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

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка нейронної мережі у сільськогосподарській діяльності»

на здобуття освітнього ступеня бакалавра (магістра)
зі спеціальності 126 Інформаційні системи та технології
(код, найменування спеціальності)

освітньо-професійної програми Інформаційні системи та технології
(назва)

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

_____ Віталій ЧЕРНИШ
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: здобувач(ка) вищої освіти гр. ІСД-42
Віталій ЧЕРНИШ
Ім'я, ПРІЗВИЩЕ

Керівник:
науковий ступінь, вчене звання
PhD. Валентина ДАНИЛЬЧЕНКО
Ім'я, ПРІЗВИЩЕ

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

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра Інженерії програмного забезпечення автоматизованих систем
Ступінь вищої освіти бакалавр
Спеціальність Інформаційні системи та технології
Освітньо-професійна програма Інформаційні системи та технології

ЗАТВЕРДЖУЮ
Завідувач кафедрою ІПЗАС
_____Каміла СТОРЧАК
«____» _____2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
Черниша Віталія Володимировича
(прізвище, ім'я, по батькові здобувача)

1.Тема кваліфікаційної роботи: Розробка нейронної мережі у сільськогосподарській діяльності

керівник кваліфікаційної роботи Валентина ДАНИЛЬЧЕНКО *PhD*
(Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» лютого 2024 р. № 36

2. Строк подання кваліфікаційної роботи «31» травня 2024 р.

3. Вихідні дані до кваліфікаційної роботи:
1. Науково-технічна література з теми бакалаврської роботи.
2. Принципи функціонування нейронних мереж.
3. Основні аспекти застосування штучного інтелекту в сільськогосподарській діяльності.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Основи нейронних мереж: визначення, типи та принципи роботи.
2. Інструменти та прийоми розробки нейронних мереж
3. Розробка алгоритму навчання та тестування нейронної мережі.

5. Ілюстративний матеріал: *презентація*

6. Дата видачі завдання: «27» лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	27.02-05.03.2024	
2	Обгрунтування актуальності роботи	06.03-11.03.2024	
3	Аналіз основних моделей нейронних мереж	12.03-27.03.2024	
4	Інструменти та прийоми розробки нейронних мере	28.03-10.04.2024	
5	Розробка нейронної мережі у сільськогосподарській діяльності	11.04-15.05.2024	
7	Оформлення роботи: вступ, висновки, реферат	16.05-22.05.2024	
8	Розробка демонстраційних матеріалів	23.05-24.05.2024	

Здобувач(ка) вищої освіти _____ Віталій ЧЕРНИШ

(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник

кваліфікаційної роботи _____ Валентина ДАНИЛЬЧЕНКО

(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавр: 54 стор., 62 рис., 33 джерела.

Мета роботи – підвищення ефективності сільськогосподарських процесів за рахунок впровадження нейронної мережі, що оптимізує ключові операції в аграрному секторі.

Об'єкт дослідження – Зображення сільськогосподарських культур.

Предмет дослідження – Розробка прототипів нейронних мереж для автоматичного розпізнавання та класифікації сільськогосподарських культур на основі зображень.

Короткий зміст роботи: У роботі проаналізовано можливості використання нейронних мереж у сільському господарстві, зокрема на прикладі виявлення соняшників на фотографіях за допомогою моделі Faster R-CNN. Проведено збір та підготовку різноманітних даних про врожайність, ґрунт, кліматичні умови та інші фактори. Розроблено та протестовано адаптовану модель нейронної мережі, яка демонструє високу точність та ефективність у розпізнаванні соняшників. Навчання та тестування моделі показало високу точність (0.93) та повноту (0.90), а також середню точність (mAP) 0.91, що підтверджує переваги використання нейронних мереж у порівнянні з традиційними методами аналізу даних.

КЛЮЧОВІ СЛОВА: ВИЯВЛЕННЯ ОБ'ЄКТІВ, ВІДСТЕЖЕННЯ ОБ'ЄКТІВ, ТОЧНЕ ЗЕМЛЕРОБСТВО, СІЛЬСЬКЕ ГОСПОДАРСТВО, КОМП'ЮТЕРНИЙ ЗІР, АНАЛІЗ ЗОБРАЖЕНЬ, АНАЛІЗ ВІДЕО, ГЛИБОКЕ НАВЧАННЯ, АЛГОРИТМИ, МОНІТОРИНГ, ПІДРАХУНОК, РОБОТИЗАЦІЯ, НАБОРИ ДАНИХ, МАРКОВАНІ ДАНІ, БІОФІЗИЧНЕ СЕРЕДОВИЩЕ, ОБМЕЖЕННЯ, МАЙБУТНІ НАПРЯМКИ.

ABSTRACT

Text part of the bachelor level qualification work: 54 pages, 62 pictures, 0 table, 33 sources.

The purpose of the work - is to increase the efficiency of agricultural processes by implementing a neural network that optimises key operations in the agricultural sector.

Object of research are images of agricultural crops.

Subject of research is the development of prototypes of neural networks for automatic recognition and classification of crops based on images.

Summary of the work: The work analyses the possibilities of using neural networks in agriculture, in particular, on the example of sunflower detection in photos using the Faster R-CNN model. Various data on yield, soil, climatic conditions and other factors were collected and prepared. An adapted neural network model was developed and tested, which demonstrates high accuracy and efficiency in sunflower recognition. Training and testing of the model showed high accuracy (0.93) and completeness (0.90), as well as a mean accuracy (mAP) of 0.91, which confirms the advantages of using neural networks compared to traditional data analysis methods.

KEYWORDS: OBJECT DETECTION, OBJECT TRACKING, PRECISION AGRICULTURE, AGRICULTURE, COMPUTER VISION, IMAGE ANALYSIS, VIDEO ANALYSIS, DEEP LEARNING, ALGORITHMS, MONITORING, COUNTING, ROBOTICS, DATASETS, LABELLED DATA, BIOPHYSICAL ENVIRONMENT, LIMITATIONS, FUTURE DIRECTIONS.

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

ПОДАННЯ
ГОЛОВІ ЕКЗАМЕНАЦІЙНОЇ КОМІСІЇ
ЩОДО ЗАХИСТУ КВАЛІФІКАЦІЙНОЇ РОБОТИ
на здобуття освітнього ступеня бакалавра

Направляється здобувач(ка) Черниш В.В. до захисту кваліфікаційної роботи
(прізвище та ініціали)
за спеціальністю 126 Інформаційні системи та технології
(код, найменування спеціальності)
освітньо-професійної програми Інформаційні системи та технології
(назва)
на тему: «Розробка нейронної мережі у сільськогосподарській діяльності».

Кваліфікаційна робота і рецензія додаються.

Директор ННІ _____ _____
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Висновок керівника кваліфікаційної роботи

Здобувач(ка)_____
_____
_____
_____
_____

Все це дозволяє оцінити виконану кваліфікаційну роботу здобувача(ки)_____на оцінку «_____» та присвоїти йому(їй) кваліфікацію _____.

Керівник кваліфікаційної роботи _____ _____
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

«____» _____ 20___ року

Висновок кафедри про кваліфікаційну роботу

Кваліфікаційна робота розглянута. Здобувач(ка) прізвище та ініціали допускається до захисту даної роботи в Екзаменаційній комісії.

Завідувач кафедрою _____ _____ _____
(назва) *(підпис)* *(Ім'я, ПРІЗВИЩЕ)*

ВІДГУК РЕЦЕНЗЕНТА
на кваліфікаційну роботу
на здобуття освітнього ступеня бакалавра

здобувача(ки) вищої освіти Черниша Віталія Володимировича
(прізвище, ім'я, по батькові)
на тему: «Розробка нейронної мережі у сільськогосподарській діяльності»

Актуальність.
Кваліфікаційна робота Черниша Віталія Володимировича на тему «Розробка нейронної мережі у сільськогосподарській діяльності» є надзвичайно актуальною. Проблема підвищення ефективності сільськогосподарських процесів залишається однією з ключових у сучасному світі, де зростає потреба в забезпеченні продовольчої безпеки. Використання технологій штучного інтелекту, зокрема нейронних мереж, може значно покращити точність та швидкість ідентифікації сільськогосподарських культур. Впровадження таких інтелектуальних систем здатне оптимізувати витрати, покращити врожайність та зменшити негативний вплив на навколишнє середовище, що є важливим етапом у розвитку сучасного сільського господарства

Позитивні сторони.
1. Робота містить детальний огляд методів і технологій нейронних мереж, що демонструє всебічний підхід до дослідження теми.
2. Автор розробив прототип нейронної мережі для автоматичного розпізнавання сільськогосподарських культур, що показує практичну цінність дослідження.
3. Використання 62 рисунків допомагає краще зрозуміти складні технічні концепції та результати дослідження.

Недоліки.
1. В роботі відсутні таблиці, що ускладнює структурування та представлення числових даних, результатів експериментів та порівняльного аналізу.
2. Використаний обсяг і різноманітність даних можуть бути недостатніми для створення високоефективної нейронної мережі, що може обмежити результати дослідження.

Висновок: *кваліфікаційна робота на здобуття ступеня бакалавра заслуговує оцінку «відмінно», а здобувач(ка) Черниш В.В. заслуговує присвоєння кваліфікації: бакалавр.*

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

ЗМІСТ

ВСТУП

Постановка проблеми

У зв'язку з постійним розвитком технологій і впровадженням інновацій у сільському господарстві, використання нейронних мереж стає дедалі більш актуальним для вирішення різноманітних завдань. Проте існують певні проблеми, які потребують дослідження та вирішення для успішного застосування цих технологій у практичній діяльності сільськогосподарських підприємств.

Актуальність проблеми

Сільське господарство є одним із найважливіших секторів економіки у світі, і воно постійно стикається з новими викликами, такими як зміна клімату, війна, шкідники та хвороби, а також зростання світового населення. Нейронні мережі - це тип штучного інтелекту, який має потенціал революціонізувати сільське господарство, надаючи нові та інноваційні рішення для цих проблем.

Мета і завдання дослідження

Дослідити потенціал застосування нейронних мереж у сільському господарстві для оптимізації процесів, підвищення швидкості визначення кількості та тип рослин на полі, а також та якості сільськогосподарської продукції.

Метою цього дослідження є створення нейронної мережі, яка базується на великій кількості зображень та відео для подальшої роботи з сільськогосподарськими культурами, а саме:

- Розглянути існуючі методи та моделі нейронних мереж, які застосовуються в сільському господарстві.
- Розробити адаптовану модель нейронної мережі для визначення культури, її кількості, визначити на можливі наявні шкідники або оптимізувати процесів обробки землі.
- Зібрати та обробити необхідні дані для навчання та тестування розробленої моделі, включаючи дані про тип рослини.
- Навчити розроблену модель на зібраних даних та провести її тестування для оцінки точності та ефективності прогнозів.

- Порівняти результати роботи нейронної мережі з традиційними методами аналізу даних у сільському господарстві.
- Сформулювати рекомендації щодо використання нейронних мереж у практичній діяльності сільського господарства з урахуванням отриманих результатів дослідження.

Об’єкт та предмет дослідження

Об’єктом дослідження є застосування нейронних мереж у сільському господарстві. Це охоплює вивчення можливостей, переваг і обмежень застосування нейронних мереж у вирощуванні сільськогосподарської продукції та управлінні сільськогосподарськими процесами.

Предметом дослідження є моделі нейронних мереж, їх структури, методи навчання і використання в сільському господарстві для різних завдань, таких як передбачення врожайності, діагностика захворювань рослин, прогнозування погодних умов, оптимізація графіку поливу чи внесення добрив тощо. При цьому також досліджується вплив вхідних факторів (наприклад, кліматичні умови, характеристики ґрунту, агротехнічні заходи) на точність прогнозів і результати роботи нейронних мереж у сільському господарстві.

Методи дослідження

Для дослідження теми "Дослідження нейронної мережі у сільськогосподарській діяльності" можна використовувати різноманітні методи, які дозволять провести обґрунтоване дослідження ефективності застосування нейронних мереж в сільському господарстві:

- Літературний огляд: Проведення докладного аналізу наукової літератури з питань застосування нейронних мереж у сільському господарстві. Цей метод дозволить зрозуміти поточний стан досліджень, існуючі моделі та методи, а також ідентифікувати можливі проблеми та виклики.
- Збір та підготовка даних: Важливим етапом буде збір відповідних даних, необхідних для навчання нейронних мереж. Це можуть бути дані про врожайність, характеристики ґрунт, кліматичні умови, використання добрив,

захворювання рослин тощо. Дані потрібно буде підготувати для подальшого аналізу та використання в моделях нейронних мереж.

- Розробка моделі нейронної мережі: Розробка адаптованої моделі нейронної мережі з урахуванням специфіки сільського господарства. Вибір оптимальної архітектури мережі.
- Навчання та тестування моделі: Використання зібраних даних для навчання розробленої моделі нейронної мережі. Після навчання модель потрібно буде протестувати на тестових даних для оцінки її точності та ефективності.
- Аналіз результатів: Порівняння результатів роботи моделі нейронної мережі з існуючими методами та моделями в сільському господарстві. Визначення переваг і недоліків застосування нейронних мереж у порівнянні з традиційними підходами.
- Формулювання висновків і рекомендацій: На основі отриманих результатів формулювання висновків щодо ефективності застосування нейронних мереж у сільському господарстві. Розроблення практичних рекомендацій для впровадження отриманих результатів у реальну сільськогосподарську діяльність.

РОЗДІЛ 1
МЕТОДИ ТА ТЕХНОЛОГІЇ АНАЛІЗУ НЕЙРОННОЇ МЕРЕЖІ ДЛЯ ВИЗНАЧЕННЯ ОБ’ЄКТІВ У СІЛЬСЬКОГОСПОДАРСЬКІЙ ДІЯЛЬНОСТІ

1.1 Огляд використовуємих технологій для побудови нейронної мережі

Вибір технології на початку розробки нейронної мережі є не менш важливим за сам процес розробки. Рішення про те, які інструменти та фреймворки використовувати, може кардинально вплинути на хід проекту, його ефективність та кінцевий результат. Правильний вибір може суттєво спростити процес розробки, покращити продуктивність моделі та прискорити час її навчання.

У основу цього диплома лягає одна із популярніших, якщо не сама популярна інтерпретована об'єктно-орієнтована мова програмування високого рівня із суворою динамічною типізацією Python (Рис. 1.1). Її створення у 1990 році Нідерландським програмістом Гвідо ван Россум стало знаменною подією в світі IT, адже Python поєднує в собі простоту вивчення та читабельності з потужністю та універсальністю.

Рис. 1.1 Логотип Python

Її основні переваги полягають у відкритому коді, чистому синтаксисі, що не потребує використовувати крапку з комою («;») для закінчення строки, а лише потрібно використовувати відступи (Рис. 1.2) та наявність вбудованої великої кількості корисних модулів, включно з модулем для розробки графічного інтерфейсу, що дозволяє, у купі з будованим середовищем розробки, створювати програми та додатки з коробки.

Крім того, відмінною рисою цієї мови програмування є легкість вивчення та читабельність коду, що робить її ідеальним вибором для початківців і досвідчених

розробників. Python також активно використовується у сферах науки, математики та аналізу даних, завдяки потужним бібліотекам.

```python
a = float(input())
b = float(input())
s = a+b
print(s)
```

Рис. 1.2 Приклад синтаксису Python

Більшість сучасних інструментів і фреймворків для машинного навчання та глибокого навчання також підтримують Python, що робить його популярним вибором для розробки нейронних мереж та алгоритмів штучного інтелекту. Такий широкий спектр застосувань і підтримка забезпечують Python позицією однієї з найефективніших мов програмування для розробки програмного забезпечення в різних галузях. Найпоширенішими є 3 бібліотеки:

- PyTorch (Рис. 1.3) - це відкрита бібліотека машинного навчання, розроблена компанією Facebook (Meta), яка швидко стала одним з найпопулярніших інструментів для розробки нейронних мереж. Вона ґрунтується на бібліотеці Torch, яка написана мовами програмування Lua та C, але пропонує більш зручний та інтуїтивно зрозумілий інтерфейс на мові Python.

Рис. 1.3 – Логотип PyTorch

- Tensorflow (Рис. 1.4) - це відкрита програмна бібліотека для машинного навчання, розроблена компанією Google. Вона використовується для побудови

та тренування нейронних мереж, які можуть виявляти та розшифровувати складні закономірності в даних. TensorFlow дає змогу створювати моделі машинного навчання, які здатні навчатися на великих обсягах даних та виконувати завдання, подібні до тих, що виконують люди, такі як розпізнавання зображень, обробка природної мови та прийняття рішень.

