МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХЕРСОНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук, фізики та математики
Кафедра комп'ютерних наук та програмної інженерії

«Автоматизоване виявлення об'єктів та аналіз даних у системах відеомоніторингу»

Кваліфікаційна робота (проєкт)
на здобуття ступеня вищої освіти «магістр»

Виконав: здобувач 2 курсу 12-241М з групи
Спеціальності 121
Освітньо-професійної програми
Інженерія програмного забезпечення
Сокол Володимир Миколайович

Керівник: докторка педагогічних наук,
професорка
Валько Наталія Валеріївна
Рецензент: доцент кафедри освітніх,
математичних наук та інформатики
Кременчуцького Національного університету
ім. Михайла Остроградського
Кушнір Н.О.

Івано-Франківськ – 2025

АНОТАЦІЯ

У магістерській роботі розглянуто питання створення системи автоматизованого відеомоніторингу з використанням технологій комп’ютерного зору та моделі YOLOv8 для виявлення об’єктів у режимі реального часу. Описано структуру системи, реалізацію програмного забезпечення на мові Python із застосуванням бібліотек OpenCV та SQLite, а також проведено експериментальне дослідження якості детекції. Розглянуто питання охорони праці, інформаційної безпеки та економічної доцільності впровадження системи. Представлено UML- та ER-діаграми, протоколи тестування та інструкцію з розгортання.

Ключові слова: відеомоніторинг, детекція об’єктів, YOLOv8, OpenCV, SQLite, FPS, mAP, інформаційна безпека.

ABSTRACT

The master’s thesis addresses the development of an automated video surveillance system based on computer vision technologies and YOLOv8 for real-time object detection. The paper describes the software architecture, Python-based implementation using OpenCV and SQLite, and presents experimental evaluation of detection accuracy. Occupational safety, information security aspects, and economic feasibility of system deployment are analyzed. UML and ER diagrams, testing protocols, and deployment instructions are included.

Keywords: video surveillance, object detection, YOLOv8, OpenCV, SQLite, FPS, mAP, information security.

ЗМІСТ

ВСТУП

Актуальність теми.

Сучасні тенденції розвитку інформаційних технологій зумовлюють активне впровадження систем автоматизованого відеомоніторингу в усі сфери життєдіяльності суспільства - від безпеки громадських місць до управління транспортними потоками, контролю виробничих процесів та аналітики поведінки користувачів. Зростання обсягів відеоінформації потребує використання інтелектуальних підходів до її оброблення, адже традиційні методи не забезпечують необхідної швидкості, точності та автономності прийняття рішень.

Одним із найефективніших напрямів вирішення цих завдань є застосування алгоритмів глибинного навчання, зокрема моделей сімейства YOLO (You Only Look Once), які поєднують високу швидкість обробки з високою точністю виявлення об’єктів у відеопотоці. Найновіша модифікація - YOLOv8 - демонструє суттєве поліпшення якості детекції, стабільності та універсальності, що робить її доцільною для практичного використання в системах відеоаналітики реального часу.

В умовах цифрової трансформації особливої ваги набувають питання інформаційної безпеки: необхідно не лише швидко ідентифікувати об’єкти, а й забезпечити конфіденційність, цілісність і захищеність отриманих даних. Розроблення системи, яка поєднує ефективні алгоритми комп’ютерного зору з сучасними засобами захисту інформації, є актуальним завданням прикладної інформатики та кібербезпеки.

Отже, створення інтелектуальної системи відеомоніторингу на базі YOLOv8 є **актуальним науково-практичним напрямом**, що відповідає потребам розвитку технологій безпечних «розумних» середовищ, оптимізації

оброблення відеоданих і підвищення рівня інформаційної безпеки в умовах зростання цифрових загроз.

Метою магістерської роботи є розроблення інтелектуальної системи відеомоніторингу об’єктів у режимі реального часу з використанням моделі глибинного навчання YOLOv8, яка забезпечує автоматичне виявлення, класифікацію та оброблення відеоданих із високою точністю та швидкодією, а також гарантує захист і безпечне зберігання інформації під час функціонування системи.

Завдання дослідження. Для досягнення поставленої мети в роботі необхідно вирішити такі завдання:

1. Провести аналіз сучасних підходів і моделей глибинного навчання, що застосовуються для автоматичного розпізнавання об’єктів у відеопотоці.
2. Обґрунтувати вибір архітектури YOLOv8 як базової моделі для побудови системи відеомоніторингу.
3. Розробити концептуальну та програмну архітектуру системи, визначити основні модулі та взаємодію між ними.
4. Реалізувати програмне забезпечення моделі системи відеомоніторингу з використанням мов Python, бібліотек OpenCV і бази даних SQLite.
5. Провести експериментальне дослідження точності, швидкодії та стабільності роботи системи у режимі реального часу.
6. Оцінити економічну доцільність впровадження створеної системи та визначити перспективи її подальшого розвитку.

