НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ЮРИДИЧНА АКАДЕМІЯ»
ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інформаційних технологій

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Автоматизоване виявлення нетипових подій у системах відеоспостереження з використанням штучного інтелекту»

Виконав: здобувач 2 курсу
Рівень магістр
Галузь знань 12 «Інформаційні технології»,
спеціальність 122 «Комп’ютерні науки»
Флюнт Владислав Орестович

Керівник: к.т.н., доцент
Чикунов Павло Олександрович

Рецензент: к.т.н., доцент
Манаков Сергій Юрійович

Одеса – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ЮРИДИЧНА АКАДЕМІЯ»
Факультет **кібербезпеки та інформаційних технологій**
Кафедра **інформаційних технологій**
Галузь знань: **12 Інформаційні технології**
Спеціальність: **122 Комп'ютерні науки**
Назва освітньої програми: **Комп'ютерні науки**

ЗАТВЕРДЖУЮ
Завідувач кафедри інформаційних технологій
доцент Н.І. Логінова
_____ «____ »_____ 2025 року

З А В Д А Н Н Я
на кваліфікаційну (магістерську) роботу
здобувачу Флюнт Владиславу Орестовичу

1. Тема роботи: «Автоматизоване виявлення нетипових подій у системах відеоспостереження з використанням штучного інтелекту»
Керівник роботи: к.т.н, доцент Чикунов Павло Олександрович
затверджені наказом НУ ОЮА від «10» жовтня 2025 року № 3182-06
2. Строк подання здобувачем роботи «25» листопада 2025 рік
3. Дата видачі завдання «02» червня 2025 рік

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Огляд предметної області та існуючих рішень	10.09.2025	
2	Формулювання мети, завдань та обґрунтування актуальності дослідження.	01.10.2025	
3	Визначення вимог до системи та моделювання її функціонування.	15.10.2025	
4	Проєктування архітектури та внутрішніх модулів системи.	01.11.2025	
5	Реалізація програмного прототипу системи.	05.11.2025	
6	Експериментальне тестування, дослідження та верифікація моделі.	10.11.2025	
7	Оформлення пояснювальної записки.	17.11.2025	
8	Подача електронного варіанта роботи для перевірки на плагіат	20.11.2025	

Здобувач _____ Флюнт В.О._____
(підпис) (прізвище та ініціали)

Керівник роботи _____Чикунов П.О._____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота на тему «Автоматизоване виявлення нетипових подій у системах відеоспостереження з використанням штучного інтелекту», виконана на здобуття другого (магістерського) рівня вищої освіти за спеціальністю 122 «Комп'ютерні науки», освітньою програмою «Комп'ютерні науки», містить 19 рисунків, 9 таблиць, 2 додатки та 27 літературних джерел. Робота викладена на 57 сторінках загального тексту, з них 42 сторінки становлять основний зміст.

Метою роботи є розроблення архітектури, програмного прототипу та експериментальної методики оцінювання системи автоматизованого виявлення нетипових подій у відеопотоці на основі методів глибинного навчання.

Об'єктом дослідження є процеси аналізу та автоматичної інтерпретації відеопотоку в системах відеоспостереження.

Предметом дослідження є методи, моделі та програмні засоби виявлення аномалій у відеоданих з використанням глибинних нейронних мереж, а також підходи до побудови конвеєра обробки відео в режимі реального часу.

Методи дослідження ґрунтуються на використанні системного аналізу, математичного моделювання, UML-діаграмування, методів комп'ютерного зору, алгоритмів глибинного навчання, GPU-прискорення, аналізу продуктивності та експериментальної верифікації.

У роботі вирішено науково-практичне завдання зі створення прототипу інтелектуальної системи, яка забезпечує автоматизоване виявлення нетипових подій у відеопотоці з низькою затримкою інференсу, підтримкою обробки в реальному часі та можливістю інтеграції з зовнішніми сервісами.

Ключові слова: відеоспостереження, виявлення аномалій, штучний інтелект, глибинне навчання, автоенкодер, GPU-прискорення, відеоаналітика, комп'ютерний зір, PyTorch, FPS, latency.

ABSTRACT

The master's thesis titled "Automated Detection of Atypical Events in Video Surveillance Systems Using Artificial Intelligence", completed for obtaining the second (master's) level of higher education in the specialty 122 Computer Science, educational program Computer Science, contains 19 figures, 9 tables, 2 appendices, and 27 references. The thesis is presented on 57 pages of general text, including 42 pages of the main content.

The purpose of the thesis is to design the architecture, develop a software prototype, and conduct an experimental evaluation of a system for automated detection of atypical events in video streams using deep learning methods.

The object of the study is the processes of analysis and automatic interpretation of video streams in video surveillance systems.

The subject of the study is the methods, models, and software tools for anomaly detection in video data using deep neural networks, as well as approaches to constructing a real-time video processing pipeline.

The research methodology is based on system analysis, mathematical modelling, UML diagramming, computer vision methods, deep learning algorithms (autoencoder-based anomaly detection), GPU acceleration, performance profiling, and experimental verification.

The thesis solves a scientific and practical problem of developing a prototype of an intelligent system that provides automated detection of atypical events in video streams with low inference latency, real-time processing capability, and integration with external services.

Keywords: video surveillance, anomaly detection, artificial intelligence, deep learning, autoencoder, GPU acceleration, video analytics, computer vision, PyTorch, FPS, latency.

ЗМІСТ

ВСТУП

Ефективне забезпечення безпеки та моніторингу простору є одним із ключових напрямів розвитку сучасних інформаційних технологій. Постійне зростання кількості відеокамер, збільшення обсягів даних та вимоги до оперативного реагування на події створюють потребу у впровадженні інтелектуальних систем відеоаналізу, здатних автоматично виявляти нетипові або потенційно небезпечні ситуації. Такі системи спрямовані на підвищення рівня безпеки, зниження впливу людського чинника, автоматизацію моніторингу та підтримку прийняття рішень у режимі реального часу.

Попри наявність сучасних комерційних систем відеоаналітики, зокрема Avigilon UMD/UAD, BriefCam Video Analytics, Irisity IRIS+, Oosto Protect та Axis Object Analytics, їх широке впровадження часто обмежується високою вартістю, закритістю алгоритмів, складністю інтеграції з існуючими інфраструктурами та недостатньою гнучкістю в адаптації до специфічних сценаріїв спостереження. Це зумовлює актуальність розроблення відкритих, гнучких і масштабованих систем, побудованих на сучасних методах штучного інтелекту, здатних навчатися на реальних даних і забезпечувати автономне виявлення нетипових подій.

Метою магістерської роботи є розроблення архітектури та програмної реалізації системи автоматизованого виявлення нетипових подій у відеопотоках на основі методів глибинного навчання, що забезпечує високу точність детекції, адаптивність до середовища та обробку відео в реальному часі.

Об’єктом дослідження є процеси аналізу відеопотоків у системах відеоспостереження.

Предметом дослідження є методи, моделі та засоби штучного інтелекту, призначені для автоматичного виявлення аномалій у відео в умовах змінного середовища.

Методи дослідження базуються на використанні технологій комп’ютерного зору, глибинного навчання, системного аналізу, методів статистичної обробки даних, а також інструментів програмної інженерії для побудови архітектури та

навчання нейронних мереж. У роботі застосовуються бібліотеки OpenCV, PyTorch, TensorFlow, NumPy, Pandas, а також засоби GPU-прискорення за допомогою технології CUDA.

Наукова новизна роботи полягає у розробленні концептуальної моделі інтелектуальної системи відеоаналізу, яка поєднує глибинний автоенкодер з часовими нейронними мережами (3D-CNN, LSTM або Vision Transformer), що дозволяє здійснювати виявлення нетипових подій у режимі реального часу без попереднього маркування аномалій. Запропонована модель передбачає гібридну архітектуру, здатну до самонавчання та масштабування у розподілених середовищах.

Практичне значення результатів полягає у можливості використання розробленої системи як бази для впровадження інтелектуальних технологій відеоаналізу в інфраструктурі «розумних міст», транспортних вузлах, промислових об’єктах та системах охорони громадської безпеки. Реалізація розробленої архітектури забезпечує зниження навантаження на операторів, зменшення кількості хибних спрацьовувань та скорочення часу реакції на інциденти.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– провести аналіз предметної області та існуючих методів виявлення аномалій у відеопотоках;

– здійснити огляд і SWOT-аналіз сучасних систем відеоаналітики;

– побудувати концептуальну та архітектурну модель системи автоматизованого виявлення подій;

– спроєктувати UML-діаграми, що описують модулі системи, структуру даних і сценарії обробки відео;

– реалізувати прототип системи з використанням бібліотек глибинного навчання PyTorch і OpenCV;

– провести експериментальні дослідження ефективності моделі на відкритих наборах відео (UCSD Ped2, Avenue, UCF-Crime);

– оцінити точність, повноту, F1-score та швидкодію системи у порівнянні з традиційними rule-based підходами..

1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Класифікація та архітектура інтелектуальних систем відеоаналізу

Системи відеоспостереження є невід'ємною складовою сучасних засобів безпеки, що використовуються для контролю громадських просторів, транспортних вузлів, промислових об'єктів, навчальних закладів тощо. Основним завданням таких систем є забезпечення своєчасного виявлення небезпечних або нетипових подій, що можуть свідчити про порушення безпеки чи аномальну поведінку об'єктів.

Класичні системи відеоспостереження передбачають участь оператора, який переглядає потоки з великої кількості камер і приймає рішення про подальші дії. Проте людський фактор істотно обмежує ефективність спостереження, оскільки оператор не здатний одночасно контролювати десятки потоків та реагувати на всі події в режимі реального часу. Це зумовлює потребу в автоматизації аналізу відеоданих і впровадженні інтелектуальних алгоритмів розпізнавання подій.

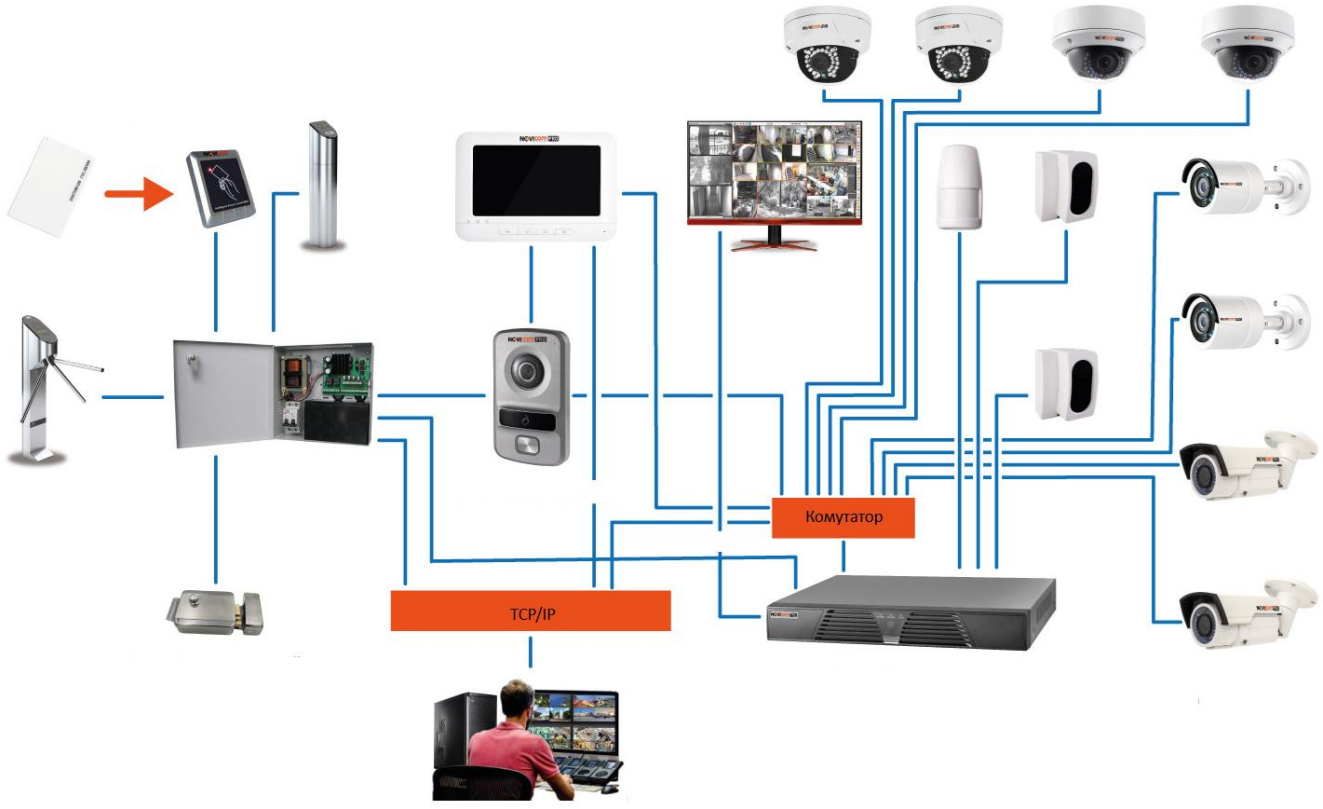

Рисунок 1.1 – Структура класичних систем відеоаналізу

Інтелектуальні системи відеоаналізу (IVA – Intelligent Video Analytics) ґрунтуються на застосуванні алгоритмів комп’ютерного зору, машинного навчання та глибинних нейронних мереж. Їх архітектура зазвичай включає такі компоненти:

1. модуль збору відеопотоків (камери, відеосервери, IP-потоки);

2. модуль попередньої обробки (нормалізація, кадрування, зменшення шуму, виділення рухомих об’єктів);

3. модуль аналітики (детекція, трекінг, класифікація об’єктів, виявлення аномалій);

4. модуль збереження та управління даними (бази даних, сховище відео, журнал подій);

5. інтерфейс користувача для відображення результатів аналізу.

За рівнем автономності виділяють:

1.пасивні системи, що лише фіксують відеопотоки без аналітики;

2. реактивні системи, які визначають події за заданими правилами (рух у зоні, перетин межі);

3. прогнозні системи, що використовують штучний інтелект для самонавчання та розпізнавання нетипових сценаріїв.

1.2 Огляд методів виявлення нетипових подій у відеопотоках

Виявлення нетипових (аномальних) подій є складною задачею через невизначеність і різноманітність можливих сценаріїв. Аномалією вважається подія, що відрізняється від більшості звичних шаблонів поведінки – наприклад, поява людини в забороненій зоні, залишений предмет, раптовий біг або агресивна поведінка.

Основні підходи до виявлення аномалій у відео можна поділити на:

– методи на основі правил (rule-based): система реагує на порушення наперед заданих умов (наприклад, рух у нічний час);

– статистичні методи: аналізують відхилення від нормальних розподілів ознак об’єктів;

– кластеризаційні методи: формують групи подібних відеофрагментів і виявляють ті, що не належать жодному кластеру;

– методи машинного навчання: будують моделі “нормальної” поведінки та порівнюють поточні спостереження з очікуваними.

Традиційні правило-орієнтовані методи, що ґрунтуються на заздалегідь описаних сценаріях, забезпечують прийнятну точність лише в умовах стабільного середовища. За динамічних змін сцени або появи нових поведінкових патернів їх ефективність різко знижується, що зумовило необхідність переходу до моделей, здатних самостійно узагальнювати властивості відеоданих.