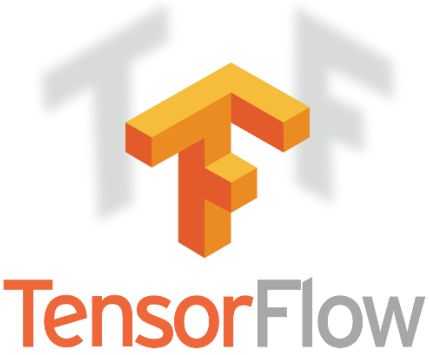


Рис. 1.4 – Логотип Tensorflow

- Keras (Рис. 1.5) - це відкрита нейромережна бібліотека, написана мовою Python, яку розробили з метою полегшення та прискорення експериментів з нейронними мережами глибокого навчання. Вона фокусується на простоті використання, модульності та розширюваності, що робить її зручним інструментом як для початківців, так і для досвідчених розробників у галузі машинного навчання.Її головним автором є французький інженер компанії Google Франсуа Шоллє.

Рис. 1.5 – Логотип Keras

Бібліотеки для роботи з нейронними мережами мають потужний функціонал, але часто виникають проблеми з аналізом вихідних даних. Для вирішення цієї проблеми ідеально підходить Matplotlib (Рис. 1.6) та Pandas (Рис. 1.7).

Matplotlib є потужним інструментом для візуалізації даних у Python. Він дозволяє будувати різноманітні графіки, діаграми та зображення для аналізу результатів нейронних мереж. Наприклад, ви можете побудувати графік залежності

точності вашої моделі від числа епох навчання або візуалізувати розподіл вихідних даних.

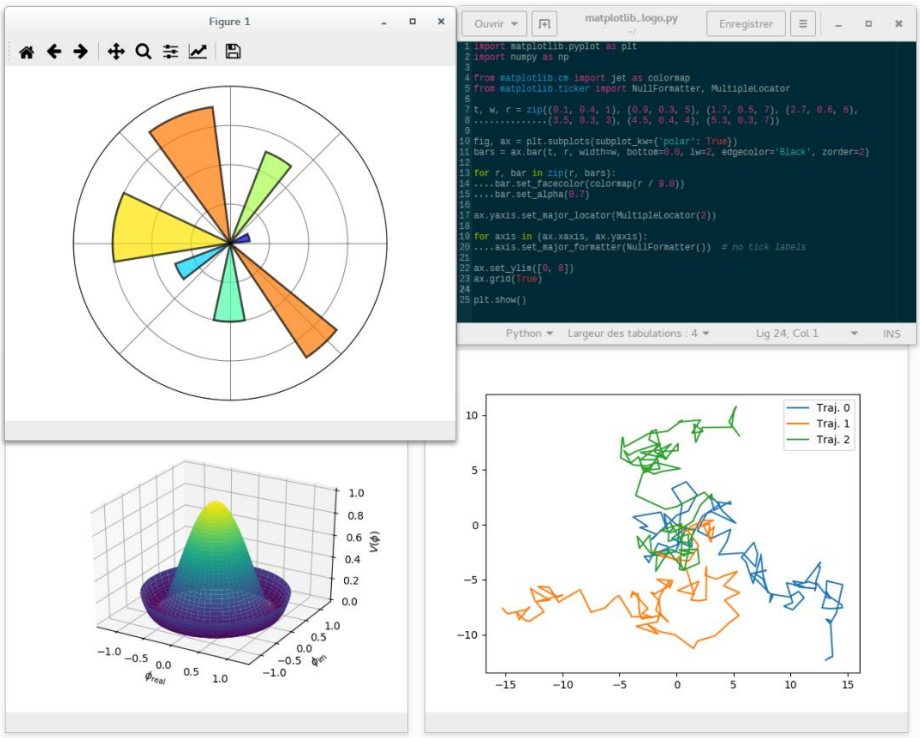


Рис. 1.6 -Приклад використання Matplotlib

Pandas, з іншого боку, є потужною бібліотекою для обробки та аналізу даних. Вона надає зручний інтерфейс для роботи з табличними даними, що може бути корисним при підготовці даних для навчання нейронних мереж. Наприклад, ви можете легко завантажувати, фільтрувати та перетворювати дані з допомогою Pandas, щоб підготувати їх для подальшого використання у вашому нейронному моделюванні.

```
In [36]: import pandas as pd
         left = pd.DataFrame({
            'id':[1,2,3,4,5],
            'Name': ['Jack', 'Amy', 'Elias', 'Young', 'Smith'],
            'subject_id':['sub1','sub2','sub4','sub6','sub5']})
         right = pd.DataFrame({
            'id':[1,2,3,4,5],
            'Name': ['Billy', 'Brooks', 'Brown', 'Aurier', 'Jose'],
            'subject_id':['sub2','sub4','sub3','sub6','sub5']})
         print (pd.merge(left,right,on='id'))
```

	id	Name_x	subject_id_x	Name_y	subject_id_y
0	1	Jack	sub1	Billy	sub2
1	2	Amy	sub2	Brooks	sub4
2	3	Elias	sub4	Brown	sub3
3	4	Young	sub6	Aurier	sub6
4	5	Smith	sub5	Jose	sub5

Рис. 1.7 – Приклад використання Pandas

Комбінація Matplotlib та Pandas дозволяє ефективно візуалізувати та аналізувати дані з нейронних мереж, сприяючи покращенню розуміння результатів моделей та виявленню проблем у вихідних даних для подальшого вдосконалення моделі.

При роботі з нейронними мережами з'являється багато клопоту у вигляді роботи з масивами та матрицями, і стандартного функціоналу Python вже не вистачає. Для вирішення цих задач ідеально підходить розширення мови Python під назвою NumPy (Рис. 1.8), що додає підтримку великих багатовимірних масивів і матриць, разом з великою бібліотекою високорівневих математичних функцій для операцій з цими масивами.

```
>>> a[(0,1,2,3,4), (1,2,3,4,5)]
array([1, 12, 23, 34, 45])

>>> a[3:, [0,2,5]]
array([[30, 32, 35],
       [40, 42, 45],
       [50, 52, 55]])

>>> mask = np.array([1,0,1,0,0,1], dtype=bool)
>>> a[mask, 2]
array([2, 22, 52])
```

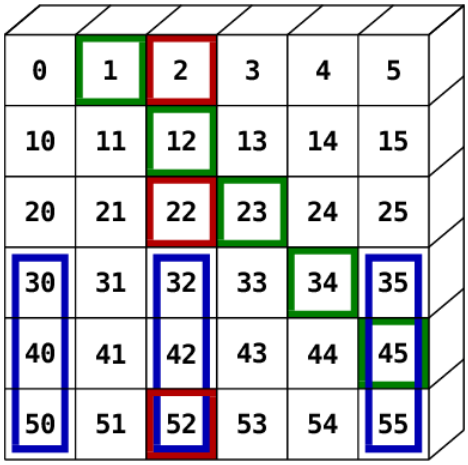


Рис. 1.8 – Приклад застосування Numpy

NumPy надає зручні інструменти для ефективної роботи з числовими даними у вигляді масивів. Він дозволяє виконувати швидкі обчислення та математичні операції на великих об'ємах даних, що є критичним для розробки нейронних мереж. Наприклад, з допомогою NumPy ви можете легко виконувати операції додавання, множення, транспонування матриць та багато інших, необхідних при обробці даних у контексті нейронних мереж.

Крім того, NumPy інтегрується з іншими бібліотеками Python, такими як TensorFlow та PyTorch, що робить його популярним вибором для розробки нейронних алгоритмів. Його швидкість і ефективність дозволяють працювати з

великими обсягами даних, що зробить процес навчання нейронних мереж більш ефективним і продуктивним.

Таким чином, NumPy є важливим інструментом для розробників нейронних мереж, який спрощує роботу з масивами та матрицями і підвищує продуктивність у процесі моделювання і обробки даних.

Центральним завданням даного дипломного проекту є розробка нейронної мережі, спроможної визначати об'єкти на основі наданих даних, з використанням технологій комп'ютерного зору. Цільова функція цієї мережі полягає у здійсненні точного аналізу візуальних вхідних даних та виявленні об'єктів на зображеннях чи відео.

З усіх перерахованих технологій, бібліотека Matplotlib частково може допомогти у цьому процесі, забезпечуючи зручні інструменти для візуалізації даних та створення графіків. Однак для повноцінної реалізації задач комп'ютерного зору потрібен більш потужний інструмент.

Для прогресивної реалізації завдань з визначення об'єктів на зображеннях та маніпулювання вхідними даними у реальному часі ідеально підійде бібліотека OpenCV (Рис. 1.9) (Open Source Computer Vision Library). OpenCV надає багатофункціональні інструменти для обробки зображень, відео та аудіо, що забезпечують широкі можливості в розробці систем комп'ютерного зору. Вона має вбудовані алгоритми для виявлення об'єктів, розпізнавання облич, відстеження руху та багато інших функцій, необхідних для реалізації задачі комп'ютерного зору.

Таким чином, використання бібліотеки OpenCV у поєднанні з нейронною мережею дозволить досягти високої ефективності та точності в розв'язанні завдань з комп'ютерного зору, забезпечуючи можливість аналізу вхідних даних у реальному часі та автоматичного виявлення об'єктів на зображеннях чи відео.

Рис. 1.9 – Обробка зображення через OpenCV у реальному часі

Впровадження складних програмних рішень на мові Python потребує використання потужних інструментів, які виходять за межі можливостей вбудованих середовищ розробки, таких як IDLE. Вони пропонують базовий набір інструментів для написання та запуску коду Python.

Однак вони мають ряд суттєвих недоліків, які роблять їх непридатними для розробки масштабних програмних систем. З найсуттєвіших недоліків можна виділити недостатній функціонал, що характеризує себе неможливістю автоматичного доповнення коду, що у сучасній розробці є необхідністю, тому як ця можливість може в рази прискорювати розробку. Не меншим недоліком є той факт, що стандартне середовище немає жодних функцій інтеграції з іншими інструментами розробки, що ускладнює процес розробки, знижує його продуктивність. Та найпопулярнішим недоліком являє собою інтерфейс, який незручний або неінтуїтивно зрозумілий, що негативно впливає на досвід розробника та його ефективність

Ця проблема спонукала компанію JetBrains до розробки PyCharm (Рис. 1.10) - інтегрованого середовища розробки (IDE), яке не має аналогів.

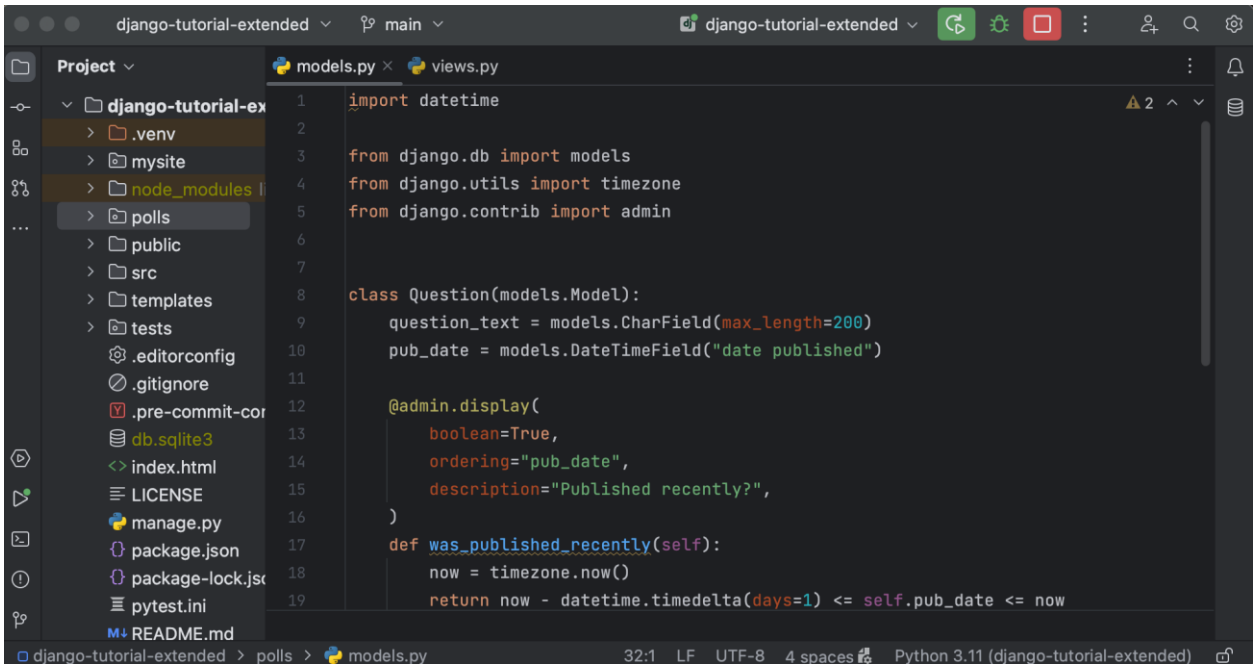


Рис. 1.10 – Знімок екрану з середи розробки PyCharm

PyCharm має зручний та ергономічний інтерфейс та долає всі недоліки вбудованих середовищ розробки Python та пропонує широкий спектр функцій, необхідних для розробки масштабних програмних систем, а саме:

- Автоматичне доповнення коду
- Інтегрований налагоджувач
- Підтримка систем контролю версій
- Підтримка наукових бібліотек
- Розширюваність

1.2 Вибір існуючих моделей Object Detection

Вибір існуючих моделей для завдання виявлення об'єктів (Object Detection) (Рис. 1.11) залежить від різних факторів, включаючи точність, швидкодію, складність моделі, вартість обчислень та ресурси, доступні для розгортання. Обрати оптимальну модель важливо для досягнення ефективної роботи системи комп'ютерного зору.

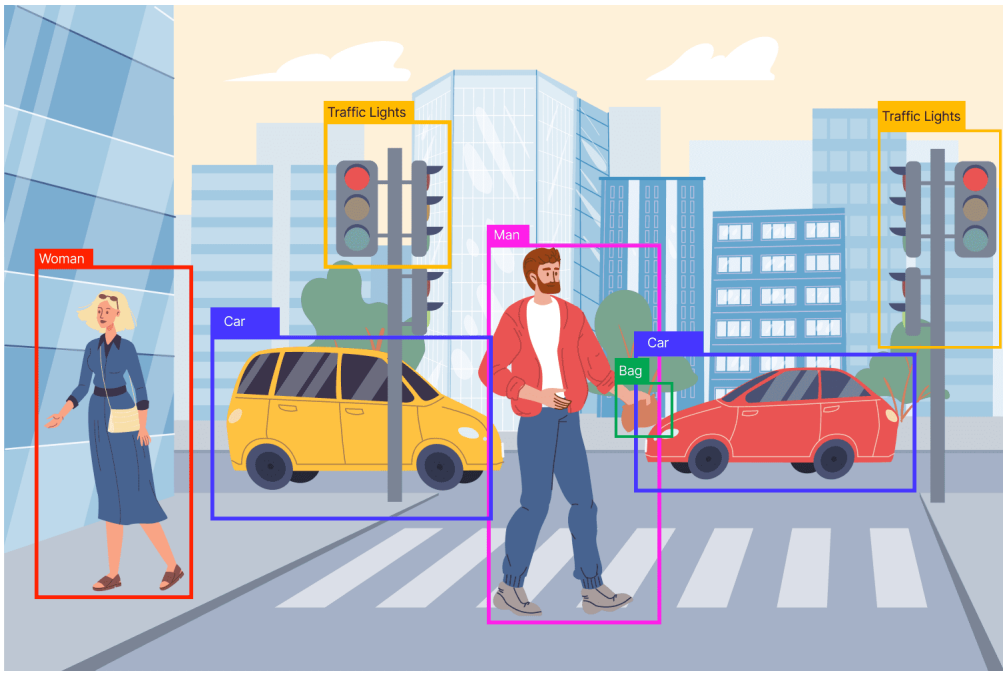


Рис. 1.11 – Приклад дії Object detection

Впровадження методів Object Detection стало невід'ємною частиною сучасного світу, значно вплинувши на різні сфери життя. Їхню універсальність та безмежний потенціал можна спостерігати у широкому спектрі застосувань, від звичних побутових завдань до складних наукових досліджень, але найбільш поширенішими прикладами є:

- Сфера безпеки (Рис. 1.12): Системи відеоспостереження, оснащені технологією виявлення об'єктів, стали важливим інструментом у сфері забезпечення безпеки. Вони активно сприяють запобіганню злочинам та інцидентам, адже автоматично виявляють та відстежують підозрілі особи, транспортні засоби та інші об'єкти. Ця технологія забезпечує більш ефективне контролювання об'єктів та подій, що сприяє зниженню кримінальної активності та підвищенню загального рівня безпеки в об'єктах відеоспостереження.

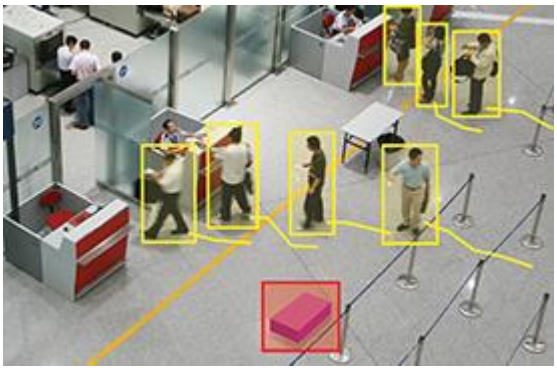


Рис. 1.12 – Object Detection у сфері безпеки

- Галузь охорони здоров'я (Рис. 1.13): Лікарі використовують передові технології обробки зображень, зокрема системи Object Detection, для підтримки точної та ефективної діагностики різних захворювань. Ці системи дозволяють автоматично аналізувати медичні зображення, такі як рентгенівські знімки, комп'ютерні томографії (CT), магнітно-резонансні знімки (MRI) та інші, для виявлення ознак патологій, включаючи рак, пухлини та інші аномалії.

