ДОДАТОК А

Апробація наукових результатів

МОЛОДІЖНА НАУКОВА ЛІГА

Громадська організація «Молодіжна наукова ліга».
Номер запису в Реєстрі громадських об'єднань: 1506433.
Адреса: вул. Зодчих, буд. 40, офіс 103; м. Вінниця, Вінницька обл., 21037
Організація функціонує як відокремлений підрозділ ТОВ «UKRLOGOS Group».
ЄДРПОУ: 44574526
IBAN: UA433052990000026002046104529
Банк ВФ АТ КБ «ПриватБанк»; МФО 44574526
Свідоцтво суб'єкта видавничої справи: ДК № 7860 від 22.06.2023.

Д О В І Д К А

ПРО ПРИЙНЯТТЯ СТАТТІ ДО ПУБЛІКАЦІЇ

09.01.2025

Шановний(і) автор(и):
Давидов Нікіта Денисович,

Редакційний комітет з радістю повідомляє, що стаття «Використання штучного інтелекту для автоматизації роботизованих систем за допомогою штучного інтелекту» прийнята до публікації в № 16 студентського наукового журналу «UNIVERSUM», випуск якого заплановано на 20 січня 2025 року.

Опублікована стаття буде доступна з 20.01.2025 за посиланням:
https://archive.liga.science/index.php/universum/issue/view/january2025

. . . . . .

Електронні сертифікати про публікацію та подяки науковим керівникам також будуть доступні з 20 січня. Розсилка замовлених друкованих примірників, сертифікатів та подяк відбудеться з 3 по 10 лютого.

З повагою,

Директор Молодіжної наукової ліги
Голова редакційного комітету
ІГОР КОРЕНЮК

МОЛОДІЖНА НАУКОВА ЛІГА

Використання штучного інтелекту для автоматизації роботизованих систем за допомогою штучного інтелекту

Давидов Нікіта Денисович

здобувач вищої освіти факультету автоматики і комп'ютеризованих систем

Національний технічний університет «Харківський національний університет радіоелектроніки», Україна

Науковий керівник: Чала Олена Олександрівна

доцент кафедри КІТАР, кандидат технічних наук, доцент

Національний технічний університет «Харківський національний університет радіоелектроніки», Україна

У статті розглядається розробка автоматизації для роботів із використанням штучного інтелекту, акцентуючи увагу на тренуванні моделей машинного навчання та їх інтеграції в роботизовані системи. Основна мета таких технологій полягає в адаптації роботів до складних і змінних умов роботи через аналіз даних, отриманих із сенсорів, прийняття рішень та виконання завдань у реальному часі. У дослідженні представлено математичні основи моделей, що використовуються в автоматизації, включаючи функції втрат, алгоритми оптимізації та методи посиленого навчання.

Ключові слова: ONNX, моделі штучного інтелекту, TensorFlow

Розробка автоматизації для роботів із використанням штучного інтелекту ґрунтується на інтеграції алгоритмів машинного навчання, які дозволяють роботам сприймати середовище, приймати рішення та виконувати завдання. Це досягається через обробку даних, отриманих із сенсорів, таких як камери, лідарами чи мікрофони, з подальшою інтерпретацією інформації для дій у реальному часі. Для цього необхідно створити моделі, здатні навчатися

розпізнавати закономірності в даних, будувати прогнози та адаптувати поведінку робота.

Тренування моделей передбачає оптимізацію функцій втрат. Наприклад, у задачах класифікації часто використовується крос-ентропійна функція, яка визначає різницю між прогнозом моделі та реальними даними. Вона обчислюється як сума негативних логарифмів імовірностей, передбачених для правильних класів. Мінімізація цієї функції здійснюється через градієнтний спуск, коли параметри моделі оновлюються на основі обчисленого градієнта, що визначає напрямок найшвидшого зменшення помилки. Наприклад, формула оновлення параметрів виглядає так:

$$\theta \leftarrow \theta - \eta \nabla_\theta L(\theta), \quad (1)$$

де:

η *– швидкість навчання;*

$\nabla_\theta L(\theta)$ *– градієнт функції втрат.*

У випадку посиленого навчання робот вчиться шляхом взаємодії зі своїм середовищем, отримуючи нагороди або штрафи залежно від результатів своїх дій. Основною метою є максимізація сумарної нагороди, яка враховує як негайні, так і майбутні вигоди. Це описується функцією цінності, що оцінює очікувану нагороду зі стану s за політикою π, виражену рівнянням:

$$V^\pi(s) = E_\pi[\sum_{t=0}^{\infty} \quad = \Upsilon^t r^t \mid s_0 = s\,], \quad (2)$$

де:

Υ *– коефіцієнт дисконтування.*

Алгоритм *Q-learning* дозволяє оновлювати політику на основі спостережень, використовуючи правило:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma max_{\alpha'} Q(s',a') - Q(s,a)], \quad (3)$$

де:

r *– нагорода за поточну дію;*

α *– швидкість навчання.*

Інтеграція моделей у роботів потребує оптимізації, щоб забезпечити виконання складних обчислень на обмежених апаратних ресурсах. Для цього моделі конвертуються в легкі формати, такі як TensorFlow Lite чи ONNX.

ONNX (Open Neural Network Exchange) та TensorFlow Lite є форматами, що дозволяють розгортати моделі машинного навчання на різних платформах, зокрема вбудованих системах, мобільних пристроях чи серверному середовищі. Вони вирішують задачі ефективного перенесення моделей між фреймворками і їхньої оптимізації для швидкої роботи на обмежених ресурсах.

