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

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

НА ТЕМУ

Програмний модуль класифікації комах-шкідників сільськогосподарських культур

Галузь знань **12 «Інформаційні технології»**

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

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

Освітній рівень: бакалавр

Виконала: студентка 4 курсу, групи КН-41
Бромот Д.І.
_____
(прізвище та ініціали)

Керівник: Іларіонов О.Є.
_____
(прізвище та ініціали)
к.т.н., доцент
_____
(науковий ступінь, звання)

Випускна кваліфікаційна робота бакалавра допущена до захисту
рішенням кафедри *інтелектуальних технологій*
Протокол №11 від 06.06.2022 р.
зав. кафедри_____ доц. Іларіонов О.Є.

Київ – 2022

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра інтелектуальних технологій

Спеціальність 122 «Комп'ютерні науки»

ЗАТВЕРДЖУЮ

Завідувач кафедри
інтелектуальних технологій

Іларіонов О.Є.

_____

"___"_____2022 р.

ЗАВДАННЯ
НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Бромот Дар'ї

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи)

Програмний модуль класифікації комах-шкідників сільськогосподарських культур

затверджена протоколом засідання кафедри від «23» грудня 2021 р. №4

2. Термін здачі студентом закінченого проекту (роботи) 31 травня 2022 року

3. Вихідні дані до проекту (роботи)

Програмний модуль класифікації комах-шкідників, який складається веб-сайту, нейронної мережі та хмарного сервісу

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

4.1 Аналіз технології моніторингу комах-шкідників для запобігання пошкодження посівів культур
4.2 Розробка архітектури модулю класифікації та виявлення комах-шкідників
4.3 Технологічні особливості реалізації програмного модуля

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

5.1 Вступ, тема, мета 1 - 2 слайди
5.2 Аналіз технології моніторингу комах-шкідників 3 - 7 слайди
5.3 Розробка архітектури модулю класифікації та виявлення комах-шкідників 8-11 слайд
5.4 Технологічні особливості реалізації програмного модуля 12-16 слайд
5.5 Висновки 16-17 слайд

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

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1			
2			
3			

7. Дата видачі завдання 15 лютого 2022 року

Керівник _____ / О.Є. Іларіонов /
(підпис) (ПІБ)

Завдання прийняв до виконання _____ / Д.І. Бромот /
(підпис) (ПІБ)

КАЛЕНДАРНИЙ ПЛАН

Пор. №	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів випускної кваліфікаційної роботи	Примітка
1.	Обговорення з керівником постановки завдання та змісту пояснювальної записки	25.01.2022 - 02.02.2022	
2.	Аналіз постановки задачі, формалізація задачі, вибір методів та засобів реалізації поставленої задачі, аналіз літературних джерел	03.02.2022 - 23.02.2022	
3.	Проектування програмного модулю класифікації комах-шкідників на жовтій липкій стрічці	24.03.22 - 18.04.22	
4.	Розробка та тестування програмного модулю класифікації шкідників сільськогосподарських культур	19.04.22 - 10.05.22	
5.	Оформлення пояснювальної записки, підготовка презентації	11.05.2022 - 31.05.2022	

Студент-дипломник _____ / Д.І. Бромот /
(підпис) (ПІБ)

Керівник випускної кваліфікаційної роботи _____ / О.Є. Іларіонов /
(підпис) (ПІБ)

Зміст

РЕФЕРАТ

Кваліфікаційна робота складається зі вступу, 3 розділів, 18 підрозділів, висновків, списку використаних джерел (26 джерел) та 1 додатку. Загальний обсяг роботи 70 сторінки. Робота містить 7 таблиць та 38 рисунків.

Актуальність теми. Комахи-шкідники є однією з основних причин, що впливають на врожайність та якість сільськогосподарських культур у всьому світі. Швидкий і надійний моніторинг комах-шкідників відіграє вирішальну роль у прогнозуванні популяції та заходах боротьби. Великий прорив технології глибинного навчання (ГН) привів до її успішного застосування в різних областях, включаючи автоматичний моніторинг комах-шкідників (МКШ).

Метою кваліфікаційної роботи розробити програмний модуль, який допоможе агрономам, екологам та фермерам здійснювати щоденний моніторинг кількість комах-шкідників для запобігання пошкодження посівів сільськогосподарських культур.

Об'єктом дослідження дипломної роботи є запобігання пошкодження посівів культур від комах-шкідників.

Предметом дослідження є моніторинг комах-шкідників, що потрапити на досліджувану липку стрічку в пасці.

Результати роботи. У результаті виконання дипломної роботи було розроблено програмний модуль для розпізнавання та класифікації комах-шкідників на жовтій липкій стрічці для запобігання пошкодження сільськогосподарських культур. Відповідно до поставленої мети, модуль виконує моніторинг комах-шкідників: оброблює зображення жовтої липкої стрічки у пасці, розпізнає та класифікує комах на різні класи, підраховує загальну кількість. Модуль викладений на платформу AWS для зовнішнього використання. Точність навченої нейронної мережі становить 94%.

Ключові слова: програмний модуль для розпізнавання комах-шкідників, жовта липка стрічка в пасці, глибинне навчання, хмарні платформи, моніторинг.

ABSTRACT

Qualification work consists of an introduction, 3 chapters, 18 subsections, conclusions, a list of sources used (26 sources) and 1 application. The total volume of the work is 70 pages. The work contains 7 tables and 38 figures.

Actuality of theme. Insect pests are one of the main reasons affecting crop yields and quality around the world. Rapid and reliable monitoring of insect pests plays a crucial role in population forecasting and control measures. The great breakthrough of deep learning technology (DE) has led to its successful application in various fields, including automatic pest monitoring (ISS).

The aim of the qualification work is to develop a software module that will help agronomists, ecologists and farmers to monitor the number of pests on a daily basis to prevent damage to crops.

The object of the thesis research is to prevent damage to crops by pests.

The subject of the study is the monitoring of insect pests that get on the studied adhesive tape in the paste.

Results of work. As a result of the thesis, a software module was developed for the identification and classification of pests on yellow sticky tape to prevent damage to crops. In accordance with the set goal, the module monitors pests: processes the image of yellow sticky tape in the trap, recognizes and classifies insects into different classes, counts the total number. The module is installed on the AWS platform for outdoor use. The accuracy of the trained neural network is 94%.

Keywords: software module for pest detection, yellow sticky tape in the paste, in-depth training, cloud platforms, monitoring.

ВСТУП

Комахи-шкідники є однією з основних причин, що впливають на врожайність та якість сільськогосподарських культур у всьому світі. Швидкий і надійний моніторинг комах-шкідників відіграє вирішальну роль у прогнозуванні популяції та заходах боротьби. Великий прорив технології глибинного навчання (ГН) привів до її успішного застосування в різних областях, включаючи автоматичний моніторинг комах-шкідників (МКШ). ГН створює як нові перспективи, так і низку проблем для обробки даних у розумному моніторингу шкідників. У цьому огляді викладено технічні методи фреймворків ГН та їх застосування в сфері моніторингу з акцентом на класифікацію комах-шкідників і виявлення їх за допомогою польових зображень.

Методології та технічні деталі, розроблені при класифікації комах-шкідників і виявленні за допомогою ГН, підсумовуються та проганяються на різних етапах обробки: отримання зображення, методи попередньої обробки та моделювання даних. Нарешті, надається загальна структура для полегшення розумного моніторингу комах, а також висвітлюються майбутні проблеми та тенденції.

Мета роботи — надати дослідникам і науково-технічним співробітникам краще розуміння технік ГН та їх найсучасніших досягнень у сфері моніторингу комах-шкідників. Полегшення щоденної роботи технологів сільськогосподарського виробництва та екологів завдяки впровадженню додатку для МКШ.

РОЗДІЛ 1. АНАЛІЗ ТЕХНОЛОГІЇ МОНІТОРИНГУ КОМАХ-ШКІДНИКІВ ДЛЯ ЗАПОБІГАННЯ ПОШКОДЖЕННЯ ПОСІВІВ КУЛЬТУР

1.1. Аналіз області застосування моніторингу комах-шкідників

Шкідники є однією з основних причин збитків у сільському господарстві. Комахи можуть бути особливо шкідливими, оскільки вони можуть харчуватися листям, впливаючи на фотосинтез, а також є переносниками ряду

серйозних захворювань. Існує багато хімічних і біологічних методів боротьби з шкідниками, що розроблювалися з метою скорочення загального використання пестицидів та зосередження на більш точному застосуванні. Але для досягнення їх максимальної ефективності зазвичай рекомендується ретельний моніторинг за всіма властивостями.

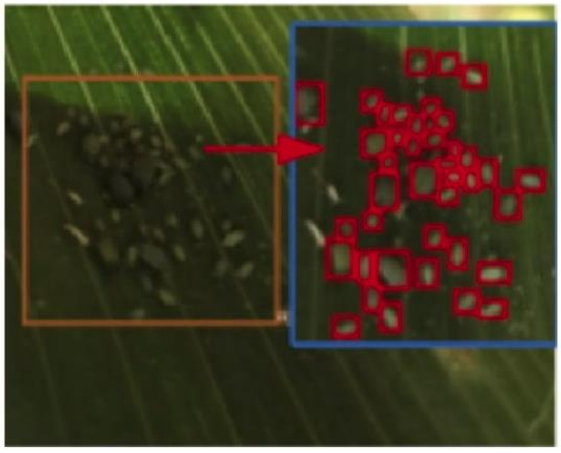
У багатьох випадках моніторинг здійснюється пасивно працівниками, коли вони виконують свою повсякденну роботу. Проблема цього методу полягає в тому, що коли зараження виявлено, може бути достатньо пізно і вже було завдано незворотної шкоди. Раннє виявлення шкідників вимагає більш системного підходу, особливо у великих господарствах. Пастки є найбільш поширеним інструментом для систематичного моніторингу шкідників. При правильному застосуванні цей тип пристрою успішно досліджує популяції комах на всій території. Однак без певної автоматизації пастки все одно потрібно розміщувати та збирати вручну, а оцінку зараження необхідно виконувати візуально, вводячи ступінь суб'єктивності, який може призвести до упередженої оцінки ситуації. Таким чином, незалежно від використання пасток, існує потреба в методах, здатних швидко, точно та автономно оцінити стан шкідників, щоб розробити відповідну стратегію боротьби з ними.

1.2. Аналітичний огляд методів виявлення та класифікації комах-шкідників

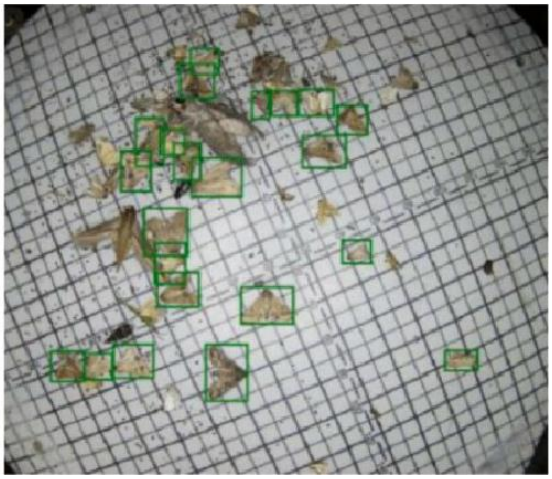
1.2.1. Методи виявлення шкідників

Виявлення комах за допомогою цифрових зображень є ще одним важливим аспектом МКШ, який включає локалізацію та класифікацію комах. Раннє виявлення комах-шкідників у режимі реального часу та оцінка популяції можуть підвищити точність розпилення пестицидами та ефективно зменшити економічні та екологічні втрати. Загалом, застосування виявлення комах у сільському господарстві та лісі можна розділити на три категорії різні сценарії: виявлення комах на пастці з липким папером (рис. 1.1а), виявлення комах на рослинах (рис. 1.1б) та виявлення комах на опорній плиті в пастці (рис. 1.1в).

а)



б)



в)

Рисунок 1.1 – Різні способи виявлення комах-шкідників: а) на пастці липкого паперу; б) на рослинах; в) на опорній плиті в пастці

1.2.1.1. Виявлення шкідників на липкій пастці

Метою використання пасток для виявлення комах є прогнозування чисельності комах-шкідників, щоб можна було вчасно вжити заходів боротьби. Липка пастка є ключовим компонентом для моніторингу комах-шкідників у різних галузях (город, теплиця, сад, тощо). Однак комахи, що потрапили на липкий папір, часто є мішенями невеликого розміру. Крім того, відбиття світла від липкого паперу, переміщення пастки та загнивання або пошкодження комах у полі створюють великі проблеми для виявлення комах на липких зображеннях пастки.

Щоб подолати ці перешкоди, багато дослідників намагалися дослідити методи автоматичного виявлення дрібних шкідників на зображеннях липких

пасток[1][2][3][4] за допомогою комп’ютерного зору та машинного навчання. З точки зору розміру даних, кількість оригінальних зображень для виявлення шкідників на зображеннях липких пасток відносно невелика, але кількість анотованих цілей велика, а оригінальні зображення зазвичай розділити на невеликі фрагменти зображення для точного виявлення дрібних шкідників .

1.2.1.2. Виявлення комах на рослинах

Фото зображене на рисунку 1.1б означає безпосереднє виявлення комах на листках рослин, стовбурі чи інших відкритих польових ділянках. У підсумку, виявлення комах на рослинах включає 3 перешкоди:

1. Виявлення комах на фоні зовнішнього середовища.

Чэнцзюнь Се, Жуцзин Ван, Цзе Чжан, Пэн Чэньа, Жуй Ля, Тяньцзяо Чэньа та Хунбо Чэнь оцінили різноманітні моделі виявлення олійного ріпаку на основі глибоких згорткових нейронних мереж у складних середовищах. Було запропонувано використати двошаровий алгоритм виявлення, заснований на технології глибокого навчання для виявлення ріпаку, і результати показали, що він перевершував одношаровий алгорит за середньою швидкістю запам'ятовування [5].

2. Щільне розміщення комах.

Вирішення проблеми запропоноване в публікації [6] описує систему на основі глибокого навчання для виявлення та підрахунку пшеничних кліщів з точністю 88,5%. Аналогічно, оскільки попелиця в польових умовах є крихітною і часто має щільне розповсюдження, було запропоновано двоступеневий детектор під назвою Coarse-to-Fine Network (CFN) для виявлення шкідника на польових зображеннях з середньою точністю 76,8% [7].

3. Вивчення методів виявлення комах у відеопотоці.

Для побудови системи відеовиявлення шкідників рослин була запропонована архітектура відеовиявлення на основі DL зі спеціальною

магістраллю для виявлення шкідників рослин на відео, що більше підходить для виявлення непідготовлених відео з рису [8].

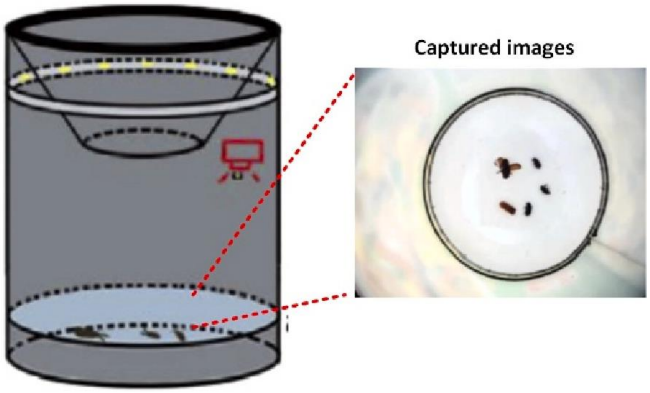
Хоча мережі виявлення об’єктів на основі CNN показали нормальну точність у загальних програмах виявлення об'єктів, їх застосування у великомасштабних багатокласових наборах даних про шкідників у природі все ще страждає від деяких обмежень:

- велика різниця розподілу щільності та розмірів крихітних шкідників призводить до ситуаціїї, що деякі об'єкти залишаються непомічені мережами ГН.
- розмір вибірки різних видів комах-шкідників незбалансований, що ускладнює підходи глибокого навчання для досягнення високої точності для всіх видів комах-шкідників.

1.2.1.3. Виявлення комах на опорній плиті в пастці

Пастки на основі світла (рис. 1.2а) або феромонів (рис. 1.2б) були вивчені та розроблені для моніторингу наявності та чисельності шкідників за останні кілька років [9]. У цьому методі відлову, чутливі до певного спектру або хімічних феромонів, комахи притягуються та захоплюються. Зразок зображень виявлення в цій програмі наведено в рисунку 1.1в.

а)



б)

Рисунок 1.2 – Пастки для автоматичного виявлення комах на основі комп'ютерного зору: а) світлова [9]; б) на основі феромонів [9]

При застосуванні світлових пасток комахи різних категорій потрапляють на базову пластину завдяки фототаксису (властивість клітин рухатися до або від джерела світла), що створює більше проблем для точного виявлення кількох комах порівняно з застосуванням пастки на основі феромонів. Камера, встановлена в пасці, робить фотокартки, наприклад, рисової п'ятиці, яку приваблює феромон. Для виявлення даного шкідника було запропоновано метод ГН.

Експерименти показали, що середня точність становила 84%, а середня тривалість виконання — 0.4с і 23.4с на двох різних вбудованих платформах, що продемонструвало практичний метод раннього попередження розмноження шкідників. Щоб забезпечити великий ресурс даних для навчання моделей ГН для виявлення шкідників, було створено великомасштабний багатоцільовий стандартизований набір даних про шкідників сільського господарства під назвою Pest24, який складається з 25 378 анотованих зображень і включає 24 категорії шкідників [10].

1.2.2. Методи класифікації шкідників

Класифікація комах передбачає виконання різних етапів. Потік кроків для класифікації комах проілюстровано на рисунку 1.3 Збільшення зображення застосовується до зображень набору даних комах, щоб розширити навчальний набір даних. Потім для класифікації класів комах застосовуються елементи

форми, витягнуті із зображень комах, і алгоритми машинного навчання ANN, SVM, KNN і NB. Класифікація комах на основі згорткових нейронних мереж (ЗНМ), адаптована для порівняння продуктивності. ЗНМ — це клас багатошарових нейронних мереж, розроблених для розпізнавання візуальних шаблонів безпосередньо з піксельних зображень з мінімальною попередньою обробкою.

