техніки безпеки для працівників. Усі роботи повинні виконуватися відповідно до затвердженого плану, з урахуванням специфіки теплиці та розташування обладнання. Заборонено підтримувати роботу в умовах підвищеної небезпеки (висока температура, підвищена вологість) без спеціальних засобів індивідуального захисту.

2. Вимоги до працівників

До виконання робіт допускаються особи, які:

✓ Пройшов медичний огляд і визнано придатними для виконання відповідних робіт.

✓ Пройшли навчання з охорони праці, ознайомтеся з цією інструкцією.

✓ Забезпечені засобами індивідуального захисту (каски, рукавиці, ізолюючі підкладки тощо).

Працівники повинні знати:

- ✓ Основи роботи з електронним обладнанням.
- ✓ Правила безпечного поводження з електроінструментами.
- ✓ План евакуації на випадок надзвичайної ситуації.

3. Вимоги під час монтажу системи

Підготовка робочого місця:

- ✓ Забезпечити чистоту робочої зони, прибрати зайві предмети, які можуть заважати монтажу.
- ✓ Встановити стабільне освітлення у теплиці.

Робота з обладнанням передбачає наступне. Заборонено підключати систему моніторингу до електромережі без перевірки ізоляції кабелів. Монтаж кабелів і датчиків слід проводити з використанням ізолюючих матеріалів. Забороняється використовувати обладнання з видимими пошкодженнями.

Висотні роботи слід проводити із врахуванням наступного. Роботи на висоті понад 1,5 метра передбачають викоритання страхових засобів. Драбини або інші підйомні конструкції повинні бути міцно закріплені.

4. Вимоги під час обслуговування системи

Перед початком обслуговування відключити систему від електромережі. Використовувати тестове обладнання для перевірки працездатності датчиків і кабелів.

Для очищення датчиків використовують лише спеціальні засоби, рекомендовані виробником. Забороняється очищувати обладнання у ввімкненому стані.

У разі виявлення несправності повідомити керівництво та припинити використання обладнання. Ремонт повинен виконувати лише кваліфіковані спеціалісти.

5. Дії в разі надзвичайних ситуацій

У разі пожежі, витоку хімічних речовин або іншої небезпеки:

- ✓ Негайно припинити роботу та відключити систему від електромережі.
- ✓ Евакуювати працівників із теплиці відповідно до плану евакуації.

✓ Повідомити відповідальні служби (пожежна охорона, медична допомога).

У разі травмування:

✓ Надати першу медичну допомогу постраждалому.

✓ Повідомити керівництво та викликати медичну допомогу.

6. Відповідальність

Працівники, які порушили вимоги цієї інструкції, не несуть відповідальності згідно з чинним законодавством. Роботодавець зобов’язаний забезпечити належні умови праці, навчання та постачання засобів захисту.

Дотримання інструкції з охорони праці є основою для безпечного виконання робіт із монтажу та обслуговування системи моніторингу в теплицях. Це забезпечує зниження ризику травматизму, підвищення ефективності робіт і запобігання аварійним ситуаціям.

4.4. Заходи безпеки у надзвичайних ситуаціях

Під надзвичайною ситуацією розуміють порушення нормальних умов життя і діяльності людей, об’єктів або територій унаслідок аварій, катастроф, стихійних лих або інших чинників, що спричинили або можуть спричинити загибель людей та значні матеріальні втрати.

Головною функцією адміністрації господарства у разі виникнення надзвичайної ситуації є захист населення та організації його життєзабезпечення.

Заходи щодо захисту цивільного населення плануються проводяться по населених пунктах де розміщені господарства і охоплюють населення навколишніх сіл. Водночас характер та зміст захисних засобів встановлюються вид ступеня загрози, місцевих умов з урахуванням важливості виробництва для безпеки населення і інших економічних і соціальних чинників.

Основні заходи щодо захисту населення плануються та здійснюються завчасно і мають випереджувальний характер, це стосується насамперед

підготовки, підтримання у постійній готовності індивідуальних та колективних засобів захисту, їх накопичення, а також підготовки до проведення евакуації населення із зон підвищеного ризику.

Керівництво господарства є безпосередніми виконавцями цих заходів, у нашому господарстві розробляються завчасно, проводиться навчання робітників та службовців способам захисту та діям в умовах надзвичайних ситуацій.

Також раз в три роки проводяться навчання по підготовці близьких до військових дій, що в разі небезпеки могло би не дістати людину зненацька. Керівництво докладає максимум зусиль, щоб працівники господарства були хоча би мінімально захищенні в разі будь-якої небезпеки пов’язаної з тими чи іншими обставинами.