ONNX — це відкритий стандарт, який забезпечує сумісність між різними фреймворками машинного навчання, такими як PyTorch, TensorFlow, Scikit-learn тощо. Математично ONNX представляє модель як граф із вузлами, що відповідають обчисленням, наприклад, матричному множенню чи згортці. Кожен вузол описує операцію через набір параметрів і входів. Наприклад, згорткова операція:

$$y[i,j,k] = \sum_{m,n,p} W[m,n,p] \cdot x[i+m, j+n, k+p] + b[k], \qquad (4)$$

де:

$y[i, j, k]$ – вихідний тензор після згортки в точці;

$x[i+m,\ j+n,\ k+p]$ – значення вхідного тензора з урахуванням зміщення фільтра;

$W[m,n,p]$ – елементи фільтра (ядра згортки);

$b[k]$ – зміщення (bias) для каналу k;

m, n, p – індекси фільтра.

у форматі ONNX буде задана як вузол із входами *x* (вхідний тензор), *W* (фільтр) і *b* (зміщення), а також параметрами, такими як розмір фільтра, кількість каналів та крок. Модель, описана в ONNX, оптимізується для виконання обчислень на графічних або центральних процесорах, спрощуючи її розгортання на різноманітних пристроях.

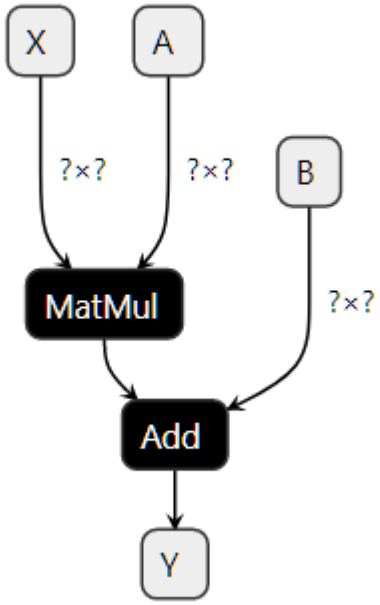

Рис. 1. Концепція формата моделі ONNX

TensorFlow Lite фокусується на зменшенні обсягу моделей і підвищенні їхньої ефективності на пристроях із низькою потужністю. Формат використовує квантизацію, яка зменшує розрядність чисел у вагових параметрах із 32-бітних до 8-бітних цілочислових. Наприклад, ваги, які зазвичай зберігаються як числа з плаваючою точкою, перетворюються на цілочислові представлення із масштабним коефіцієнтом *s*, як представлено формулою:

$$W = W' \cdot s, \tag{5}$$

де:

W – вихідна (оригінальна) матриця ваг;

W' – квантизована (цілочислова) матриця ваг;

s – масштабний коефіцієнт (scale), який визначає точність перетворення.

що дозволяє відновити ваги у вигляді:

$$L = \frac{1}{N}\sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \quad (6)$$

де:

y_i *– реальні значення;*

$\hat{y}_i$ *– передбачення моделі;*

N – кількість прикладів у вибірці.

Це значно зменшує обсяг пам'яті та прискорює обчислення за рахунок використання інструкцій для роботи з цілими числами.

Обидва формати використовують графи обчислень для представлення моделі. Наприклад, нехай модель виконує операцію матричного множення для передбачення:

$$y = Wx + b, \quad (7)$$

де:

W – матриця ваг;

x – вхідний вектор;

b – вектор зміщення.

У TensorFlow Lite чи ONNX граф складається з вузлів для матричного множення та додавання. Оптимізація у TensorFlow Lite включає попереднє обчислення констант, що дозволяє скоротити кількість операцій у реальному часі, тоді як ONNX забезпечує перетворення операцій у вигляді, сумісному з різними апаратними прискорювачами.

Ці формати дозволяють створювати компактні та високопродуктивні моделі, зберігаючи точність і полегшуючи інтеграцію штучного інтелекту в реальні застосунки.

Процес автоматизації дозволяє поєднувати точність класичних методів керування, таких як PID-регулятори, із гнучкістю та адаптивністю алгоритмів штучного інтелекту. Це забезпечує створення роботів, які здатні ефективно адаптувати свою поведінку до мінливих умов середовища, гарантуючи безпеку та високу продуктивність.

Список використаних джерел:

1. Топузов О., Алєксєєва С. Можливості використання штучного інтелекту в освітньому процесі закладів середньої освіти в умовах воєнного стану – 2024. – № 1. – С. 5–11.

2. Мельник О. В. Використання програм зі штучним інтелектом у загальній середній освіті: переваги та ризики – 2024. – № 228. – С. 31–44.

3. Мар’єнко М., Беркещук І. С., Ван Цяньці. Штучний інтелект та ефективність його використання в освіті – 2023. – № 12. – С. 88–99.

4.Порівняння статичних і динамічних методів візуалізації у навчанні алгоритмів.

5. Спецкурс «Основи машинного навчання для робототехніки».

6. Документація концепцій ONNX.