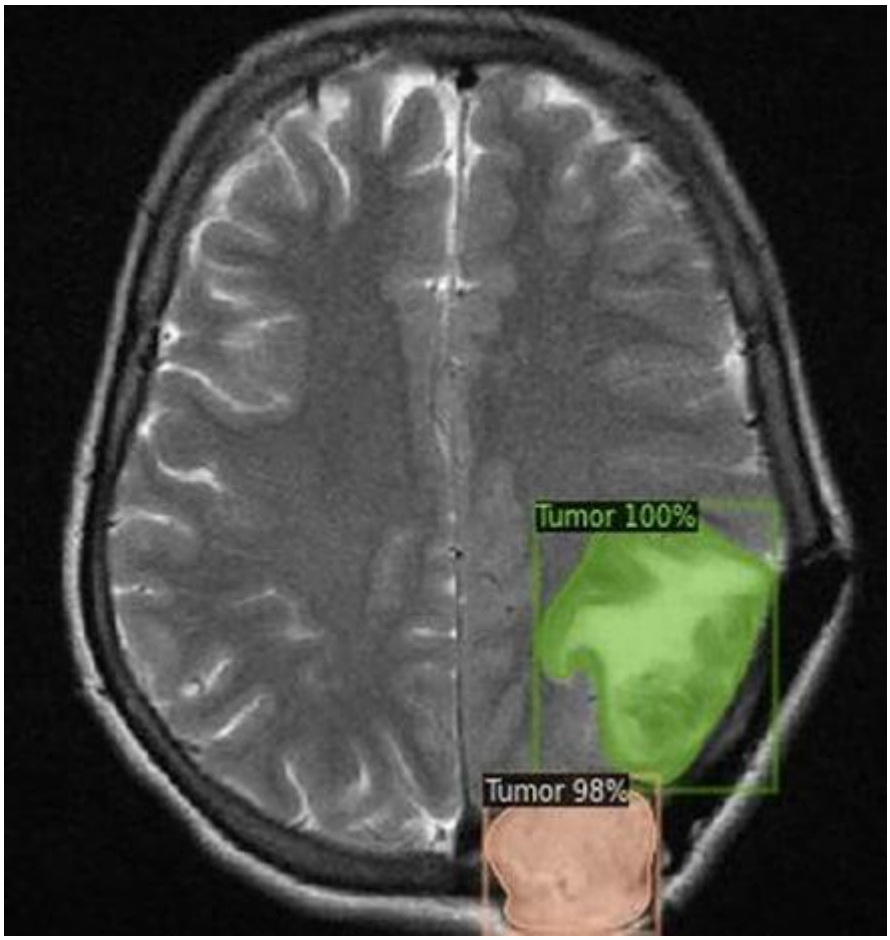


Рис. 1.13 – Виявлення пухлин головного мозку за допомоги Object Detection

- Виробнича сфера (Рис. 1.14): Впровадження Object Detection у виробничу сферу відкриває нові горизонти для оптимізації процесів, підвищення якості продукції та безпеки праці. Ця технологія використовується для відстеження та контролю об'єктів на виробничих лініях, що забезпечує ряд суттєвих переваг.

Рис. 1.14 – Пошук браку на виробничій лінії за допомогою Object Detection

- Військова промисловість (Рис. 1.15): Впровадження технології Object Detection у військовій сфері відкриває нові горизонти для значного підвищення боєздатності, покращення ситуаційної обізнаності та рятування життів. Ця інноваційна технологія може використовуватися для автоматизації виявлення та відстеженню ворожих сил, техніки або інших загроз на зображеннях, отриманих з безпілотників, супутників та інших джерел розвідки.

Рис. 1.15 – Використання Object Detection у військових цілях

- Сільське господарство (Рис. 1.16): Фермери використовують Object Detection для визначення рослинних культур, їхньої кількості, худоби, а також моніторингу стану посівів, виявляючи шкідників, хвороби та інші проблеми.

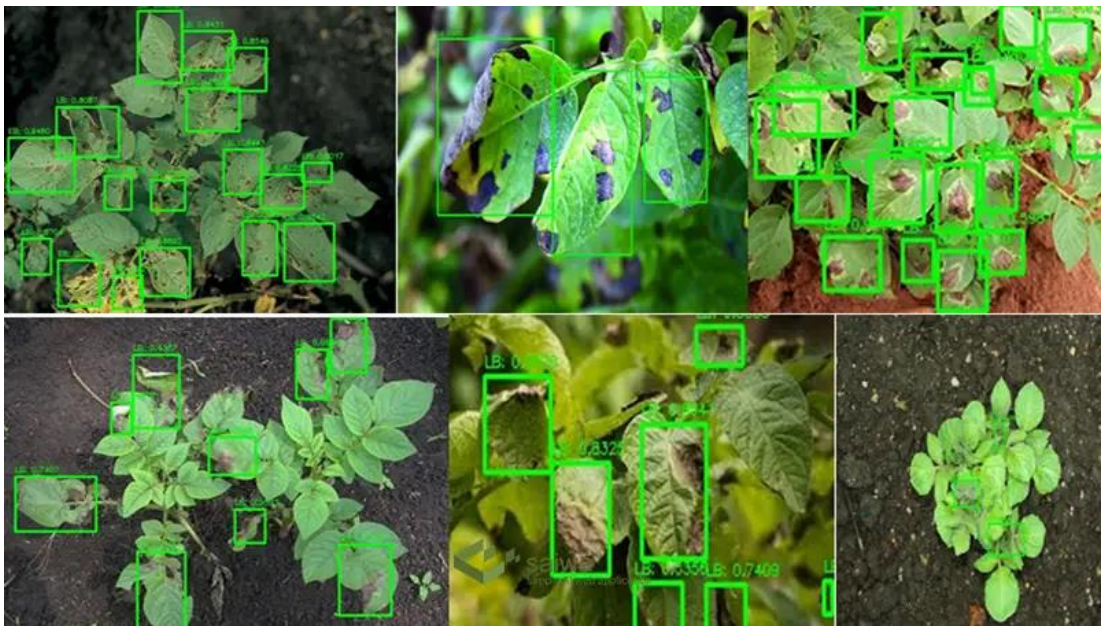


Рис. 1.16 – Використання Object Detection у сільськогосподарській промисловості

У сфері визначення об’єктів існує розмаїття моделей, проте найбільш поширеними і визнаними є наступні:

- Faster R-CNN (Region-based Convolutional Neural Network) (Рис. 1.17):

Модель на базі якої буде розроблений прототип цього диплому - Faster R-CNN є потужною моделлю для виявлення об'єктів, яка складається з двох основних компонентів: відображувача (backbone) і Region Proposal Network (RPN). Відображувач використовується для винайдення функцій зображення, після чого RPN генерує пропозиції областей, де можуть знаходитися об'єкти. Після цього використовується класифікатор для точного визначення класу об'єктів та їхніх bounding boxes. Faster R-CNN відомий своєю високою точністю, але вимагає більш великих обчислювальних ресурсів порівняно з іншими моделями.

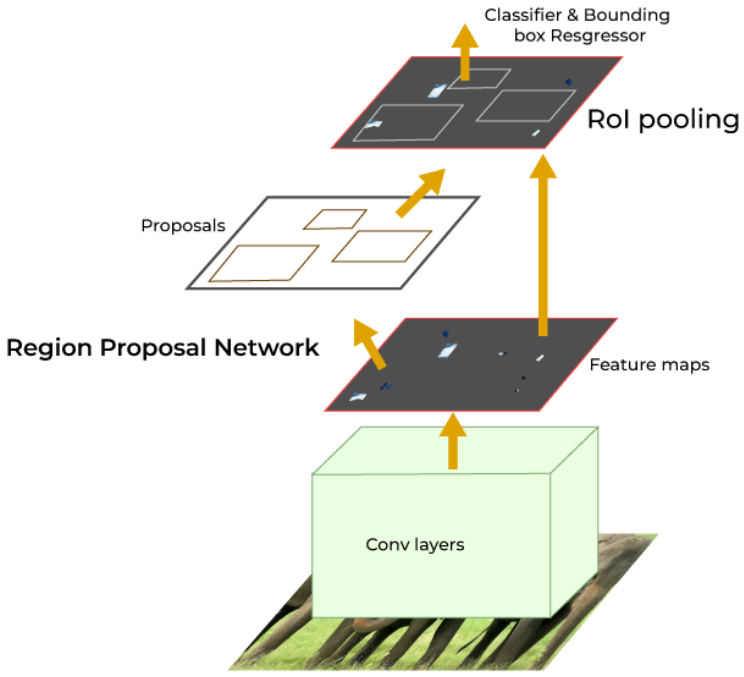


Рис. 1.17 – Приклад обробки Faster R-CNN

- YOLO (You Only Look Once) (Рис. 1.18):

YOLO є швидкою моделлю для виявлення об'єктів, яка поділяє зображення на сітку та застосовує одну сверточну мережу для одночасного прогнозування класів та bounding boxes. YOLO забезпечує високу швидкодію, роблячи його ідеальним вибором для застосувань, де потрібна реального часу обробка, наприклад, у системах моніторингу або автономних транспортних засобах.

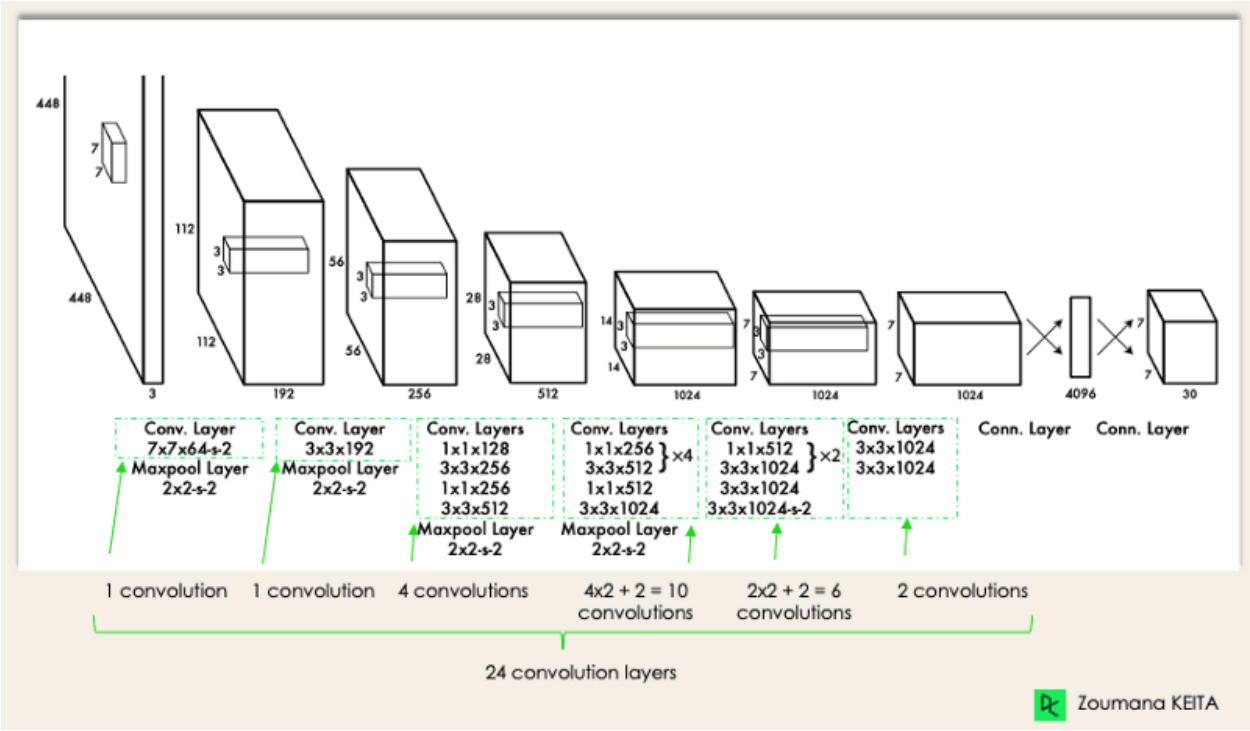


Рис. 1.18 – Приклад обробки YOLO

- SSD (Single Shot MultiBox Detector) (Рис. 1.19):

  SSD є іншою швидкою моделлю для виявлення об'єктів, яка використовує одну сверточну мережу для прогнозування класів та bounding boxes на різних масштабах. SSD є компактнішою за Faster R-CNN і може бути ефективною у вимогливих до швидкості застосуваннях, таких як системи відеоспостереження.

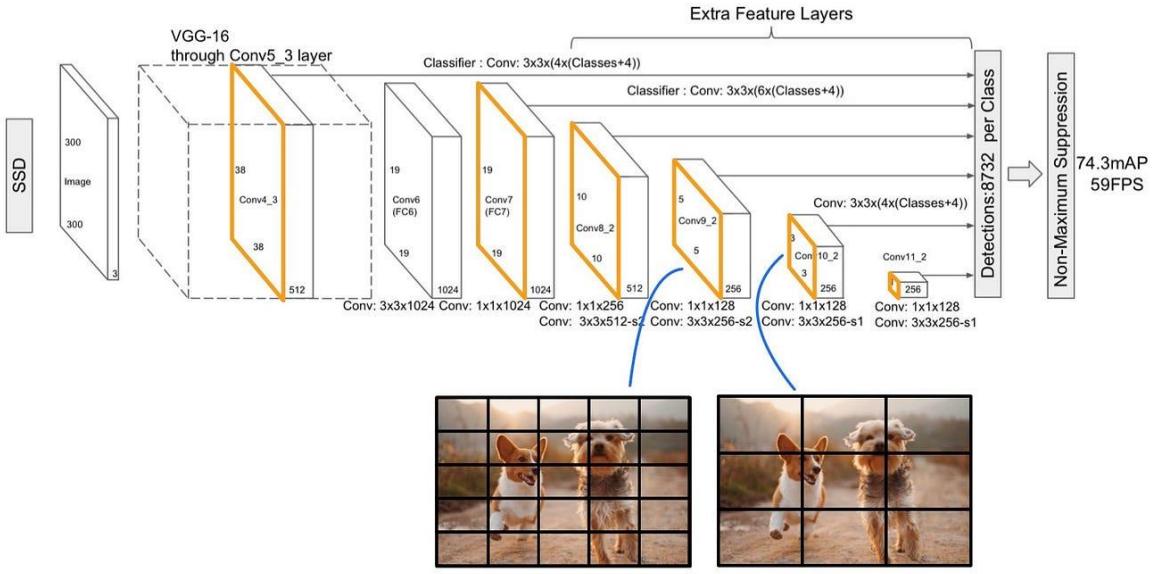


Рис. 1.19 - Приклад обробки SSD

- Mask R-CNN (Рис. 1.20):

  Mask R-CNN є розширенням Faster R-CNN, яке дозволяє додатково прогнозувати маски для кожного об'єкта. Це дозволяє точніше визначати форму та контур об'єктів, що є корисним для сегментації об'єктів у складних зображеннях або відео.