РОЗДІЛ 5.
ЕКОНОМІЧНА ЕФЕКТИВНІСТЬ ВІД МОНІТОРИНГУ СТАНУ РОСЛИН У ТЕПЛИЦЯХ НА ОСНОВІ АЛГОРИТМІВ ГЛИБОКОГО НАВЧАННЯ ДЛЯ СЕГМЕНТАЦІЇ

Моніторинг стану рослин у теплицях із використанням алгоритмів глибокого навчання забезпечує високу точність оцінки стану рослин, автоматизацію процесів, а також значне зниження витрат на ручний огляд і лікування хвороб рослин. Економічна ефективність системи визначається з урахуванням таких факторів, як скорочення втрат врожаю, економія часу, зменшення витрат на хімічні засоби захисту рослин і підвищення продуктивності праці.

Методика оцінки економічної ефективності полягає у наступному. Для оцінки економічної ефективності використовується показник чистого економічного ефекту ($ЧЕЕ$), який розраховується за формулою:

$$ЧЕЕ = \Delta B - \Delta Bm, \qquad (5.1)$$

де ΔB – додатковий дохід від впровадження системи, грн; ΔB_m – витрати на впровадження та обслуговування системи, грн.

Додатковий дохід **(**ΔB**)** визначається за формулою:

$$\Delta B = \Delta У_р \cdot Ц, \qquad (5.2)$$

де $\Delta У_р$ – збільшення врожайності завдяки покращенню умов, кг; $Ц$ – ціна реалізації продукції, грн/кг.

Витрати на впровадження **(**ΔB_m**)** включають витрати на обладнання ($О$), витрати на навчання персоналу ($Н$), щорічні витрати на обслуговування ($О_б$).

$$\Delta B_m = О + Н + О_б, \qquad (5.3)$$

Розрахунок економічної ефективності проводимо із використанням вище поданих формул. Вхідні дані:

Площа теплиці – 10000 м².

Врожайність до впровадження системи – 20 кг/м².

Збільшення врожайності після впровадження – 10%.

Ціна реалізації продукції – 40 грн/кг.

Вартість системи (обладнання та встановлення) – 500000 грн.

Вартість навчання персоналу – 50000 грн.

Щорічні витрати на обслуговування – 100000 грн.

Скорочення витрат на хімічні засоби – 20%.

Частка витрат на хімічні засоби у загальних витратах – 15%.

Додатковий дохід (ΔB) визначається за формулою (5.2). При цьому збільшення врожайності $\Delta У_р$ становить:

$$\Delta У_р = 10000 \cdot 20 \cdot 0.10 = 20000 \text{ кг.}$$

Додатковий дохід становить:

$$\Delta B = 20000 \cdot 40 = 800000 \text{ грн.}$$

Для встановлення зменшення витрат на хімічні засоби потрібно визначити загальні витрати на хімічні засоби до впровадження:

$$З_{хім} = 10000 \cdot 20 \cdot 15\% \cdot 40 = 1200000 \text{ грн.}$$

Зменшення витрат становить:

$$\Delta З_{хім} = 1200000 \cdot 0.20 = 240000 \text{ грн.}$$

Витрати на впровадження ($\Delta B_т$) системи:

$$\Delta B_т = 500000 + 50000 + 100000 = 650000 \text{ грн.}$$

Чистий економічний ефект ($ЧЕЕ$) становить:

$$ЧЕЕ = \Delta B + \Delta Зхім - \Delta Bт = 800000 + 240000 - 650000 = 390000 \text{ грн.}$$

Отримані результати визначення економічної ефективності від моніторингу стану рослин у теплицях на основі алгоритмів глибокого навчання для сегментації подано у табл. 5.1.

Таблиця 5.1 – Економічна ефективність від моніторингу стану рослин у теплицях на основі алгоритмів глибокого навчання для сегментації

Показник	Одиниці виміру	Значення
Площа теплиці	м²	10000
Врожайність до впровадження	кг/м²	20
Збільшення врожайності	кг	20000
Ціна реалізації продукції	грн/кг	40
Додатковий дохід (ΔB)	грн	800000
Вартість системи (O)	грн	500000
Вартість навчання (H)	грн	50000
Щорічні витрати ($O_б$)	грн	100000
Зменшення витрат на хімічні засоби	грн	240000
Чистий економічний ефект ($ЧЕЕ$)	грн	390000

Результати розрахунків свідчать про значну економічну вигоду від впровадження системи моніторингу стану рослин. Основними джерелами економії є збільшення врожайності завдяки своєчасному виявленню хвороб та дефіцитів, зменшення витрат на хімічні засоби через точкове застосування пестицидів і добрив.

Чистий економічний ефект у розмірі $ЧЕЕ = 390000$ грн свідчить про рентабельність інвестицій у систему. Термін окупності становить менше одного року, що робить впровадження моніторингу стану рослин у теплицях на основі алгоритмів глибокого навчання для сегментації надзвичайно перспективним для великих тепличних господарств.