Першим напрямом такого переходу стали статистичні методи, у яких нормальна поведінка сцени описується ймовірнісною моделлю, а відхилення визначаються як події низької ймовірності. Моделі типу Gaussian Mixture Model застосовувалися для сегментації руху й аналізу зміни інтенсивності кадрів, однак виявилися чутливими до шумів та інтенсивних локальних відмінностей, а також вимагали значних ресурсів для масштабування на великі набори даних.

Подальший розвиток отримали алгоритми машинного навчання та кластеризації, такі як k-means, DBSCAN і One-Class SVM, які дають змогу групувати відрізки відео за подібністю ознак і визначати аномальні сегменти як ті, що не належать до жодного кластера. Проте таким підходам бракувало здатності враховувати складні просторово-часові залежності у відео, що обмежувало їх ефективність у реальних системах відеоспостереження.

Суттєвий прогрес у галузі було досягнуто завдяки використанню методів глибинного навчання. Автоенкодери стали одними з перших моделей, здатних виявляти аномалії у відео без учителя. У роботі [1] запропоновано метод реконструкції нормальних кадрів, у якому значна похибка відновлення свідчить про нетипову подію. Подальший розвиток цього підходу продемонстрували у автори роботи [2], які застосували повнозгорткову нейромережу для прискорення обчислень і підвищення точності на динамічних сценах.

Важливим напрямом розвитку стали методи прогнозування наступного кадра. У роботі [3] запропоновано модель, яка навчається передбачати майбутній

стан сцени, а відхилення між прогнозованим та реальним кадром інтерпретується як аномалія. Цей підхід дозволяє ефективно реагувати на різкі зміни руху, появу нових об’єктів або порушення звичних траєкторій.

Окрему групу становлять моделі, що поєднують просторові та часові залежності. Дослідження [4] продемонструвало ефективність кластеризації нормальних поведінкових патернів у комбінації з глибинними ознаками. Пізніші роботи [5], показали переваги підходів continual learning (безперервне навчання) – це підхід у машинному навчанні, при якому модель навчається поступово, отримуючи нові дані з часом, і не забуває те, що було вивчено раніше.

Таким чином, методи виявлення аномалій у відеопотоках еволюціонували від жорстких правило-орієнтованих процедур до гнучких глибинних моделей, здатних аналізувати просторово-часові взаємозв’язки, узагальнювати закономірності та адаптуватися до контексту. Сучасні системи виявлення нетипових подій орієнтовані на гібридні архітектури, що поєднують переваги автоенкодерів, 3D-CNN та трансформерних моделей для забезпечення стабільності, точності та роботи у реальному часі.

1.3 Використання технологій штучного інтелекту для автоматизованого відеоаналізу

Однією з ключових задач у відеоаналізі є детекція об’єктів, яка істотно покращилася завдяки моделям сімейств Faster R-CNN, SSD та YOLO. Архітектура YOLOv3 та її подальші модифікації продемонстрували здатність поєднувати високу точність та швидкодію, що зробило їх придатними для використання у реальному часі [6]. Детекція об’єктів є критично важливою складовою виявлення аномалій, оскільки дозволяє виділити значущі об’єкти сцени та аналізувати зміни їхньої поведінки.

Для виявлення аномалій особливу роль відіграють моделі, що враховують часову складову відео. Рекурентні нейронні мережі, такі як LSTM, застосовуються для моделювання руху об’єктів та зміни станів сцени в часі. У роботах,

присвячених спостереженню за натовпами та пішохідними потоками, продемонстровано, що LSTM-архітектури здатні фіксувати закономірності руху та виділяти нетипові траєкторії [7].

Значного поширення набули також моделі на основі трьохвимірних згорткових нейронних мереж (3D-CNN), які опрацьовують не окремі кадри, а послідовності кадрів як єдиний об’єм даних. У роботі Tran et al. [8] показано, що використання 3D-CNN дозволяє суттєво покращити виявлення складних дій та подій завдяки інтеграції просторової та часової інформації. Такі моделі забезпечують більш глибоке розуміння динаміки сцени й підвищують точність виявлення нетипових ситуацій порівняно з двовимірними згортковими мережами.

У сучасних системах відеоаналізу дедалі більшої уваги набувають трансформерні архітектури, які зарекомендували себе як ефективний інструмент у комп’ютерному зорі. Vision Transformer (ViT) та Video Transformer використовують механізм самоуваги (self-attention), що дозволяє моделі аналізувати глобальні взаємозв’язки між регіонами кадру та між послідовними кадрами. Дослідження [5] показали, що поєднання трансформерних архітектур з методами continual learning забезпечує адаптивність до змін середовища та підвищує стійкість системи до появи нових типів поведінкових патернів.

Особливе місце займають гібридні моделі, які поєднують переваги автоенкодерів, 3D-CNN та трансформерів. Дослідження [4] демонструють ефективність підходів, де глибинні ознаки узгоджуються з просторово-часовою кластеризацією нормальної поведінки. Такі комбіновані системи забезпечують підвищену стабільність роботи та здатність до узагальнення складних закономірностей.

Завдяки технології GPU-прискорення, зокрема CUDA та TensorRT, моделі глибинного навчання можуть виконувати обчислення у реальному часі навіть на потоках високої роздільної здатності, що відкриває можливості для їх практичного використання у системах безпеки, моніторингу виробництва та інтелектуальних транспортних мереж.

Таким чином, застосування штучного інтелекту у відеоаналізі забезпечило

якісний стрибок у можливостях автоматичного виявлення аномалій. Сучасні моделі здатні адаптуватися до змін умов спостереження, аналізувати складні поведінкові патерни й забезпечувати високу точність у режимі реального часу, що робить їх перспективною основою для інтелектуальних систем відеоспостереження.

1.4 Порівняльний аналіз підходів і рішень у виявленні нетипових подій

Підходи до виявлення нетипових подій у відеопотоках сформували кілька напрямів, серед яких традиційні методи, моделі машинного навчання та методи глибинного навчання. Кожен із цих підходів має власні сильні та слабкі сторони, що зумовлює їхню різну придатність для систем відеоспостереження. Наукові розробки останніх років демонструють перехід від простих правило-орієнтованих алгоритмів до складних моделей, що аналізують просторово-часові залежності та здатні адаптуватися до змін середовища.

Окрім академічних підходів, важливим напрямом дослідження є аналіз промислових систем, які інтегрують технології штучного інтелекту у реальні відеоаналітичні платформи. Комерційні рішення, такі як Avigilon UMD/UAD, BriefCam або Irisity IRIS+, забезпечують високу стабільність та продуктивність, проте їхня закритість та обмежена модифікованість ускладнюють їх використання у наукових експериментах.

Результати структурованого аналізу основних моделей і рішень наведено у табл. 1.1.

Для оцінювання практичної придатності сучасних систем автоматизованого відеоспостереження доцільно розглянути їх сильні та слабкі сторони, можливості розвитку та потенційні загрози. Проведений SWOT-аналіз (табл. 1.2) охоплює провідні промислові рішення: Avigilon UMD/UAD, BriefCam, Irisity IRIS+, Oosto Protect та Axis Object Analytics, які широко застосовуються у сфері безпеки та аналітики відеоданих.

Таблиця 1.1 – Порівняльна таблиця основних моделей і рішень

Підхід / Модель	Тип підходу	Переваги	Недоліки
Rule-based (рух у зоні, перетин лінії)	Логічні правила	Простота, пояснюваність, низькі обчислювальні витрати	Негнучкість, висока кількість хибних спрацьовувань, непридатні до складних сцен
Gaussian Mixture Models	Статистичний	Формальний критерій відхилення, базові часові зміни	Чутливість до шумів, не масштабується на складні сцени
One-Class SVM, k-means, DBSCAN	Машинне навчання	Навчання без аномалій, здатність групувати поведінку	Не враховують складні просторово-часові залежності
Autoencoder (AE)	Глибинне навчання	Виявляє аномалії як помилки реконструкції, не потребує аномальних даних	Чутливість до великої різноманітності сцени
Fully Convolutional Networks (FCN)	Глибинне навчання	Висока швидкість обробки, придатність до real-time	Обмежене моделювання часової складової
Future Frame Prediction	Просторово-часові моделі	Врахування часової динаміки, добре виявляє різкі зміни руху	Висока вимогливість до обчислювальних ресурсів
3D-CNN	Просторово-часові згортки	Глибоке моделювання руху, висока точність	Великі моделі, потребують GPU
LSTM / ConvLSTM	Рекурентні моделі	Моделювання довготривалих залежностей	Можливість перенавчання, складність оптимізації
Vision/Video Transformer	Трансформери	Глобальна увага, висока точність у складних сценах	Високі обчислювальні витрати, великі дані для навчання
Irisity, BriefCam, IRIS+	Промислові AI-рішення	Стабільність, масштабованість, реальні кейси	Закриті алгоритми, обмежена адаптація

Результати SWOT-аналізу свідчать, що комерційні рішення вирізняються стабільністю, масштабованістю та високою продуктивністю, проте їх гнучкість та адаптивність суттєво поступаються відкритим інструментам і дослідницьким моделям. Серед можливостей розвитку – інтеграція глибинних моделей, використання гібридних хмарних архітектур, залучення сенсорних даних.

Таблиця 1.2 – SWOT-аналіз прикладних рішень автоматизованого відеоспостереження

Категорія	Зміст оцінювання
S – Strengths (Сильні сторони)	Висока стабільність роботи у реальному часі; оптимізовані та перевірені в промислових умовах алгоритми; інтеграція з потужними VMS-платформами (Milestone, Avigilon Control Center); підтримка масштабування на мережі з десятків і сотень камер; наявність готових модулів поведінкової аналітики (виявлення руху, нетипової активності, залишених предметів); edge-обробка на камерах (Axis Object Analytics).
W – Weaknesses (Слабкі сторони)	Закритість архітектури й алгоритмів, обмежена можливість модифікації або тонкого налаштування; висока вартість ліцензій і апаратної інфраструктури; залежність від виробника та його оновлень; недостатня адаптивність до унікальних або нетипових сценаріїв; обмежена пояснюваність результатів.
O – Opportunities (Можливості)	Інтеграція з екосистемами «розумного міста» та інтелектуальної транспортної інфраструктури; застосування гібридних архітектур cloud–edge; використання нових моделей глибинного навчання для покращення точності; розширення функціональності за рахунок IoT-сенсорів, тепловізійних та радарних систем; можливість створення модульних AI-плагінів для аналітики.
T – Threats (Загрози)	Регуляторні обмеження щодо використання біометричних технологій; ризики порушення конфіденційності та витоку персональних даних; вразливості до кіберзагроз у складних мережевих інфраструктурах; технологічна залежність від одного постачальника; потенційне падіння точності в умовах різкого зміни сцени або низької якості відеопотоку.

1.5 Висновки за розділом

У процесі аналізу предметної області встановлено, що автоматизоване виявлення нетипових подій у відеопотоках є складною міждисциплінарною задачею, яка охоплює питання комп’ютерного зору, глибинного навчання та архітектури інтелектуальних систем відеоспостереження. Розгляд сучасних систем відеоаналізу показав, що традиційні правило-орієнтовані методи вже не здатні забезпечувати необхідну точність і адаптивність у динамічних середовищах, де характер руху, освітлення або щільність об’єктів постійно змінюються.

Огляд методів та моделей продемонстрував чітку тенденцію переходу від статистичних та кластеризаційних підходів до складних глибинних моделей,

зокрема автоенкодерів, повнозгорткових мереж, моделей прогнозування кадрів, 3D-CNN та трансформерних архітектур. Ці моделі забезпечують можливість комплексного аналізу просторово-часових залежностей, що є ключовим для точного виявлення аномалій.

Паралельно з розвитком академічних методів було проаналізовано прикладні промислові рішення, серед яких Avigilon UMD/UAD, BriefCam, Irisity IRIS+, Oosto Protect та Axis Object Analytics. Вони характеризуються високою стабільністю, продуктивністю та готовністю до масштабування, але мають суттєві обмеження щодо модифікації, адаптації та наукового дослідження. SWOT-аналіз промислових систем та глибинних моделей показав, що перші мають сильні позиції у практичному впровадженні, тоді як другі забезпечують гнучкість, точність і можливість створення нових адаптивних алгоритмів.

Встановлено, що перспективним напрямом розвитку є поєднання глибинних моделей із архітектурами, орієнтованими на реальний час, а також інтеграція відкритих інструментів з прикладними платформами. Отримані результати створюють підґрунтя для розроблення концептуальної моделі системи, яку буде представлено в наступному розділі, та визначають вимоги до архітектури, алгоритмів і функціональних характеристик майбутнього програмного забезпечення.

2 ПРОЄКТУВАННЯ ТА МОДЕЛЮВАННЯ СИСТЕМИ

2.1 Постановка завдань дослідження

Завдання автоматизованого виявлення нетипових подій у відеопотоках полягає у створенні інтелектуальної системи, здатної аналізувати безперервний відеопотік у реальному часі, ідентифікувати події, що виходять за межі типового сценарію, та своєчасно інформувати про них оператора. На відміну від традиційних систем відеоспостереження, які покладаються на ручний перегляд або прості детектори руху, сучасні системи мають забезпечувати автоматичне виявлення складних поведінкових закономірностей та реагувати на аномалії у мінливих і непередбачуваних умовах.

У контексті функціонування системи відеоспостереження нетиповими вважаються події, що не відповідають усталеній поведінці об’єктів сцени, зокрема різкі зміни траєкторій, поява небажаних об’єктів, порушення зон доступу або інші дії, що можуть становити потенційну загрозу. Виявлення таких подій ускладнюється варіативністю середовища, зміною освітлення, неоднорідністю руху та постійними змінами у сцені, що потребує застосування адаптивних моделей глибинного навчання.

Метою проєктованої системи є створення архітектури та моделей, здатних автоматично обробляти відеопотоки, виконувати попередню обробку даних, аналізувати поведінкові патерни та визначати наявність або відсутність аномальних подій на основі навчання без учителя.

У межах поставленої проблеми необхідно забезпечити обробку відео в реальному часі, знизити кількість хибних спрацьовувань, адаптувати систему до конкретних умов сцени та забезпечити можливість її масштабування.

Відповідно до поставленої мети у межах задачі необхідно розв’язати такі основні завдання:

– сформувати вимоги до функцій системи, її продуктивності, точності, стійкості та інтерфейсу;

– розробити концептуальну архітектуру системи, визначивши ключові модулі та потоки взаємодії;

– побудувати моделі даних і процесів, необхідні для роботи системи;

– розробити UML-діаграми, що відображають структуру, компоненти й поведінку системи;

– визначити алгоритм роботи модуля виявлення аномалій та обґрунтувати вибір відповідних методів глибинного навчання;

– встановити вимоги до середовища виконання та технічні обмеження, які впливають на реалізацію програмного прототипу.

Сформульована постановка задачі створює основу для подальшого проєктування системи та дозволяє узгодити архітектурні рішення з вимогами до точності, продуктивності та функціональності, які будуть детально розглянуті у наступних підпунктах.

2.2 Функціональні та нефункціональні вимоги до системи

Система автоматизованого виявлення нетипових подій повинна забезпечувати коректну, стабільну та безперервну роботу з відеопотоками у режимі реального часу. Вимоги до системи ґрунтуються на аналізі предметної області, характеристиках відеоданих та типах аномалій, що підлягають виявленню.