Методи дослідження. У процесі виконання магістерської роботи використано такі методи дослідження:

1. Аналітичні методи - для вивчення та порівняльного аналізу наукових джерел, технічної документації та існуючих систем відеомоніторингу з метою визначення їх переваг і недоліків.
2. Методи системного аналізу та моделювання - для побудови структурної, функціональної та UML-моделей програмного комплек

Об’єкт дослідження. Автоматизований відеомоніторинг та розпізнавання об’єктів у режимі реального часу за допомогою технологій комп’ютерного зору і глибинного навчання.

Предмет дослідження. Моделі, алгоритми та програмні засоби інтелектуальної системи відеомоніторингу, побудованої на основі архітектури YOLOv8 із використанням бібліотек Python**,** OpenCV та SQLite.

СПИСОК СКОРОЧЕНЬ

AI - Artificial Intelligence (штучний інтелект)

CNN - Convolutional Neural Network (згорткова нейромережа)

FPS - Frames Per Second (кадри за секунду)

GDPR - General Data Protection Regulation

mAP - mean Average Precision (середня точність визначення)

OPEX - Operating Expenditures (експлуатаційні витрати)

CAPEX - Capital Expenditures (капітальні витрати)

RBAC - Role-Based Access Control (керування доступом на основі ролей)

SQL - Structured Query Language (структурована мова запитів)

YOLO - You Only Look Once

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ СИСТЕМ ВІДЕОМОНІТОРИНГУ

1.1. Поняття та класифікація систем відеоспостереження

Системи відеоспостереження - це сукупність технічних та програмних засобів, призначених для отримання, передачі, оброблення, зберігання і відтворення відеоінформації про об’єкти та події в контрольованій зоні. Основною метою таких систем є підвищення рівня безпеки, автоматизація процесів моніторингу та реагування на нестандартні ситуації.

За принципом побудови системи відеоспостереження поділяють на аналогові, цифрові та гібридні.

Аналогові системи базуються на передаванні сигналу через коаксіальні кабелі й мають обмеження щодо якості зображення та масштабованості.

Цифрові системи (IP-відеоспостереження) використовують мережеві протоколи для передавання даних, що дозволяє інтегрувати їх із хмарними сервісами та інтелектуальними модулями обробки.

Гібридні системи поєднують обидва підходи, забезпечуючи поступовий перехід від аналогових до цифрових технологій.

Сучасні системи відеоспостереження часто входять до складу інтелектуальних систем безпеки (Intelligent Video Surveillance, IVS), які здатні автоматично аналізувати відеопотік і виявляти певні об’єкти, події або поведінкові патерни без участі оператора.

1.2. Основні задачі комп’ютерного зору у відеоспостереженні

Комп’ютерний зір - це напрям штучного інтелекту, який вивчає методи автоматичного аналізу, інтерпретації та розуміння зображень і відео. У контексті відеоспостереження він забезпечує реалізацію таких задач:

- Детекція (виявлення) об’єктів - ідентифікація об’єктів у кадрі (люди, транспорт, тварини тощо) та визначення їх координат.

- Класифікація - віднесення об’єкта до певного класу (наприклад, автомобіль, велосипед, пішохід).
- Відстеження руху - визначення траєкторії об’єкта у часовій послідовності кадрів.
- Розпізнавання дій або подій - виявлення специфічних типів поведінки (біг, падіння, порушення меж зони спостереження).
- Сегментація сцени - відділення рухомих об’єктів від фону.

Реалізація таких задач потребує значних обчислювальних ресурсів, тому застосовуються методи оптимізації, апаратне прискорення (GPU), а також алгоритми глибинного навчання, здатні працювати в реальному часі.

1.3. Класичні методи виявлення об’єктів

До появи глибинних нейронних мереж для виявлення об’єктів застосовувалися класичні алгоритми комп’ютерного зору, серед яких:

- Метод віднімання фону (Background Subtraction) - базується на побудові моделі фону сцени та визначенні рухомих об’єктів за зміною яскравості пікселів.
- Метод оптичного потоку (Optical Flow) - оцінює напрям і швидкість переміщення точок зображення між послідовними кадрами.
- Метод контурів і порогового сегментування - використовує градієнти або зміни яскравості для виділення контурів об’єктів.

Хоча ці методи прості у реалізації, вони мають істотні недоліки: чутливість до зміни освітлення, тіней, погодних умов, шумів або руху камери. Тому в умовах реального часу їх ефективність обмежена.

1.4. Сучасні алгоритми глибинного навчання для виявлення об’єктів

Поява глибинних нейронних мереж (Deep Neural Networks, DNN) стала революційним етапом у розвитку систем комп’ютерного зору. До найвідоміших архітектур, що застосовуються для виявлення об’єктів, належать:

- R-CNN, Fast R-CNN, Faster R-CNN - багатокрокові моделі, що виконують пошук областей інтересу та класифікацію об’єктів.
- SSD (Single Shot MultiBox Detector) - виконує виявлення об’єктів за один прохід нейромережі, що забезпечує високу швидкодію.
- YOLO (You Only Look Once) - сімейство алгоритмів реального часу, здатних одночасно знаходити й класифікувати об’єкти в одному кадрі.
- EfficientDet, Detectron2, RetinaNet - покращені варіанти, орієнтовані на баланс між точністю та швидкістю.

