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

**Наконечний Микола Валентинович**

# Високопродуктивний модуль комп'ютерного зору для потокового відео / High-performance streaming video computer vision module

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

Кваліфікаційна робота

Виконав студент групи КН-42
М. В. Наконечний

Науковий керівник:
к.т.н., доцент, О.Р. Осолінський

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

Завідувач кафедри
_____ Н. В. Дзюбановська

**Тернопіль – 2026**

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

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Н.В. Дзюбановська
«____» _____20 р.

**ЗАВДАННЯ**
**НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ**
**НАКОНЕЧНОМУ Миколі Валентиновичу**
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: Високопродуктивний модуль комп'ютерного зору для потокового відео / High-performance streaming video computer vision module
керівник роботи О.Р. Осолінський, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від 01 грудня 2025 року № 894.
2. Строк подання студентом закінченої кваліфікаційної роботи 25 травня 2026 р.
3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.
4. Основні питання, які потрібно розробити:
- провести аналіз предметної області потокової відеоаналітики;
- провести огляд і аналіз існуючих рішень;
- сформувати постановку задачі;
- розробити загальну структуру модуля;
- розробити алгоритмічне забезпечення модуля;
- розробити інформаційне забезпечення;
- провести вибір програмних і апаратних засобів;
- реалізувати ключові модулі;
- реалізувати інтерфейс взаємодії з користувачем та провести оцінку продуктивності.

5. Перелік графічного матеріалу в роботі:
- розгорнута структурна схема високопродуктивного модуля комп'ютерного зору для потокового відео;
- діаграма проходження кадру та службових даних через модуль потокової відеоаналітики.

6. Консультанти розділів кваліфікаційної роботи

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

7. Дата видачі завдання 01 грудня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів кваліфікаційної роботи | Строк виконання етапів кваліфікаційної роботи | Примітка |
|---|---|---|---|
| 1 | Огляд літературних джерел відповідно до предметної області та складання плану роботи | до 01.01.2026 р. | |
| 2 | Написання 1 розділу кваліфікаційної роботи | до 01.02.2026 р. | |
| 3 | Написання 2 розділу кваліфікаційної роботи | до 15.03.2026 р. | |
| 4 | Написання 3 розділу кваліфікаційної роботи | до 01.05.2026 р. | |
| 5 | Представлення попереднього варіанту кваліфікаційної роботи, перевірка та внесення змін керівником | до 15.05.2026 р. | |
| 6 | Опрацювання зауважень та представлення завершеного варіанту кваліфікаційної роботи. Підготовка супроводжуючих документів | до 25.05.2026 р. | |
| 7 | Перевірка кваліфікаційної роботи на оригінальність тексту | до 30.05.2026 р. | |
| 8 | Оформлення кваліфікаційної роботи та отримання допуску до захисту | до 10.06.2026 р. | |
| 9 | Подання кваліфікаційної роботи до захисту на засіданні атестаційної комісії | до 10.06.2026 р. | |

Студент_____ М.В. Наконечний
( підпис ) (прізвище та ініціали)

Керівник кваліфікаційної роботи_____ О.Р. Осолінський
( підпис ) (прізвище та ініціали)

# АНОТАЦІЯ

Пояснювальна записка до кваліфікаційної роботи: 64 сторінка, 19 ілюстрацій, 3 таблиці, 4 додатки, 23 використані джерела.

Метою кваліфікаційної роботи є розробка високопродуктивного модуля комп'ютерного зору для потокового відео, орієнтованого на використання в edge-середовищі та на платформі Nvidia Jetson. В роботі розглянуто предметну область потокової відеоаналітики, виконано аналіз існуючих edge-рішень, сформульовано вимоги до модуля та обґрунтовано доцільність використання платформи Nvidia Jetson, проміжного формату ONNX і засобу оптимізації TensorRT.

В роботі використано методи системного та структурно-функціонального аналізу, алгоритмічного моделювання, комп'ютерного зору, глибокого навчання, структурного проектування програмних систем та експериментального оцінювання продуктивності.

Розроблено загальну структуру модуля, алгоритмічне забезпечення, інформаційне забезпечення для опису вхідних і вихідних даних, а також програмно-технологічний стенд з web-інтерфейсом, модуль завантаження відео, модуль роботи з відеоджерелом, модуль обробки кадрів, засоби візуалізації.

Під час реалізації використано Linux-середовище, мову Python, фреймворк Flask, бібліотеки OpenCV, NumPy та PyYAML.

Практичне значення роботи полягає у створенні працездатного прототипу системи потокової відеоаналітики, який може бути використаний як основа для систем відеоспостереження, транспортного моніторингу, контролю виробничих процесів та інших прикладних edge-рішень.