ВИСНОВКИ І ПРОПОЗИЦІЇ

Наша робота скерована на вибір та налаштування моделей глибокого навчання для сегментації зображення рослин у теплицях, що дозволяють дозволити автоматичний моніторинг їх стану, виявити хвороби та дефекти, а також прогнозувати подальший розвиток рослин. Важливою складовою роботи є порівняння різних підходів до сегментації та вибір найбільш ефективного рішення для практичного використання в аграрних умовах.

Нами обґрунтовано доцільність моніторингу стану рослин у теплицях на основі машинного навчання. Встановлено, що моделі глибокого навчання, які показують хорошу прогностичну продуктивність, вимагають високопродуктивних графічних процесорів або паралельної обробки зображень. При цьому вони сьогодні мають практичне застосування і можуть використовуватися в більшості реальних теплиць.

На відміну від «чорної скриньки» моделей глибокого навчання мають структуру засновану на посиленні, легше інтерпретуються, що полегшує розуміння важливості змінних і їх впливу на прогнози. Було запропоновано та успішно застосовано для прогнозування багато різноманітних методів нейронних мереж (включаючи ANN, RNN та NARX). Порівнювали структурні відмінності в алгоритмах навчання на основі часу (рис. 1.5).

Для тепличних комплексів розроблено модель ШНМ з точністю R2=0,9, враховуючи середню температуру, відносну вологість, інтенсивність світла, вік рослин, площу листя та висоту рослин як вхідні параметри для прогнозування висоти рослин дині та площі листя через два дні.

Аналіз існуючих систем моніторингу стану рослин у теплицях виявив їхні обмеження, зокрема низьку точність і вузьку функціональність. Впровадження моделей глибокого навчання для сегментації зображень є доцільним завдяки їхній здатності забезпечувати високу точність, автоматизацію процесів і універсальність. Це дозволяє зменшити втрати врожаю, оптимізувати використання ресурсів і підвищити ефективність тепличного господарства.

У роботі виконано дослідження щодо можливості створення системи моніторингу здоров’я рослин завдяки обґрунтуванню та налаштуванню моделей глибокого навчання. Очікуваним результатом є інтелектуальна система, яка виконує низку завдань у теплицях (рис. 2.1).

Незважаючи на постійний прогрес, сучасніші навчені моделі ViT (Vision Transformer) зазвичай навчаються на зображеннях 224x224 або 384x384 пікселів. У цих низьких роздільних здатностях важко відстежувати невеликі, дрібні зміни в деталях об’єкта, що контролюється (рослина у теплиці).

Відомо, що алгоритм K-means (рисунок 2.2) успішно сегментує рослини у теплиці, хоча є певний фоновий витік з об’єктів, не пов’язаних з рослинами теплиці. Однак у більш складних сценаріях швидко стає очевидним, що групування пікселів на основі подібності або мінімального чи максимального простору, не вдається чітко виділити потрібні об’єкти (рослин), як це видно на рисунку 2.3. Зображення на рисунку 2.3 є сповільненою зйомкою. Відео з вертикальної теплиці, де кілька вирощуваних рослин складаються у одне зображення. Для обробки складнішого динамічного зображення потрібні кращі методи сегментації фону.

Нами побудовано графік змін висоти рослин. Нахил вниз у кінці є артефактом непослідовного відстеження, викликаного неоднозначністю в зображенні поводження з об’єктом і фоном, або іншими формами невідповідностей, які модель інтерпретує як продовження моделі росту рослини.

Використання обмежувальних прямокутників для виявлення стану здоров’я рослин має обмеження через низьку специфічність визначення області дефектів. Для точнішого виявлення стану рослин доцільно застосовувати методи, які мінімізують нерелевантні області зображення, як це забезпечує використання Roboflow Plant Leaf Dataset Version 2.

Використання DeepPlants AGM-HS для сегментації листків демонструє обмеження моделі у точності та послідовності виявлення дефектів. Сегментація часто застосовується до неправильного листка, а дефектні області залишаються невиявленими, як видно з невідповідності результатів у різних циклах. Це вказує

на потребу в подальшому вдосконаленні моделі для підвищення точності та стабільності її роботи.

Запропонована нами система моніторингу рослин орієнтована на широке впровадження в тепличних господарствах, використовуючи лише стандартну відеокамеру для збору даних. Навчання моделі базується на діагностичних наборах даних із різних тепличних середовищ, що забезпечує її універсальність і адаптивність, як зазначено в таблиці 2.1.

Для налаштування моделей моніторингу стану рослин у теплицях було обрано кілька підходів, які відрізняються своїми технічними характеристиками. Модель YOLOv9e забезпечує базове визначення об’єктів, тоді як YOLOv9e-seg додає можливість окреслення меж об’єктів. Модифікація YOLOv9e-seg із SAHI дозволяє ефективно обробляти великі й деталізовані зображення. FastSAM-X із ByteTrack забезпечує функціональну швидку сегментацію та відстеження об’єктів, забезпечуючи найбільш комплексний підхід до моніторингу. Такий вибір моделей дозволяє підвищити точність і продуктивність аналізу стану рослин у різних умовах.