7. Розробка автоматизації роботизованих систем за допомогою штучного інтелекту

ДОДАТОК Б

Код програм, використаних у проєкті

Код обробки пакетів з пульта управління

```
import struct
# Функція для обчислення CRC8
def calculate_crc8(data):
crc = 0x00
    for byte in data:
        crc ^= byte
        for _ in range(8):
            if crc & 0x80:
                crc = (crc << 1) ^ 0x07
            else:
                crc <<= 1
            crc &= 0xFF
    return crc

# Функція для перевірки CRC8 пакету
def verify_crc8(packet):
        packet_without_crc = packet[:-1]
    calculated_crc = calculate_crc8(packet_without_crc)
    packet_crc = packet[-1]
    return calculated_crc == packet_crc

# Перетворення байтів каналу на значення
def convert_bytes_to_channel(byte1, byte2):
    # Відновлюємо 11 біт з 2 байтів
    channel_value = (byte1 << 3) | (byte2 >> 5)
```

```
    return channel_value

# Функція для розбору CRSF пакету
def parse_crsf_packet(packet):
    if len(packet) != 39:
raise ValueError("Invalid packet length")

    sync = packet[:2]
    if sync != [0xA5, 0x5A]:
        raise ValueError("Invalid sync bytes")

    packet_type = packet[2]
    packet_id = packet[3]

    # Канали
    channels = []
    for i in range(16):
        byte1 = packet[4 + i * 2]
        byte2 = packet[5 + i * 2]
        channel_value = convert_bytes_to_channel(byte1, byte2)
        channels.append(channel_value)

    # Контрольна сума (CRC)
    crc = packet[-1]

    # Перевірка CRC8
```

```
        if not verify_crc8(packet):
            raise ValueError("CRC8 mismatch")

        return packet_type, packet_id, channels, crc
```

Код конвертації форматів пакету

```
    def parse_package(package):
        if len(package) != 19:
        raise ValueError("Неправильна довжина пакету. Очікується 19 байтів.")

        # Перевіряємо синхронізаційні байти
        if package[0] != 0x18 or package[1] != 0x16:
            raise ValueError("Неправильні синхронізаційні байти.")

        # Зчитуємо значення каналів (11 біт на канал, 2 байти на канал)
        channels = []
        for i in range(2, 18, 2):  # Канали починаються з 2-го байту
            high_byte = package[i]  # Старший байт
            low_byte = package[i + 1]  # Молодший байт

            # Комбінуємо байти у 11-бітове значення
            channel_value = ((high_byte << 8) | low_byte) & 0x07FF  # Маска для 11 біт
            channels.append(channel_value)

        crc_received = package[-1]
        crc_calculated = calculate_crc8(package[:-1])
```

```
            if crc_received != crc_calculated:
                raise ValueError(f"CRC8 не збігається. Очікувалося: {crc_calculated}, отримано: {crc_received}")



            result = ", ".join([f"ch{i:02}=\"{value}" for i, value in enumerate(channels)])
            return result


        def calculate_crc8(data):
            crc = 0
            for byte in data:
                crc ^= byte
                for _ in range(8):
                    if crc & 0x80:
                        crc = (crc << 1) ^ 0x07
                    else:
                        crc <<= 1
                    crc &= 0xFF
            return crc
```

Код модифікації пакету за допомогою даних моделі

```
from ultralytics import YOLO

model = YOLO('best.pt')
while True:
image_path = 'image.jpg'
```

```
        results = model(image_path)
        for result in results:
            boxes = result.boxes.xyxy
            scores = result.boxes.conf
            labels = result.boxes.cls
        ch_min, ch_max = 900, 2100
        ch_center = 992
        k_x, k_y = 500
        # Розміри кадру
        frame_width, frame_height = 1280, 720


        x_min, y_min, x_max, y_max = 400, 300, 800, 600
        x_frame_center, y_frame_center = frame_width / 2, frame_height / 2
        x_center = (x_min + x_max) / 2
        y_center = (y_min + y_max) / 2


        # Нормалізоване зміщення
        delta_x_norm = (x_center - x_frame_center) / frame_width
        delta_y_norm = (y_center - y_frame_center) / frame_height


        ch_yaw = ch_center + k_x * delta_x_norm
        ch_pitch = ch_center + k_y * delta_y_norm
        ch_yaw = max(ch_min, min(ch_max, ch_yaw))
        ch_pitch = max(ch_min, min(ch_max, ch_pitch))
        packet = f"ch00={int(ch_yaw)}, ch01={int(ch_pitch)}, " + ",
".join([f"ch{i:02}=992" for i in range(2, 16)])
```

Код для відправки пакету у дрон

```
import struct
def calculate_crc8(data):
    crc = 0x00
    for byte in data:
        crc ^= byte
        for _ in range(8):
            if crc & 0x80:
                crc = (crc << 1) ^ 0x07
            else:
                crc <<= 1
            crc &= 0xFF
    return crc

def verify_crc8(packet):
    packet_without_crc = packet[:-1]
    calculated_crc = calculate_crc8(packet_without_crc)
    packet_crc = packet[-1]
    return calculated_crc == packet_crc

def convert_bytes_to_channel(byte1, byte2):
    # Відновлюємо 11 біт з 2 байтів
    channel_value = (byte1 << 3) | (byte2 >> 5)
    return channel_value

def parse_crsf_packet(packet):
```

```
# Перевірка довжини пакету
if len(packet) != 39:
    raise ValueError("Invalid packet length")

sync = packet[:2]
if sync != [0xA5, 0x5A]:
    raise ValueError("Invalid sync bytes")

packet_type = packet[2]
packet_id = packet[3]

channels = []
for i in range(16):
    byte1 = packet[4 + i * 2]
    byte2 = packet[5 + i * 2]
    channel_value = convert_bytes_to_channel(byte1, byte2)
    channels.append(channel_value)
crc = packet[-1]
if not verify_crc8(packet):
    raise ValueError("CRC8 mismatch")
return packet_type, packet_id, channels, crc
```

