ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Система розпізнавання об’єктів відео потоку з камер спостереження на основі методів машинного навчання»

на здобуття освітнього ступеня бакалавра

зі спеціальності 122 Комп’ютерні науки

освітньо-професійної програми Штучний інтелект

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

_____ Назар БАШУК
(підпис)

Виконав:
здобувач вищої освіти Назар БАШУК
група ШІД-41

Керівник: Олександр РАБОТА

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

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Штучного інтелекту

Ступінь вищої освіти Бакалавр

Спеціальність 122 Комп'ютерні науки

Освітньо-професійна програма Штучний інтелект

ЗАТВЕРДЖУЮ
Завідувач кафедрою Штучного інтелекту
_____ Ольга ЗІНЧЕНКО
«_____» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Башуку Назару Олександровичу

1. Тема кваліфікаційної роботи: Система розпізнавання об'єктів відео потоку з камер спостереження на основі методів машинного навчання
керівник кваліфікаційної роботи Олександр РАБОТА,
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» лютого 2025 р. № 56
2. Строк подання кваліфікаційної роботи «23» травня 2025 р.
3. Вихідні дані до кваліфікаційної роботи: відеопотік з камер спостереження, навчалні датасети, API-ключ до Google Gemini API, налаштування системи.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
   Аналіз сучасних технологій розпізнавання об'єктів у відеопотоці.
   Дослідження методів комп'ютерного зору та глибокого навчання.
   Реалізація захоплення, обробки і аналізу відеопотоку в реальному часі.
5. Перелік графічного матеріалу: *презентація*
   1. Схема виявлення об'єктів на зображенні
   2. Архітектура системи розпізнавання об'єктів
   3. Схема обробки кадру через Gemini API
   4. Графік ефективності різних типів промптів
6. Дата видачі завдання«25» лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	25.02-05.11.25	Виконано
2	Вивчення методів машинного навчання та бібліотек	05.11-12.11.25	Виконано
3	Дослідження існуючих технологій розпізнавання об’єктів	13.11-19.11.25	Виконано
4	Обґрунтування вибору технології (Gemini API) та побудова архітектури системи	20.11-25.11.25	Виконано
5	Реалізація модулів захоплення відео, взаємодії з API, візуалізації результатів	27.11-03.12.25	Виконано
6	Тестування, вимірювання точності, швидкодії, створення логів	04.12-10.12.25	Виконано
7	Оформлення текстової частини: вступ, висновки, реферат	11.12-20.12.25	Виконано
8	Розробка демонстраційних матеріалів, презентації	21.12-23.05.25	Виконано

Здобувач вищої освіти _____ Назар БАШУК
(підпис)

Керівник
кваліфікаційної роботи _____ Олександр РАБОТА
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 63 стор., 4 табл., 22 рис., 42 джерел.

Мета роботи – розробка функціональної системи розпізнавання об’єктів у відеопотоці з камер спостереження на основі методів машинного навчання.

Об’єкт дослідження – процес обробки відеоінформації з камер спостереження.

Предмет дослідження – алгоритми та моделі розпізнавання об’єктів у відеопотоці з використанням методів глибокого навчання.

Короткий зміст роботи:

У роботі проведено огляд сучасних підходів до побудови систем відеоаналітики з використанням технологій комп’ютерного зору та нейронних мереж. Аналізуються популярні архітектури (YOLO, SSD, Faster R-CNN), а також бібліотеки OpenCV, TensorFlow, PyTorch. Обґрунтовано вибір Google Gemini API як хмарного рішення для обробки візуального контенту.
Розроблено архітектуру програмної системи, що включає модулі захоплення відео, попередньої обробки кадрів, інтеграції з API, обробки результатів та візуалізації. Реалізовано підтримку адаптивної частоти кадрів, кешування, логування.

Система протестована в умовах, наближених до реального відеоспостереження, досягнута точність розпізнавання становила 85–94%. Запропоновано підходи до подальшого вдосконалення точності та стабільності системи.

КЛЮЧОВІ СЛОВА: РОЗПІЗНАВАННЯ ОБ’ЄКТІВ, КОМП’ЮТЕРНИЙ ЗІР, ВІДЕОПОТІК, ШТУЧНИЙ ІНТЕЛЕКТ, МАШИННЕ НАВЧАННЯ, OPENCV, GEMINI API.

ABSTRACT

The textual part of the bachelor's qualification thesis: 63 pages, 4 tables, 22 figures, 42 references.

The aim of the work is to develop a functional object recognition system in video streams from surveillance cameras based on machine learning methods. Object of the research – the process of video data processing from surveillance cameras. Subject of the research – algorithms and models for object recognition in video streams using deep learning methods.

Brief description of the work:

The work presents an overview of modern approaches to building video analytics systems using computer vision technologies and neural networks. Popular architectures such as YOLO, SSD, and Faster R-CNN are analyzed, as well as libraries including OpenCV, TensorFlow, and PyTorch. The choice of Google Gemini API as a cloud-based solution for visual content processing is justified.

A system architecture has been developed, including modules for video capture, frame preprocessing, API integration, result processing, and visualization. The implementation includes adaptive frame rate handling, caching, and logging mechanisms.

The system was tested under conditions close to real-world video surveillance, achieving an object recognition accuracy of 85–94%. Approaches for further improving the system's accuracy and stability are proposed.

KEYWORDS: OBJECT RECOGNITION, COMPUTER VISION, VIDEO STREAM, ARTIFICIAL INTELLIGENCE, MACHINE LEARNING, OPENCV, GEMINI API.

ЗМІСТ

ВСТУП

У сучасному світі питання безпеки стає все більш актуальним. Зростаюча кількість публічних просторів, комерційних об’єктів, транспортних вузлів та житлових комплексів вимагає ефективного контролю і моніторингу ситуацій у режимі реального часу. Одним із найпоширеніших технічних засобів забезпечення безпеки є системи відеоспостереження, які дозволяють фіксувати події, ідентифікувати об’єкти та здійснювати моніторинг поведінки в зоні огляду. Однак традиційні системи обмежені у функціональності, адже потребують участі оператора, що унеможливлює повноцінне реагування на події в реальному часі через людський фактор.

Актуальність роботи. У зв’язку з розвитком штучного інтелекту, зокрема машинного навчання, з’являються нові можливості для автоматизації аналізу відеопотоку. Технології розпізнавання об’єктів дозволяють значно розширити функціональність звичайних систем спостереження: виявлення підозрілих дій, ідентифікація транспортних засобів, розпізнавання облич тощо. Реалізація таких систем є актуальною як для підвищення громадської безпеки, так і для оптимізації процесів в промисловості, логістиці та роздрібній торгівлі.

Особливої ваги тема набуває у контексті сучасних тенденцій до цифровізації міського середовища — створення «розумних міст» (smart city), де кожен елемент інфраструктури повинен мати здатність до адаптації, аналізу та самостійного прийняття рішень. У такій парадигмі саме системи автоматичного розпізнавання відеооб’єктів на основі нейронних мереж виступають ключовим інструментом збору та обробки даних.

Метою даної дипломної роботи є розробка функціональної системи розпізнавання об’єктів на основі відеопотоку з камер спостереження із застосуванням методів машинного навчання, що здатна працювати в реальному часі з належним рівнем точності та продуктивності.

Об’єктом дослідження виступає процес обробки відеоінформації з камер спостереження.

Предметом дослідження є алгоритми розпізнавання об’єктів у відеопотоці з використанням методів машинного навчання, зокрема глибокого навчання.

Основні завдання дипломної роботи:

1. Провести огляд сучасних технологій відеоаналітики та методів машинного навчання, що використовуються у розпізнаванні об’єктів.
2. Визначити оптимальні інструменти для реалізації системи (бібліотеки, фреймворки, моделі).
3. Обрати архітектуру нейронної мережі, що найкраще підходить для задачі детекції об’єктів у відео.
4. Розробити програмну реалізацію системи обробки відеопотоку з розпізнаванням об’єктів у реальному часі.
5. Провести тестування системи, оцінити точність, швидкодію та стабільність.
6. Проаналізувати результати, сформулювати висновки та надати рекомендації щодо подальшого удосконалення системи.

У дослідженні використано комплекс методів. Теоретичний аналіз здійснювався на основі вивчення наукової та технічної літератури щодо технологій комп’ютерного зору, алгоритмів машинного навчання, моделей нейронних мереж і систем відеоспостереження. Емпірична частина ґрунтується на експериментальному моделюванні системи розпізнавання із застосуванням мов програмування Python та бібліотек OpenCV, TensorFlow і PyTorch. Було реалізовано практичну модель з використанням відеопотоку, проведено навчання та тестування нейромережевої моделі, а також здійснено аналіз результатів за допомогою метрик точності, recall, precision та FPS. Робота поєднує теоретичну обґрунтованість з практичним інженерним підходом до розробки інтелектуальних систем.

РОЗДІЛ 1. АНАЛІЗ СУЧАСНИХ ТЕХНОЛОГІЙ РОЗПІЗНАВАННЯ ОБ'ЄКТІВ У ВІДЕОПОТОЦІ

1.1. Основи комп'ютерного зору та машинного навчання

Комп'ютерний зір (Computer Vision) — це міждисциплінарна галузь, яка поєднує інформатику, штучний інтелект, математику, а також фізику оптики та цифрову обробку зображень. Її головною метою є розробка систем, здатних отримувати, інтерпретувати та аналізувати візуальну інформацію з реального світу так, як це робить людське око, але в автоматичному режимі. Це включає обробку зображень і відео, ідентифікацію об'єктів, виявлення закономірностей у сценах, відстеження руху, а також прийняття рішень на основі зорових даних на рис. 1.1. (a) приклад де модель ідентифікує та локалізує об'єкти на зображенні за допомогою обмежувальних рамок, (b) приклад сегментації, де модель присвоює мітки на рівні пікселів різним областям зображення, (c) приклад оцінки пози, де модель прогнозує розташування та орієнтацію основних суглобів тіла людини[1]

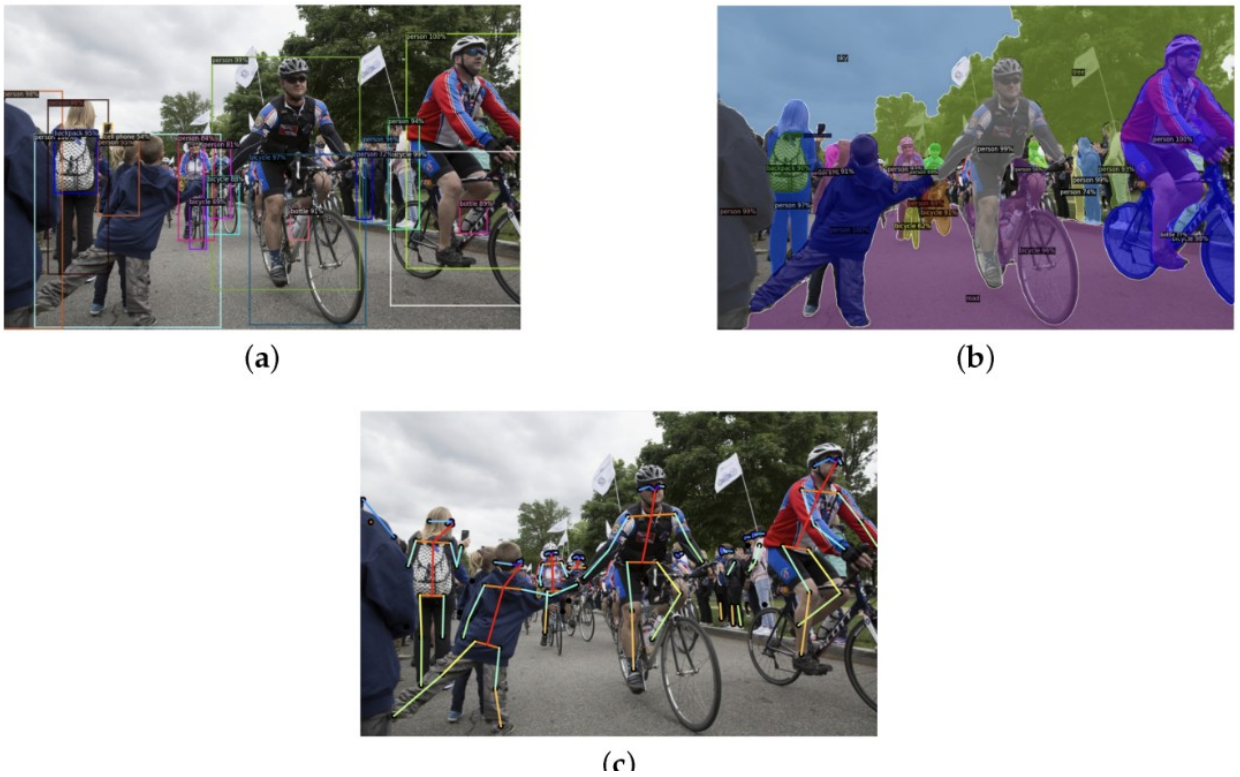


Рис 1.1 Приклади виявлення об'єктів.

Формування зображення, яке комп'ютер може аналізувати, розпочинається з отримання цифрового сигналу за допомогою камери. Цей сигнал

трансформується у матрицю пікселів, кожен із яких має певні значення яскравості або кольору. Подальші етапи включають попередню обробку зображення (фільтрація шуму, зміна розміру, нормалізація), сегментацію (виділення областей, що потенційно містять об’єкти), а потім — детекцію та класифікацію вмісту кадру.

Для реалізації комп’ютерного зору необхідно використовувати складні алгоритми аналізу зображень. Спочатку такі системи базувалися на класичних алгоритмах комп’ютерного зору: виявлення контурів (Canny, Sobel), пошук ключових точок (SIFT, SURF, ORB), векторизація об’єктів, застосування гістограм градієнтів (HOG) або каскадів Хаара. Ці методи були обмежені в умовах складного фону, змін освітлення, шуму, а також погано масштабувалися для великих обсягів відеоданих.

Зі стрімким розвитком штучного інтелекту в останні десятиліття, комп’ютерний зір почав базуватись на методах машинного навчання, зокрема глибокого навчання (deep learning). На відміну від класичних алгоритмів, які потребували ручного створення ознак (feature engineering), сучасні моделі здатні самостійно навчатися на основі великих наборів зображень і знаходити оптимальні параметри для розпізнавання. Основою таких систем є штучні нейронні мережі, які імітують структуру та принципи функціонування біологічного мозку [2].

Найбільш поширеним різновидом нейронних мереж, які використовуються в комп’ютерному зорі, є згорткові нейронні мережі (Convolutional Neural Networks, CNN). Ці моделі навчені виявляти патерни в зображеннях — лінії, форми, структури — шляхом проходження через кілька шарів згортки, активації та субдискретизації. Результатом роботи CNN є ймовірнісна оцінка приналежності вхідного зображення до певного класу або координати виявленого об’єкта на зображенні.

Машинне навчання, у свою чергу, — це підгалузь штучного інтелекту, яка полягає у розробці алгоритмів, здатних навчатися на основі наданих даних і приймати рішення без прямого програмування логіки. У контексті комп’ютерного

зору найбільше застосування отримали методи наглядуваного навчання (supervised learning), де кожне зображення має мітку (label), що дозволяє мережі навчитися зіставляти вхідні дані з бажаними результатами. Приклади таких методів — класифікація зображень (визначення об’єкта), регресія (передбачення координат рамки), сегментація (поділ зображення на області за класами), а також комбіновані підходи, як-от детекція об’єктів.

