МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління

ОМЕЛЬЧУК Олександр Андрійович

Програмна система інтелектуального відеоспостереження в режимі реального часу на основі згорткових нейронних мереж та туманних обчислень / Software System for Real-Time Intelligent Video Surveillance Based on Convolutional Neural Networks and Fog Computing

Спеціальність 122 – Комп'ютерні науки
Освітньо-професійна програма – Штучний інтелект

Кваліфікаційна робота

Виконав студент групи КНШІ-41
О. А. Омельчук

Науковий керівник:
д.т.н., професор, М. П. Комар

Кваліфікаційну роботу допущено
до захисту
«___»_____ 2026 р.

Завідувач кафедри
_____ Н.В. Дзюбановська

Тернопіль – 2026

Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «бакалавр»
спеціальність: 122 – Комп'ютерні науки
освітньо-професійна програма – Штучний інтелект

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Н.В. Дзюбановська
«____» _____20__ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
ОМЕЛЬЧУКУ Олександру Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи
Програмна система інтелектуального відеоспостереження в режимі реального часу на основі згорткових нейронних мереж та туманних обчислень / Software System for Real-Time Intelligent Video Surveillance Based on Convolutional Neural Networks and Fog Computing
керівник роботи д.т.н., професор М. П. Комар
затверджені наказом по університету від 01 грудня 2025 року № 894.
2. Строк подання студентом закінченої кваліфікаційної роботи 25 травня 2026 р.
3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.
4. Основні питання, які потрібно розробити
– проаналізувати особливості систем відеоспостереження реального часу та сучасні підходи до їх інтелектуалізації;
– дослідити можливості застосування згорткових нейронних мереж у задачах аналізу відеопотоку;
– розглянути принципи використання туманних обчислень у системах реального часу;
– виконати аналіз існуючих рішень інтелектуального відеоспостереження;
– сформувати вимоги до програмної системи та розробити її загальну архітектуру;
– спроєктувати функціональні модулі системи та алгоритми її роботи;
– реалізувати основні програмні компоненти системи;
– провести експериментальні дослідження роботи розробленої системи та оцінити отримані результати.
5. Перелік графічного матеріалу в роботі
– загальна архітектура програмної системи інтелектуального відеоспостереження;
– структура моделі оброблення відеопотоку на основі згорткової нейронної мережі encoder-decoder;
– схема алгоритму функціонування програмної системи інтелектуального відеоспостереження;

– схема алгоритму взаємодії між edge-, fog- і cloud-рівнями;
– схема організації структури проєкту програмної системи.

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 01 грудня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Затвердження теми кваліфікаційної роботи, ознайомлення з літературними джерелами та складання плану роботи	до 01.01. 2026 р.	
2	Написання 1 розділу кваліфікаційної роботи	до 01.02. 2026 р.	
3	Написання 2 розділу кваліфікаційної роботи	до 15.03.2026 р.	
4	Написання 3 розділу кваліфікаційної роботи	до 01.05. 2026 р.	
5	Представлення попереднього варіанту кваліфікаційної роботи, перевірка та внесення змін керівником	до 15.05.2026 р.	
6	Опрацювання зауважень та представлення завершеного варіанту кваліфікаційної роботи. Підготовка супроводжуючих документів	до 25.05.2026 р.	
7	Перевірка кваліфікаційної роботи на оригінальність тексту	до 30.05.2026 р.	
8	Оформлення кваліфікаційної роботи та отримання допуску до захисту	до 10.06.2026 р.	
9	Подання кваліфікаційної роботи до захисту на засіданні атестаційної комісії	до 10.06. 2026 р.	

Студент_____ О. А. Омельчук
підпис

Керівник роботи_____ д.т.н., професор М. П. Комар
підпис

АНОТАЦІЯ

Пояснювальна записка до кваліфікаційної роботи: 78 с., 16 рис., 15 табл., 2 додатки, 46 джерел.

Об’єктом дослідження є процес інтелектуального оброблення відеопотоку в системах відеоспостереження реального часу.

Метою кваліфікаційної роботи є розроблення програмної системи інтелектуального відеоспостереження, що забезпечує аналіз відеопотоку в режимі реального часу на основі згорткових нейронних мереж і туманних обчислень.

Методи дослідження включають аналіз наукових джерел у сфері інтелектуального відеоспостереження, комп’ютерного зору, згорткових нейронних мереж і туманних обчислень, моделювання архітектури програмної системи, розроблення CNN-моделі типу encoder–decoder для сегментації області дії, а також експериментальне тестування якості сегментації та швидкодії.

Результати дослідження полягають у створенні програмної системи інтелектуального відеоспостереження, яка забезпечує отримання відеопотоку, попереднє оброблення кадрів, сегментацію області дії, формування маски об’єкта, виділення цільової області за допомогою Mask-Crop, формування події та збереження результатів.

Результати роботи можуть бути використані для розробки інтелектуальних систем відеоспостереження, автоматизованого аналізу відеопотоку, систем безпеки та програмних рішень на основі комп’ютерного зору й туманних обчислень.

Ключові слова: ІНТЕЛЕКТУАЛЬНЕ ВІДЕОСПОСТЕРЕЖЕННЯ, ВІДЕОПОТІК, КОМП’ЮТЕРНИЙ ЗІР, ЗГОРТКОВА НЕЙРОННА МЕРЕЖА, ENCODER–DECODER, СЕГМЕНТАЦІЯ, ТУМАННІ ОБЧИСЛЕННЯ.

ANNOTATION

The bachelor’s thesis report: 78 pages, 16 figures, 15 tables, 2 appendices, 46 sources.

The object of the research is the process of intelligent video stream processing in real-time video surveillance systems.

The purpose of the qualification work is to develop a software system for intelligent video surveillance that provides real-time video stream analysis based on convolutional neural networks and fog computing.

The research methods include the analysis of scientific sources in the fields of intelligent video surveillance, computer vision, convolutional neural networks, and fog computing; modeling the architecture of the software system; developing an encoder–decoder CNN model for action area segmentation; and experimentally testing segmentation quality and processing speed.

The research results consist in the development of an intelligent video surveillance software system that provides video stream acquisition, frame preprocessing, action area segmentation, object mask generation, target area extraction using Mask-Crop, event formation, and result storage.

The results of the work can be used for the development of intelligent video surveillance systems, automated video stream analysis, security systems, and software solutions based on computer vision and fog computing.

Keywords: INTELLIGENT VIDEO SURVEILLANCE, VIDEO STREAM, COMPUTER VISION, CONVOLUTIONAL NEURAL NETWORK, ENCODER–DECODER, SEGMENTATION, FOG COMPUTING.

ЗМІСТ

ВСТУП

У сучасних умовах стрімкого розвитку цифрових технологій системи відеоспостереження стали важливим складником інформаційної інфраструктури безпеки. Такі системи застосовуються на об’єктах критичної інфраструктури, у транспортних вузлах, громадських просторах, освітніх закладах, промислових підприємствах і приватному секторі. Зростання кількості камер, безперервне надходження великих обсягів відеоданих і потреба в оперативному реагуванні на потенційно небезпечні події зумовили перехід від звичайного пасивного запису відео до інтелектуального автоматизованого аналізу відеопотоку [4, 19].

Традиційні системи відеоспостереження, у яких основний акцент зроблено на передавання та централізоване збереження відео, мають низку суттєвих обмежень. До них належать значне навантаження на мережеву інфраструктуру, висока затримка оброблення, обмежені можливості масштабування, а також залежність від центрального обчислювального вузла. За таких умов знижується ефективність системи в задачах реального часу, де критичне значення мають швидке виявлення об’єктів, подій або аномальних ситуацій та негайне формування повідомлення про них [1, 6, 8].

Одним із найбільш перспективних напрямів розв’язання цієї задачі стало поєднання методів глибокого навчання із розподіленими підходами до оброблення даних. Зокрема, згорткові нейронні мережі продемонстрували високу ефективність у задачах комп’ютерного зору: виявленні об’єктів, сегментації зображень, розпізнаванні осіб, аналізі сцен і подій у відеопотоці [10, 14, 23]. Водночас для забезпечення роботи в режимі реального часу важливим стало перенесення частини обчислень ближче до джерела даних, тобто до камер або локальних вузлів опрацювання.

Саме тому дедалі ширшого застосування набувають туманні обчислення, які забезпечують проміжний рівень між периферійними пристроями та хмарною інфраструктурою, зменшуючи затримку, обсяг даних, що передаються і навантаження на центральні сервери [6, 15, 31].

Актуальність теми полягає в необхідності розроблення програмних систем

інтелектуального відеоспостереження, здатних автоматично аналізувати відеопотік у режимі реального часу, виявляти об’єкти або події та ефективно розподіляти обчислювальні ресурси між edge-, fog- та cloud-рівнями.

Такий підхід дає змогу підвищити швидкодію системи, зменшити затримку під час прийняття рішень, підвищити надійність функціонування та адаптувати систему до практичного використання в умовах обмежених апаратних ресурсів [4, 19, 31].

Метою кваліфікаційної роботи є розроблення програмної системи інтелектуального відеоспостереження, що забезпечує аналіз відеопотоку в режимі реального часу на основі згорткових нейронних мереж і туманних обчислень.

Для досягнення поставленої мети сформовано такі завдання дослідження:

– проаналізувати особливості систем відеоспостереження реального часу та сучасні підходи до їх інтелектуалізації;

– дослідити можливості застосування згорткових нейронних мереж у задачах аналізу відеопотоку;

– розглянути принципи використання туманних обчислень у системах реального часу;

– виконати аналіз існуючих рішень інтелектуального відеоспостереження;

– сформувати вимоги до програмної системи та розробити її загальну архітектуру;

– спроєктувати функціональні модулі системи та алгоритми її роботи;

– реалізувати основні програмні компоненти системи;

– провести експериментальні дослідження роботи розробленої системи та оцінити отримані результати.

Об’єктом дослідження є процес інтелектуального оброблення відеопотоку в системах відеоспостереження реального часу.

Предметом дослідження є методи, моделі та програмні засоби аналізу відеоданих на основі згорткових нейронних мереж у середовищі туманних обчислень.

Результати кваліфікаційної роботи апробовані та опубліковані у матеріалах IV Всеукраїнської науково-практичної конференції молодих вчених і студентів

«Інтелектуальні комп’ютерні системи та мережі» (ІКСМ), м. Тернопіль, ЗУНУ, 20 травня 2026 р. (додаток Б).

Кваліфікаційна робота складається із вступу, трьох розділів, висновків, списку використаних джерел та додатків.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА СУЧАСНИХ ПІДХОДІВ ДО ІНТЕЛЕКТУАЛЬНОГО ВІДЕОСПОСТЕРЕЖЕННЯ

1.1 Особливості систем відеоспостереження реального часу та їх функціональні можливості

Системи відеоспостереження реального часу призначені для безперервного отримання, передавання, оброблення та збереження відеоданих із подальшим виявленням подій, що мають практичне значення для безпеки, контролю або моніторингу об’єктів. На відміну від традиційних систем, орієнтованих переважно на запис відеоархіву, сучасні рішення дедалі частіше виконують функції автоматизованого аналізу потоку кадрів, розпізнавання об’єктів, відстеження їх переміщення та формування повідомлень про підозрілі ситуації [4, 5].

Основною особливістю систем відеоспостереження реального часу є жорстке обмеження на час реакції. Оброблення даних має виконуватися достатньо швидко, щоб результат аналізу залишався актуальним у момент надходження відеопотоку. Це означає, що система повинна не лише приймати кадри з камери, а й виконувати попередню обробку, аналіз сцени, виявлення цільових об’єктів та видачу результату з мінімальною затримкою. За таких умов критичними стають продуктивність алгоритмів, обсяг передаваних даних, пропускна здатність мережі та ефективність використання обчислювальних ресурсів [1, 6].

Узагальнену структуру системи відеоспостереження реального часу подано на рисунку 1.1.

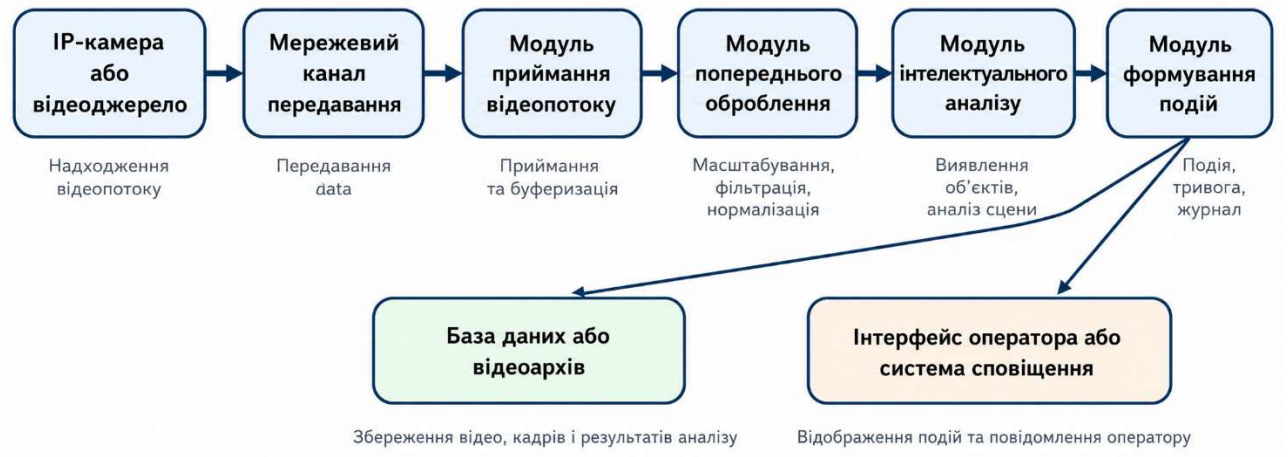

Рисунок 1.1 – Загальна структура системи відеоспостереження реального часу

У загальному вигляді система відеоспостереження реального часу містить кілька взаємопов’язаних складових. Першою складовою є джерело відеоданих, тобто цифрові камери або IP-камери, які формують потік зображень. Другою складовою є комунікаційне середовище, через яке відеодані передаються до вузлів оброблення. Третьою складовою є програмно-апаратний рівень аналізу, на якому виконуються задачі фільтрації кадрів, виявлення об’єктів, класифікації сцен чи подій. Четвертою складовою є модуль збереження результатів і формування повідомлень оператору або іншій підсистемі. У більш розвинених рішеннях до цієї структури також входять аналітична панель, журнал подій, модуль керування політиками спостереження та засоби інтеграції з іншими інформаційними системами.

Основні функції систем відеоспостереження реального часу узагальнено в таблиці 1.1.

Таблиця 1.1 – Основні функції систем відеоспостереження реального часу

Функція	Призначення	Результат виконання
Захоплення відеопотоку	Отримання кадрів з камери в реальному часі	Безперервний потік зображень
Передавання даних	Доставка відео до вузла оброблення	Потік кадрів у системі
Попереднє оброблення	Масштабування, нормалізація, фільтрація шуму	Підготовлені кадри
Виявлення об’єктів	Пошук цільових об’єктів у кадрі	Координати та класи об’єктів
Відстеження об’єктів	Аналіз переміщення в послідовності кадрів	Траєкторія руху
Аналіз подій	Визначення контрольованих ситуацій	Список подій
Формування сповіщень	Інформування оператора або підсистеми	Повідомлення або сигнал тривоги
Збереження результатів	Архівування відео та журналу подій	База відео і подій

Функціональні можливості сучасних систем відеоспостереження істотно ширші, ніж звичайний перегляд зображення з камери. До базових функцій належать захоплення відеопотоку, відображення зображення в реальному часі,

запис і збереження відеоархіву, пошук потрібних фрагментів та віддалений доступ до камер. Проте інтелектуальні системи додатково забезпечують автоматичне виявлення об’єктів у кадрі, розпізнавання людини або транспортного засобу, відстеження траєкторії руху, аналіз поведінки об’єктів, виявлення вторгнення в задану зону, підрахунок об’єктів, розпізнавання аномалій і генерацію сигналів тривоги [21, 24].

Окремою важливою властивістю таких систем є контекстна чутливість. Ефективність відеоспостереження залежить не лише від якості камери або точності моделі, а й від умов освітлення, складності фону, щільності об’єктів у кадрі, кута огляду, наявності шумів та динаміки сцени. У реальних умовах система має працювати в середовищі з частковими перекриттями об’єктів, зміною погоди, варіативною дистанцією до цілі та нестабільною якістю мережевого з’єднання. Саме тому вимоги до програмної системи відеоспостереження охоплюють не тільки точність розпізнавання, а й стійкість до зовнішніх впливів, масштабованість і адаптивність.

Для систем реального часу принципове значення має архітектура оброблення даних. У класичному централізованому варіанті відеопотоки з усіх камер передаються до віддаленого сервера або хмарної платформи, де виконується основний аналіз. Такий підхід є зручним з погляду централізованого керування, але в умовах великої кількості камер створює значне мережеве навантаження, підвищує затримку і ускладнює оперативне реагування. Тому в сучасних системах дедалі частіше використовують розподілену обробку, за якої частина обчислень виконується ближче до джерела даних. Це особливо важливо для сценаріїв, у яких рішення повинно бути сформоване негайно, наприклад при виявленні несанкціонованого проникнення, небезпечної поведінки або появи об’єкта в контрольованій зоні [4, 8].

З урахуванням цього системи інтелектуального відеоспостереження дедалі частіше проєктуються як багаторівневі. На периферійному рівні відбувається збір кадрів і первинна підготовка даних. На проміжному рівні можуть виконуватися ресурсоощадні моделі виявлення об’єктів, фільтрація подій або агрегування результатів. Центральний рівень доцільно використовувати для довготривалого

збереження, аналітики, адміністрування та складнішої обробки. Така організація підвищує швидкодію системи, раціональніше використовує мережеві ресурси та полегшує масштабування.

Ще однією важливою особливістю систем відеоспостереження є безперервність функціонування. На відміну від багатьох інших прикладних програм, система спостереження працює постійно або впродовж тривалих часових інтервалів. Це висуває додаткові вимоги до відмовостійкості, стабільності оброблення потоку, ефективності використання пам’яті та коректної роботи модулів збереження. Якщо система виконує інтелектуальний аналіз, то до цих вимог додаються вимоги до стабільної роботи моделі на різних типах даних та до можливості оперативного оновлення програмних компонентів без повного припинення спостереження.

Практичне застосування таких систем охоплює велику кількість сценаріїв. У міському середовищі вони використовуються для моніторингу вулиць, перехресть, площ, транспортних потоків і громадських місць. На підприємствах вони застосовуються для контролю доступу, спостереження за виробничими зонами та виявлення порушень техніки безпеки. У навчальних закладах, офісних приміщеннях і житлових комплексах такі системи забезпечують контроль периметра, реєстрацію подій та автоматизоване сповіщення відповідальних осіб. У всіх перелічених випадках ключову роль відіграє саме здатність системи швидко інтерпретувати візуальні дані, а не лише накопичувати відеоархів.

Отже, системи відеоспостереження реального часу характеризуються безперервним надходженням великих обсягів відеоданих, високими вимогами до швидкодії, потребою в автоматичному аналізі сцен і подій, а також залежністю ефективності від архітектури оброблення даних. Їх функціональні можливості охоплюють не тільки запис і передавання відео, а й виявлення об’єктів, відстеження руху, аналіз подій, формування сповіщень та підтримку прийняття рішень. Саме ці особливості зумовили потребу в застосуванні методів комп’ютерного зору, згорткових нейронних мереж і розподілених обчислювальних підходів, що становить основу подальшого дослідження у кваліфікаційній роботі.