ДОДАТОК В

Демонстраційний матеріал

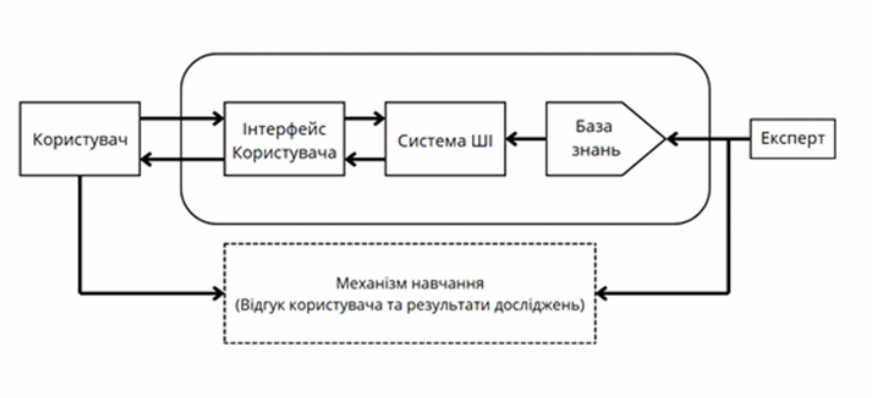

Рисунок А.1 – Схема експертної системи

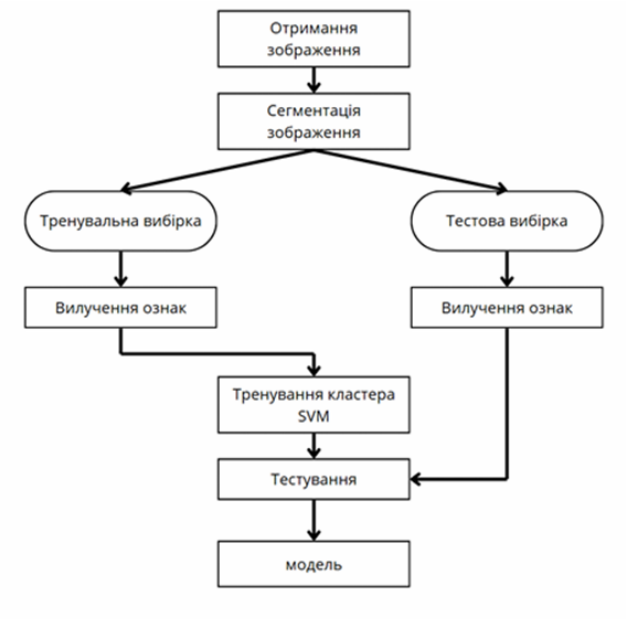

Рисунок А.2 – Приклад машинного навчання моделі комп’ютерного зору

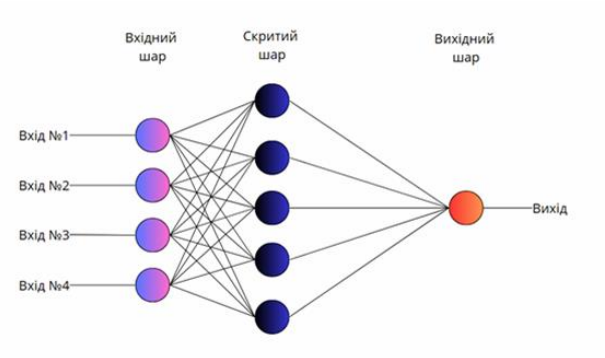

Рисунок 2.3 – Приклад шарів нейронної мережі

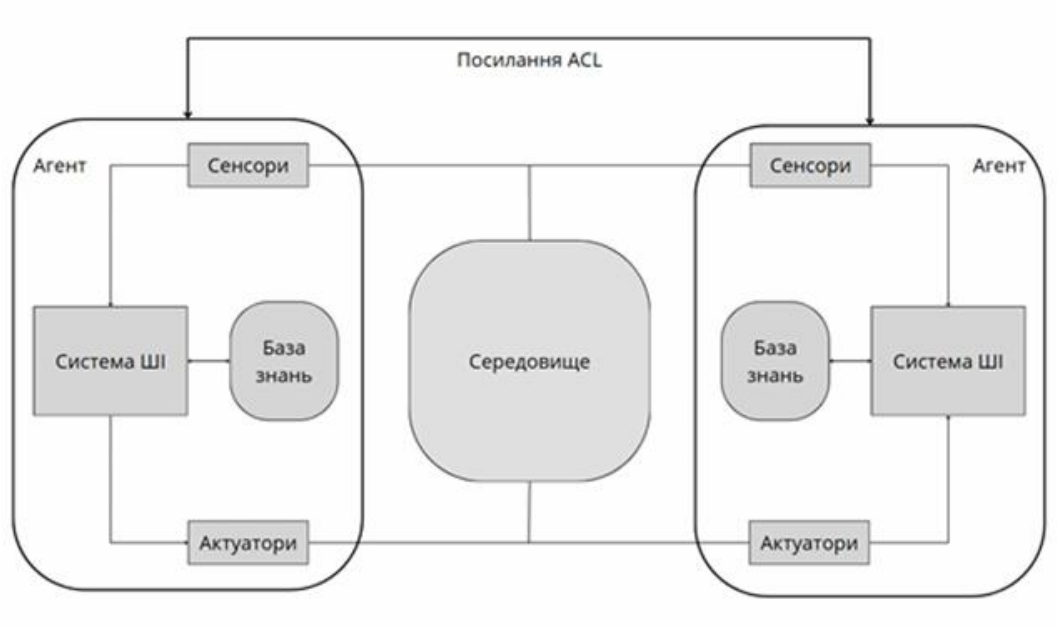