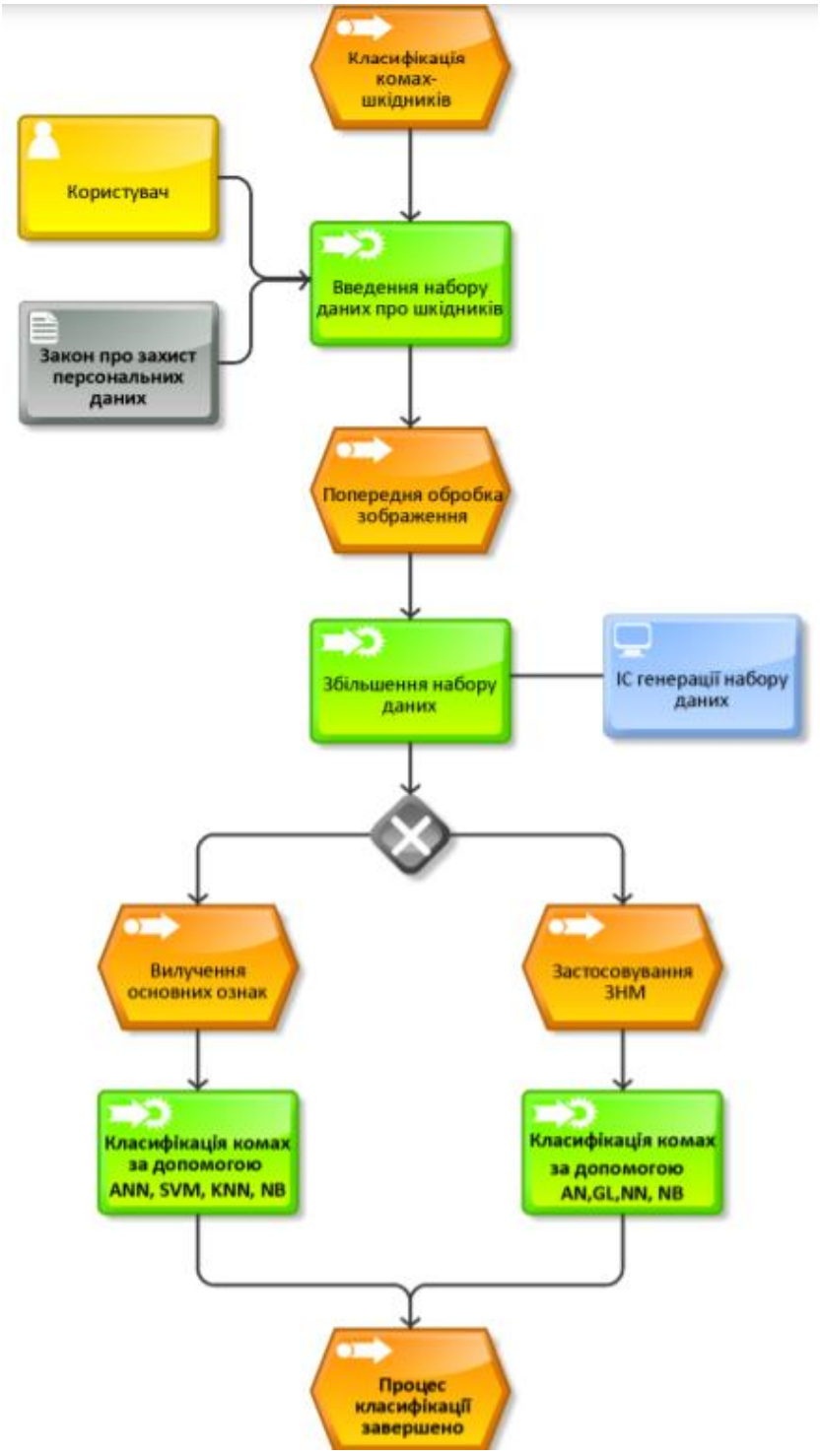


Рисунок 1.3 – Діаграма методів класифікації комах

ЗНМ складається з багатьох об’єднаних шарів. Згорткові шари виконують зважені згортки між своїми вхідними параметрами та їх вагами, які можна вивчати. Таким чином, вони знаходять локальні закономірності у вхідних даних. Шари об’єднання – це шари, які не підлягають навчанню, які зменшують розмірність свого введення шляхом локального відображення невеликого квадрата на вході в одне число. Класифікатор зазвичай складається з одного або кількох повністю пов’язаних шарів і функції softmax. Порівняємо основні типи ЗНМ:

– *AlexNet* був розроблений у 2012 році, і йому вдалося перевершити всі попередні алгоритми в класифікації ImageNet. Він містить вісім доступних для навчання шарів: перші п’ять є згортковими, а решта три повнозв’язними. За кожним шаром, який можна вивчати, слідує випрямлений лінійний блок (ReLU) як функція активації. Серед згорткових шарів використовуються шари максимального об’єднання, щоб зменшити розмірність прихованих шарів. Вихід останнього повністю підключеного шару подається у функцію softmax, яка створює розподіл ймовірностей за можливими вихідними класами.

– *GoogleNet* складається з 22 шарів CNN. Він має лише 4 мільйони параметрів, що є дуже низьким числом у порівнянні з 60 мільйонами параметрів AlexNet. Основною особливістю GoogleNet є використання початкових шарів. Вони виготовлені з різних згорткових шарів різного розміру, виходи яких об’єднані на кінці модуля. Ідея полягає в тому, що фільтри різного розміру можуть виявляти різні шаблони вхідних даних.

– *ShuffleNet* — це дуже легка мережа, яка використовує згруповані згортки та перемішування каналів, щоб мати дуже низьку складність. Згрупована згортка — це згортки розміром 1×1, які розглядають лише підмножину каналів прихованого шару. Таким чином кількість множень сильно зменшується. Потім, щоб посилити зв’язок між нейронами, канали виходу перемішуються. В результаті мережа працює в 13 разів швидше, ніж AlexNet, зберігаючи подібну точність.

– *MobileNetv2* — це легка згортка мережа для мобільних додатків. Він містить кілька розділених по глибині згорток, які можна розглядати як згорткові шари, де тензор тривимірних ваг розкладається на один двовимірний тензор і одновимірний тензор, що вимагає набагато менше пам'яті. Цей тип шарів часто використовується в легких згорткових мережах. Що стосується інших мереж, MobileNetv2 має менше нелінійності. Автори дають інтерпретацію, чому це повинно дати кращі результати в їхній мережі, і стверджують, що та сама мережа працювала гірше, коли вони намагалися додати більше лінійності. Крім того, ця мережа має інвертовані пропуски підключень. Це означає, що приховані шари, з'єднані за допомогою пропускних з'єднань, мають низькі розміри, що зменшує кількість операцій, які виконує мережа.

– *DenseNet201* є великою та високопродуктивною згортковою мережею. Його назва походить від того, що кожен шар пов'язаний з усіма попередніми. Ця архітектура допомагає градієнтним потокам і заохочує повторне використання функцій. DenseNet201 є конкурентоспроможним з іншими найсучаснішими мережами ImageNet, маючи менші параметри, ніж AlexNet.

1.3. Аналіз методів та засобів підрахунку шкідників

Підрахунок захоплених комах можна виконувати вручну або за допомогою програмного забезпечення для обробки зображень. Системи моніторингу можна класифікувати як повністю автоматизовані, коли система оснащена програмним забезпеченням для інтерпретації зображень та ідентифікації видів спійманих комах та напівавтоматизовані, коли людина дистанційно може ідентифікувати та підрахувати спійманих комах, спостерігаючи за зображеннями, зробленими камерою-пасткою.

У деяких із прикладних досліджень[11], спійманих комах перевіряли вручну дистанційно, а зображення спостерігалися людським оком через комп'ютер або смартфон. Цей підхід потребує навченого спостерігача на контрольній станції, щоб правильно ідентифікувати види комах, і, незважаючи

на фактичне уникнення виїзду на місце, цей процес займає багато часу, особливо людино-годин, що підвищує вартість обслуговування такої системи. Крім того, розмір комах може бути проблемою.

Щоб полегшити роботу спостерігача, можна використовувати алгоритми обробки зображень для ідентифікації та автоматичного підрахунку комах. У деяких випадках точність автоматичного підрахунку має бути перевірена та підтверджена експертом. Наприклад, автоматичні пастки на основі комерційних камер TrapView® автоматично обробляють зображення з високою роздільною здатністю для підрахунку комах і мають в наявності ручне підтвердження як частину обслуговування клієнтів.

Розробка повністю автоматизованого підрахунку комах спочатку була заснована на датчиках руху і протягом останніх десятиліть використовувався для комах-шкідників. Насправді, використання датчиків фотопереривання (наприклад, інфрачервоних датчиків, які генерують електричний сигнал) є гарним підходом до автоматичного підрахунку сотень особин, як, наприклад, у випадку плодових мушок. Однак цей метод у деяких випадках може призвести до дуже низької точності системи моніторингу. Наприклад, автори статті [12] розробили систему, в якій метелики в момент потрапляння в пастку, спинялися просоченою інсектицидом смужкою, перш ніж впасти через воронку всередині пастки і пройти повз оптичний датчик. Однак комахи були вбиті не відразу і все ще були активними всередині пастки, кілька разів літаючи вгору-вниз через датчик і, тим самим, викликаючи переоцінку кількості.

Техніки обробки зображень і комп'ютерного зору використовуються для автоматичної ідентифікації кількох комах-шкідників, таких як ромбовидна моль або рисовий клоп. Однак, коли схожі види приваблюються до однієї і тієї ж пастки, потрібен навчений оператор, щоб визначити цільовий вид. Наприклад, «Е-пастка» McPhail привертала різні види тефритів, які були легко розрізнені людиною-оператором, але не комп'ютерним алгоритмом [13]. Також можуть виникати помилки підрахунку та інтерпретації з віддаленими

зчитуваннями знімків захоплення, оскільки вибірковість приманки та синхронне зустрічання більшої кількості видів із подібними морфологічними характеристиками можуть відігравати важливу роль у точності ідентифікації лову комах. Проте в цілому висока специфічність пастки та приманки разом із якісними знімками забезпечують хорошу точність ідентифікації та підрахунку комах-шкідників (як вручну, так і автоматично).

1.4. Аналіз існуючих рішень

Як правило, компанії з боротьби зі шкідниками використовують методи ручного контролю для моніторингу популяції комах і відповідних завдань контролю. Це трудомістке завдання і вимагає величезних витрат людський ресурси для ефективної роботи з густонаселеною міською територією та великими сільськогосподарськими фермами. Система дистанційного моніторингу – це нова техніка. Рисунок 1.4 показує схему віддаленого моніторингу пасток і виявлення комах на основі ГН, створену студентами кафедри електротехніки, Університету Західного Онтаріо (Лондон).

Система складається з рівня сприйняття, транспортного рівня, рівня обробки та прикладного рівня. Нейронна мережа навчається і працює на рівні обробки для виконання завдання віддаленої ідентифікації комах.

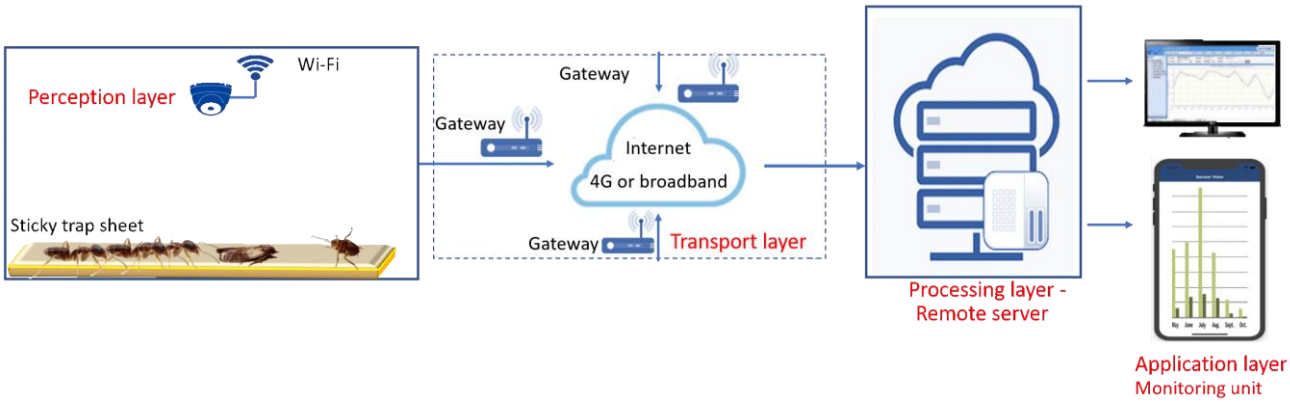


Рисунок 1.4 – Архітектура системи виявлення комах [14]

Прикладний рівень доставляє інформацію про стан пастки для комах кінцевому користувачеві. Для виконання завдання прикладного рівня

використовуються смартфони та веб-інтерфейси. У цій роботі розроблено веб-інтерфейс GUI та Android Mobile для відстеження інформації про статус. Рисунок 1.5 показує макет мобільного додатка Android, розробленого для моніторингу комах.

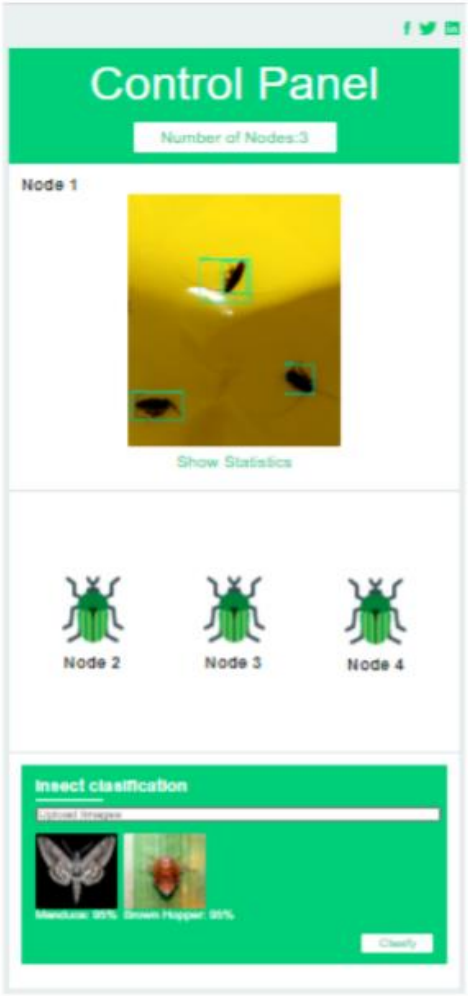


Рисунок 1.5 – Вигляд мобільного додатка для Android [14]

Для побудови системи віддаленого моніторингу комах-пасток було використано чотирьох рівневий каркас, а для автоматичного визначення класу комах використовувалася платформа виявлення об’єктів Faster RCNN ResNet. Ефективність ідентифікації комах на основі глибокого навчання була перевірена в автономному режимі та онлайн за допомогою різноманітних баз даних зображень комах, які включають базу даних комах із вбудованим середовищем та базу даних комах на фермі. На відміну від інших систем виявлення об’єктів, включаючи SSD та Yolo, запропонована схема мала кращу точність виявлення комах.

Точність ідентифікації сільськогосподарської комахи 97%, а обробка одного зображення займала в середньому 0,2с. Це дослідження показало, схема моніторингу комах на основі ГН автоматизують метод віддаленої ідентифікації комах за допомогою навченої згорткової нейронної і долають недоліки системи боротьби з комахами.

Ще одним прикладом існуючих систем є автоматизована система МКШ за допомогою мережі датчиків встановлених в навколишньому середовищі, створений в Національному тайванському університет (кафедра біомехатроніки) під керівництвом в професора Та-Те Лін. Остання версія даного пристрою моніторингу комах-шкідників показана на рисунку 1.6.

Рисунок 1.6 – Пристрій моніторингу комах-шкідників у теплиці [15]

Пристрій включає в себе датчик температури та вологості, датчик інтенсивності світла та камеру, спрямовану на липку паперову пастку. Фермери можуть встановити необхідну кількість пристроїв залежно від розміру своєї ферми. Як правило, пристрої встановлюють у місцях найбільшого скупчення комах, щоб запобігти подальшому пошкодженню врожаю. Пристрої підключаються до маршрутизатора Wi-Fi і надсилають дані про навколишнє середовище та зображення липкого паперу на віддалений

сервер. Після обробки такі дані, як поширення комах, кількість, сигнали тривоги та стан навколишнього середовища, відображаються на веб-сайті та на смартфонах користувачів. Структура системи показана на малюнку 1.7

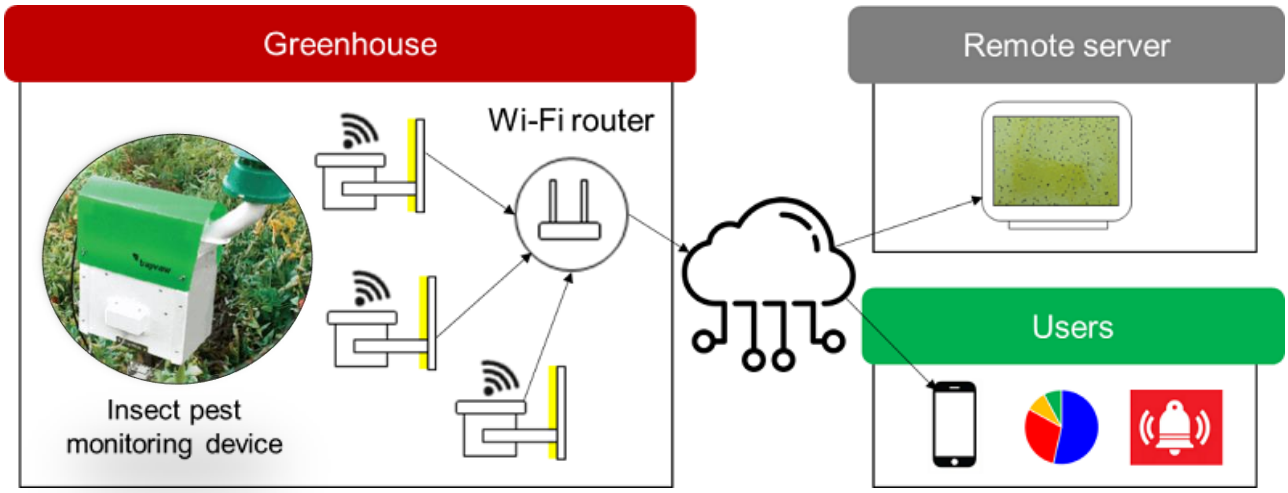


Рисунок 1.7 – Архітектура системи автоматизованого моніторингу комах-шкідників [15]

Описана система зараз використовується на кількох фермах на Тайвані. Після встановлення було отримано кілька довгострокових даних, які показують переваги наявності системи моніторингу комах-шкідників, щоб допомогти фермерам у прийнятті рішень. З довгострокових даних було виявлено, що відсоток комах-шкідників, знайдених у кожному господарстві, варіювався в залежності від сезону, ведення теплиці та багатьох інших факторів.

Крім того, систему можна використовувати для розробки більш корисних додатків. Для цього в систему моніторингу входить додаток для смартфонів під назвою NTU IPM, який надається фермерам, доступний як на Android, так і на iOS. Для користувачів системи моніторингу вони можуть перейти до вкладки WSN Tools, де вони можуть побачити дані всередині своєї ферми. Зразок скріншотів інформаційної панелі показаний на рисунку 1.8.

а) б)

Рисунок 1.8 – Скріншоти мобільного додатку NTU IPM в Android(а) та iOS(б)[15]

Програма надсилає сповіщення користувачу, які містить основну інформацію щодо кожної ферми, наприклад стан навколишнього середовища або кількість комах за поточний день. Часто вихідні дані, показані в NTU IPM, недостатньо прості, щоб їх могли використовувати фермери. Для вирішення даної проблеми кількість комах поділяється на чотири рівні: низький, середній, високий і серйозний. Це робиться шляхом обчислення відсотка збільшення комах-шкідників за певний період, коли система могла збирати дані з конкретної ферми. Наприклад, сигнали тривоги на фермі YUNLIN_GH показують низький рівень для комарів. Це означає, що кількість комарів, знайдених на фермі, є нормальною і не викликає тривоги. Однак, для мух, трипсів та білокрилок наявний помірний рівень. Це означає, що кількість комах для 3-х видів не є нормальною і вище, ніж зазвичай. Але якщо будь-яка

кількість комах була на високому рівні, то можливо, настав час, коли фермерам потрібно використати пестициди. Особливо це відбувається, коли наявний серйозний рівень, що є досить екстремальним випадком.

Результати аналізу даних та розроблена система сигналізації довели, що за допомогою системи моніторингу комах-шкідників боротьба зі шкідниками в теплицях може бути ефективнішою.

1.5. Аналіз зацікавлених сторін

1. Внутрішні зацікавлені сторони (офіційно залучені до функціонування системи моніторингу комах-шкідників) :

- Розробник СМКШ;
- Агроном;
- Еколог;
- Фермерське господарство;
- Інші сільськогосподарські організації;

2. Зовнішні зацікавлені сторони (не приймають безпосередньої участі, але можуть впливати на функціонування СМКШ):

- Контролюючі органи (Міністерство аграрної політики та продовольства України, Міністерство екології та природних ресурсів України і т.д.)
- Державні органи (Державна податкова служба України (ДПС), Державна екологічна інспекція).
- Конкурентні організації;

3. Прихильники (протагоністи) і супротивники (антагоністи):

- Протагоністи — центральні особи, які отримають вигоду від реалізації проекту, тому вони зацікавлені в реалізації проекту і готові до взаємодії. Прикладом протагоніста є фермерський кооператив, агроном, еколог, розробник, інші

сільськогосподарські організації, що хочуть зробити добудови для моніторингу комах-шкідників.

– Антагоністи — конкуренти, опоненти, суперники проекту, що розроблюється. Прикладом вважають конкурентів, що пропонують свої продукти на розгляд, фермерів, які можуть провести аналіз програмного додатку та знайти його слабкі сторони, щоб уникнути відповідальності або зекономити на пестицидах.