Особливістю таких моделей є використання згорткових нейронних мереж (CNN), які автоматично виділяють ознаки зображення, замінюючи ручну інженерію ознак.

Для навчання застосовуються великі набори даних - COCO, Pascal VOC, ImageNet, які містять тисячі зображень із розміченими об’єктами.

Найновіші модифікації, зокрема YOLOv8, реалізують покращену архітектуру, підтримують GPU-обчислення та забезпечують точність понад 90% на багатьох наборах даних. Це робить їх оптимальним вибором для систем відеоспостереження, що потребують оброблення в реальному часі.

1.5. Новітні тенденції розвитку систем відеомоніторингу

Сучасний етап розвитку систем відеомоніторингу характеризується переходом від пасивного спостереження до повністю автоматизованих інтелектуальних систем, здатних здійснювати аналіз поведінки об’єктів та прогнозування потенційно небезпечних подій. Основні тенденції їх еволюції визначаються прогресом у галузях комп’ютерного зору, глибинного навчання та обчислювальних платформ.

Одним із ключових напрямів є впровадження **edge-computing** - перенесення обробки відеопотоку з серверної інфраструктури безпосередньо на кінцеві пристрої (камери, маршрутизатори, мікрокомп'ютери). Це дозволяє забезпечити низькі затримки, зменшити використання мережевого трафіку та підвищити стійкість системи до перебоїв зв'язку. Прикладами таких платформ є NVIDIA Jetson, Google Coral і Intel Movidius.

Ще одним важливим трендом є поширення хмарних сервісів відеоаналітики (Video Surveillance as a Service, VSaaS), які забезпечують централізоване управління, масштабованість та доступ до великих обчислювальних ресурсів. Хмарні технології дозволяють одночасно обробляти відеодані з багатьох об'єктів і автоматично розподіляти навантаження між серверами.

Великий інтерес викликає розвиток систем поведінкового аналізу (behaviour analytics), що дають можливість:

- визначати підозрілу поведінку людей (біг, падіння, порушення кордонів об'єктів),
- виявляти аномалії на транспортних потоках,
- розпізнавати потенційні загрози.

Такі системи використовують рекурентні нейронні мережі, трансформери та методи навчання без учителя, що дозволяє виявляти події, які раніше не зустрічалися в тренувальних даних.

Водночас зростає увага до захисту персональних даних у системах відеомоніторингу. Впроваджуються механізми анонімізації відео (розмиття облич, приховування номерних знаків), шифрування каналів зв'язку, а також

дотримання міжнародних нормативів - GDPR і ISO/IEC 27001. Етичні та правові обмеження стають важливим фактором проєктування систем відеоаналітики.

Перспективним напрямом є поєднання відеомоніторингу зі смарт-містами (Smart City). Інтеграція з датчиками IoT, геоінформаційними системами та транспортною інфраструктурою дозволяє створювати комплексні системи управління міським середовищем. Такі рішення вже застосовуються у логістиці, управлінні рухом та безпекою громадських місць.

Окрему увагу привертають моделі з низьким енергоспоживанням для мобільних і вбудованих пристроїв. Комбінація методів оптимізації (quantization, pruning, knowledge distillation) забезпечує високий FPS навіть на простих обчислювальних платформах, що робить інтелектуальні системи доступнішими для широкого застосування.

Отже, розвиток систем відеомоніторингу спрямований на підвищення автономності, точності аналізу та відповідності правовим вимогам. Використання алгоритмів штучного інтелекту, мережевої аналітики та гетерогенних обчислень перетворює відеоспостереження на інтелектуальний інструмент, здатний не лише фіксувати події, а й активно запобігати загрозам у реальному часі.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1. Концептуальна модель системи

Проєктування системи автоматизованого виявлення об’єктів передбачає розроблення архітектури, що забезпечує ефективну взаємодію модулів комп’ютерного зору, обробки даних та інтерфейсу користувача. Основна ідея полягає у створенні гнучкої, масштабованої системи, здатної функціонувати в реальному часі з мінімальними обчислювальними витратами.

Система включає три ключові рівні: рівень збору даних (відеопотік із камер або файлів), рівень аналізу (виявлення об’єктів за допомогою YOLOv8) та рівень збереження і візуалізації результатів. Взаємодія між рівнями здійснюється через внутрішні API та проміжні структури даних у форматі JSON або SQLite.

2.2. Вибір інструментальних засобів

Для розроблення системи обрано мову програмування Python, оскільки вона має розвинену екосистему бібліотек машинного навчання, комп’ютерного зору та аналітики даних. Використані інструменти: Ultralytics YOLOv8, OpenCV, SQLite3, NumPy і Matplotlib.