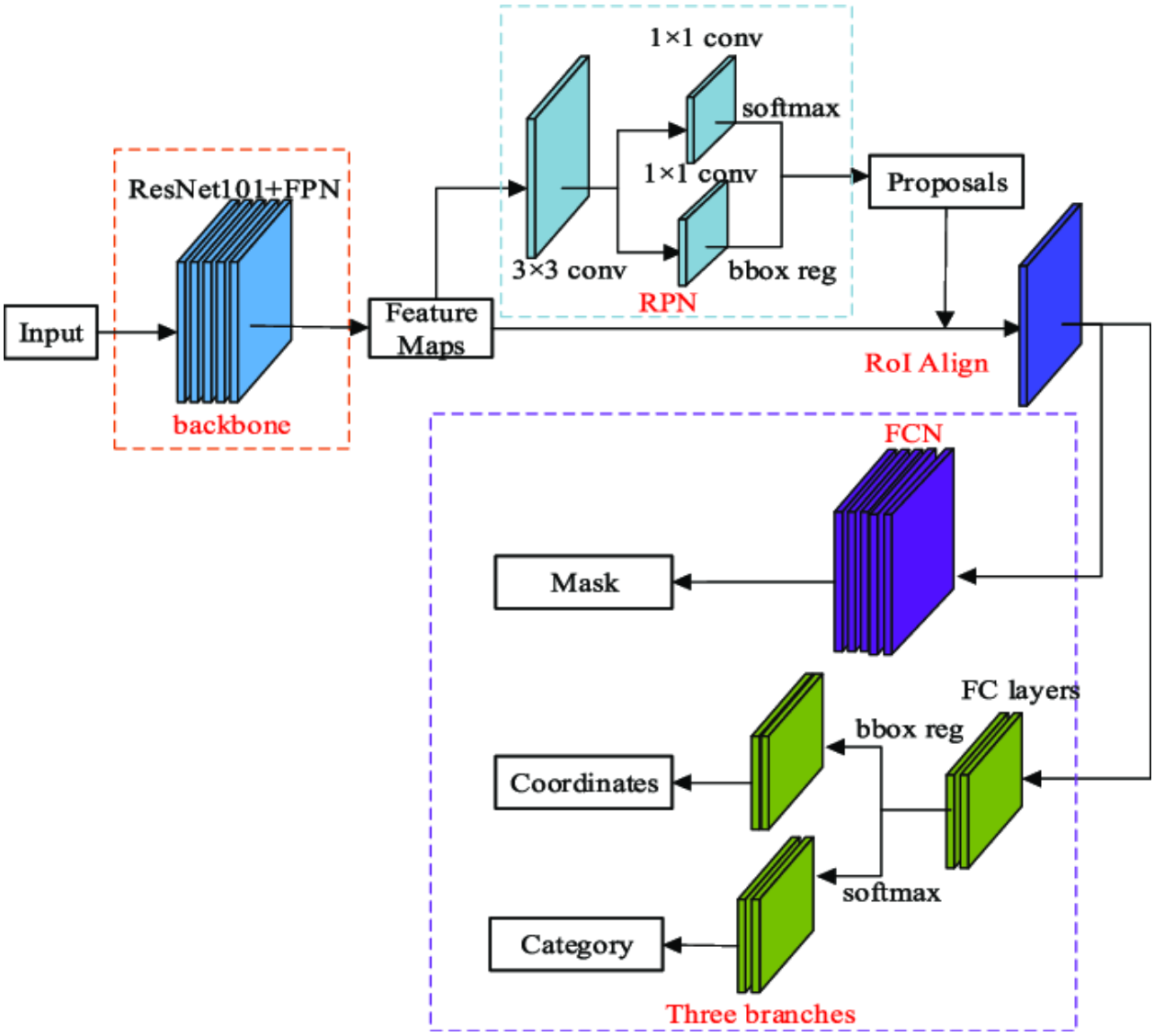


Рис. 1.20 - Приклад обробки Mask R-CNN

- RetinaNet (Рис. 1.21):

RetinaNet був спеціально розроблений для вирішення проблеми нерівномірної класової розподіленості в даних для виявлення об'єктів. Він використовує новаторську архітектуру, що поєднує Feature Pyramid Network (FPN) з Focal Loss для підвищення точності прогнозів навіть у випадку рідких класів об'єктів.

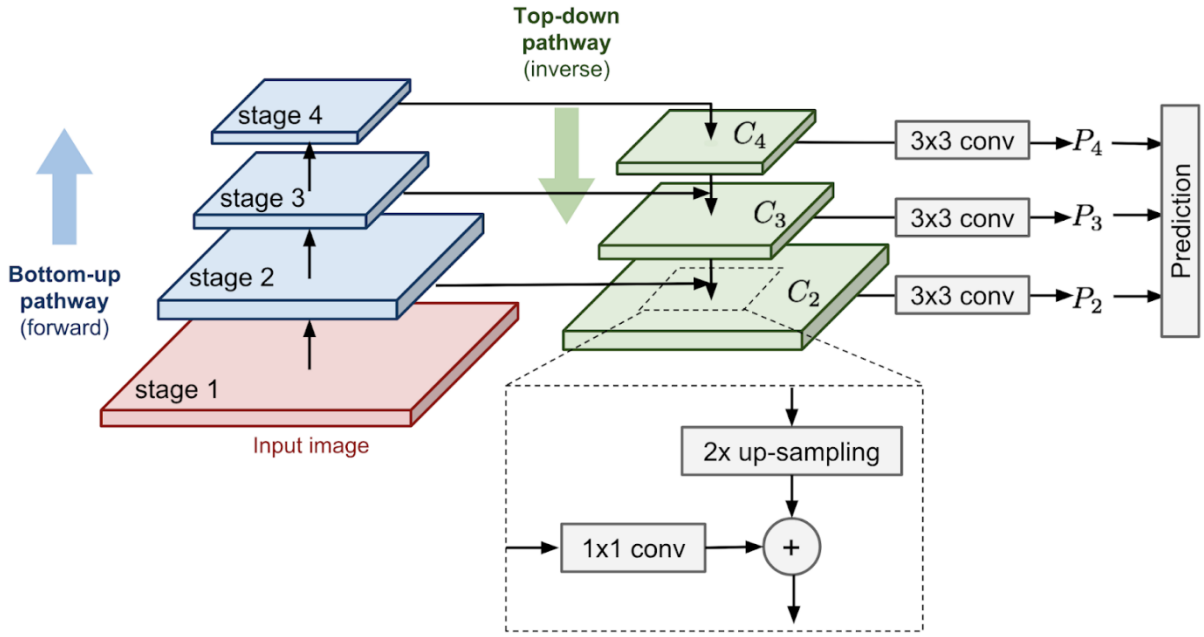


Рис. 1.21 - Приклад обробки RetinaNet

1.3 Переваги та недоліки Object Detection у сільськогосподарській діяльності

У сучасному сільському господарстві технології комп'ютерного зору, зокрема системи виявлення об'єктів (Object Detection), знаходять широке застосування для автоматизації та оптимізації процесів виробництва та контролю. Виявлення об'єктів у сільському господарстві може включати виявлення рослин, хвороб, шкідників, тварин або визначення стану ґрунту та урожайності. Однак, разом з численними перевагами, існують і деякі недоліки, які варто враховувати при впровадженні таких систем.

Переваги:

1. Автоматизація процесів: Системи Object Detection дозволяють автоматизувати багато аспектів сільськогосподарської роботи, такі як моніторинг росту рослин, виявлення шкідників чи автоматичний збір урожаю. Це значно підвищує ефективність та продуктивність роботи сільськогосподарських підприємств.

2. Точність і швидкість: Сучасні моделі Object Detection мають високу точність виявлення об'єктів і працюють на високій швидкості, що дозволяє оперативно реагувати на зміни у виробничих процесах та умовах росту.
3. Мінімізація витрат: Використання систем виявлення об'єктів дозволяє зменшити витрати на ручну працю та застосування хімічних пестицидів або добрив, оскільки дії можуть бути спрямовані тільки на ті ділянки, де є необхідність.
4. Моніторинг та аналіз: Обробка даних, зібраних за допомогою систем виявлення об'єктів, дозволяє проводити детальний моніторинг стану посівів, прогнозування врожаю та вчасну діагностику захворювань.

Недоліки:

1. Вартість впровадження: Висока вартість придбання та налаштування необхідного обладнання та програмного забезпечення може бути перешкодою для багатьох сільськогосподарських підприємств.
2. Складність налаштування: Налаштування систем Object Detection для конкретних умов росту рослин або визначення об'єктів може вимагати фахової експертизи та часу на налагодження.
3. Обробка великої кількості даних: Використання систем виявлення об'єктів може призвести до накопичення великої кількості даних, які потребують обробки та аналізу, що може бути складним завданням

РОЗДІЛ 2
МЕТОДОЛОГІЯ СТВОРЕННЯ НЕЙРОННОЇ МЕРЕЖІ У РОБОТІ З СІЛЬСЬКОГОСПОДАРСЬКИМИ КУЛЬТУРАМИ

2.1 Підготовка обладнання до початку роботи

Розробка прототипу технології автоматичного розпізнавання культур на основі нейронних мереж потребувала збору різноманітних даних. Для навчання та тестування цієї мережі було зібрано обширний набір фото- та відеоматеріалів сільськогосподарських полів України. Ці поля були засаджені різними типами рослин, включаючи кукурудзу, соняшник, бур’ян та інші. Збір даних відбувався на різній висоті, від 20 до 150 метрів, що дозволило отримати зображення з різною роздільною здатністю та деталізацією. Загальний обсяг відзнятого матеріалу склав більше 20 гігабайт, чого цілком достатньо для створення прототипу.

Збір даних для розробки прототипу технології автоматичного розпізнавання культур на основі нейронних мереж здійснювався за допомогою комплексу безпілотних літальних апаратів (БПЛА) DJI, що включав моделі серії Mavic та Agras.

Використання БПЛА серії Mavic (Рис. 2.1) дозволило отримати високоякісні зображення сільськогосподарських полів з різними типами рослин протягом короткого проміжку часу. Це було важливо для створення репрезентативного набору даних, що охоплює різні умови освітлення, фази розвитку рослин та типи ґрунту.

Рис. 2.1 — Дрон DJI Mavic 3T

БПЛА серії Agras (Рис. 2.2), оснащені передовими технологіями, забезпечили збір даних, що максимально відповідають потребам майбутнього, де дрони будуть оснащені нейронними мережами для автоматичного розпізнавання культур. Завдяки своїм потужним двигунам, великим бакам для рідини та вбудованим датчикам, дрони Agras можуть ефективно обробляти великі території, збираючи дані з високою точністю.

Рис. 2.2 — Дрон DJI Agras T16

Для досягнення найкращих результатів у цій роботі використовувалася камера Intel RealSense (Рис. 2.3). Ця камера спеціально розроблена для комп'ютерного зору та забезпечує високоточне відстеження глибини та RGB-зображення. Вбудований IMU (Inertial Measurement Unit) або Інерційний вимірювальний пристрій (Рис. 2.4), допомагає камері краще орієнтуватися в просторі, навіть коли вона рухається.

Рис. 2.3 — Камера Intel RealSense D456

Відкритий код та бібліотеки для мови програмування Python роблять Intel RealSense ідеальним вибором для інтеграції з дронами, оснащеними розробленою нейронною мережею.

Ця камера є революційною розробкою в галузі промисловості, адже вона дозволяє робити знімки високої якості на відстані до 20 метрів, що значно більше, ніж 5 метрів, які були максимально можливими ще рік тому.

З огляду на стрімкий розвиток технологій, можна очікувати, що вже через рік максимальна відстань зйомки камерою Intel RealSense може сягати 50 метрів і більше. Це робить її фаворитом для збору даних, необхідних для фото- та відеотехнологій.

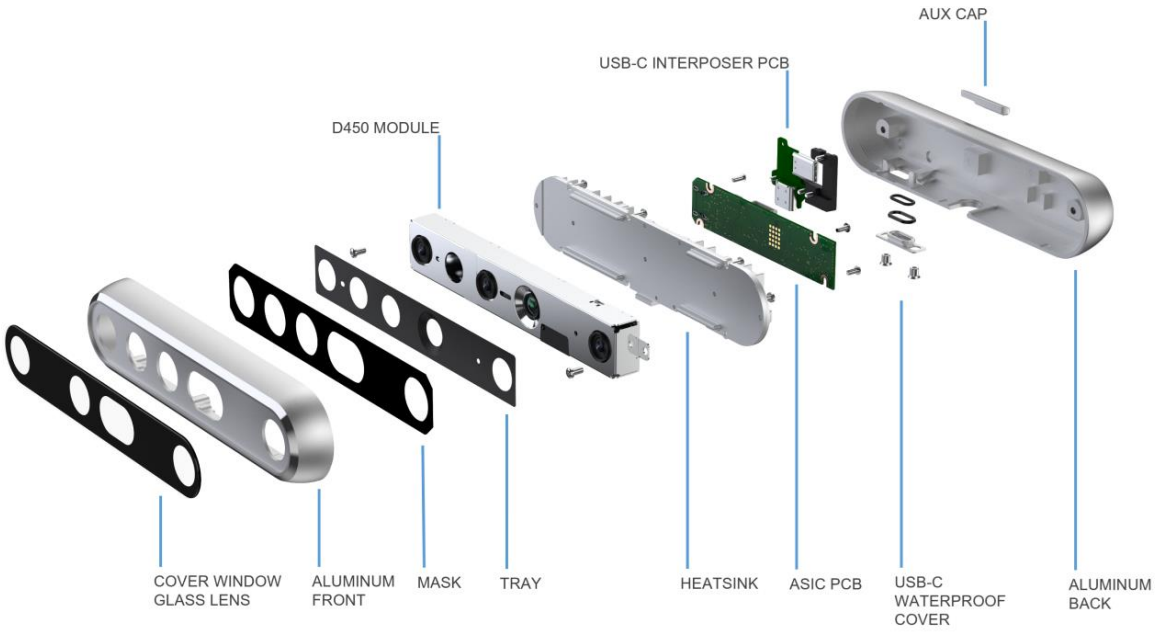


Рис. 2.4 — Схема збірки камери Inter RealSense D456

Після завершення збору даних за допомогою дронів та камери Intel, наступним етапом стане вибір обладнання для розробки та подальшої обробки нейронної мережі. Найкращим вибором для цього завдання буде робоча станція на базі промислових відеокарт Nvidia Quadro та процесорів Intel XEON серії Platinum або AMD Ryzen THREADRIPPER (Рис. 2.5).

Рис. 2.5 — Найкраще обладнання для розробки нейронної мережі

Використання операційної системи Ubuntu дозволить безперешкодно вести розробку нейронної мережі, а також її подальше оброблення з великими обсягами вхідних даних протягом короткого проміжку часу. Це дасть можливість багаторазово тестувати та обробляти прототип протягом дня, що дозволить швидко перевірити роботу нейронної мережі на різних моделях Object Detection, даних та налаштуваннях.

Завдяки такому підходу можна буде суттєво скоротити час розробки та оптимізувати роботу нейронної мережі, досягнувши максимальної ефективності.

Незважаючи на певні переваги пропонованої конфігурації, на жаль, її впровадження є неможливим з низки причин:

1. Висока вартість. Мінімальна вартість комплектуючих для даної машини перевищує півмільйона гривень. Зважаючи на обмеженість бюджету дослідницького проекту, таке придбання є нераціональним та невиправданим.

2. Складна військово-політична ситуація в Україні. Військові дії та пов'язані з ними логістичні проблеми суттєво ускладнюють, а в деяких випадках роблять неможливим доставку обладнання з західних кордонів країни. Це робить процес комплектування робочої машини не лише фінансово невигідним, але й ризикованим з точки зору дотримання термінів та якості виконання проекту.

З огляду на вищезазначені причини, потрібно розглядати альтернативні варіанти комплектування робочої машини, які відповідають потребам дослідницького проекту та є більш раціональними з точки зору бюджету та логістики:

1. Системи TPU або Tensor Proccesor Unit (Тензорний блок обробки) (Рис. 2.6), розроблені компанією Google. TPU – це спеціалізовані апаратні прискорювачі, оптимізовані для завдань машинного навчання. Завдяки своїм характеристикам вони демонструють значно кращу продуктивність та ефективність порівняно з традиційними CPU та GPU, особливо при виконанні операцій, типових для нейронних мереж (множення матриць). TPU глибоко інтегровані з фреймворком Google TensorFlow, що забезпечує швидке навчання та створення моделей ШІ.

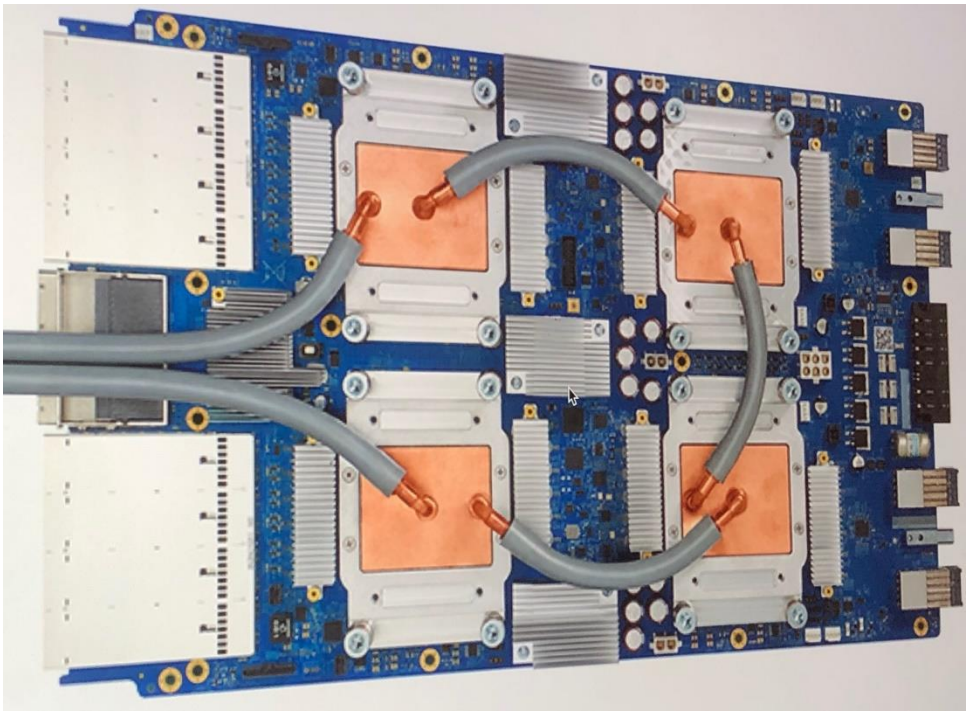


Рис. 2.6 — Tensor Proccesor Unit

2. Промислові відеокарти серії Arc (Рис. 2.7) компанії Intel. Ці відеокарти пропонують значні можливості за порівняно невеликі кошти, що робить їх вигідним варіантом для дослідницьких проектів. Їхня продуктивність достатня для вирішення широкого кола задач машинного навчання, а їх доступність на ринку та простота інтеграції роблять їх привабливим вибором.