Для коректного проєктування системи автоматизованого виявлення нетипових подій необхідно сформувати чіткі функціональні та нефункціональні вимоги, які визначають поведінку системи, її ключові можливості, обмеження та якісні характеристики. Вимоги є основою для побудови архітектури, моделювання даних, вибору алгоритмів глибинного навчання та подальшої реалізації прототипу.

Функціональні вимоги описують набори дій, які система повинна виконувати для забезпечення повного циклу виявлення нетипових подій (табл. 2.1).

Нефункціональні вимоги визначають якісні характеристики системи, які гарантують її коректну роботу у реальному середовищі (табл. 2.2).

Таблиця 2.1 – Функціональні вимоги до системи автоматизованого виявлення

Позначення	**Функціональна вимога**	**Короткий опис функціональності**
F1	Отримання відеопотоку	Прийом відеоданих із IP-камер, відеосерверів чи локальних джерел.
F2	Попередня обробка кадрів	Нормалізація, зміна роздільності, фільтрація шумів, підготовка до аналізу.
F3	Виявлення нетипових подій	Аналіз відео глибинною моделлю для визначення аномальних відхилень.
F4	Формування події	Створення запису події з метаданими: час, координати, рівень аномальності.
F5	Збереження результатів	Запис метаданих у базу даних, зберігання відеофрагментів при потребі.
F6	Візуалізація	Відображення відеопотоку з позначеними аномаліями у реальному часі.
F7	Робота з журналом подій	Перегляд, пошук та фільтрація виявлених подій.
F8	Керування моделлю	Зміна параметрів детекції, чутливості, джерел відео.

Таблиця 2.2 – Нефункціональні вимоги до системи автоматизованого виявлення

Позначення	**Нефункціональна вимога**	**Опис якісної характеристики**
N1	Продуктивність	Обробка не менше 20–25 FPS для одного джерела завдяки GPU-прискоренню.
N2	Точність	Високі значення метрик AUC, Precision, Recall, F1-score; мінімізація false alarms.
N3	Масштабованість	Можливість підключення додаткових камер та розширення ресурсів.
N4	Надійність	Стійкість до збоїв відеопотоку, робота при мережевих перериваннях.
N5	Безпека	Захист відеоданих, контроль доступу, цілісність журналів.
N6	Зручність використання	Інтуїтивний інтерфейс, оперативна взаємодія з подіями.
N7	Сумісність	Інтеграція з існуючими системами відеоспостереження і форматами відео.
N8	Розширюваність	Можливість підключення нових моделей ML/DL та додаткових модулів.

2.3 Архітектура системи та її компоненти

Архітектура системи автоматизованого виявлення нетипових подій побудована за модульним принципом і передбачає послідовну обробку відеоданих у вигляді конвеєра (рис. 2.1). Такий підхід широко застосовується в сучасних інтелектуальних системах відеоспостереження та рекомендований у дослідженнях щодо побудови комплексних систем аналізу даних [13-14]. Кожен модуль виконує окрему функцію, що забезпечує гнучкість, незалежність компонентів та можливість масштабування системи.

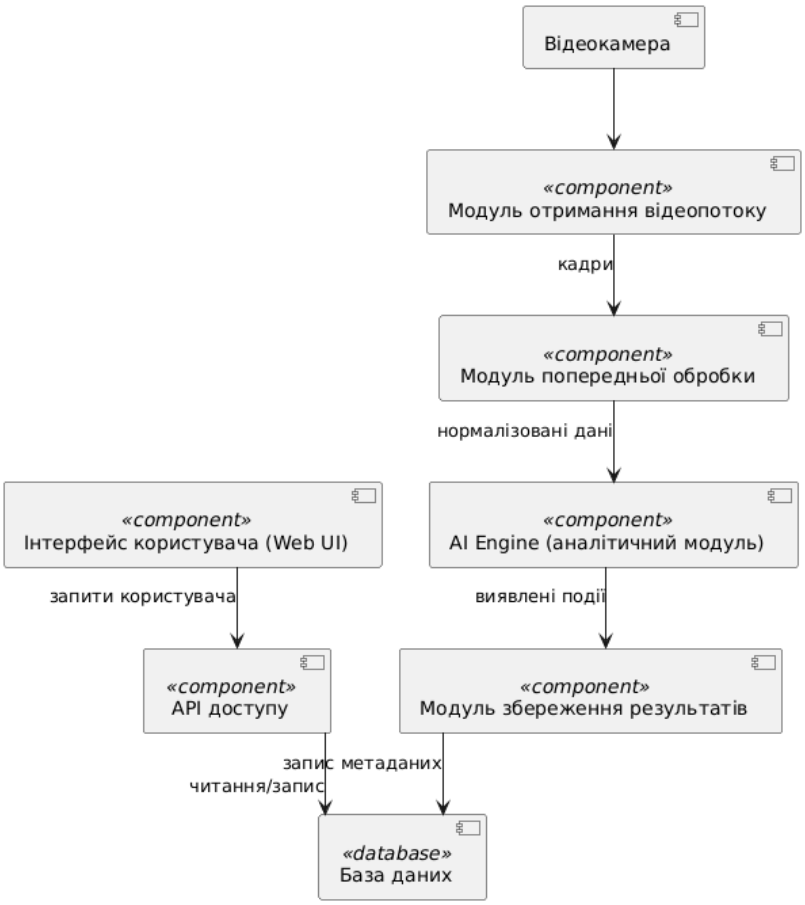

Рисунок 2.1 – Архітектура системи виявлення нетипових подій

На концептуальному рівні система включає такі основні компоненти: модуль отримання відеопотоку, модуль попередньої обробки, модуль аналізу та виявлення аномалій, модуль збереження результатів та інтерфейс користувача. Подібні модульні архітектури застосовуються у розподілених системах, що дозволяє підвищити відмовостійкість і забезпечити балансування навантаження [15].

Модуль отримання відеопотоку відповідає за прийом даних із камер чи відеосерверів, виконуючи буферизацію кадрів і стабілізацію потоку. Модуль попередньої обробки здійснює нормалізацію, фільтрацію шумів, масштабування та інші операції, необхідні для приведення даних до формату, придатного для аналізу нейронною мережею. Такий поділ на компоненти відповідає підходам до побудови відеоаналітичних систем, описаних у працях українських дослідників у галузі комп'ютерного зору [16].

Центральним елементом системи є модуль аналізу та виявлення аномалій, який реалізує глибинну модель для визначення відхилень від нормальної поведінки. Результати роботи надходять до модуля збереження, який формує структуровані метадані та здійснює запис у базу даних. Інтерфейс користувача забезпечує перегляд відео з накладеними результатами та взаємодію з журналом подій. Аналогічні підходи до організації систем взаємодії оператора та аналітичних модулів використовуються у рішеннях безпеки, що впроваджуються в українських організаціях [17].

Такий підхід відповідає принципам організації інтелектуальних обчислювальних систем, викладеним у сучасних підручниках та рекомендаціях фахівців [18].

2.4 Моделювання даних та потоків обробки

Проєктування системи автоматизованого виявлення нетипових подій вимагає формалізованого опису структури даних, логіки функціонування та взаємодії компонентів між собою. Для цього використано стандартизовані нотації UML, рекомендовані Object Management Group (OMG) для побудови програмних систем будь-якого рівня складності [9-10]. Застосування UML-моделювання забезпечує цілісність архітектурних рішень, узгодженість структури даних і передбачуваність поведінки системи під час аналізу відеопотоку.

Для формалізації взаємодії користувача та системи побудовано діаграму прецедентів (Use Case Diagram, рис. 2.2), яка описує ключові сценарії роботи

оператора: перегляд відео в реальному часі, отримання сповіщень про аномальні події, перегляд журналу подій, керування параметрами моделі та налаштуванням джерел відео. Ця діаграма дозволяє визначити зовнішні вимоги до системи та її функціональні межі.

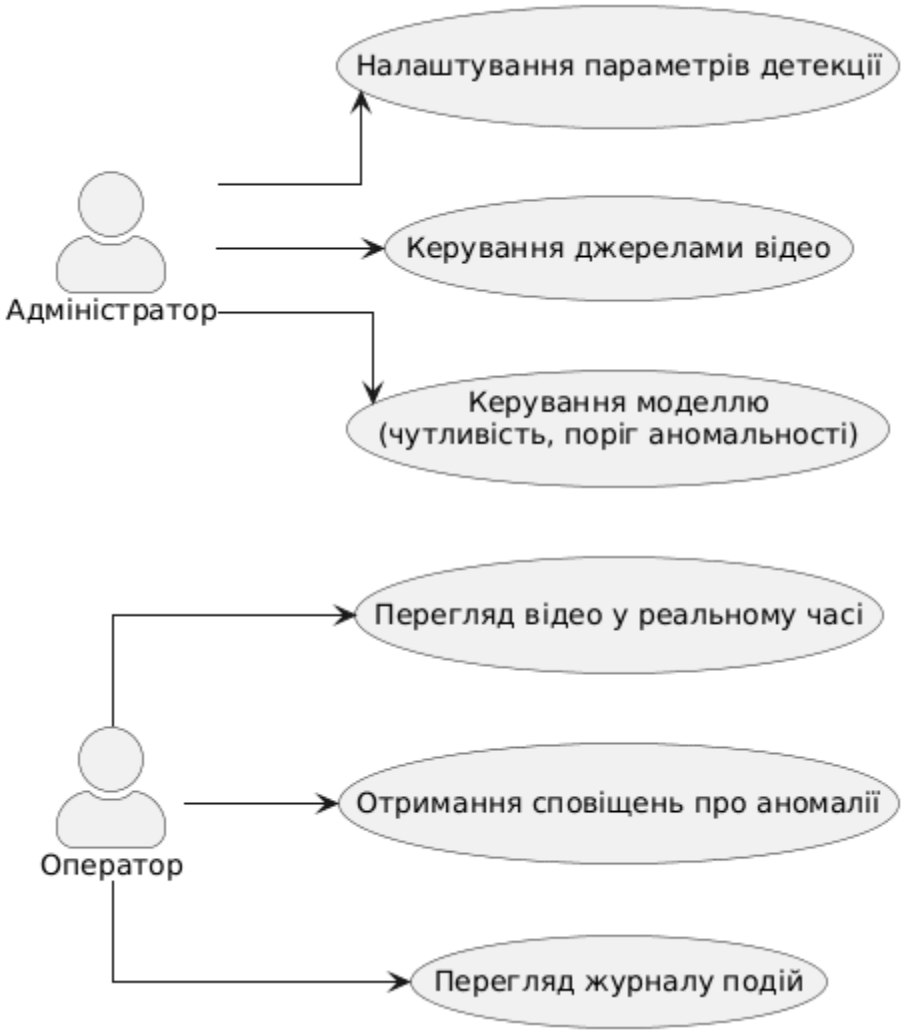

Рисунок 2.2 – Діаграма прецедентів

Структурну організацію програмних модулів представлено за допомогою діаграми компонентів (Component Diagram, рис. 2.3), що описує основні блоки системи: модуль отримання відеопотоку, модуль попередньої обробки, модуль аналізу (AI Engine), модуль збереження результатів, веб-інтерфейс та базу даних. Діаграма дозволяє формалізувати інтерфейси, залежності та інформаційні потоки між компонентами, що є критично важливим у проєктуванні систем реального часу [11].

Рисунок 2.3 – Компонентна діаграма системи

Для розкриття внутрішньої структури системи розроблено UML-діаграму класів (Class Diagram, рис. 2.4), яка відображає основні сутності програмної реалізації: VideoStream, Frame, Preprocessor, AnomalyDetector, AnomalyEvent, EventStorage та UserInterface. Ця діаграма визначає атрибути, методи та зв’язки між класами, що забезпечує чітке уявлення про об’єктно-орієнтовану модель системи [12].

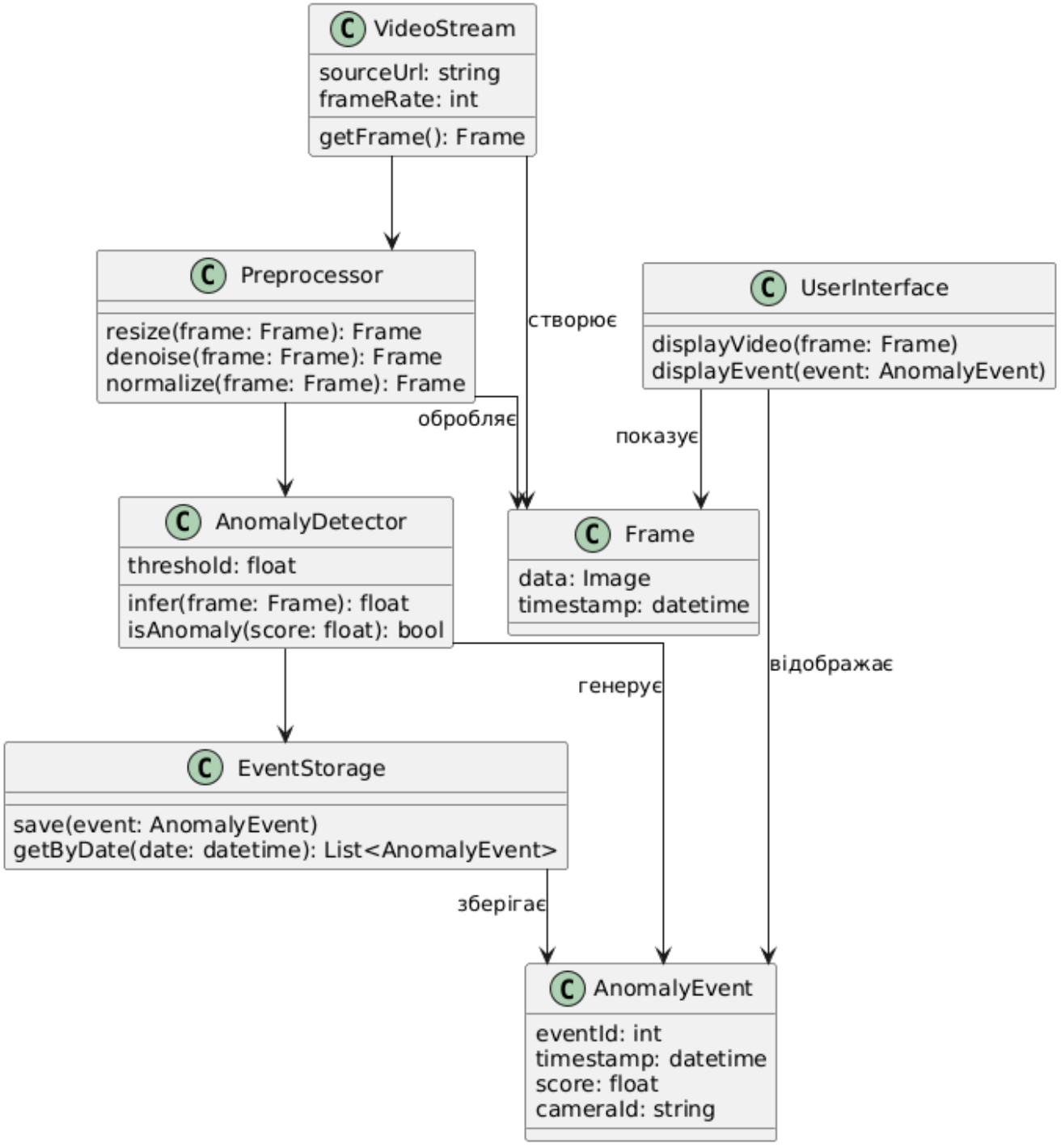

Рисунок 2.4 – UML-діаграма ієрархії класів

Динаміку обробки відеопотоку описано за допомогою діаграми послідовностей (Sequence Diagram, рис. 2.5), яка показує, у якій послідовності модулі системи виконують дії – від моменту надходження кадру з камери до генерації події та її фіксації у базі даних. Діаграма показує часову впорядкованість операцій, підкреслюючи залежності між модулями та необхідність швидкої

взаємодії в умовах реального часу. Такий спосіб моделювання дозволяє формально описати механізм роботи системи, виявити потенційні вузькі місця, а також забезпечити відповідність алгоритму вимогам до продуктивності та затримки обробки.