1.2 Згорткові нейронні мережі та туманні обчислення в задачах інтелектуального відеоспостереження

Розвиток інтелектуальних систем відеоспостереження безпосередньо пов’язаний із двома технологічними напрямами: застосуванням згорткових нейронних мереж для аналізу візуальних даних і використанням туманних обчислень для організації оброблення відеопотоку з малою затримкою. Поєднання цих підходів дало змогу перейти від звичайного перегляду та збереження відео до автоматичного виявлення об’єктів, аналізу сцен і формування повідомлень у режимі реального часу [4, 6].

Узагальнену модель оброблення відеопотоку із використанням згорткової нейронної мережі та туманних обчислень подано на рисунку 1.2.

Рисунок 1.2 – Модель інтелектуального аналізу відеопотоку на основі згорткової нейронної мережі та туманних обчислень

Згорткові нейронні мережі стали базовим інструментом сучасного комп’ютерного зору. Їх ефективність пояснюється здатністю автоматично виділяти просторові ознаки зображення на різних рівнях узагальнення. На початкових шарах така мережа визначає прості структури, наприклад контури, кути та текстурні переходи. На глибших рівнях формуються складніші ознаки, що відображають

частини об’єктів, їх форму, взаємне розміщення та цілісні візуальні шаблони. Саме тому CNN-моделі виявилися придатними для задач класифікації зображень, виявлення об’єктів, сегментації сцени, розпізнавання осіб і аналізу відеопослідовностей [10, 14, 27].

У системах відеоспостереження згорткові нейронні мережі використовуються насамперед для виявлення об’єктів у кадрі. Такий підхід дає змогу автоматично знаходити людей, транспортні засоби або інші цільові об’єкти, визначати їх положення та передавати результати до наступних модулів системи. Для цієї задачі поширення набули моделі родин Faster R-CNN, SSD, YOLO та EfficientDet. Їхня відмінність полягає у співвідношенні між точністю та швидкодією. Двоетапні моделі, як правило, забезпечують вищу точність локалізації, але потребують більших обчислювальних ресурсів. Одноетапні підходи орієнтовані на швидше опрацювання кадрів, тому частіше використовуються у прикладних системах реального часу [17, 23, 24, 30].

Порівняльну характеристику типових моделей, які можуть бути використані в системах інтелектуального відеоспостереження, наведено в таблиці 1.2.

Таблиця 1.2 – Порівняльна характеристика типових CNN-моделей для задач відеоспостереження

Модель	Основне призначення	Перевага	Обмеження
Faster R-CNN	Виявлення об’єктів	Висока точність	Нижча швидкодія
SSD	Виявлення об’єктів	Баланс точності та швидкості	Менша точність для дрібних об’єктів
YOLO	Виявлення об’єктів у реальному часі	Висока швидкодія	Може поступатися в точності локалізації
EfficientDet	Масштабоване виявлення об’єктів	Добре співвідношення точності й ресурсів	Складніша оптимізація
U-Net	Сегментація	Точне виділення областей	Не є базовою моделлю для швидкої детекції
DeepLab	Семантична сегментація	Висока якість сегментації	Вищі вимоги до ресурсів

Окрім детекції, важливе значення для інтелектуального відеоспостереження

має сегментація зображення. На відміну від задачі виявлення об’єкта прямокутною рамкою, сегментація дає змогу визначити належність окремих пікселів до певних класів. Це підвищує точність аналізу сцени, особливо в умовах складного фону, часткових перекриттів або великої щільності об’єктів. Для таких задач використовуються повнозгорткові архітектури, зокрема FCN, U-Net, PSPNet і DeepLab [3, 18, 25, 38]. Проте в межах цієї роботи доцільніше орієнтуватися на модель детекції, оскільки вона краще відповідає вимогам до реального часу та прикладної програмної реалізації.

У задачах інтелектуального відеоспостереження CNN можуть бути використані і для розпізнавання осіб, ідентифікації конкретної людини, аналізу активності, контролю переміщення об’єктів та оцінювання поведінкових шаблонів. Проте в межах кваліфікаційної роботи доцільно зосередитися на базовішому та технічно обґрунтованому сценарії – виявленні об’єктів у відеопотоці та формуванні повідомлення про подію. Такий підхід дозволяє побудувати практично реалізовану програмну систему без надмірного ускладнення архітектури.

Особливістю відеоспостереження є те, що система працює не з окремим статичним зображенням, а з послідовністю кадрів, які надходять безперервно. Це означає, що нейронна мережа повинна бути не лише точною, а й достатньо швидкою для оброблення потоку даних у наближеному до реального часу режимі. Якщо час аналізу одного кадру є занадто великим, система втрачає актуальність результату, а подія може бути зафіксована із запізненням. Саме тому для практичного використання важливими стають легкі моделі або оптимізовані версії архітектур, придатні до розгортання на обмежених апаратних платформах, зокрема на edge- або fog-вузлах [11, 21].

Не менш важливою є проблема організації обчислень у системі відеоспостереження. Якщо весь відеопотік передавати до віддаленого хмарного сервера, виникають суттєві недоліки: збільшується затримка, зростає трафік у мережі, підвищується навантаження на центральний сервер і погіршується здатність системи масштабуватися при збільшенні кількості камер. Для задач реального часу такий підхід не завжди є ефективним. Саме тому дедалі більшу роль відіграють туманні обчислення, які передбачають розміщення частини аналітики

на проміжному обчислювальному рівні між камерою та хмарою [1, 6, 8].

Туманні обчислення доцільно розглядати як розподілену модель організації сервісів, у якій оброблення даних виконується ближче до джерела їх виникнення. У системі відеоспостереження це означає, що відеопотік не обов'язково надсилається в повному обсязі до центральної інфраструктури. Частина операцій, наприклад захоплення кадрів, фільтрація, масштабування, попереднє нормалізування або навіть первинне виявлення об'єктів, може виконуватися на локальному чи проміжному вузлі. Унаслідок цього зменшується затримка, скорочується обсяг передавання даних і підвищується швидкість реакції системи на події [4, 19, 20].

Розподіл функцій між рівнями edge, fog і cloud подано на рисунку 1.3.

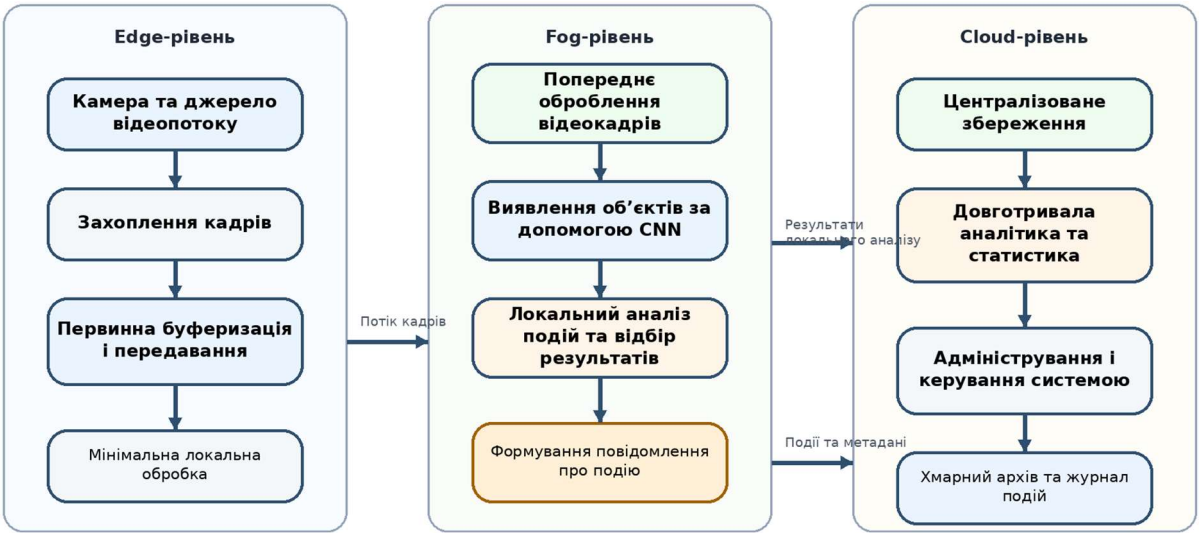

Рисунок 1.3 – Розподіл оброблення даних між edge-, fog- та cloud-рівнями

Для інтелектуального відеоспостереження така архітектура є особливо важливою. На edge-рівні можуть працювати камери та модулі первинного збору кадрів. На fog-рівні доцільно розміщувати нейромережеву обробку, пов'язану з виявленням об'єктів, фільтрацією кадрів, локальним аналізом подій або агрегуванням результатів. Cloud-рівень доцільно використовувати для довготривалого збереження даних, централізованого адміністрування, статистичного аналізу та складніших аналітичних процедур. Така багаторівнева структура дає змогу узгодити вимоги до швидкодії та до функціональної повноти системи [4, 31].

Перевага поєднання CNN і fog computing полягає в тому, що згорткова нейронна мережа виконує інтелектуальний аналіз відеокадру, а туманна інфраструктура забезпечує раціональний розподіл цього аналізу між обчислювальними вузлами. У результаті система здатна не просто розпізнавати об’єкти, а робити це достатньо швидко для практичної експлуатації. Особливо важливо, що такий підхід зменшує залежність від постійного високошвидкісного каналу зв’язку з хмарою, що є суттєвим для розподілених або ресурсно обмежених систем спостереження.

Разом із перевагами слід враховувати і певні обмеження. Розгортання нейромережевих моделей на edge- і fog-вузлах потребує добору компактних архітектур, оптимізації розміру моделі, урахування обмежень пам’яті та продуктивності пристрою. Крім того, у багаторівневій системі ускладнюється взаємодія між модулями, зростають вимоги до синхронізації та надійності передавання результатів. Проте саме такий підхід є найбільш перспективним для побудови систем інтелектуального відеоспостереження реального часу, оскільки він поєднує високий рівень автоматизації аналізу з практичною ефективністю оброблення.

Отже, згорткові нейронні мережі забезпечують основу інтелектуального аналізу відеоданих, оскільки дають змогу автоматично виявляти й класифікувати об’єкти в кадрі, а туманні обчислення створюють придатне середовище для виконання таких обчислень із малою затримкою. Поєднання цих двох підходів є методично та технічно обґрунтованою основою для розроблення програмної системи інтелектуального відеоспостереження в режимі реального часу.

1.3 Аналіз існуючих рішень інтелектуального відеоспостереження

Сучасні системи інтелектуального відеоспостереження розвиваються у напрямі автоматизації аналізу відеопотоку, зменшення участі оператора в рутинному перегляді кадрів і підвищення швидкості реагування на події. Існуючі рішення відрізняються за архітектурою, способом оброблення даних, рівнем інтелектуалізації та складністю програмної реалізації. Частина з них орієнтована

переважно на централізоване зберігання та базовий моніторинг, тоді як сучасніші системи використовують методи комп'ютерного зору, глибокого навчання та розподіленого оброблення даних [4, 5].

Класифікацію основних існуючих рішень інтелектуального відеоспостереження подано на рисунку 1.4.

Найпростішу групу становлять традиційні системи відеоспостереження, у яких основною функцією є отримання відео з камер, його передавання на сервер, збереження в архіві та виведення зображення оператору. Такі рішення залишають значну частину аналітичної роботи людині. Оператор повинен самостійно відстежувати події, аналізувати сцену та визначати, чи є ситуація потенційно небезпечною. За невеликої кількості камер такий підхід ще може бути прийнятним, однак у масштабних системах він втрачає ефективність. Зі збільшенням кількості відеопотоків зростає навантаження на персонал, знижується ймовірність своєчасного виявлення події та підвищується ризик пропуску важливої інформації.

Рисунок 1.4 – Класифікація існуючих рішень інтелектуального відеоспостереження

Інший напрям представлений хмарними системами відеоспостереження, у яких відеодані надсилаються до віддалених серверів або дата-центрів, де виконується зберігання та аналіз. Перевагою таких рішень є централізоване адміністрування, доступність з різних пристроїв, гнучке масштабування сховища та зручність інтеграції з іншими сервісами. Проте для задач реального часу хмарний підхід має очевидні обмеження. Постійне передавання великих обсягів

відеоданих створює значне мережеве навантаження, а затримка між надходженням кадру та результатом аналізу може бути надто великою для оперативного реагування [1, 5, 33].

З розвитком методів штучного інтелекту з'явилися системи, у яких основний акцент зроблено на автоматичному виявленні об'єктів і подій у кадрі. У таких рішеннях застосовуються згорткові нейронні мережі, здатні розпізнавати людей, транспортні засоби, елементи сцени або інші цільові об'єкти. На практиці це дало змогу реалізувати функції виявлення руху нового рівня, підрахунку об'єктів, контролю визначених зон, розпізнавання осіб, аналізу траєкторій переміщення та виявлення атипової поведінки. Саме цей клас рішень можна вважати основою сучасного інтелектуального відеоспостереження, оскільки він забезпечує перехід від пасивного запису до автоматизованого аналізу відеоконтенту [23, 24, 43].

Окрему групу становлять системи, побудовані на базі edge- та fog-підходів. У них частина обчислень переноситься ближче до джерела даних, що є особливо важливим для задач відеоаналітики. Наприклад, первинне виявлення об'єктів, сегментація або фільтрація подій можуть виконуватися не в хмарі, а на проміжному вузлі. Така організація дозволяє зменшити обсяг даних, що передаються мережею, скоротити затримку та підвищити оперативність реакції системи на подію. Саме цей підхід розглядається як найбільш перспективний для систем інтелектуального відеоспостереження реального часу [4, 19, 20].

Порівняння основних підходів до побудови систем інтелектуального відеоспостереження наведено в таблиці 1.3.

У наукових і прикладних працях можна виокремити кілька типових напрямів реалізації інтелектуального відеоспостереження. Перший напрям пов'язаний із виявленням об'єктів у відеокадрі. Такі системи орієнтовані на визначення наявності людини, автомобіля або іншого об'єкта та фіксацію його координат. Другий напрям стосується сегментації сцени, коли об'єкт виділяється точніше, на рівні його просторової області. Третій напрям охоплює трекінг, тобто безперервне відстеження об'єкта в послідовності кадрів. Четвертий напрям спрямований на аналіз дій і поведінки, коли система не лише бачить об'єкт, а й оцінює його активність або характер руху. У практичних системах ці підходи часто

поєднуються, але ступінь їх інтеграції залежить від складності архітектури та доступних ресурсів.

Таблиця 1.3 – Порівняння існуючих підходів до побудови систем інтелектуального відеоспостереження

Тип рішення	Характеристика	Переваги	Недоліки
Традиційне централізоване відеоспостереження	Запис і перегляд відео без глибокої аналітики	Простота реалізації	Велике навантаження на оператора
Хмарне відеоспостереження	Оброблення й зберігання на віддаленому сервері	Централізоване керування	Висока затримка, мережеве навантаження
AI-система на CNN	Автоматичне виявлення об’єктів і подій	Інтелектуальний аналіз кадрів	Високі обчислювальні вимоги
Edge/Fog-орієнтована система	Частина оброблення переноситься ближче до камери	Менша затримка, менше трафіку	Складніша архітектура

Розглядаючи конкретні приклади, слід відзначити рішення, орієнтовані на оброблення міських відеопотоків у fog-середовищі. У таких системах наголос зроблено на розподілі навантаження між локальними вузлами та центральною платформою. Це дає змогу опрацьовувати потоки з декількох камер більш ефективно, ніж у повністю централізованій моделі [4]. Інші дослідження акцентують увагу на застосуванні fog computing для скорочення вартості оброблення, зменшення пропускної здатності каналу та швидшого формування результату під час аналізу відео у смарт-місті [19]. Такі рішення підтверджують доцільність використання багаторівневої архітектури для задач спостереження.

Перспективним є і напрям поєднання відеоспостереження з новішими моделями штучного інтелекту, зокрема підходами візуально-мовного спостереження та токенізації, орієнтованої на дані. Такі системи розширюють можливості класичного спостереження, оскільки потенційно дають змогу не лише виявляти об’єкти, а й формувати семантично узгоджені представлення подій для швидкого передавання й інтерпретації в edge-cloud середовищі [31]. Проте для

кваліфікаційної роботи такі рішення є надто складними, оскільки вимагають значно ширшої моделі даних, складнішого конвеєра оброблення та вищих обчислювальних витрат.

Окремо слід розглянути прикладні системи, пов’язані з аналізом окремих типів об’єктів у зображеннях. До них належать рішення для підрахунку автомобілів на супутникових знімках, виявлення сонячних панелей, ідентифікації особи у відеопотоці або розпізнавання спортивної активності [9, 13, 39, 41]. Хоча ці системи не є повноцінними комплексами відеоспостереження, вони демонструють практичну придатність нейромережевих моделей до виділення об’єктів, класифікації візуальних сцен і роботи з реальними зображеннями. Їх цінність полягає в тому, що вони підтверджують універсальність CNN-підходу та можливість його адаптації до різних прикладних сценаріїв.

Українські дослідження також засвідчують зростання інтересу до застосування згорткових нейронних мереж у задачах комп’ютерного зору, безпеки та відеоаналітики. У роботах, присвячених використанню CNN для розпізнавання об’єктів у відеопотоці, наголошено на доцільності автоматизації аналізу кадрів і підвищення точності виявлення об’єктів у системах безпеки [40, 43]. Дослідження з ідентифікації особи у відеопотоці показують, що машинне навчання може бути ефективною основою для побудови інтелектуальних систем контролю доступу та відеомоніторингу [41]. Роботи, присвячені покращенню пошуку відеофрагментів, підтверджують актуальність автоматизованого аналізу й структурування відеоконтенту як окремого напряму розвитку інтелектуальних систем відеооброблення [42].

Попри значний прогрес, більшість існуючих рішень мають певні обмеження. Частина систем забезпечує високу точність, але потребує суттєвих апаратних ресурсів і тому є складною для розгортання на периферійних або проміжних вузлах. Інші рішення є швидкими, але демонструють нижчу точність або обмежену функціональність. Досить поширеною є ситуація, коли дослідження зосереджується лише на одному модулі, наприклад на виявленні об’єкта, без повного проєктування програмної системи як цілісного комплексу. Крім того, далеко не всі роботи приділяють достатню увагу саме режиму реального часу,

тобто вимірюванню затримки, швидкодії та стійкості системи в умовах безперервного потоку даних.

Відомі рішення або надто орієнтовані на окрему модель, або описують загальну архітектуру без достатньо конкретної програмної реалізації. Це створює підстави для розроблення програмної системи, яка поєднає кілька ключових вимог: використання згорткової нейронної мережі для виявлення об’єктів у відеопотоці, підтримку функціонування в режимі реального часу та організацію оброблення даних у туманному середовищі. Саме така система є практично доцільною, технічно обґрунтованою та відповідає сучасним тенденціям розвитку інтелектуального відеоспостереження.

Отже, аналіз існуючих рішень показав, що сучасні системи інтелектуального відеоспостереження розвиваються у напрямі поєднання комп’ютерного зору, нейромережевого аналізу та розподіленого оброблення даних. Найбільш перспективними є рішення, що забезпечують автоматичне виявлення об’єктів у кадрі та переносять частину обчислень на edge- або fog-рівень. Водночас наявні розробки не повною мірою поєднують точність, швидкодію, простоту реалізації та придатність до використання в режимі реального часу.

1.4 Постановка задачі розроблення програмної системи

Проведений аналіз предметної області показав, що сучасні системи відеоспостереження дедалі активніше переходять від простого запису відео до автоматизованого аналізу подій у потоці кадрів. Використання згорткових нейронних мереж суттєво підвищило ефективність виявлення об’єктів, а застосування туманних обчислень створило умови для перенесення частини оброблення ближче до джерела даних. Це особливо важливо для систем, які повинні працювати в режимі реального часу та забезпечувати швидке реагування на події.