Рисунок А.3 – Приклад багатоагентної системи

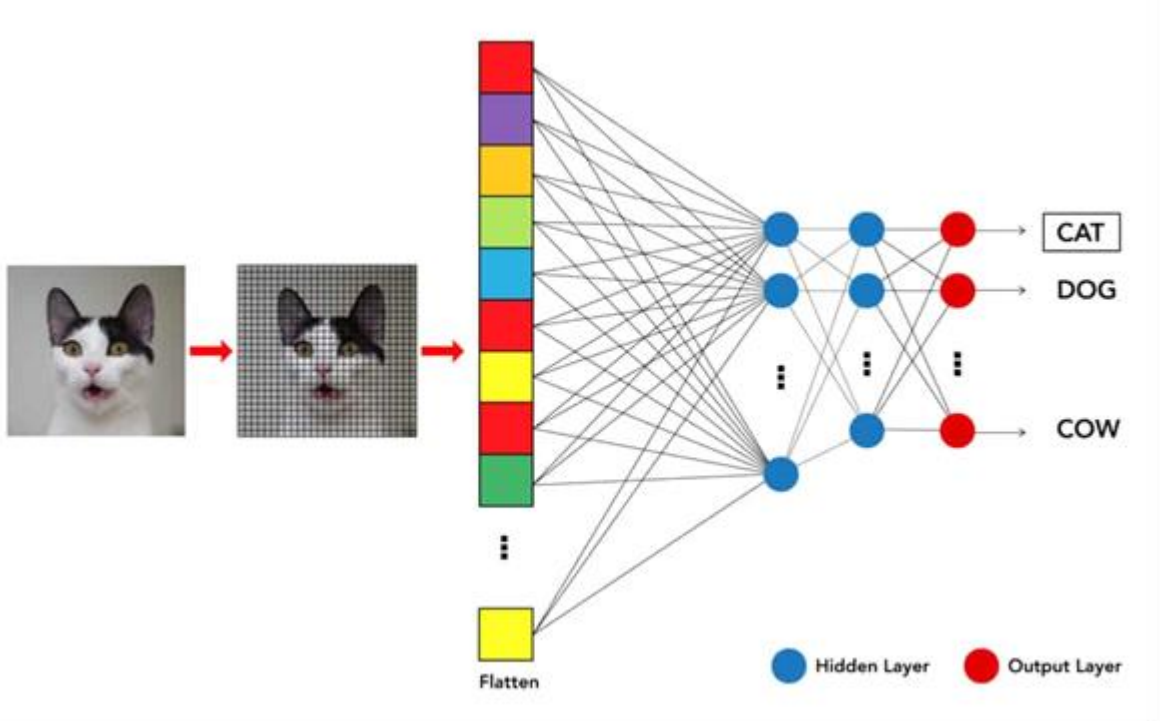

Рисунок А.4 – приклад алгоритму розпізнавання образів

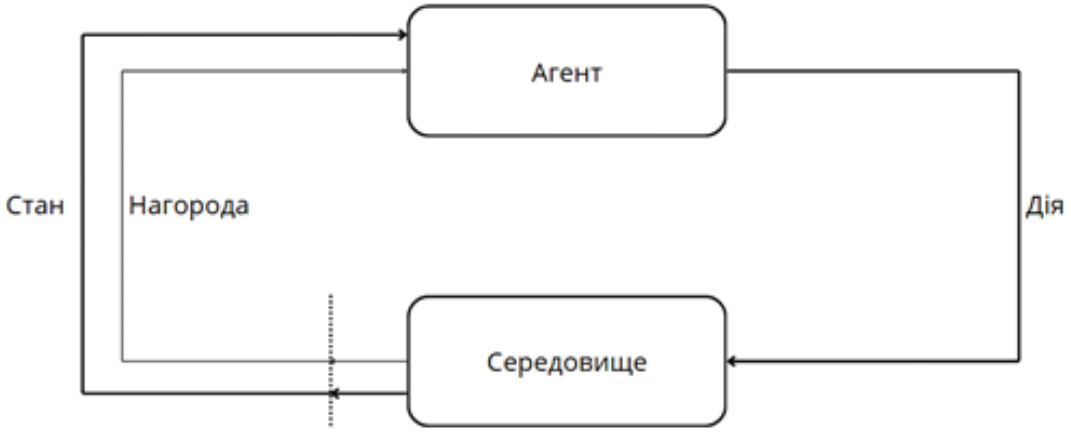

Рисунок А.5 – схема підсилювального навчання

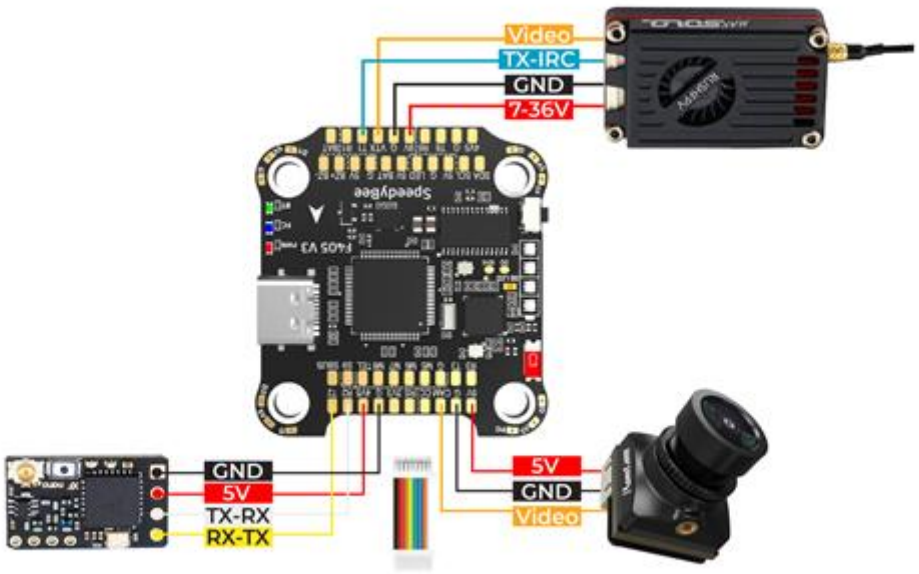

Рисунок А.7 – Схема розпайки компонентів FPV дрона

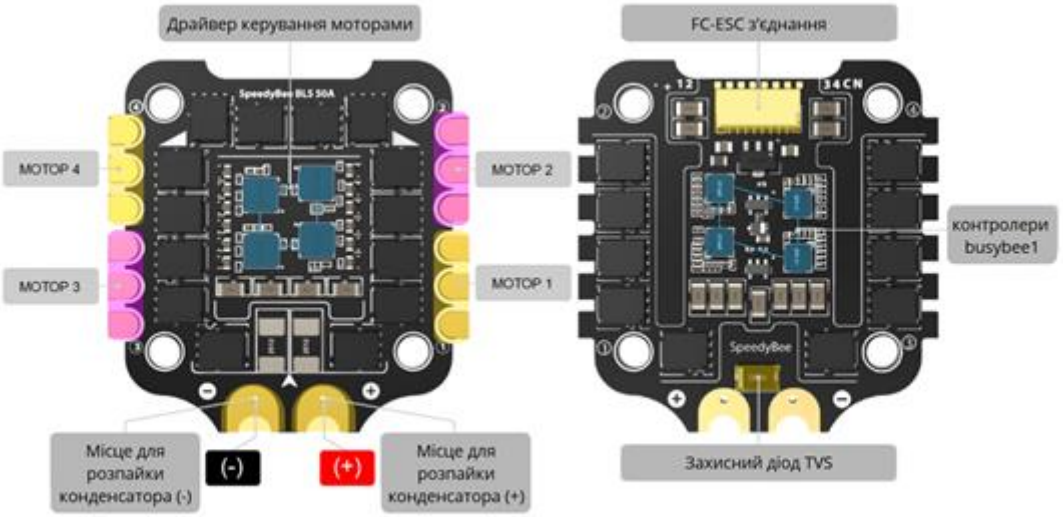