Ключові слова: ПОТОКОВА ВІДЕОАНАЛІТИКА, КОМП'ЮТЕРНИЙ ЗІР, NVIDIA JETSON, EDGE-ОБРОБКА, ONNX, TENSORRT, ВИЯВЛЕННЯ ОБ'ЄКТІВ, ВІДЕОПОТІК, WEB-ІНТЕРФЕЙС, ПРОДУКТИВНІСТЬ, АНОТОВАНЕ ВІДЕО, МЕТРИКИ.

# ANNOTATION

Explanatory note to the qualification thesis: 64 pages, 19 figures, 3 tables, 4 appendices, 23 references.

The aim of the qualification thesis is to develop a high-performance computer vision module for streaming video, intended for use in an edge environment and on the Nvidia Jetson platform. The work examines the domain of streaming video analytics, analyzes existing edge solutions, formulates the requirements for the module, and substantiates the feasibility of using the Nvidia Jetson platform, the ONNX intermediate format, and the TensorRT optimization tool.

The work employs methods of system and structural-functional analysis, algorithmic modeling, computer vision, deep learning, structural design of software systems, and experimental performance evaluation.

The general structure of the module, the algorithmic support, the information support for describing input and output data, as well as a software and technology testbed with a web interface, a video upload module, a video source handling module, a frame processing module, and visualization tools have been developed.

During implementation, a Linux environment, the Python programming language, the Flask framework, and the OpenCV, NumPy, and PyYAML libraries were used.

The practical significance of the work lies in the creation of a functional prototype of a streaming video analytics system that can be used as a basis for video surveillance systems, transport monitoring, production process control, and other applied edge solutions.

Keywords: STREAMING VIDEO ANALYTICS, COMPUTER VISION, NVIDIA JETSON, EDGE PROCESSING, ONNX, TENSORRT, OBJECT DETECTION, VIDEO STREAM, WEB INTERFACE, PERFORMANCE, ANNOTATED VIDEO, METRICS.

ЗМІСТ

# ВСТУП

Сучасний етап розвитку інформаційних технологій характеризується стрімким зростанням обсягів мультимедійних даних, що генеруються камерами спостереження, промисловими системами контролю, транспортною інфраструктурою, безпілотними платформами та побутовими пристроями. Значна частина цих даних представлена у вигляді відеопотоків, які містять велику кількість корисної інформації про об’єкти, події та зміни середовища. Це зумовлює активний розвиток систем потокової відеоаналітики, що дозволяють автоматично виявляти об’єкти, відстежувати їхню поведінку, класифікувати сцени та формувати результати, придатні для подальшого прийняття рішень.

Поширення методів комп’ютерного зору та глибокого навчання розширило можливості аналізу відеоданих.

Такі системи знаходять застосування у сферах громадської безпеки, міського моніторингу, транспортної аналітики, промислового контролю, охорони периметра, логістики та сервісних робототехнічних рішень. В практичних сценаріях це означає потребу не просто передавати відео, а безперервно обробляти його в режимі, близькому до реального часу, визначати розташування об’єктів, формувати їхні координати, оцінювати достовірність виявлення та передавати підготовлені результати до зовнішніх систем.

Але зростання кількості камер і збільшення роздільної здатності відео призводять до різкого підвищення вимог до пропускної здатності мережі, обчислювальних ресурсів і затримки обробки. Передавання повного відеопотоку до віддалених серверів або хмарних платформ не завжди є доцільним, оскільки воно збільшує латентність, навантажує канали зв’язку та знижує придатність системи до автономної роботи.

Тому одним із головних напрямів розвитку сучасних рішень є підхід на граничних обчисленнях, за якого основна обробка виконується безпосередньо поблизу джерела даних. Така організація дає змогу зменшити затримку, скоротити

обсяг передаваних даних, підвищити стійкість системи до втрати зв’язку та покращити оперативність реагування.

Для реалізації подібних систем важливу роль відіграють спеціалізовані апаратні платформи, орієнтовані на виконання задач комп’ютерного зору на периферії мережі. Одним із найбільш придатних класів таких рішень є пристрої сімейства Nvidia Jetson, які поєднують компактний форм-фактор, підтримку GPU-прискорення та можливість запуску моделей глибокого навчання безпосередньо на edge-вузлі. Але навіть за наявності апаратного прискорення ефективність системи визначається не лише вибором моделі, а і правильною організацією всього конвеєра обробки: від приймання відеопотоку та декодування кадрів до попередньої обробки, виконання моделі, постобробки та формування кінцевих результатів.