Разом із тим аналіз існуючих рішень засвідчив, що більшість із них мають певні обмеження. Частина систем орієнтована переважно на централізоване збереження та перегляд відео, тому не забезпечує достатнього рівня автоматизації

аналізу. Інші рішення реалізують інтелектуальне виявлення об’єктів, але потребують значних апаратних ресурсів або не враховують особливості розподіленого оброблення даних. Також поширеними є системи, у яких увагу зосереджено на окремому алгоритмі чи моделі, без побудови цілісної програмної архітектури, придатної до практичного використання в умовах безперервного відеопотоку.

Для задач інтелектуального відеоспостереження реального часу важливими є одразу кілька характеристик: здатність системи своєчасно обробляти потік кадрів, автоматично виявляти об’єкти у сцені, інтерпретувати результат виявлення як подію прикладного рівня та забезпечувати подальше збереження або передавання результату. За відсутності такої узгодженої роботи програмних компонентів система або втрачає оперативність, або зводиться лише до часткового розв’язання задачі.

Отже, у межах даної кваліфікаційної роботи поставлено задачу розроблення програмної системи інтелектуального відеоспостереження, яка забезпечує автоматичне виявлення об’єктів у відеопотоці в режимі реального часу на основі згорткової нейронної мережі та використовує принципи туманних обчислень для організації оброблення даних. Така система повинна поєднувати програмні засоби приймання відеопотоку, підготовки кадрів, нейромережевого аналізу, інтерпретації результатів та формування повідомлення про подію.

У загальному вигляді вхідними даними для системи є відеопотік, що надходить із камери або з тестового відеоджерела. У процесі функціонування системи ці дані повинні пройти послідовні етапи оброблення: отримання кадрів, попередню підготовку, подання до моделі згорткової нейронної мережі, аналіз результатів виявлення та формування реакції системи. Вихідним результатом повинна бути не лише технічна інформація про знайдені об’єкти, а й змістовний результат прикладного рівня, зокрема фіксація події, збереження відповідних даних або формування сповіщення.

Отже, метою кваліфікаційної роботи є розроблення програмної системи інтелектуального відеоспостереження, що забезпечує аналіз відеопотоку в режимі реального часу на основі згорткових нейронних мереж і туманних обчислень.

Для досягнення поставленої мети сформовано такі завдання дослідження:

– проаналізувати особливості систем відеоспостереження реального часу та сучасні підходи до їх інтелектуалізації;

– дослідити можливості застосування згорткових нейронних мереж у задачах аналізу відеопотоку;

– розглянути принципи використання туманних обчислень у системах реального часу;

– виконати аналіз існуючих рішень інтелектуального відеоспостереження;

– сформувати вимоги до програмної системи та розробити її загальну архітектуру;

– спроєктувати функціональні модулі системи та алгоритми її роботи;

– реалізувати основні програмні компоненти системи;

– провести експериментальні дослідження роботи розробленої системи та оцінити отримані результати.

2 ПРОЄКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ ІНТЕЛЕКТУАЛЬНОГО ВІДЕОСПОСТЕРЕЖЕННЯ

2.1 Формування вимог і загальної архітектури програмної системи

Розроблення програмної системи інтелектуального відеоспостереження потребує попереднього формування вимог, які визначають її функціональні можливості, особливості структури, режим роботи та обмеження реалізації.

Розроблювана система повинна бути орієнтована на аналіз відеопотоку в режимі реального часу або в наближеному до нього режимі. Це означає, що її функціонування має забезпечувати достатньо швидке опрацювання послідовності кадрів для виявлення цільових об’єктів без критичного запізнення результату. У межах кваліфікаційної роботи не ставиться задача створення високонавантаженої промислової платформи, проте система повинна демонструвати працездатну логіку автоматизованого відеоспостереження, модульну організацію та придатність до подальшого розширення.

Насамперед було сформовано функціональні вимоги до системи. Вона повинна приймати відеопотік із джерела даних, виконувати виділення кадрів, забезпечувати їх попереднє оброблення та передавати підготовлені дані до модуля нейромережевого аналізу. Далі система повинна виявляти об’єкти у кадрі на основі згорткової нейронної мережі, інтерпретувати результати детекції відповідно до прикладної логіки та формувати повідомлення про подію. Окрім цього, вона повинна забезпечувати збереження результатів оброблення та підтримувати можливість подальшого перегляду або аналізу отриманих даних.

Функціональні вимоги до програмної системи наведено в таблиці 2.1.

Окрім функціональних вимог, важливе значення мають нефункціональні вимоги, оскільки саме вони визначають якість роботи системи, її архітектурні властивості та придатність до практичного використання. Для розроблюваної системи особливо важливими є вимоги до швидкодії, модульності, масштабованості, зрозумілості структури та можливості розподілу функцій між різними рівнями обчислювальної інфраструктури. Оскільки система проєктується з урахуванням fog-підходу, її архітектура повинна передбачати таку організацію, за

якої оперативне оброблення виконується ближче до джерела даних, а більш ресурсоємні або довготривалі функції можуть бути винесені на інші рівні.

Таблиця 2.1 – Функціональні вимоги до програмної системи інтелектуального відеоспостереження

№	Функціональна вимога	Зміст
1	Приймання відеопотоку	Отримання даних із камери або тестового відеоджерела
2	Формування кадрів	Виділення послідовності кадрів для подальшого аналізу
3	Попереднє оброблення	Масштабування, нормалізація та підготовка кадрів
4	Виявлення об’єктів	Використання CNN-моделі для детекції об’єктів у кадрі
5	Аналіз результатів	Перевірка умов виникнення контрольованої події
6	Формування повідомлення	Генерація повідомлення про виявлену подію
7	Збереження результатів	Запис кадрів, журналу подій або службових даних

Нефункціональні вимоги до системи узагальнено в таблиці 2.2.

Таблиця 2.2 – Нефункціональні вимоги до програмної системи

Група вимог	Характеристика
Швидкодія	Оброблення кадрів із затримкою, що не порушує логіку реального часу
Модульність	Поділ системи на незалежні функціональні компоненти
Масштабованість	Можливість розширення функцій або зміни джерел даних
Надійність	Стійке виконання основних операцій оброблення відеопотоку
Розширюваність	Можливість заміни моделі або додавання нових модулів
Адаптивність	Придатність до роботи з різними сценаріями спостереження

З урахуванням сформованих вимог було визначено, що архітектура системи повинна бути модульною. Такий підхід є найбільш доцільним для дослідницької програмної системи, оскільки дозволяє розділити загальну задачу на низку послідовних підзадач і реалізувати кожну з них у вигляді окремого логічного компонента. Крім того, модульна архітектура полегшує тестування, налагодження

та модифікацію системи, а також забезпечує можливість подальшого перенесення окремих функцій між edge-, fog- і cloud-рівнями.

Узагальнену архітектуру програмної системи спроєктовано у вигляді кількох взаємопов'язаних модулів, кожен із яких виконує чітко визначену функцію (рисунок 2.1).

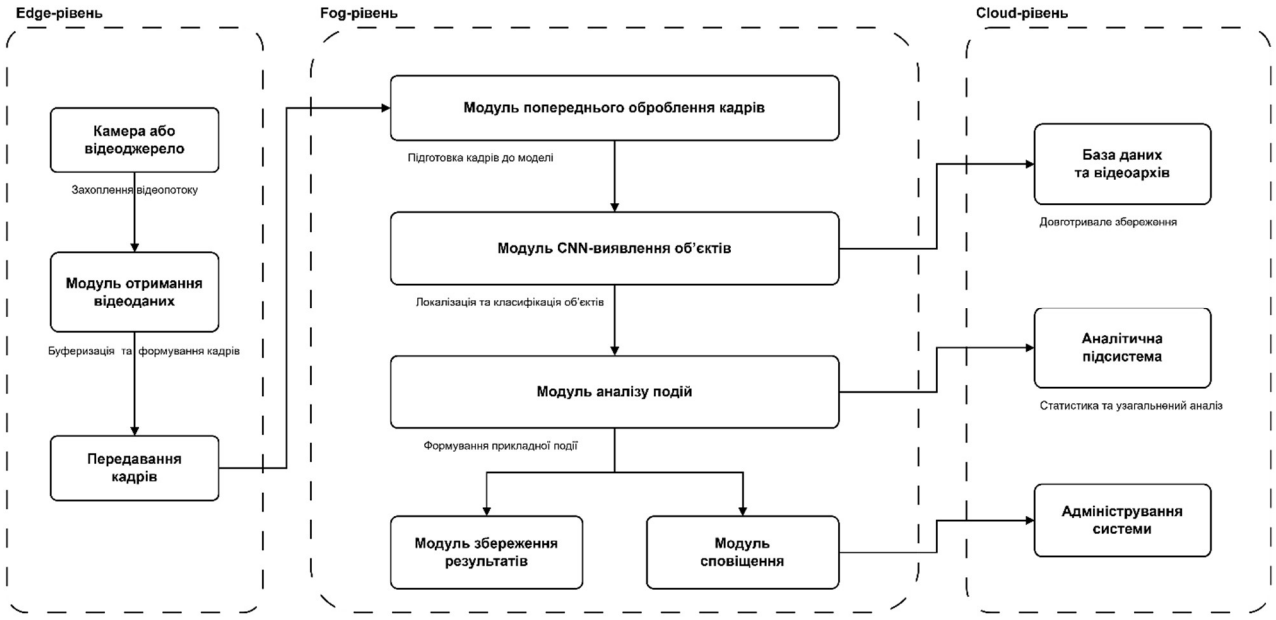

Рисунок 2.1 – Загальна архітектура програмної системи інтелектуального відеоспостереження [44]

Першим функціональним компонентом є модуль отримання відеоданих. Його призначення полягає у підключенні до джерела відео та формуванні послідовності кадрів для подальшої обробки. В якості джерела можуть використовуватися IP-камера, вебкамера або тестовий відеофайл. На цьому етапі система повинна забезпечувати стабільне надходження кадрів, їх зчитування у правильному порядку та передавання до наступного модуля без порушення часової послідовності. Цей модуль є початковою точкою роботи всієї системи, тому від його коректного функціонування залежить стабільність подальшого аналізу [44].

Другим компонентом є модуль попереднього оброблення кадрів. Він призначений для приведення відеоданих до формату, придатного для аналізу нейронною мережею. У його межах виконуються масштабування кадру, зміна роздільної здатності, перетворення кольорового простору за потреби, нормалізація вхідних даних та усунення надлишкових елементів, які можуть ускладнювати

подальше оброблення. Важливість цього модуля полягає в тому, що якість попередньої підготовки кадрів безпосередньо впливає на точність роботи моделі виявлення об’єктів [44].

Третім і центральним компонентом системи є модуль виявлення об’єктів на основі згорткової нейронної мережі. Саме в ньому реалізується інтелектуальна складова всієї системи. На вхід цього модуля подається підготовлений кадр, а на виході формується набір знайдених об’єктів із координатами, класами та оцінкою впевненості. У контексті цієї роботи основною задачею такого модуля є виявлення цільових об’єктів у кадрі з прийнятною швидкодією. Тому модель повинна бути достатньо точною, але водночас придатною до використання в умовах обмежених обчислювальних ресурсів fog-рівня.

Після отримання результатів детекції вони передаються до модуля аналізу подій. Його функція полягає в інтерпретації результатів нейромережевого виявлення з позиції прикладної логіки. Якщо модуль CNN-виявлення визначає технічний факт наявності об’єкта в кадрі, то модуль аналізу подій визначає змістовний факт події. Наприклад, саме тут перевіряється, чи належить знайдений об’єкт до контрольованого класу, чи виконуються умови фіксації події, чи слід передати інформацію на збереження та чи потрібно згенерувати повідомлення. Таким чином, цей модуль є сполучною ланкою між комп’ютерним зором і прикладною логікою системи спостереження [44].

П’ятим компонентом є модуль збереження результатів. Його призначення полягає у фіксації результатів роботи системи у структурованому вигляді. До таких результатів можуть належати кадри з виявленими об’єктами, службові метадані, записи журналу подій, часові мітки, класи об’єктів та рівні впевненості детекції. Збереження результатів є необхідним не лише для архівування, а й для подальшого аналізу роботи системи, перевірки коректності її функціонування та формування статистичних звітів [44].

Шостим компонентом є модуль сповіщення, який відповідає за формування реакції системи на виявлену подію. Його робота може полягати у відображенні повідомлення в інтерфейсі, створенні текстового запису, передаванні сигналу до зовнішньої підсистеми або в іншій формі інформування користувача [44].

Основні модулі програмної системи та їх призначення узагальнено в таблиці 2.3.

Таблиця 2.3 – Основні модулі програмної системи та їх призначення

Модуль	Призначення
Модуль отримання відеоданих	Підключення до джерела відео та формування потоку кадрів
Модуль попереднього оброблення	Підготовка кадрів до аналізу нейронною мережею
Модуль виявлення об’єктів	Виконання детекції об’єктів у кадрі
Модуль аналізу подій	Інтерпретація результатів детекції та перевірка умов події
Модуль збереження результатів	Запис кадрів, журналу подій та службової інформації
Модуль сповіщення	Формування повідомлень про виявлені події

Особливістю архітектури є те, що вона повинна враховувати принципи туманних обчислень. Це означає, що не всі модулі системи мають бути прив’язані до одного обчислювального вузла. Найбільш чутливі до затримки функції, пов’язані з прийманням відеопотоку, попереднім обробленням кадрів і первинним виявленням об’єктів, доцільно виконувати на fog-рівні. Такий підхід дозволяє скоротити час між надходженням кадру та отриманням результату. Функції, пов’язані з архівуванням, накопиченням статистики та централізованим керуванням системою, можуть виконуватися на cloud-рівні. Edge-рівень у цій структурі представлений насамперед джерелом відеопотоку та базовими операціями отримання даних.

Отже, у цьому підрозділі спроєктовано узагальнену структуру програмної системи інтелектуального відеоспостереження та визначено призначення модулів, їх результати роботи і місце в загальній архітектурі.

2.2 Розроблення моделі оброблення відеопотоку на основі згорткової нейронної мережі

Ключовим елементом програмної системи інтелектуального відеоспостереження є модель оброблення відеопотоку, яка забезпечує перетворення вхідного кадру у результат сегментації та подальше виділення цільового об’єкта. На відміну від спрощеного підходу, за якого результатом роботи моделі є лише факт наявності об’єкта або його прямокутна область, у цій роботі використано модель, орієнтовану на побудову маски об’єкта. Такий підхід є доцільним для задач інтелектуального відеоспостереження, оскільки дозволяє точніше відокремити об’єкт від фону та підвищити інформативність подальшого аналізу.

Для розроблюваної системи обрано модель на основі згорткової нейронної мережі типу encoder–decoder. Вона забезпечує покадрове оброблення відеопотоку, у межах якого кожен кадр проходить послідовні етапи стискання ознак, формування компактного представлення сцени, відновлення просторової структури та побудови вихідної маски. Після цього за допомогою окремого блоку Mask-Crop формується виділена область об’єкта, яка може бути використана для подальшого прикладного аналізу.

Структуру розробленої моделі подано на рисунку 2.2.

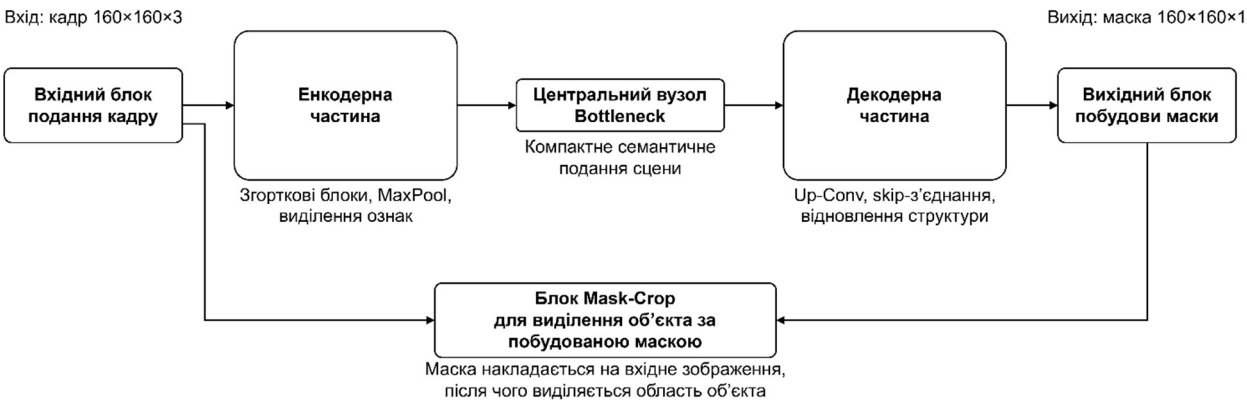

Рисунок 2.2 – Структура моделі оброблення відеопотоку на основі згорткової нейронної мережі encoder–decoder

Структурно модель складається з таких основних блоків:

– вхідний блок подання кадру;
– енкодерна частина;
– центральний вузол або bottleneck;
– декодерна частина;
– вихідний блок побудови маски;
– блок Mask-Crop для виділення об’єкта за побудованою маскою.

Вхідний блок подання кадру призначений для приймання поточного зображення з відеопотоку та приведення його до формату, сумісного з нейромережевою моделлю. У межах цієї роботи кадр розглядається як базова одиниця аналізу. Перед поданням у мережу він має бути нормалізований і приведений до фіксованого розміру. Таким чином, вхідний блок формує уніфіковане подання даних, на основі якого працюють усі наступні компоненти моделі.

Енкодерна частина виконує послідовне виділення ознак із вхідного кадру. На цьому етапі мережа переходить від початкового зображення до багаторівневого ознакового подання сцени. У процесі такого переходу просторовий розмір карт ознак зменшується, а кількість семантичної інформації збільшується. Саме енкодер забезпечує формування високорівневих ознак, що відображають форму, контури та розміщення об’єкта в кадрі. У практичному сенсі цей блок виконує стискання вхідної інформації з одночасним посиленням її змістовності.

Центральний вузол bottleneck є найглибшим рівнем моделі. У ньому формується компактне семантичне представлення сцени, яке містить найбільш узагальнену інформацію про об’єкт і фон. Цей блок виконує роль проміжної ланки між енкодером і декодером. Його призначення полягає не лише в подальшому стисканні ознак, а й у збереженні найбільш значущої інформації, необхідної для подальшого відновлення просторової структури об’єкта.

Декодерна частина виконує зворотне перетворення ознак. Якщо енкодер поступово зменшував просторову роздільну здатність, то декодер відновлює її, поєднуючи глибокі семантичні ознаки з інформацією про локальну структуру об’єкта. Саме на цьому етапі формується наближене просторове представлення маски. Декодер є критично важливим для сегментації, оскільки він забезпечує

повернення від стислого ознакового подання до зображення, у якому можна визначити межі об’єкта по пікселях.

Вихідний блок побудови маски формує результат роботи моделі у вигляді одноканальної маски. У такій масці одна частина пікселів відповідає об’єкту або області дії, а інша – фону. Саме цей блок завершує нейромережевий цикл оброблення кадру. На відміну від класичної детекції, де результатом є координати рамки, у цьому випадку результатом є просторово точніше піксельне подання об’єкта.

Після побудови маски використовується блок Mask-Crop, який виконує виділення об’єкта за сформованою маскою. Його функція полягає у накладанні маски на початковий кадр і відокремленні цільової області від надлишкового фону. Унаслідок цього формується очищене представлення об’єкта, яке є більш придатним для наступного етапу аналізу, ніж повний кадр. Такий підхід зменшує вплив фонового шуму та підвищує інформативність оброблення.

У межах цієї моделі вхідними даними є кадри відеопотоку, а вихідними – маска об’єкта та виділена за нею область. Така структура вихідних даних є більш інформативною, ніж звичайний список координат, оскільки дає змогу перейти до точнішого просторового аналізу. Це особливо важливо у випадках, коли фон є складним або об’єкт має нерегулярну форму.