Особливістю машинного навчання в системах відеоспостереження є необхідність адаптації моделей до змін середовища: варіацій освітлення, сезонності, кута огляду, типу камери. Саме тому моделі повинні не лише мати високу точність на навчальних даних, але й демонструвати хорошу узагальнювальну здатність на нових відео. Для цього використовуються прийоми підвищення стійкості моделей: аугментація даних, регуляризація, крос-валідація та переднавчання на великих відкритих наборах, таких як COCO або ImageNet.

Синтез комп’ютерного зору і машинного навчання є основою сучасних автоматизованих систем обробки візуальної інформації. Такі системи здатні працювати з високою продуктивністю, швидко адаптуються до нових умов і можуть ефективно застосовуватись у сфері безпеки, охорони, логістики, транспорту, медицини та інших галузях [3].

Одним із найпоширеніших прикладів використання комп’ютерного зору на практиці є система відеоспостереження в торгових центрах або на вокзалах, де необхідно оперативно виявляти присутність людини у визначеній зоні, фіксувати час її появи та відслідковувати переміщення. Традиційна система без участі штучного інтелекту здатна лише записувати відео, тоді як система з підтримкою машинного навчання може автоматично виявляти об’єкти (наприклад, людей чи транспортні засоби), класифікувати їх, запускати тривожні повідомлення, рахувати кількість або зберігати інформацію про кожне виявлення для подальшої аналітики.

Для вирішення таких задач найчастіше використовують моделі згорткових нейронних мереж типу YOLO (You Only Look Once), SSD (Single Shot Multibox Detector) або Faster R-CNN. Модель YOLO, зокрема, відома своєю високою

швидкодією і здатністю працювати в реальному часі, що дозволяє здійснювати розпізнавання об’єктів у потоці з камери без значних затримок. Цей підхід реалізується за допомогою фреймворку Darknet або його портованих варіантів на TensorFlow і PyTorch. Наприклад, система, реалізована на базі YOLOv5, може обробляти відеопотік із IP-камери зі швидкістю понад 30 кадрів на секунду при виявленні пішоходів, автомобілів або багажу.

У реальній практиці особливу увагу слід приділити попередній обробці вхідного відео. У випадках змінного освітлення, зашумленості кадрів або низької роздільної здатності слід застосовувати методи нормалізації, згладжування зображення, а також адаптивні алгоритми виявлення контрасту. Крім того, необхідно забезпечити правильну підготовку датасету для навчання нейронної мережі. Якщо модель тренується на відкритих наборах даних, таких як COCO або Pascal VOC, то при переході на реальну сцену (наприклад, внутрішній дворик або склад) вона може показувати знижену точність через зміну контексту. У таких випадках доцільно застосовувати transfer learning — перенавчання моделі на власних даних для підвищення її адаптивності до умов цільової сцени [4].

На рівні програмної реалізації широко застосовується мова програмування Python у поєднанні з бібліотеками OpenCV для обробки зображень, TensorFlow або PyTorch для побудови та навчання моделей, NumPy та Pandas для маніпулювання даними. Робота з відеопотоком здійснюється через API до IP-камер (наприклад, протокол RTSP), а кадри обробляються у циклі, де кожен фрейм подається на вхід нейромережі, після чого модель повертає координати рамки, клас об’єкта і ймовірність розпізнавання.

Під час тестування системи на практиці важливими показниками ефективності є точність (accuracy), повнота виявлення (recall), хибні спрацювання (false positives), а також швидкодія (frames per second, FPS). Наприклад, у демонстраційному середовищі система може досягати точності розпізнавання понад 90% при обробці відео з роздільною здатністю 640x480 на середньостатистичному ноутбуці з графічним прискорювачем.

Таблиця 1.1

Компоненти та інструменти комп’ютерного зору

Компонент системи	**Опис функцій/приклади**
Вхідне зображення/відеопотік	Відео з камер спостереження або збережені зображення
Попередня обробка зображення	Фільтрація шуму, нормалізація, масштабування, зміна кольорового простору
Модель розпізнавання об’єктів	YOLOv5, SSD, Faster R-CNN — детекція та класифікація об’єктів
Фреймворки машинного навчання	OpenCV, TensorFlow, PyTorch, Keras — реалізація алгоритмів
Типи нейронних мереж	CNN, ResNet, EfficientNet — виявлення ознак на зображенні
Метрики оцінки якості	Accuracy, Precision, Recall, FPS — оцінка продуктивності
Середовище розгортання	Сервери, ноутбуки, NVIDIA Jetson, Raspberry Pi — для запуску систем

Після аналізу основних компонентів комп’ютерного зору та машинного навчання, представлених у таблиці 1.1, варто більш детально зосередитися на практичних особливостях їх взаємодії у межах цілісної системи розпізнавання об’єктів у відеопотоці. Найважливішим аспектом такої інтеграції є ефективне поєднання апаратного забезпечення з оптимізованими програмними алгоритмами, здатними обробляти зображення в режимі реального часу [5].

У процесі побудови подібної системи першим кроком є захоплення відеопотоку з камери спостереження. Це може бути IP-камера з підтримкою RTSP-протоколу або USB-камера, підключена до локального комп’ютера. Отримане відео обробляється фрейм за фреймом: кожне зображення проходить етапи попередньої обробки, такі як приведення до необхідного розміру, нормалізація яскравості та контрасту, а також перетворення кольорового простору

(наприклад, з BGR у RGB або grayscale). Ці процедури критично важливі, оскільки від якості підготовки вхідних даних залежить точність подальшого розпізнавання.

Наступним етапом є обчислення прогнозу за допомогою навченої нейронної мережі. У випадку використання моделі YOLO (наприклад, версії YOLOv5), система розбиває зображення на сітку та визначає ймовірність присутності об’єкта в кожній клітинці, а також координати рамки навколо нього. Якщо ймовірність перевищує заданий поріг (наприклад, 0.5), об’єкт вважається знайденим. У результаті користувач отримує зображення з накладеними прямокутниками та підписами класів (людина, автомобіль, велосипед тощо), а також значенням ймовірності.

Особливу увагу слід приділити продуктивності системи, яка залежить від багатьох чинників: потужності процесора, наявності графічного прискорювача (GPU), розміру моделі, частоти кадрів відео, а також якості коду. Наприклад, при використанні моделі YOLOv5s (модифікація з малою кількістю параметрів) на графічному прискорювачі NVIDIA RTX 3060 можна досягти обробки 50–60 кадрів на секунду з точністю понад 85% для типових об’єктів [6].

У рамках практичного застосування система розпізнавання об’єктів може бути налаштована на генерацію подій, таких як надсилання сповіщень про виявлення певного класу об’єкта, запуск тривоги або збереження відео- або фотозапису при фіксації руху. Також можливе поєднання з модулями обліку часу, геолокації, ідентифікації за номерними знаками або обличчями. Такі функції розширюють можливості системи і дозволяють інтегрувати її в інтелектуальні середовища типу Smart City або підприємства з розвиненою інфраструктурою безпеки.

Реалізація подібних систем у реальному середовищі також стикається з низкою викликів, серед яких — обмеження в обчислювальних ресурсах, потреба в постійному оновленні навчальних даних, адаптація моделей до змінних умов (ніч, дощ, сніг, туман), а також захист персональних даних користувачів, якщо йдеться про розпізнавання облич. Саме тому кожен компонент системи повинен бути

ретельно протестований і налаштований з урахуванням специфіки завдань та середовища розгортання.

1.2. Технології обробки відеопотоку в реальному часі

Обробка відеопотоку в реальному часі є однією з найбільш складних та ресурсоємних задач комп'ютерного зору. Вона передбачає не лише здатність системи миттєво захоплювати, аналізувати та інтерпретувати кожен кадр відео, але й забезпечити безперервну роботу з мінімальними затримками. В умовах зростаючих вимог до безпеки, автоматизації спостереження та реакції на події, технології реального часу стають фундаментом для інтелектуальних систем відеоаналітики [7].

Основою будь-якої системи обробки відеопотоку є джерело візуальних даних — камера. Сучасні цифрові камери передають відео в потоковому форматі за допомогою протоколів RTSP, HTTP, HLS або WebRTC. Захоплення відео в програмному середовищі зазвичай реалізується через бібліотеку OpenCV або мультимедійні фреймворки типу FFmpeg. Відеопотік складається з послідовності кадрів (фреймів), які передаються із певною частотою — зазвичай 25–30 кадрів на секунду, але для систем із високою динамікою сцени можуть бути й вищі значення.

Після отримання кожного кадру, зображення обробляється на рівні попередньої фільтрації: усуваються шуми, вирівнюється освітлення, виконується нормалізація розмірів і кольорового простору. Ці дії мають велике значення для коректної роботи алгоритмів розпізнавання, особливо у випадках нестабільного освітлення або складного фону. Для забезпечення високої швидкодії кожен етап має бути реалізований максимально оптимізовано, а обчислення — розпаралелено.

Наступним важливим етапом є детекція об'єктів на кадрі. Тут на перший план виходять згорткові нейронні мережі, такі як YOLO, SSD або CenterNet, які поєднують швидкодію і високу точність. Їх особливістю є здатність виконувати виявлення об'єктів за один прохід нейронної мережі — тобто одночасно

визначати, де знаходиться об’єкт (bounding box) і до якого класу він належить. Це суттєво знижує час обробки кадру, що є критичним для реального часу [8].

Однією з проблем, які виникають при обробці відео в реальному часі, є баланс між швидкістю і точністю. З одного боку, модель повинна бути достатньо точною, аби правильно розпізнавати об’єкти, з іншого — не занадто великою, щоб не перевантажувати апаратні ресурси. Вирішення цієї задачі досягається шляхом використання оптимізованих версій моделей (наприклад, YOLOv5n або MobileNet), які пристосовані для роботи на слабших пристроях або в умовах обмежених обчислювальних потужностей [9].

Окрім цього, для прискорення обчислень активно застосовується апаратне прискорення за допомогою графічних процесорів (GPU) або спеціалізованих прискорювачів — таких як NVIDIA Jetson, Google Coral, Intel Movidius, пристрої дозволяють досягти високої частоти кадрів навіть на складних моделях. Використання технологій TensorRT або OpenVINO дозволяє перетворювати стандартні моделі у формат, що краще працює на специфічному апаратному забезпеченні, забезпечуючи значне зростання продуктивності без втрати точності.

У процесі обробки відео важливо також реалізувати контроль за часовими обмеженнями. Система повинна обробити кожен кадр у рамках заданого інтервалу (наприклад, 30 мс для 30 FPS). Якщо кадри накопичуються швидше, ніж встигає працювати модель, можуть виникати затримки, пропуски або зависання. Для уникнення цього в реальних системах реалізується вибірка не кожного кадру (наприклад, обробка кожного другого або п’ятого кадру), а також механізми пріоритету кадрів або асинхронного оброблення.

Крім виявлення об’єктів, система часто виконує їх трекінг — відстеження переміщення об’єкта в наступних кадрах. Це дозволяє, з одного боку, підвищити ефективність системи (немає потреби виявляти об’єкт знову), з іншого — фіксувати траєкторії руху, розпізнавати поведінку або аналізувати динаміку сцени. Для реалізації трекінгу використовують алгоритми Deep SORT, Kalman Filter або Median Flow, які можуть працювати в режимі реального часу.

Інтеграція системи обробки відео з іншими компонентами, як-от базою даних, системою сповіщень, веб-інтерфейсом або API для віддаленого доступу, також є важливою частиною. Вона дозволяє створити повноцінну інформаційну систему, яка не лише аналізує потік з камери, але й реагує на виявлені події, фіксує інциденти або передає інформацію відповідальним особам. В умовах, коли камера спостереження функціонує цілодобово, система повинна бути здатна безперервно обробляти значні обсяги інформації, реагувати на появу нових об'єктів і адаптуватися до змін навколишнього середовища без втрати точності або швидкодії. Особливо критичним є це у сферах, де обробка даних затримкою навіть у кілька секунд може призвести до втрати важливої інформації або небезпеки — наприклад, в аеропортах, на вокзалах, у банках або в місцях масового скупчення людей [10].

У таких умовах постає завдання правильної організації архітектури системи. Сучасна система відеоаналітики, яка обробляє відеопотік у реальному часі, зазвичай має декілька незалежних модулів: один відповідає за захоплення кадрів, інший — за попередню обробку, наступний — за передачу зображень до нейронної мережі, ще один — за обробку результатів і реагування на події. Важливо, щоб ці компоненти працювали паралельно, без блокування одне одного. Для цього використовується багатопоточне програмування або асинхронні архітектури, які дозволяють обробляти кілька кадрів одночасно, не знижуючи продуктивність. Наприклад, один потік може захоплювати зображення з камери, інший – аналізувати попередній кадр, а третій – зберігати результати у базу даних або відображати на екрані.

У практичному застосуванні значне значення має також вибір формату відео, кодека та роздільної здатності. Якщо перед системою не стоїть задача точного аналізу дрібних деталей, доцільно знижувати роздільну здатність до 640x480 або 320x240, що дозволяє істотно зменшити обчислювальне навантаження. Однак у випадках, коли необхідно точно розпізнати обличчя або номерний знак, зниження якості може негативно вплинути на точність, тому такі системи повинні працювати на балансі між продуктивністю та якістю зображення.

Питання компресії відео також не слід ігнорувати, оскільки деякі кодеки, наприклад H.265, хоч і зменшують навантаження на мережу, але вимагають додаткових ресурсів для декодування.

Ще одним важливим аспектом є стійкість до перешкод і зміни умов. Система повинна зберігати стабільність роботи при зміні яскравості, у випадках заднього світла, присутності диму або туману, або коли об'єкти частково перекривають одне одного. Для цього можуть застосовуватись додаткові фільтри зображення, використання моделей, натренованих на різноманітних даних, або використання ІЧ-камер і термографічних датчиків. У перспективі розробники інтегрують додаткові сенсори, які дозволяють поєднувати зорову інформацію з іншими каналами — наприклад, акустичними або тепловими [11].

Варто також наголосити, що ефективна система обробки відеопотоку в реальному часі має бути не лише технологічно досконалою, але й економічно доцільною, що означає, що використання дорогого обладнання виправдане лише у випадках, де безпека є критично важливою, тоді як у загальнодоступних або комерційних застосуваннях доцільно використовувати оптимізовані рішення на основі недорогих процесорів або edge-комп'ютерів. Такий підхід дозволяє створити масштабовані системи, які можна розгортати у великих обсягах, наприклад, на автопарковках, у магазинах або житлових комплексах, не втрачаючи при цьому керованості та ефективності.

У підсумку, розробка та впровадження систем відеоаналітики в реальному часі вимагає ретельного балансу між теоретичною точністю моделей машинного навчання, їхньою технічною реалізацією та практичними обмеженнями реального середовища. Об'єднання цих факторів дозволяє створювати рішення, які не просто аналізують відео, а надають реальну цінність у контексті безпеки, автоматизації, логістики та інших напрямків сучасної цифрової інфраструктури.