Для кращого розуміння, намальована модель стейкхолдерів Менделоу (рис. 1.9)

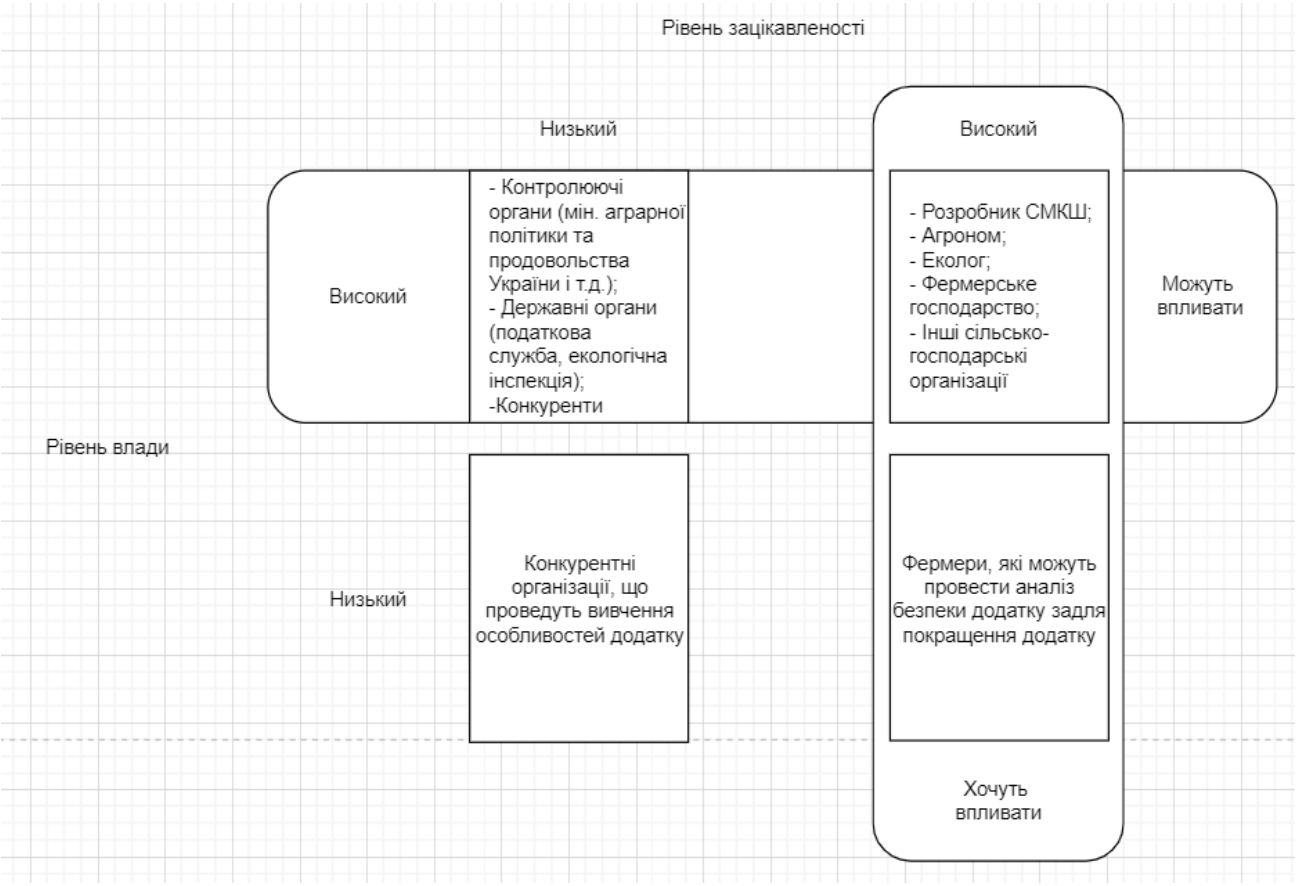


Рисунок 1.9 – Схема Менделоу зацікавлених сторін

1.6. Постановка задачі та етапи виконання роботи

Мета роботи: розробити повноцінну систему, яка допоможе агрономам, екологам та фермерам здійснювати щоденний моніторинг кількість комах-шкідників для запобігання пошкодження посівів сільськогосподарських

культур. Розроблена система буде видавати користувачам список класифікованих комах та їх кількість, виявлених на липких стрічках.

*Об'єкт*ом роботи називають явище або процес, який створює певну проблематику, за вирішення якої береться дослідник, зацікавлена особо. Об’єкт: запобігання пошкодження посівів культур від комах-шкідників

Предметом є все що лежить в межах об'єкта. Предмет дипломної роботи - це моніторинг комах-шкідників, що потрапити на досліджувану липку стрічку.

Основні цілі автоматизованої системи моніторингу комах-шкідників:

- боротьба з комахами-шкідниками на ранніх стадіях;
- пошук певного балансу у взаємодії між комахами і зерновими культурами для отримання їх взаємної користі;
- підвищення ефективності пестицидів;
- запобігання та зменшення пошкодження врожаю;
- поліпшення врожайності та її якості.

Функціональні вимоги СМКШ:

- визначає види та кількість наявних комах-шкідників;
- використовується лише користувачами операційної системи Windows;
- забезпечує вивід даних стосовно моніторингу шкідників;
- надає можливість користувачу редагувати навчання класифікаційної моделі.

Нефункціональні вимоги системи:

- зрозуміла для користувача у використанні; надійна, з точки зору захисту персональних даних;
- простий у використані користувацький (зовнішній) інтерфейс;
- містить обмеження у часі (система має оброблювати дані та видавати результат впродовж 5 хвилин)

1.7. Вибір та коригування набору даних для навчання нейронної мережі

В Європі Trialeurodes vaporariorumi (тепличний білокрилка) у помідорах входять до ТОП-10 найбільш проблемних шкідників тепличних овочевих культур [16]. У рамках проекту CIPM PeMaTo-Europep увага приділяється комплексній підхід у боротьбі з цими шкідниками. Біологічна боротьба в бельгійських та голландських теплицях помідорів заснована на розповсюдженні хижого клопа Macrolophus pygmaeusi та Nesidiocoris. Цей хижак взаємодіє з усіма шкідниками і є корисними в теплиці. Тому за рівнем хижака і проблемної комахи слід ретельно стежити.

Набір даних Yellow Sticky Traps складається з 284 зображень, записані в статичній установці за допомогою Scoutboxes. Кожен Scoutbox оснащений рамкою для утримання пасток і дзеркальною камерою, яка записує зображення з роздільною здатністю 5184×3456 пікселів. Оригінальний набір даних містить 7413 мічені комахи трьох класів (таблиця 1.1, 1-й стовпець).

Таблиця 1.1. Об'єкти на клас у вхідних і виправлених наборах даних

Клас	Оригінал	Після виправлення
Macrolophus pygmaeus (MR)	1312	1619
Nesidiocoris tenuis (NC)	510	688
Trialeurodes vaporariorum (TV)	5591	5807

Комахами, включеними до цього набору даних, є Macrolophus pygmaeus (MR), Nesidiocoris tenuis (NC) і Trialeurodes vaporariorum (TV) (рис. 1.10 а,б,в відповідно). Macrolophus pygmaeus розміром від 3 до 4 мм — це літаюча комаха, яка використовується для боротьби з білокрилками. Macrolophus pygmaeus здебільшого зелені з прозорими крилами і темними вусиками. Вони їдять яйця та личинки білокрилки, але може також пити сік рослин, якщо немає інших джерел їжі. Оскільки вони також використовує рослини як джерело їжі, якщо білокрилок недостатньо для їжі, важливо розрахувати точну кількість Macrolophus pygmaeus, необхідну для підтримки балансу між хижаком і жертвою [17].

Nesidiocoris tenuis належить до того ж сімейства, що і Macrolophus pygmaeus, але в цілому більший. Його розміри від 5 до 6 мм тому, йому потрібно більше їжі і він швидко розмножується. У порівнянні з меншим Macrolophus pygmaeus, він більше поселяється у верхній частині рослини і поїдає молоді частини рослини, що часто завдає більшої шкоди рослині. Nesidiocoris tenuis (рис. 1.10в) також переважно зеленого кольору, але має чорні акценти на крилах, чорні коліна та чорне кільце на шиї, а також чорні смуги на вусиках [18].

Trialeurodes vaporariorum (рис. 1.10в) званий білокрилкою, є двоміліметровою літаючою комахою, яка вражає переважно рослини помідорів. Він п’є сік рослини і таким чином пошкоджує рослину. Білокрилка має білі пудроподібні крила і світло-жовте тіло. Через дуже маленький розмір його може бути важко розпізнати на зображеннях [19] навіть візуально.

Рисунок 1.10 – класи набору даних, зліва-направо: MR , NC, TV

Існують також деякі мітки для комах-трипсів як четвертого класу, але було позначено лише кілька екземплярів, тому цей клас не використовується. Окрім мічених комах, які потрапляють у пастки, на зображеннях присутні також деякі інші комахи, такі як джмелі та мухи (рис. 1.11б), а також частини рослин, як-от листя (рис. 1.11б), що посилює складність розпізнавання комах.

Рисунок 1.11 – приклади зображень із набору даних жовтих липких пасток

Проблеми та коригування анотацій набору даних:

Під час тренування нейронної мережі було виявлено кілька проблем у вихідному позначеному наборі даних, які довелося виправити. По-перше, деякі зображення орієнтовані в альбомній орієнтації, а інші – в портретній. Обмежувальні рамки не можуть відповідати об’єктам на зображеннях, оскільки вони орієнтовані в альбомній орієнтації. Таким чином, було змінено інформацію про поворот Exif зображень із портретної на альбомну у програмному середовищі Roboflow, що усунуло проблему.

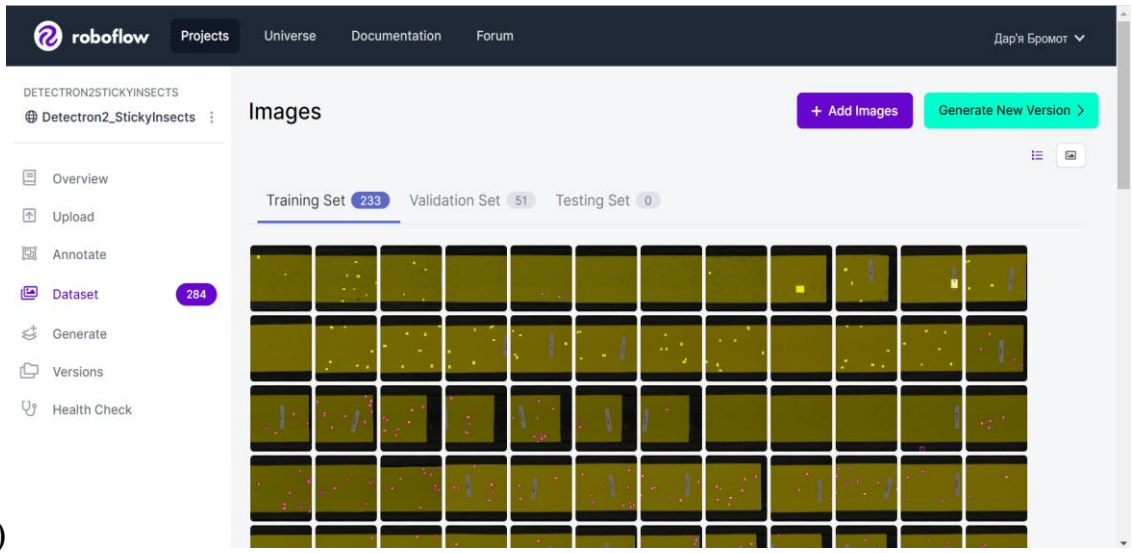
Roboflow дає розробникам всі інструменти, необхідні для перетворення необроблених зображень у спеціально навчену модель комп’ютерного зору та розгортання її для використання в програмах.

Roboflow можна використовувати для:

- додавання анотацй до зображень або завантаження наявних анотацій;
- перетворення наявних анотації VOC XML в анотації COCO JSON;

- попередної обробки зображень: зміна розміру, відтінки сірого, автоматична орієнтація, налаштування контрасту;
- доповнення зображень, щоб збільшити дані про тренування: перевертання, повертання, освітлення/затемнення, обрізання, зсування, розмивання та додавання випадкового шуму;
- створення форматів анотацій, таких як TFRecords, CreateML і Turi Create, і користувацьких реалізацій YOLOv3 (плоскі текстові файли або Darknet);
- швидкої оцінки якісті набору даних;
- спільного навчання зі своєю командою наборами даних, які контролюються версіями;
- легкого використоння даних в моделях, створених у Tensorflow, PyTorch, fast.ai, Keras тощо;
- розгортання моделі як безмежно масштабований API, на периферійному пристрої або у веб-браузері;
- пошук загальнодоступних наборів даних [20].

Приклад роботи в середовищі Roboflow наведено на рисунку 1.12 (а,б). Зокрема вкладки перегляду датасету, що використовується для навчання мережі, поділеного на набори, і анотування вручну в онлайн режимі комах-шкідників на зображеннях.



а)

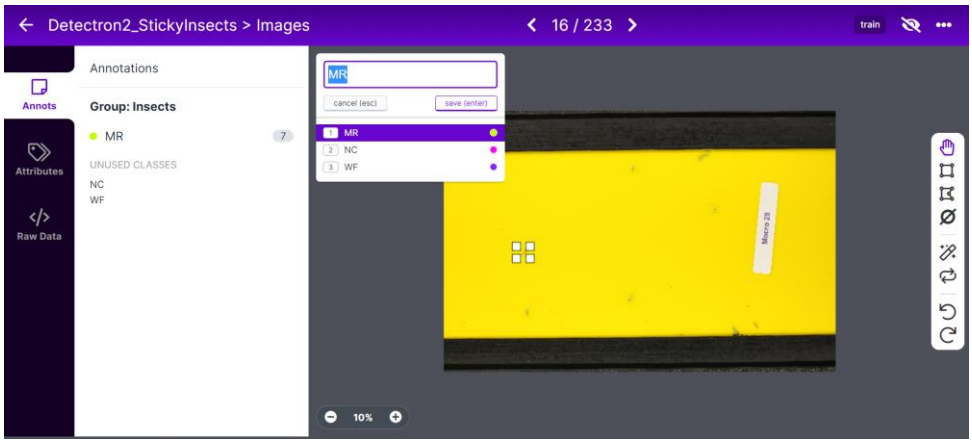
б) 

Рисунок 1.12 – робота з : а) вкладка перегляду; б) вкладка анотування

Крім того, багато прямокутників були погано розташовані на об’єктах, або об’єкти взагалі не були позначені. Це може призвести до неправильного відхилення об’єктів через низькі показники IOU (Intersection over Union — термін, який характеризує ступінь перекриття двох блоків) із неправильними обмежуючими рамками або через те, що вони не позначені. Дійсно, під час перших експериментів було знайдено кілька зображень у тестовому наборі з непозначеними об’єктами (рис. 1.13), які були правильно виявлені мережею.

а)

б)

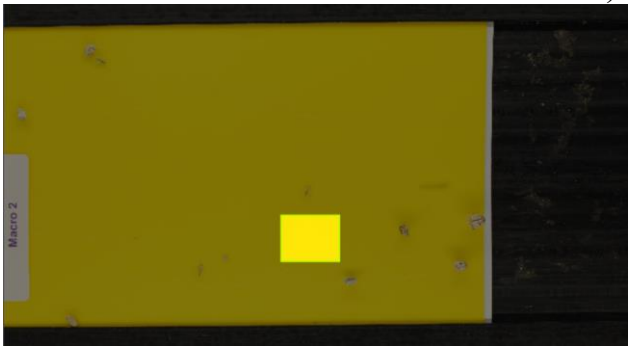
в) 

Рисунок 1.13 – приклади проблемних анотацій: а) велика область з відсутніми мітками; б) зайві міткі; в) неправильно обмежуючі бокси

Після виправлення анотацій виправлений набір даних містить загалом 8114 (+701) комах (таблиця 1, 2-й стовпець). Набір даних був розділений на набори для навчання та тестування із співвідношенням 0,7 та 0,3. Основна інформація зберігається у файлах JSON у форматі COCO. Слід також зазначити, що класи в цьому наборі даних є незбалансованими, як 71.57% з усіх об'єктів складається з Trialeurodes vaporariorum, тоді як Nesidiocoris tenuis вносить лише 8.48%. Через це важливо також розраховувати інші показники, такі як Precision и Recall, які будуть представлені в цій роботі.

Висновок перший розділ

Отже, перший розділ дипломної роботи включає в себе описання області застосування моніторингу комах-шкідників (актуальність роботи, мета та цілі створення відповідного програмного застосунку), аналітичний огляд методів виявлення комах-шкідників (розглянуті способи виявлення комах в пастках з липкою стрічкою, на рослинах, на опорній плиті в пастці), аналіз методів класифікації комах (порівняння основних типів сучасних згортковий нейронних мереж), виявлення профілів зацікавлених сторін проекту (намальована модель стейкхолдерів Менделоу, для кращої візуалізації), розглянуті наявні рішення (існуючі автоматичні системи МКШ, створені для конкретних регіонів з досить дорогим обладнанням), представлена коротка теоретична відомість про можливі способи підрахунку шкідників, що будуть використовуватись у програмі, а також розписана постановка задачі (тема, мета, об’єкт та предмет досліджень, функціональні та нефункціональні вимоги до програмного продукту).

Також було здійснено пошук набору даних для МКШ на жовтих липких стрічках, виправлено помилки в зображеннях, покращивши розміщення обмежувальних рамок, виправивши мітки класів і позначивши об’єкти без міток.

РОЗДІЛ 2. РОЗРОБКА АРХІТЕКТУРИ МОДУЛЮ КЛАСИФІКАЦІЇ ТА ВИЯВЛЕННЯ КОМАХ-ШКІДНИКІВ

2.1. Функціональний аналіз

Технології ГН внесли значний внесок у сферу МКШ, метою якої є розпізнавання комах-шкідників на експертному рівні з мінімальною підготовкою оператора. Існують методи для реалізації цього завдання з більш високою точністю і меншою вартістю обчислень. Дані методи глибокого навчання та комп'ютерного зору, можна узагальнити та розділити на 2 етапи: побудова класифікаційної моделі та аналіз зображення за допомогою моделі.

Розглянемо детально кожен етап.

Побудова моделі класифікації включає в себе такі підфункції:

1. Отримання зображення:

– пошук в Інтернеті (отримання зображення з інтернет-пошукових систем або баз даних, таких як Google, Baidu, Naver і FreshEye тощо);

– відкриті набори даних (добре побудовані експериментальні набори даних є загальнодоступними для вільного доступу для вдосконалення та порівняння моделі

– отримання фотографії зі стаціонарної камери (отримання зображення за допомогою розумних пасток та камери, закріплені в них, що автоматично фіксують комах-шкідників протягом тривалого періоду).

2. Попередня обробка зображення:

– геометричні перетворення (включає поворот зображення на певний кут, горизонтальне або вертикальне положення, зсув і масштабне перетворення тощо)

– зміна розміру зображення (збільшення/кадрування розмірів зображень комах до фіксованого розміру, щоб відповідати вхідним вимогам мережі ГН);

– покращення якості зображення (зменшення шуму та/або підвищення різкості зображення для вдалого виявлення комах-шкідників);

– перетворення колірного простору (оскільки цифрове зображення RGB складається з 3-х окремих кольорових каналів, кожен з яких може бути перетворений в інший колірний простір (наприклад HSV, YUV і CMY), то значення пікселів колірного каналу можна налаштувати на постійне значення або на певне мінімальне/максимальне значення).

4. Налаштування параметрів моделі.

3. Процес навчання моделі.

5. Тестування.

6. Зберігання моделі класифікації.

У свою чергу, аналіз зображення включає:

– завантаження зображення з ПК;

– виявлення комах (пошук як поодиноких дрібних комах, так і багатомасштабне виявлення шкідників);

– класифікація (розпізнавання комах-шкідників за допомогою згорткої нейронної мережі)

– підрахунок і поділ на класи шкідників (для розуміння загальної картини підраховується кількість комах кожного наявного класу).