Вхідні та вихідні дані моделі наведено в таблиці 2.4.

Таблиця 2.4 – Вхідні та вихідні дані моделі

Етап моделі	Вхідні дані	Вихідні дані
Вхідний блок	Кадр відеопотоку	Підготовлений кадр
Енкодер	Підготовлений кадр	Ознакові карти
Bottleneck	Ознакові карти енкодера	Компактне семантичне представлення
Декодер	Стиснені ознаки	Відновлені ознаки
Вихідний блок	Ознаки декодера	Маска об’єкта
Mask-Crop	Маска та початковий кадр	Виділена область об’єкта

Розроблена модель має послідовну логіку функціонування. Спочатку поточний кадр подається на вхідний блок, потім проходить через енкодер і

bottleneck, після чого відновлюється в декодері та завершується побудовою маски. Далі блок Mask-Crop формує виділений фрагмент об’єкта. Таким чином, результатом роботи моделі є не лише ознака наявності цільового об’єкта, а конкретна сегментована область, яка може бути використана в системі для формування події або збереження результату.

Важливою перевагою такої моделі є її придатність до інтеграції в систему відеоспостереження реального часу. Хоча сегментація є складнішою за базову детекцію, вона забезпечує значно точніше виділення об’єкта. Для прикладної системи це означає можливість формування більш змістовного результату та зменшення впливу фону на подальші етапи аналізу. Саме тому для цієї роботи використання encoder–decoder-моделі є методично виправданим.

Отже, у цьому підрозділі розроблено модель оброблення відеопотоку на основі згорткової нейронної мережі типу encoder–decoder, структура якої включає вхідний блок подання кадру, енкодерну частину, центральний вузол bottleneck, декодерну частину, вихідний блок побудови маски та блок Mask-Crop. Така модель дозволяє перейти від звичайного кадру відеопотоку до сегментованого подання об’єкта, що підвищує інформативність подальшого аналізу в системі інтелектуального відеоспостереження.

2.3 Алгоритми функціонування системи та взаємодії між рівнями

Алгоритмічний опис дозволяє формалізувати послідовність дій системи, встановити порядок взаємодії між модулями та показати, яким чином реалізується розподіл оброблення між edge-, fog- і cloud-рівнями. Для системи інтелектуального відеоспостереження це має особливе значення, оскільки її робота пов’язана з безперервним потоком відеоданих, циклічним виконанням операцій і необхідністю оперативного формування результату.

У межах цієї роботи розглянемо два основні алгоритми. Перший алгоритм описує загальне функціонування програмної системи інтелектуального відеоспостереження як цілісного комплексу. Другий алгоритм визначає взаємодію між edge-, fog- і cloud-рівнями, тобто розподіл даних і функцій у багаторівневому

середовищі. Такий підхід дозволяє окремо показати внутрішню логіку оброблення відеопотоку та архітектурну логіку обміну результатами між рівнями системи.

Першочерговим є алгоритм загального функціонування системи. Його призначення полягає в описі повного циклу оброблення відеопотоку – від отримання кадру до формування події, збереження результатів і переходу до наступного циклу. Цей алгоритм має циклічний характер, оскільки система працює безперервно, а кожен новий кадр проходить однакову послідовність оброблення.

Алгоритм функціонування програмної системи інтелектуального відеоспостереження подано у вигляді кроків:

1. Ініціалізувати програмну систему та підключити джерело відеопотоку.
2. Отримати черговий кадр із камери або тестового відеоджерела.
3. Перевірити коректність отриманого кадру.
4. Передати кадр до модуля попереднього оброблення.
5. Виконати масштабування, нормалізацію та підготовку кадру до подання у модель.
6. Передати підготовлений кадр до модуля CNN-виявлення об’єктів.
7. Отримати набір прогнозів моделі.
8. Виконати фільтрацію результатів за порогом достовірності.
9. Передати достовірні результати до модуля аналізу подій.
10. Перевірити, чи виявлено об’єкт цільового класу.
11. Якщо об’єкт виявлено, сформувати подію.
12. Передати результати до модуля збереження та до модуля сповіщення.
13. Якщо подію не виявлено, перейти до оброблення наступного кадру.
14. Повторювати цикл доти, доки активне джерело відеопотоку.

Схему цього алгоритму подано на рисунку 2.3.

Запропонований алгоритм відображає основну логіку функціонування системи. Його особливістю є те, що всі етапи оброблення пов’язані в єдиний конвеєр. У межах цього конвеєра кожний блок виконує завершену функцію, а результат одного етапу є вхідними даними для наступного. Завдяки такій організації система має чітку структуру, що спрощує програмну реалізацію, тестування та подальшу модифікацію окремих компонентів.

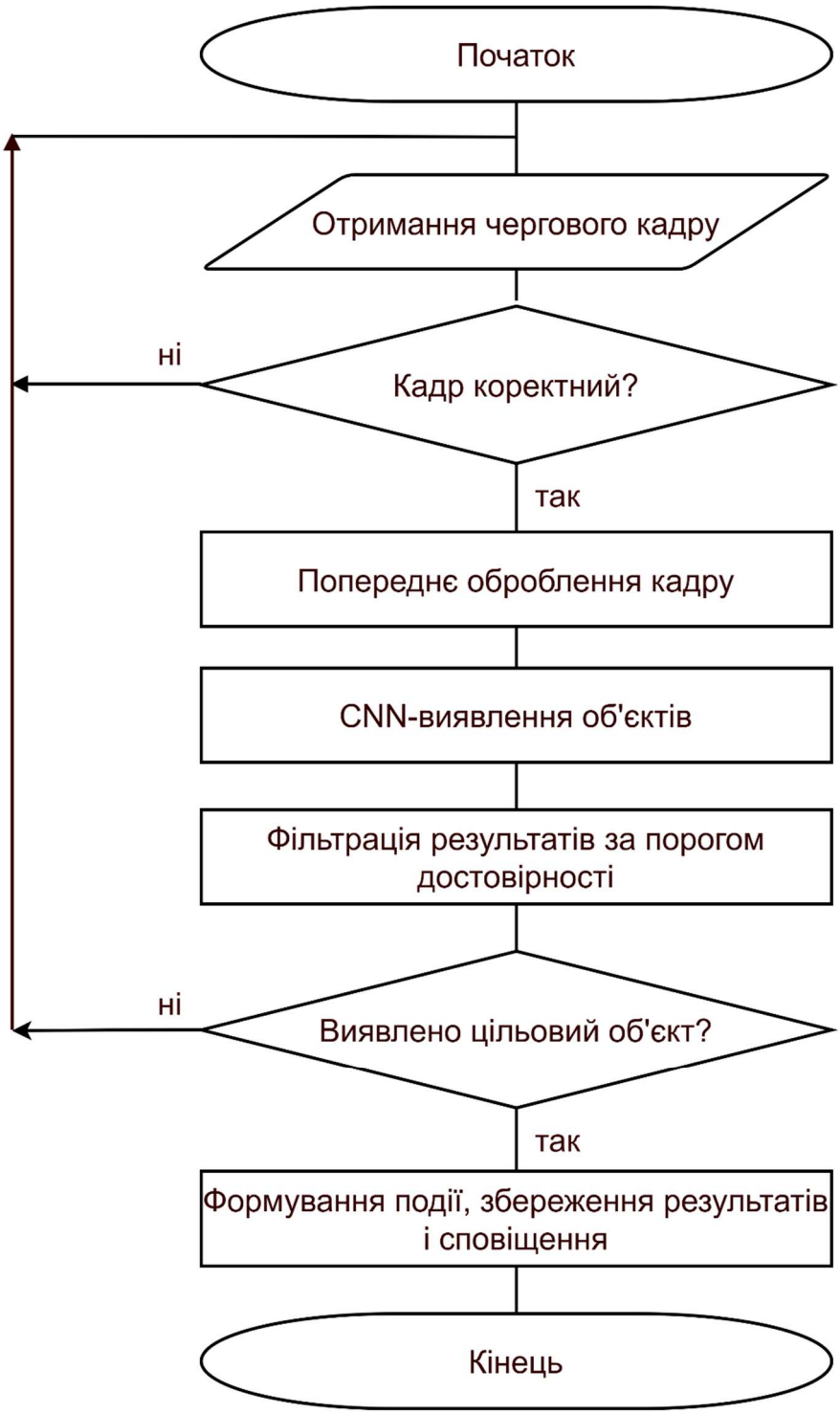

Рисунок 2.3 – Схема алгоритму функціонування програмної системи інтелектуального відеоспостереження

Однак для системи, побудованої на принципах туманних обчислень, недостатньо описати лише внутрішній цикл оброблення кадру. Не менш важливо формалізувати алгоритм взаємодії між edge-, fog- і cloud-рівнями, оскільки саме він визначає, де саме виконуються ті чи інші функції та які дані передаються між

обчислювальними рівнями. Це дає змогу показати, як архітектурне рішення впливає на швидкодію системи та на зменшення навантаження на мережу.

У запропонованій системі edge-рівень відповідає за формування відеопотоку та передавання кадрів до fog-рівня. Fog-рівень виконує основний цикл оперативного аналізу, включаючи попереднє оброблення, нейромережеве виявлення об’єктів і первинне формування події. Cloud-рівень використовується для довготривалого збереження результатів, аналітики та централізованого адміністрування. Таким чином, алгоритм взаємодії між рівнями будується навколо принципу: оперативне оброблення виконується ближче до джерела даних, а ресурсоємні та довготривалі функції – на віддаленому рівні.

Алгоритм взаємодії між edge-, fog- і cloud-рівнями:

1. Edge-рівень формує потік відеокадрів із камери або іншого відеоджерела.
2. Отримані кадри передаються до fog-рівня.
3. Fog-рівень виконує попереднє оброблення кадрів.
4. Fog-рівень запускає модель згорткової нейронної мережі для виявлення об’єктів.
5. Результати детекції аналізуються на fog-рівні відповідно до правил формування події.
6. Якщо подію виявлено, fog-рівень формує прикладний результат.
7. Результат події, кадр або метадані передаються до cloud-рівня.
8. Cloud-рівень виконує збереження результатів у базі даних або архіві.
9. Cloud-рівень забезпечує подальшу аналітичну обробку, накопичення статистики та адміністрування.
10. Fog-рівень продовжує оброблення наступних кадрів без очікування завершення довготривалих cloud-операцій.

Схему цього алгоритму подано на рисунку 2.4.

Логіка цього алгоритму демонструє, що ключовою особливістю системи є винесення найбільш чутливих до часу операцій на fog-рівень. Саме тут виконується основний обчислювальний цикл, від якого залежить оперативність виявлення події. Якби ці функції були повністю перенесені до хмари, система стала б більш залежною від пропускної здатності мережі та віддаленого сервера, що призвело б

до збільшення затримки. Отже, запропонований алгоритм взаємодії між рівнями є безпосереднім наслідком вимог до роботи системи в режимі реального часу.

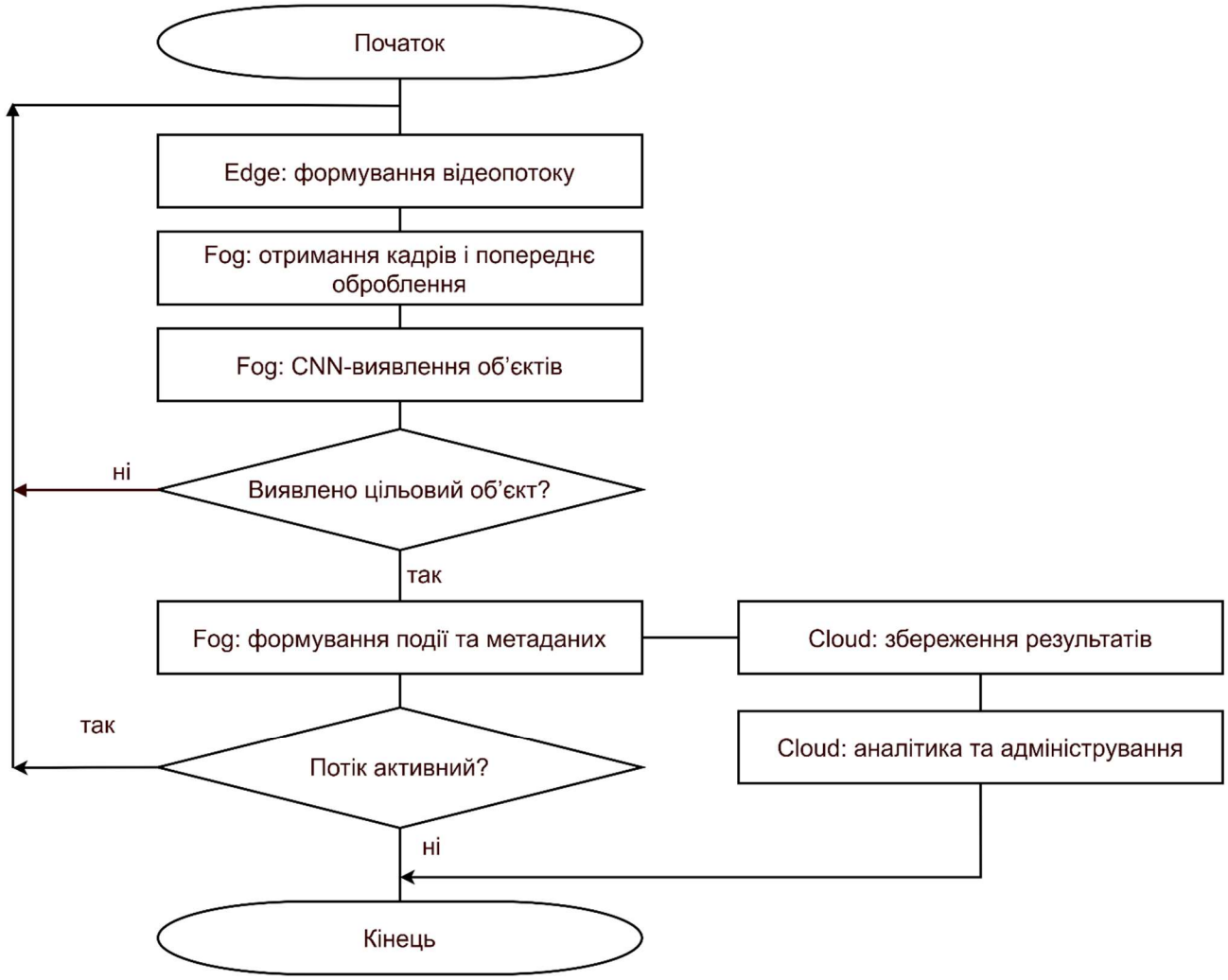

Рисунок 2.4 – Схема алгоритму взаємодії між edge-, fog- і cloud-рівнями

Окрім двох базових алгоритмів, також розглянемо логіку прийняття рішення про подію, оскільки саме вона поєднує нейромережевий результат із прикладною реакцією системи. У розроблюваній системі таке рішення приймається за простим правилом: якщо у поточному кадрі після фільтрації результатів виявлено хоча б один об’єкт цільового класу з достатнім рівнем достовірності, система фіксує подію. Якщо ж таких об’єктів не знайдено, реакція не формується. Така схема є доцільною для кваліфікаційної роботи, оскільки не ускладнює систему надмірною кількістю умов, але демонструє повний цикл переходу від вхідного кадру до прикладного результату.

Логіку прийняття рішення щодо формування події наведено в таблиці 2.5.

Таблиця 2.5 – Правила формування події в системі

Умова	Рішення системи
Цільовий об’єкт не виявлено	Подія не формується
Виявлено об’єкт із низькою достовірністю	Результат відхиляється
Виявлено цільовий об’єкт із достатньою достовірністю	Формується подія
Подію сформовано	Дані передаються до збереження та сповіщення

З погляду програмної реалізації запропоновані алгоритми є основою для побудови послідовності викликів між модулями системи. Алгоритм загального функціонування визначає порядок виконання внутрішніх компонентів, а алгоритм взаємодії між рівнями формує основу для розподіленої архітектури. Разом вони створюють завершену логічну модель роботи системи, яка надалі може бути безпосередньо реалізована у програмному коді.

Отже, у цьому підрозділі розроблено алгоритми функціонування програмної системи інтелектуального відеоспостереження та взаємодії між edge-, fog- і cloud-рівнями. Сформовано послідовність оброблення відеопотоку та встановлено правила формування події.

3 РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ПРОГРАМНОЇ СИСТЕМИ

3.1 Вибір засобів і технологій реалізації програмної системи

Для реалізації програмної системи інтелектуального відеоспостереження необхідно обрати такі засоби, які забезпечують роботу з відеопотоком, попереднє оброблення кадрів, реалізацію згорткової нейронної мережі типу encoder–decoder, побудову маски об’єкта та збереження результатів роботи системи. У межах цієї роботи важливими критеріями вибору є доступність технологій, їх поширеність у задачах комп’ютерного зору та придатність до створення дослідницького програмного прототипу.

Основною мовою реалізації обрано Python, оскільки вона широко використовується в задачах машинного навчання, комп’ютерного зору та швидкого створення прототипів. Її перевага полягає у наявності великої кількості бібліотек для роботи із зображеннями, відеопотоками, нейронними мережами та чисельними обчисленнями. Для даної роботи це дозволяє зосередитися на структурі програмної системи та логіці її функціонування.

В якості середовище розроблення використано Visual Studio Code, яке підтримує Python-проєкти, налагодження коду, запуск окремих модулів і зручну організацію структури програмного проєкту. Це є важливим для модульної реалізації системи, у якій окремо виділяються модулі роботи з відеопотоком, сегментації, аналізу подій і збереження результатів.

Для роботи з відеопотоком і кадрами використано бібліотеку OpenCV. Вона забезпечує зчитування відео з камери або файлу, виділення окремих кадрів, зміну розміру зображень, перетворення кольорового простору та візуалізацію результатів. Саме OpenCV є базовим засобом для реалізації вхідного блоку подання кадру та модуля попереднього оброблення даних.

Для реалізації нейромережевого модуля застосовано PyTorch, оскільки цей фреймворк є зручним для побудови та використання сегментаційних згорткових нейронних мереж. У межах цієї роботи він використовується для реалізації моделі типу encoder–decoder, яка виконує сегментацію області дії та формування маски

об’єкта.

Для допоміжних операцій з даними використано NumPy, який дозволяє ефективно працювати з багатовимірними масивами, проміжними поданнями кадрів, масками та ознаковими структурами. Цей інструмент є базовим для внутрішнього перетворення даних між окремими модулями системи.

Для збереження результатів роботи системи достатньо використати файлову систему або SQLite. Це дозволяє записувати кадри, побудовані маски, журнал подій та службові дані без залучення складної серверної інфраструктури. Для даної роботи такого підходу достатньо, оскільки він забезпечує повний цикл фіксації результатів і подальшої перевірки роботи системи.

Для побудови графіків і візуального подання результатів експерименту використано Matplotlib. Цей засіб може бути застосований під час аналізу продуктивності системи, подання часових показників або ілюстрації узагальнених результатів тестування.

Основні засоби і технології реалізації програмної системи наведено в таблиці 3.1.

Таблиця 3.1 – Засоби і технології реалізації програмної системи

Засіб / технологія	Призначення
Python	Основна мова програмної реалізації системи
Visual Studio Code	Середовище розроблення та налагодження
OpenCV	Робота з відеопотоком, кадрами та візуалізацією
PyTorch	Реалізація моделі encoder–decoder для сегментації
NumPy	Оброблення масивів даних і допоміжні обчислення
SQLite / файлова система	Збереження журналу подій, масок, кадрів і результатів
Matplotlib	Побудова графіків і візуалізація результатів експерименту

У межах цієї роботи центральним елементом реалізації є саме сегментаційна модель encoder–decoder, а не модель детекції прямокутними рамками. Такий вибір пояснюється тим, що система повинна не лише виявляти наявність об’єкта, а й формувати його маску та виділяти область дії. Це дозволяє зменшити вплив фону, отримати точніше просторове представлення об’єкта та підвищити інформативність подальшого аналізу.