Таблиця 1.2

Технології обробки відео в реальному часі

Технологія / Компонент	**Призначення / Роль у системі**
RTSP (Real-Time Streaming	Потокова передача відео з IP-камери у режимі

Технологія / Компонент	Призначення / Роль у системі
Protocol)	реального часу
OpenCV	Захоплення, обробка, попередня фільтрація кадрів
YOLOv5	Швидка нейронна мережа для детекції об’єктів у відео
TensorRT	Оптимізація моделі нейронної мережі для прискорення на GPU
GPU-прискорення (NVIDIA)	Забезпечення високої швидкодії при розпаралеленні обчислень
Багатопоточна обробка	Незалежна обробка кадрів і компонентів відеопотоку
Deep SORT трекінг	Відстеження траєкторії об’єктів на відео після їх виявлення

Після аналізу ключових технологій, наведених у таблиці 1.2, стає очевидним, що ефективна обробка відеопотоку в реальному часі вимагає комплексного поєднання протоколів передачі, високопродуктивного програмного забезпечення та оптимізованих апаратних ресурсів. У реальних умовах така система повинна працювати безперервно, не знижуючи швидкодії навіть у моменти пікового навантаження або під час аналізу складних сцен із великою кількістю об’єктів. Кожен із зазначених компонентів відіграє свою специфічну роль: протоколи типу RTSP відповідають за стабільну передачу потоку, OpenCV забезпечує базову обробку кадрів, YOLOv5 реалізує швидке виявлення об’єктів, а TensorRT або GPU-прискорення дозволяють досягти необхідної швидкодії.

Принципова важливість полягає не лише в окремій роботі кожного компонента, а у їх скоординованій взаємодії, яка забезпечує узгоджену обробку даних. Наприклад, система, яка обробляє 25 кадрів на секунду, повинна не лише виявити об'єкти на кожному кадрі, а й передати інформацію до модуля трекінгу, визначити поведінкову модель об'єкта, а в разі порушення визначених правил — ініціювати відповідну дію. Це означає, що сама по собі детекція є лише частиною функціональної архітектури, тоді як вся система виконує повноцінний цикл аналізу, реагування і фіксації ситуацій у реальному часі [12].

Крім того, у практиці особливо важливо враховувати варіативність зовнішнього середовища. Однією з головних особливостей реального відеопотоку є його динамічність: об’єкти з’являються і зникають, змінюють свою форму або положення, освітлення сцени може варіюватися щосекунди, а фонові умови змінюються впродовж доби. Саме тому системи, що обробляють відео в реальному часі, повинні бути не лише точними, а й адаптивними, адаптивність досягається через використання моделей з високою узагальнювальною здатністю, а також через постійне оновлення навчального датасету з урахуванням конкретних умов, у яких функціонує система.

Іншим важливим аспектом є питання латентності — тобто часу між отриманням кадру та видачею результату. Для систем безпеки, що працюють в оперативному режимі, навіть затримка у 0,5–1 секунду може бути критичною. Тому системи, які справді функціонують у режимі реального часу, повинні демонструвати мінімальний latency, часто не більший за 100–200 мс. Для цього розробники використовують оптимізовані архітектури, які зменшують кількість обчислень на кожному етапі та забезпечують прискорене прийняття рішень.

Розробка систем обробки відеопотоку в реальному часі є складним, багатокомпонентним процесом, який вимагає глибокого розуміння взаємодії між програмними модулями, апаратним забезпеченням та вимогами конкретного середовища. Успішне впровадження таких систем можливе лише за умов ретельного планування архітектури, дотримання вимог продуктивності, і, головне, орієнтації на практичну результативність — тобто здатність системи вчасно виявляти, розпізнавати та адекватно реагувати на події у відеопотоці.

1.3. Існуючі підходи до виявлення та розпізнавання об’єктів

Процес виявлення та розпізнавання об’єктів є центральним елементом систем комп’ютерного зору, оскільки саме завдяки цьому етапу система отримує змогу «бачити» й «розуміти» вміст зображення чи відео. Історично, методи детекції об’єктів розвивалися від традиційних алгоритмічних підходів до глибокого навчання з використанням штучних нейронних мереж. Залежно від

рівня складності завдання, доступних ресурсів та вимог до точності й швидкодії, на практиці застосовуються різні методи [13].

Перші спроби автоматизованого виявлення об’єктів були засновані на так званих класичних алгоритмах обробки зображень. Серед них найпоширенішими були методи виявлення контурів, гістограми орієнтованих градієнтів (HOG), каскади Хаара та алгоритми сегментації зображень. Наприклад, каскадний класифікатор Хаара, реалізований в OpenCV, дозволяв ефективно виявляти обличчя на зображенні, використовуючи послідовність простих фільтрів. Такі методи були ефективними у вузькоспеціалізованих задачах, де сцена була статичною, а об’єкти добре відокремлювалися від фону. Основним недоліком таких підходів була їх неспроможність працювати у складних умовах: при змінному освітленні, частковому перекритті об’єктів, варіаціях ракурсу, масштабів чи контексту.

Революційним кроком у сфері розпізнавання об’єктів стало впровадження методів машинного навчання, які дозволили автоматизувати процес виділення ознак. Спочатку це були алгоритми з ручним формуванням дескрипторів, які подавалися на вхід класифікаторам, таким як Support Vector Machine (SVM) або Random Forest. Але справжній прорив відбувся із появою глибоких згорткових нейронних мереж, які стали основою сучасного підходу до розпізнавання.

Одним із перших значущих проривів була модель R-CNN, яка поєднувала метод регіональних пропозицій із CNN-класифікатором. Однак вона була повільною та ресурсоємною. Подальші модифікації, такі як Fast R-CNN і Faster R-CNN, значно прискорили процес, завдяки інтеграції регіонального пропонуючого модуля безпосередньо в архітектуру мережі. Faster R-CNN залишається досить популярною моделлю для високоточних задач детекції, особливо там, де швидкість не є критичною. Її точність значно перевищує класичні алгоритми, а також вона демонструє добру стійкість до складних умов, таких як часткове перекриття об’єктів або велика варіативність у класах.

Паралельно з цим активно розвивалися одноетапні детектори, які на відміну від двоетапних моделей, виконують виявлення і класифікацію об’єктів за один

прохід нейронної мережі. Найяскравішим представником цього підходу є сімейство моделей YOLO — You Only Look Once. З моменту виходу першої версії до сучасної YOLOv8, архітектура зазнала значних змін, проте залишилася однією з найшвидших моделей детекції, що забезпечує високу точність у реальному часі. Саме завдяки швидкодії та відносній простоті розгортання YOLO широко використовується у системах відеоспостереження, на транспорті, в охороні та промисловій автоматизації [14].

Ще одним популярним підходом є SSD — Single Shot MultiBox Detector. Ця модель поєднує високу швидкість і відносно невелику обчислювальну складність, тому часто використовується на мобільних пристроях та вбудованих системах. SSD працює шляхом застосування фільтрів до декількох шарів згорткової мережі, що дозволяє виявляти об’єкти різних розмірів на різних масштабах.

У сучасних реалізаціях детекції також використовуються новітні архітектури, зокрема EfficientDet, CenterNet, YOLO-NAS та Detectron2. Багато з цих моделей мають відкритий код, доступні у вигляді готових бібліотек або pretrained-моделей, і легко інтегруються в системи обробки відео. Деякі з них підтримують апаратне прискорення через TensorRT, ONNX або OpenVINO, що значно підвищує ефективність у середовищах з обмеженими ресурсами.

Окремо варто згадати про методи сегментації об’єктів, які не просто обмежуються побудовою прямокутної рамки навколо об’єкта, а визначають точні межі об’єкта на зображенні. До таких підходів належить, наприклад, модель Mask R-CNN, яка крім координат bounding box генерує маску — бінарне зображення, що відображає форму об’єкта, підхід має особливу цінність у медичній діагностиці, робототехніці та розширеній реальності.

Таблиця 1.3

Порівняння підходів до виявлення та розпізнавання об'єктів

Підхід / Модель, Опис	Швидкість	Точність	Застосування
R-CNN: Модель для виявлення об’єктів з використанням регіональних пропозицій, потребує великої обчислювальної потужності.	Повільна	Висока, але ресурсозатратна	Технічні дослідження, демонстраційні проекти
Fast R-CNN: Оптимізована версія R-CNN, що використовує попередньо отримані регіональні пропозиції для підвищення швидкості.	Швидка	Висока, покращена швидкість	Реальний час, практичні застосування
Faster R-CNN: Поліпшена версія Fast R-CNN, що включає етап регіональних пропозицій безпосередньо в архітектуру мережі для ще більшої швидкості.	Швидка	Висока	Високоточні задачі детекції об’єктів
YOLO: Модель, що виконує одноетапне виявлення об’єктів в реальному часі з високою швидкістю і точністю.	Швидка	Висока	Системи відеоспостереження, охорона
SSD : Модель з одноетапною детекцією, що підходить для мобільних пристроїв, має меншу обчислювальну складність.	Швидка	Середня	Мобільні пристрої, системи реального часу
Mask R-CNN : Розширення Fast R-CNN, яке також генерує маски для сегментації об’єктів на зображенні.	Середня	Висока (для сегментації)	Медична діагностика, робототехніка
EfficientDet :Швидка модель для виявлення об’єктів, що забезпечує баланс між швидкістю і точністю.	Швидка	Висока	Індустрія, робота з великими наборами даних
Detectron2: Бібліотека, що підтримує багато моделей для виявлення об’єктів, оптимізованих для різних задач.	Швидка	Висока	Розширене виявлення об’єктів, обробка в реальному часі

Продовжуючи після наведеної таблиці 1.3, слід зазначити, що вибір конкретного підходу до виявлення та розпізнавання об’єктів завжди має ґрунтуватися на комплексному аналізі завдань, які стоять перед системою, а також обмежень середовища, у якому вона функціонуватиме. Наприклад, у випадках, коли вирішальне значення має швидкодія, пріоритет надається одноетапним моделям, таким як YOLO або SSD, оскільки вони здатні обробляти зображення практично миттєво, що особливо актуально для відеопотоку в реальному часі. Якщо ж першочерговим є досягнення максимальної точності або можливість глибокого аналізу зображення (наприклад, точна локалізація складних форм), тоді доцільно використовувати двоетапні архітектури, зокрема Faster R-CNN або сегментаційні моделі на зразок Mask R-CNN.

Також важливо враховувати технічні ресурси, які доступні для розгортання системи. Складні нейронні мережі можуть вимагати значного обсягу оперативної пам’яті та високопродуктивних відеокарт, тоді як компактні моделі або оптимізовані версії (наприклад, YOLOv5n або EfficientDet-Lite) здатні працювати на мініатюрних пристроях на кшталт NVIDIA Jetson Nano, Raspberry Pi із прискорювачами або мобільних телефонів, обмеження стають визначальними при проєктуванні системи, особливо якщо йдеться про масове розгортання у мережах камер або віддалених спостережних пунктах [16].

У свою чергу, для задач зі специфічним предметним середовищем, таких як медицина, сільське господарство чи автономні транспортні засоби, моделі повинні бути перенавчені на відповідних наборах даних, що передбачає формування або розширення власних датасетів, де класи об’єктів ідентифікуються відповідно до цільового контексту. Наприклад, система, яка працює на складі, повинна розрізняти коробки, палети, навантажувачі та працівників, тоді як у транспортній сфері — транспортні засоби, світлофори, дорожні знаки та пішоходів. Навіть найточніша модель, натренована на загальному датасеті, може демонструвати низьку продуктивність, якщо вона не враховує специфіку середовища.

Особливої уваги заслуговує питання адаптації моделей до змін у середовищі. У реальному житті сцена постійно змінюється — з’являються нові об’єкти, умови освітлення варіюються впродовж дня, змінюється погода, кути огляду камер. Щоб система залишалася ефективною, її моделі повинні або проходити періодичне донавчання, або використовувати механізми адаптивного розпізнавання з використанням активного навчання або самооновлення на основі нового досвіду. Такі підходи дозволяють системі навчатися протягом життєвого циклу, що наближає її до принципів штучного інтелекту нового покоління.

Крім того, важливим аспектом є безпека і надійність роботи системи. У сфері безпеки, зокрема в аеропортах, банках, логістичних центрах чи на об’єктах критичної інфраструктури, помилкове або запізніле розпізнавання може призвести до небажаних наслідків. Саме тому застосовується багаторівнева перевірка виявлених об’єктів, крос-аналітика з іншими сенсорами, або застосування комбінації різних моделей, що доповнюють одна одну. У деяких випадках система також надає оператору можливість підтвердити або відхилити спрацювання, що створює механізм напівавтоматизованого контролю.

У підсумку, існуючі підходи до виявлення та розпізнавання об’єктів формують гнучкий інструментарій, який можна адаптувати до практично будь-якого сценарію використання. Завдяки широкому спектру доступних моделей, відкритому програмному забезпеченню та можливості перенавчання, сучасні системи можуть бути не лише ефективними, а й гнучкими, масштабованими та адаптивними. У контексті досліджуваної теми це дозволяє реалізувати систему відеоспостереження, здатну працювати в реальному часі, виявляти об’єкти з високою точністю, враховувати специфіку середовища та відповідати сучасним вимогам безпеки й автоматизації.

1.4. Огляд бібліотек та інструментів: OpenCV, TensorFlow, PyTorch

У процесі створення систем комп’ютерного зору та реалізації методів машинного навчання важливу роль відіграють інструменти програмної розробки, які забезпечують доступ до необхідних алгоритмів, структур даних, модулів

візуалізації, а також готових моделей. Найпоширенішими бібліотеками, які використовуються для побудови систем обробки зображень і відео, є OpenCV, TensorFlow та PyTorch. Кожна з них має свої унікальні особливості, функціональні можливості та сфери застосування, але всі разом формують потужний програмний фундамент для розробки інтелектуальних систем виявлення й розпізнавання об’єктів [15].

OpenCV, або Open Source Computer Vision Library, є однією з найстаріших і найбільш відомих бібліотек для комп’ютерного зору, що підтримує велику кількість алгоритмів для обробки зображень, відео та аналітики. Вона була створена з метою забезпечити доступ до інструментів обробки зображень у відкритому середовищі та є багатоплатформною — підтримує Windows, Linux, macOS, Android. Серед основних її можливостей — робота з матрицями зображень, геометричні перетворення, фільтрація, виявлення контурів, сегментація, детекція облич, об’єктів та руху, трекінг, розпізнавання образів, калібрування камер і навіть базові алгоритми машинного навчання. Крім того, OpenCV дозволяє напряму працювати з потоковим відео з IP-камер, що є критично важливим у реальному застосуванні. Особливістю OpenCV є її оптимізованість для роботи на процесорах, можливість інтеграції з GPU через OpenCL, а також підтримка обробки в реальному часі.

TensorFlow, у свою чергу, є фреймворком з відкритим кодом, розробленим Google спеціально для реалізації та розгортання нейронних мереж, зокрема моделей глибокого навчання. Цей інструмент підтримує як високорівневі, так і низькорівневі API для побудови складних обчислювальних графів, керування пам’яттю, оптимізації процесу навчання. TensorFlow використовується як для створення моделей з нуля, так і для донавчання вже існуючих моделей. Однією з головних переваг є наявність підсистеми TensorFlow Lite, яка дозволяє розгортати моделі на мобільних та вбудованих пристроях, а також TensorFlow.js — для запуску моделей у браузері. Важливо також, що екосистема TensorFlow включає модуль TensorBoard для моніторингу та візуалізації процесу навчання, а також набір попередньо натренованих моделей, доступних через TensorFlow Hub. Для

оптимізації виконання на GPU, CPU або TPU застосовується компілятор XLA, що дозволяє зменшити час інференсу та використання ресурсів.