Поділені функціональні підкласи можна побачити на рисунку (рис. 2.1)

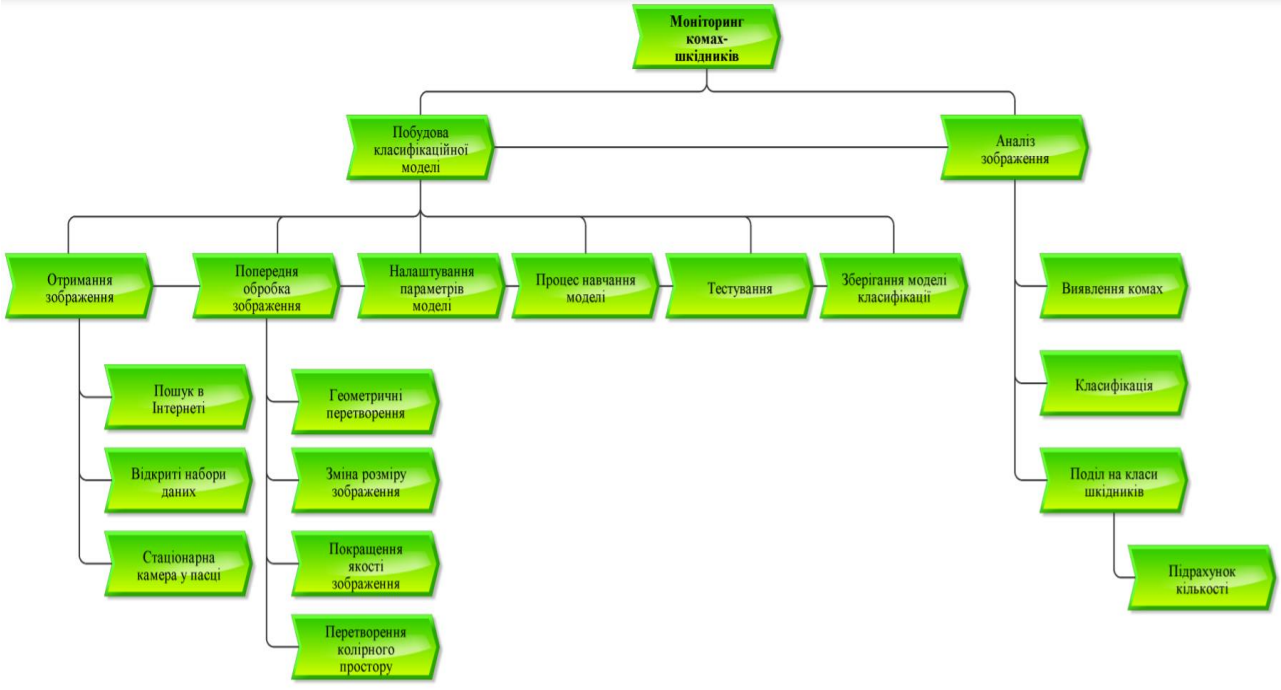


Рисунок 2.1 – Дерево функцій модулю МКШ

2.2. Архітектура системи у нотації IDEF0

Забезпечує як процес, так і мову для побудови моделі рішень, дій і діяльності в організації.

– визначення, задокументування та повідомлення про основну діяльність системи.
– для розуміння як діяльність пов’язана одна з одною.
– для визначення види діяльності, які необхідно вдосконалити.

На рисунку нижче представлена діаграма IDEF0 (рис. 2.2)

Рисунок 2.2 – IDEF0 процесу МКШ

Опис стрілок (ICOM) системи:

1. Input (Входи) – це реальні об'єкти або дані, необхідні для виконання моніторингу комах-шкідників: завантаження зображення з Інтернету, ПК користувача, стаціонарної камери встановленої з розумної пастки, тощо.

2. Control (Керування) – це правила чи документи, які керують виконанням функції або визначають результати: закон України "Про захист персональних даних" (контроль доступу до файлів на комп’ютері/телефоні користувача) та правило розподілу на групи (вирішення питання стосовно того, який відсоток подібності достатній, щоб зарахувати комаху-шкідника до наявної групи).

3. Mechanism (Механізм) – це особа, пристрій або дані, які виконують функцію: користувач (фермер, агроном, еколог, автоматична розумна пастка, тощо), який дає на вхід зображення, та інформаційна система класифікації комах-шкідників, яка виявить на розподілить комах на групи.

4. Output (Вихід) – це дані, отримані як результат функції: список виявлених, підрахованих та класифікованих по групам комах-шкідників, записаних у таблицю.

2.3. Архітектура інформаційної системи

StarUML — це інструмент моделювання програмного забезпечення з відкритим вихідним кодом, який підтримує фреймворк UML для моделювання системи та програмного забезпечення. Він активно підтримує підхід MDA (Model Driven Architecture), підтримуючи концепцію профілю UML і дозволяючи генерувати код для кількох мов.

Для побудови архітектури інформаційної системи МКШ, необхідно визначити компоненти системи та в програмному середовищі StarUML створити компонентну діаграму (рис 2.3). Додаток матиме просту клієнт-серверну архітектуру і, відповідно, 2 вузли.

Клієнтський вузол («Client») буде складатися з таких компонентів:

- отримання даних від користувача («Data Package»);
- мережева взаємодія з сервером («ClientNetwork»);
- опис інтерфейсу користувача («ClientGUI»).

Вузол серверної частина («Server») матиме три компоненти:

- підготовка вихідних даних («Data Preprocessing»);
- попередня обробка зображення («Image preprocessing»);
- навчання нейронної мережі («NN training»);
- класифікація («Classification»);
- бізнес-логіка сервера («ServerNetwork»);
- мережева взаємодія з клієнтом («ServerBusinessLogic»);

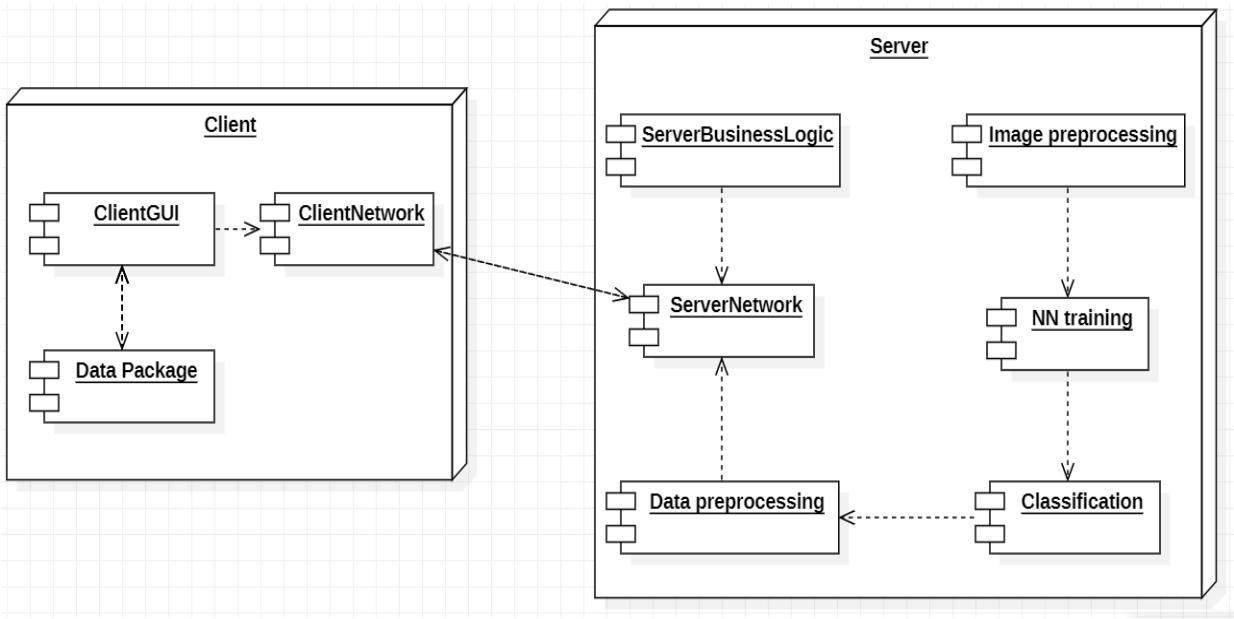


Рисунок 2.3 – Компонентна архітектура застосунку

Додаток складається з 2-х частин: «Client» і «Server».

«Client» містить два компоненти - ClientGUI , в якому знаходиться опис інтерфейсу користувача, і «ClientNetwork», що відповідає за мережеву взаємодію з сервером. Причому перший компонент залежить від другого. Завданням клієнта буде збирання даних («DataPackage») від користувача та виведення результатів МКШ.

«Server» містить усі проекти програми, які реалізують роботу сервера. «ServerBusinessLogic» містить весь код, що реалізує бізнес-логіку сервера, «ServerNetwork» реалізує мережеве повідомлення з клієнтом. «Image preprocessing» відповідає за попередню обробку зображення, результат роботи, якої подається на вхід до «NN training», де відбувається навчання нейронної мережі. Вхідні дані класифікуються в «Classification» та передаються в «Data Preprocessing», де відбувається підготовка вихідних даних, які автоматично відправляються користувачу.

2.4. Діаграми життєвих циклів

Перейдемо до описання основних життєвих циклів та їх переходів. Побудуємо життєвий цикл моніторингу комах-шкідників (рис. 2.4):

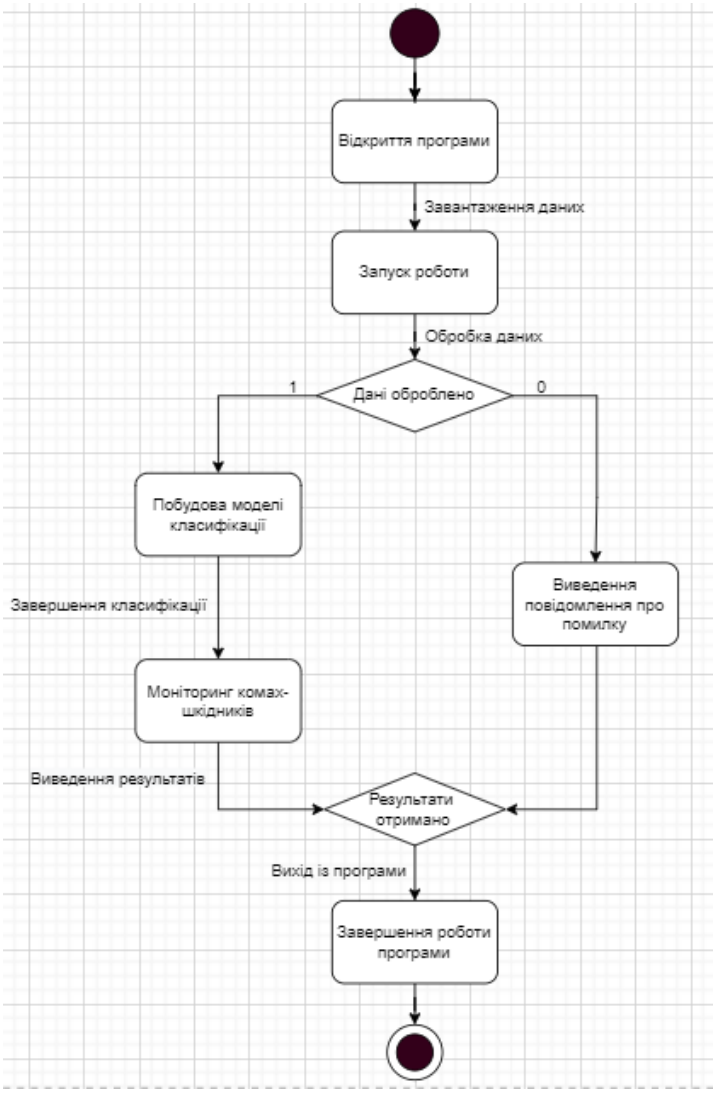


Рисунок 2.4 – Життєвий цикл прогнозу

Побудуємо життєвий цикл моделі класифікації (рис. 2.6).

Після завантаження програми, користувач вибирає модель прогнозування. Вхідні дані користувача оброблюються і класифікуються системою, після чого надсилаються назад. Процес тренування майбутньої класифікаційної моделі матиме такі стани життєвого циклу:

- відкриття програми;
- завантаження даних;
- попередня обробка даних;
- налаштування параметрів моделі;
- навчання;
- зберігання класифікаційної моделі.

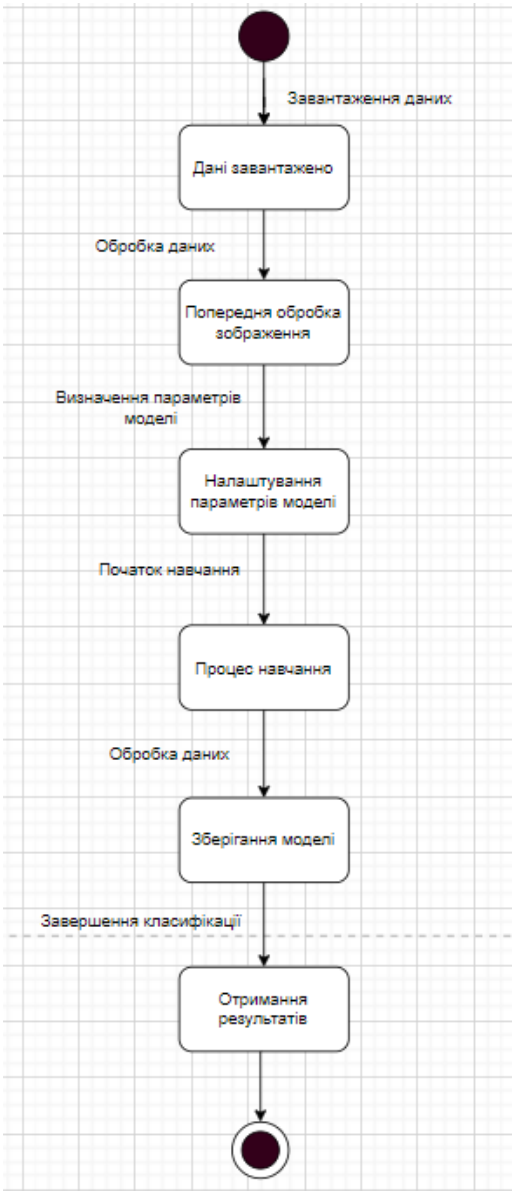


Рисунок 2.6 – Життєвий цикл навчання моделі

Визначимо акторів, прецедентів та їх відношення системи. Як визначалося раніше, в роботі беруть участь безпосередньо користувач та система.

Для користувача буде доступно наступні дії:

- завантаження даних;
- отримання класифікаційної моделі;
- отримати результатів МКШ.

Для системи буде характерно наступні дії:

- попередня обробка зображення;
- налаштування параметрів моделі;

- навчання моделі;
- виявлення та класифікація шкідників;
- моніторинг комах-шкідників.

З неведеного вище опису побудуємо UML діаграму (рис 2.7).

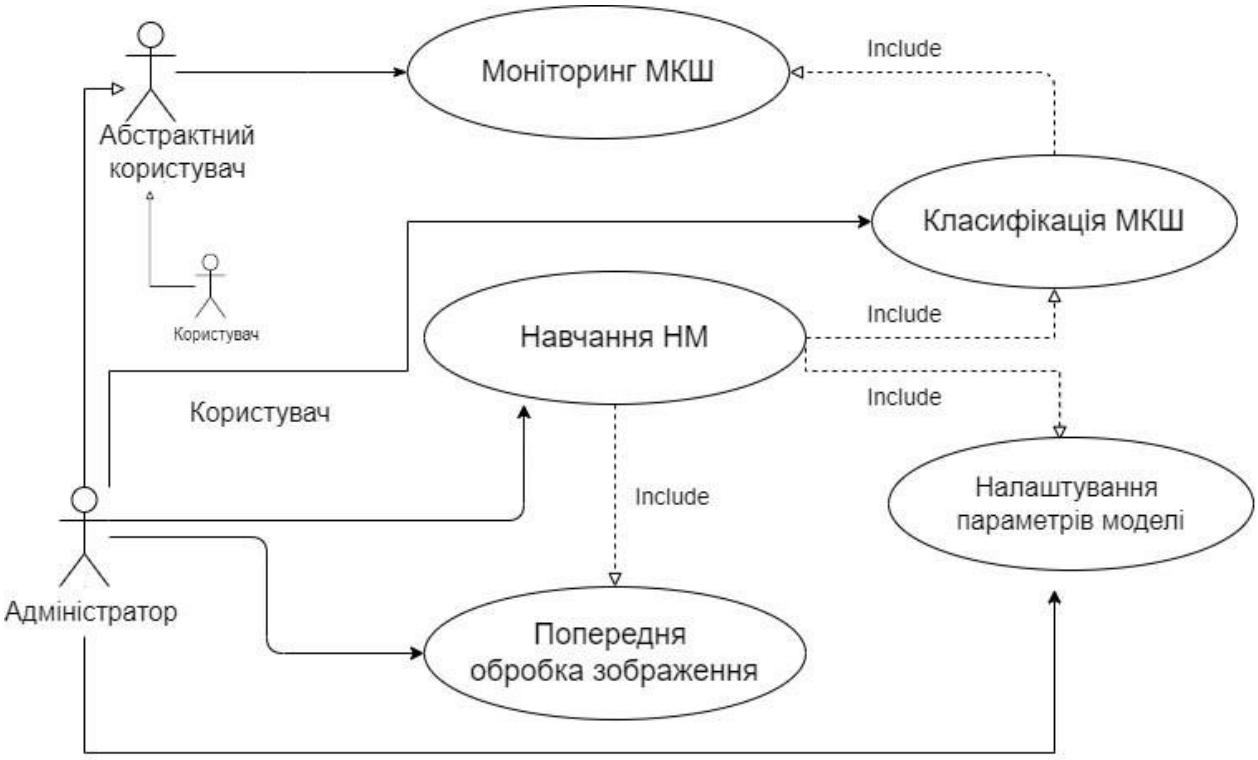


Рисунок 2.7 – UML діаграма прецедентів

2.5. Узагальнена технічна архітектура

1. Акторами у системі є користувачі (фермери, екологи, агрономи, просто зацікавлені особи), що заходить зі свого комп'ютера на сайт для моніторингу комах на своїх земельних ділянках). До сайту дані можна завантажити з комп'ютеру, телефону чи стаціонарної веб-камери. А також актором є адміністратор, який займається налаштуванням мережі на віддаленому сервері.

2. Користувач робить запит до системи на перевірку його зображень.

3. Система для моніторингу комах-шкідників (у нашому випадку Amazon Web Services) відправляє запит до створеного Amazon Elastic Compute Cloud, що повинен обробити зображення на віддаленому комп'ютері і провести розпізнавання та класифікацію комах.

4. Під час розпізнавання йде підрахунок комах-шкідників за вказаних користувачем відсотком якості.

5. Як результат на екран користувача виводиться оброблене зображення яке містить прямокутні бокси відповідно до виявлених комах-шкідників. На кожному боксі зображено клас до якого відноситься комаха, а також її відсоток приналежності до певного класу. У текстовому форматі виводить загальна кількість шкідників різних класів.

6. Якщо користувач завантажить на сервер відразу кілька картинок, програма обробить кожне зображення і виведе загальну кількість шкідників на всі зображення.