Для підвищення відтворюваності програмного рішення використано віртуальне середовище Python, у якому фіксуються версії основних бібліотек. Це дає змогу уникнути конфліктів залежностей і забезпечує можливість повторного запуску системи на іншому комп'ютері без істотної зміни програмного коду.

Таким чином, для реалізації програмної системи обрано технологічний стек, який охоплює всі основні задачі: отримання відеопотоку, попереднє оброблення кадрів, сегментацію області дії, формування маски об'єкта, збереження результатів та подання експериментальних даних. Обрані засоби є достатніми для створення працездатного програмного прототипу й узгоджуються з розробленою моделлю оброблення відеопотоку на основі згорткової нейронної мережі encoder–decoder.

3.2 Програмна реалізація основних модулів системи

Після обґрунтування технологічного стеку наступним етапом стала програмна реалізація основних модулів системи інтелектуального відеоспостереження. Реалізацію виконано відповідно до архітектури, сформованої в другому розділі, із поділом на окремі функціональні компоненти. Такий підхід забезпечив зрозумілу структуру програмного коду, спростив взаємодію між частинами системи та створив основу для подальшого тестування і розширення розробленого прототипу.

У межах програмної системи реалізовано такі основні модулі:

– модуль отримання відеоданих;

– модуль попереднього оброблення кадрів;

– модуль сегментації області дії на основі згорткової нейронної мережі encoder–decoder;

– модуль побудови маски та виділення об'єкта;

– модуль аналізу подій;

– модуль збереження результатів;

– модуль сповіщення.

Схему організації структури проєкту програмної системи подано на рисунку 3.1. У структурі окремо виділено каталог вихідних даних, каталог моделей,

програмні модулі системи та каталог результатів, що забезпечує модульність реалізації, зручність тестування і подальше розширення програмного прототипу.

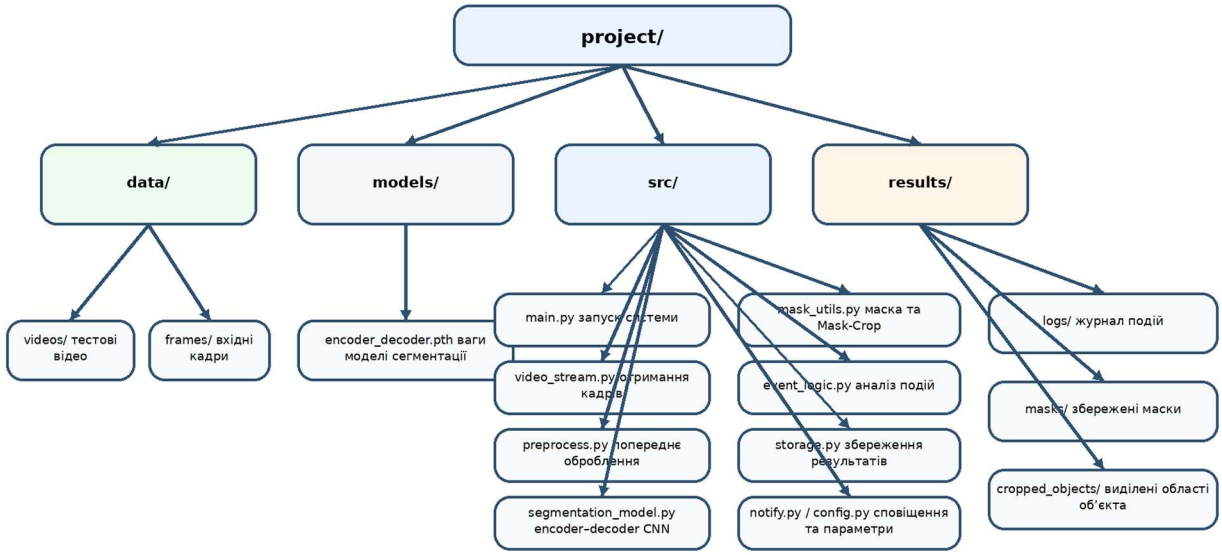

Рисунок 3.1 – Схема організації структури проєкту програмної системи

Першим реалізовано модуль отримання відеоданих. Його завдання полягає у підключенні до джерела відео та зчитуванні поточного потоку кадрів. У межах програмного прототипу цей модуль забезпечує роботу як із камерою, так і з тестовим відеофайлом, що є зручним для експериментальної перевірки системи. Така реалізація дозволила використовувати один і той самий програмний контур як для моделювання потоку даних, так і для аналізу заздалегідь підготовлених відеосценаріїв.

Фрагмент програмної реалізації модуля отримання відеоданих:

```
import cv2

class VideoStream:
    def __init__(self, source=0):
        self.capture = cv2.VideoCapture(source)

    def read_frame(self):
        success, frame = self.capture.read()
        if not success:
            return None
        return frame

    def release(self):
        self.capture.release()
```

Наведений фрагмент ілюструє базову логіку підключення до джерела відео, зчитування кадрів і завершення роботи з ресурсом.

Приклад запуску програмної системи в термінальному режимі подано на рисунку 3.2. Під час запуску система ініціалізує джерело відеоданих, завантажує модель encoder–decoder, встановлює параметри оброблення кадрів і створює каталоги для збереження результатів.

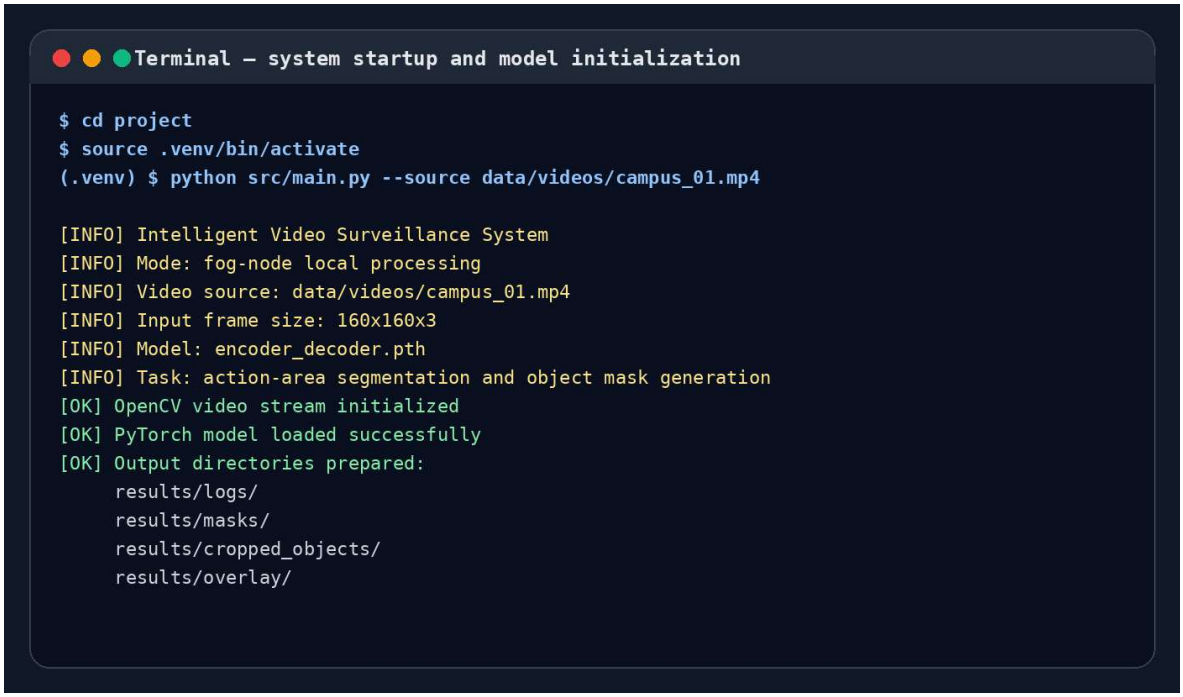

Рисунок 3.2– Запуск програмної системи та ініціалізація моделі encoder–decoder

Наступним компонентом є модуль попереднього оброблення кадрів. Він забезпечує приведення зображення до формату, придатного для подання в нейромережеву модель. У межах цього модуля виконуються зміна розміру кадру, нормалізація та підготовка вхідних даних до сегментації. Відокремлення цього модуля в самостійний блок дозволило зробити програмну структуру більш логічною та спростити подальшу модифікацію параметрів оброблення.

Приклад реалізації попереднього оброблення:

```
import cv2
import numpy as np

def preprocess_frame(frame, size=(160, 160)):
    resized = cv2.resize(frame, size)
    normalized = resized.astype("float32") / 255.0
    return normalized
```

Центральним компонентом системи є модуль сегментації області дії на

основі згорткової нейронної мережі encoder–decoder. Саме він реалізує основну інтелектуальну функцію системи. На вхід цього модуля подається підготовлений кадр, а на виході формується маска об’єкта або області дії. На відміну від детекційного підходу, результатом є не прямокутна область, а піксельне виділення цільового об’єкта, що є більш інформативним для подальшого аналізу.

Узагальнений фрагмент програмної реалізації модуля сегментації має такий вигляд:

```
import torch
import torch.nn as nn

class EncoderDecoderNet(nn.Module):
    def __init__(self):
        super().__init__()
        # Узагальнена структура моделі
        self.encoder = nn.Sequential(
            nn.Conv2d(3, 32, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(2)
        )
        self.bottleneck = nn.Sequential(
            nn.Conv2d(32, 64, kernel_size=3, padding=1),
            nn.ReLU()
        )
        self.decoder = nn.Sequential(
            nn.ConvTranspose2d(64, 32, kernel_size=2, stride=2),
            nn.ReLU(),
            nn.Conv2d(32, 1, kernel_size=1)
        )

    def forward(self, x):
        x = self.encoder(x)
        x = self.bottleneck(x)
        x = self.decoder(x)
        return x
```

Наведений приклад не відтворює повну архітектуру моделі з усіма блоками, але показує загальний принцип її програмної організації: енкодер, bottleneck і декодер формують єдиний ланцюг сегментації.

Після роботи нейромережевого модуля результат надходить до модуля побудови маски та виділення об’єкта. Його призначення полягає у перетворенні вихідної карти ознак у бінарну маску та виділенні області об’єкта за принципом Mask-Crop. Унаслідок цього з повного кадру формується очищений фрагмент, у якому збережено лише цільовий об’єкт або область дії.

Спрощений приклад такого модуля:

```
import numpy as np

def build_mask(prediction, threshold=0.5):
```

```
    mask = (prediction > threshold).astype(np.uint8)
    return mask

def apply_mask(frame, mask):
    return frame * mask[:, :, np.newaxis]
```

Такий підхід забезпечує перехід від нейромережевого виходу до прикладного результату, придатного для подальшого аналізу події.

Наступним є модуль аналізу подій. Його функція полягає у прийнятті рішення про наявність події на основі побудованої маски або виділеної області об’єкта. У межах прототипу використано спрощену логіку: якщо маска містить достатню кількість пікселів, що належать цільовому об’єкту, система фіксує подію. Це дозволяє перевести результат сегментації в прикладний висновок.

Приклад такого фрагмента:

```
def check_event(mask, min_pixels=500):
    object_area = mask.sum()
    return object_area >= min_pixels
```

Після визначення події результат передається до модуля збереження результатів. У межах програмного прототипу цей модуль відповідає за запис інформації про подію, збереження масок, окремих кадрів або виділених фрагментів та формування журналу результатів. Технічно це може бути реалізовано через файлову систему або SQLite, що є достатнім для навчального дослідження.

Приклад запису події:

```
from datetime import datetime

def save_event(log_path, event_text):
    with open(log_path, "a", encoding="utf-8") as file:
        timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
        file.write(f"{timestamp} - {event_text}\n")
```

Окремо реалізовано модуль сповіщення, який формує реакцію системи на зафіксовану подію. У межах прототипу така реакція може бути подана як текстове повідомлення, запис у консоль або позначення на кадрі. Його призначення полягає в завершенні повного циклу роботи системи: від надходження відеопотоку до формування прикладної реакції.

Важливою особливістю програмної реалізації є те, що всі модулі поєднані в єдиний послідовний цикл оброблення. Кожен модуль отримує чітко визначений вхід, виконує завершену функцію та передає результат до наступного етапу. Це

робить систему зрозумілою, придатною до налагодження та зручною для подальшого розширення.

Приклад оброблення відеопотоку та формування події в термінальному режимі подано на рисунку 3.3. На скріншоті відображено послідовну обробку кадрів, площу сформованої маски, час оброблення кадру, факт виникнення події та збереження маски, виділеної області об’єкта й накладення результату на кадр.

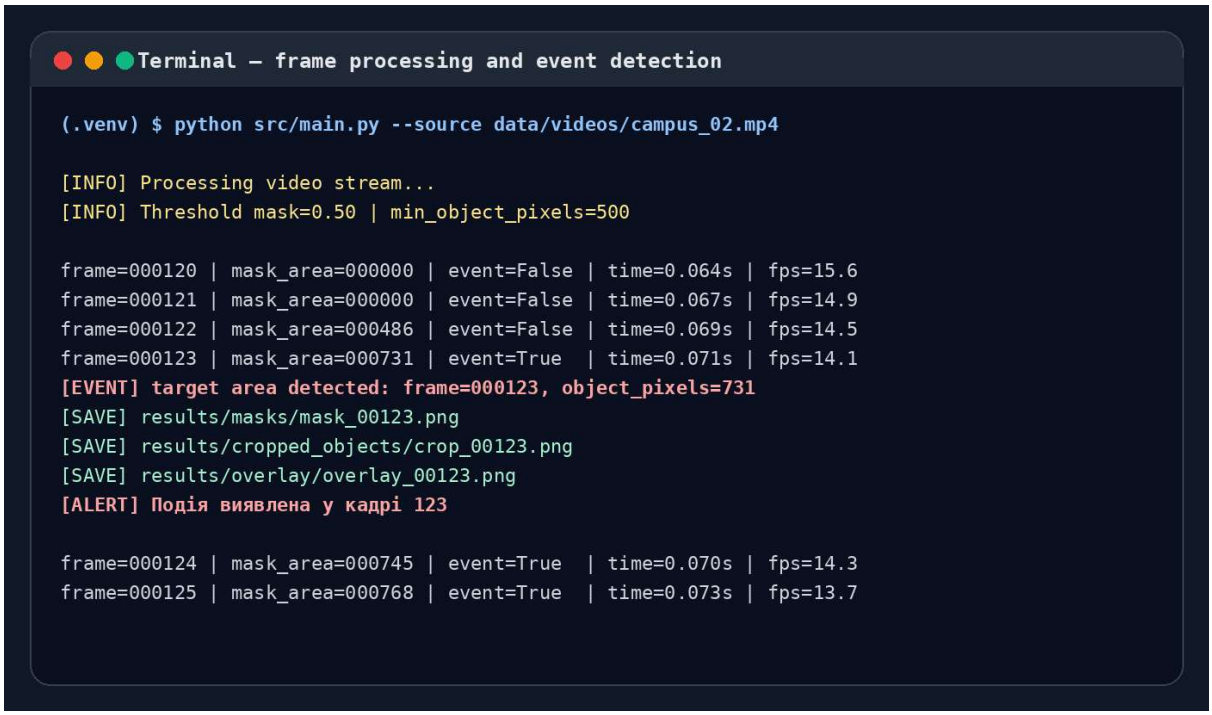

Рисунок 3.3 – Оброблення відеопотоку та формування події

Повні лістинги програмних модулів подано в додатку А. Модуль `video_stream.py` забезпечує зчитування відеопотоку, `preprocess.py` виконує підготовку кадру, `segmentation_model.py` реалізує модель encoder–decoder для побудови маски, `mask_utils.py` формує маску та виконує Mask-Crop, `event_logic.py` визначає наявність події, а `storage.py` і `notify.py` відповідають за збереження результатів та інформування про подію. Центральний сценарій роботи системи реалізовано у файлі `main.py`.

Отже, у цьому підрозділі розглянуто програмну реалізацію основних модулів системи інтелектуального відеоспостереження. Реалізовано модулі роботи з відеопотоком, попереднього оброблення кадрів, сегментації області дії, побудови маски об’єкта, аналізу подій, збереження результатів і сповіщення.

3.3 Організація експериментів та тестування програмної системи

Після реалізації основних модулів програмної системи було організовано експериментальне дослідження її роботи. Метою цього етапу стала перевірка працездатності програмного прототипу в умовах оброблення реального відеопотоку, оцінювання якості сегментації області дії, аналіз швидкодії системи та визначення її придатності до функціонування в режимі, наближеному до реального часу.

Експерименти проводилися на наборі коротких відеофрагментів, у яких присутня людина в кадрі за різних умов спостереження. Для перевірки роботи системи було сформовано тестову вибірку з 12 відеофрагментів тривалістю від 20 до 40 с, отриманих із вебкамери та відкритих демонстраційних відео. Загальний обсяг тестових даних становив 4320 кадрів, із яких 2780 кадрів містили цільовий об’єкт у повному або частковому вигляді, а 1540 кадрів відповідали сценам без цільового об’єкта або з невиразною появою людини на задньому плані.

Для забезпечення більш об’єктивного аналізу відеофрагменти було поділено на три групи (таблиця 3.2):

– сцени з однорідним фоном і стабільним освітленням;

– сцени зі складним фоном;

– сцени з частковим перекриттям об’єкта або зміною освітлення.

Такий поділ дав змогу оцінити роботу системи не лише в базових умовах, а й у більш наближених до реального використання сценаріях.

Таблиця 3.2 – Характеристика тестової вибірки для експерименту

Група відеоданих	Кількість фрагментів	Кількість кадрів	Особливості сцени
Прості сцени	4	1360	Однорідний фон, стабільне освітлення
Складні сцени	4	1480	Наявність зайвих об’єктів і складного фону
Сцени з частковим перекриттям	4	1480	Часткове закриття об’єкта, змінне освітлення
Разом	12	4320	–

Під час експерименту перевірялася повна послідовність роботи системи: зчитування відеопотоку, попереднє оброблення кадрів, сегментація області дії за допомогою моделі encoder–decoder, побудова маски об’єкта, виділення області через Mask-Crop, формування події та збереження результатів. Для кожного тестового фрагмента фіксувалися кількість оброблених кадрів, кількість кадрів із коректною сегментацією, кількість сформованих подій, а також середній час оброблення одного кадру.

Оскільки розроблена система орієнтована саме на сегментацію, основними показниками якості було обрано:

– Pixel Accuracy – частка правильно класифікованих пікселів;

– IoU (Intersection over Union) – міра перекриття між еталонною та передбаченою масками;

– F1-score для маски об’єкта;

– середній час оброблення одного кадру;

– кількість сформованих подій.

Для оцінювання сегментації на частині тестової вибірки було сформовано набір еталонних масок для 600 кадрів, рівномірно розподілених між трьома групами сцен. Саме на цьому піднаборі виконувалося обчислення основних метрик якості сегментації. Узагальнені результати наведено в таблиці 3.3.

Таблиця 3.3 – Основні результати оцінювання якості сегментації

Група сцен	Pixel Accuracy	IoU	F1-score
Прості сцени	0,962	0,884	0,921
Складні сцени	0,934	0,812	0,876
Часткове перекриття	0,918	0,768	0,842
Середнє значення	0,938	0,821	0,880

Отримані результати свідчать, що найвища якість сегментації досягалася в простих сценах, де фон був однорідним, а контури об’єкта – чітко вираженими. У складних сценах значення IoU та F1-score знижувалися, що пояснюється появою додаткових об’єктів у кадрі та більшою схожістю кольорових і текстурних характеристик фону й цільового об’єкта. Найскладнішими виявилися сцени з

частковим перекриттям, де сегментаційна модель зберігала працездатність, але демонструвала нижчу точність формування маски.