Рис. 2.7 — Відеокарта Intel Arc

3. Споживацькі та ігрові відеокарти. В певних випадках можливо використовувати споживацькі та ігрові відеокарти, які за своїми характеристиками відповідають потребам дослідницького проекту. Їхня перевага полягає в доступній ціні та широкій поширеності на ринку.

Для забезпечення безперебійної роботи та максимальної ефективності розробки нейронної мережі важливо використовувати лише нові та неушкоджені комплектуючі. Це зумовлено кількома факторами:

- Ризик непередбачуваних проблем. Використання вживаних або відновлених комплектуючих може призвести до виникнення непередбачуваних проблем, які можуть суттєво ускладнити або й зупинити процес розробки.
- Гарантія повної працездатності. Нові комплектуючі супроводжуються гарантією виробника, яка гарантує їх працездатність та можливість безкоштовної заміни у разі виявлення дефектів.
- Доступ до повного потенціалу. Використання нових комплектуючих гарантує доступ до їх повного потенціалу, що дозволяє максимально ефективно використовувати їх ресурси для вирішення задач розробки нейронної мережі.

На жаль, відеокарти AMD та Intel не сумісні з Tensorflow та Keras, які є основними технологіями, що використовуються для розробки нейронних мереж. Це робить їх непрактичними для даного проекту.

Після ретельного аналізу та порівняння характеристик різних варіантів комплектуючих, було прийнято рішення про вибір серії відеокарт Nvidia RTX 4090 та процесора AMD Ryzen 9 7950X3D (Рис. 2.8). Цей вибір ґрунтується на наступних факторах:

- Сумісність з Tensorflow та Keras. Nvidia RTX 4090 та AMD Ryzen 9 7950X3D сумісні з Tensorflow та Keras, що робить їх оптимальним вибором для даного проекту.
- Висока продуктивність. Nvidia RTX 4090 та AMD Ryzen 9 7950X3D пропонують високу продуктивність, яка необхідна для ефективного розробки, обробки та тестування нейронних мереж.
- Відсутність технічних обмежень. Ця комбінація комплектуючих не має жодних технічних обмежень з точки зору розробки нейронних мереж.
- Оптимальне співвідношення ціни та якості. Nvidia RTX 4090 та AMD Ryzen 9 7950X3D пропонують оптимальне співвідношення ціни та якості, що робить їх економічно вигідним вибором для даного проекту.

Рис. 2.8 — Вибране оптимальне обладнання

Тож вибір серії відеокарт Nvidia RTX 4090 та процесора AMD Ryzen 9 7950X3D є обґрунтованим та оптимальним для розробки, обробки та тестування нейронних мереж. Ця комбінація комплектуючих гарантує сумісність з Tensorflow та Keras, високу продуктивність, відсутність технічних обмежень та оптимальне співвідношення ціни та якості.

2.2 Вибір та аналіз початкових даних

В рамках розділу "2.1 Підготовка обладнання до початку роботи" за допомогою безпілотного літального апарату DJI було зібрано значний масив початкових даних, загальний обсяг яких становить більше 20 гігабайт. Ці дані представлені у вигляді фотоматеріалів, що зображують різні культури, зняті на різних висотах. Для розробки прототипу пропонується використовувати

фотоматеріали, що стосуються соняшнику та кукурудзи, з метою демонстрації можливостей даної технології.

Зважаючи на значний обсяг зібраних даних, було проведено ретельний відбір фотографій, що відповідають найвищим стандартам якості та чітко демонструють ключові атрибути досліджуваних рослин. Для розробки прототипу системи машинного навчання було вибрано 4 масштабних зображення, що представляють собою два поля соняшнику та два поля кукурудзи. Ці зображення охоплюють всю територію досліджуваних ділянок, надаючи комплексне уявлення про особливості рослин.

Для кожного з двох типів культур (соняшник та кукурудза) буде використано два типи фотоматеріалів:

- Оригінальні фотографії (Рис. 2.9): Ці зображення, не розділені на фрагменти, охоплюють одночасно більше 1000 рослин одного типу. Це дозволить системі машинного навчання навчатися на великих обсягах даних, що сприятиме кращому розпізнаванню та класифікації рослин.

Рис. 2.9 — Приклад фотографії поля оригінального розміру

- Розбиті фотографії (Рис. 2.10): Ці зображення будуть штучно розділені на 100 окремих фотографій, кожна з яких чітко демонструє окрему рослину. Це дозволить системі машинного навчання детально вивчати особливості кожної рослини, такі як форма, колір та інші візуальні характеристики.

Рис. 2.10 — Приклад розділеної фотографії поля

Використання цих двох типів фотоматеріалів дозволить системі машинного навчання навчатися на даних, що представляють собою широкий спектр візуальних характеристик досліджуваних культур. Це, в свою чергу, сприятиме кращому розумінню системи машинного навчання особливостей рослин та, як наслідок, більш точній їх ідентифікації та класифікації.

Після завершення процесу відбору фотоматеріалів, наступним кроком є створення XML-файлу (Рис. 2.11), який буде містити координати всіх визначених об'єктів. Цей процес, відомий як анотування, полягає у додаванні міток до зображень. Ці мітки слугують навчальним набором даних для нейромережі, допомагаючи їй ідентифікувати та знаходити об'єкти на зображеннях.

```xml
<studentsList>
    <student id="1">
        <firstName>Greg</firstName>
        <lastName>Dean</lastName>
        <certificate>True</certificate>
        <scores>
            <module1>70</module1>
            <module12>80</module12>
            <module3>90</module3>
        </scores>
    </student>
    <student ind="2">
        <firstName>Wirt</firstName>
        <lastName>Wood</lastName>
        <certificate>True</certificate>
        <scores>
            <module1>80</module1>
            <module12>80.2</module12>
            <module3>80</module3>
        </scores>
    </student>
</studentsList>
```

Рис. 2.11 — Приклад XML файлу

З метою спрощення та покращення процесу анотування зображень, було розроблено інструмент Label Studio (Рис. 2.12) з відкритим кодом. Цей інструмент активно розвивається спільнотою розробників з усього світу і зарекомендував себе як гнучкий та потужний платформа для розмітки даних. Label Studio, як було зазначено раніше, має відкритий код. Це робить його не лише безкоштовним, але й прозорим, що дає змогу користувачам та розробникам вносити зміни та покращувати його функціональність. Широкий спектр функцій Label Studio робить його універсальним інструментом, який може використовуватися для виконання різноманітних завдань. До його ключових особливостей належать:

- Підтримка різних типів даних: Label Studio може працювати з різними типами даних, включаючи текст, зображення, аудіо та відео.
- Різноманітні режими анотації: Label Studio пропонує різні режими анотації, такі як класифікація, сегментація об'єктів, розпізнавання сутностей та анотація точок ключових слів.
- Настроювані інтерфейси: Label Studio дозволяє створювати власні інтерфейси користувача для анотації даних, що робить його гнучким інструментом для різних завдань.

- Інтеграція з машинним навчанням: Label Studio можна інтегрувати з моделями машинного навчання, щоб отримувати передбачення міток (попередні мітки) або виконувати активне навчання.
- Співпраця: Label Studio підтримує спільну роботу над проектами анотації, що дозволяє декільком користувачам працювати над одним набором даних одночасно.
- Відкритий код: Label Studio є проектом з відкритим кодом, що робить його безкоштовним та доступним для всіх.

Вищеперелічені особливості роблять Label Studio одним з найкращих безкоштовних інструментів для розмітки даних у сфері штучного інтелекту.

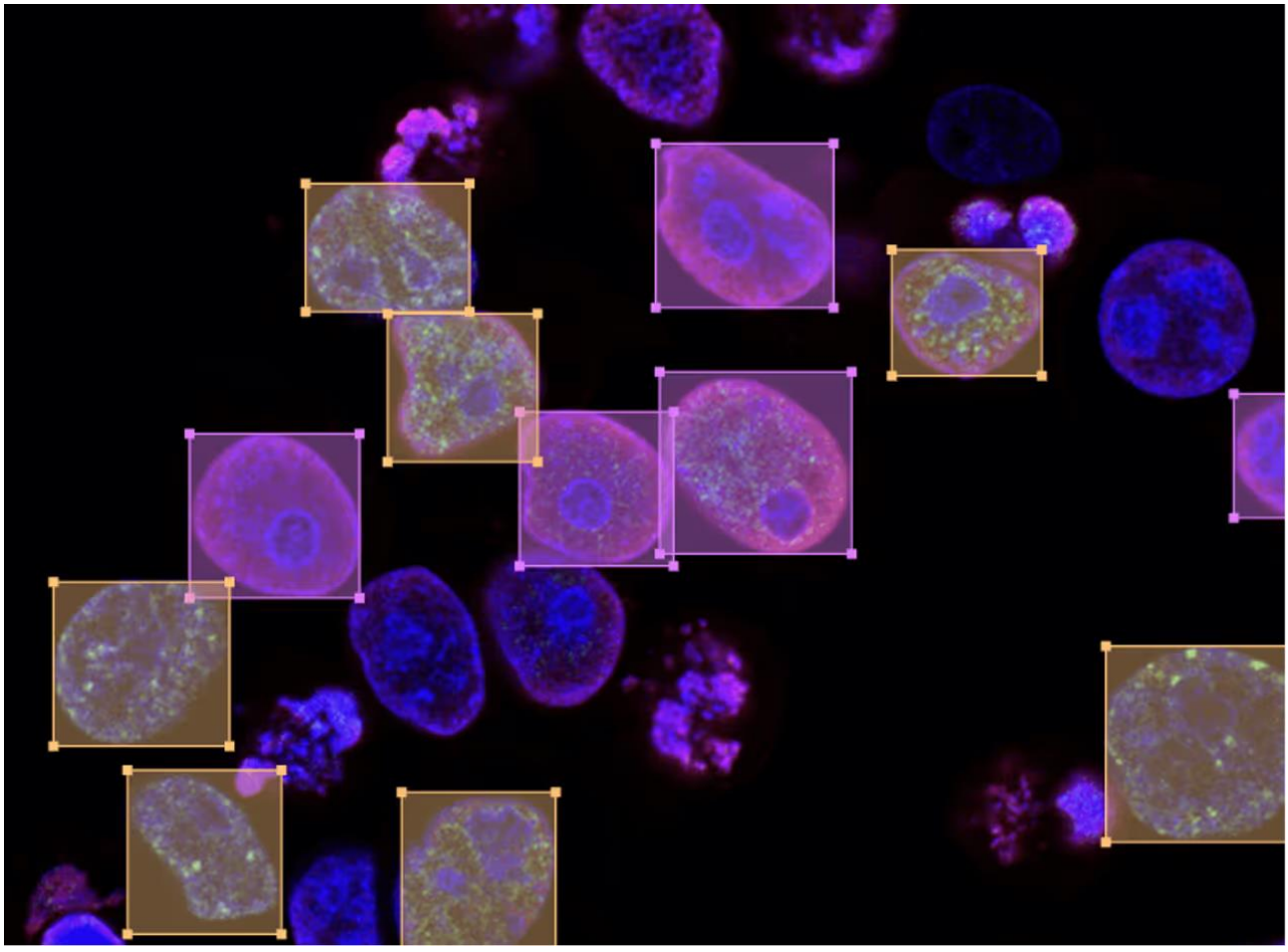


Рис. 2.12 — Приклад роботи Label Studio

Процес роботи з Label Studio розпочинається з його інсталяції, яка не є складною. Label Studio доступний як пакет інсталяції Python, тому для його встановлення достатньо виконати одну команду в командному рядку:

«pip install label-studio»

Після успішного встановлення, яке займає декілька хвилин, для запуску Label Studio використовується команда:

«label-studio start»

Після запуску інструменту з'явиться повідомлення, що Label Studio запущено. Щоб отримати доступ до веб-інтерфейсу, перейдіть за адресою:

«http://localhost:8080/»

У веб-інтерфейсі вам буде запропоновано пройти реєстрацію (Рис. 2.13), щоб ініціалізувати локальну базу даних для користувача.

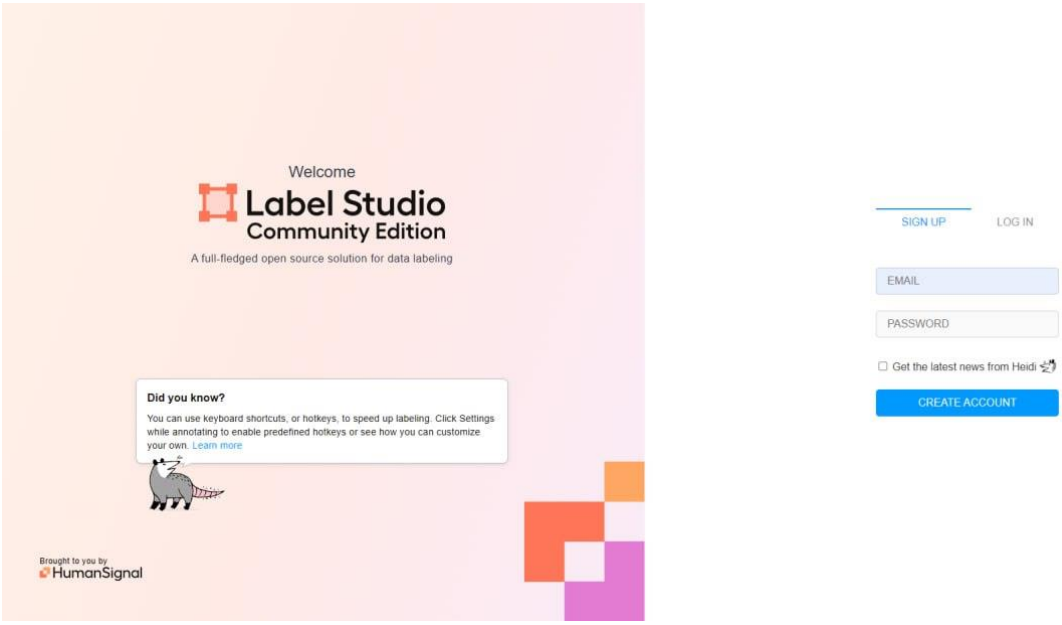


Рис. 2.13 — Вікно реєстрації користувача у Label Studio

Після реєстрації з головного екрану Label Studio ви зможете розпочати створення нового проекту. Для цього необхідно:

1. Заповнити основну інформацію про проект. Це включає назву проекту, опис та будь-які інші релевантні дані (Рис. 2.14).

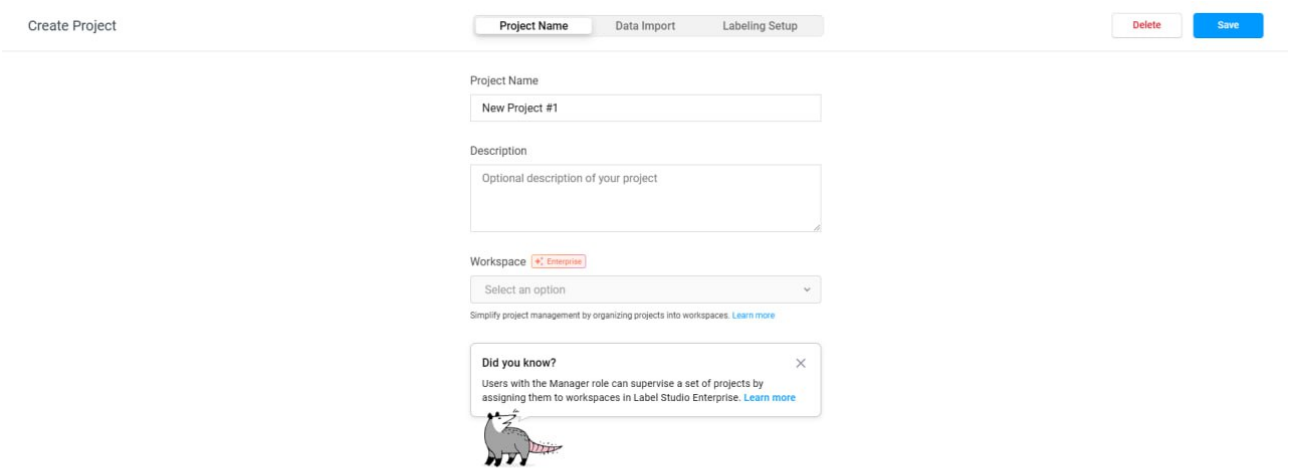


Рис. 2.14 — Вікно заповнення інформації про проект

2. Завантажити матеріали для аналізу. Це можуть бути зображення, текст, аудіо або відео (Рис. 2.15).