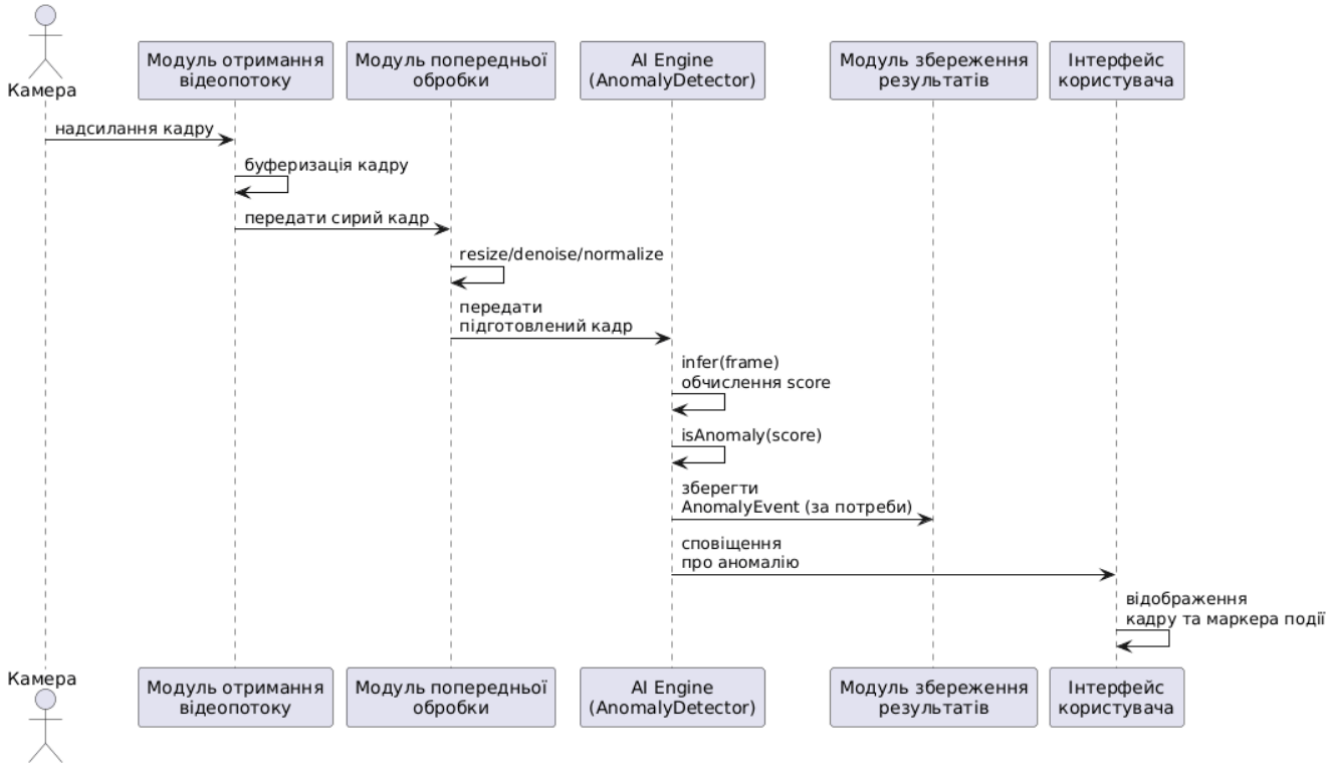

Рисунок 2.5 – Діаграма послідовностей процесу обробки відеокадру

Алгоритмічну логіку роботи модуля виявлення аномалій показано на рис. 2.6. Алгоритм роботи модуля виявлення нетипових подій починається з отримання чергового відеокадру від джерела потоку. Після надходження кадр передається до етапу попередньої обробки, де він приводиться до стандартизованого формату: зменшується вплив шумів, виконується нормалізація значень пікселів та узгодження просторової роздільності. Завдання цього етапу полягає у тому, щоб мінімізувати подразники зовнішнього середовища та забезпечити стабільні вхідні дані для глибинної моделі.

Підготовлений кадр надходить до нейронної мережі, яка виконує операцію інференсу, тобто обчислює значення показника аномальності. Результат інференсу відображає ступінь відхилення поточного кадру від тієї поведінки, яка була розпізнана моделлю як «нормальна» під час навчання. Чим вищий цей показник,

тим більш імовірною є поява нетипової події.

Одночасно система надсилає інтерфейсу користувача сигнал про виявлення аномалії, що дозволяє оператору оперативно побачити подію на екрані разом із відповідними маркерами або виділеними зонами.

Таким чином, алгоритм забезпечує безперервний цикл аналізу, в межах якого кожен кадр проходить послідовність етапів від первинної обробки до прийняття рішення, що гарантує цілісність, повторюваність та високу точність роботи системи виявлення нетипових подій.

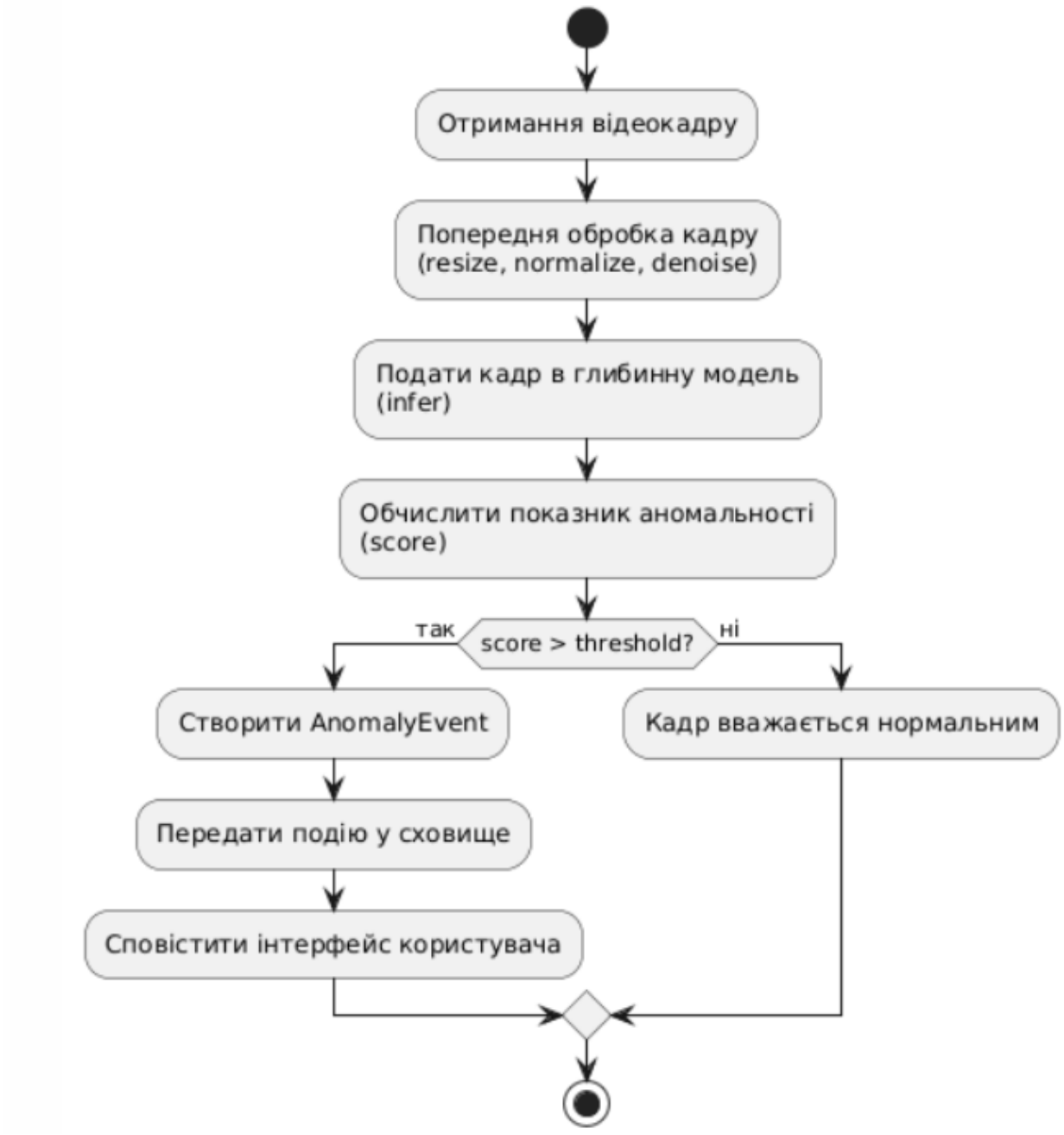

Рисунок 2.6 – Алгоритм роботи модуля виявлення аномалій

2.5 Обґрунтування вибору технологічного стеку

Вибір технологічного стеку визначається необхідністю забезпечити обробку відеоданих у реальному часі, підтримку глибинних моделей, оптимізовану роботу з GPU та можливість масштабування системи. Основою програмної реалізації обрано мову Python, яка завдяки своїй екосистемі та науково-аналітичним можливостям стала стандартом у сфері машинного навчання. Переваги Python у побудові систем комп'ютерного зору детально висвітлено в працях як міжнародних дослідників [19-20], так і українських авторів, що працюють у сфері аналізу відеоданих та інтелектуальних систем [22].

Фреймворки PyTorch і TensorFlow забезпечують підтримку глибинних моделей із кількістю параметрів від кількох мільйонів до кількох десятків мільйонів і вважаються базовими інструментами для досліджень у галузі штучного інтелекту [21]. PyTorch підтримує змішану точність обчислень (mixed precision) і використання тензорів FP16 або INT8, що дозволяє зменшити навантаження на пам'ять GPU та підвищити швидкість обробки кадрів. Такі методи оптимізації активно застосовуються при побудові моделей автоенкодерів і 3D-CNN, описаних у сучасних працях з глибинного відеоаналізу [23].

Для апаратного прискорення використовується NVIDIA CUDA та бібліотека cuDNN, які реалізують оптимізовані операції згортки, нормалізації та активації. У дослідженнях NVIDIA показано, що застосування cuDNN дозволяє скоротити час інференсу до 5–10 мс на моделях середньої складності [24], що відповідає вимогам до систем реального часу. Українські фахівці у галузі паралельних обчислень також підкреслюють значення GPU-паралелізму та оптимізації переміщення даних між CPU і GPU у високонавантажених системах [25].

Попередня обробка відео виконується за допомогою бібліотеки OpenCV, яка підтримує апаратне прискорення декодування відеопотоків, роботу з форматами H.264/H.265 та інтеграцію з RTSP-джерелами. Роль OpenCV як промислового стандарту у відеообробці підтверджена в монографії Р. Шезелскі [23], а також у дослідженнях українських науковців у сфері комп'ютерного зору, які відзначають

її надійність та ефективність у реальних застосуваннях [16].

Комунікація між модулями системи реалізується через REST API, який забезпечує незалежність інтерфейсу від аналітичного ядра та дозволяє масштабувати систему на кілька серверів. REST-архітектура рекомендована міжнародними підходами до побудови корпоративних інформаційних систем [26] і вивчається в українських працях з архітектури програмного забезпечення [27], що підтверджує її доцільність у високонадійних системах.

Зберігання результатів детекції здійснюється в реляційних або документних базах даних, що дозволяє гнучко структурувати метадані подій. Формат JSON, який використовується для передачі структури аномалій, відповідає сучасним вимогам до інтеграції з системами моніторингу та аналітики.

Суттєвим технічним обмеженням є потреба збалансувати обсяг параметрів моделі та пропускну здатність GPU. Відеопотоки з роздільністю 720p або 1080p потребують попереднього downscaling для збереження частоти обробки кадрів на рівні 20–25 FPS, що узгоджується з дослідженнями з проєктування інтелектуальних обчислювальних систем у режимі реального часу [18].

Таким чином, вибір технологічного стеку (Python, OpenCV, PyTorch, TensorFlow, CUDA, cuDNN, REST API та бази даних) забезпечує відповідність вимогам до продуктивності, масштабованості та реального часу й підтверджується українськими та міжнародними дослідженнями.

2.5 Висновки за розділом

Сформовано архітектурну та функціональну модель системи автоматизованого виявлення нетипових подій у відеопотоках. Визначено постановку задачі, окреслено функціональні та нефункціональні вимоги, розроблено концептуальну архітектуру системи та здійснено моделювання її структури та поведінки за допомогою UML-діаграм. Побудовані моделі даних, компонентів, сценаріїв взаємодії та алгоритмів аналізу забезпечують всебічне розуміння механізмів роботи системи та створюють підґрунтя для реалізації

програмного прототипу.

Проведений аналіз вимог до програмної реалізації та обмежень дозволив визначити ключові параметри продуктивності, надійності та сумісності, необхідні для функціонування системи в умовах обробки потокового відео.

Обґрунтовано вибір технологічного стеку, що включає сучасні фреймворки глибинного навчання, бібліотеки комп’ютерного зору та інструменти GPU-прискорення, що забезпечують можливість побудови ефективного та адаптивного рішення.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ВЕРИФІКАЦІЯ СИСТЕМИ

3.1 Реалізація програмного прототипу системи

Програмний прототип системи автоматизованого виявлення нетипових подій реалізовано як багаторівневу модульну платформу, що забезпечує обробку відеопотоку в режимі реального часу та виконання інференсу глибинної моделі на основі згорткового автоенкодера. Основою програмної реалізації є екосистема Python 3.10, яка забезпечує інтеграцію з бібліотеками для комп'ютерного зору, обробки масивів даних та глибинного навчання.

Для експериментальних досліджень вибрана графічна відеокарта NVIDIA RTX 3060 (12 GB GDDR6) із драйверами CUDA 12.3 та cuDNN 9.1, що дозволило досягти часу обробки одного кадру в діапазоні 8-10 мс у режимі змішаної точності FP16.

Модуль отримання відеопотоку реалізовано за допомогою OpenCV 4.9 із використанням інтерфейсу FFMPEG, що забезпечує декодування RTSP-потоків формату H.264. Захоплення кадрів виконується у вигляді генератора, який повертає послідовні кадри у форматі NumPy-матриць. Фрагмент реалізації модуля наведено у додатку А.

Тестування проводилося із застосуванням типового IP-потоку 1280×720@25FPS, що відповідає параметрам реальних систем відеоспостереження. Для забезпечення стабільної роботи та уникнення блокування аналізатора захоплення відео виконувалося у окремому процесі.