Результати оцінювання якості сегментації для різних груп сцен у термінальному режимі подано на рисунку 3.4. Система виводить кількість тестових відеофрагментів, загальну кількість кадрів, кількість анотованих масок і значення метрик Pixel Accuracy, IoU та F1-score.

Рисунок 3.4 – Результати оцінювання якості сегментації в термінальному режимі

Для оцінювання швидкодії системи було проведено окремі вимірювання часу оброблення кадрів. Експеримент виконувався на персональному комп’ютері з процесором середнього класу та графічним прискоренням споживчого рівня. У середньому час оброблення одного кадру становив 0,071 с, що відповідає приблизно 14 кадрам за секунду. Для окремих груп сцен ці значення незначно відрізнялися через складність побудови маски.

Результати наведено в таблиці 3.4.

Таблиця 3.4 – Показники швидкодії програмної системи

Група сцен	Середній час оброблення кадру, с	Орієнтовна швидкодія, кадр/с
Прості сцени	0,066	15,2
Складні сцени	0,072	13,9
Часткове перекриття	0,075	13,3
Середнє значення	0,071	14,1

Для наочного подання якості сегментації використано графік порівняння метрик для трьох груп сцен (рисунки 3.5 та 3.6).

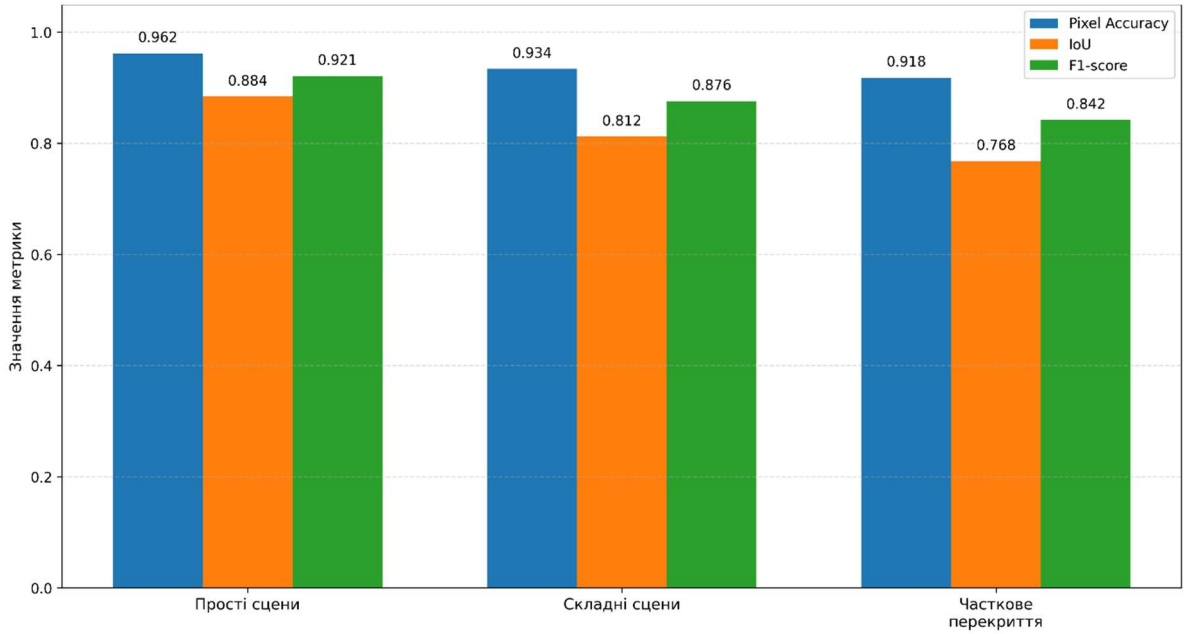

Рисунок 3.5 – Порівняння метрик Pixel Accuracy, IoU та F1-score для різних груп сцен

У межах тестування також перевірялася правильність формування події на основі побудованої маски. Для цього в системі використовувалося просте правило: якщо площа сегментованої області перевищувала заданий поріг, кадр вважався таким, що містить цільовий об’єкт, і формувалася подія. За результатами оброблення 4320 кадрів було сформовано 2546 подій, із яких 2361 відповідали коректному виявленню об’єкта, а 185 були віднесені до хибних спрацьовувань. Таким чином, точність формування події за результатами експерименту становила близько 92,7 %.

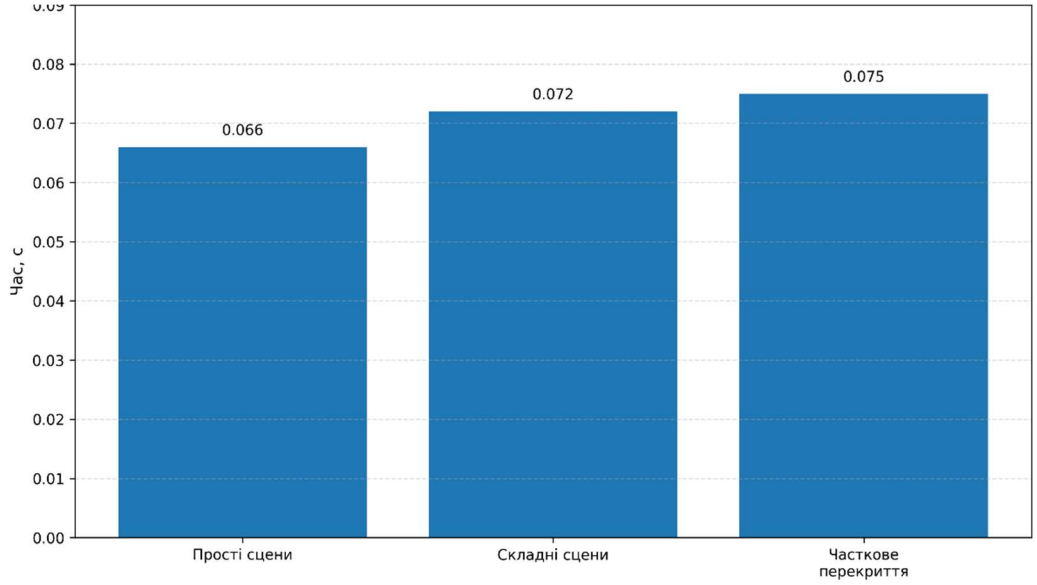

Рисунок 3.6 – Середній час оброблення кадру для різних груп сцен

Під час тестування було встановлено, що система коректно проходить повний цикл оброблення даних: зчитує кадри, будує маску, виконує Mask-Crop, формує подію та зберігає результат у журналі. Також підтверджено, що після формування події система не припиняє роботу, а переходить до оброблення наступних кадрів, що є важливим для режиму безперервного відеоспостереження.

Отже, організація експерименту дозволила перевірити систему на реальних відеоданих, оцінити якість сегментації, швидкодію та коректність логіки формування події.

3.4 Аналіз результатів роботи системи

Результати експериментального дослідження показали, що розроблена програмна система інтелектуального відеоспостереження є працездатною та забезпечує виконання повного циклу оброблення відеопотоку: від отримання кадру до побудови маски об’єкта, формування події та збереження результатів. Аналіз отриманих показників дозволяє оцінити систему не лише як програмний прототип, а як завершене прикладне рішення для задач сегментації області дії в умовах безперервного спостереження.

Підсумковий звіт експериментального дослідження в термінальному режимі

подано на рисунку 3.7.

Рисунок 3.7 – Підсумковий звіт експериментального дослідження в термінальному режимі

Насамперед було підтверджено, що використання моделі encoder–decoder є обґрунтованим для даної задачі. Середнє значення Pixel Accuracy = 0,938, IoU = 0,821 та F1-score = 0,880 свідчать про достатньо високу якість сегментації для програмного прототипу. Це означає, що модель не лише розпізнає наявність цільового об’єкта, а й формує просторово точну маску, придатну для подальшого аналізу.

Особливо важливо, що найкращі результати отримано в простих сценах: IoU = 0,884 та F1-score = 0,921. Це підтверджує, що архітектура моделі добре працює в умовах стабільного освітлення та відносно чистого фону. Для систем відеоспостереження це означає високу ефективність у контрольованих приміщеннях, коридорах, навчальних аудиторіях або офісних зонах, де сцена має обмежену кількість завад.

Водночас експеримент показав, що зі зростанням складності сцени якість сегментації знижується. Для сцен зі складним фоном середній IoU зменшився до

0,812, а для сцен із частковим перекриттям – до 0,768. Така поведінка є очікуваною, оскільки в умовах накладання об’єктів, змінного освітлення або неоднорідного фону межі цільового об’єкта стають менш виразними. Проте навіть у таких умовах система зберігала працездатність і формувала коректні маски в більшості випадків.

Узагальнені результати експерименту подано в таблиці 3.5.

Таблиця 3.5 – Узагальнені результати експериментального дослідження

Показник	Значення
Загальна кількість оброблених кадрів	4320
Кількість кадрів для оцінювання сегментації	600
Середній Pixel Accuracy	0,938
Середній IoU	0,821
Середній F1-score	0,880
Середній час оброблення кадру	0,071 с
Середня швидкодія	14,1 кадр/с
Кількість сформованих подій	2546
Кількість коректних подій	2361
Точність формування події	92,7 %

Важливим результатом стало підтвердження того, що система може працювати в режимі, наближеному до реального часу. Середній час оброблення одного кадру 0,071 с означає, що програмний прототип здатний обробляти близько 14 кадрів за секунду. Ці дані підтверджують практичну реалізованість системи без використання високопродуктивної спеціалізованої інфраструктури.

Ще однією важливою перевагою є те, що сегментаційний підхід виявився інформативнішим порівняно з базовою детекцією прямокутною рамкою. Побудована маска дозволяє точніше виділити об’єкт у кадрі, а блок Mask-Crop – зменшити вплив фону на подальший аналіз. Це особливо важливо у сценах, де присутні зайві елементи, складні текстури або часткові перекриття. Таким чином, система забезпечує не лише виявлення факту наявності об’єкта, а і формування його точнішого просторового представлення.

Переваги розробленої системи доцільно узагальнити в таблиці 3.6.

Таблиця 3.6 – Основні переваги розробленої програмної системи

Перевага	Характеристика
Піксельна сегментація	Формується маска об’єкта, а не лише рамка
Виділення області дії	Використовується блок Mask-Crop для очищення фону
Достатня точність	Середній IoU 0,821 і F1-score 0,880
Працездатність у потоці	Середня швидкодія 14,1 кадр/с
Модульність реалізації	Система побудована з окремих функціональних модулів
Узгодженість із fog-підходом	Основна обробка виконується локально

Разом із тим під час аналізу результатів було виявлено і певні обмеження системи. По-перше, якість сегментації знижується в сценах із частковим перекриттям і складним фоном. По-друге, швидкодія системи є достатньою для дослідницького прототипу, однак для високочастотного відеопотоку промислового рівня вона потребуватиме подальшої оптимізації. По-третє, у межах роботи використано спрощене правило формування події на основі площі сегментованої області, тоді як у більш складних системах можуть враховуватися траєкторія руху, динаміка маски та інші часові характеристики.

Основні обмеження системи наведено в таблиці 3.7.

Таблиця 3.7 – Основні обмеження розробленої програмної системи

Обмеження	Зміст
Чутливість до складного фону	Зниження IoU та F1-score у складних сценах
Часткові перекриття	Погіршення точності меж маски
Середня швидкодія	Потрібна додаткова оптимізація для вищої частоти кадрів
Спрощене правило формування події	Подія визначається за площею маски без часової аналітики
Дослідницький характер реалізації	Не виконано повноцінного промислового розгортання

Для наочного подання результатів використано підсумковий графік (рисунок 3.8).

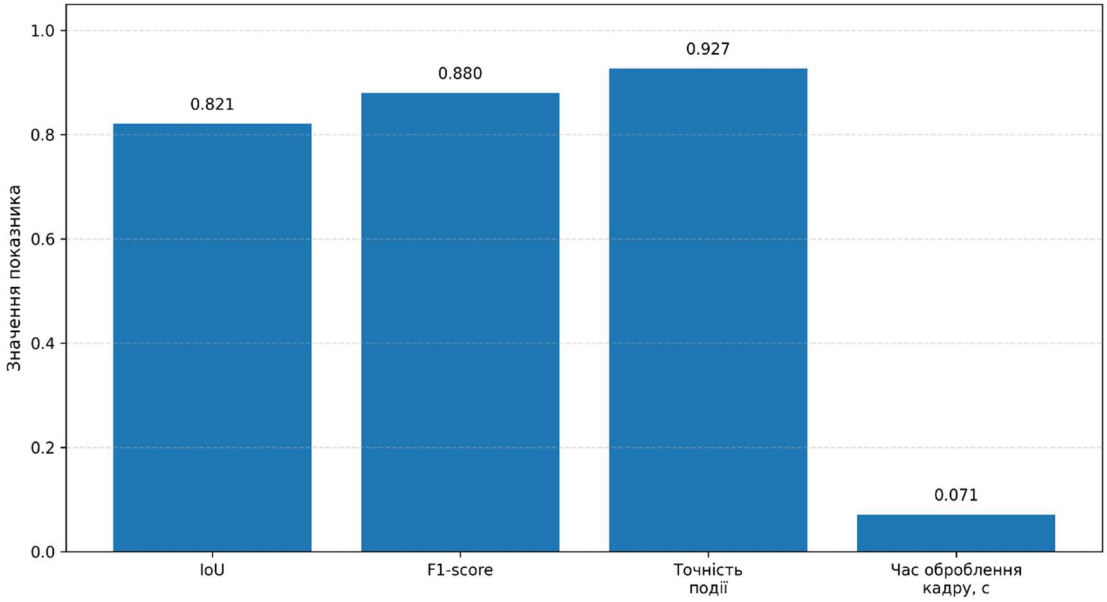

Рисунок 3.8 – Узагальнення результатів експериментального дослідження програмної системи

Незважаючи на зазначені обмеження, результати експерименту підтвердили досягнення мети роботи. Розроблена система забезпечує оброблення відеопотоку, сегментацію області дії, формування маски об’єкта, виділення корисної області через Mask-Crop, формування події та збереження результатів. Тобто створений програмний прототип є не окремим нейромережевим модулем, а цілісною системою інтелектуального відеоспостереження.

Таким чином, аналіз результатів показав, що розроблена програмна система є працездатною, структурно завершеною та придатною до подальшого розвитку. Її основними перевагами є сегментаційний підхід до виділення об’єкта, модульна побудова, достатня точність та узгодженість із концепцією туманних обчислень. Обмеження системи пов’язані переважно зі складністю сцен та дослідницьким характером реалізації, що відкриває перспективи для подальшого вдосконалення моделі й програмної архітектури.

ВИСНОВКИ

У кваліфікаційній роботі розв’язана актуальна задача розроблення програмної системи інтелектуального відеоспостереження в режимі реального часу на основі згорткових нейронних мереж і туманних обчислень.

У процесі виконання роботи отримано наступні результати:

1. Проаналізовано предметну область інтелектуального відеоспостереження, сучасні підходи до оброблення відеопотоку, використання згорткових нейронних мереж у задачах комп’ютерного зору та особливості застосування туманних обчислень у системах реального часу. Проведений аналіз показав, що традиційні системи відеоспостереження не забезпечують достатнього рівня автоматизації, а сучасні інтелектуальні рішення потребують поєднання точної моделі аналізу з архітектурою, придатною до оперативного функціонування.

2. Сформовано вимоги до програмної системи та розроблено її загальну архітектуру. Визначено склад основних функціональних модулів, зокрема модуль отримання відеоданих, модуль попереднього оброблення кадрів, модуль сегментації області дії, модуль побудови маски та Mask-Crop, модуль аналізу подій, модуль збереження результатів і модуль сповіщення. Запропонована архітектура узгоджена з концепцією туманних обчислень, у межах якої оперативні обчислення виконуються ближче до джерела даних, а довготривале збереження та аналітика можуть бути винесені на окремий рівень.

3. Розроблено модель оброблення відеопотоку на основі згорткової нейронної мережі типу encoder–decoder. Її структура охоплює вхідний блок подання кадру, енкодерну частину, центральний вузол bottleneck, декодерну частину, вихідний блок побудови маски та блок Mask-Crop для виділення об’єкта за побудованою маскою. На відміну від спрощеного детекційного підходу, така модель забезпечує піксельне виділення об’єкта, що підвищує інформативність подальшого аналізу та зменшує вплив фону на формування прикладного результату.

4. Розроблено алгоритми функціонування програмної системи та взаємодії між edge-, fog- і cloud-рівнями. Описано загальний цикл оброблення кадру, умови

формування події та логіку переходу до наступного кадру. Запропоновані алгоритми забезпечили формальну основу для програмної реалізації системи та підтвердили узгодженість між архітектурою, моделлю та функціональною логікою роботи.

5. Виконано програмну реалізацію основних модулів системи із використанням Python, OpenCV, PyTorch, NumPy, файлової системи або SQLite та Matplotlib. Реалізовано модульну структуру проєкту, що дозволило розділити оброблення відеоданих, сегментацію, побудову маски, аналіз подій і збереження результатів на окремі логічні компоненти. Підготовлено повні лістинги програмного забезпечення, що відображають цілісну реалізацію системи інтелектуального відеоспостереження.

6. У процесі експериментального дослідження перевірено роботу системи на тестовій вибірці з 12 відеофрагментів загальним обсягом 4320 кадрів. Для оцінювання якості сегментації використано метрики Pixel Accuracy, IoU та F1-score. За результатами експерименту отримано середні значення Pixel Accuracy = 0,938, IoU = 0,821 та F1-score = 0,880, що підтверджує достатню якість побудови маски для програмного прототипу. Середній час оброблення одного кадру становив 0,071 с, що відповідає приблизно 14,1 кадр/с і підтверджує можливість функціонування системи в режимі, наближеному до реального часу.

7. Під час аналізу результатів встановлено, що система найкраще працює в простих сценах зі стабільним фоном і освітленням, тоді як у сценах зі складним фоном та частковим перекриттям точність сегментації знижується. Разом із тим навіть у складніших умовах система зберегла працездатність, коректно формувала маску, виконувала виділення області дії та формувала подію за заданою логікою. Це підтвердило досягнення поставленої мети та працездатність розробленого програмного рішення.

8. Перспективи подальшого розвитку роботи пов’язані з розширенням набору тестових даних, удосконаленням сегментаційної моделі, підвищенням швидкодії, використанням часової інформації з послідовності кадрів, а також реалізацією складніших сценаріїв аналізу, зокрема трекінгу об’єктів, поведінкової аналітики та повноцінного розгортання окремих edge-, fog- і cloud-вузлів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Abbas N., Zhang Y., Taherkordi A., Skeie T. Mobile Edge Computing: A Survey. *IEEE Internet of Things Journal*. 2018. Vol. 5, no. 1. P. 450–465.

2. Bewley A., Ge Z., Ott L., Ramos F., Upcroft B. Simple Online and Realtime Tracking. *2016 IEEE International Conference on Image Processing (ICIP)*. 2016. P. 3464–3468.

3. Chen L.-C., Zhu Y., Papandreou G., Schroff F., Adam H. Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018. P. 801–818.

4. Chen N., Chen Y., You Y., Ling H., Liang P., Zimmermann R. Dynamic Urban Surveillance Video Stream Processing Using Fog Computing. *Proceedings of the IEEE Second International Conference on Multimedia Big Data (BigMM)*. 2016. P. 105–112.

5. Chen X., Xu J.-B., Guo W.-Q. The Research About Video Surveillance Platform Based on Cloud Computing. *Proceedings of the International Conference on Machine Learning and Cybernetics*. 2013. P. 979–983.