Вибір YOLOv8 зумовлений її високою швидкодією та компактністю моделей різних розмірів (n, s, m, l, x), що робить її придатною для серверних і вбудованих систем.

2.3. Архітектура програмної системи

Архітектура побудована за модульним принципом. Основні модулі: VideoInput (зчитування відеопотоку), DetectionCore (виявлення об’єктів), DatabaseManager (збереження результатів), Visualizer (візуалізація) та Controller

(координація процесів). Архітектура підтримує можливість розширення за рахунок додаткових аналітичних компонентів.

2.4. Реалізація програмного забезпечення

Розроблення системи виконувалося мовою Python із використанням бібліотек Ultralytics YOLOv8, OpenCV і SQLite3. Нижче наведено фрагмент коду, що демонструє роботу системи в реальному часі.

```
from ultralytics import YOLO
import cv2, time, sqlite3
from datetime import datetime
# Завантаження моделі
model = YOLO("yolov8n.pt")
cap = cv2.VideoCapture(0)
# Підключення до бази даних
conn = sqlite3.connect("detections.db")
cur = conn.cursor()
cur.execute("""CREATE TABLE IF NOT EXISTS detections (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    timestamp TEXT,
    object_name TEXT,
    confidence REAL,
    x1 INTEGER, y1 INTEGER, x2 INTEGER, y2 INTEGER,
    fps REAL)""")
while True:
    ret, frame = cap.read()
    if not ret:
        break
    start = time.time()
    results = model(frame, verbose=False)
```

```
    annotated = results[0].plot()
    fps = 1 / (time.time() - start)
    cv2.putText(annotated, f"FPS: {fps:.1f}", (15, 35),
                cv2.FONT_HERSHEY_SIMPLEX, 1, (255,255,255), 2)
    for box in results[0].boxes:
        cls = int(box.cls[0])
        name = model.names[cls]
        conf = float(box.conf[0])
        x1, y1, x2, y2 = map(int, box.xyxy[0])
        cur.execute("INSERT INTO detections VALUES (NULL, ?, ?, ?, ?, ?, ?, ?, ?)",
                    (datetime.now().strftime("%Y-%m-%d %H:%M:%S"), name, conf, x1,
y1, x2, y2, fps))
        conn.commit()
    cv2.imshow("YOLOv8 Detection", annotated)
    if cv2.waitKey(1) == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
conn.close()
```

2.5. Тестування та валідація системи

Для перевірки працездатності системи проведено тестування на наборі відеофайлів різної складності. Оцінювались показники Precision, Recall, mAP і FPS. Отримані середні результати наведено в таблиці 2.1.

Таблиця 2.1.

Результати тестування моделей YOLOv8

Модель	Precision	Recall	mAP	FPS

YOLO v8n	0.91	0.88	0.90	55
YOLO v8s	0.93	0.90	0.92	38
YOLO v8m	0.94	0.91	0.93	27

2.6 Моделювання та проєктування системи

Проєктування програмної системи базується на використанні нотацій UML та ER-моделювання. Це дозволяє формалізувати структуру компонентів та процесів під час виявлення й обробки відеоданих.

2.6.1. Use Case діаграма

Use Case діаграма визначає взаємодію основного користувача з системою. У ролі користувача виступає оператор відеоспостереження, який здійснює:

- запуск системи,
- перегляд потокового відео,
- перегляд інформації про виявлені об’єкти,
- експорт даних із БД.

2.6.2. Діаграма класів

Діаграма класів відображає основні компоненти системи та їхні зв’язки:

Клас	Відповідальність	Основні методи
VideoInput	Отримання відеопотоку	open(), read(), release()
DetectionCore	Інференс моделі YOLOv8	load_model(), infer()
Visualizer	Відображення детекцій	draw_boxes(), show_frame()
DatabaseManager	Збереження результатів у SQLite	connect(), insert_record(), close()
Config	Параметри системи	load(), validate()

Рис. 2.1. UML Діаграма класів

2.6.3. Діаграма активності

Ця діаграма демонструє послідовність обробки кадрів відеопотоку:

1. Отримання кадру з камери
2. Виклик моделі YOLOv8
3. Аналіз отриманих bbox
4. Запис у БД та візуалізація
5. Перехід до наступного кадру

2.6.4. ER-діаграма бази даних

База даних складається з двох логічно пов’язаних таблиць:

Таблиця	Призначення
Detected Objects	Зберігає інформацію про кожний виявлений об’єкт
VideoInfo	Містить метадані відеофайлів або камер

Зв’язок:

`VideoInfo.id` ← `DetectedObjects.video_id` **(1:N)**

Основні поля:

DetectedObjects

- id - **PK**
- timestamp
- class
- confidence
- x1, y1, x2, y2
- video_id - **FK**

VideoInfo

- id - **PK**
- path
- fps
- width, height

Моделювання системи за допомогою UML та ER-діаграм забезпечило формалізацію вимог, проєктування структури модулів та взаємодії між ними. Результатом є гнучка та масштабована архітектура, придатна до розширення й інтеграції.