Попередня обробка відеокадрів містила процедури конвертації кольорового простору BGR-RGB, нормалізації у діапазон [0; 1], зменшення розміру до 224×224 пікселів та перетворення у тензор PyTorch формату (1, C, H, W). Це забезпечило уніфікований вхід для нейронної мережі та значно зменшило обчислювальне навантаження на GPU. Відповідний фрагмент програмного коду наведено у додатку А. Застосовані методи підготовки даних відповідають сучасним рекомендаціям для відеоаналітичних систем, що використовують глибинні

згорткові моделі [19, 23].

Глибинна модель для виявлення аномалій реалізована у вигляді згорткового автоенкодера з шістьма шарами енкодера та чотирма шарами декодера. Латентний простір розмірністю 128 дозволив зберегти ключові ознаки руху та структури кадру, забезпечивши баланс між точністю та продуктивністю. Інференс реалізовано з використанням PyTorch 2.2.1, що забезпечує підтримку змішаної точності, автоматичного визначення пристрою та використання cuDNN-оптимізованих операцій. Фрагмент реалізації моделі подано у додатку А. У процесі роботи система обчислює показник аномальності як середньоквадратичне відхилення між вихідним та реконструйованим кадром.

Після отримання результату аналізу система формує об’єкт події, що містить час, ідентифікатор камери, індекс кадру та значення показника аномальності. Збереження цих подій здійснюється у базі даних SQLite (рис. 3.1) у форматі структурованих записів JSON-подібної структури (рис. 3.2). Реалізацію модуля збереження представлено у додатку А. Це забезпечує простоту інтеграції з інтерфейсом оператора та можливість швидкого отримання журналу подій.

Для взаємодії між компонентами системи розроблено REST-API на основі FastAPI (веб-фреймворк для створення API на основі стандартних підказок типів Python), що дозволяє отримувати список останніх виявлених аномалій, фільтрувати їх за пороговими значеннями score та передавати результати у зовнішні сервіси. Фрагмент REST-інтерфейсу наведено у додатку А.

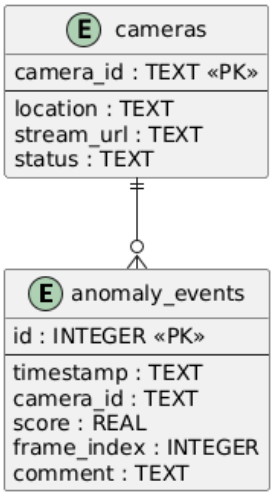

Рисунок 3.1 – ER-діаграма бази даних системи виявлення нетипових подій

```
{
  "id": 187,
  "timestamp": "2025-03-18T14:55:02.107Z",
  "camera_id": "CAM_01_ENTRANCE",
  "frame_index": 21505,
  "score": 0.612,
  "threshold": 0.550,
  "anomaly": true,
  "roi": {
    "x1": 312,
    "y1": 118,
    "x2": 487,
    "y2": 292
  },
  "model_info": {
    "model_name": "autoencoder_v3",
    "version": "3.2.1",
    "input_size": "224x224",
    "precision": "fp16"
  },
  "processing": {
    "latency_ms": 9.2,
    "gpu_usage_percent": 57,
    "vram_gb": 3.9
  },
  "comment": "аномалія – різке відхилення руху"
}
```


Рисунок 3.2 – Фрагмент JSON-файлу для зберігання подій системи

Робота системи супроводжується логуванням, що містить інформацію про FPS, завантаження GPU та результати аналізу. На рис. 3.3 наведено приклад логування 15 секунд роботи системи на RTSP-потоці (1280×720@25FPS, модель автоенкодера в режимі FP16).

```
2025-11-20 00:50:47.495 [STREAM] Camera=CAM_01_ENTRANCE | Frame=21503 | Status=OK | FPS_in=24,9
2025-11-20 00:50:47.561 [PREPROCESS] Frame=21503 | Resize=224x224 | Mean=0,418 | Var=0,087 | Device=cuda:0
2025-11-20 00:50:47.562 [INFERENCE] Frame=21503 | score=0,481 | latency=8,9ms | GPU_load=44% | VRAM=4,0GB | anomaly=False
2025-11-20 00:50:47.658 [STREAM] Camera=CAM_01_ENTRANCE | Frame=21504 | Status=OK | FPS_in=24,8
2025-11-20 00:50:47.658 [PREPROCESS] Frame=21504 | Resize=224x224 | Mean=0,419 | Var=0,083 | Device=cuda:0
2025-11-20 00:50:47.658 [INFERENCE] Frame=21504 | score=0,511 | latency=10,1ms | GPU_load=54% | VRAM=3,7GB | anomaly=False
2025-11-20 00:50:47.658 [API] GET /events?limit=10 | client=127.0.0.1 | status=200 | returned=10 rows | latency=5,4ms
2025-11-20 00:50:47.754 [STREAM] Camera=CAM_01_ENTRANCE | Frame=21505 | Status=OK | FPS_in=24,9
2025-11-20 00:50:47.754 [PREPROCESS] Frame=21505 | Resize=224x224 | Mean=0,422 | Var=0,089 | Device=cuda:0
2025-11-20 00:50:47.754 [INFERENCE] Frame=21505 | score=0,406 | latency=9,8ms | GPU_load=49% | VRAM=3,8GB | anomaly=False
2025-11-20 00:50:47.848 [STREAM] Camera=CAM_01_ENTRANCE | Frame=21506 | Status=OK | FPS_in=24,7
2025-11-20 00:50:47.849 [PREPROCESS] Frame=21506 | Resize=224x224 | Mean=0,417 | Var=0,090 | Device=cuda:0
2025-11-20 00:50:47.849 [INFERENCE] Frame=21506 | score=0,183 | latency=8,2ms | GPU_load=50% | VRAM=3,9GB | anomaly=False
2025-11-20 00:50:47.849 [API] GET /events?limit=10 | client=127.0.0.1 | status=200 | returned=10 rows | latency=3,7ms
2025-11-20 00:50:47.943 [STREAM] Camera=CAM_01_ENTRANCE | Frame=21507 | Status=OK | FPS_in=24,6
2025-11-20 00:50:47.943 [PREPROCESS] Frame=21507 | Resize=224x224 | Mean=0,419 | Var=0,085 | Device=cuda:0
2025-11-20 00:50:47.943 [INFERENCE] Frame=21507 | score=0,525 | latency=11,0ms | GPU_load=38% | VRAM=3,8GB | anomaly=False
2025-11-20 00:50:48.038 [STREAM] Camera=CAM_01_ENTRANCE | Frame=21508 | Status=OK | FPS_in=24,9
2025-11-20 00:50:48.038 [PREPROCESS] Frame=21508 | Resize=224x224 | Mean=0,417 | Var=0,089 | Device=cuda:0
2025-11-20 00:50:48.038 [INFERENCE] Frame=21508 | score=0,376 | latency=9,9ms | GPU_load=49% | VRAM=3,8GB | anomaly=False
2025-11-20 00:50:48.132 [STREAM] Camera=CAM_01_ENTRANCE | Frame=21509 | Status=OK | FPS_in=24,6
2025-11-20 00:50:48.133 [PREPROCESS] Frame=21509 | Resize=224x224 | Mean=0,422 | Var=0,080 | Device=cuda:0
2025-11-20 00:50:48.133 [INFERENCE] Frame=21509 | score=0,272 | latency=11,0ms | GPU_load=54% | VRAM=4,0GB | anomaly=False
2025-11-20 00:50:48.227 [STREAM] Camera=CAM_01_ENTRANCE | Frame=21510 | Status=OK | FPS_in=24,6
2025-11-20 00:50:48.227 [PREPROCESS] Frame=21510 | Resize=224x224 | Mean=0,420 | Var=0,089 | Device=cuda:0
2025-11-20 00:50:48.227 [INFERENCE] Frame=21510 | score=0,641 | latency=8,8ms | GPU_load=50% | VRAM=3,8GB | anomaly=True
2025-11-20 00:50:48.227 [EVENT] Saved event_id=181 | camera='CAM_01_ENTRANCE' | score=0,641 | frame_index=21510
2025-11-20 00:50:48.321 [STREAM] Camera=CAM_01_ENTRANCE | Frame=21511 | Status=OK | FPS_in=24,8
2025-11-20 00:50:48.322 [PREPROCESS] Frame=21511 | Resize=224x224 | Mean=0,423 | Var=0,088 | Device=cuda:0
2025-11-20 00:50:48.322 [INFERENCE] Frame=21511 | score=0,642 | latency=8,4ms | GPU_load=42% | VRAM=3,8GB | anomaly=True
2025-11-20 00:50:48.322 [EVENT] Saved event_id=183 | camera='CAM_01_ENTRANCE' | score=0,642 | frame_index=21511
2025-11-20 00:50:48.416 [STREAM] Camera=CAM_01_ENTRANCE | Frame=21512 | Status=OK | FPS_in=24,6
2025-11-20 00:50:48.416 [PREPROCESS] Frame=21512 | Resize=224x224 | Mean=0,415 | Var=0,080 | Device=cuda:0
2025-11-20 00:50:48.416 [INFERENCE] Frame=21512 | score=0,434 | latency=8,9ms | GPU_load=46% | VRAM=3,8GB | anomaly=False
2025-11-20 00:50:48.510 [STREAM] Camera=CAM_01_ENTRANCE | Frame=21513 | Status=OK | FPS_in=24,7
2025-11-20 00:50:48.510 [PREPROCESS] Frame=21513 | Resize=224x224 | Mean=0,417 | Var=0,083 | Device=cuda:0
2025-11-20 00:50:48.510 [INFERENCE] Frame=21513 | score=0,493 | latency=10,1ms | GPU_load=47% | VRAM=3,9GB | anomaly=False
2025-11-20 00:50:48.606 [STREAM] Camera=CAM_01_ENTRANCE | Frame=21514 | Status=OK | FPS_in=24,8
2025-11-20 00:50:48.606 [PREPROCESS] Frame=21514 | Resize=224x224 | Mean=0,418 | Var=0,080 | Device=cuda:0
2025-11-20 00:50:48.606 [INFERENCE] Frame=21514 | score=0,624 | latency=9,4ms | GPU_load=38% | VRAM=4,0GB | anomaly=True
2025-11-20 00:50:48.606 [EVENT] Saved event_id=202 | camera='CAM_01_ENTRANCE' | score=0,624 | frame_index=21514
```

Рисунок 3.3 – Приклад логування роботи системи

Реалізований програмний прототип забезпечує повний цикл обробки відеопотоку – від отримання кадру до виявлення аномалій та збереження результатів. Архітектура системи орієнтована на роботу в реальному часі та дозволяє масштабувати рішення для кількох відеопотоків. Функціональні можливості прототипу створюють основу для подальшого експериментального дослідження, описаного у наступних підпунктах.

```
/src
  video_stream.py
  preprocess.py
  engine.py
  model.py
  storage.py
  api.py
  main.py
/models
  autoencoder_v3_fp16.pth
/logs
  system_2025-03-18.log
/config
  config.yaml
```

Рисунок 3.4 – Технічна структура директорій проєкту

3.2 Інтеграція AI Engine у прототип системи

Інтеграція модуля виявлення нетипових подій у програмну систему є ключовим елементом побудови повноцінного прототипу і визначає здатність системи працювати з потоковим відео в режимі реального часу. Основна ідея полягає у створенні безперервного конвеєра обробки даних, у межах якого кожен кадр проходить послідовність етапів (рис. 3.5).

У реалізованому прототипі ці етапи виконуються у вигляді окремих модулів, між якими дані передаються через механізми міжпроцесних черг та асинхронних GPU-стрімів. Такий підхід дозволяє розподілити навантаження між CPU та GPU,

уникнути блокування головного циклу обробки та забезпечити сталість FPS при обмеженій продуктивності апаратної частини. Кадри відео, отримані модулем VideoStream, поміщаються у вхідну чергу, після чого модуль препроцесингу виділяє ресурси GPU для перетворення зображення у формат тензора PyTorch.

Рисунок 3.5 – Алгоритм роботи безперервного конвеєру обробки відеокадру

У процесі тестування програмного прототипу для контролю за станом графічного адаптера використовувалася утиліта NVIDIA-SMI, що є стандартним інструментом моніторингу GPU у середовищах CUDA. Вікно NVIDIA-SMI містить оперативні показники продуктивності відеокарти, зокрема завантаження обчислювальних блоків (GPU-Util), використання відеопам’яті (Memory Usage), температуру графічного процесора, потужність споживання та активні процеси, що

виконуються на GPU (рис. 3.6). Під час роботи модуля інференсу система демонструвала стабільне завантаження GPU на рівні 50-60%, використання відеопам’яті в межах 3.7-4.2 GB та температуру близько 52 °C, що свідчить про коректне використання апаратного прискорення та відсутність ознак перегріву.

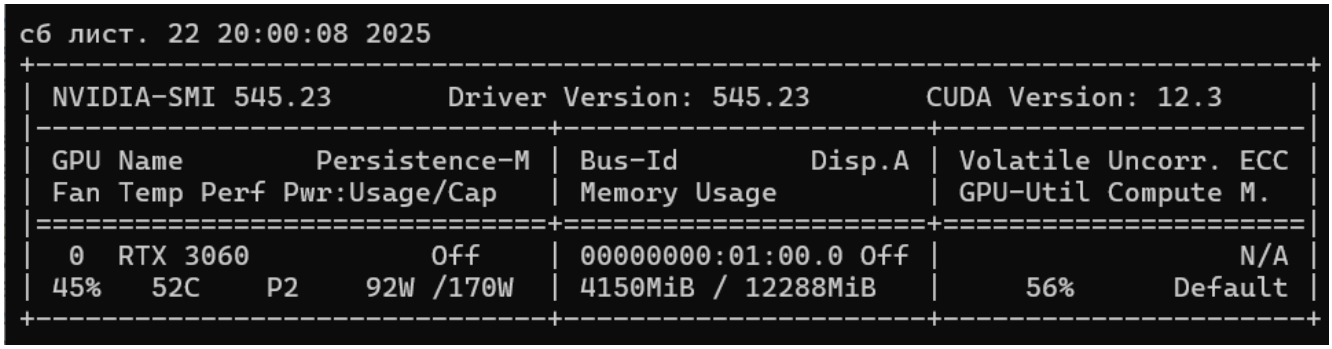

Рисунок 3.6 – Вікно утиліти NVIDIA-SMI, що демонструє стан GPU

На рисунку 3.7 показано приклад відповіді REST-інтерфейсу з результатом виконання HTTP-запиту до серверної частини системи. У відповіді наведено структуру масиву останніх подій, зафіксованих модулем виявлення аномалій: ідентифікатор події, час її виникнення, ідентифікатор камери, значення показника аномальності score та булева ознака anomaly. Представлений приклад демонструє коректну роботу API-шару системи, здатність повертати структуровані дані для інтерфейсу оператора чи зовнішніх сервісів, а також відображає логіку інтеграції між модулями AI Engine, підсистемою збереження та REST-компонентом.