Для виконання завдань моніторингу стану рослин у теплицях проведено налаштування та навчання кількох моделей, зокрема YOLOv9, SAM, DINOv2, із фокусом на семантичну сегментацію, виявлення хвороб, підрахунок листків, обчислення висоти рослин та сегментацію фону. В процесі були підібрані відповідні бібліотеки та інструменти, а також розроблено програмний код, що забезпечує реалізацію цих завдань, створюючи комплексну та ефективну систему аналізу рослин у теплицях.

Нами виконано сегментацію листя однієї рослини. Результати сегментації листя рослини показують високу точність моделі для класу листя, що підтверджує значення відкликання в розділі від 0,9 до 1. Однак модель має обмеження, зокрема труднощі у розрізнянні листя від решти рослин та неправильній класифікації фону. Використаний навчальний набір даних, який не відповідає реальному тепловому середовищу, обмежує загальну ефективність. Для покращення результатів необхідне розширення набору даних за рахунок

зображення ізольованого теплового середовища, що допоможе зменшити вплив відбиваючих поверхонь і несуттєвих об’єктів на реальну сегментацію.

Процес навчання моделі YOLOv9e-seg для сегментації листя в умовах теплиці показав поступове вдосконалення за всіма основними метриками. Значення втрат для Box Loss, Segmentation Loss, Classification Loss і DFL Loss значно зменшилися протягом 60 епох, що свідчить про покращення точності, передбачених обмежувальними рамками, сегментацією та класифікацією листів у складному тепличному середовищі. Це підтверджує ефективність підходу та потенціал моделі для застосування в реальних умовах.

Показники точності прогнозів як для обмежувальної рамки (B), так і для маски сегментації (M) починаються з низького рівня, але спостерігаються суттєві покращення, досягаючи піку в пізніші епохи (наприклад, 0,8363 для обмежувальної рамки та 0,83847 для маски в епоху 60). Значення запам’ятовування також значно покращуються, як можна побачити на рисунку 3.17.

Аналіз прикладу показує, що ByteTrack не забезпечує відстеження хвороб рослин, особливо у випадку втрати відстеження. Для вирішення цієї проблеми необхідно виконувати повторну ідентифікацію, яка дозволяє перерозподіляти ідентифікатори навіть після їх втрати. Подібні підходи, як-от Deep OC Sort, можуть суттєво підвищити точність і стабільність спостереження за рослинами у теплиці.

Розроблені заходи з охорони праці та безпеки у надзвичайних ситуаціях під час монтажу інтелектуальної системи моніторингу стану рослин у теплиці сприяють зниженню негативних втрат і підвищенню рівня безпеки як для працівників, так і для об’єкта. Впровадження цих заходів дозволяє мінімізувати ризики травматизму, запобігти аварійним ситуаціям та створити безпечні умови для виконання робіт, що є важливим для ефективного функціонування системи.

Результати розрахунків підтверджують значну економічну вигоду від впровадження системи моніторингу стану рослин у теплицях. Основними джерелами економіки є підвищення врожайності за рахунок випадкового

виявлення хвороб і дефіцитів, а також зменшення витрат на хімічні засоби через точкове їх застосування. Чистий економічний ефект $ЧЕЕ = 390000$ грн і термін окупності менше року свідчать про високу рентабельність таких інвестицій, що робить цю технологію перспективною для широкого впровадження в тепличних господарствах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Agatonovic-Kustrin, S.; Beresford, R. Basic Concepts of Artificial Neural Network (ANN) Modeling and Its Application in Pharmaceutical Research. J. Pharm. Biomed. Anal. 2000, 22, 717–727.

2. Ahonen, T.; Virrankoski, R.; Elmusrati, M. Greenhouse Monitoring with Wireless Sensor Network. In Proceedings of the 2008 IEEE/ASME International Conference on Mechtronic and Embedded Systems and Applications, Beijing, China, 12–15 October 2008; pp. 403–408.

3. Alhnaity, B.; Pearson, S.; Leontidis, G.; Kollias, S. Using Deep Learning to Predict Plant Growth and Yield in Greenhouse Environments. Acta Hortic. 2020, 1296, 425–432.

4. Ferrández-Pastor F.J., Mora-Pascual J., Díaz-Lajara D. Agricultural traceability model based on IoT and Blockchain: Application in industrial hemp production. J. Ind. Inf. Integr., 29, 2022, Article 100381

5. Franchetti B. and Pirri F., Detection and localization of tip-burn on large lettuce canopies, Frontiers in Plant Science, vol. 13, May 2022. doi: 10.3389/fpls.2022.874035.

6. Ghorbel O., Frikha T., Alabdali R., Ayadi R., Abbas Elmasry M. Blockchain-Based Supply Chain System for Olive Fields Using WSNs. Comput. Intell. Neurosci. 2022.