Рисунок А.8 – Схема регулятора моторів

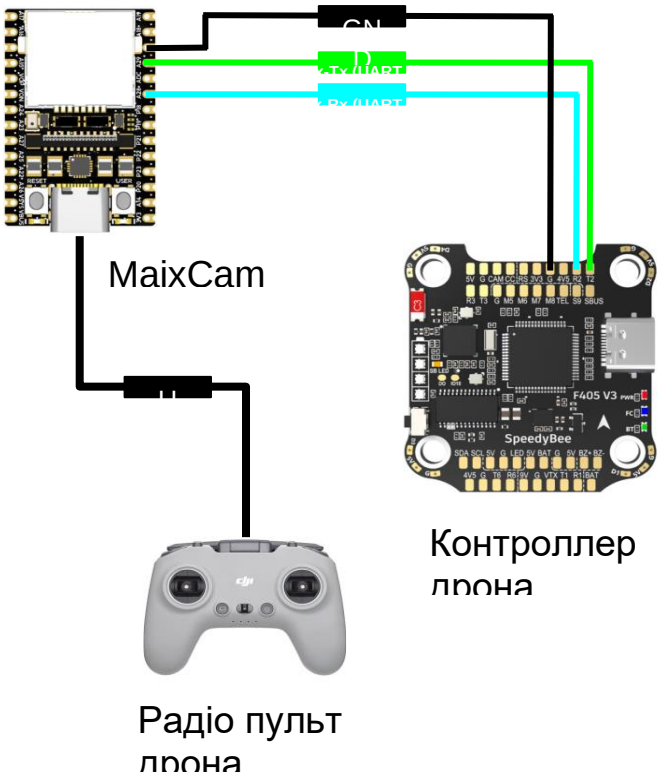

Рисунок А.9 – Розпіновка з’єднання MaixCam та контроллера дрона

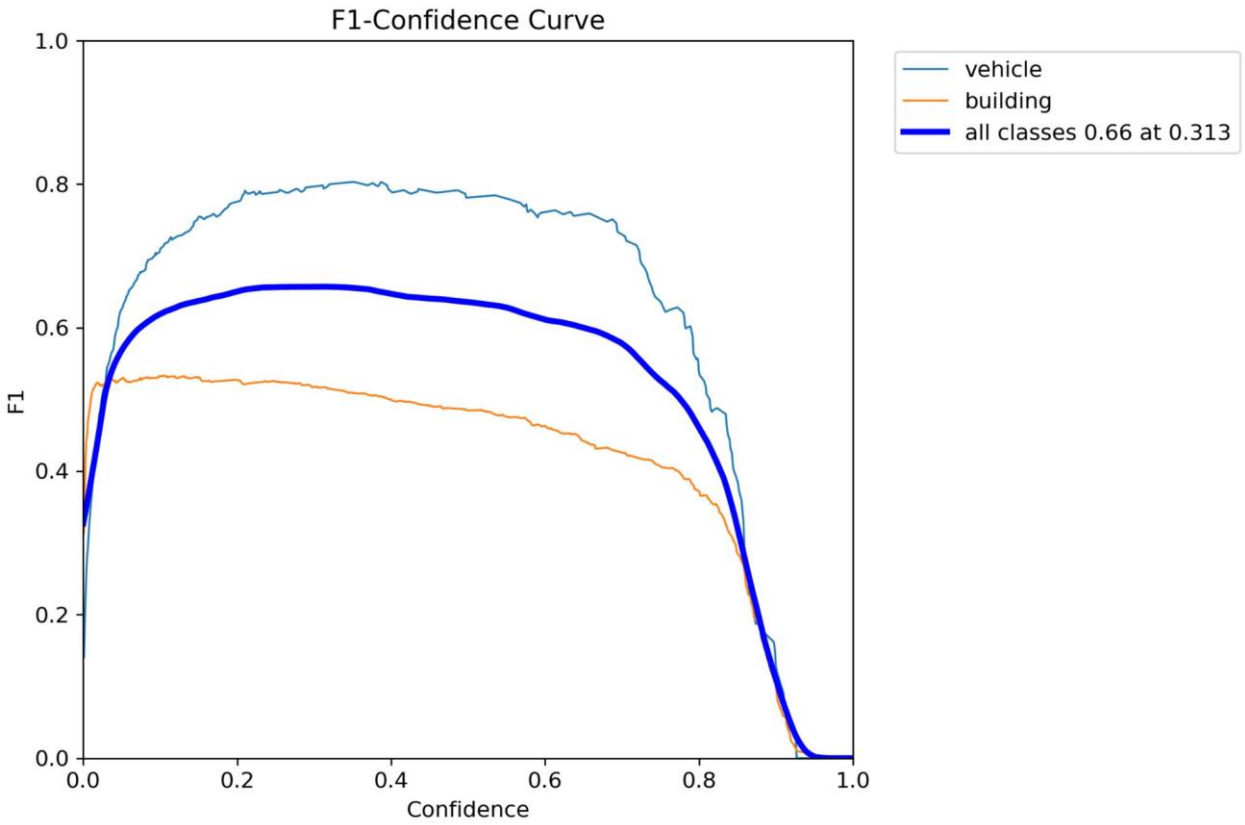

Рисунок А.10 – графік впевненості моделі

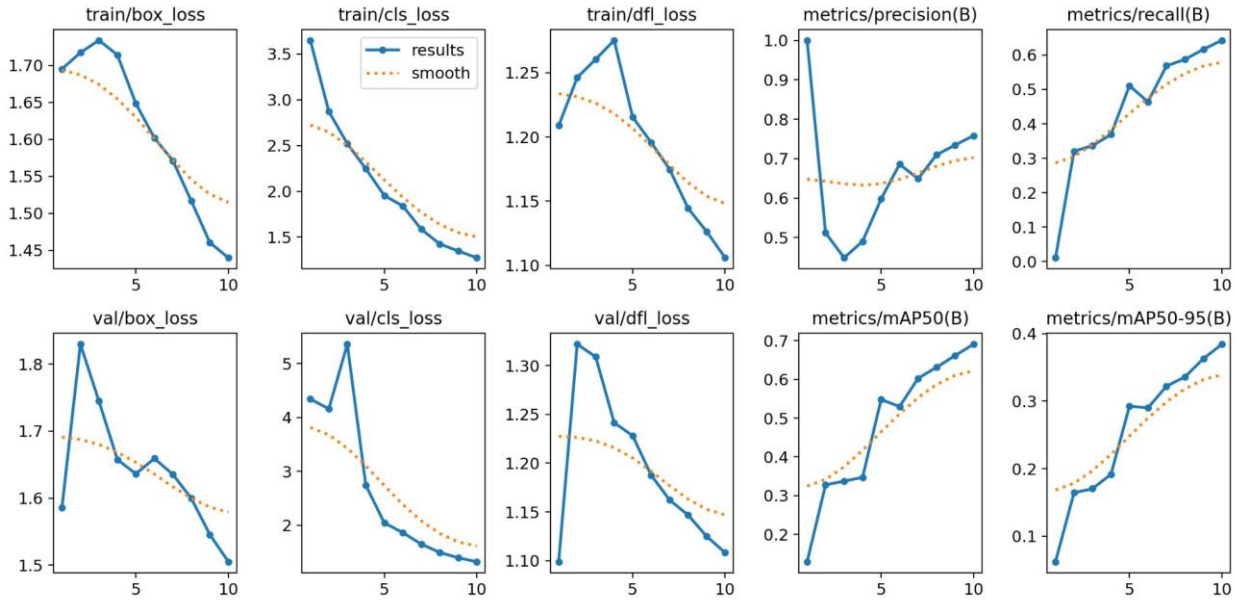

Рисунок А.11 – графіки якості моделі

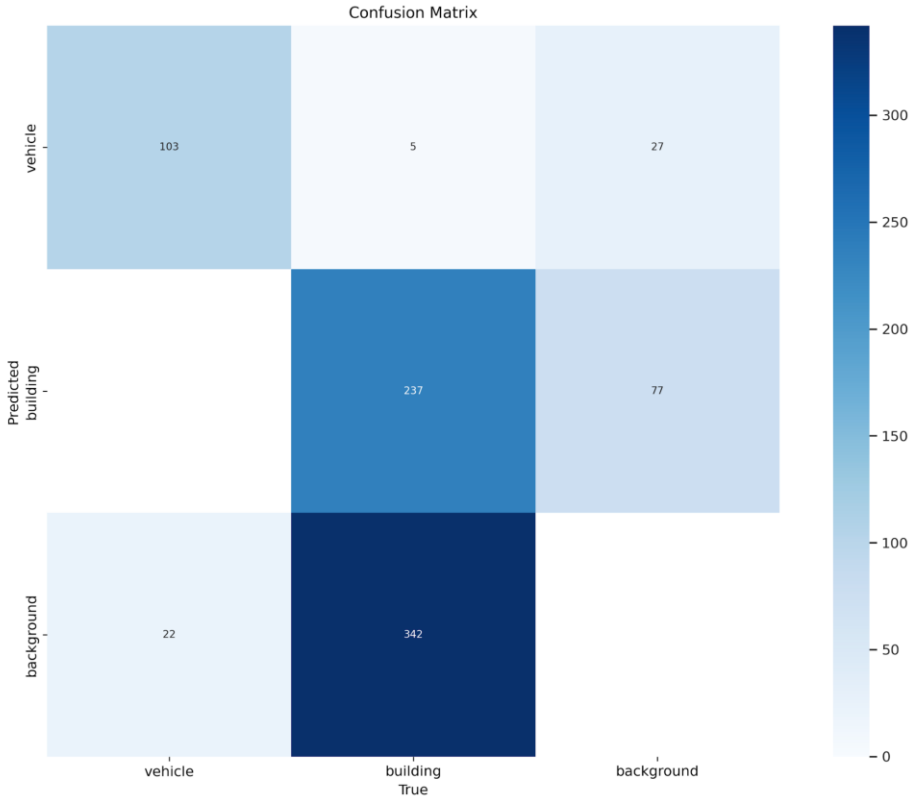