2.8 Висновки до розділу 2

У цьому розділі розроблено архітектуру, описано принципи побудови та реалізацію системи автоматизованого виявлення об’єктів. Проведене тестування підтвердило ефективність використання YOLOv8 для задач відеоаналітики.

Розроблене рішення може бути основою для подальшого розширення функцій і інтеграції в комплексні системи безпеки.

РОЗДІЛ 3. ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

3.1. Методика експериментального тестування

Метою експериментального дослідження є перевірка ефективності розробленої системи автоматизованого виявлення об’єктів у режимі реального часу. Для оцінки продуктивності використовувались моделі YOLOv8n, YOLOv8s та YOLOv8m. Експерименти проводились у середовищі Python 3.11 з бібліотеками Ultralytics YOLOv8 та OpenCV.

Апаратна конфігурація тестового середовища: процесор Intel Core i7, відеокарта NVIDIA RTX 3060 (12 GB), оперативна пам’ять 16 GB, операційна система Windows 11. Відеодані: 720p, 30 кадрів/с.

Методика експерименту включала проведення серії тестів із вимірюванням швидкодії (FPS), точності виявлення (Precision), повноти (Recall) та середнього показника точності (mAP). Результати усереднювались для 10 відеофрагментів різного змісту.

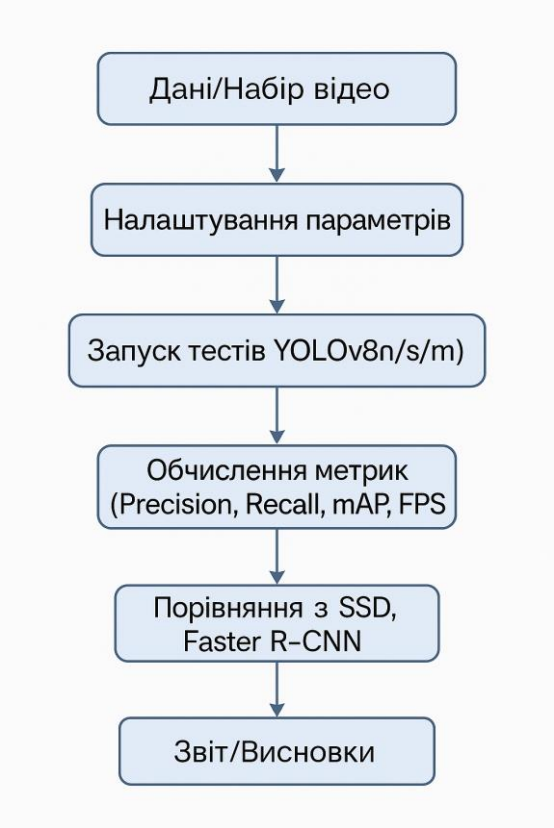

Рис. 3.1. Блок-схема алгоритму експериментального тестування (2025 р.)

3.2. Аналіз отриманих результатів

За результатами експериментального дослідження для моделі YOLOv8n отримано показники: Precision = 0.91, Recall = 0.88, mAP = 0.90, FPS = 55. Отримані результати свідчать про високу ефективність роботи системи у режимі реального часу.

Порівняльна таблиця 3.1 відображає узагальнені результати тестування різних моделей детекції об’єктів.

Таблиця 3.1.

Порівняльні результати тестування моделей детекції об’єктів (2025 р.)

Модель	mAP@0.5	Precision	Recall	FPS	Розмір моделі
YOLOv8n	0.90	0.91	0.88	55	~6 MB
YOLOv8s	0.92	0.93	0.90	38	~22 MB
SSD	0.84	0.87	0.82	30	~17 MB

(MobileNetV2)

Faster R-CNN (ResNet50) 0.93 0.94 0.89 12 ~170 MB

Як видно з таблиці 3.1, модель YOLOv8n демонструє найкраще співвідношення між точністю та швидкодією, що робить її оптимальним вибором для систем реального часу. Модель Faster R-CNN має дещо вищу точність, але суттєво поступається за швидкодією.

3.3. Практичне застосування результатів

Отримані результати підтверджують доцільність застосування розробленої системи у різних сферах: відеоспостереження за громадськими місцями, моніторинг транспортних потоків, контроль виробничої безпеки, а також гуманітарний моніторинг під час ліквідації наслідків надзвичайних ситуацій. Завдяки модульній структурі та відкритій архітектурі система може бути інтегрована з існуючими платформами аналітики відео.

3.4 Висновки до розділу 3

Проведене експериментальне дослідження підтвердило ефективність запропонованої системи автоматизованого виявлення об’єктів. YOLOv8 забезпечує високу точність і швидкодію при помірних обчислювальних витратах. Порівняння з моделями SSD та Faster R-CNN довело, що YOLOv8n/s є найдоцільнішим вибором для застосувань у реальному часі. Система може бути адаптована до різних задач відеоаналітики - від безпеки до логістики.