7. Hearst, M.A.; Dumais, S.T.; Osuna, E.; Platt, J.; Scholkopf, B. Support Vector Machines. IEEE Intell. Syst. Their Appl. 1998, 13, 18–28.

8. Igarashi T., Kazuhiko T., Kobayashi Y., Kuno H., Diehl E. Photrace: A blockchain-based traceability system for photographs on the internet. 2021 IEEE International Conference on Blockchain (Blockchain), IEEE (2021, December), pp. 590-596.

9. Ikotun A. M., Ezugwu A. E., Abualigah L., Abuhaija B. and Heming J., K-means clustering algorithms: A comprehensive review, variants analysis, and

advances in the era of big data, Information Sciences, vol. 622, pp. 178–210, Apr 2023. doi: 10.1016/j.ins.2022.11.139.

10. Jagannath S. and Dhaked U., A k-means-galactic swarm optimization-based clustering algorithm with otsu's entropy for brain tumor detection, Applied Artificial Intelligence, vol. 33, no. 2, (2018, October). doi: 10.1080/08839514.2018.1530869.

11. Jung, D.-H.; Kim, H.S.; Jhin, C.; Kim, H.-J.; Park, S.H. Time-Serial Analysis of Deep Neural Network Models for Prediction of Climatic Conditions inside a Greenhouse. Comput. Electron. Agric. 2020, 173, 105402.

12. Kim T., Transparent-background, Online; accessed 20-May-2024, 2024. [Online]. Available: https://pypi.org/project/transparent-background/

13. Kirillov A., He K., Girshick R., Rother C. and Dollar P., Panoptic segmentation, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 9396–9405. doi: 10.1109/CVPR.2019.00963.

14. Maggiolino G., Ahmad A., Cao J. and Kitani K., Deep oc-sort: Multi-pedestrian tracking by adaptive re-identification, Feb 2023. arXiv: 2302.11813

15. Meta, Dinov2 : Meta AI. URL: https://dinov2.metademolab.com/demos?category=segmentation

16. Mitrea C.A., Lee C.K.M., Wu Z. A comparison between neural networks and traditional forecasting methods: a case study. Int. J. Eng. Bus. Manag., 1 (2009), pp. 19-24, 10.5772/6777.

17. Nemali, K. History of Controlled Environment Horticulture: Greenhouses. HortScience 2022, 57, 239–246.

18. Omid, M. A Computer-Based Monitoring System to Maintain Optimum Air Temperature and Relative Humidity in Greenhouses. Int. J. Agric. Biol. 2004, 6, 869–873.

19. Oussama G., Rami A., Tarek F., Alanazi A.S., Abid M. Fast and intelligent irrigation system based on WSN. Comput. Intell. Neurosci. 2022.

20. Research, Bracot dataset. URL: https://universe.roboflow.com/research-aqjf1/bracot

21. Roboflow datasets. URL: https://docs.ultralytics.com/ru/yolov5/tutorials/roboflow_datasets_integration/

22. Roboflow. URL: https : / / universe . roboflow . com / yolo - 0qx2l / leaf - e69nw

23. Sama N., David E., Rossetti S., Antona A., Franchetti B. and Pirri F., A new large dataset and a transfer learning methodology for plant phenotyping in vertical farms, in Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops, Oct. 2023, pp. 540-551.

24. School, Orchid_side dataset, https://universe.roboflow.com/school-x2usm/orchid_ side, Accessed: May 20, 2024, Dec. 2023.

25. Suhardiyanto, H.; Hasbullah, R. Supriyanto Artificial Neural Network Models to Estimate Growth of Melon (Cucumis melo L.) at Vegetative Phase in Greenhouse with Evaporative Cooling. IOP Conf. Ser. Earth Environ. Sci. 2022, 1038, 012011.

26. Sun, M.; Zhang, D.; Liu, L.; Wang, Z. How to Predict the Sugariness and Hardness of Melons: A near-Infrared Hyperspectral Imaging Method. Food Chem. 2017, 218, 413–421.

27. Tabachnick, B.G.; Fidell, L.S.; Ullman, J.B. Using Multivariate Statistics, 7th ed.; Pearson: New York, NY, USA, 2019; ISBN 978-0-13-479054-1.

28. Taki, M.; Abdanan Mehdizadeh, S.; Rohani, A.; Rahnama, M.; Rahmati-Joneidabad, M. Applied Machine Learning in Greenhouse Simulation; New Application and Analysis. Inf. Process. Agric. 2018, 5, 253–268.

29. Ultralytics. URL: https://www.ultralytics.com/

30. Wang L., Huang B., Zhao Z.et al., Videomae v2: Scaling video masked autoencoders with dual masking, Mar 2023. arXiv: 2303.16727.

31. Wang X. and Liu J., Vegetable disease detection using an improved yolov8 algorithm in the greenhouse plant environment, Scientific Reports, vol. 14, no. 1, p. 4261, Feb. 2024. doi: 10.1038/s41598-024-54540-9.