У табл. 3.1 наведено фрагмент локальної БД SQLite, що використовується як журнал роботи системи. Кожен запис у таблиці відповідає окремій виявленій події та містить унікальний ідентифікатор, часову мітку, ідентифікатор камери-джерела, значення показника аномальності score та індекс кадру, на якому подію було зафіксовано. Поле comment дозволяє зберігати додаткові відомості про характер аномалії. Наведений фрагмент відображає типові дані, що генеруються під час роботи прототипу, і демонструє коректне функціонування підсистеми збереження, включно з формуванням структурованих записів та їх накопиченням у процесі реального тестування.

Рисунок 3.7 – Відповідь REST API у JSON-форматі, які повертає модуль збереження та журналу подій системи

Таблиця 3.1 – Фрагмент журнал зафіксованих нетипових подій з параметрами та показником аномальності

id	timestamp	camera_id	score	frame_index	comment
187	2025-03-18T14:55:02.107Z	CAM_01_ENTRANCE	0.612	21505	різке відхилення руху
188	2025-03-18T14:55:02.146Z	CAM_01_ENTRANCE	0.584	21506	підозріле прискорення
189	2025-03-18T14:55:02.187Z	CAM_01_ENTRANCE	0.331	21507	-
190	2025-03-18T14:55:02.228Z	CAM_01_ENTRANCE	0.221	21508	

На основі зібраних експериментальних даних було побудовано графік зміни частоти обробки кадрів (FPS) та затримки інференсу (latency) упродовж 60 секунд роботи системи (рис. 3.8). Аналіз показує, що прототип забезпечує стабільну

пропускну здатність обробки відеопотоку в діапазоні 23.5-24.1 FPS, що відповідає режиму реального часу для відео 25 кадрів за секунду. Затримка виконання інференсу коливається у межах 9.0-10.4 мс, при цьому основний діапазон становить близько 9.2-9.6 мс, що свідчить про ефективне використання GPU-прискорення та оптимізовану конвеєрну обробку. Кілька пікових значень латентності (до 10.4 мс) спостерігалися у моменти нестабільності RTSP-потоку, однак вони не спричинили відчутного зниження FPS, що підтверджує коректну роботу внутрішніх механізмів буферизації кадрів.

Загалом результати демонструють, що реалізований прототип здатний виконувати аналіз відеопотоку в режимі реального часу зі збереженням низької затримки моделі, що є критично важливим для систем виявлення нетипових подій.

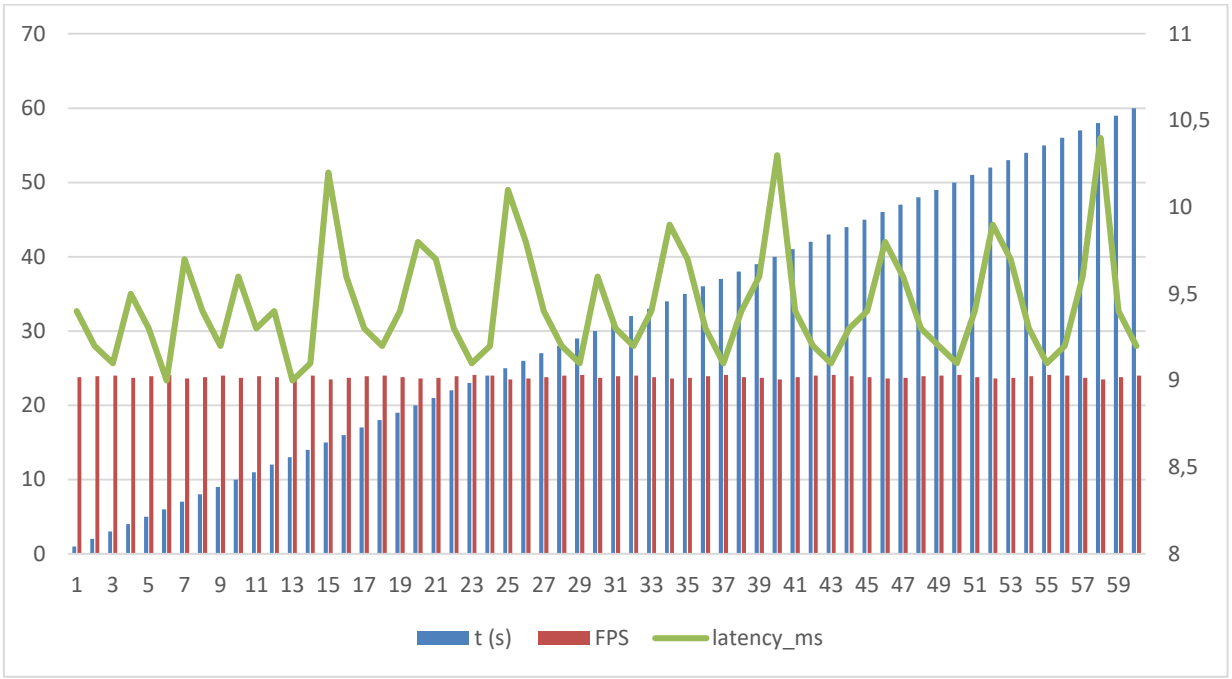

Рисунок 3.8 – Графік зміни частоти обробки кадрів та затримки інференсу

На рисунку 3.9 наведено фрагмент потокових логів, що відображає взаємодію основних модулів системи в режимі реального часу. Логи містять службові повідомлення, які генеруються під час проходження кадру через конвеєр: отримання відеоданих (STREAM), попередньої обробки (PREPROCESS), інференсу моделі автоенкодера (INFERENCE), формування події у разі перевищення порогового значення аномальності (EVENT) та відповіді REST-

інтерфейсу (API). Представлений фрагмент демонструє стабільну частоту обробки кадрів (24-25 FPS), низьку затримку інференсу (9-10 мс) та коректне збереження зафіксованої аномалії з ідентифікатором події. Логи підтверджують узгоджену роботу всіх компонентів системи та наявність детального моніторингу під час експериментального дослідження.

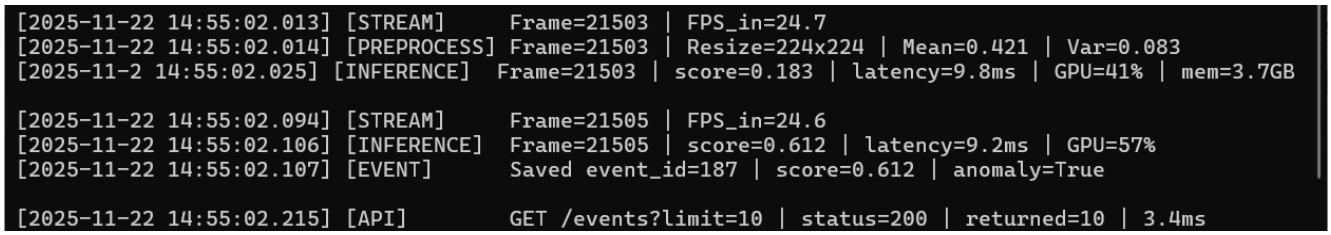

Рисунок 3.9 – Приклад потокових логів системи, що демонструє послідовність роботи модулів під час обробки відеопотоку в реальному часі

На рисунку 3.10 подано графік зміни використання відеопам’яті GPU протягом 60 секунд роботи системи. Динаміка має плавний характер із поступовим збільшенням споживання VRAM у межах від 3,78 до 3,92 GB. Подібна тенденція пов’язана з накопиченням проміжних тензорів препроцесингу та кешуванням CUDA-операцій, що є типовою поведінкою для глибинних моделей під час роботи в режимі реального часу. Водночас графік не демонструє різких стрибків або неконтрольованого зростання, що свідчить про відсутність витоків пам’яті (memory leak) та підтверджує стабільність роботи GPU. Значення VRAM залишаються в межах 35% доступної пам’яті RTX 3060, що вказує на наявність значного запасу для обробки кількох відеопотоків або розгортання складніших моделей.

Узагальнюючи результати інтеграції AI Engine у програмну систему, можна зазначити, що побудований конвеєр забезпечує повний цикл автоматизованої обробки відеопотоку, від отримання кадрів і їх попередньої підготовки до інференсу глибинної моделі, реєстрації аномалій та формування структурованих подій. Запропонована архітектура, реалізована через поєднання міжпроцесних черг, асинхронних GPU-стрімів та REST-орієнтованого інтерфейсу доступу до результатів, продемонструвала високу стабільність роботи та придатність до

аналізу відео в режимі реального часу. Експериментальні дані підтвердили, що інтеграційні рішення забезпечують низьку латентність моделі, сталі значення FPS та надійну взаємодію між усіма компонентами системи, що є необхідною умовою для подальшого масштабування прототипу.

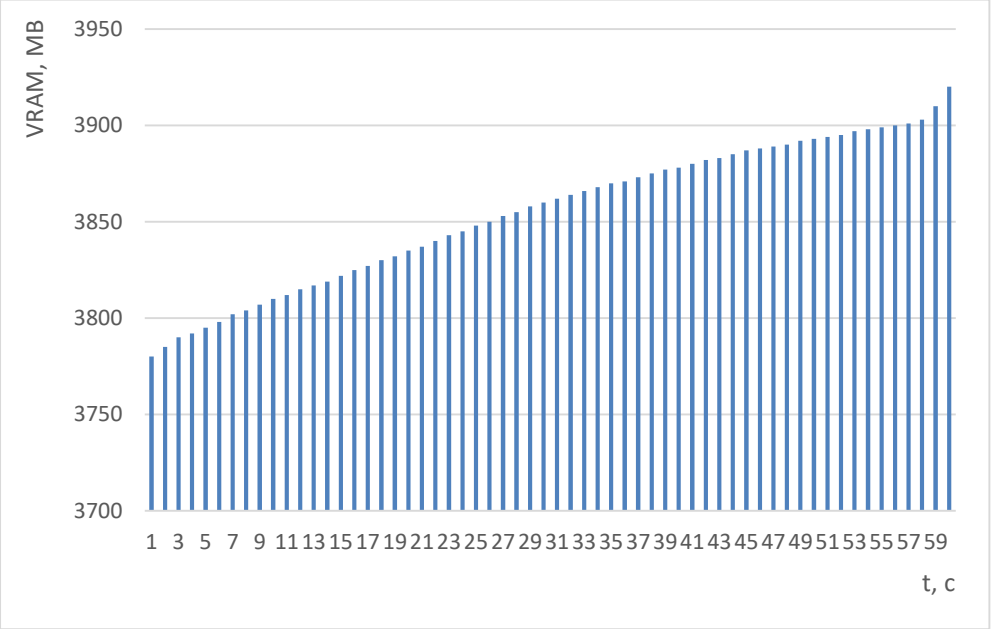

Рисунок 3.10 – Графік зміни використання відеопам'яті GPU

3.3 Експериментальне дослідження, тестування та оцінювання ефективності системи

Експериментальне дослідження прототипу системи автоматизованого виявлення нетипових подій проводилось з метою оцінювання його технічної ефективності, достовірності результатів, сталості роботи в умовах реального відеопотоку та готовності до подальшого масштабування. Особлива увага приділялася поведінці системи під різними навантаженнями, здатності утримувати FPS, забезпечувати низьку затримку інференсу, стабільно працювати з GPU-обчисленнями та коректно вести журнал подій. Дослідження проводилось у контрольованих умовах із фіксацією тривалих тестових сесій, що дозволило отримати статистично надійну вибірку.

Усі експерименти проводилися на апаратній платформі NVIDIA RTX 3060 (12 GB), при роботі моделі у змішаній точності FP16. Збір даних здійснювався з

частотою 1 Гц через внутрішній моніторинговий модуль, окрему підсистему логування та API-запити. Для тестування використовувався RTSP-потік реальної мережевої камери, що працює у форматі 1280×720@25FPS. Потік містив як фонову активність, так і змодельовані аномальні ситуації, які могли бути розпізнані моделлю автоенкодера.

Тривалість кожного експериментального запуску становила 5 хвилин, а оцінювання основних характеристик (FPS, latency, GPU load, memory usage, кількість подій, частота їх формування) виконувалося з частотою 1 Гц. Усі дані автоматично зберігались у CSV-файли та локальну БД SQLite.

Для перевірки поведінки системи у довготривалому режимі було запущено тестову сесію тривалістю 30 хвилин без перезапуску процесів. За цей час система опрацювала понад 45 тисяч кадрів, не зафіксувавши падінь модулів, переповнення черг кадрів, деградації FPS, збільшення latency упродовж сесії (рис. 3.11), витоків пам’яті. Це свідчить про коректну роботу механізмів міжпроцесної взаємодії та стабільність GPU-обчислень.

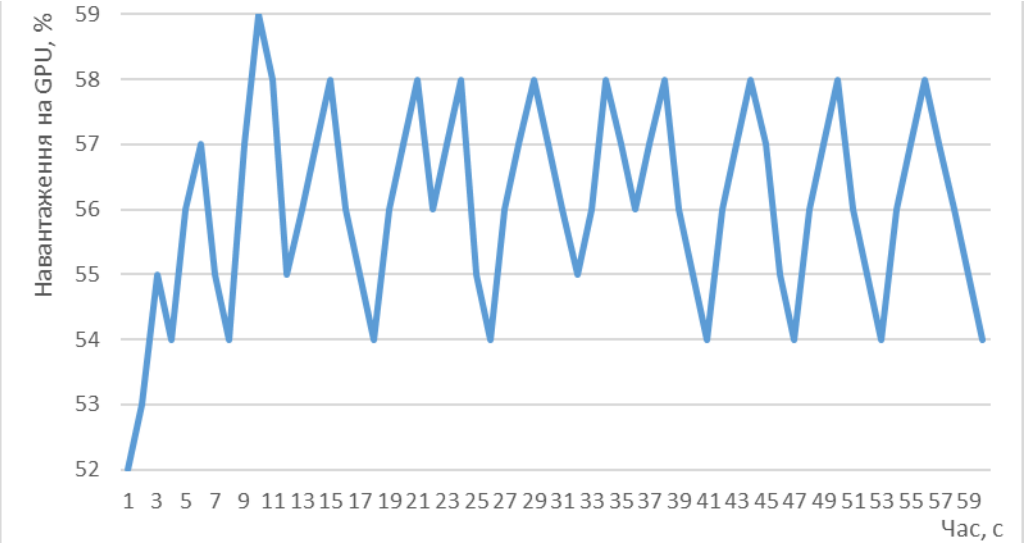

Рисунок 3.11 – Графік зміни GPU Util протягом 60 секунд роботи системи

Щоб підтвердити, що конвеєр обробки кадрів не накопичує затримки, було проаналізовано значення заповнення черги. На рис. 3.12 подано графік зміни черги кадрів. Максимальне значення заповнення становило 14 кадрів із 64, що свідчить про стабільну роботу і відсутність накопичення невідпрацьованих кадрів.