Узагальнена архітектура системи наведено на (рис. 2.8)

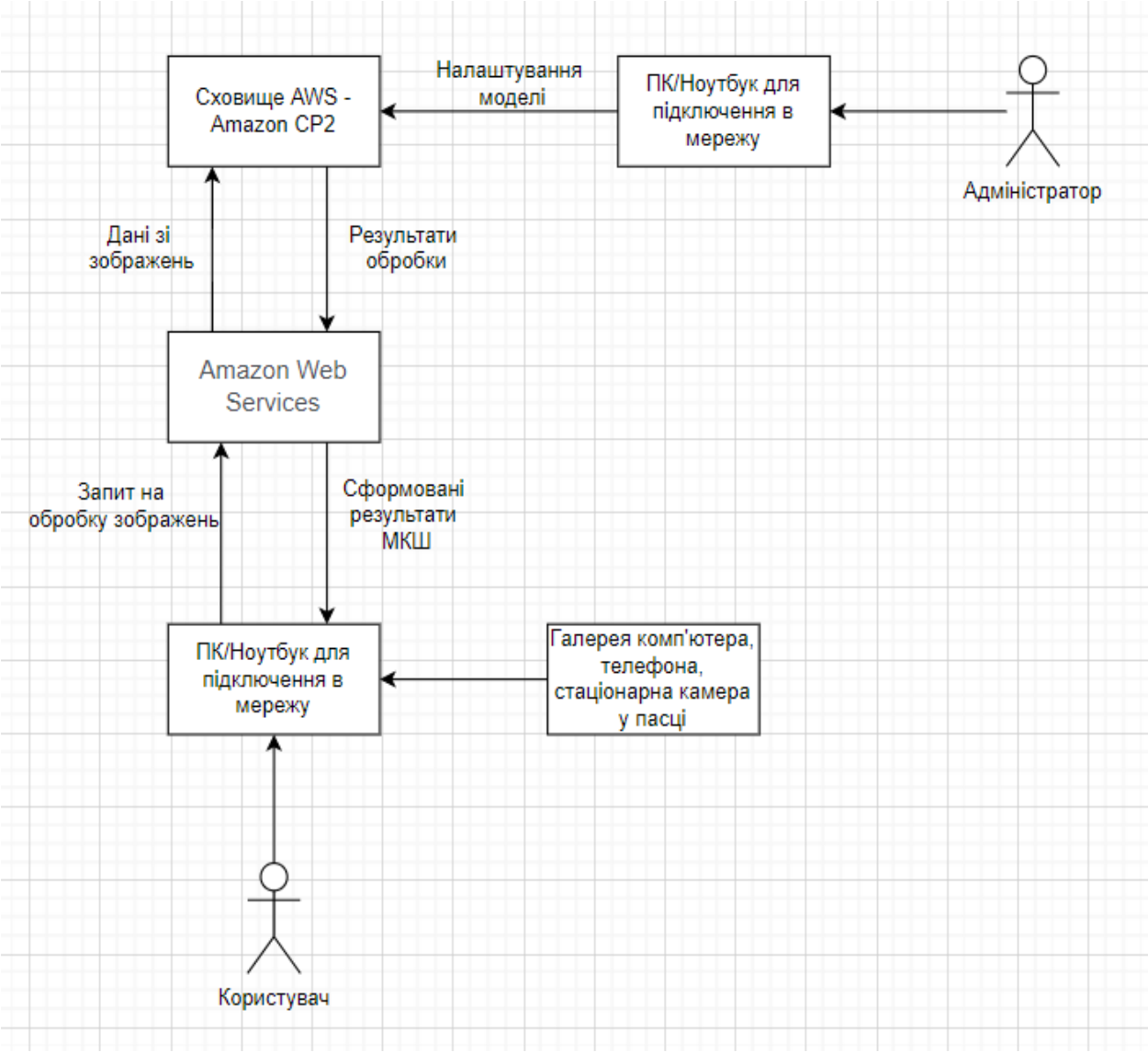


Рисунок 2.8 – Узагальнена архітектура системи

2.6. Побудова бізнес-моделі програмного модуля за допомогою EPC

Методологія ARIS дає можливість описувати досить різнорідні підсистеми у вигляді взаємопов'язаної і взаємоузгодженої сукупності різних моделей, які зберігаються в єдиному сховищі. Саме взаємопов'язаність і взаємоузгодженість моделей є відмінними рисами методології ARIS.

EPC (Event-Driven Process Chain) – нотація відображення ходу виконання процесу, ключовими елементами якої є Event – Події і Функції. Нотація EPC була розроблена в 90-х роках XX століття в рамках методології ARIS В.А. Шєєром [21].

Діаграма бізнес-процесу в EPC повинна починатися і закінчуватися подією. Після функції завжди має відбуватись подія, тобто виконання Функції створює деяка подія або стан. Бази даних, програмне забезпечення, документи тощо мають графічне позначення.

Діаграма бізнес-процесу описує:

– які дії виконуються в ході процесу;

– які організаційні підрозділи беруть участь у виконанні процесу

– які вхідні та вихідні дані використовуються;

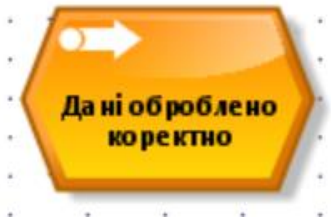
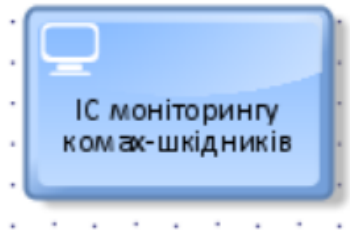
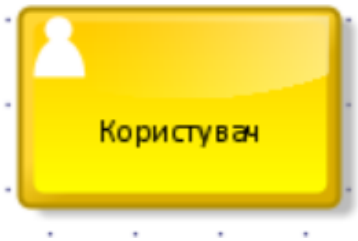
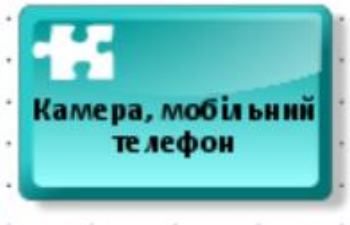
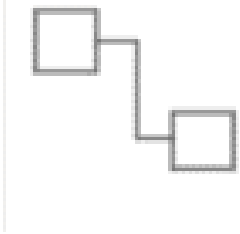
– які IT-системи задіяні;

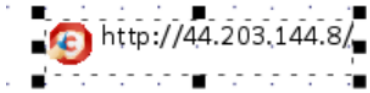
– які події відбуваються в процесі.

Усі можливі варіанти робочого процесу представлені альтернативно за допомогою правил. Діаграма наведена у роботі побудована у нотації Event-Driven Process Chain у програмі ARIS Express. Далі представлена таблиця із умовними позначеннями , що містяться на діаграмі (Таблиця 2.1)

Таблиця 2.1. – Event-Driven Process Chain

Позначення елементів	Опис
	*Activity/(Діяльність)* – кроки, які необхідно виконати – описує, що відбувається під час процесу, тобто що саме робиться. Вони є основними елементами процесу. Приклад: завантажити зображення.

	*Event/(Подія)* – стан системи, процесу, що впливає на виконання функцій або керує ними. Наприклад: зображення завантажено.
	*IT-system (IT-система)* – посилання на EDP (electronic data processing): дії можуть виконуватися вручну або автоматично. Автоматизовану діяльність здійснюють IT-системи. Приклад: AWS/EC2.
	*Role (роль, посада)* – розподіл ролей для діяльності – ілюструє, хто виконує діяльність. Приклад: людина (користувач, адміністратор).
	*Entity (сутність, об'єкт)* – об'єкти систем, зовнішні об'єкти. Приклад: підключена стаціонарна камера з теплиць, камера мобільного телефону, гелерея ПК користувача.
	*Потік даних* – потік даних у процесі: вхідні та вихідні дані. Процес генерує дані або вимагає даних для продовження. Ці дані моделюються як вхідні або вихідні дані діяльності.
	Правила – описують альтернативи робочого процесу і таким чином ілюструють можливі варіанти виконання. Доступні такі символи правил: Оператор АБО: можна використовувати будь-яку кількість шляхів. Оператор І: усі наступні/попередні шляхи застосовні. Виключне АБО: може мати місце лише один з наступних/попередніх варіантів (як у цьому прикладі: або дані обробилися коректно, або виникла помилка).

	*Атрибути* – об’єкти, моделі та зв’язки можуть мати властивості. Ці властивості в ARIS називаються «атрибутами» і можуть підтримуватися в поданні «Атрибути» (головне меню: Перегляд/Атрибути). У нашому випадку показано атрибут «Посилання».

1. Ролі:

1.1. Користувач (відвідувач веб-сайту: фермер, агроном, еколог, зацікавлена особа).

1.2. Адміністратор (той, хто налаштовує параметрі моделі і відповідає за функціональність серверу).

2. Діяльність:

2.1. Відкрити веб-сайт (користувач вводить в пошуковому рядку назву сайту, атрибут на діаграмі, для моніторингу комах-шкідників).

2.2. Завантажити зображення (користувач завантажує одну або декілька зображень жовтої липкої стрічки з комп’ютеру, мобільного телефону чи стаціонарної камери).

2.3. Ввести допустиму точність моделі (користувач вводить задовільну для нього Score Treshhold, чим менше -> більше областей класифіковано (може бути менш точним)).

2.4. Обробка даних (розпізнавання, класифікація та підрахунок комах-шкідників на сервері).

2.5. Відправити дані користувачу (вивести на екран користувача оброблені зображення, яке містить виявлених комах-шкідників, їх відсоток приналежності до певного класу та загальна кількість шкідників різних класів).

2.6. Відправити повідомлення про помилку (виведення в консолі адміністратора повідомлення про помилку в момент коли програма дала збій).

2.7. Повторне введення даних (користувач може вести нові дані, або коригувати попередні).

3. Події:

3.1. Сайт вдало завантажився (веб-сайт готовий для використання);

3.2. Сталася помилка (користувач має проблеми зі своїм пристроєм, маршрутизатором чи провайдером, є загроза наявності фішингової сторінки або вірусу, проблеми з DNS-адресами, тощо).

3.3. Зображення завантажено (вибрані користувачем одне або декілька зображень жовтої липкої стрічки вдало додалися до системи).

3.4. Відправка даних на сервер (зображення відправляються на сервер для обробки і моніторингу).

3.5. Дані оброблено коректно (введені користувачем зображення задовільні).

3.6. Сталася помилка при обробці (технічні причини, наприклад, проблеми з провайдером, або ж програмні – завантажене зображення неможливо обробити).

3.7. Вивід результатів на екран (вивести на екран користувача оброблені зображення, яке містить виявлених комах-шкідників, їх відсоток приналежності до певного класу та загальна кількість, виведення в консолі адміністратора повідомлення про помилку в момент коли програма дала збій).

3.8. Користувач бажає вести нові дані (користувач може вести нові дані, або коригувати попередні).

3.9. Моніторинг комах-шкідників завершено (користувач закінчив роботу і виходить з сайту).

4. IT-система:

4.1. ІС моніторингу комах-шкідників (знаходиться на хмарному сервісі AWS, зберігає у собі інформацію навченою нейронною моделлю).

5. Зовнішні об'єкти:

5.1. Стаціонарна камера, камера мобільного телефону (користувач може використовувати утиліти для завантаження зображення).

Діаграма EPC наведено на рисунку 2.9:

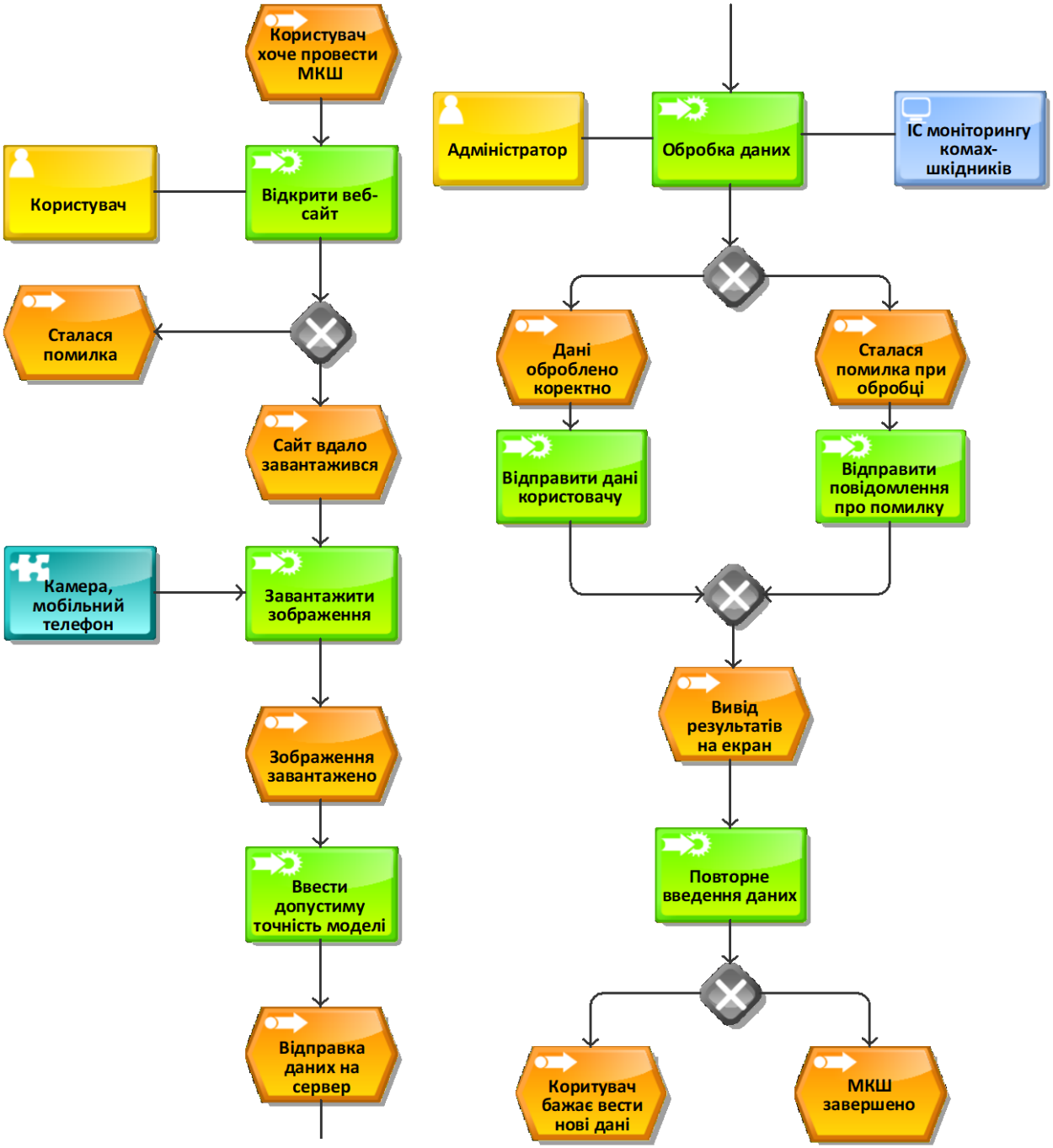


Рисунок 2.9 – Діаграма бізнес-процесу в (EPC), розроблюваного модуля

2.7. Фізична архітектура системи

Система складається із наступних об’єктів:

Папки:

– *insects* – загальна папка, яка містить в собі весь проект.

– *container* – папка створена для полегшення роботи з Docker.

– *venv* (скорочення від Virtual Environment) – автоматично створена середовищем розробки PyCharm папка у якій містяться усі бібліотеки з якими встановлена залежність у проекта, буде видалена перед викладенням сайту на сервер.

Скрип-нейромережі:

Папка *model* є моделлю, створеної та натренованої нейронної мережі. У ній міститься папка *output* з наступними вихідними файлами:

– *instances_predictions.pth* – файл у PyTorch-форматі, який містить усі вихідні прогнози.

– *coco_instances_results.json* – усі виявлені об’єкти (ідентифікатор зображення, обмежувальна рамка, індекс класу та точність), записаний у COCO-форматі.

– *config.yaml* – конфігурація моделі Detectron2.

– *model_final.pth* – ваги моделі у форматі PyTorch.

– *metrics.json* – навчальні показники (наприклад, час, загальні втрати тощо) кожні 20 ітерацій.

Python скрипти:

– *streamlit_insects_app.py* – основний скрипт для інтерактивного застосування нейронної мережі, який виконує усі вищеописані функції. Тобто записує оброблює зображення надані користувачем, розпізнає і класифікує комах–шкідників, а також веде підрахунок комах відповідно від класу приналежності.

AWS–скрипти:

– *docker-compose.yml* – це файл YAML, який визначає служби, мережі та команди для програми Docker.

– *Dockerfile* – це текстовий документ, який містить усі команди, які користувач може викликати в командному рядку, щоб зібрати додаток.

Інші файли:

– *requirements.txt* – залежності проекту, бібліотеки для встановлення.

Фізична архітектура системи представлена на рисунку 2.10:

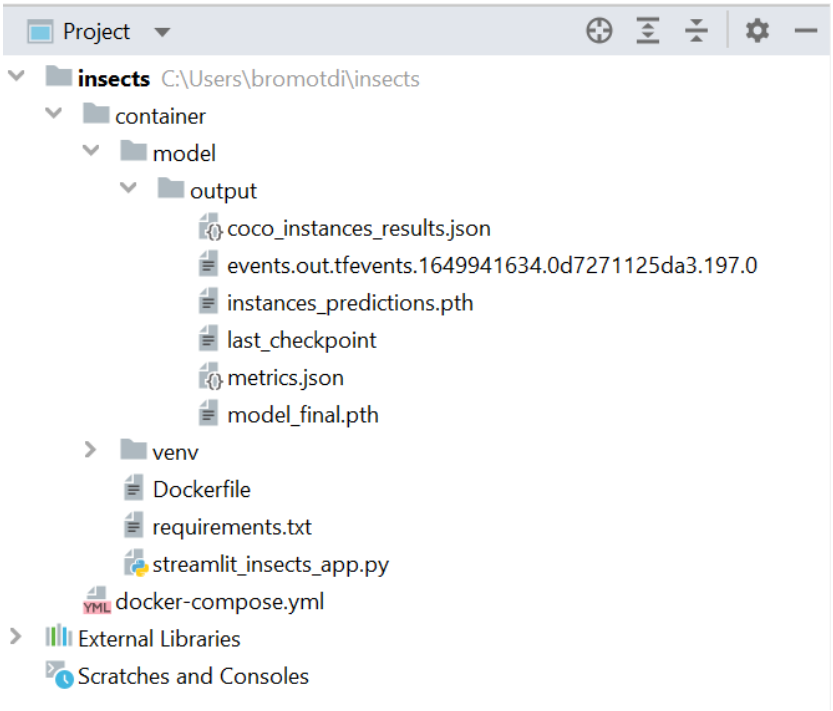


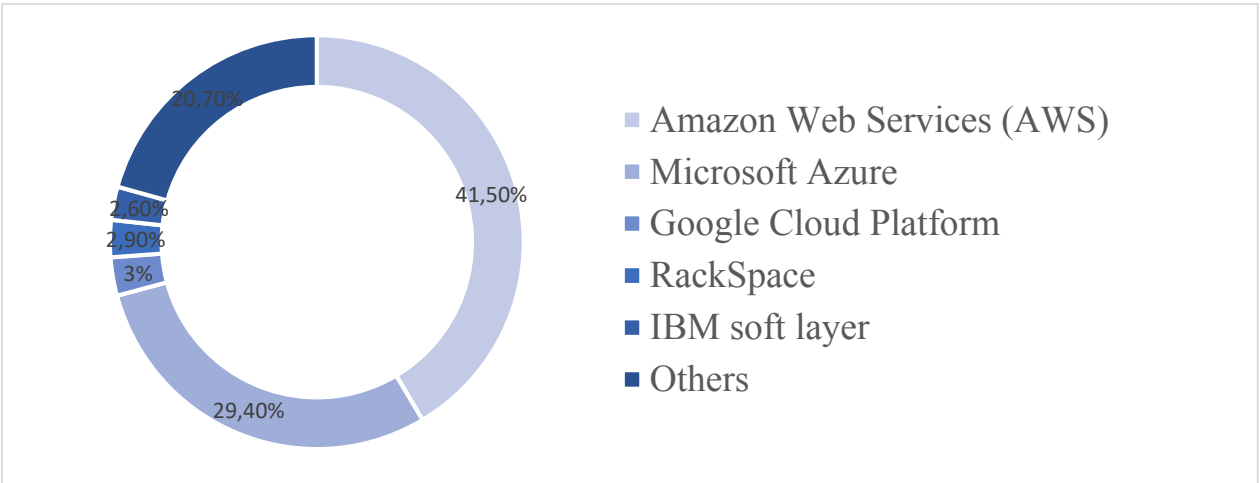
Рисунок 2.10 – Фізична архітектура системи

Детальний розбір функцій кожного скрипта наведено в розділі 3.1.