32. Widi I., Köppen M. Blockchain simulation environment on multi-image encryption for smart farming application. Advances in Intelligent Networking and

Collaborative Systems: The 13th International Conference on Intelligent Networking and Collaborative Systems (INCoS-2021) 13, Springer International Publishing (2022), pp. 316-326.

33. Введення в машинне навчання за допомогою Python и Scikit-Learn. URL: https://habr.com/ua/company/mlclass/blog/247751/ (дата звернення: 30.09.2024).

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

35. Класифікація в Python з Scikit-Learn та Pandas. URL: https://stackabuse.com/classification-in-pythonwith-scikit-learn-and-pandas/ (дата звернення: 21.07.2024).

36. Коваль Н.Я., Кондисюк І.В., Тригуба А.М. Алгоритм навчання нейронної мережі для планування часу виконання робіт у гібридних проєктах. Молодь у світі сучасних технологій за тематикою: Сучасні інформаційні технології: стан та перспективи розвитку : матеріали міжнар. наук.-практ. конф. (4 червня 2021 р., м. Херсон) / за заг. ред. Г.О. Райко. – Херсон: Видавництво ФОП Вишерський В. С., 2021. – С. 153-156.

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

38. Огляд методів класифікації у машинному навчанні за допомогою Scikit-Learn. URL: https://tproger. ru/translations/scikit-learn-in-python/s://stackabuse.com/classification-in-python-with-scikit-learn-and-pandas/ (дата звернення: 15.09.2024).

39. Tryhuba A., Kondysiuk I., Tryhuba I., Koval N., Boiarchuk O., Tatomyr A. Intellectual information system for formation of portfolio projects of motor transport enterprises, in: I Workshop Information Technologies in Energy and Agro-industrial Complex, ITEA-WS 2021, CEUR Workshop Proceedings 3109, Dubliany, Lviv region, 2021, pp. 44–52.

40. Тригуба А.М., Кондисюк І., Коваль Н. Алгоритм прийняття управлінських рішень в умовах невизначеності із використанням машинного

навчання. Вчені Львівського національного аграрного університету виробництву: каталог інноваційних розробок за заг. ред. В. В. Снітинського, І. Б. Яціва. Вип. 20. Львів: Львів. нац. аграр. ун-т, 2020. С. 39.

41. Тригуба А.М., Кондисюк І.В., Татор А.В., Шолудько Я.В., Боярчук О.В. Інтелектуальна інформаційна система формування портфелів проєктів автотранспортних підприємств. Інформаційні технології в енергетиці та агропрословому комплексі: матеріали X-ї міжнародної наукової конференції, присвяченої 165-річчю університету. Львів-Дубляни, 2021, С. 113–115.