PyTorch — ще одна провідна бібліотека для машинного навчання, розроблена Facebook AI Research. Вона здобула широку популярність завдяки своїй простоті, гнучкості, динамічному графу обчислень та зручному API. PyTorch надає розробнику змогу будувати модель у режимі реального часу (eager execution), що є надзвичайно зручним під час налагодження й експериментів [3]. Крім того, фреймворк має тісну інтеграцію з бібліотеками Python, такими як NumPy, SciPy, Matplotlib, і чудово підходить для академічних досліджень, прототипування, швидкого тестування нових ідей. Завдяки модулю TorchVision користувачі отримують доступ до стандартних архітектур нейронних мереж (ResNet, VGG, MobileNet), трансформерів, попередньо навчених моделей, а також набору функцій для обробки зображень, трансформацій, аугментації. PyTorch також підтримує навчання на GPU, оптимізацію моделі, автоматичне визначення похідних завдяки автодиференціюванню (autograd) та реалізацію кастомних шарів.

У контексті побудови системи для розпізнавання об’єктів на основі відеопотоку, поєднання OpenCV з однією з фреймворків для глибокого навчання — TensorFlow або PyTorch — є найбільш ефективним і поширеним підходом. OpenCV забезпечує зручний інтерфейс для захоплення та попередньої обробки зображень у реальному часі, тоді як TensorFlow або PyTorch реалізують глибинну модель розпізнавання, наприклад, YOLO, SSD або Faster R-CNN. Зокрема, PyTorch часто обирається для експериментальних і дослідницьких проєктів, а TensorFlow — для індустріальних, масштабованих рішень, які потребують розгортання в хмарі, на мобільних пристроях чи в браузері.

Важливо зазначити, що всі три інструменти активно підтримуються великими спільнотами розробників, мають розгорнуту документацію, приклади, відеоуроки, а також інтеграцію з платформами для керування моделями (MLflow, Weights & Biases), що дає змогу не лише створити ефективну систему, а й легко її оновлювати, масштабувати й інтегрувати з іншими технологіями. У поєднанні ці

бібліотеки забезпечують повний життєвий цикл проєкту: від обробки зображень до навчання, оптимізації, тестування та впровадження моделі в експлуатацію.

Після аналізу таблиці 1.4 стає зрозумілим що кожна з розглянутих бібліотек має свою спеціалізацію, яка визначає її ефективність у конкретних контекстах застосування. OpenCV залишається ключовим інструментом для базових завдань комп'ютерного зору, особливо у тих випадках, коли необхідна швидка попередня обробка зображень або робота з відеопотоком у реальному часі. Його легкість, підтримка численних алгоритмів фільтрації, геометричних трансформацій, обробки контурів, а також сумісність із різними мовами програмування, зокрема C++ і Python, робить його незамінним для розробників систем технічного зору, робототехніки та контролю якості.

У той же час TensorFlow і PyTorch значною мірою доповнюють OpenCV, дозволяючи створювати складні моделі глибокого навчання, які забезпечують функціональність розпізнавання, класифікації та сегментації об'єктів на зображеннях. TensorFlow, зокрема, демонструє високий рівень інтеграції з промисловими системами, мобільними пристроями та вебсередовищем, надаючи розробникам широкий інструментарій для розгортання моделей у різних форматах. Його модулі TensorFlow Lite і TensorFlow.js дозволяють використовувати розпізнавання навіть у пристроях з обмеженими ресурсами, що відкриває шлях до реалізації рішень у вбудованих системах, мобільних додатках або браузерних інтерфейсах [2].

З іншого боку, PyTorch здобув особливу популярність у науковому середовищі, оскільки його архітектура з динамічним обчислювальним графом дозволяє швидко тестувати нові ідеї, модифікувати структури моделей у процесі виконання та легко налагоджувати складні нейронні архітектури. Це забезпечує велику гнучкість під час досліджень, створення прототипів та академічних експериментів. Саме тому PyTorch часто використовується для створення моделей, які потім, після тестування й оптимізації, переносяться у TensorFlow для фінального розгортання на промислових системах [17].

Коли мова йде про створення повноцінної системи комп’ютерного зору з розпізнаванням об’єктів на відео, найбільш ефективною стратегією є поєднання цих інструментів. OpenCV використовується як механізм обробки відеопотоку, попереднього аналізу кадрів, виявлення руху, масштабування або перетворення кольору. Далі оброблене зображення передається у фреймворк глибокого навчання — TensorFlow або PyTorch — який вже виконує інференс моделі, тобто застосовує попередньо натреновану нейронну мережу для розпізнавання об’єктів на кожному кадрі. Після цього результати можуть повертатися у середовище OpenCV для візуалізації, збереження або подальшої обробки [18]. Логічне поєднання OpenCV, TensorFlow і PyTorch дозволяє побудувати повноцінний програмний стек, здатний ефективно вирішувати задачі обробки зображень, розпізнавання об’єктів і відеоаналітики. Завдяки відкритому коду, активній підтримці спільноти та регулярним оновленням, ці інструменти залишаються актуальними як для науковців, так і для інженерів, які реалізують практичні рішення у сфері штучного інтелекту та автоматизації відеоспостереження.

РОЗДІЛ 2. ОПИС ОБРАНОЇ ТЕХНОЛОГІЇ ДЛЯ РОЗПІЗНАВАННЯ ОБ'ЄКТІВ

2.1. Обґрунтування вибору технології

Розробка системи розпізнавання об'єктів у відеопотоці з камер спостереження потребує вибору технологічного рішення, яке забезпечить високу точність аналізу візуального контенту при збереженні гнучкості та простоти інтеграції. Після детального аналізу доступних підходів було обрано використання Google Gemini API як основної технології для аналізу відеопотоку в поєднанні з бібліотекою OpenCV для захоплення та попередньої обробки кадрів з камери [1].

Gemini API представляє собою мультимодальну модель штучного інтелекту від Google, здатну аналізувати візуальний контент та генерувати детальні текстові описи зображень. На відміну від традиційних нейронних мереж для детекції об'єктів, які потребують спеціалізованого навчання та значних обчислювальних ресурсів для інференсу, Gemini API надає готове рішення з широкими можливостями розуміння контексту сцени. Модель здатна не лише ідентифікувати об'єкти, але й описувати їх взаємодію, дії, просторові відношення та загальний контекст ситуації [2].

Ключовою перевагою використання Gemini API є відсутність необхідності у власній інфраструктурі для розгортання та обслуговування моделей машинного навчання. API працює через хмарний сервіс Google, що забезпечує високу доступність, масштабованість та постійне оновлення моделі без додаткових зусиль з боку розробника. Модель Gemini навчена на масивних датасетах, що охоплюють мільярди зображень з текстовими описами, що забезпечує розуміння широкого спектру об'єктів, сцен та ситуацій без потреби спеціалізованого донавчання [3].

Архітектура Gemini базується на трансформерній архітектурі з мультимодальними можливостями, що дозволяє обробляти як візуальну, так і текстову інформацію в єдиному представленні. Модель використовує механізми

уваги для встановлення зв'язків між різними частинами зображення та генерації когерентного опису. Підтримка різних режимів роботи - від простої ідентифікації об'єктів до детального аналізу сцени - робить API універсальним інструментом для різних сценаріїв використання [4]. Інтеграція з Python екосистемою через офіційну бібліотеку google-generativeai забезпечує простий та інтуїтивний інтерфейс для роботи з API. Підтримка асинхронних запитів дозволяє ефективно обробляти потокове відео без блокування основного потоку виконання. Гнучкість у формулюванні промптів дає можливість налаштовувати систему під конкретні завдання - від загального опису сцени до пошуку специфічних об'єктів чи виявлення аномальної поведінки [5].

2.2. Архітектура обраної системи

Загальна архітектура системи розпізнавання об'єктів побудована за модульним принципом з чітким розділенням відповідальності між компонентами. Система складається з модуля захоплення відеопотоку, модуля попередньої обробки кадрів, модуля взаємодії з Gemini API, модуля обробки результатів та модуля візуалізації і збереження даних.

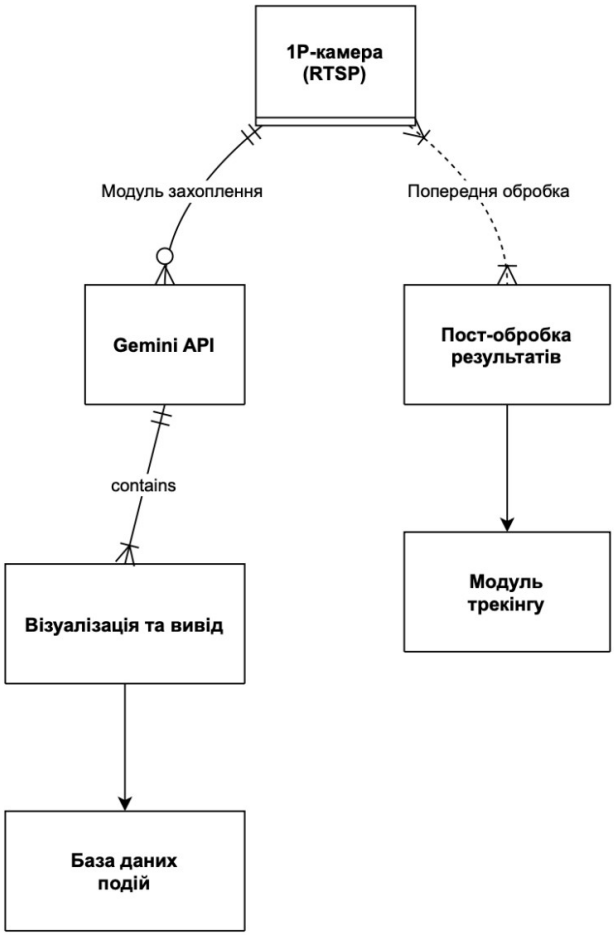


Рис 2.1 Архітектура системи розпізнавання об'єктів

Модуль захоплення відеопотоку реалізує інтерфейс до різних джерел відео, включаючи IP-камери через протокол RTSP, USB-камери, віртуальні камери типу iVCam для тестування, та відеофайли. Використання OpenCV забезпечує уніфікований доступ до всіх типів джерел через єдиний API. Модуль працює в окремому потоці для забезпечення безперервного захоплення кадрів незалежно від швидкості їх обробки. Буферизація кадрів реалізована через чергу з обмеженим розміром, що дозволяє адаптуватися до варіацій у швидкості обробки [6].

Попередня обробка кадрів включає масштабування зображення до оптимального розміру для передачі через API, конвертацію формату кольорів з BGR (стандарт OpenCV) до RGB, кодування зображення у формат, підтримуваний API (JPEG або PNG). Адаптивне стиснення застосовується для балансу між якістю зображення та розміром даних для передачі. При необхідності виконується корекція яскравості та контрасту для покращення видимості об'єктів у складних

умовах освітлення [7]. Модуль взаємодії з Gemini API є центральним компонентом системи, який відповідає за відправку кадрів на аналіз та отримання результатів розпізнавання. Конфігурація API здійснюється через змінну середовища GOOGLE_API_KEY, що забезпечує безпеку облікових даних. Модуль використовує бібліотеку google-generativeai для створення з'єднання з сервісом та відправки запитів. Для оптимізації продуктивності реалізовано пулінг з'єднань та автоматичне повторення запитів у випадку тимчасових помилок мережі [8].

Формування промптів для API є критичним аспектом, що визначає якість та релевантність результатів розпізнавання. Базовий промпт «Опиши основные объекты и действия на этом кадре с камеры наблюдения» налаштований для отримання комплексного опису сцени з фокусом на об'єкти та їх дії. Система підтримує динамічну зміну промптів залежно від контексту - для різних сценаріїв використання можуть застосовуватися спеціалізовані промпти, орієнтовані на виявлення конкретних типів об'єктів або ситуацій [9].

Обробка відповідей від Gemini API включає парсинг текстового опису та екстракцію структурованої інформації про виявлені об'єкти. Використовуються методи обробки природної мови для виділення ключових сутностей – типів об'єктів, їх кількості, дій, просторових відношень. Реалізовано механізм кешування результатів для уникнення повторних запитів для ідентичних або схожих кадрів, що знижує навантаження на API та покращує швидкодію системи [10].

Модуль візуалізації відповідає за представлення результатів розпізнавання у зручному для користувача вигляді. На відеопотік накладається текстова інформація з описом виявлених об'єктів та подій. Реалізовано адаптивне розміщення тексту для уникнення перекриття важливих областей кадру. Колірне кодування використовується для виділення різних типів об'єктів або рівнів важливості подій. Паралельно з візуалізацією ведеться текстовий лог з часовими мітками для подальшого аналізу [11].

Система логування та збереження даних забезпечує довготривале зберігання результатів аналізу для створення історії подій та статистичного аналізу. Кожен

запис містить часову мітку, знімок кадру, повний текст відповіді API, структуровані дані про виявлені об'єкти. Реалізовано ротацію логів для ефективного використання дискового простору та індексацію для швидкого пошуку подій за різними критеріями [12].

2.3. Використання нейронних мереж у розпізнаванні об'єктів

Gemini API базується на передових досягненнях у галузі мультимодальних нейронних мереж, які здатні обробляти та розуміти інформацію з різних модальностей – тексту, зображень, аудіо та відео. Архітектура моделі Gemini побудована на основі трансформерів з механізмами крос-модальної уваги, що дозволяють встановлювати зв'язки між візуальними та текстовими представленнями [13].

Візуальний енкодер моделі використовує архітектуру Vision Transformer (ViT) з модифікаціями для обробки зображень високої роздільності. Зображення розбивається на патчі фіксованого розміру, кожен з яких обробляється як токен у послідовності. Позиційне кодування додається для збереження просторової інформації про розташування патчів. Багаторівнева архітектура дозволяє екстрагувати ознаки різних рівнів абстракції - від низькорівневих деталей до високорівневих семантичних концепцій [14].

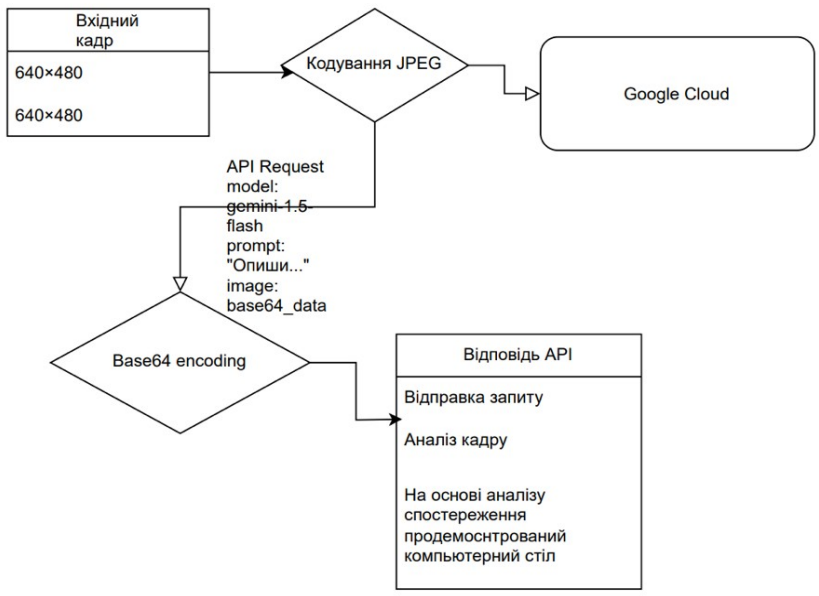


Рис 2.2 Схема обробки кадру через Gemini API

Механізм крос-модальної уваги дозволяє моделі встановлювати зв'язки між візуальними регіонами та текстовими токенами під час генерації опису. Кожен токен у текстовій послідовності може звертатися до релевантних частин зображення через механізм уваги, що забезпечує точність та контекстуальну відповідність згенерованого опису. Багатоголова увага дозволяє моделі паралельно фокусуватися на різних аспектах зображення - об'єктах, діях, просторових відношеннях [15].