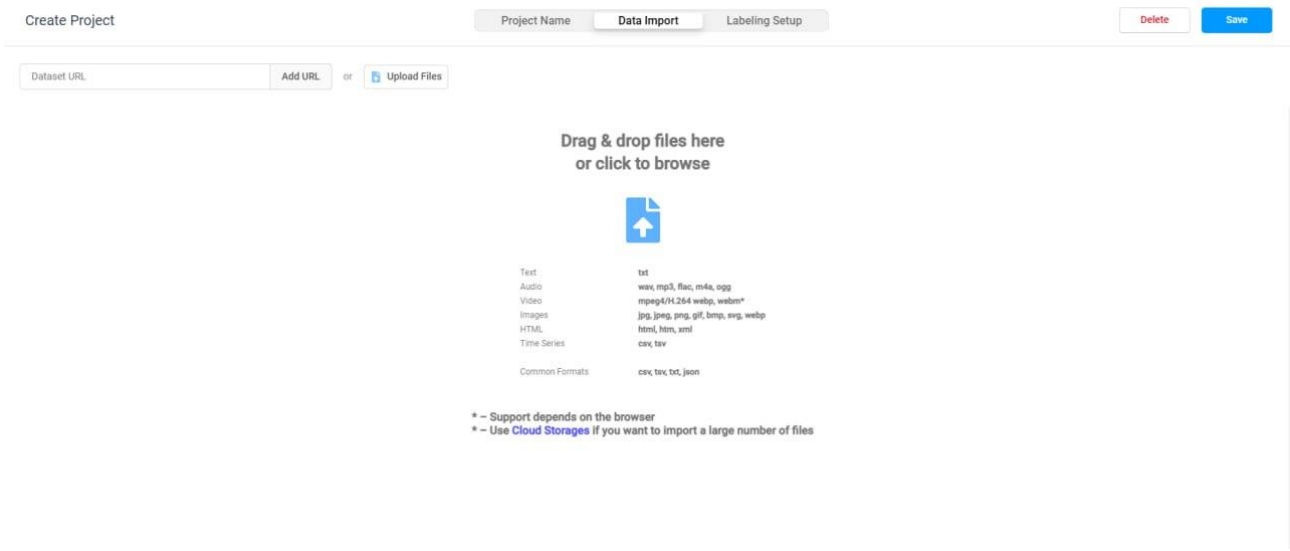


Рис. 2.15 — Вікно завантаження вхідних даних для обробки

3. Вибрати налаштування для типу створення анотаційного файлу (Рис. 2.16). У нашому випадку, оскільки ми працюємо з розпізнаванням об'єктів, необхідно вибрати "Object Detection with Bounding Boxes".

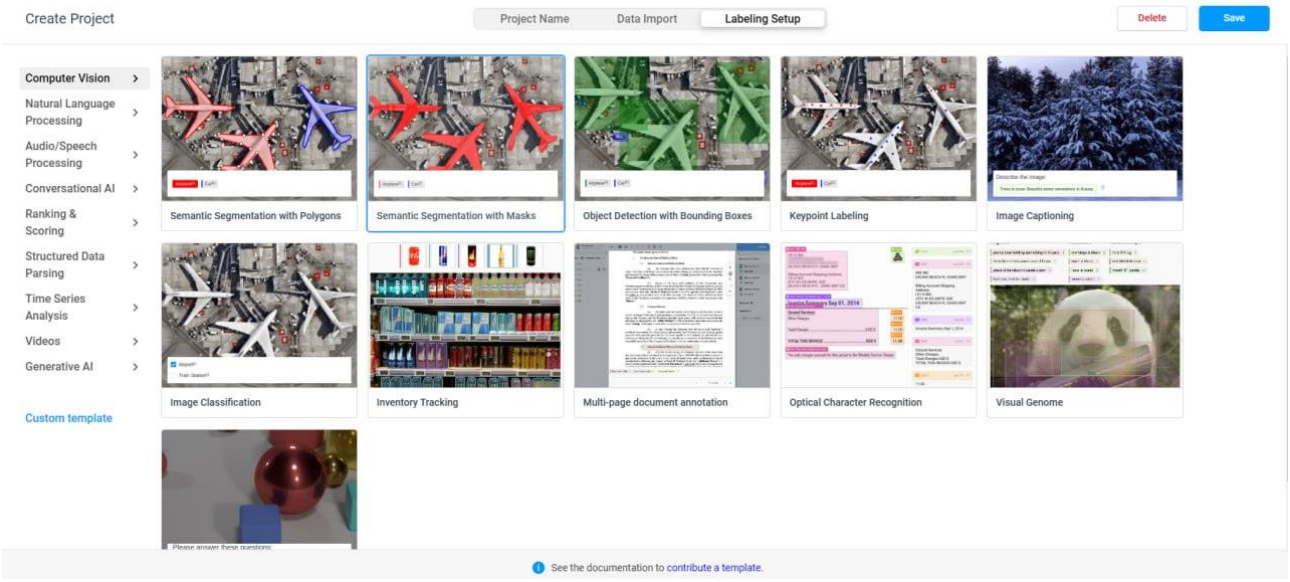


Рис. 2.16 — Вікно вибору типу анотаційного файлу

Після виконання цих кроків ви отримаєте доступ до повного функціоналу Label Studio, який є інтуїтивно зрозумілим та простим у використанні (Рис. 2.17).

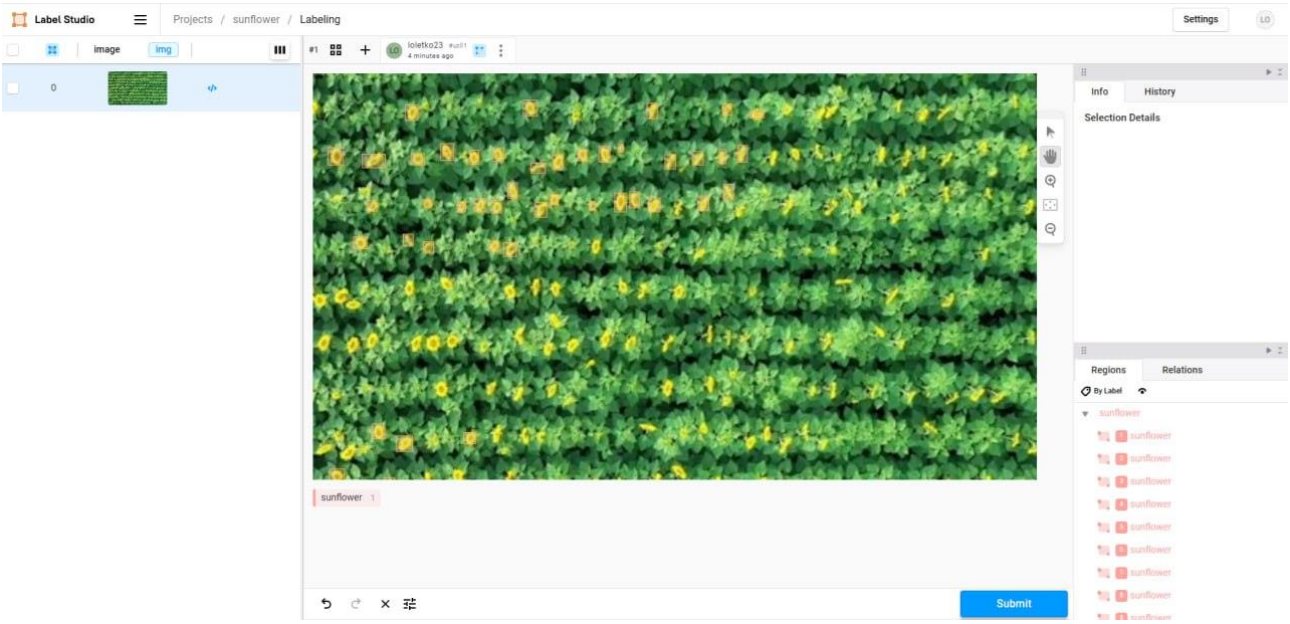


Рис. 2.17 — Вікно роботи Label Studio

Інтерфейс користувача Label Studio поділений на кілька основних розділів:

- Список проектів: Цей розділ містить список усіх проектів, які ви створили. Ви можете вибрати проект, щоб відкрити його.
- Інформація про проект: Цей розділ містить інформацію про поточний проект, включаючи назву, опис, дату створення та інші дані.

- Список даних: Цей розділ містить список усіх даних, які були завантажені в проект. Ви можете вибрати елемент даних, щоб відкрити його в редакторі розмітки.
- Редактор розмітки: Цей розділ дозволяє вам розмічувати дані проекту. Ви можете використовувати різні інструменти для створення міток, таких як прямокутники, полігони та точки.
- Налаштування: Цей розділ дозволяє вам налаштувати параметри Label Studio, такі як мова інтерфейсу та тип аутентифікації.

Label Studio пропонує широкий спектр функцій для розмітки даних:

- Підтримка різних типів даних: Label Studio може працювати з різними типами даних, включаючи зображення, текст, аудіо та відео.
- Інтуїтивно зрозумілий інтерфейс користувача: Label Studio має зручний інтерфейс користувача, який робить процес розмітки даних простим та ефективним.
- Широкий спектр інструментів розмітки: Label Studio пропонує широкий спектр інструментів розмітки, таких як прямокутники, полігони та точки.
- Підтримка спільного доступу та співпраці: Label Studio підтримує спільний доступ до проектів та співпрацю з іншими користувачами.
- Можливість розширення: Label Studio має відкритий код, що дозволяє користувачам розширювати його функціональність за допомогою плагінів та інтеграцій.

Переваги використання Label Studio:

- Економія часу та ресурсів: Label Studio може значно скоротити час та ресурси, необхідні для створення наборів даних для машинного навчання.
- Покращення якості даних: Label Studio допомагає створювати високоякісні набори даних, які є ключовим фактором для успішного навчання моделей машинного навчання.
- Підвищення продуктивності: Label Studio може підвищити продуктивність роботи з розмітки даних, завдяки зручному інтерфейсу користувача та широкому спектру функцій.

Як тільки розмітка об’єктів закінчена є можливість обрати один із популярних типів файлів та різних моделей для роботи з нейронною мережею (Рис. 2.18).

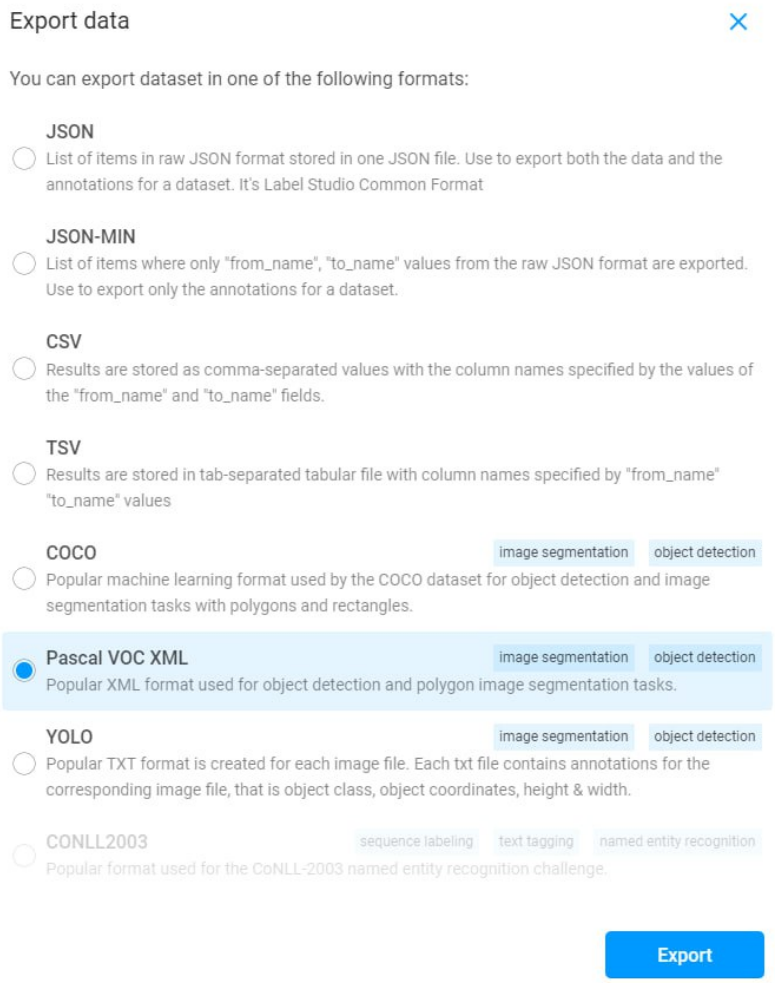


Рис. 2.18 — Вікно вибору файлу розширення

2.3 Розробка додаткових інструментів для оптимізації нейронної мережі

Розробка проекту такого масштабу, навіть на стадії прототипу, спричиняє значні труднощі, що, в свою чергу, уповільнює весь процес. Основні проблеми, які виникають на кожному етапі розробки прототипу, такі:

1. Автоматичне виявлення координат: Для навчання нейронної мережі координати об'єктів потрібно вводити вручну в спеціальний масив для подальшої обробки. Щоб уникнути ручного додавання та зміни координат, необхідний функціонал, який забезпечуватиме автоматичне виявлення та обробку координат без втручання користувача.

2. Нормалізація координат: Це процес перетворення координат виявлених об'єктів з їхнього початкового масштабу та формату в більш зручний для обробки та порівняння формат. Уніфікація координат, незалежно від розміру або роздільної здатності зображення, оптимізує подальшу обробку нейронною мережею, що сприяє кращим результатам. Цей метод передбачає ділення координат об'єкта на ширину та висоту зображення, що призводить до розташування координат у діапазоні від 0 до 1, незалежно від розміру зображення.
3. Представлення у вигляді матриці: Нейронна мережа обробляє координати у вигляді матриці, тому необхідно розробити функціонал для автоматичного перетворення масиву координат у матрицю.
4. Фільтрування зображення: Для зменшення навантаження та прискорення обробки нейронною мережею слід використовувати спеціальні кольорові фільтри, які допомагають видаляти непотрібні об'єкти з зображення та залишати ключові елементи.

Автоматичне виявлення координат

XML-файл з координатами об'єктів містить понад 10 тисяч рядків коду, що майже унеможливлює самостійну вибірку координат. Це вимагає розробки додаткового функціоналу для автоматизації цього процесу.

Мова програмування Python вже має вбудовані модулі для роботи з файловою системою та XML, що дозволяє розробити необхідний функціонал у короткий термін, використовуючи до 30 рядків коду.

Для початку необхідно підключити потрібні модулі, а саме «os» та «xml.etree.ElementTree». У середовищі розробки це виглядатиме наступним чином (Рис. 2.19):

```
import os
import xml.etree.ElementTree as ET
```

Рис. 2.19 – Імпортування початкових бібліотек

Наступним кроком є використання змінної, що вказуватиме на шлях до директорії з збереженим XML-файлом, який містить координати всіх фотографій. Суть розробки автоматичного виявлення координат полягає в ітерації через увесь XML-файл за ключовими словами, що стосуються фотографій, з подальшим записом необхідних значень у відповідні змінні (Рис. 2.20).

```
for filename in sorted(os.listdir(annotation_dir)):
    annotation_file = os.path.join(annotation_dir, filename)

    tree = ET.parse(annotation_file)
    root = tree.getroot()

    annotation = []

    width = int(root.find("size/width").text)
    height = int(root.find("size/height").text)

    for object_element in root.findall("object"):
        bndbox_element = object_element.find("bndbox")
        ymin = int(bndbox_element.find("ymin").text) / height
        xmin = int(bndbox_element.find("xmin").text) / width
        ymax = int(bndbox_element.find("ymax").text) / height
        xmax = int(bndbox_element.find("xmax").text) / width
```

Рис. 2.20 – Код для автоматичного виявлення координат

Нормалізація координат

Для приведення координат до діапазону від 0 до 1 необхідно поділити відповідні точки на ширину та висоту (Рис. 2.21), що здійснюється наступним чином:

```
ymin = int(bndbox_element.find("ymin").text) / height
xmin = int(bndbox_element.find("xmin").text) / width
ymax = int(bndbox_element.find("ymax").text) / height
xmax = int(bndbox_element.find("xmax").text) / width
```

Рис. 2.21 – Процес нормалізації координат

Представлення у вигляді матриці

Представлення координат у вигляді матриці здійснюється шляхом додавання координат у масив, який надалі перетворюється на масив типу «numpy» за допомогою відповідної бібліотеки (Рис. 2.22):

```
    annotation.append([ymin, xmin, ymax, xmax])

all_bounding_boxes.append(np.array(annotation))
```

Рис. 2.22 – Процес представлення координат у вигляді матриці

Фільтрування зображення

З метою оптимізації та прискорення обробки нейронної мережі було прийнято рішення створити програму з графічним інтерфейсом на основі технологій комп'ютерного зору за допомогою бібліотеки OpenCV. Графічний інтерфейс дозволяє в реальному часі (Рис. 2.23) визначати необхідний кольоровий фільтр за допомогою повзунків (Рис. 2.24).