6. Chiang M., Ha S., Risso F., Zhang T., Chih-Lin I. Clarifying Fog Computing and Networking: 10 Questions and Answers. *IEEE Communications Magazine*. 2017. Vol. 55, no. 4. P. 18–20.

7. Da Xu L., He W., Li S. Internet of Things in Industries: A Survey. *IEEE Transactions on Industrial Informatics*. 2014. Vol. 10, no. 4. P. 2233–2243.

8. De Donno M., Tange K., Dragoni N. Foundations and Evolution of Modern Computing Paradigms: Cloud, IoT, Edge, and Fog. *IEEE Access*. 2019. Vol. 7. P. 150936–150948.

9. Golovko V., Kroshchanka A., Mikhno E., Komar M., Sachenko A. Deep Convolutional Neural Network for Detection of Solar Panels. In: *Data-Centric Business and Applications*. *Lecture Notes on Data Engineering and Communications Technologies*. 2021. Vol. 48. P. 371–389.

10. He K., Zhang X., Ren S., Sun J. Deep Residual Learning for Image Recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016. P. 770–778.

11. Howard A. G., Zhu M., Chen B., Kalenichenko D., Wang W., Weyand T., Andreetto M., Adam H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv preprint*. 2017. arXiv:1704.04861.

12. Jian Y., Xin W., Xue Z., ZhenYou D. Cloud Computing and Visual Attention Based Object Detection for Power Substation Surveillance Robots. *2015 IEEE 28th Canadian Conference on Electrical and Computer Engineering (CCECE)*. 2015. P. 337–342.

13. Komar M., Savchyshyn R., Lipianina-Honcharenko K., Osolinskyi O. Intelligent Method for Counting Cars from Satellite Images. *CEUR Workshop Proceedings*. 2023. Vol. 3538. P. 295–303.

14. Krizhevsky A., Sutskever I., Hinton G. E. ImageNet Classification with Deep Convolutional Neural Networks. *Communications of the ACM*. 2017. Vol. 60, no. 6. P. 84–90.

15. Li C., Xue Y., Wang J., Zhang W., Li T. Edge-Oriented Computing Paradigms: A Survey on Architecture Design and System Management. *ACM Computing Surveys*. 2018. Vol. 51, no. 2. Art. 39.

16. Lin J., Yu W., Zhang N., Yang X., Zhang H., Zhao W. A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications. *IEEE Internet of Things Journal*. 2017. Vol. 4, no. 5. P. 1125–1142.

17. Liu W., Anguelov D., Erhan D., Szegedy C., Reed S., Fu C.-Y., Berg A. C. SSD: Single Shot MultiBox Detector. *Proceedings of the European Conference on Computer Vision (ECCV)*. 2016. P. 21–37.

18. Long J., Shelhamer E., Darrell T. Fully Convolutional Networks for Semantic Segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015. P. 3431–3440.

19. Nasir M., Muhammad K., Lloret J., Sangaiah A. K., Sajjad M. Fog Computing Enabled Cost-Effective Distributed Summarization of Surveillance Videos for Smart Cities. *Journal of Parallel and Distributed Computing*. 2019. Vol. 126. P. 161–170.

20. Neto A. J., Zhao Z., Rodrigues J. J., Camboim H. B., Braun T. Fog-Based Crime-Assistance in Smart IoT Transportation System. *IEEE Access*. 2018. Vol. 6. P. 11101–11111.

21. NVIDIA. *Jetson Nano Developer Kit*.

22. Paul A. K., Park J. S. Multiclass Object Recognition Using Smartphone and Cloud Computing for Augmented Reality and Video Surveillance Applications. *Proceedings of the 2013 International Conference on Informatics*. 2013. P. 1–6.

23. Redmon J., Divvala S., Girshick R., Farhadi A. You Only Look Once: Unified, Real-Time Object Detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016. P. 779–788.

24. Ren S., He K., Girshick R., Sun J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2017. Vol. 39, no. 6. P. 1137–1149.

25. Ronneberger O., Fischer P., Brox T. U-Net: Convolutional Networks for Biomedical Image Segmentation. *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention*. 2015. P. 234–241.

26. Schroff F., Kalenichenko D., Philbin J. FaceNet: A Unified Embedding for Face Recognition and Clustering. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015. P. 815–823.

27. Simonyan K., Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *3rd International Conference on Learning Representations (ICLR)*. 2015.

28. Song S., Lan C., Xing J., Zeng W., Liu J. An End-to-End Spatio-Temporal Attention Model for Human Action Recognition from Skeleton Data. *Proceedings of the AAAI Conference on Artificial Intelligence*. 2017. Vol. 31, no. 1.

29. Sun K., Xiao B., Liu D., Wang J. Deep High-Resolution Representation Learning for Human Pose Estimation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019. P. 5693–5703.

30. Tan M., Pang R., Le Q. V. EfficientDet: Scalable and Efficient Object Detection. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020. P. 10781–10790.

31. Tkach I., Petrov Y., Komar M., Dmitro D. Edge-Cloud Information-Quanta Ontology for Vision-Language Surveillance: Data-Centric Tokenization for Low-

Latency IoT Deployment. *2025 IEEE 16th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)*. 2025. P. 565–573.

32. Tuli S., Basumatary N., Gill S. S., Kahani M., Arya R. C., Wander G. S., Buyya R. HealthFog: An Ensemble Deep Learning-Based Smart Healthcare System for Automatic Diagnosis of Heart Diseases in Integrated IoT and Fog Computing Environments. *Future Generation Computer Systems*. 2020. Vol. 104. P. 187–200.

33. Wen Y., Yang X., Xu Y. Cloud-Computing-Based Framework for Multi-Camera Topology Inference in Smart City Sensing System. *Proceedings of the 2010 ACM Multimedia Workshop on Mobile Cloud Media Computing*. 2010. P. 65–70.

34. Wojke N., Bewley A., Paulus D. Simple Online and Realtime Tracking with a Deep Association Metric. *2017 IEEE International Conference on Image Processing (ICIP)*. 2017. P. 3645–3649.

35. Yan S., Xiong Y., Lin D. Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. *Proceedings of the AAAI Conference on Artificial Intelligence*. 2018. Vol. 32, no. 1.

36. Zhang F., Zhu X., Dai H., Ye M., Zhu C. Distribution-Aware Coordinate Representation for Human Pose Estimation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020. P. 7093–7102.

37. Zhang J., Tao D. Empowering Things with Intelligence: A Survey of the Progress, Challenges, and Opportunities in Artificial Intelligence of Things. *IEEE Internet of Things Journal*. 2021. Vol. 8, no. 10. P. 7789–7817.

38. Zhao H., Shi J., Qi X., Wang X., Jia J. Pyramid Scene Parsing Network. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017. P. 2881–2890.

39. Безобразов С., Шелех А., Кислюк С. та ін. Artificial Intelligence for Sport Activity Recognition. *Proceedings of the 2019 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*. 2019. Vol. 2. P. 628–632.

40. Зінченко О. В., Звенігородський О. С., Кисіль Т. М. Згорткові нейронні мережі для вирішення задач комп'ютерного зору. *Телекомунікаційні та інформаційні технології*. 2022. № 2(75). С. 4–12.

41. Кунак І. С., Шпінарева І. М., Пенко В. Г. Ідентифікація особи у відеопотоці методами машинного навчання. *Інформатика та математичні методи в моделюванні*. 2021. Т. 11, № 4. С. 287–295.

42. Мельникова Н. І., Поберейко П. Б. Покращення можливостей пошуку відео: інтеграція нейронної мережі прямого поширення для ефективного фрагментного пошуку. *Computer Design Systems. Theory and Practice*. 2024. Vol. 6, no. 1. P. 149–160.

43. Сватюк Д., Сватюк О., Белей О. Застосування згорткових нейронних мереж для безпеки розпізнавання об'єктів у відеопотоці. *Кібербезпека: освіта, наука, техніка*. 2020. № 4(8). С. 97–112.

44. Омельчук О.А. Інтелектуальне відеоспостереження в режимі реального часу на основі згорткових нейронних мереж та туманних обчислень. Інтелектуальні комп'ютерні системи та мережі» (ІКСМ) : матеріали IV Всеукраїнської науково-практичної конференції молодих вчених і студентів (м. Тернопіль, ЗУНУ, 20 травня 2026 р.). Тернопіль, 2026. С. 137–140.

45. Островерхов В. М., Біловус Л. І., Возьний К. З., Луцишин О. О., Монастирський Г. Л., Надвиничний С. А., Питель С. В., Шандрук С. К. Загальні методичні рекомендації з підготовки, оформлення, захисту та оцінювання кваліфікаційних робіт здобувачів вищої освіти першого (бакалаврського) і другого (магістерського) рівнів. Тернопіль: ЗУНУ, 2024. 83 с.

46. Комар М.П., Саченко А.О., Васильків Н.М., Гладій Г.М., Коваль В.С., Ліп'яніна-Гончаренко Х.В. Методичні рекомендації до виконання кваліфікаційної роботи з освітньо-професійної програми «Комп'ютерні науки» спеціальності 122 «Комп'ютерні науки» за першим (бакалаврським) рівнем вищої освіти. Тернопіль: ЗУНУ, 2024. 52 с.

Додаток А

Лістинги основних програмних модулів

Лістинг А.1 – Файл config.py

```
from dataclasses import dataclass
from pathlib import Path


@dataclass
class AppConfig:
    video_source: int | str = 0
    input_width: int = 160
    input_height: int = 160
    mask_threshold: float = 0.5
    min_object_pixels: int = 500
    output_dir: str = "results"
    save_masks: bool = True
    save_crops: bool = True
    save_overlay: bool = True
    log_filename: str = "events.log"
    model_path: str = "models/encoder_decoder.pth"
    use_camera: bool = True
    display_window: bool = True
    max_frames: int | None = None


def ensure_output_structure(config: AppConfig) -> dict[str, Path]:
    root = Path(config.output_dir)
    logs_dir = root / "logs"
    masks_dir = root / "masks"
    crops_dir = root / "cropped_objects"
    overlay_dir = root / "overlay"

    for directory in [root, logs_dir, masks_dir, crops_dir, overlay_dir]:
        directory.mkdir(parents=True, exist_ok=True)

    return {
        "root": root,
        "logs": logs_dir,
        "masks": masks_dir,
        "crops": crops_dir,
        "overlay": overlay_dir,
    }
```

Лістинг А.2 – Файл video_stream.py

```
import cv2


class VideoStream:
    def __init__(self, source=0) -> None:
        self.source = source
        self.capture = cv2.VideoCapture(source)

        if not self.capture.isOpened():
            raise RuntimeError(f"Не вдалося відкрити джерело відео: {source}")

    def read_frame(self):
        success, frame = self.capture.read()
        if not success:
```

```
            return None
        return frame

    def release(self) -> None:
        if self.capture is not None:
            self.capture.release()
```

Лістинг А.3 – Файл preprocess.py

```
from __future__ import annotations

import cv2
import numpy as np
import torch


def preprocess_frame(frame: np.ndarray, size: tuple[int, int] = (160, 160)) ->
tuple[np.ndarray, torch.Tensor]:
    """
    Повертає:
    1. resized_frame - кадр після зміни розміру для подальших візуалізацій;
    2. tensor - тензор форми [1, 3, H, W] для моделі.
    """
    resized_frame = cv2.resize(frame, size)
    rgb = cv2.cvtColor(resized_frame, cv2.COLOR_BGR2RGB)
    normalized = rgb.astype("float32") / 255.0
    chw = np.transpose(normalized, (2, 0, 1))
    tensor = torch.from_numpy(chw).unsqueeze(0)
    return resized_frame, tensor


def postprocess_mask(mask_tensor: torch.Tensor, threshold: float = 0.5) ->
np.ndarray:
    """
    Перетворює вихід моделі у бінарну маску типу uint8.
    """
    mask = mask_tensor.squeeze().detach().cpu().numpy()
    binary = (mask >= threshold).astype("uint8")
    return binary
```

Лістинг А.4 – Файл segmentation_model.py

```
from __future__ import annotations

import torch
import torch.nn as nn


class ConvBlock(nn.Module):
    def __init__(self, in_channels: int, out_channels: int) -> None:
        super().__init__()
        self.block = nn.Sequential(
            nn.Conv2d(in_channels, out_channels, kernel_size=3, padding=1,
bias=False),
            nn.BatchNorm2d(out_channels),
            nn.ReLU(inplace=True),
            nn.Conv2d(out_channels, out_channels, kernel_size=3, padding=1,
bias=False),
            nn.BatchNorm2d(out_channels),
            nn.ReLU(inplace=True),
        )

    def forward(self, x: torch.Tensor) -> torch.Tensor:
```

```
        return self.block(x)


class EncoderDecoderNet(nn.Module):
    def __init__(self) -> None:
        super().__init__()

        self.enc1 = ConvBlock(3, 32)
        self.pool1 = nn.MaxPool2d(2)

        self.enc2 = ConvBlock(32, 64)
        self.pool2 = nn.MaxPool2d(2)

        self.enc3 = ConvBlock(64, 128)
        self.pool3 = nn.MaxPool2d(2)

        self.bottleneck = ConvBlock(128, 256)

        self.up3 = nn.ConvTranspose2d(256, 128, kernel_size=2, stride=2)
        self.dec3 = ConvBlock(256, 128)

        self.up2 = nn.ConvTranspose2d(128, 64, kernel_size=2, stride=2)
        self.dec2 = ConvBlock(128, 64)

        self.up1 = nn.ConvTranspose2d(64, 32, kernel_size=2, stride=2)
        self.dec1 = ConvBlock(64, 32)

        self.out_conv = nn.Conv2d(32, 1, kernel_size=1)
        self.activation = nn.Sigmoid()

    def forward(self, x: torch.Tensor) -> torch.Tensor:
        e1 = self.enc1(x)
        p1 = self.pool1(e1)

        e2 = self.enc2(p1)
        p2 = self.pool2(e2)

        e3 = self.enc3(p2)
        p3 = self.pool3(e3)

        b = self.bottleneck(p3)

        u3 = self.up3(b)
        d3 = self.dec3(torch.cat([u3, e3], dim=1))

        u2 = self.up2(d3)
        d2 = self.dec2(torch.cat([u2, e2], dim=1))

        u1 = self.up1(d2)
        d1 = self.dec1(torch.cat([u1, e1], dim=1))

        out = self.out_conv(d1)
        return self.activation(out)


def load_model(weights_path: str | None = None, device: str = "cpu") ->
EncoderDecoderNet:
    model = EncoderDecoderNet().to(device)
    if weights_path:
        state = torch.load(weights_path, map_location=device)
        model.load_state_dict(state)
    model.eval()
    return model
```

Лістинг А.5 – Файл mask_utils.py

```
from __future__ import annotations

import cv2
import numpy as np


def build_mask(mask_prediction: np.ndarray, threshold: float = 0.5) -> np.ndarray:
    return (mask_prediction >= threshold).astype("uint8")


def apply_mask(frame: np.ndarray, mask: np.ndarray) -> np.ndarray:
    if mask.ndim == 2:
        mask_3d = np.stack([mask] * 3, axis=-1)
    else:
        mask_3d = mask
    return (frame * mask_3d).astype("uint8")


def create_overlay(frame: np.ndarray, mask: np.ndarray, alpha: float = 0.4) ->
np.ndarray:
    color_mask = np.zeros_like(frame)
    color_mask[:, :, 1] = mask * 255
    overlay = cv2.addWeighted(frame, 1.0, color_mask, alpha, 0)
    return overlay


def crop_by_mask(frame: np.ndarray, mask: np.ndarray) -> np.ndarray:
    ys, xs = np.where(mask > 0)
    if len(xs) == 0 or len(ys) == 0:
        return np.zeros((1, 1, 3), dtype=np.uint8)

    x_min, x_max = xs.min(), xs.max()
    y_min, y_max = ys.min(), ys.max()

    crop = frame[y_min:y_max + 1, x_min:x_max + 1]
    return crop


def mask_area(mask: np.ndarray) -> int:
    return int(mask.sum())
```

Лістинг А.6 – Файл event_logic.py

```
from __future__ import annotations

import numpy as np


def check_event(mask: np.ndarray, min_pixels: int = 500) -> bool:
    """
    Подія формується, якщо площа сегментованої області
    перевищує заданий поріг.
    """
    area = int(mask.sum())
    return area >= min_pixels


def build_event_text(frame_index: int, object_pixels: int) -> str:
    return (
        f"Подія виявлена у кадрі {frame_index}. "
        f"Площа сегментованої області: {object_pixels} пікселів."
    )
```

Лістинг А.7 – Файл storage.py

```
from __future__ import annotations

from datetime import datetime
from pathlib import Path

import cv2
import numpy as np


def save_event(log_path: Path, event_text: str) -> None:
    timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    with open(log_path, "a", encoding="utf-8") as file:
        file.write(f"{timestamp} - {event_text}\n")


def save_mask(mask_path: Path, mask: np.ndarray) -> None:
    cv2.imwrite(str(mask_path), mask * 255)


def save_crop(crop_path: Path, crop: np.ndarray) -> None:
    if crop.size > 0:
        cv2.imwrite(str(crop_path), crop)


def save_overlay(overlay_path: Path, overlay: np.ndarray) -> None:
    cv2.imwrite(str(overlay_path), overlay)
```

Лістинг А.8 – Файл notify.py

```
def notify(event_text: str) -> None:
    print(f"[ALERT] {event_text}")
```

Лістинг А.9 – Файл main.py

```
from __future__ import annotations

import cv2
import torch

from config import AppConfig, ensure_output_structure
from event_logic import build_event_text, check_event
from mask_utils import apply_mask, build_mask, create_overlay, crop_by_mask,
mask_area
from notify import notify
from preprocess import postprocess_mask, preprocess_frame
from segmentation_model import load_model
from storage import save_crop, save_event, save_mask, save_overlay
from video_stream import VideoStream


def run_system(config: AppConfig) -> None:
    device = "cuda" if torch.cuda.is_available() else "cpu"
    paths = ensure_output_structure(config)

    stream = VideoStream(config.video_source)
    model = load_model(config.model_path if config.model_path else None,
device=device)

    frame_index = 0
    log_path = paths["logs"] / config.log_filename
```

```
    try:
        while True:
            frame = stream.read_frame()
            if frame is None:
                break

            frame_index += 1
            if config.max_frames is not None and frame_index > config.max_frames:
                break

            resized_frame, input_tensor = preprocess_frame(
                frame,
                size=(config.input_width, config.input_height)
            )
            input_tensor = input_tensor.to(device)

            with torch.no_grad():
                prediction = model(input_tensor)

            mask = postprocess_mask(prediction, threshold=config.mask_threshold)
            binary_mask = build_mask(mask, threshold=0.5)

            masked_object = apply_mask(resized_frame, binary_mask)
            cropped_object = crop_by_mask(resized_frame, binary_mask)
            overlay = create_overlay(resized_frame, binary_mask)

            event_detected = check_event(binary_mask,
min_pixels=config.min_object_pixels)

            if event_detected:
                area = mask_area(binary_mask)
                event_text = build_event_text(frame_index, area)
                save_event(log_path, event_text)
                notify(event_text)

                if config.save_masks:
                    save_mask(paths["masks"] / f"mask_{frame_index:05d}.png",
binary_mask)

                if config.save_crops:
                    save_crop(paths["crops"] / f"crop_{frame_index:05d}.png",
cropped_object)

                if config.save_overlay:
                    save_overlay(paths["overlay"] /
f"overlay_{frame_index:05d}.png", overlay)

            if config.display_window:
                cv2.imshow("Input Frame", resized_frame)
                cv2.imshow("Mask", binary_mask * 255)
                cv2.imshow("Overlay", overlay)
                cv2.imshow("Masked Object", masked_object)

                key = cv2.waitKey(1) & 0xFF
                if key == 27:
                    break

    finally:
        stream.release()
        cv2.destroyAllWindows()


if __name__ == "__main__":
    app_config = AppConfig(
        video_source=0,
        model_path="models/encoder_decoder.pth",
```

```
        display_window=True,
        max_frames=None,
    )
    run_system(app_config)
```