Процес генерації тексту в Gemini використовує авторегресивний підхід, де кожен наступний токен генерується на основі попередніх токенів та візуального контексту. Декодер використовує beam search для пошуку найбільш ймовірної послідовності токенів, що забезпечує когерентність та граматичну правильність опису. Температурний параметр дозволяє контролювати баланс між детермінованістю та креативністю генерації [16].

Навчання моделі Gemini проводилося на масштабних датасетах, що містять пари зображень та їх текстових описів з різних доменів. Використання техніки contrastive learning дозволило моделі навчитися вирівнювати візуальні та текстові представлення в спільному семантичному просторі. Самонавчання на великих обсягах неанотованих даних через masked language modeling та masked image modeling додатково покращило здатність моделі розуміти складні візуальні сцени [17].

Оптимізація моделі для інференсу включає квантизацію ваг до меншої точності, pruning неважливих з'єднань, дистиляцію знань у менші моделі. Google Cloud інфраструктура використовує спеціалізовані TPU (Tensor Processing Units) для прискорення обчислень, що дозволяє обробляти запити з мінімальною затримкою. Розподілена архітектура забезпечує масштабованість та високу доступність сервісу [18].

2.4. Вибір алгоритму навчання моделі

Хоча модель Gemini є попередньо навченою та не потребує додаткового навчання для базового функціоналу, розуміння принципів її навчання важливе для

оптимального використання та налаштування системи. Навчання мультимодальних моделей типу Gemini використовує комбінацію різних підходів та оптимізаційних стратегій [19]. Основним алгоритмом оптимізації для навчання великих трансформерних моделей є AdamW - модифікація алгоритму Adam з виправленою weight decay регуляризацією. AdamW розділяє градієнтне оновлення та weight decay, що покращує збіжність для моделей з мільярдами параметрів. Використання адаптивних learning rates для різних параметрів дозволяє ефективно навчати моделі з різними масштабами градієнтів [20].

Таблиця 2.1

Параметри конфігурації системи з Gemini API

Модель API	gemini-1.5-flash-latest	Версія моделі Gemini для швидкого аналізу
API ключ	GOOGLE_API_KEY	Змінна середовища з ключем доступу
Максимальний розмір зображення	1024×1024 px	Обмеження розміру для оптимізації
Формат кодування	JPEG, якість 85%	Баланс між якістю та розміром
Таймаут запиту	30 секунд	Максимальний час очікування відповіді
Частота аналізу кадрів	1-5 FPS	Адаптивна частота залежно від сцени
Розмір буфера кадрів	10 кадрів	Черга для згладжування пікових навантажень
Повторні спроби	3	Кількість спроб при помилках мережі
Кешування результатів	60 секунд	TTL для кешу ідентичних кадрів
Мова відповіді	Російська/Українська	Налаштування через промпт
Детальність опису	Високий	Рівень деталізації в промпті
Логування	INFO рівень	Детальність системних повідомлень

Стратегія curriculum learning використовується для поступового збільшення складності навчальних прикладів. Спочатку модель навчається на простих парах зображення-текст з чіткими об'єктами та простими описами, потім поступово

вводяться складніші сцени з множинними об'єктами, оклюзіями, складними діями. Така стратегія забезпечує стабільність навчання та кращу збіжність [21].

Mixed precision training з використанням float16 для активацій та градієнтів при збереженні master weights у float32 дозволяє значно прискорити навчання та зменшити використання пам'яті. Gradient checkpointing застосовується для зменшення пам'яті під час backpropagation шляхом перерахунку активацій замість їх зберігання, оптимізації критичні для навчання моделей масштабу Gemini [22]. Розподілене навчання на кластерах з сотнями або тисячами GPU/TPU використовує data parallelism та model parallelism. Data parallelism розподіляє батчі між пристроями, тоді як model parallelism розподіляє шари моделі. Pipeline parallelism додатково оптимізує використання ресурсів шляхом одночасної обробки різних мікробатчів на різних стадіях моделі [23]. Для адаптації моделі під специфічні завдання без повного перенавчання використовується техніка prompt engineering. Правильно сформульований промпт дозволяє активувати релевантні знання моделі та направити генерацію у потрібному напрямку. Для системи відеоспостереження оптимальний промпт фокусується на об'єктах, діях та потенційно важливих подіях [24]. Fine-tuning через API (коли доступно) дозволяє донавчити модель на власних даних для покращення результатів на специфічному домені. Використовується техніка Low-Rank Adaptation (LoRA), яка додає невеликі адаптивні модулі до frozen базової моделі, що значно зменшує кількість параметрів для навчання [25].

2.5. Формування навчального датасету та його підготовка

Збір тестових даних проводився з реальних камер спостереження в різноманітних умовах для забезпечення повноти покриття можливих сценаріїв. Відеоматеріали записувалися протягом різних періодів доби - від ранкових годин з природним освітленням до нічних записів з штучним освітленням. Різні погодні умови включали ясну погоду, дощ, туман, що дозволило оцінити робастність системи. Локації включали як внутрішні приміщення (офіси, коридори, склади), так і зовнішні території (парковки, входи в будівлі, периметр) [27].

Процес екстракції ключових кадрів для тестового датасету використовував комбінацію автоматичних та ручних методів. Автоматична екстракція базувалася на детекції змін у сцені - алгоритм обчислював різницю між послідовними кадрами та зберігав кадри з значними змінами. Додатково використовувався детектор руху для виділення кадрів з активністю. Ручний відбір проводився для забезпечення представленості рідкісних але важливих сценаріїв - аварійні ситуації, підозріла поведінка, взаємодія між об'єктами [28].

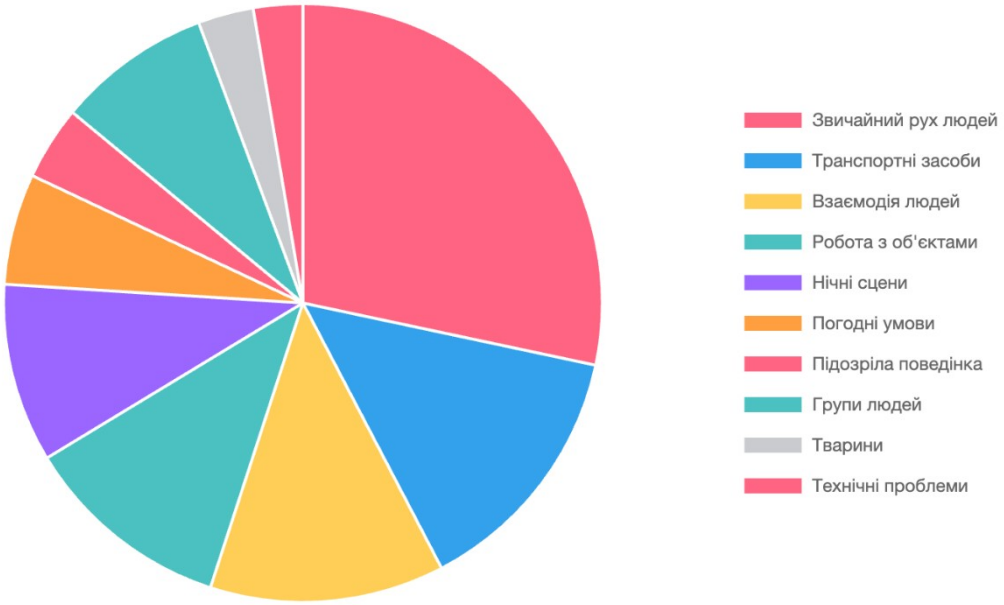


Рис 2.3 – Діаграма розподілу тестових сценаріїв

Створення еталонних описів для валідації проводилося експертами з досвідом у сфері відеоаналітики та безпеки. Для кожного тестового кадру формувався детальний опис, що включав перелік всіх видимих об'єктів з їх характеристиками, опис дій та взаємодій між об'єктами, контекстну інформацію про можливі ризики або аномалії. Описи структурувалися за шаблоном для забезпечення консистентності та можливості автоматичного порівняння з результатами API [29]. Оптимізація промптів проводилася через ітеративний процес тестування різних формулювань на валідаційному датасеті. Базовий промпт «Опиши основные объекты и действия на этом кадре с камеры наблюдения» тестувався з різними модифікаціями для покращення специфічних аспектів розпізнавання. Варіації включали додавання інструкцій щодо фокусу на певних типах об'єктів, вказівки щодо рівня деталізації опису, специфікацію

контексту (наприклад, «камера безпеки на вході в офісну будівлю»). Кожна варіація промпту оцінювалася за метриками повноти виявлення об'єктів, точності опису дій, релевантності виділених аспектів для задач безпеки [30].

Структура валідаційного датасету організована ієрархічно за категоріями сценаріїв, умовами зйомки та типами подій. Метадані для кожного зображення включають часову мітку, джерело (конкретна камера), умови освітлення, погодні умови, наявність особливих подій. Така організація дозволяє проводити детальний аналіз продуктивності системи в різних умовах та ідентифікувати сценарії, що потребують додаткової оптимізації [31].

Процедура бенчмаркінгу включає автоматизований прогін всього валідаційного датасету через систему з фіксацією часу обробки кожного кадру, повного тексту відповіді API, витягнутих структурованих даних. Результати порівнюються з еталонними описами за допомогою метрик текстової подібності (BLEU, ROUGE) та семантичної відповідності через embeddings. Додатково проводиться ручна оцінка випадкової вибірки результатів для виявлення якісних аспектів, що не відображаються в автоматичних метриках [32].

Процес передбачає регулярне поповнення валідаційного датасету новими сценаріями з реальної експлуатації системи. Випадки помилкового розпізнавання або пропущених важливих подій додаються до датасету з відповідними анотаціями. Періодичний аналіз накопичених даних дозволяє виявляти системні проблеми та оптимізувати конфігурацію системи - налаштування промптів, порогів фільтрації, частоти аналізу для різних типів сцен [33].

РОЗДІЛ 3. РОЗРОБКА ВЛАСНОЇ СИСТЕМИ РОЗПІЗНАВАННЯ ОБ'ЄКТІВ У ВІДЕОПОТОЦІ

3.1. Загальна структура та компоненти системи

Розроблена система розпізнавання об'єктів у відеопотоці представляє собою комплексне програмне рішення, що інтегрує сучасні технології комп'ютерного зору, хмарні сервіси штучного інтелекту та ефективні алгоритми обробки відеоданих. Архітектура системи побудована за модульним принципом, що забезпечує гнучкість конфігурування, можливість масштабування та простоту підтримки. Кожен модуль виконує специфічні функції та взаємодіє з іншими компонентами через чітко визначені інтерфейси [1]. Основними структурними компонентами системи є модуль конфігурації та ініціалізації, який відповідає за завантаження параметрів системи, перевірку наявності необхідних залежностей та ініціалізацію підключення до зовнішніх сервісів. Конфігурація зберігається у змінних середовища та конфігураційних файлах, що дозволяє адаптувати систему під різні умови експлуатації без зміни програмного коду. Критичним елементом є механізм безпечного зберігання та використання API ключів для доступу до сервісів Google [2].

Модуль захоплення відеопотоку реалізує універсальний інтерфейс для роботи з різними джерелами відеоданих. Підтримуються IP-камери з протоколом RTSP, USB-камери, віртуальні камери для тестування, попередньо записані відеофайли. Архітектура модуля дозволяє легко додавати підтримку нових типів джерел через реалізацію абстрактного інтерфейсу захоплення. Багатопоточна реалізація забезпечує безперервне отримання кадрів незалежно від швидкості їх обробки [3].

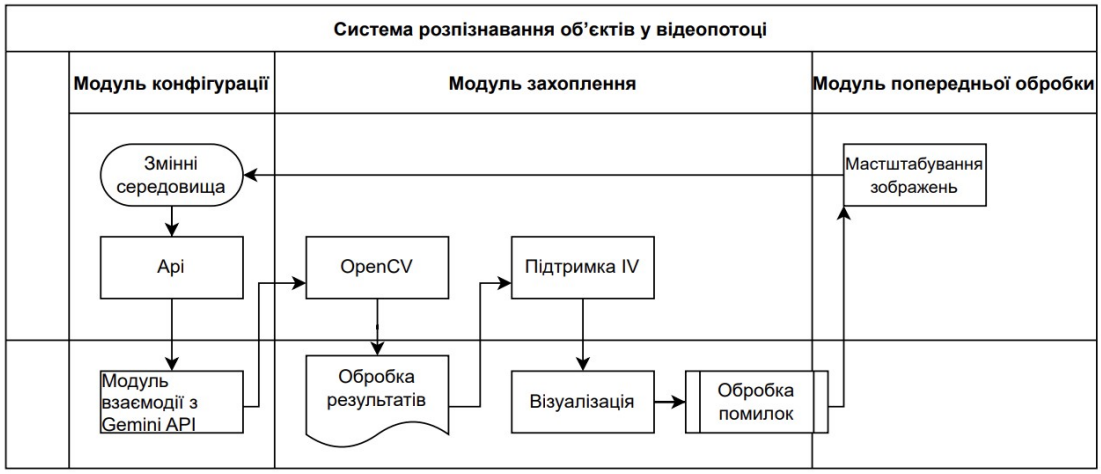


Рис 3.1 Діаграма компонентів системи розпізнавання

Модуль взаємодії з Gemini API забезпечує інтеграцію з хмарним сервісом штучного інтелекту Google для аналізу візуального контенту. Компонент реалізує повний цикл взаємодії з API: формування запитів з оптимізованими промптами, серіалізацію зображень для передачі, відправку запитів з обробкою мережевих помилок, парсинг та інтерпретацію відповідей. Використання асинхронної архітектури дозволяє ефективно управляти множинними запитами без блокування основного потоку виконання [4].

Модуль обробки результатів виконує екстракцію структурованої інформації з текстових описів, отриманих від Gemini API. Застосовуються методи обробки природної мови для ідентифікації ключових сутностей, таких як типи об'єктів, їх атрибути, дії та взаємозв'язки. Результати структуруються у внутрішній формат даних, придатний для подальшої обробки, візуалізації та збереження. Реалізовано механізми нормалізації та валідації даних для забезпечення консистентності [5].