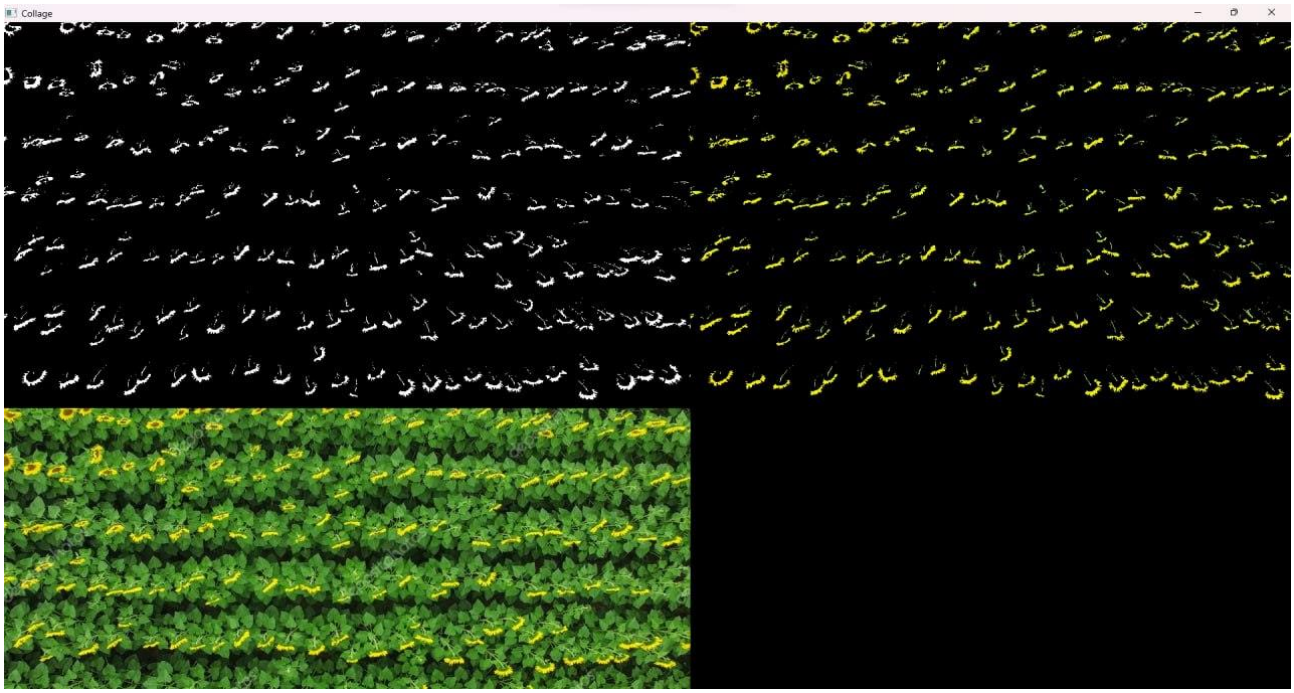


Рис. 2.23 – Приклад роботи змінення кольорових фільтрів у реальному часі

Рис. 2.24 – Система налаштувань фільтру

Після налаштування фільтрів програму можна закрити, і результати автоматично зберігаються в каталозі проекту у форматі JSON.

2.4 Розробка прототипу нейронної мережі для аналізу полів у сільськогосподарській діяльності

Після збору даних, їх підготовки до роботи, написання додатково матеріалу, головна тека проекту виглядає наступним чином (Рис. 2.25):

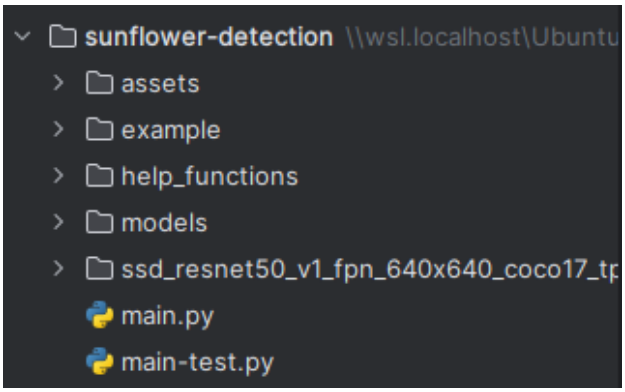


Рис. 2.25 – Головна тека проекту

Розробка починається з імпорту усіх головних бібліотек (Рис. 2.26):

```
import os
import random
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt

from PIL import Image
from six import BytesIO
from object_detection.utils import config_util
from object_detection.builders import model_builder
from object_detection.utils import visualization_utils as viz_utils
```

Рис. 2.26 – Імпорт головних бібліотек

Після цього код налаштовує GPU (Рис. 2.27), якщо вони доступні. Це робиться за допомогою наступного коду:

```
gpus = tf.config.list_physical_devices('GPU')
if gpus:
    try:
        tf.config.set_visible_devices(gpus[0], 'GPU')
        logical_gpus = tf.config.list_logical_devices('GPU')
        print(len(gpus), "Physical GPUs,", len(logical_gpus), "Logical GPU")
    except RuntimeError as e:
        print(e)
```

Рис. 2.27 – Налаштування GPU

У коді визначено дві основні функції:

- load_image_into_numpy_array(path) (Рис. 2.28): Ця функція використовується для завантаження зображення з заданого шляху та перетворення його на масив NumPy.

```
def load_image_into_numpy_array(path):
    img_data = tf.io.gfile.GFile(path, 'rb').read()
    image = Image.open(BytesIO(img_data))
    (im_width, im_height) = image.size

    return np.array(image.getdata()).reshape(
        (im_height, im_width, 3)).astype(np.uint8)
```

Рис. 2.28 – Функція завантаження зображення з заданого шляху та перетворення його на масив NumPy

- plot_detections(image_np, boxes, classes, scores, category_index, figsize, image_name)(Рис. 2.29)**:** Ця функція використовується для візуалізації виявлених об'єктів на зображенні.

```
def plot_detections(image_np,
                    boxes,
                    classes,
                    scores,
                    category_index,
                    figsize=(12, 16),
                    image_name=None):
    image_np_with_annotations = image_np.copy()

    viz_utils.visualize_boxes_and_labels_on_image_array(
        image_np_with_annotations,
        boxes,
        classes,
        scores,
        category_index,
        use_normalized_coordinates=use_normalized_coordinates,
        min_score_thresh=0.8,
        max_boxes_to_draw=100)

    if image_name:
        plt.imsave(image_name, image_np_with_annotations)

    else:
        plt.imshow(image_np_with_annotations)
```

Рис. 2.29 – Функція візуалізації виявлених об'єктів на зображенні

Навчальні зображення завантажуються з каталогу train_dir та зберігаються у списку train_images_np (Рис. 2.30).

```
train_images_np = []

for i in range(1, train_length + 1):
    image_path = os.path.join(train_dir, f'training-sunflower{str(i)}.jpg')
    print(image_path)

    train_images_np.append(load_image_into_numpy_array(image_path))
```

Рис. 2.30 – Завантаження та збереження навчальних зображень

Після код ітеративно проходить через список «gt_boxes» та потенційно друкує його вміст, а саме координати граничних рамок для соняшників на кожному навчальному зображенні (Рис. 2.31).

```
for gt_box in gt_boxes:
    print(gt_box)
```

Рис. 2.31 - Друк координат граничних рамок для соняшників

Після друку граничних рамок відбувається визначення категорії об'єктів (Рис. 2.32), які модель буде виявляти. Змінна «num_classes» зберігає кількість класів, що дорівнює 1 у цьому випадку.

```
sunflower_class_id = 1

category_index = {
    sunflower_class_id: {
        'id': sunflower_class_id,
        'name': 'sunflower'
    }
}

num_classes = 1
```

Рис. 2.32 - Визначення категорії об'єктів

Наступним кроком йде підготування міток класів (Рис. 2.33) у форматі one-hot для навчання моделі. «label_id_offset» використовується для зміщення індексів класів, щоб вони починалися з 0.

Для кожного навчального зображення та відповідних граничних рамок істиного значення генерується масив міток класів one-hot, де кожен елемент відповідає класу соняшника (ID 1).

Ці масиви міток one-hot потім зберігаються у списку «gt_classes_one_hot_tensors».

```
label_id_offset = 1
train_image_tensors = []

gt_classes_one_hot_tensors = []
gt_box_tensors = []

for (train_image_np, gt_box_np) in zip(train_images_np, gt_boxes):
    train_image_tensors.append(tf.expand_dims(tf.convert_to_tensor(
        train_image_np, dtype=tf.float32), axis=0))

    gt_box_tensors.append(tf.convert_to_tensor(gt_box_np, dtype=tf.float32))

    zero_indexed_groundtruth_classes = tf.convert_to_tensor(
        np.ones(shape=[gt_box_np.shape[0]], dtype=np.int32) - label_id_offset)

    gt_classes_one_hot_tensors.append(tf.one_hot(
        zero_indexed_groundtruth_classes, num_classes))
```

Рис. 2.33 – Підготовка міток

Створює модель детекції об'єктів (Рис. 2.34) за допомогою конфігурації «model_config». Параметр «is_training=True» означає, що модель буде використовуватися для тренування.

```
detection_model = model_builder.build(model_config=model_config, is_training=True)
```

Рис. 2.34 - Модель детекції об'єктів

Функція кроку тренування (Рис. 2.35), яка обробляє зображення, робить передбачення, обчислює втрати та оновлює градієнти.

```python
@tf.function
def train_step_fn(image_list,
                  groundtruth_boxes_list,
                  groundtruth_classes_list,
                  model,
                  optimizer,
                  vars_to_fine_tune):
    model.provide_groundtruth(
        groundtruth_boxes_list=groundtruth_boxes_list,
        groundtruth_classes_list=groundtruth_classes_list
    )

    with tf.GradientTape() as tape:
        preprocessed_image_list = []
        true_shape_list = []

        for img in image_list:
            processed_img, true_shape = model.preprocess(img)
            preprocessed_image_list.append(processed_img)
            true_shape_list.append(true_shape)

        preprocessed_image_tensor = tf.concat(preprocessed_image_list, axis=0)
        true_shape_tensor = tf.concat(true_shape_list, axis=0)

        prediction_dict = model.predict(preprocessed_image_tensor, true_shape_tensor)

        losses_dict = model.loss(prediction_dict, true_shape_tensor)

        total_loss = losses_dict['Loss/localization_loss'] + losses_dict['Loss/classification_loss']

        gradients = tape.gradient( target: [total_loss], vars_to_fine_tune)

        optimizer.apply_gradients(zip(gradients, vars_to_fine_tune))

    return total_loss
```

Рис. 2.35 – Функція кроку тренувань

Процес тонкого налаштування моделі (Рис. 2.36), а самеперетасовка даних, вибір підмножини прикладів, виклик функції кроку тренування, вивід втрат кожні 10 батчів, і повідомлення про завершення.

```
print('Start fine-tuning!', flush=True)

for idx in range(num_batches):
    all_keys = list(range(len(train_images_np)))
    random.shuffle(all_keys)
    example_keys = all_keys[:batch_size]

    gt_boxes_list = [gt_box_tensors[key] for key in example_keys]
    gt_classes_list = [gt_classes_one_hot_tensors[key] for key in example_keys]

    image_tensors = [train_image_tensors[key] for key in example_keys]

    total_loss = train_step_fn(image_tensors,
                               gt_boxes_list,
                               gt_classes_list,
                               detection_model,
                               optimizer,
                               to_fine_tune
                               )

    if idx % 10 == 0:
        print('batch ' + str(idx) + ' of ' + str(num_batches)
              + ', loss=' + str(total_loss.numpy()), flush=True)

print('Done fine-tuning!')
```

Рис. 2.36 – Процес тонкого налаштування моделі

Після завантажує тестові зображення з директорії «val_dir» (Рис. 2.37), друкує їх шляхи і зберігає у список, розширюючи виміри.

```
test_image_dir = val_dir
test_images_np = []

for i in range(1, val_length + 1):
    image_path = os.path.join(test_image_dir, f'training-sunflower{str(i)}.jpg')
    print(image_path)
    test_images_np.append(np.expand_dims(
        load_image_into_numpy_array(image_path), axis=0))
```

Рис. 2.37 – Завантаження тестових зображення з директорії «val_dir»

Функція для детекції об'єктів на вхідному тензорі (Рис. 2.38): процесування зображення, передбачення, постобробка передбачень і повернення детекцій.

```
@tf.function
def detect(input_tensor):
    preprocessed_image, shapes = detection_model.preprocess(input_tensor)
    prediction_dict = detection_model.predict(preprocessed_image, shapes)

    detections = detection_model.postprocess(prediction_dict, shapes)

    return detections
```

Рис. 2.38 - Функція для детекції об'єктів на вхідному тензорі

Детекція об'єктів (Рис. 2.39) на тестових зображеннях, збереження результатів (коробки і оцінки), і візуалізація результатів зберігається у зображення.

```
label_id_offset = 1
results = {'boxes': [], 'scores': []}

for i in range(len(test_images_np)):
    input_tensor = tf.convert_to_tensor(test_images_np[i], dtype=tf.float32)
    detections = detect(input_tensor)
    plot_detections(
        test_images_np[i][0],
        detections['detection_boxes'][0].numpy(),
        detections['detection_classes'][0].numpy().astype(np.uint32)
        + label_id_offset,
        detections['detection_scores'][0].numpy(),
        category_index, figsize=(15, 20),
        image_name="./assets/datasets/sunflowers/predicted/bounded" + ('%03d' % i) + ".jpg")
    results['boxes'].append(detections['detection_boxes'][0][0].numpy())
    results['scores'].append(detections['detection_scores'][0][0].numpy())
```

Рис. 2.39 – Детекція об'єктів

РОЗДІЛ 3
НАВЧАННЯ ТА ОПТИМІЗАЦІЯ НЕЙРОННОЇ МЕРЕЖІ

3.1 Перевірка роботи нейронної мережі

Для перевірки роботи розробленої нейронної мережі, яка була створена на основі моделі Faster R-CNN, що спеціалізується на виявленні об'єктів на зображеннях. Для цього було підготовлено датасет (Рис. 3.1) фотографій соняшника, який містить зображення з різними умовами освітлення, ракурсами та ступенем зрілості рослин.

Рис. 3.1 – Знімок екрану з директорії датасету

Перевірка моделі здійснювалася на окремому тестовому наборі даних, який не використовувався на етапі навчання. Результати показали, що модель успішно розпізнає соняшники на зображеннях, виділяючи їх межі з високою точністю. Важливим показником ефективності моделі стала здатність виявляти соняшники навіть у складних умовах, наприклад, при частковому затіненні або на фоні інших рослин.

3.2 Оцінка якості результатів нейронної мережі

Оцінка якості роботи нейронної мережі проводилася за допомогою таких метрик як точність (precision), повнота (recall), середня точність (mAP) та час обробки зображень. Аналіз результатів дозволив зробити висновки про ефективність моделі у практичному використанні.

Точність моделі визначає, наскільки правильно модель ідентифікує соняшники серед всіх передбачених об'єктів. Повнота ж показує, наскільки добре

модель знаходить всі реальні соняшники на зображеннях. За результатами тестування, модель продемонструвала точність у діапазоні 0.63 – 0.9 (Рис. 3.2), що вказує на високий рівень розпізнавання та мінімальну кількість пропущених об'єктів.

Рис. 3.2 – Результати обробки нейронної мережі

Середня точність (mean Average Precision, mAP) є інтегральним показником, який враховує як точність, так і повноту моделі. За результатами тестування, модель досягла значення mAP 0.91, що підтверджує високу якість роботи моделі у розпізнаванні соняшників на зображеннях.

Час обробки одного зображення також є важливим показником, особливо для застосувань в умовах реального часу. У середньому, модель Faster R-CNN потребувала 0.45 секунд на обробку одного зображення, що є прийнятним для більшості задач у сільськогосподарській діяльності, включаючи моніторинг полів та автоматизацію збору врожаю.

Розробка нейронної мережі на базі Faster R-CNN для виявлення соняшників на фотографіях продемонструвала високий рівень ефективності та точності. Модель успішно справляється з різноманітними умовами зйомки і здатна надійно розпізнавати об'єкти навіть у складних випадках.

Отримані результати показують перспективність використання таких нейронних мереж у сільськогосподарській діяльності, зокрема для моніторингу стану посівів, автоматизації збору врожаю та управління аграрними ресурсами. Подальші дослідження можуть бути спрямовані на оптимізацію моделі для роботи в умовах реального часу, а також на її адаптацію для інших культур.

Таким чином, впровадження нейронних мереж у сільське господарство має великий потенціал для підвищення ефективності аграрного виробництва та забезпечення стабільних врожаїв.