Лістинг А.10 – Файл requirements.txt

```
numpy>=1.24
opencv-python>=4.8
matplotlib>=3.7
torch>=2.0
torchvision>=0.15
```

Додаток Б

Апробація результатів кваліфікаційної роботи

Почесні співголови конференції:

Десятнюк О.М., ректор Західноукраїнського національного університету, д-р економічних наук, професор;

Дивак М.П., д-р технічних наук, професор, проректор з наукової роботи ЗУНУ;

Голова конференції:

Березький О.М., д-р технічних наук, професор, професор кафедри комп'ютерної інженерії Західноукраїнський національний університет;

Заступники голови конференції:

Мельник Г.М., к.т.н, доцент, Західноукраїнський національний університет;

Піцун О.Й., к.т.н. доцент, Західноукраїнський національний університет;

ПРОГРАМНИЙ КОМІТЕТ КОНФЕРЕНЦІЇ

Склад програмного комітету:

Семанюк В.З., д-р економічних наук, професор, начальник науково-дослідної частини Західноукраїнського національного університету

Антощук С.Г., д.т.н, професор, Національний університет «Одеська політехніка»;

Баловсяк С.В., д.т.н, професор, Чернівецький національний університет

Бармак О.В., д.т.н., професор, Хмельницький національний університет

Батько Ю.М., к.т.н., доцент, Західноукраїнський національний університет

Березька К.М.,к.т.н.,доцент, Західноукраїнський національний університет

Винокурова О.А., д.т.н., професор, Львівський національний університет імені Івана Франка

Возна Н.Я., д.т.н., професор, Західноукраїнський національний університет;

Говорущенко Т.О., д.т.н., професор, Хмельницький національний університет;

Дубчак Л.О., к.т.н., доцент, Західноукраїнський національний університет;

Дунець Р.Б., д.т.н., професор, НУ "Львівська політехніка";

Ізонін І.В., д.т.н., доцент, НУ "Львівська політехніка";

Комар М.П., д.т.н, професор, Західноукраїнський національний університет;

Литвиненко В.І., д.н.т, професор, Національний технічний університет України "Київський політехнічний інститут";

Лупенко С.А., д.т.н., професор, Опольський технологічний університет, Польща;

Ляцинський П.Б., доктор філософії з комп'ютерних наук, НУ "Львівська політехніка" ;

Мельникова Н.І., д.т.н., професор, НУ "Львівська політехніка" ;

Мулеса О.Ю., д.т.н., професор, Пряшівський університет, Словаччина

Пелешко Д.Д., д.т.н., професор, Львівський національний університет імені Івана Франка;

Сельський П.Р., д.м.н., професор, Тернопільський національний медичний університет імені І. Я. Горбачевського ;
Субботін С.О., д.т.н., професор, Національний університет «Запорізька політехніка» ;
Теслюк В.М., д.т.н., професор, НУ "Львівська політехніка" ;
Тимченко Л.І., д.т.н., професор, Державний університет інфраструктури та технологій;
Цмоць І.Г., д.т.н., професор, НУ "Львівська політехніка" ;
Якименко І.З., к.т.н., доцент, Західноукраїнський національний університет;
Яровий А.А., д.т.н., професор, Вінницький національний технічний університет ;
Яцків В.В., д.т.н., професор, Західноукраїнський національний університет.

ОРГАНІЗАЦІЙНИЙ КОМІТЕТ КОНФЕРЕНЦІЇ

Склад організаційного комітету:

Мельник Г.М. к.т.н., доцент, Західноукраїнський національний університет
Піцун О.Й. к.т.н., доцент, Західноукраїнський національний університет

Технічна підтримка:

Зінькевич О. В. студентка, Західноукраїнський національний університет
Кіт М. О. студентка, Західноукраїнський національний університет
Лебединський Т.В. студент,Західноукраїнський національний університет;
Кришталь Е.Р., студент, Західноукраїнський національний університет;

Метою конференції є представлення та обговорення наукових і практичних результатів, сприяння активізації творчої і інноваційної діяльності студентів, аспірантів і молодих вчених.
Конференція проводиться із залученням Ради Молодих Вчених та Студентського Наукового Товариства ФКІТ ЗУНУ.

IV Всеукраїнська науково-практична конференція студентів, аспірантів та молодих вчених «Інтелектуальні комп'ютерні системи та мережі» (ІКСМ весна 2026), м. Тернопіль, ЗУНУ, 29 травня 2026 р. Тернопіль, 2026

Адреса:
Вул. О. Теліги, 8, корпус №6, Тернопіль
URL: https://ki.wunu.edu.ua/conference/
e-mail: kistudconference@gmail.com

Тези доповідей подаються за оригіналом рукопису

Омельчук О.А.
студент 4 курсу ФКІТ ЗУНУ
Науковий керівник д.т.н., професор Комар М.П., кафедра ІОСУ ЗУНУ

ІНТЕЛЕКТУАЛЬНЕ ВІДЕОСПОСТЕРЕЖЕННЯ В РЕЖИМІ РЕАЛЬНОГО ЧАСУ НА ОСНОВІ ЗГОРТКОВИХ НЕЙРОННИХ МЕРЕЖ ТА ТУМАННИХ ОБЧИСЛЕНЬ

Вступ. У сучасних умовах стрімкого розвитку цифрових технологій системи відеоспостереження стали важливим складником інформаційної інфраструктури безпеки. Такі системи застосовуються на об'єктах критичної інфраструктури, у транспортних вузлах, громадських просторах, освітніх закладах, промислових підприємствах і приватному секторі. Зростання кількості камер, безперервне надходження великих обсягів відеоданих і потреба в оперативному реагуванні на потенційно небезпечні події зумовили перехід від звичайного пасивного запису відео до інтелектуального автоматизованого аналізу відеопотоку [1, 2].

Традиційні системи відеоспостереження, у яких основний акцент зроблено на передавання та централізоване збереження відео, мають низку суттєвих обмежень. До них належать значне навантаження на мережеву інфраструктуру, висока затримка оброблення, обмежені можливості масштабування, а також залежність від центрального обчислювального вузла. За таких умов знижується ефективність системи в задачах реального часу, де критичне значення мають швидке виявлення об'єктів, подій або аномальних ситуацій та негайне формування повідомлення про них [2-5].

Одним із найбільш перспективних напрямів розв'язання цієї задачі стало поєднання методів глибокого навчання із розподіленими підходами до оброблення даних. Зокрема, згорткові нейронні мережі продемонстрували високу ефективність у задачах комп'ютерного зору: виявленні об'єктів, сегментації зображень, розпізнаванні осіб, аналізі сцен і подій у відеопотоці [6, 7]. Водночас для забезпечення роботи в режимі реального часу важливим стало перенесення частини обчислень ближче до джерела даних, тобто до камер або локальних вузлів опрацювання. Саме тому дедалі ширшого застосування набувають туманні обчислення, які забезпечують проміжний рівень між периферійними пристроями та хмарною інфраструктурою, зменшуючи затримку, обсяг даних, що передаються і навантаження на центральні сервери [4, 8].

Постановка задачі. Проведений аналіз предметної області показав, що сучасні системи відеоспостереження дедалі активніше переходять від простого запису відео до автоматизованого аналізу подій у потоці кадрів. Використання згорткових нейронних мереж суттєво підвищило ефективність виявлення об'єктів, а застосування туманних обчислень створило умови для перенесення частини оброблення ближче до джерела даних. Це особливо важливо для систем, які повинні працювати в режимі реального часу та забезпечувати швидке реагування на події.

Разом із тим аналіз існуючих рішень засвідчив, що більшість із них мають певні обмеження. Частина систем орієнтована переважно на централізоване збереження та перегляд відео, тому не забезпечує достатнього рівня автоматизації аналізу. Інші рішення реалізують інтелектуальне виявлення об'єктів, але потребують значних апаратних ресурсів або не враховують особливості розподіленого оброблення даних. Також поширеними є системи, у яких увагу зосереджено на окремому алгоритмі чи моделі, без побудови цілісної програмної архітектури, придатної до практичного використання в умовах безперервного відеопотоку.

Для задач інтелектуального відеоспостереження реального часу важливими є одразу кілька характеристик: здатність системи своєчасно обробляти потік кадрів, автоматично виявляти об'єкти у сцені, інтерпретувати результат виявлення як подію прикладного рівня та забезпечувати подальше збереження або передавання результату. За відсутності такої узгодженої роботи програмних компонентів система або втрачає оперативність, або зводиться лише до часткового розв'язання задачі.

Об'єктом дослідження є процес інтелектуального оброблення відеопотоку в системах

відеоспостереження реального часу. Предметом дослідження є методи, моделі та програмні засоби аналізу відеоданих на основі згорткових нейронних мереж у середовищі туманних обчислень. Метою роботи є проєктування програмної системи інтелектуального відеоспостереження, що забезпечує аналіз відеопотоку в режимі реального часу на основі згорткових нейронних мереж і туманних обчислень.

Основний матеріал. Узагальнену архітектуру програмної системи спроєктовано у вигляді кількох взаємопов'язаних модулів, кожен із яких виконує чітко визначену функцію (рисунок 1).

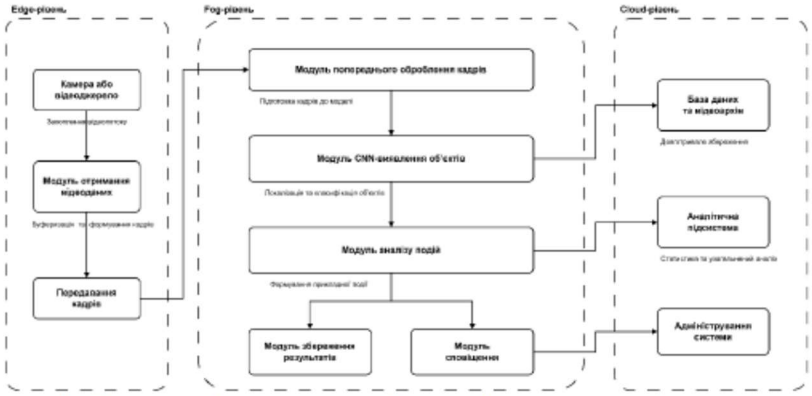

Рисунок 1 – Загальна архітектура програмної системи інтелектуального відеоспостереження

Першим функціональним компонентом є модуль отримання відеоданих. Його призначення полягає у підключенні до джерела відео та формуванні послідовності кадрів для подальшої обробки. Як джерело можуть використовуватися IP-камера, вебкамера або тестовий відеофайл. На цьому етапі система повинна забезпечувати стабільне надходження кадрів, їх зчитування у правильному порядку та передавання до наступного модуля без порушення часової послідовності. Цей модуль є початковою точкою роботи всієї системи, тому від його коректного функціонування залежить стабільність подальшого аналізу.

Другим компонентом є модуль попереднього оброблення кадрів. Він призначений для приведення відеоданих до формату, придатного для аналізу нейронною мережею. У його межах виконуються масштабування кадру, зміна роздільної здатності, перетворення кольорового простору за потреби, нормалізація вхідних даних та усунення надлишкових елементів, які можуть ускладнювати подальше оброблення. Важливість цього модуля полягає в тому, що якість попередньої підготовки кадрів безпосередньо впливає на точність роботи моделі виявлення об'єктів.

Третім і центральним компонентом системи є модуль виявлення об'єктів на основі згорткової нейронної мережі. Саме в ньому реалізується інтелектуальна складова всієї системи. На вхід цього модуля подається підготовлений кадр, а на виході формується набір знайдених об'єктів із координатами, класами та оцінкою впевненості. У контексті цієї роботи основною задачею такого модуля є виявлення цільових об'єктів у кадрі з прийнятною швидкодією. Тому модель повинна бути достатньо точною, але водночас придатною до використання в умовах обмежених обчислювальних ресурсів fog-рівня.

Після отримання результатів детекції вони передаються до модуля аналізу подій. Його функція полягає в інтерпретації результатів нейромережевого виявлення з позиції прикладної логіки. Якщо модуль CNN-виявлення визначає технічний факт наявності об'єкта в кадрі, то модуль аналізу подій визначає змістовний факт події. Наприклад, саме тут перевіряється, чи належить знайдений об'єкт до контрольованого класу, чи виконуються умови фіксації події, чи

слід передати інформацію на збереження та чи потрібно згенерувати повідомлення. Таким чином, цей модуль є сполучною ланкою між комп'ютерним зором і прикладною логікою системи спостереження.

П'ятим компонентом є модуль збереження результатів. Його призначення полягає у фіксації результатів роботи системи у структурованому вигляді. До таких результатів можуть належати кадри з виявленими об'єктами, службові метадані, записи журналу подій, часові мітки, класи об'єктів та рівні впевненості детекції. Збереження результатів є необхідним не лише для архівування, а й для подальшого аналізу роботи системи, перевірки коректності її функціонування та формування статистичних звітів.

Шостим компонентом є модуль сповіщення, який відповідає за формування реакції системи на виявлену подію. Його робота може полягати у відображенні повідомлення в інтерфейсі, створенні текстового запису, передаванні сигналу до зовнішньої підсистеми або в іншій формі інформування користувача.

Ключовим елементом програмної системи інтелектуального відеоспостереження є модель оброблення відеопотоку, яка забезпечує перетворення вхідного кадру у результат сегментації та подальше виділення цільового об'єкта. На відміну від спрощеного підходу, за якого результатом роботи моделі є лише факт наявності об'єкта або його прямокутна область, у цій роботі використано модель, орієнтовану на побудову маски об'єкта. Такий підхід є доцільним для задач інтелектуального відеоспостереження, оскільки дозволяє точніше відокремити об'єкт від фону та підвищити інформативність подальшого аналізу.

Для розроблюваної системи обрано модель на основі згорткової нейронної мережі типу encoder–decoder. Вона забезпечує покадрове оброблення відеопотоку, у межах якого кожен кадр проходить послідовні етапи стискання ознак, формування компактного представлення сцени, відновлення просторової структури та побудови вихідної маски. Після цього за допомогою окремого блоку Mask-Crop формується виділена область об'єкта, яка може бути використана для подальшого прикладного аналізу. Структуру розробленої моделі подано на рисунку 2.

Рисунок 2 – Структура моделі оброблення відеопотоку на основі згорткової нейронної мережі encoder–decoder

Вхідний блок подання кадру призначений для приймання поточного зображення з відеопотоку та приведення його до формату, сумісного з нейромережевою моделлю. У межах цієї роботи кадр розглядається як базова одиниця аналізу. Перед поданням у мережу він має бути нормалізований і приведений до фіксованого розміру. Таким чином, вхідний блок формує уніфіковане подання даних, на основі якого працюють усі наступні компоненти моделі.

Енкодерна частина виконує послідовне виділення ознак із вхідного кадру. На цьому етапі мережа переходить від початкового зображення до багаторівневого ознакового подання сцени. У процесі такого переходу просторовий розмір карт ознак зменшується, а кількість семантичної інформації збільшується. Саме енкодер забезпечує формування високорівневих ознак, що відображають форму, контури та розміщення об'єкта в кадрі. У практичному сенсі цей блок виконує стискання вхідної інформації з одночасним посиленням її змістовності.

Центральний вузол bottleneck є найглибшим рівнем моделі. У ньому формується компактне семантичне представлення сцени, яке містить найбільш узагальнену інформацію про об'єкт і фон. Цей блок виконує роль проміжної ланки між енкодером і декодером. Його призначення полягає не лише в подальшому стисканні ознак, а й у збереженні найбільш значущої інформації, необхідної для подальшого відновлення просторової структури об'єкта.

Декодерна частина виконує зворотне перетворення ознак. Якщо енкодер поступово зменшував просторову роздільну здатність, то декодер відновлює її, поєднуючи глибокі семантичні ознаки з інформацією про локальну структуру об'єкта. Саме на цьому етапі формується наближене просторове представлення маски. Декодер є критично важливим для сегментації, оскільки він забезпечує повернення від стислого ознакового подання до зображення, у якому можна визначити межі об'єкта по пікселях.

Вихідний блок побудови маски формує результат роботи моделі у вигляді одноканальної маски. У такій масці одна частина пікселів відповідає об'єкту або області дії, а інша — фону. Саме цей блок завершує нейромережевий цикл оброблення кадру. На відміну від класичної детекції, де результатом є координати рамки, у цьому випадку результатом є просторово точніше піксельне подання об'єкта.

Після побудови маски використовується блок Mask-Crop, який виконує виділення об'єкта за сформованою маскою. Його функція полягає у накладанні маски на початковий кадр і відокремленні цільової області від надлишкового фону. Унаслідок цього формується очищене представлення об'єкта, яке є більш придатним для наступного етапу аналізу, ніж повний кадр. Такий підхід зменшує вплив фонового шуму та підвищує інформативність оброблення.

Розроблена модель має послідовну логіку функціонування. Спочатку поточний кадр подається на вхідний блок, потім проходить через енкодер і bottleneck, після чого відновлюється в декодері та завершується побудовою маски. Далі блок Mask-Crop формує виділений фрагмент об'єкта. Таким чином, результатом роботи моделі є не лише ознака наявності цільового об'єкта, а конкретна сегментована область, яка може бути використана в системі для формування події або збереження результату.

Висновки. Спроєктовано модульну архітектуру системи, яка охоплює отримання відеоданих, попереднє оброблення кадрів, нейромережевий аналіз, формування подій, збереження результатів і сповіщення. Запропоновано модель оброблення відеопотоку на основі CNN типу encoder–decoder, яка забезпечує побудову маски об'єкта та його виділення за допомогою блоку Mask-Crop. Визначено, що використання туманних обчислень дає змогу перенести основні операції аналізу ближче до джерела відеоданих, зменшити затримку оброблення та підвищити оперативність реагування системи.

Список літератури

1. Chen N., Chen Y., You Y., Ling H., Liang P., Zimmermann R. Dynamic Urban Surveillance Video Stream Processing Using Fog Computing. Proceedings of the IEEE Second International Conference on Multimedia Big Data (BigMM). 2016. P. 105–112.
2. Nasir M., Muhammad K., Lloret J., Sangaiah A. K., Sajjad M. Fog Computing Enabled Cost-Effective Distributed Summarization of Surveillance Videos for Smart Cities. Journal of Parallel and Distributed Computing. 2019. Vol. 126. P. 161–170.
3. Abbas N., Zhang Y., Taherkordi A., Skeie T. Mobile Edge Computing: A Survey. IEEE Internet of Things Journal. 2018. Vol. 5, no. 1. P. 450–465.
4. Chiang M., Ha S., Risso F., Zhang T., Chih-Lin I. Clarifying Fog Computing and Networking: 10 Questions and Answers. IEEE Communications Magazine. 2017. Vol. 55. P. 18–20.
5. De Donno M., Tange K., Dragoni N. Foundations and Evolution of Modern Computing Paradigms: Cloud, IoT, Edge, and Fog. IEEE Access. 2019. Vol. 7. P. 150936–150948.
6. He K., Zhang X., Ren S., Sun J. Deep Residual Learning for Image Recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016. P. 770–778.
7. Redmon J., Divvala S., Girshick R., Farhadi A. You Only Look Once: Unified, Real-Time Object Detection. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016. P. 779–788.
8. Li C., Xue Y., Wang J., Zhang W., Li T. Edge-Oriented Computing Paradigms: A Survey on Architecture Design and System Management. ACM Computing Surveys. 2018. Vol. 51. Art. 39.