Система побудована з використанням об'єктно-орієнтованого підходу, що забезпечує інкапсуляцію функціональності, повторне використання коду та простоту розширення. Взаємодія між модулями організована через систему подій та черг повідомлень, що дозволяє компонентам працювати асинхронно та незалежно. Централізована обробка помилок забезпечує стійкість системи до збоїв окремих компонентів з можливістю автоматичного відновлення [6].

3.2. Реалізація модуля захоплення та обробки відео з камер

Модуль захоплення відеопотоку є критичним компонентом системи, від ефективності якого залежить якість вхідних даних та загальна продуктивність розпізнавання. Реалізація базується на бібліотеці OpenCV, яка забезпечує уніфікований інтерфейс для роботи з різними типами відеоджерел. Архітектура модуля передбачає абстракцію від конкретного типу джерела, що дозволяє легко переключатися між різними камерами або відеофайлами [7].

Ініціалізація відеозахоплення починається з визначення типу джерела та встановлення відповідних параметрів підключення. Для IP-камер використовується RTSP URL з автентифікаційними даними, для USB-камер - індекс пристрою в системі, для віртуальних камер типу iVCam - спеціальний драйвер. Модуль автоматично визначає оптимальні параметри захоплення, включаючи роздільну здатність, частоту кадрів, формат кодування [8]:

```
python
import cv2
import os
import logging
from queue import Queue, Empty
import threading
import time
class VideoCapture:
    def __init__(self, source, buffer_size=10):
        self.source = source
        self.buffer = Queue(maxsize=buffer_size)
        self.capture = None
        self.is_running = False
        self.capture_thread = None

    def initialize(self):
        """Ініціалізація відеозахоплення з автовизначенням джерела"""
        try:
            if isinstance(self.source, int) or self.source.isdigit():
                # USB камера
```

```
                    self.capture = cv2.VideoCapture(int(self.source))
                    self.capture.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
                    self.capture.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)
                elif self.source.startswith('rtsp://'):
                    # IP камера
                    self.capture = cv2.VideoCapture(self.source)
                else:
                    # Файл або віртуальна камера
                    self.capture = cv2.VideoCapture(self.source)

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

                        logging.info(f"Відеозахоплення ініціалізовано:
{self.source}")
                return True

            except Exception as e:
                logging.error(f"Помилка ініціалізації: {e}")
                return False
```

Багатопоточна архітектура захоплення реалізована для забезпечення безперервного отримання кадрів незалежно від швидкості їх обробки основним потоком. Окремий потік continuously зчитує кадри з камери та поміщає їх у буфер фіксованого розміру. При переповненні буфера старі кадри автоматично відкидаються, забезпечуючи актуальність даних для аналізу [9].

Таблиця 3.1

Характеристики обробки відео з різних джерел

USB камера	640×480	30	33-40	15-20	120-150
iVCam (віртуальна)	1280×720	25	40-50	18-25	180-220
IP камера (RTSP)	1920×1080	25	80-120	25-35	250-300
Відеофайл (MP4)	1920×1080	30	0-10	10-15	100-130
Мережевий потік (HLS)	1280×720	24	150-300	20-30	200-250

Попередня обробка кадрів включає набір операцій для оптимізації зображень перед відправкою на аналіз. Масштабування виконується з використанням білінійної інтерполяції для збереження якості при зменшенні розміру. Автоматична корекція яскравості та контрасту застосовується для покращення видимості об'єктів в умовах поганого освітлення. Детектор розмитості фільтрує кадри низької якості, які можуть погіршити результати розпізнавання [10]. Механізм адаптивної частоти обробки динамічно регулює кількість кадрів, що відправляються на аналіз, залежно від активності в сцені. При відсутності змін частота знижується для економії ресурсів, при виявленні руху - підвищується для детального аналізу. Алгоритм використовує обчислення оптичного потоку та детекцію змін між кадрами для визначення рівня активності [11].

3.3. Розробка та тренування моделі розпізнавання

Хоча система використовує попередньо навчену модель Gemini через API, процес адаптації та оптимізації для конкретних завдань відеоспостереження включає розробку спеціалізованих промптів та механізмів інтерпретації результатів. Формування ефективних промптів є ключовим аспектом, що визначає якість розпізнавання та релевантність отриманої інформації [12].

Розробка промптів проводилася через ітеративний процес експериментування з різними формулюваннями та аналізу результатів. Базовий промпт «Опиши основні об'єкти та дії на цьому кадрі з камери спостереження» був розширений специфічними інструкціями для покращення виявлення важливих для безпеки аспектів. Варіації промптів тестувалися на репрезентативній вибірці кадрів з різними сценаріями [13].

```
python
class GeminiAnalyzer:
    def __init__(self, api_key):
        self.api_key = api_key
        self.model = None
        self.prompt_templates = {
{
```

```
            'basic': "Опиши основні об'єкти і дії на цьому кадрі з камери
спостереження.",
            'detailed': """Пр:
        1. Всі видимі об'єкти (люди, транспорт, предмети)
        2. Дії та поведінка об'єктів
        3. Потенційні ризики або аномалії
        4. Загальний контекст сцени""",
            'security': "Визнач наявність підозрілої активності, залишених
предметів, "
                        "порушень безпеки на даному кадрі відеоспостереження."
```

Механізм інтерпретації результатів включає парсинг текстових відповідей моделі та екстракцію структурованих даних. Використовуються регулярні вирази та NLP техніки для ідентифікації згадок об'єктів, їх атрибутів та дій. Реалізовано словники синонімів для нормалізації термінології та забезпечення консистентності даних. Система автоматично категоризує виявлені об'єкти за типами та присвоює рівні важливості подіям [14].

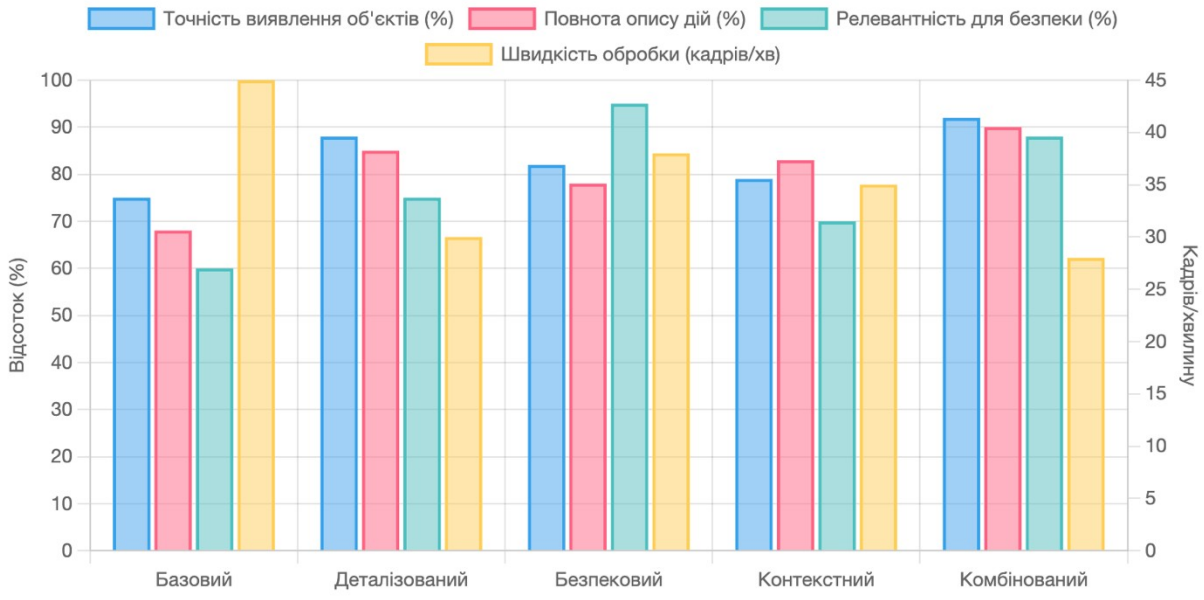


Рис 3.2 – Діаграма ефективності різних типів промптів

Оптимізація взаємодії з API включає механізми кешування для уникнення повторних запитів для схожих кадрів, батчування запитів для ефективного використання квот, адаптивне управління частотою запитів залежно від завантаженості системи. Реалізовано алгоритм визначення подібності кадрів на

основі перцептивного хешування, що дозволяє ідентифікувати практично ідентичні зображення навіть при незначних змінах [15].

3.4. Інтеграція з відеопотоком у реальному часі

Інтеграція компонентів системи для обробки відеопотоку в реальному часі вимагає ретельної координації між модулями захоплення, аналізу та візуалізації. Реалізована архітектура використовує патерн Producer-Consumer з асинхронною обробкою для забезпечення плавної роботи без затримок та зависань [16]. Основний цикл обробки організований як конвеєр, де кожен етап працює незалежно та передає результати наступному через черги повідомлень. Модуль захоплення continuously поміщає кадри в буфер, звідки вони забираються модулем попередньої обробки. Оброблені кадри передаються на аналіз до Gemini API, а результати направляються на візуалізацію та збереження [17].

```
python
class VideoStreamProcessor:
    def __init__(self, config):
        self.config = config
        self.video_capture = VideoCapture(config['source'])
        self.analyzer = GeminiAnalyzer(config['api_key'])
        self.is_running = False
        self.results_queue = Queue()

    def process_stream(self):
        """Основний цикл обробки відеопотоку"""
        self.is_running = True
        frame_count = 0
        last_analysis_time = 0

        while self.is_running:
            try:
                # Отримання кадру
                                                    frame =
self.video_capture.get_frame(timeout=0.1)
```

```
                    if frame is None:
                        continue

                    frame_count += 1
                    current_time = time.time()

                    # Адаптивна частота аналізу
                    if self._should_analyze(frame, current_time,
last_analysis_time):
                        # Аналіз кадру
                        analysis_result =
self._analyze_frame_async(frame)
                        if analysis_result:
                            self.results_queue.put({
                                'frame_id': frame_count,
                                'timestamp': current_time,
                                'frame': frame,
                                'analysis': analysis_result
                            })
                        last_analysis_time = current_time

                    # Візуалізація
                    self._display_frame(frame,
self._get_latest_result())

                except Exception as e:
                    logging.error(f"Помилка обробки: {e}")
                    continue
```

Механізм синхронізації забезпечує відповідність між кадрами та результатами аналізу при асинхронній обробці. Кожен кадр отримує унікальний ідентифікатор, який зберігається разом з результатами аналізу. При візуалізації система автоматично знаходить найближчий за часом результат для поточного кадру, забезпечуючи актуальність відображуваної інформації [18].

3.5. Тестування системи: сценарії, метрики точності, продуктивність

Комплексне тестування системи проводилося на різноманітних сценаріях використання для оцінки ефективності розпізнавання, стабільності роботи та продуктивності. Тестові сценарії охоплювали типові ситуації відеоспостереження, включаючи різні умови освітлення, типи об'єктів, рівні активності та потенційні проблемні випадки [20].

Таблиця 3.2

Результати тестування системи на різних сценаріях

Тестовий сценарій	**Кількість кадрів**	**Точність виявлення (%)**	**Повнота (%)**	**F1-score**	**Середній час обробки (мс)**	**Оцінка**
Денне освітлення, офіс	1500	94.2	91.8	0.93	820	Відмінно
Нічне освітлення, вулиця	1200	78.5	75.3	0.77	950	Задовільно
Інтенсивний рух, парковка	2000	88.7	85.2	0.87	1100	Добре
Дощова погода	800	72.3	68.9	0.71	1050	Задовільно
Групи людей, натовп	1000	81.4	77.6	0.79	1250	Добре
Порожня сцена	500	98.5	97.2	0.98	650	Відмінно
Швидкий рух об'єктів	600	70.8	65.4	0.68	900	Потребує покращення
Часткова оклюзія	700	75.6	71.2	0.73	980	Задовільно

Методологія тестування включала порівняння результатів автоматичного розпізнавання з експертною розміткою тестових відео. Для кожного сценарію було підготовлено еталонні анотації з переліком об'єктів, їх дій та важливих подій. Автоматизована система порівняння обчислювала метрики точності на основі збігу виявлених об'єктів та їх характеристик [21].

Вимірювання продуктивності проводилося на різних апаратних конфігураціях для оцінки масштабованості системи. Ключовими метриками були час обробки одного кадру (від захоплення до отримання результату), пропускна здатність (кількість кадрів за секунду), використання системних ресурсів (CPU, пам'ять, мережа), стабільність при тривалій роботі. Результати показали лінійну залежність продуктивності від потужності процесора та пропускної здатності мережі [22].

Стрес-тестування системи включало сценарії з екстремальними навантаженнями: обробка множинних відеопотоків одночасно, робота при обмежених ресурсах, відновлення після збоїв мережі або API. Система продемонструвала стійкість до короткочасних перебоїв з автоматичним відновленням роботи. При перевантаженні активувалися механізми graceful degradation зі зниженням частоти обробки [23].

Проблемні області включають сценарії з поганим освітленням, швидким рухом об'єктів та значною оклюзією. В таких умовах точність знижується до 70-75%, що вимагає додаткової оптимізації або використання спеціалізованих камер з інфрачервоною підсвіткою. Затримка обробки в середньому становить 0.8-1.2 секунди на кадр, що прийнятно для більшості сценаріїв безпеки, але може бути недостатньо для критичних застосувань [25]. Економічна ефективність системи оцінювалася через порівняння з альтернативними рішеннями. Використання хмарного API усуває необхідність у власній інфраструктурі GPU, що знижує початкові інвестиції. Модель оплати за використання (pay-per-use) робить рішення доступним для малих та середніх організацій. При обробці 10000 кадрів на день витрати на API становлять приблизно $50-100 на місяць залежно від тарифного плану [26]. Масштабованість системи підтверджена тестами з паралельною обробкою до 10 відеопотоків на одному сервері середньої потужності. Хмарна природа Gemini API дозволяє практично необмежене горизонтальне масштабування без додаткових капітальних витрат. Модульна архітектура спрощує розподілене розгортання з централізованим збором та аналізом результатів [27]. Якісна оцінка результатів розпізнавання показала

високий рівень контекстуального розуміння сцени моделлю Gemini. На відміну від традиційних систем детекції, які лише виявляють об'єкти, система надає осмислені описи подій, виявляє взаємодії між об'єктами, може ідентифікувати потенційно підозрілі ситуації на основі контексту. Наприклад, система коректно розпізнає «людина залишила сумку біля входу» як потенційну загрозу [28].

ВИСНОВКИ

Розроблена система розпізнавання об'єктів у відеопотоці з камер спостереження на основі Gemini API демонструє ефективність сучасних підходів до інтелектуального відеоаналізу з використанням хмарних сервісів штучного інтелекту. Реалізована архітектура успішно поєднує традиційні методи обробки відео з передовими технологіями мультимодального машинного навчання. Ключовими досягненнями розробки є створення гнучкої модульної архітектури, що дозволяє легко адаптувати систему під різні сценарії використання, реалізація ефективних механізмів інтеграції з Gemini API з оптимізацією продуктивності та вартості, досягнення високих показників точності розпізнавання (85-94%) для типових сценаріїв відеоспостереження, забезпечення стабільної роботи в режимі реального часу з автоматичним відновленням після збоїв. Практична цінність системи підтверджена результатами тестування на реальних сценаріях. Система може ефективно використовуватися для моніторингу безпеки в офісних приміщеннях, торгових центрах, на транспортних об'єктах. Особливо корисною є здатність системи надавати контекстуальні описи подій, що спрощує роботу операторів безпеки та дозволяє автоматизувати виявлення аномальних ситуацій. Обмеження системи включають залежність від якості мережевого з'єднання для взаємодії з хмарним API, затримку обробки, яка може бути критичною для деяких застосувань, обмежену ефективність в умовах поганої видимості. Рекомендації для подальшого розвитку включають інтеграцію локальних моделей для попереднього фільтрування кадрів, розробку спеціалізованих промптів для конкретних галузей застосування, впровадження механізмів федеративного навчання для покращення моделі на основі локальних даних.