Додатки

Додаток А

Фрагмент коду налаштування DINOv2Custom

```
import os
import random
import numpy as np
import torch
from torch.utils.data import Dataset, DataLoader
from torch.optim import AdamW
from torchvision.transforms import Compose, Resize, ToTensor, Normalize
from tqdm.auto import tqdm
from PIL import Image
import matplotlib.pyplot as plt
from transformers import Dinov2Model, Dinov2PreTrainedModel
from transformers.modeling_outputs import SemanticSegmenterOutput
import evaluate
from torchvision.transforms.functional import to_tensor
from mpl_toolkits.axes_grid1 import ImageGrid
from matplotlib.gridspec import GridSpec

def save_checkpoint(model, optimizer, scheduler, epoch, file_path="C:\\Users\\Stell\\Desktop\\DVA
309\\checkpointDINO3.pth"):
    """Save model and optimizer states to a file."""
    torch.save({
        'epoch': epoch,
        'model_state_dict': model.state_dict(),
        'optimizer_state_dict': optimizer.state_dict(),
        'scheduler_state_dict': scheduler.state_dict()
    }, file_path)
    print(f"Checkpoint saved at epoch {epoch}.")

def load_checkpoint(file_path, model, optimizer, scheduler):
    """Load model and optimizer states from a file."""
    checkpoint = torch.load(file_path)
    model.load_state_dict(checkpoint['model_state_dict'])
    optimizer.load_state_dict(checkpoint['optimizer_state_dict'])
    scheduler.load_state_dict(checkpoint['scheduler_state_dict'])
    epoch = checkpoint['epoch']
    print(f"Loaded checkpoint from epoch {epoch}.")
    return epoch

iou_metric = evaluate.load("mean_iou")
def mean_iou(preds, labels, num_classes):
    iou_list = []
    for cls in range(num_classes):
        pred_inds = (preds == cls)
        target_inds = (labels == cls)
        intersection = (pred_inds & target_inds).sum()  # Fixed intersection calculation
        union = pred_inds.sum() + target_inds.sum() - intersection
        if union == 0:
            iou = 0
        else:
            iou = float(intersection) / float(union)
        iou_list.append(iou)
    return np.mean(iou_list)

# Constants for normalization
ADE_MEAN = np.array([123.675, 116.280, 103.530]) / 255
ADE_STD = np.array([58.395, 57.120, 57.375]) / 255

# Image transformations
```

```
image_transform = Compose([
    Resize((448, 448)),  # Resize all images to 448x448
    ToTensor(),
    Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])

# Mask transformations
mask_transform = Compose([
    Resize((448, 448), interpolation=Image.NEAREST),  # Use nearest neighbor interpolation
    ToTensor()
])

class PlantDataset(Dataset):
    def __init__(self, samples, image_transform=None, mask_transform=None):
        self.samples = samples
        self.image_transform = image_transform
        self.mask_transform = mask_transform

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

    def __getitem__(self, idx):
        img_path, mask_path = self.samples[idx]
        image = Image.open(img_path).convert("RGB")
        mask = Image.open(mask_path)


        if self.image_transform:
            image = self.image_transform(image)

        # Apply transformations to the mask
        if self.mask_transform:
            mask = self.mask_transform(mask)


        mask = mask.long() if not isinstance(mask, torch.LongTensor) else mask

        return image, mask

def load_datasets(root_dir, image_transform, mask_transform, split_ratio=0.8):
    image_dir = os.path.join(root_dir, 'images')
    mask_dir = os.path.join(root_dir, 'masks')

    # List all images in the image directory
    image_files = [f for f in os.listdir(image_dir) if f.endswith('.png')]
    samples = []

    for image_file in image_files:
        img_path = os.path.join(image_dir, image_file)
        mask_path = os.path.join(mask_dir, image_file)  # Use the same filename for the mask

        if os.path.exists(mask_path):  # Check if the mask file exists
            samples.append((img_path, mask_path))
        else:
            print(f"No corresponding mask found for image {image_file}")

    # Shuffle samples to ensure random distribution for training and validation
    split_point = int(len(samples) * split_ratio)
    train_samples = samples[:split_point]
    val_samples = samples[split_point:]

    # Create dataset instances with the correct arguments
    train_dataset = PlantDataset(train_samples, image_transform=image_transform, mask_transform=mask_transform)
```

```
    val_dataset = PlantDataset(val_samples, image_transform=image_transform, mask_transform=mask_transform)

    return train_dataset, val_dataset

# Load and split datasets
root_dir = 'C:\\Users\\Stell\\Desktop\\DVA 309\\Plant segmentation'
train_dataset, val_dataset = load_datasets(root_dir, image_transform, mask_transform)
train_dataloader = DataLoader(train_dataset, batch_size=6, shuffle=True)
val_dataloader = DataLoader(val_dataset, batch_size=6, shuffle=False)

def check_data_samples(dataloader):
    images, masks = next(iter(dataloader))
    figure, ax = plt.subplots(nrows=2, ncols=5, figsize=(15, 6))
    for i in range(5):
        image = images[i].cpu().numpy().transpose((1, 2, 0))
        image = (image - image.min()) / (image.max() - image.min())
        ax[0, i].imshow(image)
        ax[0, i].axis('off')
        # Make sure mask is 2D
        mask = masks[i].cpu().squeeze()
        if mask.ndim > 2:
            mask = mask[0]  # Take the first channel if mask has an extra dimension
        ax[1, i].imshow(mask, cmap='gray')
        ax[1, i].axis('off')
    plt.tight_layout()
    plt.show()



check_data_samples(train_dataloader)

def check_mask_values(dataloader):
    images, masks = next(iter(dataloader))