2.8. Вибір середовища та інструментів реалізації

1. Amazon Web Services (AWS)

Google , Microsoft і IBM є гігантами технології, що лідирують у багатьох сферах і мають мільярди користувачів. Але є бізнес сегмент, в якому багато років вони завжди позаду Amazon (діаграма 2.1). Дуже багато відомих компаній зараз є клієнтами у AWS як-от Netflix, Twitter, New York Times, Nasdaq.



Діаграма 2.1 – Рейтинг сервісів хмарних обчислень

Саме тому для програмного модуля був обраний Amazon Web Services – комерційна публічна хмара, головними функціями AWS є обчислення (computing), зберігання (storage), безпека, аналіз даних, послуга штучного інтелекту, платформа Internet of Things .

Дві найпопулярніші послуги AWS це Amazon Elastic Compute Cloud (Amazon EC2) та Amazon Simple Storage Service (Amazon S3) **.** Обидві послуги надають безлімітні ресурси користувачам для зберігання, обчислення з можливістю швидко змінювати обсяг роботи. Для поставлених задач було обрано Amazon EC2 через можливість швидкого запуску із попередньо встановленими популярними платформами та інтерфейсами глибокого навчання, такими як TensorFlow, Keras, що дозволяє навчити складні нейроні моделі, експериментувати з новими алгоритмами [22].

2. Docker та Docker Compose

Docker спрощує керування процесами програми завдяки створенню контейнерів. Хоча контейнери певним чином схожі на віртуальні машини, вони є більш легкими та безпечними для ресурсів. Це дозволяє розробникам розбити середовище програми на кілька ізольованих серверів.

Для програм, які залежать від кількох серверів, організовувати всі контейнери для спільного запуску, зв’язку та закриття може швидко стати громіздким. **Docker Compose** — це інструмент, який дозволяє запускати багатоконтейнерні програми на основі визначень, заданих у файлі YAML. Він використовує визначення служб для створення повністю настроюваних середовищ з кількома контейнерами, які можуть спільно використовувати мережі та обсяги даних.

В даному програмному модулі Docker буде використовуватися на AWS, адже в AWS забезпечена підтримка рішень Docker з відкритим вихідним кодом та комерційного рівня. Для запуску контейнерів на AWS було обрано Amazon Elastic Container Service (ECS) – високомасштабний та високопродуктивний сервіс управління контейнерами.

3. Google Colaboratory

Google Colab – це безкоштовний хмарний сервіс, наданий компанією Google. Даний сервіс спрямований на спрощеннядосліджень науки про дані та машинного навчання. Google Colaboratory, безумовно, один із найпопулярніших хмарних сервісів. Його головна перевага у тому, що він безкоштовний і надає необхідний набір пакетів табібліотек. У Google Colaboratory не потрібно встановлювати пакети табібліотеки вручну досить просто їх імпортувати. У той час, як узвичайній IDE необхідно вручну встановлювати всі бібліотеки до роботи [23]. Google Colaboratory надає безкоштовно 12.72 ГБ RAM та 48.98 ГБ дискового простору.

4. PIL (або pillow)

Pillow побудована поверх PIL (Python imaging library). PIL - це один з важливих модулів для обробки зображень на Python. Модуль Pillow надає більше функціональних можливостей, працює у всіх основних операційних системах та підтримує Python 3. Він підтримує широкий спектр форматів зображень, таких як "jpeg", "png", "bmp", "gif", "ppm", "tiff". Окрім базової функціональності обробки зображень, включає точкові операції, фільтрацію зображень за допомогою вбудованих ядер згортки та перетворення кольорового простору.

5. Streamlit

Streamlit – це платформа програм з відкритим вихідним кодом для команд ML та Data Science. Завдяки архітектурі розширюваності компонентів можна створювати та інтегрувати більшість видів веб-фронтендів у додатки Streamlit

Streamlit – це єдиний пакет Python, який встановлюється через pip і надає набір функцій, які можуть бути вставлені в існуючий сценарій коду ML. Існують значні технічні відмінності у реалізації Streamlit від Jupyter, ipywidgets, R-Shiny, Voila, React, яка базується на декларативній моделі потоку даних, а не на проведенні зворотних викликів. Фреймворки Python, такі як scikit-learn, spaCy, Pandas, та різні фреймворки візуалізації, такі як Altair, Plotly та Matplotlib – всі вони легко працюють із Streamlit.

6. Microsoft Common Objects in Context (COCO)

COCO — це великий набір даних зображень, призначений для виявлення об’єктів, сегментації, визначення ключових точок людей, сегментації матеріалу та створення підписів. Цей пакет надає API Matlab, Python і Lua, які допомагають завантажувати, аналізувати та візуалізувати анотації в COCO. Даний набір даних широко використовується для порівняння ефективності методів комп’ютерного зору.

COCO — це специфічна структура JSON, яка визначає, як мітки та метадані зберігаються для набору даних зображення. Через популярність набору даних формат, який COCO використовує для зберігання анотацій, часто є основним форматом під час створення нового набору даних для виявлення об’єктів.

7. Detectron2

Detectron2 — це бібліотека наступного покоління Facebook AI Research, яка забезпечує найсучасніші алгоритми виявлення та сегментації. Він є наступником Detectron і maskrcnn-benchmark. Він підтримує низку дослідницьких проектів комп’ютерного зору та виробничих додатків у Facebook. При створені програмного модулю для більшої зручності використовувався інтерфейс Detectron2 для взаємодії з моделлю.

Для вирішення завдань, поставлених у рамках даної роботи, була використана мова програмування Python 3.9, так як вона найчастіше використовується для реалізації алгоритмів глибинного навчання і має високу продуктивність під час обробки даних. Також Python має достатню кількість бібліотек для роботи з цим видом мереж, даними та зображеннями, що дозволяє суттєво прискорити процес написання коду. Для обробки багатовимірних масивів використовувалася бібліотека NumPy, для обробки зображення – OpenCV, для візуалізації користувацького інтерфейсу - Streamlit.

Для глибинного навчання моделі пошуку та класифікації комах використовувалася нейронна мережа Faster RCNN R-50-FPN, а для взаємодії з моделлю - інтерфейс Detectron2. Всі обчислення проводились за допомогою хмарного сервісу Google Colab, який є інтерфейсом Jupyter Notebook та надає доступ одного GPU (NVidia Tesla K80). Це дозволило суттєво зменшити час навчання, але також призвело до низки обмежень. Першим обмеженням є час роботи на одній сесії, що становить 12 годин. Другим – обмежені обсяги оперативної пам'яті (RAM) та жорсткого диска (Disk). Експериментальним шляхом було встановлено, що для навчання з використанням Google Colab підходять згорткові мережі, що не перевищують 20 млн параметрів. У разі недотримання цієї умови навчання переривається.

Щоб зробити додаток доступним ззовні, з будь-якого пристрою користувача, було використано інстанс EC2 на AWS. Для швидкого і комфортного розгортування проекту викоритовувався Docker.

Висновок другий розділ

Отже, в результаті виконання другого розділу дипломної робити було проведено функціональний аналіз та побудовано дерево функцій системи. Наведено узагальнену технічну архітектуру системи, визначено підсистеми, модулі та зв’язки між компонентами системи. Сформовано архітектуру системи у нотації IDEF0 та побудовано бізнес-моделі програмного модуля за допомогою EPC.

Розроблено структуру та головну функціональність клієнт-серверного застосунку із використання мови програмування Python, бібліотек обробки зобрежень OpenCV та Pillow та бібліотеки Streamlit для візуалізації інтерфейсу користувача. Розгорнуто серверну частину додатку на хмарній платформі Amazon Web Services.

РОЗДІЛ 3. ТЕХНОЛОГІЧНІ ОСОБЛИВОСТІ РЕАЛІЗАЦІЇ ПРОГРАМНОГО МОДУЛЯ

3.1. Реалізація програмного модулю моніторингу комах-шкідників на жовтій липкій стрічці

3.1.1. Створення та експериментальні верифікації нейронної мережі для розпізнавання та класифікації комах

Оскільки Detectron2 приймає набор даних у форматі COCO, то для тренування нейромережі використовуємо зображення з відредагованого набору даних (див. розділ 1.9) та файли JSON, які містить деталі зображення: назва, координати обмежувальних боксів, складність виявлення (рис. 3.1).

```
<object>
        <name>MR</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
                <xmin>1572</xmin>
                <ymin>1204</ymin>
                <xmax>1680</xmax>
                <ymax>1335</ymax>
        </bndbox>
</object>
```

Рисунок 3.1 – Формат набору даних для Detectron2

Мережі та набори даних у Detectron2 можна перемикати та навчати уніфікованим способом, щоб точно налаштувати та порівняти різні мережі. У ході експериметальних верифікацій найкращу ефективність показав метод Faster R-CNN Inception-ResNet-v2 (табл. 3.1):

Таблиця 3.1. Порівняльння різних типів нейроних мереж у Detectron2

Тип нейромережі	Витрати пам’яті (GB)	Загальний час тренування (год)	Точність
mask_rcnn_R_50_FPN	4.4	3.4	0.89
faster_rcnn_X_101_FPN	3.7	2.8	0.92
efficientnet_b0	2.5	3.2	0.91
efficientnet_b3	2.8	2.5	0.87
faster_rcnn_R_50_FPN	2.1	2.3	0.95

Faster R-CNN забезпечує високу точність при виявлені дрібних об’єктів (шкідників). Модель виконує виявлення та класифікацію одночасно. Процес роботи із зображенням у Faster R-CNN складається з 4 частин:

Частина 1: Вилучення карти ознак зображення

Створюємо двовимірну матрицю – карту ознак – проходячи нейромережею уздовж вхідного зображення.

Частина 2: *Region Proposal Network (RPN)*

Використовуючи окрему невелику нейронну мережу – *RPN*, генеруємо на основі одержаної карти ознак гіпотези – визначення приблизних координат та наявність шкідників будь-якого класу.

Частина 3: шар *Region of Interest (RoI)*

Подаємо на вхід шару RoI області, запропоновані RPN (координати гіпотез). RoI зіставляє їх з картою ознак, отриманої в ч. 1. і прогнозує клас шкідника (завдання класифікації) і обмежувальні бокс (завдання регресії).

Частина 4: класифікація гіпотез

Роблячи «зріз» карти ознак, RoI шар подає його на вхід повнозв'язного шару для визначення конкретного класу та поправок до розмірів боксів.

Параметри навчання підбиралися експериментальним шляхом:

• Кількість епох навчання 1000 – значення даного коефіцієнта полягає у кількості повних проходжень тренувальних даних через нейромережу. За одну епоху відбувається кілька ітерацій, їх кількість визначається числом вхідних знімків (284 зображення жовтої стрічки) та розміром батча.

• розмір батча 512 – кількість знімків моделі, що подається на навчання за одну ітерацію.

• базова швидкість навчання 0,00025 – параметр, що визначає швидкість сходження алгоритму градієнтного спуску.

• кількість класів для тренування 3 – дві комахи-протагоністи (Macrolophus pygmaeus, Nesidiocoris tenuis) та один вид комахи-антоганіста (Trialeurodes vaporariorum).

Після проведення тренування на 1000 епохах, виводимо графіки зміни точності моделі (рис. 3.2) при зміні кількості епох за допомогою *TensorBoard* застосувавши команду `%tensorboard --logdir output`, де *output* – новостворена нейронна мережа.

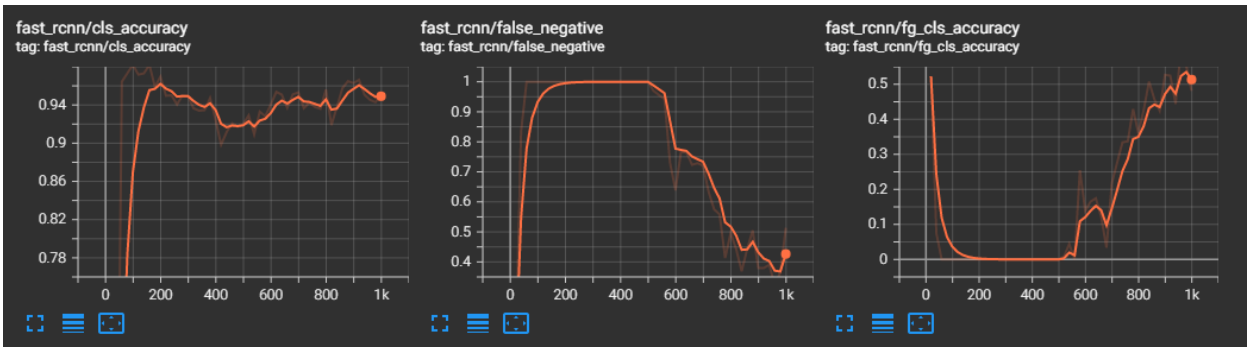


Рисунок 3.2 – Графіки зміни точності створеної моделі

З графіків можна побачити, що параметри нейронної мережі правильно підібрані, завдяки чому вдалося досягти 94% точності моделі. Випробуємо новостворену модель на перевірочному наборі даних (рис. 3.3):

Рисунок 3.3 – Тестування створеної нейромережі

Після успішного завершення навчання, архівуємо та зберігаємо вихідну папку, що містить остаточні ваги, в локальному сховищі за допомогою команди `!zip -r model.zip`. Додамо створену папку до майбутнього проєкту.

3.1.2. Реалізація основного модулю у Streamlit

Для початку потрібно додати всі бібліотеки, які будуть використовуватися в проекті:

os – надає безліч функцій для роботи з операційною системою;

cv2 – бібліотека комп'ютерного зору, яка призначена для аналізу, класифікації та обробки зображень;

numpy – для роботи з масивами;

streamlit – платформа веб-додатків, яка допомагає створювати та розробляти веб-програму на основі Python;

detectron2 – деякі поширені утиліти detectron2;

model_zoo – API моделі zoo для detectron2: набір функцій для створення загальних архітектур моделей, перерахованих у *model_zoo.md*.

Встановлюємо назву та логотип:

```
st.set_page_config("Insects Classifier", page_icon="🐞", layout="wide")
```

Отримуємо конфігурацію *detetron2* за замовчуванням. Всі розрахунки проводимо на центральному процесі, через відсутність відеокарти. Додаємо спеціальну конфігурацію проекту через *model_zoo* та *get_config_file*, для того щоб використовувати модель в основній бібліотеці *detetron2*, а саме *faster_rcnn_R_50_FPN_3x*. З файлу *model_final.pth*, який був описаний вище, вивантажуємо значення вагів нейронної мережі, попередньо вказавши шлях *model/output/*. Через змінну *NUM_CLASSES = 3*, вказуємо кількість класів для класифікації. Для уникнення помилок і можливих попереджень встановлюємо *'deprecation.showfileUploaderEncoding' = False* для вимкнення попереджень:

```
cfg = get_cfg()
cfg.MODEL.DEVICE = "cpu"
cfg.merge_from_file(model_zoo.get_config_file("COCO-
Detection/faster_rcnn_R_50_FPN_3x.yaml"))
cfg.MODEL.WEIGHTS = os.path.join("model/output/", "model_final.pth")
cfg.MODEL.ROI_HEADS.NUM_CLASSES = 3
st.set_option('deprecation.showfileUploaderEncoding', False)
```

Переходимо до створення інтерфейсу користувача. Встановлюємо заголовок на платформі *Streamlit.* Створюємо компонент для завантаження користувачем зображень, попередньо вказавши параметри для завантаження майбутнього файлу: опис, формат, опції. Вказуємо правила валідації: значення повинно бути в діапозоні від 0.1 до 0.9, значення за замовчуванням 0,5 і кроком 0,05. Записуємо в *SCORE_THRESH_TEST* введену користувачем задовільну точність розпізнавання комах. Генеруємо прогнозовані значення за допомогою функції *DefaultPredictor():*

```
st.write("# Insects Classification App")
files = st.file_uploader("Upload a Yellow tape photo", type=(["jpg",
"jpeg", "png"]), accept_multiple_files=True)

score_test = st.number_input("Score Treshhold (the less -> more areas
classified (can be less accurate))", min_value=0.1, max_value=0.9,
value=0.5, step=0.05)

cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = score_test
predictor = DefaultPredictor(cfg)
```

Переходимо для обробки зображення. Спочатку перетворимо зображення *files[0]* в одновимірний масив завдяки функції *frombuffer()*. Отримаємо масив , який складається з побітових символів від 0-255. *dtype=uint8* гарантує, що всі значення зображень є цілими числами. Використовуємо Функцію Python *cv2.imdecode()* для ефективного зчитування даних з кешу пам’яті та вказуємо за замовчуванням спосіб читання. Оскільки OpenCV має кольори в порядку BGR (синій/зелений/червоний), то потрібно перетворити зображення за допомогою функції *cv2.cvtColor()*. Модель завантажена і робить прогнози, як очікувалося, щоб отримати передбачення використовуємо *predictor():*

```
file_bytes = np.frombuffer(bytearray(files[0].read()), dtype=np.uint8)
im = cv2.imdecode(file_bytes, cv2.IMREAD_COLOR)
im = cv2.cvtColor(im, cv2.COLOR_BGR2RGB)
outputs = predictor(im)
```

Тепер зображення має три виміри: висоту, ширину та колір. Запис *im[:, :, ::-1]* означає зсув значень кольору з BGR на RGB. Використовуючи функцію *Visualizer* від *Streamlit* малюємо бокси та відсоток належності комахи до певного класу на зображенні взяті з каталогу *MetadataCatalog* змінної *bugs_metadata*. Виводимо зображення на екран користувача:

```
v = Visualizer(im[:, :, ::-1], metadata=bugs_metadata, scale=0.5)
out = v.draw_instance_predictions(outputs["instances"].to("cpu"))
st.image(out.get_image()[:, :, ::-1])
```

Підрахуємо загальну кількість комах на завантаженому зображенні. Для цього створюємо 3 змінні для кожного класу. Використовуючи вбудований автоматичний лічильник *enumerate* перебираємо елементи *list* і перевіряємо, щоб значення *['instances'].pred_classes* об'єкта *outputs* дорівнювало 1, 2 чи 3 відповідно. В результаті було одержано 3 списки різних класів комах-шкідників. У змінні записуємо довжину створених списків за допомогою вбудованої функції *len* і виводимо повідомлення на екран користувача.

```
idxofClass1 = [i for i, x in enumerate(list(outputs['instances'].pred_classes)) if x== 1]
idxofClass2 = [i for i, x in enumerate(list(outputs['instances'].pred_classes)) if x== 2]
idxofClass3 = [i for i, x in enumerate(list(outputs['instances'].pred_classes)) if x== 3]

class_1 += len((outputs['instances'].pred_classes[idxofClass1]).tolist())
class_2 += len((outputs['instances'].pred_classes[idxofClass2]).tolist())
class_3 += len((outputs['instances'].pred_classes[idxofClass3]).tolist())
```

3.1.3. Вивантаження програмного модулю на AWS

Щоб зробити створений додаток доступним зовні, для будь-якого користувача і з більшості пристроїм, було вирішено викласти його на хмарну платформу Amazon AWS. Для цього потрібно зарегеструватися на AWS як Root User і створити новий EC2.

Amazon EC2 Instance забезпечує простий і безпечний спосіб підключення до екземплярів Linux за допомогою Secure Shell (SSH). Щоб підключитися до створеного ком'ютера потрібен ssh-клієнт, тому вибираємо екземпляр Ubuntu (рис. 3.4), оскільки у Linux ssh-клієнт вбудований у

командний рядок, а у Windows потрібні додаткові налаштування. Після створення Instance, AWS запропонував зробити нову пару ключів (рис. 3.5).