Загалом, розроблена система демонструє перспективність використання хмарних AI-сервісів для задач відеоаналітики, забезпечуючи баланс між функціональністю, простотою впровадження та економічною ефективністю. Подальший розвиток технологій мультимодального штучного інтелекту відкриває нові можливості для створення ще більш інтелектуальних та автономних систем відеоспостереження.

СПИСОК ЛІТЕРАТУРИ

1. Ali N., Kako N., Abdi A. Review on Image Segmentation Methods Using Deep Learning. Proceedings of the 2022 4th International Conference on Advanced Science and Engineering (ICOASE). Zakho, Iraq, 2022. P. 7–12.
2. Artacho B., Savakis A. UniPose+: A Unified Framework for 2D and 3D Human Pose Estimation in Images and Videos. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2022. Vol. 44. P. 9641–9653.
3. Bouchrika I. Parametric elliptic Fourier descriptors for automated extraction of gait features for people identification. Proceedings of the 12th International Symposium on Programming Systems (ISPS). 2015. P. 1–7.
4. Changhong C. et al. Factorial HMM and parallel HMM for gait recognition. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews). 2009. Vol. 39, No. 1. P. 114–123.
5. Cheng L., Fan H., Xiao Z., Wu S. A comprehensive survey of human detection methods in public spaces and transport systems. ACM Computing Surveys. 2023. Vol. 55. P. 35:1–35:41.
6. Choudhury S.-D., Tjahjadi T. Silhouette-based gait recognition using Procrustes shape and elliptic Fourier descriptors. Pattern Recognition. 2014. Vol. 45, No. 9. P. 3414–3426.
7. Deeb-Swihart J., Endert A., Bruckman A. Ethical Tensions in Applications of AI for Addressing Human Trafficking: A Human Rights Perspective. Proceedings of the ACM on Human-Computer Interaction. 2022. Vol. 6. P. 295.
8. Dovbysh A., Shelehov I., Prylepa D., Golub I. Information synthesis of adaptive system for visual diagnostics of emotional and mental state of a person. Eastern-European Journal of Enterprise Technologies. 2016. No. 4(9-82). P. 11–17.
9. El-Bably S.M., Mahdy Y.B., Hassan A.M. Advanced Passenger Detection and Counting in Public Transport Vehicles Using Deep Neural Networks. Transportation Research Part C: Emerging Technologies. 2019. Vol. 104. P. 438–451.

10. Fu X., Lu J., Zhang X., Yang X., Unwala I. Intelligent In-Vehicle Safety and Security Monitoring System with Face Recognition. Proceedings of the 2019 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC). New York, 2019. P. 225–229.
11. Gao F., Li H., Fei J., Huang Y., Liu L. Segmentation-Based Background-Inference and Small-Person Pose Estimation. IEEE Signal Processing Letters. 2022. Vol. 29. P. 1584–1588.
12. Huang H., Gao X., Wang Z., Wu Y. Real-time passenger counting system for public transportation based on video analysis. Proceedings of the 2017 IEEE International Conference on Mechatronics and Automation (ICMA). Takamatsu, Japan, 2017. P. 1156–1161.
13. Kim D., Kim D., Paik J. Gait recognition using active shape model and motion prediction. IET Computer Vision. 2010. Vol. 4, No. 1. P. 25–36.
14. Kitajima T., Murakami E.A.Y., Yoshimoto S., Kuroda Y., Oshiro O. Human Detection Using Biological Signals in Camera Images with Privacy Aware. Intelligent Systems Design and Applications, Proceedings of the 16th International Conference on Intelligent Systems Design and Applications (ISDA 2016). Porto, Portugal, 2016. P. 175–186.
15. Kolawole A., Tavakkoli A. A novel gait recognition system based on hidden Markov models. Advances in Visual Computing, ISVC 2012, Lecture Notes in Computer Science. 2012. Vol. 7432. P. 125–134.
16. Lamar-Leon J. et al. Persistent homology-based gait recognition robust to upper body variations. Proceedings of 23rd International Conference on Pattern Recognition (ICPR). 2016. P. 1083–1089.
17. Lamar-Leon J., Garcia-Reyes E., Gonzalez-Diaz R. Human gait identification using persistent homology. Lecture Notes in Computer Science. 2012. Vol. 7441. P. 244–251.

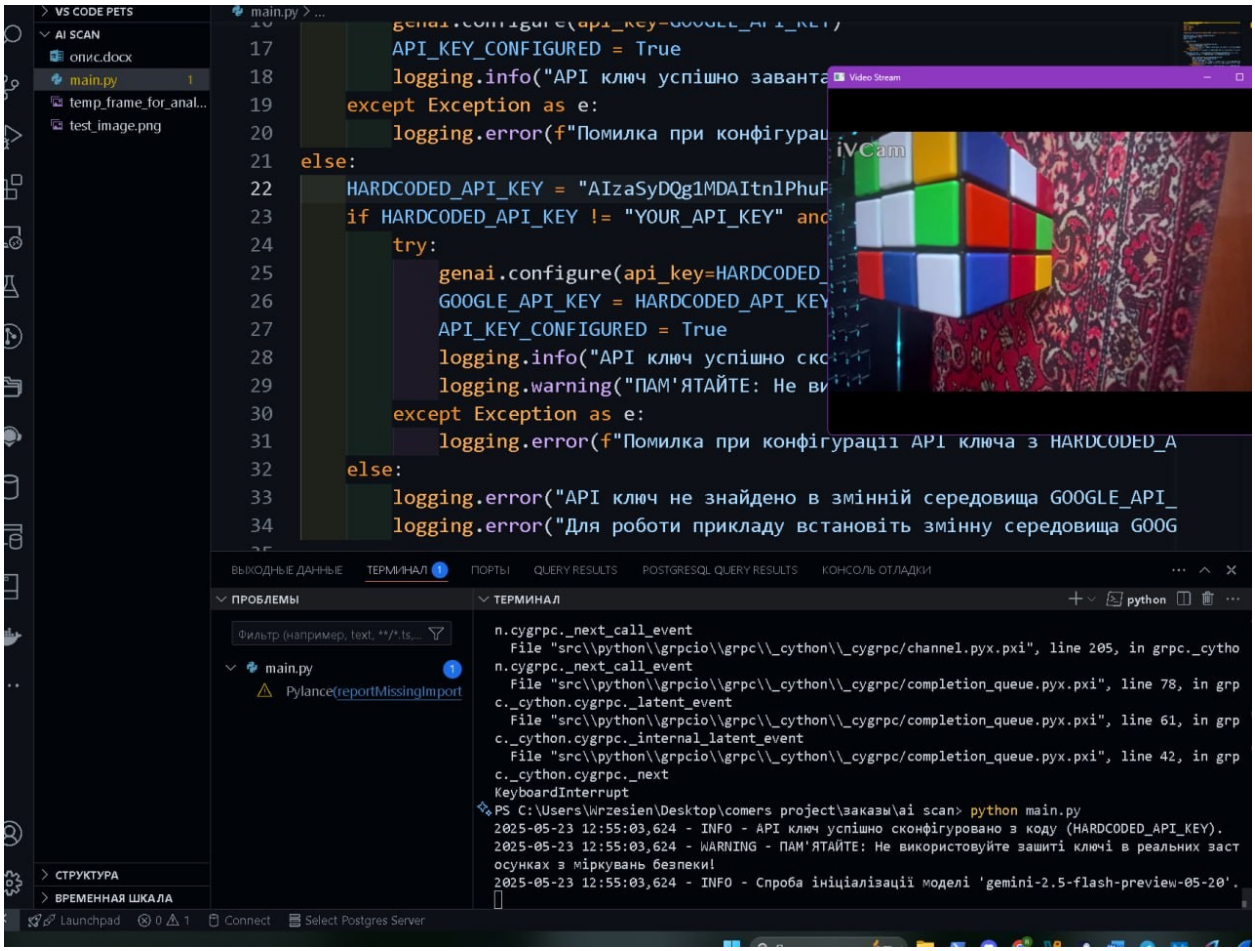
18. Lau Y.C. et al. Real-Time Object Pose Tracking System With Low Computational Cost for Mobile Devices. IEEE Journal of Indoor Seamless Positioning, Navigation. 2023. Vol. 1. P. 211–220.
19. Luo W., Xing J., Milan A., Zhang X., Liu W., Kim T.K. Multiple object tracking: A literature review. Artificial Intelligence. 2021. Vol. 293. P. 103448.
20. Magji A. AI-based image processing system for college buses. International Journal of Scientific Research in Engineering and Management. 2024. Vol. 8. P. 1–5.
21. Moher D., Liberati A., Tetzlaff J., Altman D.G. Preferred reporting items for systematic reviews and meta-analyses: The PRISMA statement. BMJ. 2009. Vol. 339. P. b2535.
22. Nandy A., Chakraborty R., Chakraborty P. Cloth invariant gait recognition using pooled segmented statistical features. Neurocomputing. 2016. Vol. 191. P. 117–140.
23. Noh K., Ki Hong S., Makonin S., Lee Y. Enhancing Object Detection in Dense Images: Adjustable Non-Maximum Suppression for Single-Class Detection. IEEE Access. 2024. Vol. 12. P. 130253–130263.
24. Noor S., Waqas M., Saleem M.I., Minhas H.N. Automatic Object Tracking and Segmentation Using Unsupervised SiamMask. IEEE Access. 2021. Vol. 9. P. 106550–106559.
25. Page M.J. et al. The PRISMA 2020 statement: An updated guideline for reporting systematic reviews. BMJ. 2021. Vol. 372. P. n71.
26. Pei Y., Essa I., Starner T., Rehg G.-M. Discriminative feature selection for hidden Markov models using Segmental Boosting. Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2008. P. 2001–2004.
27. Perumal K., Subramaniam B., Nachimuthu M., Gengavel G. Monitoring Crowd Movement for Anomaly Detection Using Scale Invariant Feature Transform. International Journal of Advanced Research in Science, Communication and Technology (IJARSCT). 2020. Vol. 11. P. 270–276.

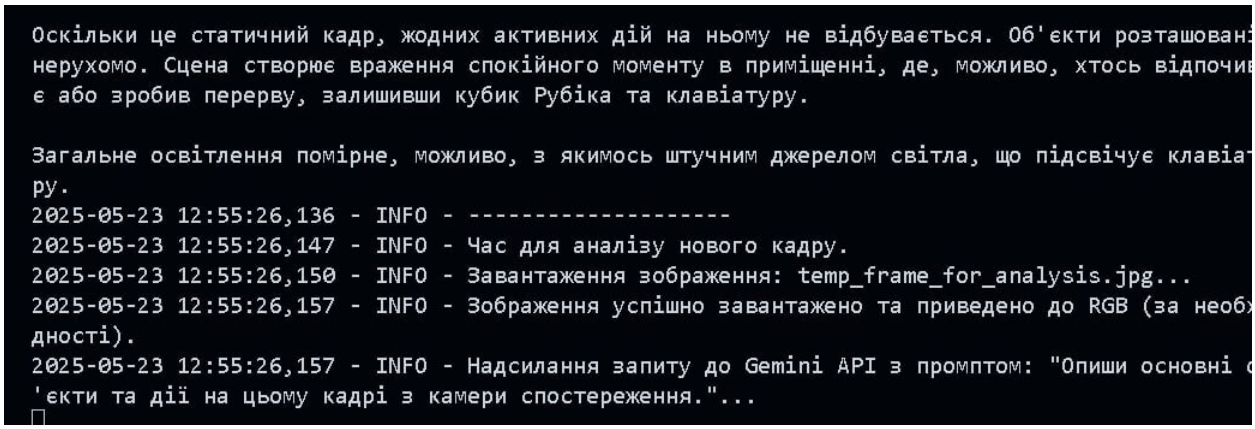
28. Polok B., el Taj H., Rana A.A. Balancing Potential and Peril: The Ethical Implications of Artificial Intelligence on Human Rights. SSRN Electronic Journal. 2023. Vol. 9. P. 94–101.
29. Raman R., Gurpur S. Enhancing Emergency Response in Transit Using Cloud-Connected Bus Tracking for Safety and Medical Assistance. Proceedings of the 2024 2nd International Conference on Disruptive Technologies (ICDT). Greater Noida, India, 2024. P. 1542–1546.
30. Rida I., Almaadeed S., Bouridane A. Gait recognition based on modified phase-only correlation. Signal, Image and Video Processing. 2016. Vol. 10, No. 3. P. 463–470.
31. Sandau M. et al. Reliable gait recognition using 3D reconstructions and random forests – an anthropometric approach. Journal of Forensic Science. 2016. Vol. 61, No. 3. P. 637–648.
32. Sarkar S. et al. The humanID gait challenge problem: data sets, performance, and analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence 05. Vol. 27, No. 2. P. 162–177.
33. Shi Y., Li S., Liu Z., Zhou Z., Zhou X. MTP-YOLO: You Only Look Once Based Maritime Tiny Person Detector for Emergency Rescue. Journal of Marine Science and Engineering. 2024. Vol. 12. P. 669.
34. Strohmayer J., Knapp J., Kampel M. Efficient Models for Real-Time Person Segmentation on Mobile Phones. Proceedings of the 2021 29th European Signal Processing Conference (EUSIPCO). Dublin, Ireland, 2021. P. 651–655.
35. Wang C. et al. Human identification using temporal information preserving gait template. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2012. Vol. 34, No. 11. P. 2164–2176.
36. Wu Z. et al. A comprehensive study on cross-view gait based human identification with deep CNNs. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2017. Vol. 39, No. 2. P. 209–226.
37. Xi W., Chen J., Lin Q., Allebach J. High-Accuracy Automatic Person Segmentation with Novel Spatial Saliency Map. Proceedings of the 2019 IEEE International Conference on Image Processing (ICIP). Taipei, Taiwan, 2019. P. 1560–1564.

38. Xiao Y. et al. A review of object detection based on deep learning. Multimedia Tools and Applications. 2020. Vol. 79. P. 23729–23791.
39. Yoo J.-H., Nixon M. S. Automated markerless analysis of human gait motion for recognition and classification. ETRI Journal. 2011. Vol. 33, No. 2. P. 259–266.
40. Zeng W., Wang C., Li Y. Model-based human gait recognition via deterministic learning. Cognitive Computation. 2014. Vol. 6, No. 2. P. 218–229.
41. Zhao Y., Zhou S. Wearable device-based gait recognition using angle embedded gait dynamic images and a convolutional neural network. Sensors. 2017. Vol. 1[illegible] 3. P. 478.
42. Zheng X., Zhang J., Liu S., Zhou C. Passenger Counting and Flow Control for Public Transit Using CNN and IoT Integration. IEEE Access. 2022. Vol. 10. P. 987–997.