Рисунок А.12 – матриця конфузії параметру recall

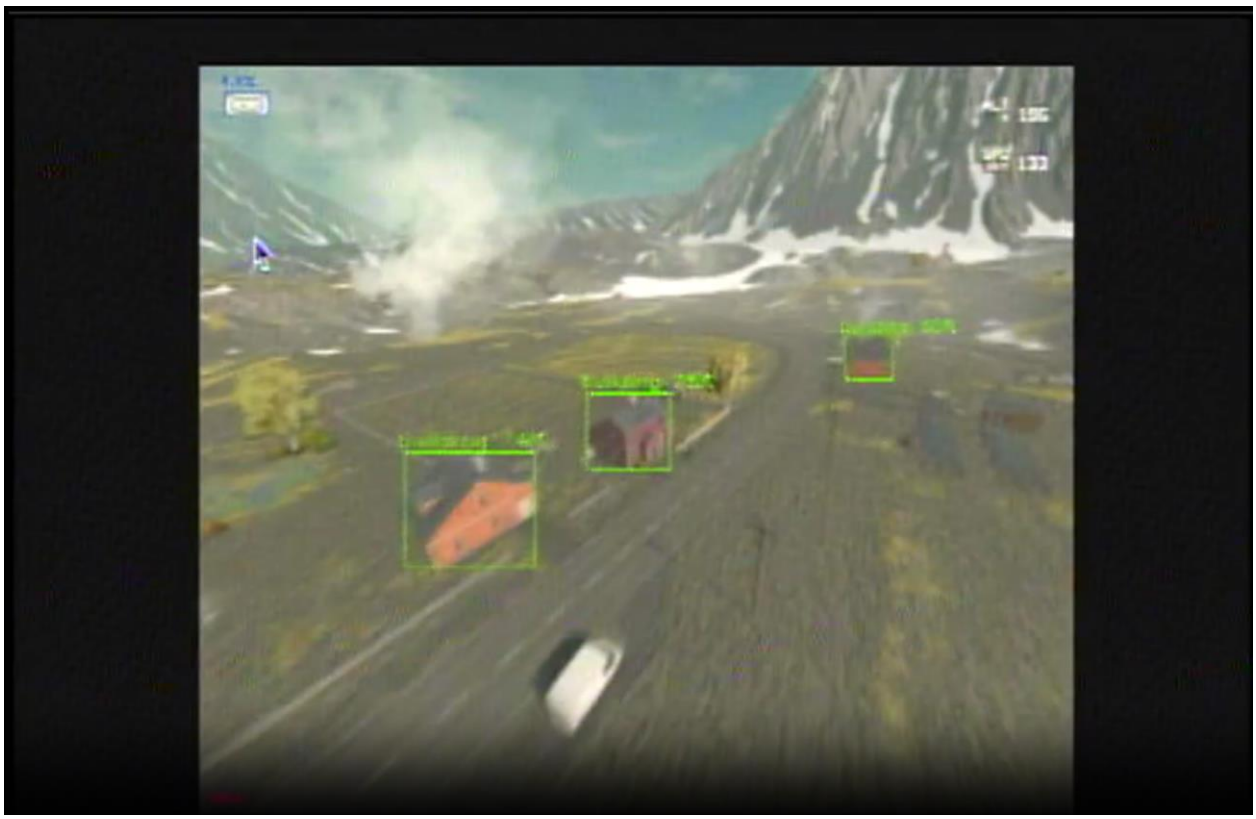

Рисунок А.13 – демонстрація роботи системи у симульованому середовищі

№	Позначення	Найменування	Дод. відомості
		Текстові документи	
1	ГЮИК. 402131.08 ПЗ	Пояснювальна записка	96 с., ф. А4
		Графічні документи	
		Демонстраційний матеріал	7 с.

Змін.	Арк.	№ документа	Підп.	Дата	ГЮИК. 402131.08 ВД				
Розроб.		Давидов Н.Д.			Розроблення системи автоматизації для управління роботизованими механізмами з використанням штучного інтелекту. Відомість кваліфікаційної роботи	Літ.	Аркуш		Аркушів
Перевір.		Чала О.О.							1
Н. контр.		Стародубцев М.Г.				ХНУРЕ Кафедра КІТАР			
Затвер.		Невлюдов І.Ш							