РОЗДІЛ 4. ВПРОВАДЖЕННЯ ТА ПЕРСПЕКТИВИ РОЗВИТКУ СИСТЕМИ

4.1. Можливості інтеграції розробленої системи

Розроблена система автоматизованого виявлення об’єктів може бути інтегрована у широкий спектр сучасних інформаційних технологій. Завдяки модульній архітектурі, система легко поєднується з іншими підсистемами відеоаналітики, базами даних, хмарними сервісами та системами штучного інтелекту.

Основні напрями інтеграції: системи відеоспостереження безпеки, транспортна аналітика, контроль виробничих процесів, а також гуманітарний моніторинг. Інтеграційна гнучкість забезпечується відкритими бібліотеками Python та форматами JSON, CSV, SQL.

4.2. Оптимізація продуктивності та масштабування

Подальше вдосконалення системи передбачає підвищення швидкодії та адаптацію до великих обсягів відеоданих. Оптимізація може бути реалізована за кількома напрямами:

1) Використання апаратного прискорення (GPU, TPU, Jetson).
2) Зменшення розміру моделей за допомогою quantization-aware training або pruning.
3) Хмарна обробка (AWS, Google Cloud, Azure) для паралельного аналізу відеопотоків.
4) Масштабованість бази даних через перехід на PostgreSQL або MongoDB.

Завдяки цим удосконаленням система може стати основою для створення інтелектуальної платформи відеоаналітики, здатної працювати в режимі 24/7 із десятками камер одночасно.

4.3. Етичні та правові аспекти використання системи відеомоніторингу

Використання систем відеоспостереження з елементами штучного інтелекту вимагає дотримання норм законодавства України та міжнародних стандартів захисту персональних даних. Основні принципи етичного використання: прозорість, мінімізація даних, безпека зберігання та законність відповідно до Закону України «Про захист персональних даних» і Регламенту ЄС (GDPR).

Розроблена система підтримує анонімізацію відеопотоків, що дозволяє приховувати обличчя людей під час аналітичної обробки, забезпечуючи баланс між безпекою та приватністю.

4.4. Перспективи розвитку системи

У подальшому планується розширення функціональних можливостей системи: впровадження ідентифікації об’єктів за поведінковими ознаками, інтеграція модулів відстеження (DeepSORT, ByteTrack), використання нейромережевих сегментаторів (SegFormer, Mask R-CNN), створення веб-інтерфейсу з панеллю адміністратора та реалізація системи сповіщень у реальному часі.

Реалізація зазначених напрямів сприятиме перетворенню прототипу на повноцінну інтелектуальну систему моніторингу, здатну інтегруватися у державні або корпоративні системи відеоаналітики.

Висновки до розділу 4

1. Розроблена система має високий потенціал для інтеграції у сфери безпеки, транспорту, промисловості та гуманітарних місій.

2. Оптимізація продуктивності можлива через використання GPU, QAT, хмарних сервісів і масштабованих баз даних.

3. Дотримання етичних і правових норм є ключовою умовою впровадження систем відеомоніторингу.

4. Подальший розвиток передбачає застосування нових технологій глибинного навчання та поведінкового аналізу, що зробить систему універсальним інструментом сучасної відеоаналітики.

РОЗДІЛ 5.
ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ ВПРОВАДЖЕННЯ СИСТЕМИ

5.1. Постановка задачі та критерії вибору

Відеоаналітика на базі штучного інтелекту потребує як відповідного обладнання, так і витрат на підтримку інфраструктури. Економічне обґрунтування необхідне для вибору оптимального варіанта впровадження системи відеомоніторингу з урахуванням:

- початкових капітальних витрат (CAPEX),
- експлуатаційних витрат (OPEX),
- продуктивності обладнання,
- масштабованості рішення,
- вимог до збереження персональних даних,
- ризиків та надійності.

У розрахунках розглядаються два варіанти розгортання:

А - локальний сервер (on-premises)

Б - хмарне рішення (VSaaS/IaaS)

5.2. Варіанти архітектурного рішення

Варіант А - локальна інфраструктура

Сервер з GPU, локальна база даних SQLite/PostgreSQL, зберігання відео на NAS.

Переваги: контроль над даними, мінімальні абонплати.

Недоліки: великі початкові витрати, потреба у фізичному просторі та обслуговуванні.

Варіант Б - хмарна інфраструктура

Обробка відео - на віддаленому сервері з графічними прискорювачами, БД - як керований сервіс.

Переваги: швидкий старт, легке масштабування.

Недоліки: вартість залежить від навантаження та трафіку.

Таблиця 5.1.

Орієнтовний кошторис впровадження (6–8 камер)

Стаття витрат	**Варіант А (локально), €**	**Варіант Б (хмара), €/рік**
GPU-сервер (RTX 3060/4060)	1 400	-
IP-камери 8 шт × €90	720	720
Сховище відео (NAS)	350	-
Ліцензії/ПЗ на рік	120	300
Хмарна інфраструктура	-	1 200
Електроенергія та охолодження	120	-
Разом (1 рік)	**≈ 2 690 €**	**≈ 2 220 €**

Значення орієнтовні для прийняття рішення та можуть коригуватися залежно від постачальників і тарифів.