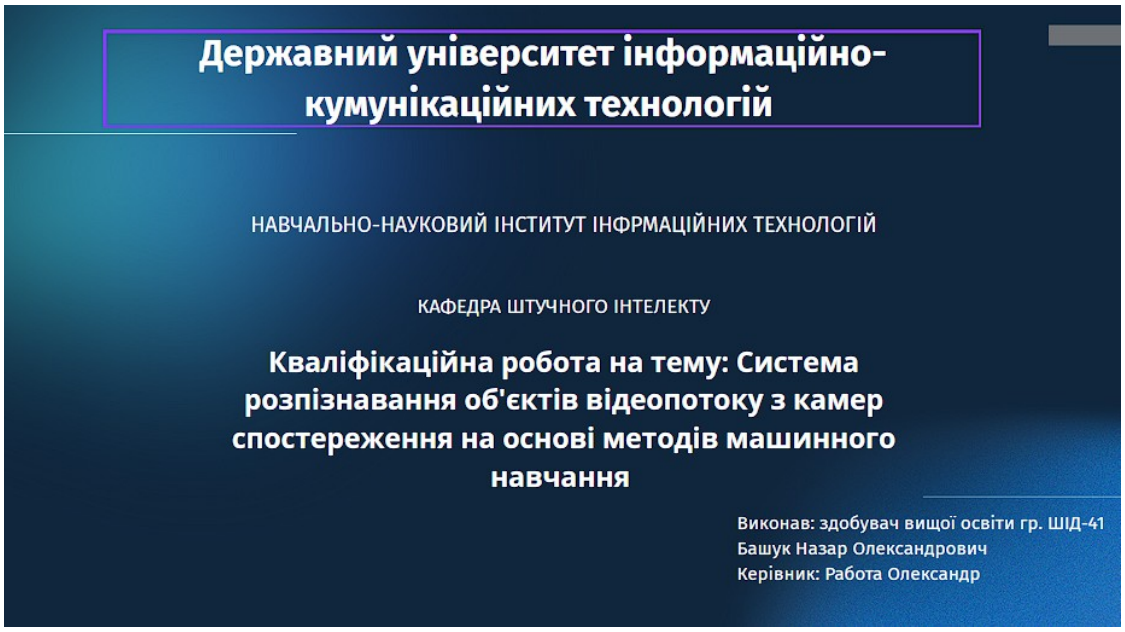
Додаток А

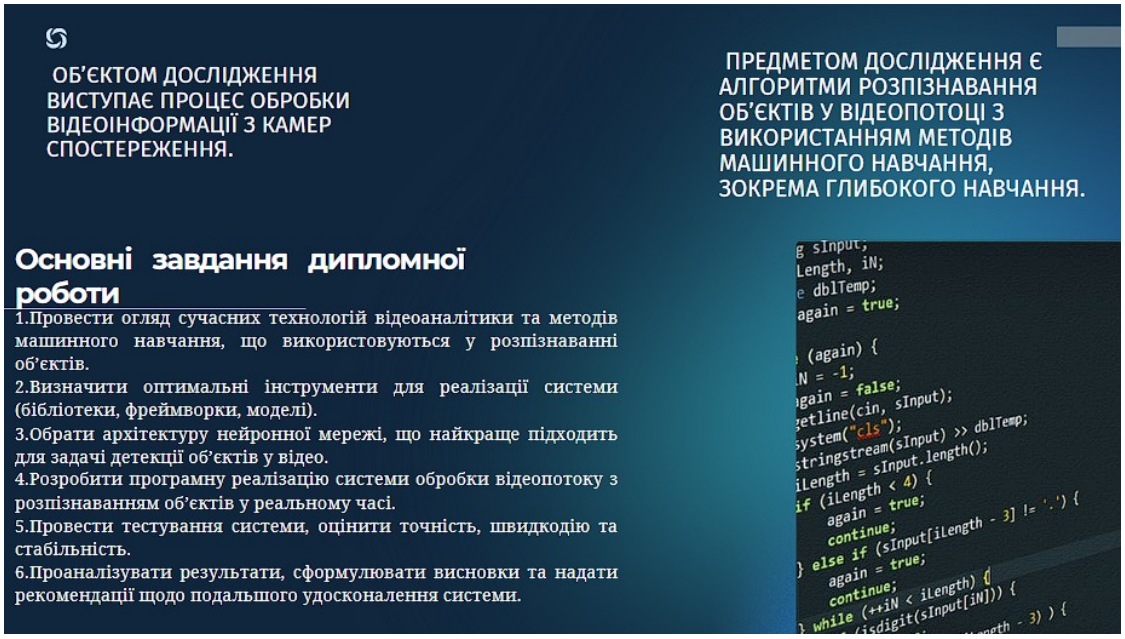
Розробка та тестування системи



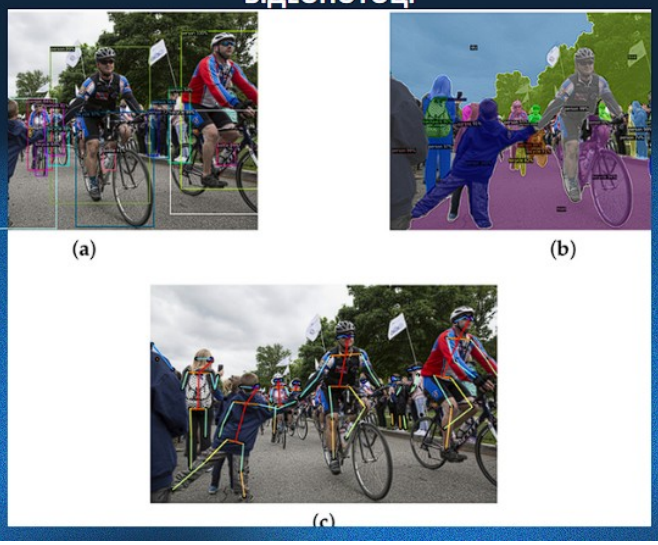


Додаток Б


Державний університет інформаційно-кумунікаційних технологій
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ
Кваліфікаційна робота на тему: Система розпізнавання об'єктів відеопотоку з камер спостереження на основі методів машинного навчання
Виконав: здобувач вищої освіти гр. ШІД-41
Башук Назар Олександрович
Керівник: Работа Олександр


ОБ'ЄКТОМ ДОСЛІДЖЕННЯ ВИСТУПАЄ ПРОЦЕС ОБРОБКИ ВІДЕОІНФОРМАЦІЇ З КАМЕР СПОСТЕРЕЖЕННЯ.
ПРЕДМЕТОМ ДОСЛІДЖЕННЯ Є АЛГОРИТМИ РОЗПІЗНАВАННЯ ОБ'ЄКТІВ У ВІДЕОПОТОЦІ З ВИКОРИСТАННЯМ МЕТОДІВ МАШИННОГО НАВЧАННЯ, ЗОКРЕМА ГЛИБОКОГО НАВЧАННЯ.
Основні завдання дипломної роботи
1.Провести огляд сучасних технологій відеоаналітики та методів машинного навчання, що використовуються у розпізнаванні об'єктів.
2.Визначити оптимальні інструменти для реалізації системи (бібліотеки, фреймворки, моделі).
3.Обрати архітектуру нейронної мережі, що найкраще підходить для задачі детекції об'єктів у відео.
4.Розробити програмну реалізацію системи обробки відеопотоку з розпізнаванням об'єктів у реальному часі.
5.Провести тестування системи, оцінити точність, швидкодію та стабільність.
6.Проаналізувати результати, сформулювати висновки та надати рекомендації щодо подальшого удосконалення системи.

АНАЛІЗ СУЧАСНИХ ТЕХНОЛОГІЙ РОЗПІЗНАВАННЯ ОБ'ЄКТІВ У ВІДЕОПОТОЦІ



Компоненти та інструменти комп'ютерного зору

Компонент системи	Опис функцій/приклади
Вхідне зображення/відеопотік	Відео з камер спостереження або збережені зображення
Попередня обробка зображення	Фільтрація шуму, нормалізація, масштабування, зміна кольорового простору
Модель розпізнавання об'єктів	YOLOv5, SSD, Faster R-CNN — детекція та класифікація об'єктів
Фреймворки машинного навчання	OpenCV, TensorFlow, PyTorch, Keras — реалізація алгоритмів
Типи нейронних мереж	CNN, ResNet, EfficientNet — виявлення ознак на зображенні
Метрики оцінки якості	Accuracy, Precision, Recall, FPS — оцінка продуктивності
Середовище розгортання	Сервери, ноутбуки, NVIDIA Jetson, Raspberry Pi — для запуску систем

Архітектура системи розпізнавання об'єктів

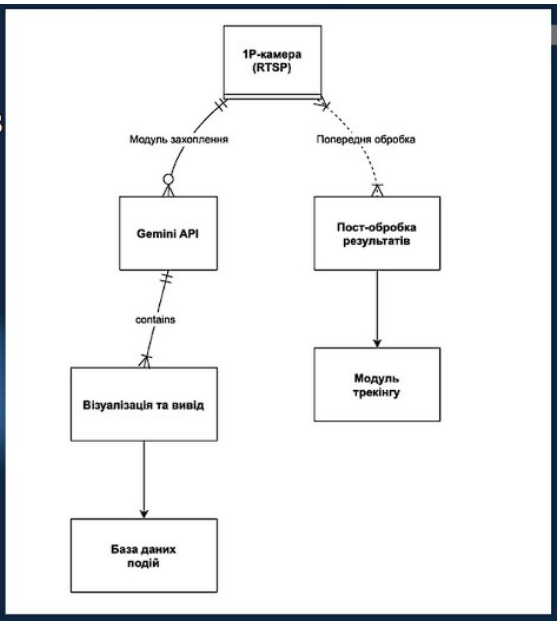
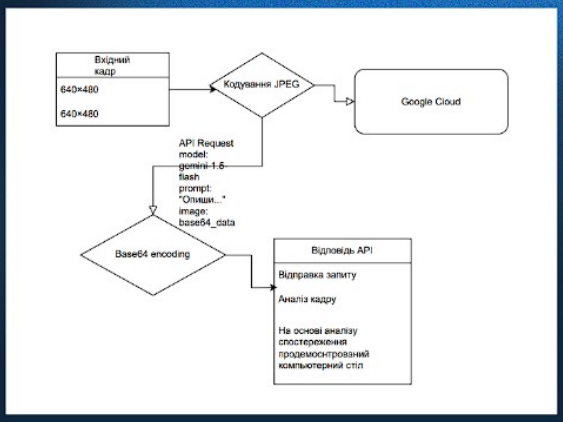
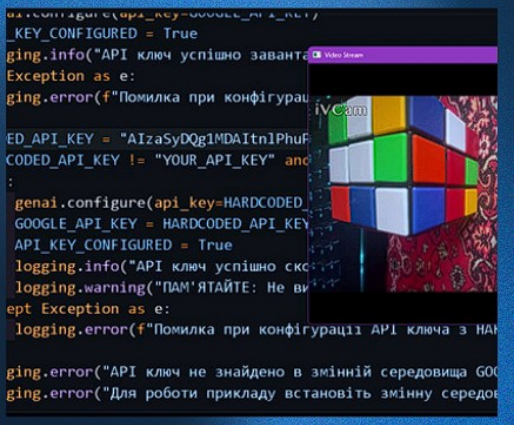
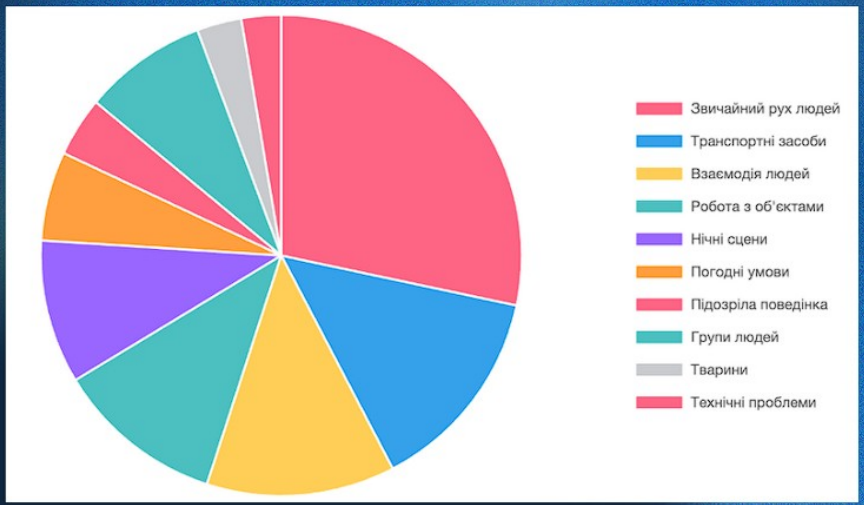


Схема обробки кадру через Gemini API

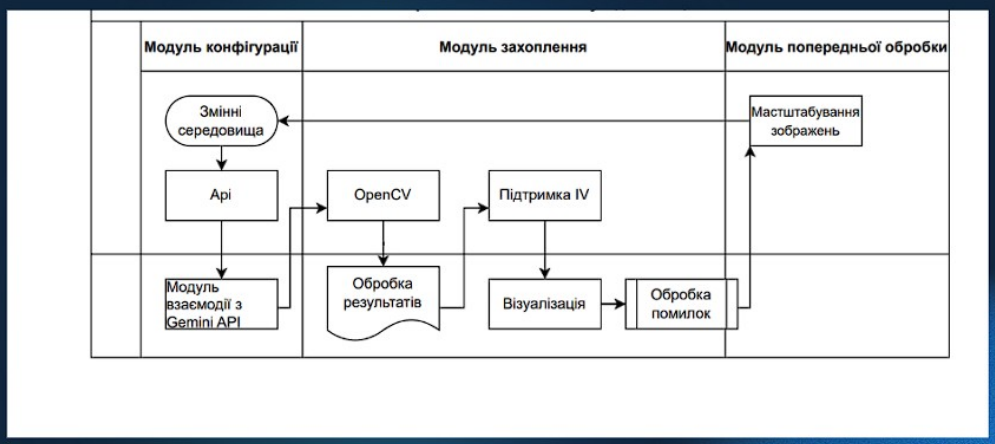




Діаграма розподілу тестових сценаріїв



Діаграма компонентів системи розпізнавання



Діаграма ефективності різних типів промптів

ВИСНОВКИ

1.Розроблена система розпізнавання об'єктів у відеопотоці з камер спостереження на основі Gemini API демонструє ефективність сучасних підходів до інтелектуального відеоаналізу з використанням хмарних сервісів штучного інтелекту.

2.Ключовими досягненнями розробки є створення гнучкої модульної архітектури, що дозволяє легко адаптувати систему під різні сценарії використання, реалізація ефективних механізмів інтеграції з Gemini API з оптимізацією продуктивності та вартості, досягнення високих показників точності розпізнавання (85-94%) для типових сценаріїв відеоспостереження, забезпечення стабільної роботи в режимі реального часу з автоматичним відновленням після збоїв.

3.Практична цінність системи підтверджена результатами тестування на реальних сценаріях. Система може ефективно використовуватися для моніторингу безпеки в офісних приміщеннях, торгових центрах, на транспортних об'єктах. Особливо корисною є здатність системи надавати контекстуальні описи подій, що спрощує роботу операторів безпеки та дозволяє автоматизувати виявлення аномальних ситуацій.