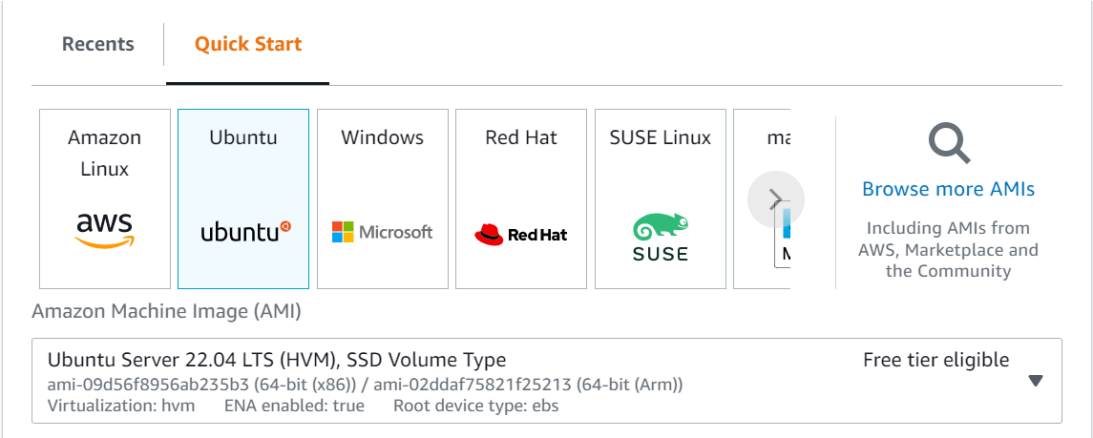


Рисунок 3.4 – Створення EC2 Instance на Ubuntu

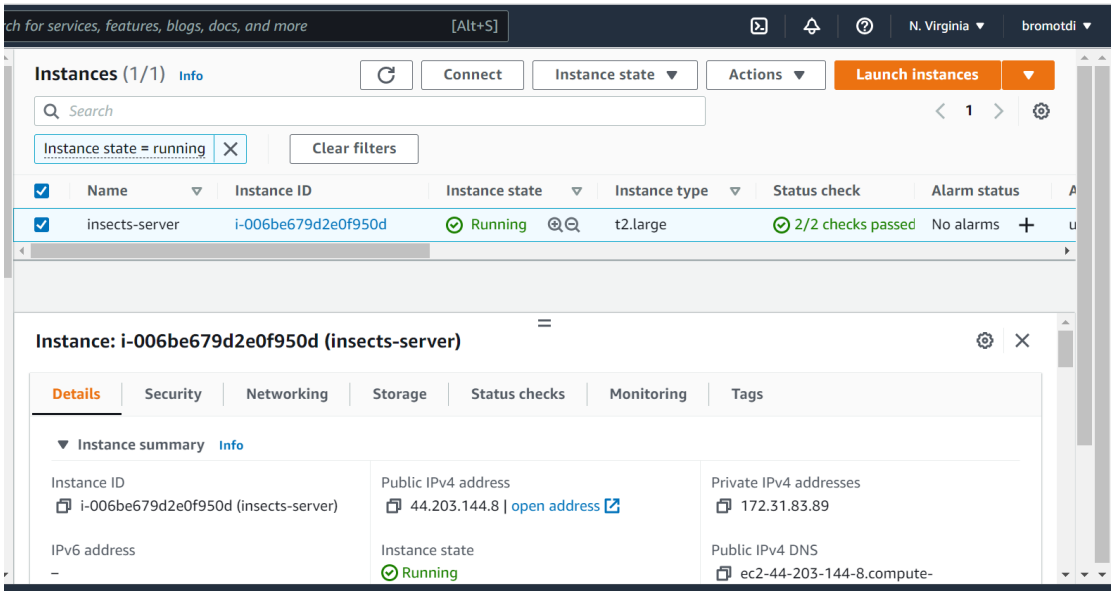


Рисунок 3.5 – Створення пари ключів для ssh-клієнта

На робочому столі є ключ доступу до сервера, за допомогою якого можна зайти на створений EC2 Instance використовуючи ssh (рис. 3.6).

Рисунок 3.6 – Папки проекту

Підключимося до серверу: у вікні термінала використовуємо команду ssh для підключення до екземпляра. Вказуємо шлях та ім’я файлу

приватного ключа (insects.pem), ім’я користувача для екземпляра та публічне ім’я DNS. Вводимо наступну команду (рис. 3.7):

```
sudo ssh -i insects.pem ubuntu@ec2-35-175-197-10.compute1.amazonaws.com
```

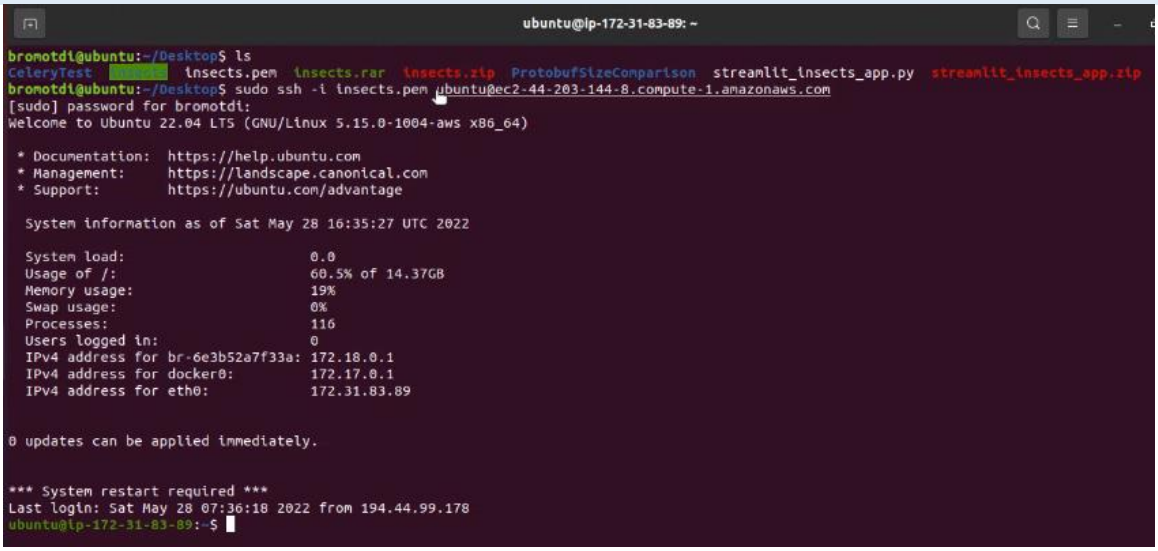


Рисунок 3.7 – Підключення до серверу EC2 Instance через ssh

Перейшли на віддалений комп'ютер, на якому потрібно створити папку insects:

```
mkdir insects
```

Перенесемо архів із створеним проектом у цю папку. Для передачі файлу між локальним комп’ютером і відаленим комп'ютером Ubunta є використання scp. Перед цим потрібно визначити розташування вихідного файлу на комп’ютері та шлях призначення на EC2. Вводимо наступну команду зі звичайного комп’ютера:

```
sudo scp -i insects.pem  insects.zip
ubuntu@ec2-35-175-197-10.compute-
1.amazonaws.com:/home/ubuntu/insec
```

де ім’я файлу приватного ключа – *insects.pem*,

файл, який потрібно передати – *insects.zip*,

ім’я користувача для екземпляра – *ubuntu,*

ім’я DNS екземпляра – *ec2-35-175-197-10.compute-1.amazonaws.com,*

шлях до папки – */home/ubuntu/insects.*

Оновлюємо систему:

```
sudo apt update
sudo apt upgrade -y
```

Розпаковуємо проект на відаленному комп'ютері:

```
sudo apt install unzip
unzip insects.zip
```

Для швидкого запуску проекту необхідно встановити Docker и docker-compose (рис. 3.8), Для цього використовуємо команди описані у [25] і [26] відповідно.

Рисунок 3.8 – Вміст файлу docker-compose.yml

Створимо файл під назвою Dockerfile (рис. 3.9):

```
touch Dockerfile
```

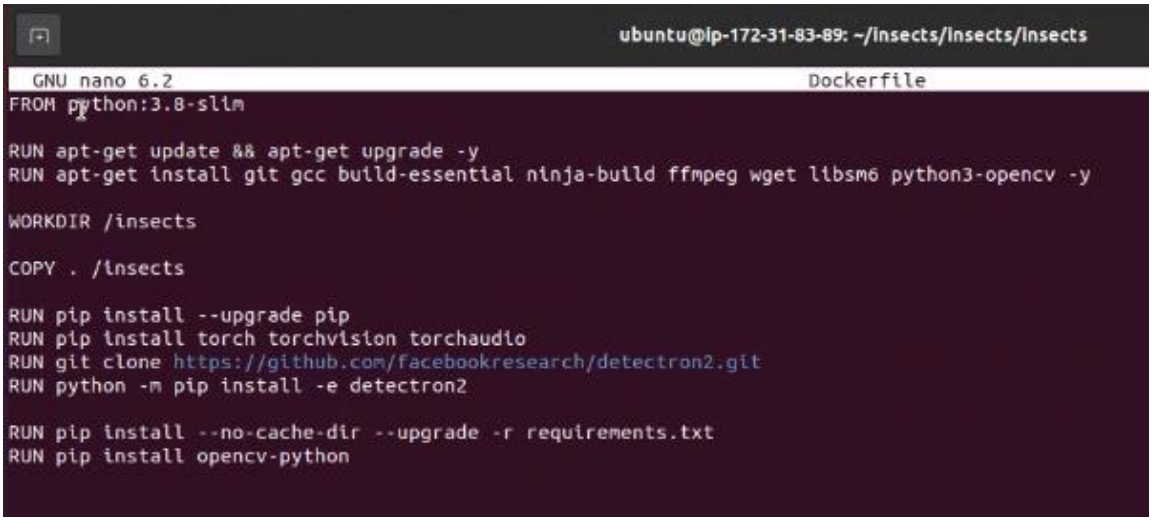


Рисунок 3.9 – Вміст файлу Dockerfile

Запустили проект за допомогою команди (рис. 3.10):

```
sudo docker-compose up -d --build
```

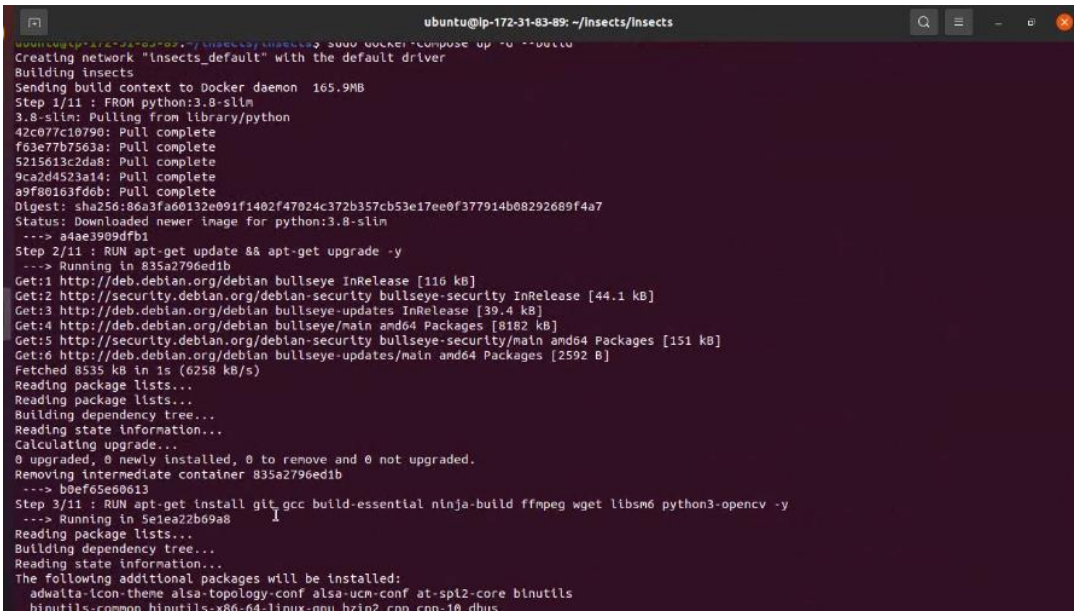


Рисунок 3.10 – Успішно створений docker-контейнер

3.2. Інструктивний матеріал з експлуатації моніторингу комах-шкідників на жовтій липкій стрічці

3.2.1. Інструктивний матеріал з експлуатації модуля моніторингу комах-шкідників для користувача

Завдяки використанню платформи AWS та EC2, було максимально спрощення використання програмного модуля для потенційних користувачів (фермерів, екологів, агрономів), які повинні просто зайти на веб-сайт http://44.203.144.8/ (рис. 3.11).

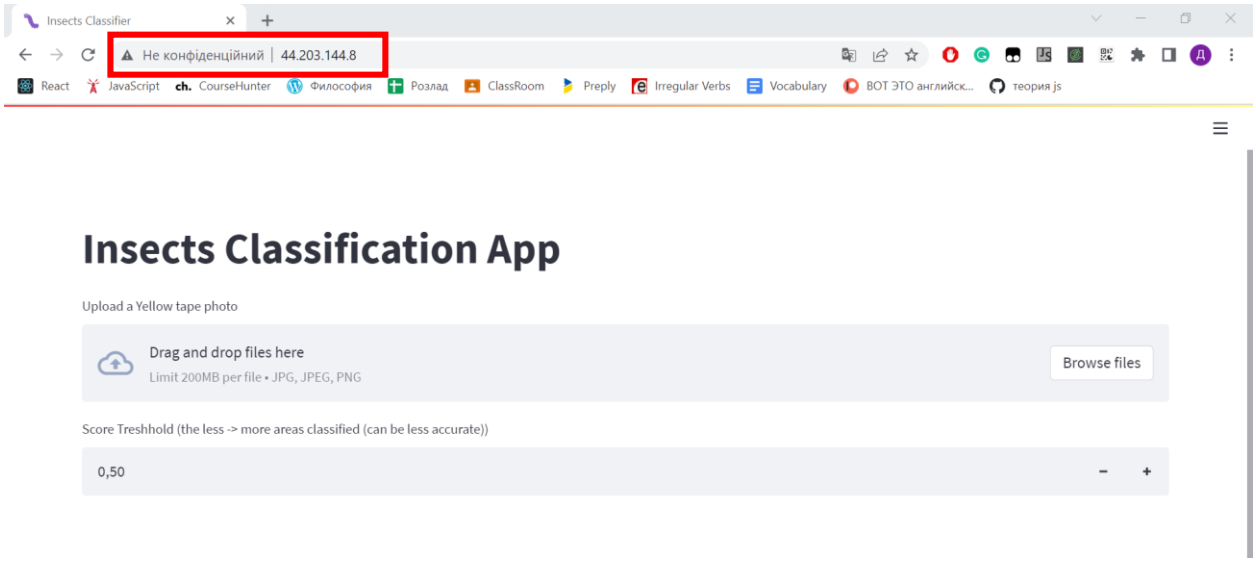


Рисунок 3.11 – Результат запуску головної сторінки

Для того, щоб здійснити моніторинг комах-шкідників необхідно натиснути на кнопку «Browse files». Після чого вибрати декілька зображень (рис. 3.12):

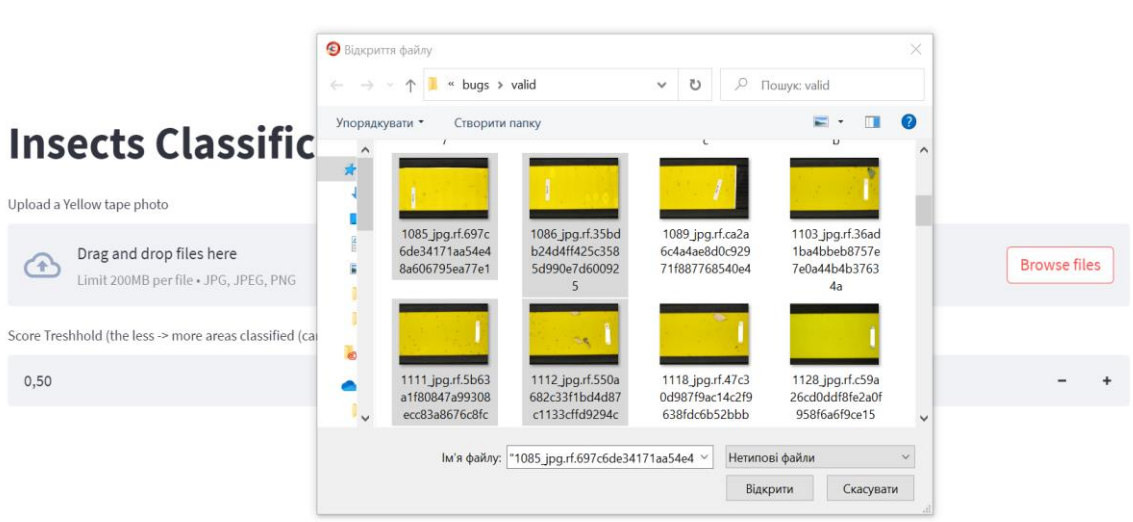


Рисунок 3.12 – Завантаження зображень у додаток

Можна побачити, що зображення завантажилися на сайт. Тепер потрібно встановити точність розпізнавання шкідниківна. Чим ближче показник до 0, то більше модель бачить комах, але більше не значить точніше – необхідно шукати золоту середину для різних картинок (рис. 3.13).

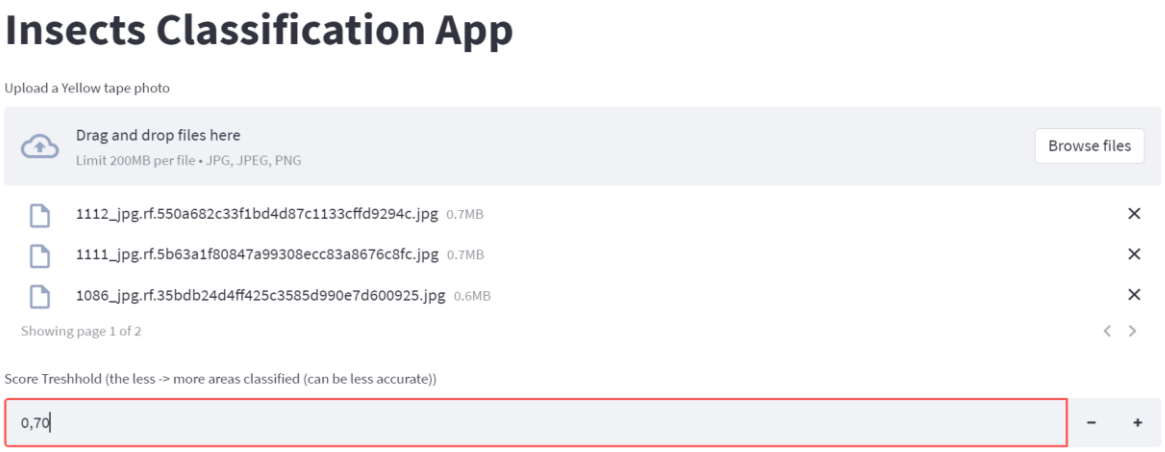


Рисунок 3.13 – Встановлення задовільної точності розпізнавання

Далі користувач може побачити власні зображення з прогнозуваними комахами-шкідниками виділеними в обмежуючі бокси з відсотком розпізнавання і класифікацією на різні класи (рис. 3.14).