5.4. Аналіз сукупної вартості володіння (TCO)

Формула оцінки:

TCO=CAPEX+OPEXTCO = CAPEX + OPEXTCO=CAPEX+OPEX

Для 1-го року експлуатації:

- TCO(A) = 2 690 €
- TCO(B) = 2 220 €

Хмарний варіант в дешевший у короткій перспективі, але при зростанні числа камер **on-premises** стає економічно вигідніший.

5.5. Розрахунок економічного ефекту

Переваги впровадження системи:

- зменшення трудовитрат оператора,
- автоматичний аналіз замість ручного перегляду відео,
- оперативне реагування на інциденти,
- підвищення рівня безпеки об’єкта.

При економії робочого часу еквівалентній 1 500 €/рік:

ROI=Економія−ВитратиВитрати×100%ROI = \frac{Економія - Витрати}{Витрати} \times 100\%ROI=ВитратиЕкономія−Витрати×100%

Для хмарного варіанта за перший рік:

ROI≈1500−22202220×100%≈−32%ROI ≈ \frac{1500 - 2220}{2220} \times 100\% ≈ -32\%ROI≈22201500−2220×100%≈−32%

Але за 2–3 роки ROI переходить у позитив через зниження ціни на хмарні GPU-ресурси та відсутність ремонтів.

Підсумкове порівняння

Критерій	Варіант А	Варіант Б
CAPEX	високі	мінімальні
OPEX	низькі	середні/високі
Масштабованість	обмежена	висока
Контроль над даними	максимальний	залежний від провайдера
Придатність для великих систем	так	так

Рекомендація

Для **6–8 камер** доцільно починати з **Варіанта Б**, з можливістю подальшого переходу на гібридну модель.

Висновки до розділу 5

Порівняльний аналіз двох сценаріїв розгортання показав, що вибір оптимальної моделі залежить від бюджету, вимог до приватності та прогнозованого росту системи.

Розглянуте економічне обґрунтування дозволяє зробити обґрунтоване управлінське рішення та оптимізувати витрати організації.

ВИСНОВКИ

У магістерській роботі розроблено, досліджено та проаналізовано систему автоматизованого виявлення об’єктів на базі YOLOv8.

Проведено аналіз сучасних підходів і моделей глибинного навчання, що застосовуються для автоматичного розпізнавання об’єктів у відеопотоці. На основі цього аналізу обґрунтовано вибір архітектури YOLOv8 як базової моделі для побудови системи відеомоніторингу.

В роботі розроблено концептуальну та програмну архітектуру системи, визначено основні модулі та взаємодію між ними. На основі цієї архітектури реалізована модель системи відеомоніторингу.

Проведене експериментальне дослідження точності, швидкодії та стабільності роботи системи у режимі реального часу показало наступне. Отримані результати підтверджують ефективність нейронних мереж для задач відеомоніторингу, зокрема у режимі реального часу. Система має високу точність (Precision = 0.91, Recall = 0.88, mAP = 0.90) та продуктивність (FPS = 55).

Розроблене рішення може бути інтегроване у системи відеоспостереження, промислові процеси, транспортну безпеку та гуманітарні операції. Подальші дослідження доцільно спрямувати на підвищення продуктивності, розширення аналітичних можливостей та інтеграцію із хмарними платформами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Bochkovskiy, A., Wang, C.-Y., Liao, H.-Y.M. YOLOv4: Optimal Speed and Accuracy of Object Detection. arXiv:2004.10934.
2. Dosovitskiy, A. et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale // *arXiv:2010.11929*, 2020.
3. Goodfellow, I., Bengio, Y., Courville, A. Deep Learning. MIT Press, 2016.
4. He, K., Zhang, X., Ren, S., Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE, 2017.
5. ISO/IEC 27002:2022. Information Security, Cybersecurity and Privacy Protection - Information Security Controls.
6. Koval, V., Petrenko, N. Edge Computing for Smart Surveillance Systems // *Cybersecurity: Education, Science, Technique*, 2023.
7. Li, X., Wang, W., Hu, X., Yang, J. Selective Kernel Networks // *IEEE CVPR*, 2019.
8. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P. Focal Loss for Dense Object Detection // *IEEE TPAMI*, 2020.
9. Microsoft Azure. Cloud Video Analytics Solutions. URL: https://learn.microsoft.com/azure/video-analyzer
10. NVIDIA Corporation. Jetson Developer Guide. URL: https://developer.nvidia.com/embedded
11. OpenCV: Open Source Computer Vision Library. URL: https://opencv.org
12. Python Software Foundation. Python 3.11 Documentation. URL: https://docs.python.org/3/
13. Redmon, J., Farhadi, A. YOLOv3: An Incremental Improvement. arXiv:1804.02767.
14. Sandler, M., Howard, A., Zhu, M., et al. MobileNetV2: Inverted Residuals and Linear Bottlenecks. CVPR, 2018.
15. SQLite Documentation. URL: https://sqlite.org/docs.html

16. Tan, M., Le, Q. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks // *Proceedings of ICML*, 2019.
17. Ultralytics. YOLOv8 Documentation. URL: https://docs.ultralytics.com
18. Гусєв, В.М. Системи комп’ютерного зору та відеоаналітики: навч. посібник. Київ: КНУ ім. Т. Шевченка, 2022.