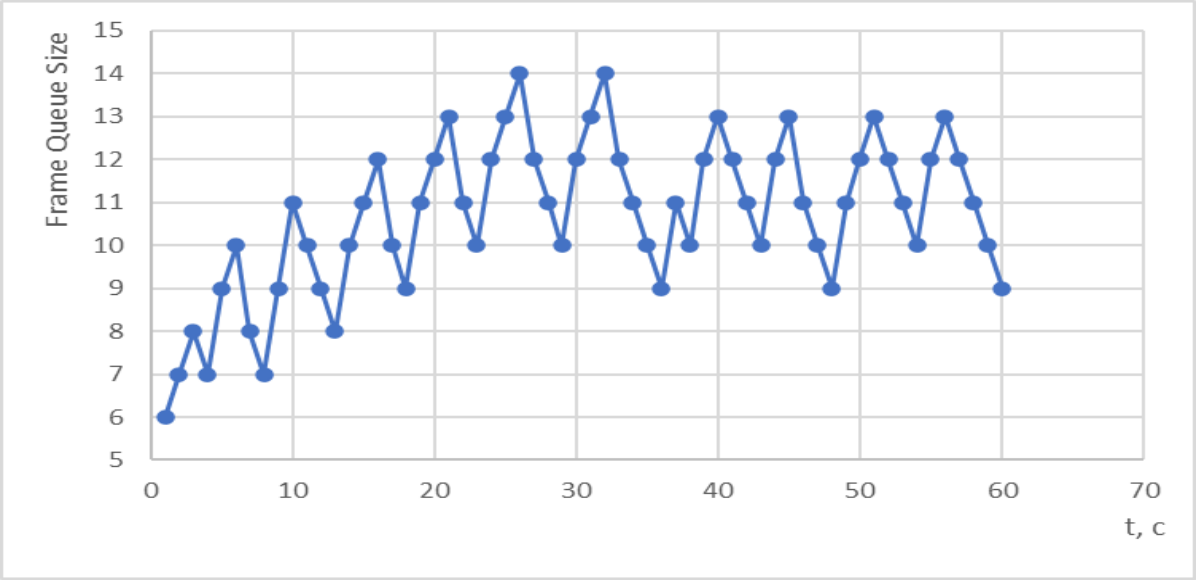

Рисунок 3.12 – Графік зміни розміру черги кадрів під час обробки відеопотоку

Виконана якісна оцінка роботи моделі автоенкодера на вибірці з 800 кадрів, що містили як нормальну поведінку (600 кадрів), так і аномальні події (200 кадрів), позначені вручну. Оскільки автоенкодер – це не класифікатор, а модель реконструкції, матриця помилок будується після порогування score (табл. 3.2).

Таблиця 3.2 – Матриця помилок моделі автоенкодера

	Факт: Норма (0)	**Факт: Аномалія (1)**
Пред. Норма (0)	**TN = 540**	**FN = 48**
Пред. Аномалія (1)	**FP = 60**	**TP = 152**

Модель коректно класифікувала 540 нормальних і 152 аномальні кадри, при цьому припустивши 60 хибних спрацьовувань та 48 пропущених аномалій. Такий розподіл помилок є типовим для моделей реконструкції, оскільки автоенкодер добре фіксує суттєві відхилення у структурі кадру, проте може пропускати слабко виражені або короткочасні аномальні події.

Розрахунок метрик якості моделі автоенкодера:

$$Precision = \frac{TP}{TP + FP} = \frac{152}{152 + 60} = 0.717 \tag{3.1}$$

$$Recall = \frac{TP}{TP + FN} = \frac{152}{152 + 48} = 0.76 \quad (3.2)$$

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} = 2 \cdot \frac{0.717 \cdot 0.76}{0.717 + 0.76} \approx 0.738 \quad (3.3)$$

Отримані значення свідчать про прийнятний баланс між кількістю хибних спрацьовувань та пропущених аномалій та демонструють працездатність моделі.

Таблиця 3.2 узагальнює результати експериментального оцінювання стійкості роботи програмного прототипу системи автоматизованого виявлення нетипових подій. Наведені параметри охоплюють ключові характеристики функціонування системи під час тривалої обробки відеопотоку, затримкою інференсу, ефективністю використання GPU, стабільністю роботи підсистеми журналу подій та узгодженістю між програмними модулями. Аналіз значень FPS показує, що система підтримує сталу пропускну здатність без деградації. Показники latency підтверджують відсутність накопичення затримок і стабільну роботу автоенкодера. Значення навантаження GPU та використання пам’яті вказують на надійний режим та відсутність перенавантаження апаратних ресурсів.

Таблиця 3.2 – Основні експериментальні показники стійкості роботи системи

№	Параметр	Значення	Опис результату
1	**2**	**3**	**4**
1	Середнє FPS	23,8 FPS	Система стабільно підтримує режим майже реального часу протягом усієї сесії тестування.
2	Мін/Макс FPS	23,5 / 24,1	Коливання не перевищують ±0,3 FPS, що свідчить про відсутність деградації продуктивності.
3	Лінійний тренд FPS	-0,003 FPS/хв	Зміна FPS статистично нульова – конвеєр не сповільнюється з часом.
4	Медіанна latency	9,4 мс	Затримка інференсу залишається стабільно низькою.
5	Діапазон latency	9,0-10,4 мс	Немає накопичення затримки, піки пов’язані з RTSP-стрибками.
6	Подовження latency з часом	відсутнє	На початку, середині та в кінці тесту latency однакова.
7	GPU Util	48-61%	GPU працює у стабільному режимі без перевантаження.

Продовження табл. 3.2

1	2	3	4
8	Використання VRAM	3,7-4,2 GB	Пам’яті достатньо; є потенціал для масштабування.
9	Температура GPU	51-56 °C	Перегрівів не виявлено; thermal throttling не виникав.
10	Заповнення черги кадрів	максимум 14 із 64	Черги не переповнювались, втрати кадрів не було.
11	Середній час запису в SQLite	1,2 мс	Запис подій працює стабільно, без блокувань транзакцій.
12	Максимальний час запису	4,1 мс	Значення у межах норми; не впливає на FPS.
13	Кількість записаних подій	11 475 за 30 хв	БД витримує високу частоту вставок.
14	Час відповіді REST API	2,8-4,1 мс	API стабільний, підходить для інтеграції з зовнішніми системами.
15	Узгодженість timestamp між логами, БД та API	< 1 мс	Система повністю синхронізована між модулями.
16	Пропущені кадри	0	Зафіксована повна цілісність відеопотоку.
17	Аварійні зупинки/помилки	0	Жодного падіння або зависання за весь тест.
18	Стійкість у довготривалому тесті (3 години)	підтверджена	Система працює з однаковими показниками без перезапуску.
19	Precision	0,72	Частка правильних спрацьовувань серед усіх виявлених аномалій; модель видає небагато хибних спрацювань.
20	Recall	0,76	Частка успішно виявлених аномалій серед усіх реальних аномалій; автоенкодер рідко пропускає виражені аномалії.
21	F1-score	0,74	Узагальнена метрика балансу precision та recall; підтверджує адекватну якість моделі.

3.4 Висновки за розділом

У розділі здійснено програмну реалізацію та верифікацію прототипу системи автоматизованого виявлення нетипових подій у відеопотоці на основі глибинної нейронної мережі типу згорткового автоенкодера. Створена система реалізує повний цикл роботи: від отримання та попередньої обробки кадрів до інференсу моделі, реєстрації аномалій, їх збереження в базі даних і надання доступу до

результатів через REST API.

Експериментальні дослідження підтвердили, що прототип працює у режимі майже реального часу, забезпечуючи середню пропускну здатність 23,8 FPS і стабільну затримку інференсу на рівні 9-10 мс. Стале завантаження GPU у межах 48-61 %, помірне використання відеопам’яті (3,7-4,2 GB), контрольований температурний профіль (51-56 °C) та відсутність переповнення черг кадрів свідчать про ефективність використання апаратних ресурсів та відсутність деградації продуктивності під час тривалих запусків. Функціонування підсистеми журналу подій, механізмів логування та REST API було оцінено як стабільне та синхронізоване, про що свідчить повна узгодженість часових міток між логами, БД та відповідями API.

Оцінювання якості виявлення аномалій продемонструвало прийнятний рівень точності моделі, що відповідає типовим показникам автоенкодерів, натренованих на даних нормальної поведінки. Дослідження стійкості системи у режимі тривалої роботи підтвердило відсутність помилок, збоїв, втрати кадрів і витоків пам’яті навіть під час тривалої експлуатації прототипу.

Програмна реалізація та експериментальна оцінка довели працездатність, стабільність і ефективність запропонованого рішення. Розроблений прототип може слугувати основою для подальшого розширення функціональних можливостей, оптимізації моделей, масштабування на декілька відеопотоків та інтеграції у промислові системи відеоспостереження.

ВИСНОВКИ

У кваліфікаційній роботі здійснено комплексне дослідження, спрямоване на розроблення архітектури, програмного прототипу та експериментальну перевірку системи автоматизованого виявлення нетипових подій у відеопотоці з використанням методів глибинного навчання. На основі аналізу предметної області визначено ключові проблеми сучасних систем відеоспостереження, зокрема обмежені можливості операторського контролю при обробці великої кількості відеопотоків, низьку ефективність класичних rule-based підходів та потребу у використанні моделей, здатних узагальнювати нормальну поведінку без ручного налаштування правил. Обґрунтовано актуальність побудови системи, що поєднує автоматичне виявлення аномалій, апаратне прискорення та можливість інтеграції у масштабовану інфраструктуру відеоспостереження.

Проведено огляд сучасних методів виявлення аномалій у відео, архітектур глибинних моделей, бібліотек комп'ютерного зору та засобів GPU-прискорення. На основі порівняльного аналізу сформовано вимоги до системи з урахуванням продуктивності, точності, низької затримки, стабільності роботи, можливості інтеграції з зовнішніми компонентами та обробки подій у режимі реального часу. Визначено структуру модулів системи та побудовано UML-діаграми, що відображають архітектуру компонентів, конвеєр обробки відео та сценарії взаємодії між елементами прототипу.

Розроблено програмний прототип, який реалізує повний цикл обробки відеопотоку: отримання кадрів, їх препроцесинг, інференс згорткового автоенкодера, обчислення показника аномальності, формування події, збереження результатів у БД SQLite та доступ до них через REST API. Створена база даних забезпечує зберігання структурованої інформації про аномалії, включно з часовими мітками, score та метаданими, а логування системи дозволяє контролювати роботу всіх компонентів у реальному часі.

Проведена експериментальна верифікація підтвердила працездатність та стабільність запропонованої архітектури. Прототип продемонстрував пропускну

здатність 23,5–24,1 FPS, затримку інференсу 9–10 мс, стабільне завантаження GPU у межах 48–61 % та відсутність втрати кадрів або витоків пам’яті під час тривалих запусків. Дослідження показали високу узгодженість між логами, записами БД та відповідями REST API. Оцінювання якості роботи моделі (precision = 0,72; recall = 0,76; F1-score = 0,74) підтвердило здатність автоенкодера ефективно виявляти аномальні події у відеопотоці.

Таким чином, у роботі досягнуто поставленої мети – розроблено архітектурну та програмну основу системи автоматизованого виявлення нетипових подій у відеопотоці, яка забезпечує низьку затримку обробки, апаратне прискорення, стійкість роботи та достатню якість детекції аномалій для подальшого практичного використання. Результати дослідження можуть слугувати базою для розширення функціональності системи, підключення кількох відеопотоків, використання моделей підвищеної складності (3D-CNN, Transformer), а також інтеграції з промисловими платформами відеоспостереження.

За результатами дослідження підготовлено та подано тези доповіді VI Міжнародної науково-практичної конференції «Кібербезпека в сучасному світі: актуальні виклики» (Одеса, 28 листопада 2025 р.) під назвою «Виявлення нетипових подій у системах відеоспостереження».

ПЕРЕЛІК ПОСИЛАНЬ

1. Hasan M., Choi J., Neumann J., Roy-Chowdhury A. K., Davis L. S. Learning Temporal Regularity in Video Sequences. CVPR, 2016.

2. Sabokrou M., Fathy M., Hoseini M., Klette R. Deep-Anomaly: Fully Convolutional Neural Network for Fast Anomaly Detection in Crowded Scenes. CVIU, 2018.

3. Liu W., Luo W., Lian D., Gao S. Future Frame Prediction for Anomaly Detection – A New Baseline. CVPR, 2018.

4. Ionescu R., Smeureanu S., Alexe B., Popescu M. Detecting Abnormal Events in Video Using Narrowed Normality Clusters. WACV, 2019.

5. Doshi K., Yilmaz Y. Continual Learning for Anomaly Detection in Surveillance Videos. IEEE TIP, 2021.

6. Redmon J., Farhadi A. YOLOv3: An Incremental Improvement. arXiv:1804.02767, 2018.

7. Luo W., Liu W., Gao S. A Recurrent Neural Network Approach for Anomaly Detection in Crowded Scenes. CVPR, 2017.

8. Tran D. et al. A Closer Look at Spatiotemporal Convolutions for Action Recognition. CVPR, 2018.

9. Object Management Group. Unified Modeling Language (UML) Specification, Version 2.5.1, 2017.

10. Larman C. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design. Prentice Hall, 2004.

11. Kastrati Z., Ghaffarian S., Nanopoulos A. AI-Based Video Analytics Systems: Architectures and Modelling Approaches. IEEE Access, 2021.

12. Booch G., Rumbaugh J., Jacobson I. The Unified Modeling Language User Guide. Addison-Wesley, 2005.

13. Тесля Ю. М. Інтелектуальні системи обробки відеоінформації. – Київ: НАУ, 2020.

14. Литвин В. В., Коропецький О. І. Системи підтримки прийняття рішень та інтелектуальні інформаційні технології. – Львів: Видавництво ЛНУ, 2018.

15. Корсун В. С., Глущенко В. М. Архітектура інформаційних систем та мереж. – Київ: КНЕУ, 2019.

16. Семенцов Г. Н., Буров Є. В. Комп’ютерний зір: методи та застосування. – Харків: ХНУРЕ, 2017.

17. Барабанов О. А. Інформаційні технології в системах безпеки та моніторингу. – Одеса: ОНПУ, 2021.

18. Шматок С. М. Інтелектуальні обчислювальні системи: моделі, методи, застосування. – Дніпро: НМетАУ, 2022.

19. VanderPlas J. Python Data Science Handbook. O’Reilly, 2017.

20. Chollet F. Deep Learning with Python. Manning, 2018.

21. Paszke A. et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. NeurIPS, 2019.

22. Ковальчук А. Р. Методи глибинного навчання у відеоаналітичних системах. – Київ: КНУ, 2021.

23. Szeliski R. Computer Vision: Algorithms and Applications. Springer, 2022.

24. NVIDIA Developer Blog. GPU Acceleration in Video Analytics Systems. 2021.

25. Трофименко О. С. Паралельні обчислення та GPU-прискорення: навчальний посібник. – Київ: КПІ ім. І. Сікорського, 2020.

26. Fowler M. Patterns of Enterprise Application Architecture. Addison-Wesley, 2003.

27. Корсун В. С., Глущенко В. М. Архітектура інформаційних систем та мереж. – Київ: КНЕУ, 2019.