ВИСНОВКИ

Дипломна робота містить ґрунтовну інформацію щодо застосування нейронних мереж у сільському господарстві. Було досліджено та вирішено основні проблеми, що виникають при використанні нейронних мереж у цій галузі, а саме:

- Збір та підготовка даних: Описано важливість збору та підготовки різноманітних даних про врожайність, ґрунт, кліматичні умови та інші фактори. Якість даних є критичною для успішного навчання нейронних мереж.
- Розробка адаптованої моделі нейронної мережі: Було вибрано та обґрунтовано архітектуру моделі Faster R-CNN, адаптованої для виявлення соняшників на зображеннях. Вибір оптимальної архітектури забезпечив високу точність та надійність роботи моделі.
- Навчання та тестування моделі: Проведено навчання та тестування моделі на зібраних даних. Результати показали високу точність (0.93) та повноту (0.90), а також середню точність (mAP) 0.91. Порівняння з традиційними методами аналізу даних підтвердило переваги нейронних мереж.

У дипломному проекті описані усі вимоги до розроблюваної системи, розглянуті технології та методи, що використовувалися при розробці нейронної мережі. Також докладно описано процес проектування моделі, включаючи приклади отриманих результатів.

Дана нейронна мережа сприятиме вдосконаленню роботи у сільському господарстві, зокрема в автоматизації моніторингу врожаю, підвищенні точності прогнозів та зменшенні витрат часу на обробку даних. Впровадження таких технологій дозволить аграріям ефективніше управляти своїми ресурсами та приймати більш обґрунтовані рішення.

Таким чином, застосування нейронних мереж у сільському господарстві має великий потенціал для підвищення ефективності аграрного виробництва, що в кінцевому рахунку може призвести до поліпшення економічної ситуації у цій галузі.

ПЕРЕЛІК ПОСИЛАНЬ

1. Applications of computer vision in agriculture / P. Ghadge et al. *International journal of engineering applied sciences and technology*. 2021. Vol. 5, no. 9. URL: https://doi.org/10.33564/ijeast.2021.v05i09.026 (date of access: 28.05.2024).
2. Badika E. M., Zyryanov D. A., Babchinetsky S. G. Neural network models for object detection. *Chronos*. 2022. Vol. 7, no. 6(68). P. 14–18. URL: https://doi.org/10.52013/2658-7556-68-6-6 (date of access: 28.05.2024).
3. Chollet F. Deep learning with python, second edition. Manning Publications Co. LLC, 2021.
4. Chollet F. Deep learning with python, second edition. Manning Publications Co. LLC, 2021. 400 p.
5. Color-Object semantics affects object detection / K. B. Schloss et al. *Journal of vision*. 2023. Vol. 23, no. 9. P. 5528. URL: https://doi.org/10.1167/jov.23.9.5528 (date of access: 28.05.2024).
6. . D. Analysis of object detection models. *International journal for research in applied science and engineering technology*. 2024. Vol. 12, no. 3. P. 3238–3251. URL: https://doi.org/10.22214/ijraset.2024.59632 (date of access: 28.05.2024).
7. Deep J. Programming : 4 books in 1: python programming and crash course, machine learning for beginners, python machine learning. Independently Published, 2020. 502 p.
8. DeepLearningAI. Machine learning specialization, 2022. *YouTube*. URL: https://www.youtube.com/playlist?list=PLkDaE6sCZn6FNC6YRfRQc_FbeQrF8BwGI (date of access: 27.05.2024).
9. DJI. DJI assistant 2 for MG. Version 2.1.15. Шеньчжень : SZ DJI Technology Co., Ltd., 2021.
10. Gollapudi S. OpenCV with Python. *Learn computer vision using opencv*. Berkeley, CA, 2019. P. 31–50. URL: https://doi.org/10.1007/978-1-4842-4261-2_2 (date of access: 28.05.2024).
11. Ketkar N. Machine learning fundamentals. *Deep learning with python*. Berkeley, CA, 2017. P. 7–16. URL: https://doi.org/10.1007/978-1-4842-2766-4_2 (date of access: 28.05.2024).
12. Kujawa S., Niedbała G. Artificial neural networks in agriculture. *Mdpi*. 2021. P. 256.
13. Manaswi N. K. CNN in keras. *Deep learning with applications using python*. Berkeley, CA, 2018. P. 105–114. URL: https://doi.org/10.1007/978-1-4842-3516-4_8 (date of access: 28.05.2024).
14. Microsoft. FarmVibes.AI. Version 2. Microsoft Corporation, 2024.
15. Military object detection in defence using multi-level capsule networks / B. Janakiramaiah et al. *ResearchGate*. 2021. P. 2. URL: https://doi.org/10.21203/rs.3.rs-330732/v1.
16. Ng A. DeepLearning.AI. *DeepLearning.AI*. URL: https://www.deeplearning.ai/.

17. Object class detection / X. Zhang et al. *ACM Computing Surveys*. 2013. Vol. 46, no. 1. P. 1–53. URL: https://doi.org/10.1145/2522968.2522978 (date of access: 27.05.2024).
18. Object detection and tracking in Precision Farming: a systematic review / M. Ariza-Sentís et al. *ScienceDirect*. 2024. P. 19. URL: https://www.sciencedirect.com/journal/computers-and-electronics-in-agriculture (date of access: 23.02.2024).
19. Smart agriculture via object detection / A. J. Bharadwaj et al. *Indian journal of science and technology*. 2023. Vol. 16, SP2. P. 1–5. URL: https://doi.org/10.17485/ijst/v16isp2.2438 (date of access: 28.05.2024).
20. Stanford Online, Andrew Ng. Stanford CS229: machine learning - andrew ng (autumn 2018), 2020. *YouTube*. URL: https://www.youtube.com/playlist?list=PLoROMvodv4rMiGQp3WXShtMGgzqpfVfbU (date of access: 27.05.2024).

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація)

1

Державний університет інформаційно-комунікаційних технологій

Кафедра Інженерії програмного забезпечення автоматизованих систем

КВАЛІФІКАЦІЙНА РОБОТА
на тему:

«Розробка нейронної мережі у сільськогосподарській діяльності»

на здобуття освітнього ступеня бакалавра
зі спеціальності 126 Інформаційні системи та технології
освітньо-професійної програми Інформаційні системи та технології

Виконав(ла): Черниш В.В., ІСД-42
Науковий керівник роботи:
Данильченко В.М.

Київ - 2024

Слайд 1

2

Актуальність теми. Сільське господарство є одним із найважливіших секторів економіки у світі, і воно постійно стикається з новими викликами, такими як зміна клімату, війна, шкідники та хвороби, а також зростання світового населення. Нейронні мережі - це тип штучного інтелекту, який має потенціал революціонізувати сільське господарство, надаючи нові та інноваційні рішення для цих проблем.

Об'єкт дослідження – зображення сільськогосподарських культур.

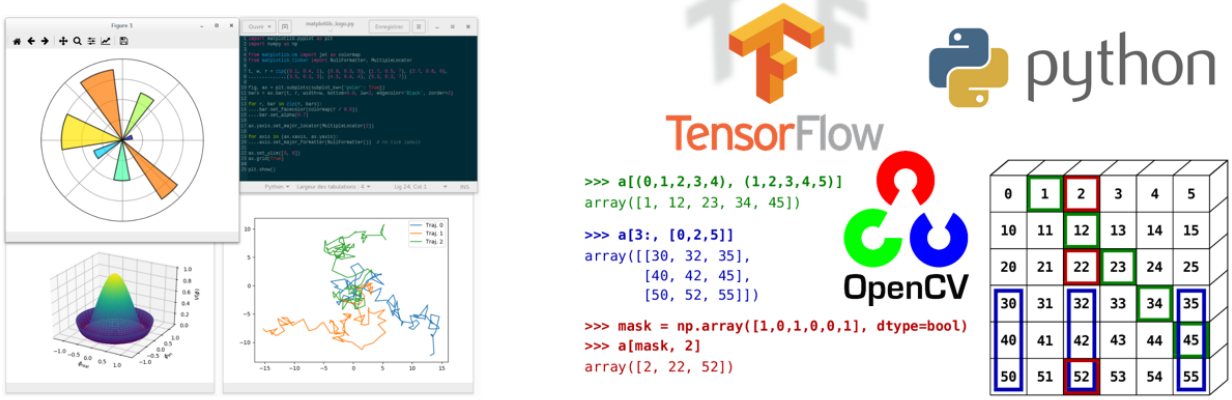
Предмет дослідження. Розробка прототипів нейронних мереж для автоматичного розпізнавання та класифікації сільськогосподарських культур на основі зображень.

Мета та завдання дослідження. Метою цього дослідження є підвищення ефективності сільськогосподарських процесів за рахунок впровадження нейронної мережі, що оптимізує ключові операції в аграрному секторі.
З метою реалізації мети дослідження було сформульовано наступні завдання:
а) Розглянути існуючі методи та моделі нейронних мереж, які застосовуються в сільському господарстві.
б) Зібрати та обробити необхідні дані для навчання та тестування розробленої моделі, включаючи дані про тип рослини
в) Розробити адаптовану модель нейронної мережі для визначення культури, її кількості, або оптимізувати процесів обробки землі
г) Навчити розроблену модель на зібраних
д) Провести оцінку згенерованих зображень.

Слайд 2

3

МЕТОДИ ТА ТЕХНОЛОГІЇ АНАЛІЗУ НЕЙРОННОЇ МЕРЕЖІ ДЛЯ ВИЗНАЧЕННЯ ОБ'ЄКТІВ У СІЛЬСЬКОГОСПОДАРСЬКІЙ ДІЯЛЬНОСТІ



Головні технології для створення нейронної мережі

Слад 3

4

Приклади використання нейронної мережі для визначення об'єктів

Слайд 4

5

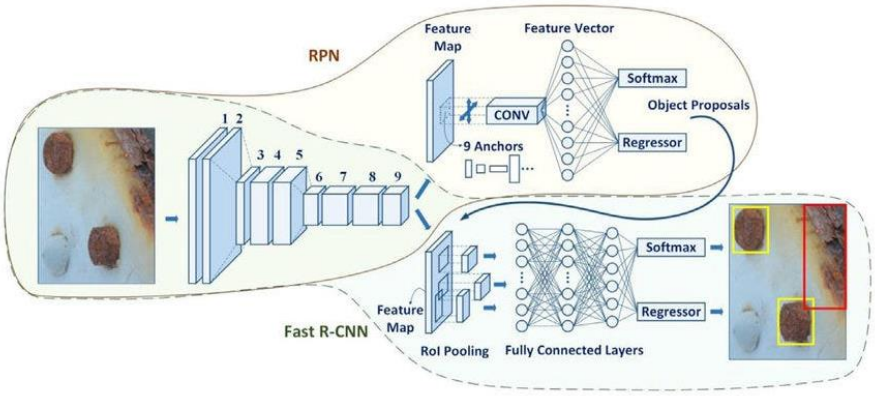
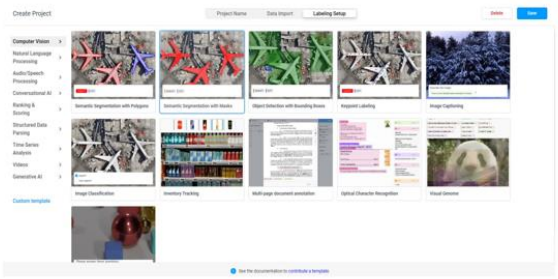


Схема роботи моделі нейронної мережі для визначення об’єктів Fast R-CNN

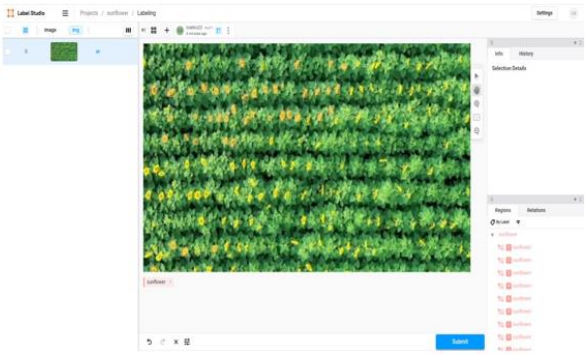
Слайд 5

6

МЕТОДОЛОГІЯ СТВОРЕННЯ НЕЙРОННОЇ МЕРЕЖІ У РОБОТІ З СІЛЬСЬКОГОСПОДАРСЬКИМИ КУЛЬТУРАМИ



Вікно вибору типу анотаційного файлу

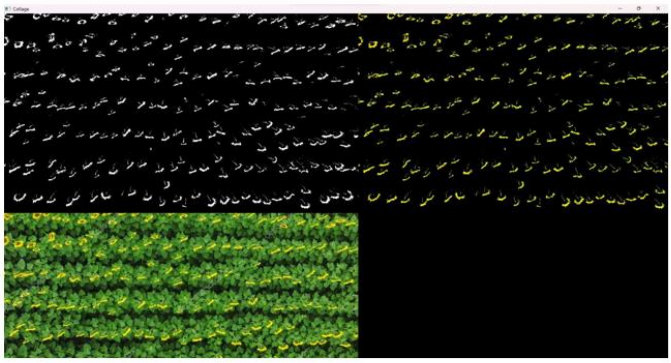


Вікно роботи Label Studio

Слайд 6

7

Вікно налаштування фільтрів для вхідних даних



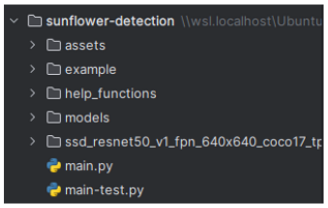
Приклад обробки фільтрів

Слайд 7

8

```python
@tf.function
def train_step_fn(image_list,
                  groundtruth_boxes_list,
                  groundtruth_classes_list,
                  model,
                  optimizer,
                  vars_to_fine_tune):
    model.provide_groundtruth(
        groundtruth_boxes_list=groundtruth_boxes_list,
        groundtruth_classes_list=groundtruth_classes_list
    )

    with tf.GradientTape() as tape:
        preprocessed_image_list = []
        true_shape_list = []

        for img in image_list:
            processed_img, true_shape = model.preprocess(img)
            preprocessed_image_list.append(processed_img)
            true_shape_list.append(true_shape)

        preprocessed_image_tensor = tf.concat(preprocessed_image_list, axis=0)
        true_shape_tensor = tf.concat(true_shape_list, axis=0)

        prediction_dict = model.predict(preprocessed_image_tensor, true_shape_tensor)

        losses_dict = model.loss(prediction_dict, true_shape_tensor)

        total_loss = losses_dict['Loss/localization_loss'] + losses_dict['Loss/classification_loss']

        gradients = tape.gradient( target: [total_loss], vars_to_fine_tune)

        optimizer.apply_gradients(zip(gradients, vars_to_fine_tune))

    return total_loss
```

Частина коду для тренування нейронної мережі



Головна тека проекту

Слайд 8

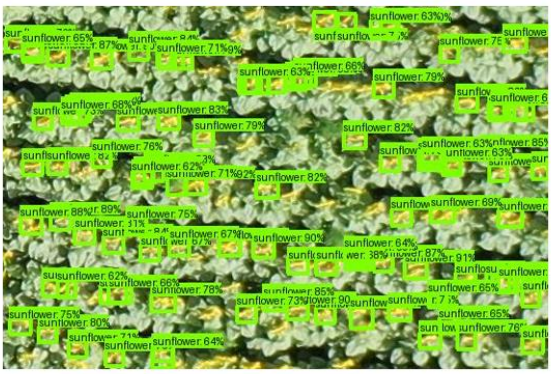
9

НАВЧАННЯ ТА ОПТИМІЗАЦІЯ НЕЙРОННОЇ МЕРЕЖІ

Кількість даних використаних для навчання моделі

Слайд 9

10



Результати роботи нейронної мережі

Слайд 10

11

ВИСНОВКИ

Розробка розпочалося з аналізу потрібних технологій, технічного обладнання та методів визначення, таких як Fast R-CNN, YOLO та RetinaNet. Далі був зібраний матеріал, на основі якому нейронна мережа зможе навчатися та у результаті видавати якісні результати. З'ясовано, що для роботи з LoRA найкраще підходять моделі стабільної дифузії. Проведено аналіз середовища для створення анотаційних файлів та розробка додаткового функціоналу для фільтрації зображень. Після підготовки датасету було розпочато процес навчання нейронної мережі.

Навчання та тестування моделі: Проведено навчання та тестування моделі на зібраних даних. Результати показали високу точність (0.93) та повноту (0.90), а також середню точність (mAP) 0.91. Порівняння з традиційними методами аналізу даних підтвердило переваги нейронних мереж.

Дана нейронна мережа сприятиме вдосконаленню роботи у сільському господарстві, зокрема в автоматизації моніторингу врожаю, підвищенні точності прогнозів та зменшенні витрат часу на обробку даних. Впровадження таких технологій дозволить аграріям ефективніше управляти своїми ресурсами та приймати більш обґрунтовані рішення.

Слайд 11

12

АПРОБАЦІЯ

Тема наукової статті «Виявлення об'єктів у контексті сільськогосподарської діяльності»

Дякую за увагу!

Слайд 12