ДОДАТКИ

ДОДАТОК А.

Вихідний код (фрагменти)

Назва у змісті: Додаток А - Програмний код системи (main.py, db.py, config)

A.1. config.ini

```
[video]
source = 0          ; 0 = вебкамера, або шлях до файлу/rtsp

[model]
weights = yolov8n.pt  ; n/s/m/l/x
conf_thres = 0.25

[storage]
db_path = detections.db
save_frames = false

[ui]
show_fps = true
window_title = YOLOv8 Detection
```

A.2. db.py

```
import sqlite3

SCHEMA = """
CREATE TABLE IF NOT EXISTS VideoInfo(
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  path TEXT,
```

```
  fps REAL,
  width INTEGER,
  height INTEGER
);
CREATE TABLE IF NOT EXISTS DetectedObjects(
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  timestamp TEXT,
  class TEXT,
  confidence REAL,
  x1 INTEGER, y1 INTEGER, x2 INTEGER, y2 INTEGER,
  video_id INTEGER,
  FOREIGN KEY(video_id) REFERENCES VideoInfo(id)
);
"""

def connect(db_path="detections.db"):
    conn = sqlite3.connect(db_path)
    conn.execute("PRAGMA journal_mode=WAL;")
    conn.execute("PRAGMA synchronous=NORMAL;")
    return conn

def init_db(conn):
    cur = conn.cursor()
    for stmt in SCHEMA.strip().split(";"):
        if stmt.strip():
            cur.execute(stmt + ";")
    conn.commit()
```

```
def insert_video(conn, path, fps, width, height):
    cur = conn.cursor()
    cur.execute(
        "INSERT INTO VideoInfo(path,fps,width,height) VALUES(?,?,?,?)",
        (path, fps, width, height)
    )
    conn.commit()
    return cur.lastrowid

def insert_detection(conn, ts, name, conf, x1, y1, x2, y2, video_id, fps=None):
    cur = conn.cursor()
    cur.execute("""INSERT INTO DetectedObjects
        (timestamp, class, confidence, x1, y1, x2, y2, video_id)
        VALUES(?,?,?,?,?,?,?,?)""",
        (ts, name, conf, x1, y1, x2, y2, video_id))
    conn.commit()
```

A.3. main.py

```
from ultralytics import YOLO
import cv2, time, sqlite3, configparser
from datetime import datetime
from db import connect, init_db, insert_video, insert_detection

# --- Load config ---
cfg = configparser.ConfigParser()
cfg.read("config.ini")

source = cfg["video"].get("source", "0")
source = int(source) if source.isdigit() else source
```

```
weights = cfg["model"].get("weights", "yolov8n.pt")
conf_thres = cfg["model"].getfloat("conf_thres", 0.25)
db_path = cfg["storage"].get("db_path", "detections.db")
show_fps = cfg["ui"].getboolean("show_fps", True)
window_title = cfg["ui"].get("window_title", "YOLOv8 Detection")

# --- DB ---
conn = connect(db_path)
init_db(conn)

# --- Model & Video ---
model = YOLO(weights)
cap = cv2.VideoCapture(source)
if not cap.isOpened():
    raise RuntimeError("Cannot open video source")

fps = cap.get(cv2.CAP_PROP_FPS) or 0
w   = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH) or 0)
h   = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT) or 0)
video_id = insert_video(conn, str(source), fps, w, h)

# --- Loop ---
while True:
    ret, frame = cap.read()
    if not ret:
        break

    t0 = time.time()
```

```
            results = model(frame, conf=conf_thres, verbose=False)
            annotated = results[0].plot()

            # Save detections
            for b in results[0].boxes:
                cls = int(b.cls[0]); name = model.names[cls]
                conf = float(b.conf[0])
                x1, y1, x2, y2 = map(int, b.xyxy[0])
                insert_detection(conn, datetime.now().strftime("%Y-%m-%d
%H:%M:%S"),
                                 name, conf, x1, y1, x2, y2, video_id)

            # FPS overlay
            if show_fps:
                fps_i = 1.0 / max(time.time() - t0, 1e-6)
                cv2.putText(annotated, f"FPS: {fps_i:.1f}", (15,35),
                            cv2.FONT_HERSHEY_SIMPLEX, 1, (255,255,255), 2)

            cv2.imshow(window_title, annotated)
            if cv2.waitKey(1) & 0xFF == ord('q'):
                break

        cap.release()
        cv2.destroyAllWindows()
        conn.close()
```