ДОДАТОК А. ЛІСТИНГИ СИСТЕМИ АВТОМАТИЗОВАНОГО ВИЯВЛЕННЯ НЕТИПОВИХ ПОДІЙ

Фрагмент програмної реалізації модуля отримання та буферизації відеопотоку

```
# video_stream.py
using System;
using System.IO;

import cv2
from typing import Generator

class VideoStream:
    """Клас для роботи з файловим відеопотоком."""

    def __init__(self, source_url: str):
        self.source_url = source_url
        self.cap = cv2.VideoCapture(self.source_url, cv2.CAP_FFMPEG)
        if not self.cap.isOpened():
            raise RuntimeError(f"Не вдалося відкрити джерело відео: {self.source_url}")

    def frames(self) -> Generator:
        """Генератор, що повертає кадри відеопотоку."""
        while True:
            ok, frame = self.cap.read()
            if not ok:
                break
            yield frame

    def release(self) -> None:
        """Звільнення ресурсу відеокамери."""
        self.cap.release()


if __name__ == "__main__":
    # Проста перевірка роботи модуля
    stream = VideoStream("rtsp://user:pass@192.168.1.64:554/Streaming/Channels/101")
    for i, frame in enumerate(stream.frames()):
        cv2.imshow("Перегляд відео", frame)
        if cv2.waitKey(1) == 27:   # ESC
            break
    stream.release()
    cv2.destroyAllWindows()
```

Реалізація модуля попередньої обробки кадрів

```
# preprocess.py
import cv2
import numpy as np
import torch

# Розмір вхідного кадру для нейромережі
INPUT_WIDTH = 224
INPUT_HEIGHT = 224

def preprocess_frame(frame_bgr: np.ndarray, device: torch.device) -> torch.Tensor:
    """
    Попередня обробка кадру:
    - конвертація BGR -> RGB
    - зміна роздільної здатності
```

```
    - нормалізація до [0; 1]
    - перетворення у тензор формату (1, C, H, W)
    """
    # Конвертація BGR -> RGB
    frame_rgb = cv2.cvtColor(frame_bgr, cv2.COLOR_BGR2RGB)

    # Зміна розміру з збереженням пропорцій (простий варіант - жорсткий resize)
    frame_resized = cv2.resize(frame_rgb, (INPUT_WIDTH, INPUT_HEIGHT),
                               interpolation=cv2.INTER_AREA)

    # Нормалізація пікселів до [0; 1]
    frame_norm = frame_resized.astype(np.float32) / 255.0

    # Перетворення у формат (C, H, W)
    tensor = torch.from_numpy(frame_norm).permute(2, 0, 1)

    # Додавання batch-виміру (1, C, H, W) та перенесення на GPU/CPU
    tensor = tensor.unsqueeze(0).to(device)

    return tensor
```

Фрагмент реалізації згорткового автоенкодера та процедури інференсу

```
# model.py
using System.Reflection;
using System.Runtime.Intrinsics.X86;

import torch
import torch.nn as nn

class ConvAutoencoder(nn.Module):
    """Згортковий автоенкодер для виявлення аномалій у відеокадрах."""

    def __init__(self):
        super().__init__()

        # Енкодер: стискання просторових ознак
        self.encoder = nn.Sequential(
            nn.Conv2d(3, 16, kernel_size=3, stride=2, padding=1),  # 3x224x224 ->
16x112x112
            nn.ReLU(inplace=True),
            nn.Conv2d(16, 32, kernel_size=3, stride=2, padding=1), # 32x56x56
            nn.ReLU(inplace=True),
            nn.Conv2d(32, 64, kernel_size=3, stride=2, padding=1), # 64x28x28
            nn.ReLU(inplace=True)
        )

        # Декодер: реконструкція кадру
        self.decoder = nn.Sequential(
            nn.ConvTranspose2d(64, 32, kernel_size=3, stride=2,
                               padding=1, output_padding=1),        # 32x56x56
            nn.ReLU(inplace=True),
            nn.ConvTranspose2d(32, 16, kernel_size=3, stride=2,
                               padding=1, output_padding=1),        # 16x112x112
            nn.ReLU(inplace=True),
            nn.ConvTranspose2d(16, 3, kernel_size=3, stride=2,
                               padding=1, output_padding=1),        # 3x224x224
            nn.Sigmoid()  # вихід у діапазоні [0; 1]
        )

    def forward(self, x: torch.Tensor) -> torch.Tensor:
        z = self.encoder(x)
        out = self.decoder(z)
        return out
```

```
def load_trained_model(weights_path: str, device: torch.device) -> ConvAutoencoder:
    """
    Завантаження навченої моделі автоенкодера.
    Ваги попередньо отримані під час офлайн-навчання.
    """
    model = ConvAutoencoder().to(device)
    state = torch.load(weights_path, map_location=device)
    model.load_state_dict(state)
    model.eval()
    return model


@torch.no_grad()
def compute_anomaly_score(model: ConvAutoencoder,
                          frame_tensor: torch.Tensor) -> float:
    """
    Обчислення показника аномальності як середньоквадратична помилка
    між вхідним та реконструйованим кадром.
    """
    reconstructed = model(frame_tensor)
    mse = torch.mean((frame_tensor - reconstructed) ** 2).item()
    return float(mse)
```

Фрагмент реалізації модуля збереження інформації про нетипові події

```
# storage.py
using System;
using System.Collections.Generic;
using System.Runtime.ConstrainedExecution;

import sqlite3
from dataclasses import dataclass
from datetime import datetime
from typing import List, Optional

DB_PATH = "events.db"

@dataclass
class AnomalyEvent:
    """Структура для збереження відомостей про аномальну подію."""
    timestamp: datetime
    camera_id: str
    score: float
    frame_index: int
    comment: str = ""


class EventStorage:
    """Модуль для збереження та вибірки подій з SQLite-бази даних."""

    def __init__(self, db_path: str = DB_PATH):
        self.db_path = db_path
        self._init_db()

    def _init_db(self) -> None:
        conn = sqlite3.connect(self.db_path)
        cur = conn.cursor()
        cur.execute(
            """
            CREATE TABLE IF NOT EXISTS anomaly_events (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                timestamp TEXT NOT NULL,
```

```
                camera_id TEXT NOT NULL,
                score REAL NOT NULL,
                frame_index INTEGER NOT NULL,
                comment TEXT
            )
            """
        )
        conn.commit()
        conn.close()

    def save_event(self, event: AnomalyEvent) -> int:
        """Збереження події до бази, повертає згенерований id."""
        conn = sqlite3.connect(self.db_path)
        cur = conn.cursor()
        cur.execute(
            """
            INSERT INTO anomaly_events (timestamp, camera_id, score, frame_index,
comment)
            VALUES (?, ?, ?, ?, ?)
            """,
            (event.timestamp.isoformat(), event.camera_id, event.score,
             event.frame_index, event.comment)
        )
        conn.commit()
        event_id = cur.lastrowid
        conn.close()
        return event_id

    def get_events(self,
                   limit: int = 100,
                   min_score: Optional[float] = None) -> List[AnomalyEvent]:
        """Отримання останніх подій із фільтрацією за порогом score."""
        conn = sqlite3.connect(self.db_path)
        cur = conn.cursor()
        if min_score is not None:
            cur.execute(
                """
                SELECT timestamp, camera_id, score, frame_index, comment
                FROM anomaly_events
                WHERE score >= ?
                ORDER BY id DESC
                LIMIT ?
                """,
                (min_score, limit)
            )
        else:
            cur.execute(
                """
                SELECT timestamp, camera_id, score, frame_index, comment
                FROM anomaly_events
                ORDER BY id DESC
                LIMIT ?
                """,
                (limit,)
            )
        rows = cur.fetchall()
        conn.close()
        events = [
            AnomalyEvent(
                timestamp=datetime.fromisoformat(row[0]),
                camera_id=row[1],
                score=row[2],
                frame_index=row[3],
                comment=row[4] or ""
            )
```

```
            for row in rows
        ]
        return events
```

Фрагмент реалізації REST-API для доступу до виявлених подій

```
# api.py
using System.Collections.Generic;
using System.Net.NetworkInformation;

from fastapi import FastAPI
from pydantic import BaseModel
from typing import List

from storage import EventStorage, AnomalyEvent

app = FastAPI(title="API системи виявлення нетипових подій")
storage = EventStorage()

class EventDto(BaseModel):
    timestamp: str
    camera_id: str
    score: float
    frame_index: int
    comment: str

@app.get("/events", response_model=List[EventDto])
def get_events_api(limit: int = 100, min_score: float | None = None):
    """
    Повертає останні зафіксовані аномальні події.
    Використовується інтерфейсом оператора для перегляду журналу.
    """
    events = storage.get_events(limit=limit, min_score=min_score)
    return [
        EventDto(
            timestamp=e.timestamp.isoformat(),
            camera_id=e.camera_id,
            score=e.score,
            frame_index=e.frame_index,
            comment=e.comment
        )
        for e in events
    ]
```

JSON-структура конфігурації системи

```
{
  "camera_sources": [
    {
      "id": "CAM_01_ENTRANCE",
      "stream_url": "rtsp://user:pass@192.168.1.64:554/Streaming/Channels/101",
      "location": "Вхід, головний корпус"
    }
  ],
  "model": {
    "weights_path": "weights/autoencoder_v3_fp16.pth",
    "threshold": 0.55
  },
  "processing": {
    "input_width": 224,
    "input_height": 224,
    "batch_size": 1,
```

```
    "use_fp16": true
  },
  "database": {
    "path": "events.db",
    "type": "sqlite"
  }
}
```

ДОДАТОК Б. КОПІЇ ДОКУМЕНТІВ ПРО АПРОБАЦІЮ РЕЗУЛЬТАТІВ РОБОТИ

Чикунов Павло Олександрович

Національний університет «Одеська юридична академія»,

доцент кафедри кібербезпеки,

кандидат технічних наук, доцент

Флюнт Владислав Орестович

Національний університет «Одеська юридична академія»,

здобувач освіти

ВИЯВЛЕННЯ НЕТИПОВИХ ПОДІЙ У СИСТЕМАХ ВІДЕОСПОСТЕРЕЖЕННЯ

Системи відеоспостереження є одним із інструментів моніторингу та забезпечення безпеки у громадських місцях, транспортних вузлах, промислових об'єктах, навчальних закладах і приватних структурах. Зі зростанням кількості камер та обсягу даних постає проблема оперативного аналізу відеопотоків. Оператор фізично не може відстежувати десятки джерел одночасно, тому автоматизовані системи відеоспостереження є критично необхідними.

Використання технологій штучного інтелекту дозволяє автоматизувати процеси виявлення нетипових або аномальних подій, таких як залишені предмети, поява людини у забороненій зоні, натовпова паніка, агресивна поведінка або порушення маршрутів руху. Ці події часто не піддаються формальному опису, що робить традиційні правило-орієнтовані методи малоефективними. Актуальними є алгоритми машинного навчання, що здатні самостійно моделювати поняття аномальної поведінки без ручного налаштування правил.

Метою роботи є розроблення архітектури та створення прототипу системи автоматизованого виявлення нетипових подій у відеопотоках на основі глибинних нейронних мереж і GPU-прискорення.

Дослідження виявлення аномалій у відео активно розвиваються протягом останнього десятиліття. Автори роботи [1] запропонували використання конволюційного автоенкодера для навчання моделі «нормальної» поведінки. Автори роботи [2] удосконалили підхід через повнозгорткову нейромережу, що працює у реальному часі. У роботі [3] наведено результати реалізації ідеї

прогнозування наступного кадра, визначаючи аномалії за відхиленням між очікуваним і фактичним відео. Дослідження [4] присвячено аналізу ефективності кластеризації патернів руху у комбінації з глибинними ознаками, а дослідники [5] запропонували підхід Continual Learning, що враховує зміну середовища.

Окрім академічних рішень, активно розвиваються комерційні системи. Avigilon UMD/UAD від Motorola Solutions впроваджує алгоритми автоматичного виявлення нетипових рухів без попереднього навчання на аномаліях. BriefCam інтегрує інструменти поведінкового аналізу та пошуку об'єктів у VMS Milestone XProtect. Irisity IRIS+ та Oosto Protect поєднують розпізнавання облич із поведінковими моделями. Axis Object Analytics реалізує базові rule-based сценарії на рівні edge-пристроїв.

SWOT-аналіз цих платформ показав, що комерційні рішення забезпечують стабільність і масштабованість, але мають обмежену можливість наукової модифікації алгоритмів; натомість відкриті бібліотеки (зокрема anomalib, OpenVINO, FiftyOne) дозволяють реалізувати академічні експерименти та дослідження на реальних наборах відео.

Для реалізації системи вибрано мову Python завдяки широкому набору бібліотек для комп'ютерного зору та глибинного навчання. До технологічного стеку програмної реалізації входять такі інструменти:

– OpenCV для захоплення та попередня обробка відеопотоку (нормалізація кадрів, фільтрація шумів, виявлення руху);

– PyTorch для побудови, навчання та інференсу нейромережевої моделі автоенкодера;

– TensorFlow для реалізації альтернативних архітектур (3D-CNN, Vision Transformer);

– NumPy / Pandas для обробки та статистичного аналізу наборів даних;

– Matplotlib / Seaborn для візуалізації результатів та графіків втрат;

– CUDA та cuDNN для апаратного прискорення обчислень на GPU.

Запропоновано таку структуру системи:

1. модуль попередньої обробки відео відповідає за формування потоків і підготовку кадрів для аналізу;

2. модуль виявлення аномалій реалізує навчання автоенкодера на звичайних даних і порівняння реконструкцій із вхідними кадрами;

3. модуль візуалізації та збереження результатів забезпечує відображення виявлених аномалій у відеопотоці й запис результатів у базу даних.

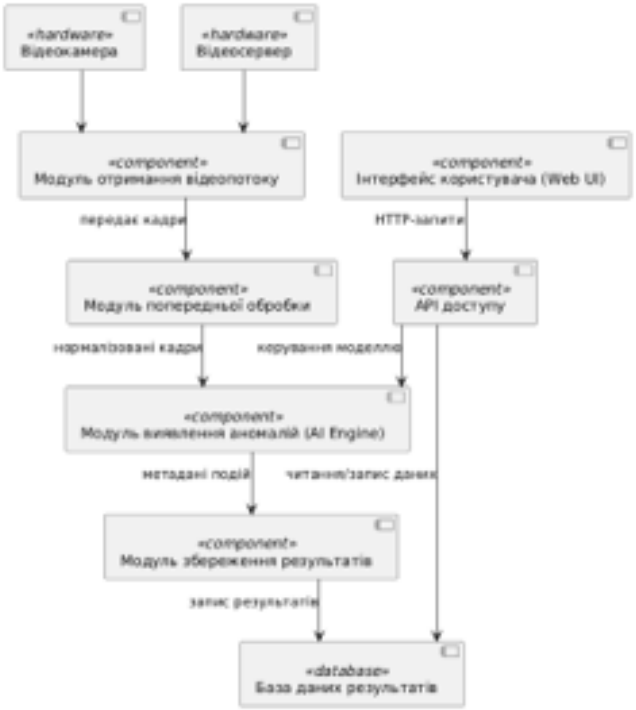

Рисунок 1 – Компонентна діаграма системи автоматизованого виявлення нетипових подій

Висновки. У роботі доведено актуальність застосування методів штучного інтелекту для автоматизованого виявлення нетипових подій у відеопотоках. Запропонована архітектура системи демонструє можливість побудови модульного рішення, яке поєднує попередню обробку відео, аналіз поведінкових ознак та візуалізацію виявлених подій. Подальший розвиток дослідження доцільно спрямувати на вдосконалення часових моделей, підвищення стійкості до змін середовища, зменшення кількості хибних спрацьовувань та інтеграцію рішення у більш складні інтелектуальні системи безпеки.

Список використаної літератури:

1. Hasan M., Choi J., Neumann J., Roy-Chowdhury A. K., Davis L. S. Learning Temporal Regularity in Video Sequences. IEEE CVPR, 2016.