    unique_values = [torch.unique(mask).tolist() for mask in masks]
    print(f"Unique mask values: {unique_values}")

check_mask_values(train_dataloader)

def inspect_single_mask(dataloader):
    images, masks = next(iter(dataloader))
    mask = masks[0]  # Get the first mask
    plt.figure(figsize=(6, 6))
    plt.imshow(mask, cmap='gray')
    plt.colorbar()
    plt.title('Single Mask Inspection')
    plt.show()

    # Print out unique values and their counts
    unique, counts = torch.unique(mask, return_counts=True)
    print(f"Unique values in the mask: {unique}")
    print(f"Counts for each value: {counts}")

inspect_single_mask(train_dataloader)


# Model setup
class Dinov2ForSemanticSegmentation(Dinov2PreTrainedModel):
    def __init__(self, config):
        super().__init__(config)
        self.dinov2 = Dinov2Model(config)
        self.classifier = torch.nn.Conv2d(config.hidden_size, 2, kernel_size=2)

    def forward(self, pixel_values, labels=None):
```

```
        outputs = self.dinov2(pixel_values, return_dict=True)
        features = outputs.last_hidden_state[:, 1:, :]
        features = features.permute(0, 2, 1).view(features.shape[0], -1, 32, 32)
        logits = self.classifier(features)
        logits = torch.nn.functional.interpolate(logits, size=pixel_values.shape[-2:], mode='bilinear', align_corners=False)

        loss = None
        if labels is not None:
            labels = labels.squeeze(1)
            loss_fct = torch.nn.CrossEntropyLoss()
            loss = loss_fct(logits, labels)

        return SemanticSegmenterOutput(loss=loss, logits=logits)

model = Dinov2ForSemanticSegmentation.from_pretrained("facebook/dinov2-large")
model.to(torch.device("cuda" if torch.cuda.is_available() else "cpu"))
optimizer = AdamW(model.parameters(), lr=1e-4)
scheduler = torch.optim.lr_scheduler.StepLR(optimizer, step_size=3, gamma=0.1)

train_losses = []
val_losses = []

# Training and validation loops
num_epochs = 10
for epoch in range(num_epochs):
    model.train()
    total_train_loss = 0
    total_loss = 0
    num_batches = 0
    for images, masks in tqdm(train_dataloader):
        images, masks = images.to(model.device), masks.to(model.device)
        optimizer.zero_grad()
        outputs = model(images, labels=masks)
        loss = outputs.loss
        loss.backward()
        optimizer.step()
        total_loss += loss.item()
        total_train_loss += loss.item()
        num_batches += 1

    avg_train_loss = total_train_loss / num_batches
    train_losses.append(avg_train_loss)


    preds = outputs.logits.argmax(dim=1)
    iou_metric.add_batch(predictions=preds, references=masks)

    num_classes = 2
    ignore_index = -1
    train_iou = iou_metric.compute(num_labels=num_classes, ignore_index=ignore_index)

    scheduler.step()
    print(f'Epoch {epoch+1}/{num_epochs}, Train Loss: {total_loss / len(train_dataloader)}, Train IoU:
{train_iou["mean_iou"]}')

    save_checkpoint(model, optimizer, scheduler, epoch, file_path=f"checkpoint_oterdata_epoch_{epoch}.pth")

    # Validation phase
    model.eval()
    total_val_loss = 0
    num_batches = 0
    val_loss = 0
    iou_metric = evaluate.load("mean_iou", num_labels=num_classes, ignore_index=ignore_index)
```

```
    with torch.no_grad():
        for images, masks in val_dataloader:
            images, masks = images.to(model.device), masks.to(model.device)
            outputs = model(images, labels=masks)
            val_loss += outputs.loss.item()
            total_val_loss += val_loss
            num_batches += 1
            # Update IoU metric
            preds = outputs.logits.argmax(dim=1)
            iou_metric.add_batch(predictions=preds, references=masks)

        avg_val_loss = total_val_loss / num_batches
        val_losses.append(avg_val_loss)

        print(f'Epoch {epoch+1}/{num_epochs}, Train Loss: {avg_train_loss}, Validation Loss: {avg_val_loss}')

    val_iou = iou_metric.compute(num_labels = 2, ignore_index = -1)
    print(f'Validation Loss: {val_loss / len(val_dataloader)}, Validation IoU: {val_iou["mean_iou"]}')

def plot_loss_curve(train_losses, val_losses):
    epochs = range(1, len(train_losses) + 1)
    plt.figure(figsize=(10, 5))
    plt.plot(epochs, train_losses, 'bo-', label='Training Loss')
    plt.plot(epochs, val_losses, 'ro-', label='Validation Loss')
    plt.title('Training and Validation Loss')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend()
    plt.grid(True)
    plt.show()


plot_loss_curve(train_losses, val_losses)


def visualize_batch(dataloader, num_images=5):

    images, masks = next(iter(dataloader))


    images_np = images.numpy().transpose((0, 2, 3, 1))
    masks_np = masks.numpy()


    if masks_np.ndim == 3:  # Shape is (batch_size, height, width)
        masks_np = masks_np[:, :, :, None]

    # Normalize image for display
    images_np = (images_np - images_np.min()) / (images_np.max() - images_np.min())


    fig = plt.figure(figsize=(15, 10))
    gs = GridSpec(3, 1, figure=fig)


    ax1 = fig.add_subplot(gs[0, 0])
    ax1.imshow(np.hstack(images_np))
    ax1.set_title('Images', fontsize=15)
    ax1.axis('off')


    ax2 = fig.add_subplot(gs[1, 0])
    ax2.imshow(np.hstack(masks_np.squeeze()), cmap='gray')
```

```
    ax2.set_title('Masks', fontsize=15)
    ax2.axis('off')


    ax3 = fig.add_subplot(gs[2, 0])
    for i in range(num_images):
        ax3.imshow(images_np[i])
        ax3.imshow(masks_np[i].squeeze(), alpha=0.4, cmap='jet')  # Adjust alpha for mask transparency
    ax3.set_title('Overlay', fontsize=15)
    ax3.axis('off')

    plt.show()

visualize_batch(train_dataloader, num_images=5)
```