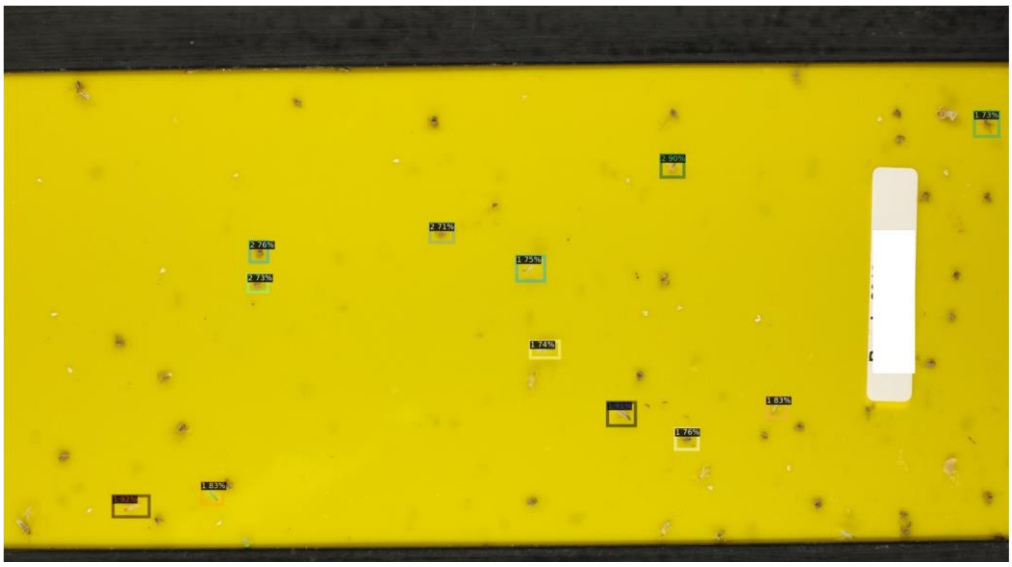


Рисунок 3.14 – Виконання прогнозування

Після виведення усіх зображень, програмний модуль підрахує загальну кількість комах-шкідників різних 3-х класів на усіх зображеннях і виведе загальну суму (рис. 3.15)

Number of *Macrolophus pygmaeus* (**1** class on image): `16 pcs.`

Number of *Trialeurodes vaporariorum (Whitefly)* (**2** class on image): `20 pcs.`

Рисунок 3.15 – Підрахунок загальної кількості шкідників

3.3. Тестування та аналіз модулю моніторинга комах-шкідників

Розглянемо можливі випадки, які можуть трапитися під час моніторингу комах-шкідників на липких стрічках. Було вирішено виділити наступні випадки:

1. *Виявлення класу комах-шкідників Nesidiocoris tenuis на жовтій стрічці у пасці* (представлено у таблиці 3.2)

Таблиця 3.2 План тестування «*Виявлення класу комах-шкідників Nesidiocoris tenuis на жовтій стрічці у пасці*»

Мета тесту	Перевірити чи коректно написаний алгоритм виявлення комах-шкідників.
Початковий стан системи	Лічильники кількості комах класу Nesidiocoris tenuis вставлені в 0.

Вхідні дані	Виявлена одна із комах Nesidiocoris tenuis
Схема проведення тесту	Вибрати і завантажити зображення, яке містить клас Nesidiocoris tenuis і подивитись чи збільшиться показник лічильника.
Очікуваний результат	При виявленні однієї із комах класу Nesidcoris tenuis лічильник збільшується на 1.
Стан системи після проведення випробувань	Результати класифікованих та підрахованих комах виводяться на екран користувача.

Результат: При виявленні комахи-шкідника класу Nesidiocoris tenuis показник відповідного лічильника збільшується.

2. *Перевірка коректності класифікації комах у випадку затемнення зображення через вимкнення електроенергії у теплиці.* (представлено у таблиці 3.3)

Якщо система МКШ підключена для стаціонарних камер безпосередньо у теплицях і здійснюється щоденний моніторинг, то варто перевірити систему на випадок якщо декілька або всі зображення у завантаженому наборі даних містять фото зроблені у момент вимкнення освітлення.

Таблиця 3.3 План тестування «*Перевірка коректності класифікації комах у випадку затемнення зображення через вимкнення електроенергії*»

Мета тесту	Перевірка коректності написаного алгоритму із визначенням комах-шкідників при затемненому зображенні через брак електроенергії у теплиці
Початковий стан	Увімкнена та правильно встановлена стаціонарна камера у теплиці, світло вимкнуте
Вхідні дані	Вдало завантажено затемнене зображення через брак світла
Схема проведення тесту	Користувач завантажує затемнене зображення для перевірки системи у разі вимкнення електроенергії.

Очікуваний результат	Якщо зображення чорне – система не розпізнає об’єкти. Якщо зображення злегка затемнене , то комахи розпізнаються та підраховуються коректно у більшості випадків.
Стан системи після проведення випробувань	Результати виводяться на екран користувача.

Результат: точність розпізнавання недостатня при затемнених умовах.

3. *Перевірка правильності розпізнавання комах при зміні параметру точності* (представлено у таблиці 3.4)

Таблиця 3.4 План тестування *«Перевірка правильності розпізнавання комах при зміні параметру точності*»

Мета тесту	Перевірка коректності написаного алгоритму із розпізнаванням комах-шкідників під час моніторингу зі зміною параметрів точності.
Початковий стан системи	Користувач завантажив зображення і встановив початковий параметр точності розпізнавання.
Вхідні дані	Оброблене зображення зі встановленою точністю.
Схема проведення тесту	Користувач переглядає оброблене зображення зі вказаним відсотком точності і під час цього змінює точність моделі.
Очікуваний результат	При зміні параметру точності, система почне розпізнавати комах-шкідників на введеному зображені, але вже з новою точністю.
Стан системи після проведення випробувань	Результати роботи програми виведено на екран користувача.

Результат: при зміні параметру точності розпізнавання продовжилося зі зміненим параметром.

4. *Перевірка розпізнавання комах-шкідників у випадку потрапляння зайвих предметів на жовту стрічку в пасці* (представлено у таблиці 3.5)

Якщо зображення комах зроблене на відкритих земельних ділянках, або у пастках, які не закриваються від зовнішнього середовища, досить часто можливе потрапляння різних предметів, які не відносяться до комах, наприклад листя з рослин, пелюстки з квітів, краплі води, тощо. Потрібно перевірити, як система буде реагувати на різні сторонні предмети.

Таблиця 3.5 План тестування «*Перевірка розпізнавання комах-шкідників у випадку потрапляння зайвих предметів на жовту стрічку в пасці*»

Мета тесту	Перевірити чи коректно написаний алгоритм виявлення комах-шкідників.
Початковий стан системи	Лічильники кількості комах вставлені в 0.
Вхідні дані	Виявлені комахи-шкідники різних класів
Схема проведення тесту	Вибрати і завантажити зображення, яке містить сторонні предмет (листя, великі комахи) і подивитись чи збільшиться показник лічильника якогось класу комах.
Очікуваний результат	Сторонні предмети будуть ігноруватися системою, значення лічильника змінюватися не буде.
Стан системи після проведення випробувань	Результати класифікованих та підрахованих комах виводяться на екран користувача.

Результат: сторонні предмети ігноруються, показники лічильників не збільшуються.

5. *Перевірка правильності обробки декілька завантажених зображень користувача для розпізнавання комах* (представлено у таблиці 3.6)

Таблиця 3.6 План тестування *«Перевірка правильності обробки декілька завантажених зображень користувача для розпізнавання комах*»

Мета тесту	Перевірка коректності написаного алгоритму під час завантаження користувачем кількох зображень

Початковий стан системи	Користувач завантажив 5 зображень і встановив початковий параметр точності розпізнавання.
Вхідні дані	Оброблені зображення зі встановленою точністю.
Схема проведення тесту	Користувач завантажує 5 зображень і встановлює задовільну точність розпізнавання.
Очікуваний результат	Під час завантаження кількох зображень система обробить кожне зображення і виведе загальну кількість комах різних класів на всіх 5 зображеннях.
Стан системи після проведення випробувань	Результати роботи програми виведено на екран користувача.

Результат: при завантажені декількох зображень система обробила кожне зображення окремо і вивела на екран загальну кількість комах-шкідників на даних зображеннях.

Висновок до третього розділу

В останньому розділі кваліфікаційної роботи розглядалися функції, написані для програмного модулю. Детально роз’яснено принцип роботи їх алгоритмів, отриманих результатів, а також наведений їх лістинг. Пояснено використання функцій сторонніх бібліотек та їх синтаксис.

Також написана інструкція як для звичайного користувача, яка включає детальний розбір користувацького інтерфейсу, так і для розробника (адміністратора), який намагається дослідити даний проєкт чи є співробітником.

Проведене тестування програмного модулю на прикладі типових винятків в сфері сільського господарства, таких як-от вимкнення електроенергії у теплиці чи потрапляння сторонніх предметів на липку стрічку.

ВИСНОВОК

У результаті виконання дипломної роботи було отримано програмний модуль для розпізнавання і класифікації комах-шкідників на жовтій липкій стрічці для запобігання пошкодження сільськогосподарських культур.

Проведено аналітичний огляд кваліфікаційної роботи. Проаналізовано наявні методи, рішення та результати, що досягнуто сьогодні у сільськогосподарській сфері. Розглядалися існуючі рішення щодо проблеми розпізнавання великої кількості дрібних об‘єктів на їх підрахунок.

Проведено функціональний аналіз та визначено вимоги до програмного модулю. Розроблено архітектуру застосунку, бізнес-логіку, визначено підсистеми, модулі та зв’язки між компонентами системи. Розроблено додаток із використанням мов програмування Python та бібліотеки Streamlit, створено нейронну мережу за допомогою моделі Faster R-CNN R50 через Detectron2 Pytorch в Google Colab. Модуль викладений на платформу AWS для зовнішнього використання кожним фермером чи екологом у світі. Створено документацію для користувача та розробника.

Отже, цей програмний модуль можна буде в подальшому використовувати для великих сільськогосподарських ферм. Користувачі зможуть приєднатись до процесу навчання класифікаційної моделі для покращення їх точності визначення кома-шкідників. З технічної точки зору цей застосунок можна покращити шляхом оптимізації і рефакторингу коду, що в подальшому підвищить швидкість повернення відповідей на запити користувачів.

Точність навченої нейронної мережі становить 94%, що є чудовим результатом для досить невдалого початкового набору даних для тренування. При збільшенні кількості користувачів, які будуть використовувати створений додаток точність моделей може підвищитись.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ebrahimi, M.A., Khoshtaghaza, M.H., Minaei, S., Jamshidi, B., 2017. Vision-based pest detection based on SVM classification method. Comput. Electron. Agric. 137, 52–58. [Електронний ресурс]. Режим доступу: http://refhub.elsevier.com/S1574-9541(21)00251-X/rf0085

2. Gutierrez, A., Ansuategi, A., Susperregi, L., Tubio, C., Rankic, I., Lenza, L., 2019. A benchmarking of learning strategies for Pest detection and identification on tomato plants for autonomous scouting robots using internal databases. J. Sens. 2019. [Електронний ресурс]. Режим доступу: http://refhub.elsevier.com/S1574-9541(21)00251-X/rf0125

3. Li, W., Chen, P., Wang, B., Xie, C., 2019d. Automatic localization and count of agricultural crop pests based on an improved deep learning pipeline. Sci. Rep. 9 [Електронний ресурс]. Режим доступу: http://refhub.elsevier.com/S1574-9541(21)00251-X/rf0235

4. Xia, C., Chon, T.S., Ren, Z., Lee, J.M., 2015. Automatic identification and counting of small size pests in greenhouse conditions with low computational cost. Ecol. Inform. 29, 139–146. [Електронний ресурс]. Режим доступу: http://refhub.elsevier.com/S1574-9541(21)00251-X/rf0500

5. Xie, C., Wang, R., Zhang, J., Chen, P., Dong, W., Li, R., Chen, T., Chen, H., 2018. Multi- level learning features for automatiic classification of field crop pests. Comput.Electron. Agric. 152, 233–241. [Електронний ресурс]. Режим доступу: http://refhub.elsevier.com/S1574-9541(21)00251-X/rf0510

6. Li, R., Wang, R., Zhang, J., Xie, C., Liu, L., Wang, F., Chen, H., Chen, T., Hu, H., Jia, X., Hu, M., Zhou, M., Li, D., Liu, W., 2019b. An effective data augmentation strategy for CNN-based Pest localization and recognition in the field. IEEE Access 7,160274–160283. [Електронний ресурс]. Режим доступу: http://refhub.elsevier.com/S1574-9541(21)00251-X/rf0225

7. Chudzik, P., Mitchell, A., Alkaseem, M., Wu, Y., Fang, S., Hudaib, T., Pearson, S., Al- Diri, B., 2020. Mobile real-time grasshopper detection and data

aggregation framework. Sci. Rep. 10, 1150 [Електронний ресурс]. Режим доступу: http://refhub.elsevier.com/S1574-9541(21)00251-X/rf0055

8. Li, D., Wang, R., Xie, C., Liu, L., Zhang, J., Li, R., Wang, F., Zhou, M., Liu, W., 2020a. A recognition method for Rice Plant diseases and pests video detection based on deep convolutional neural network. Sensors 20, 578. [Електронний ресурс]. Режим доступу: http://refhub.elsevier.com/S1574-9541(21)00251-X/rf0210

9. Sun, Y., Liu, X., Yuan, M., Ren, L., Wang, J., Chen, Z., 2018. Automatic in-trap pest detection using deep learning for pheromone-based Dendroctonus valens monitoring. Biosyst. Eng. 176, 140–150. [Електронний ресурс]. Режим доступу: http://refhub.elsevier.com/S1574-9541(21)00251-X/rf0400

10. Wang, Q.J., Zhang, S.Y., Dong, S.F., Zhang, G.-C., Yang, J., Li, R., Wang, H.-Q., 2020 c.Pest24: a large-scale very small object data set of agricultural pests for multi-target detection. Comput. Electron. Agric. 175. [Електронний ресурс]. Режим доступу: http://refhub.elsevier.com/S1574-9541(21)00251-X/rf0460

11. Ünlü L, Akdemir B, Ögür E, Şahin İ (2019) Remote monitoring of European Grapevine Moth, Lobesia botrana (Lepidoptera: Tortricidae) population using camera-based pheromone traps in vineyards. [Електронний ресурс]. Режим доступу: http://www.agrifoodscience.com/index.php/TURJAF/article/view/2382

12. Holguin GA, Lehman BL, Hull LA, Jones VP, Park J (2010) Electronic traps for automated monitoring of insect populations. IFAC Proc Vol 43(26):49–54. [Електронний ресурс]. Режим доступу: https://doi.org/10.3182/20101%20206-3-JP-3009.00008

13. Fischnaller S, Parth M, Messner M, Stocker R, Kerschbamer C, ReyesDominguez Y, Janik K (2017) Occurrence of diferent Cacopsylla species in apple orchards in South Tyrol (Italy) and detection of apple proliferation phytoplasma in Cacopsylla melanoneura and Cacopsylla picta. Cicadina 17:37–51

14. Remote Insects Trap Monitoring System Using Deep Learning Framework and IoT___[Електронний ресурс]. Режим доступу: https://www.mdpi.com/1424-8220/20/18/5280

15. Automated Insect Pest Monitoring and Analysis Using Imaging and Environmental Sensor Network [Електронний ресурс]. Режим доступу: https://www.intelligentagri.com.tw/en/xmdoc/cont?xsmsid=0J170506304287007902&sid=0J304397444351531469

16. G. J. Messelink, J. Bennison, O. Alomar, B. L. Ingegno, L. Tavella, L. Shipp, E. Palevsky, and F. L. Wackers. Approaches to conserving natural enemy populations in green-house crops: Current methods and future prospects, 2014.

17. Sautter & Stepper GmbH, “Macrolophus pygmaeus,” [Електронний ресурс]. Режим доступу: https://www.nuetzlinge.de/produkte/unter-glas/macrolophus-pygmaeus/

18. Biological Services, “Nesidiocoris,” [Електронний ресурс]. Режим доступу: https://biologicalservices.com.au/products/nesidiocoris-28.html

19. Sautter & Stepper GmbH, “Weiße fliege,” [Електронний ресурс]. Режим доступу: https://www.nuetzlinge.de/produkte/unter-glas/schaedlinge/weisse-fliege/

20. Roboflow: Overview/Build Better Computer Vision Models Faster [Електронний ресурс]. Режим доступу: https://blog.roboflow.com/getting-started-with-roboflow/

21. Методологія ARIS [Електронний ресурс]. Режим доступу: http://um.co.ua/4/4-15/4-154716.html

22. Типы инстансов Amazon EC2 [Електронний ресурс]. Режим доступу: https://aws.amazon.com/ru/ec2/instance-types/#Memory_Optimized

23. Справочник по работе с Google Colaboratory. [Електронний ресурс]. Режим доступу: https://www.geeksforgeeks.org/how-to-use-google-colab

24. Python Pillow - Overview [Електронний ресурс] / Режим доступу: https://www.tutorialspoint.com/python_pillow/python_pillow_overview.htm

25. Installing and Using Docker on Ubuntu 20.04 [Електронний ресурс]. Режим доступу: https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-20-04-ru

26. Installing and Using Docker Compose on Ubuntu 20.04 [Електронний ресурс] / Режим доступу:https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-compose-on-ubuntu-20-04-ru

ДОДАТКИ:

```
import os
import cv2
import numpy as np #для роботи з масивами
import streamlit as st #платформа веб-додатків, яка допомагає нам створювати та розробляти веб-програми на основі Python
# import some common detectron2 utilities
from detectron2.config import get_cfg
from detectron2.data import MetadataCatalog
from detectron2.engine import DefaultPredictor
from detectron2.utils.visualizer import Visualizer

from detectron2 import model_zoo #API моделі Zoo для Detectron2: набір функцій для створення загальних архітектур моделей, перерахованих у MODEL_ZOO.md
st.set_page_config("Insects Classifier", page_icon="🐛", layout="wide")
# Set title and logo
cfg = get_cfg() # отримуємо конфігурацію detetron2 за замовчуванням
cfg.MODEL.DEVICE = "cpu"
# add project-specific config (e.g., TensorMask) here if you're not running a model in detectron2's core library
cfg.merge_from_file(model_zoo.get_config_file("COCO-Detection/faster_rcnn_R_50_FPN_3x.yaml")) #завантажує значення з файлу
cfg.MODEL.WEIGHTS = os.path.join("model/output/", "model_final.pth")  # path to the model we just trained
cfg.MODEL.ROI_HEADS.NUM_CLASSES = 3
# Default Configuration# Disabling warning
st.set_option('deprecation.showfileUploaderEncoding', False)

st.write("# Insects Classification App")

files = st.file_uploader("Upload a Yellow tape photo", type=(["jpg", "jpeg", "png"]), accept_multiple_files=True) #вказуємо параметри для завантаження майбутнього файлу: опис, формат, опції

score_test = st.number_input("Score Treshhold (the less -> more areas classified (can be less accurate))", #вказуємо правила валідацію
min_value=0.1, max_value=0.9, value=0.5, step=0.05)
# set threshold for this model
#cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = score_test  # set a custom testing threshold
cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST  = 0.10 if score_test < 0.40 else score_test - 0.30

predictor = DefaultPredictor(cfg) #to get predictions
```

```
bugs_metadata = MetadataCatalog.get("bugs_dataset_test_1") #ді

if files:
    class_1 = 0
    class_2 = 0
    class_3 = 0

    st.write("**Predicted classes image**")

    for file in files:
        file_bytes = np.frombuffer(bytearray(file.read()),
dtype=np.uint8) #функція інтерпретує буфер як одновимірний масив

        im = cv2.imdecode(file_bytes, cv2.IMREAD_COLOR)
        im = cv2.cvtColor(im, cv2.COLOR_BGR2RGB)
        # Модель завантажена і робить прогнози, як очікувалося, я
використовую наступний код, щоб отримати передбачення:
        outputs = predictor(im)
        print(outputs)
        # можемо використовувати `Visualizer`, щоб намалювати
передбачення на зображенні.
        v = Visualizer(im[:, :, ::-1], metadata=bugs_metadata,
scale=0.5)
        out =
v.draw_instance_predictions(outputs["instances"].to("cpu"))

        st.image(out.get_image()[:, :, ::-1])  # Show predicted
visualization

        idxofClass1 = [i for i, x in
enumerate(list(outputs['instances'].pred_classes)) if x == 1]
        idxofClass2 = [i for i, x in
enumerate(list(outputs['instances'].pred_classes)) if x == 2]
        idxofClass3 = [i for i, x in
enumerate(list(outputs['instances'].pred_classes)) if x == 3]

        class_1 +=
len((outputs['instances'].pred_classes[idxofClass1]).tolist())
        class_2 +=
len((outputs['instances'].pred_classes[idxofClass2]).tolist())
        class_3 +=
len((outputs['instances'].pred_classes[idxofClass3]).tolist())

    if class_1 != 0:
        st.write(
            f"Number of *Macrolophus pygmaeus* (**1** class on image):
```{class_1} pcs.```")

    if class_2 != 0:
        st.write(
            f"Number of *Trialeurodes vaporariorum (Whitefly)* (**2**
class on image): ```{class_2} pcs.```")

    if class_3 != 0:
        st.write(
            f"Number of *Nesidiocoris tenuis* (**3** class on image):
```{class_3} pcs.```")
```