Крім того великий вплив мають засоби оптимізації виконання моделі. Для практичного використання на платформі Nvidia Jetson важливо забезпечити сумісність між середовищем навчання та середовищем розгортання, а також мінімізувати витрати ресурсів під час виконання. Для цього доцільним є використання проміжного формату ONNX для перенесення моделі та TensorRT як засобу апаратно-орієнтованої оптимізації її виконання. Це дає можливість підвищити частоту обробки кадрів, зменшити затримку та ефективніше використовувати ресурси GPU, CPU та пам’яті.

Актуальність теми полягає в необхідності розробки високопродуктивного модуля комп’ютерного зору для потокового відео, який би забезпечував обробку відеопотоку в режимі, близькому до реального часу, був придатним до роботи на edge-платформі Nvidia Jetson і підтримував оптимізацію виконання моделі для досягнення належної продуктивності. Такий модуль має поєднувати засоби приймання та декодування відео, підготовки кадрів, запуску моделі виявлення об’єктів, формування результатів та їх подальшого відображення, збереження або передавання до зовнішніх сервісів.

Для досягнення поставленої мети необхідно розв’язати такі завдання:

– дослідити предметну область потокової відеоаналітики та проаналізувати існуючі підходи до побудови edge-систем комп’ютерного зору;

– виконати огляд існуючих програмних і програмно-апаратних рішень для аналізу потокового відео;

– сформувати вимоги до розроблюваного модуля та визначити критерії оцінювання його ефективності;

– розробити загальну структуру проектованого модуля для приймання, обробки та аналізу відеопотоку;

– сформувати алгоритмічне та інформаційне забезпечення модуля;

– обґрунтувати вибір програмних і апаратних засобів реалізації;

– реалізувати модуль виконання моделі виявлення об’єктів на платформі Nvidia Jetson;

– забезпечити оптимізацію виконання моделі за допомогою TensorRT;

– провести тестування розробленого рішення та оцінити його продуктивність за показниками швидкодії, затримки, точності та використання ресурсів.

Об’єктом дослідження є процес потокової відеоаналітики в edge-системах комп’ютерного зору.

Предметом дослідження є методи, алгоритми та програмно-технологічні засоби обробки потокового відео.

В роботі використано методи комп’ютерного зору, глибокого навчання, аналізу відеопотоків, структурного та алгоритмічного проектування програмних систем, а також методи експериментального оцінювання продуктивності програмно-апаратних рішень.

Результати кваліфікаційної роботи апробовані та опубліковані в матеріалах V Всеукраїнська науково-практичної конференції Інтелектуальні комп’ютерні системи та мережі, Західноукраїнський національний університет, факультет комп’ютерних інформаційних технологій, кафедра комп'ютерної інженерії 20 травня.

Практичне значення одержаних результатів полягає у створенні модуля, який може бути використаний як основа для систем відеоспостереження, транспортного моніторингу, контролю виробничих процесів, аналітики подій та інших прикладних edge-рішень, де важливими є швидка реакція, автономність роботи та ефективне використання обчислювальних ресурсів.

# 1 СТАН І АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1 Опис предметної області потокової відеоаналітики

Потокова відеоаналітика є напрямом комп'ютерного зору, в межах якого відеодані аналізуються безперервно під час їх надходження від камери або іншого джерела. На відміну від обробки окремих статичних зображень, тут важливим є розпізнавання об'єктів в конкретному кадрі та врахування часової послідовності подій, зміни сцени та поведінки об'єктів у динаміці. Тому відеоаналітика перетворює неструктурований відеопотік у семантично значиму інформацію, придатну для автоматизованого прийняття рішень в системах безпеки, транспорту, промисловості, міського моніторингу та інших прикладних сферах [1, 2].

До базових задач комп'ютерного зору, що використовуються в потоковій відеоаналітиці, належать детектування об'єктів, класифікація, сегментація, відстеження рухомих цілей, розпізнавання символів і подальший аналіз поведінки об'єктів у часі. В роботі [3] показано, що відеопотік з систем спостереження є типовим джерелом даних для таких задач, а самі алгоритми мають бути швидкими й надійними, щоб уникати хибних спрацювань. Окремо написано, що для динамічних сцен важливим стає виявлення об'єкта та його супроводження між кадрами, оскільки саме траєкторія руху дозволяє далі виконувати ідентифікацію, класифікацію та інтерпретацію подій.

Типові сценарії потокової відеоаналітики охоплюють детекцію транспортних засобів і пішоходів, багатооб'єктне відстеження, підрахунок людей, класифікацію типів об'єктів, виявлення аномалій, аналіз подій та поведінки. В роботі [2] такі застосування прямо пов'язуються з smart city, безпекою, транспортним моніторингом, рятувальними задачами та інтерактивними системами. Типову схему взаємодії покадрової та міжкадрової обробки в системах потокової відеоаналітики показано на рисунку 1.1.

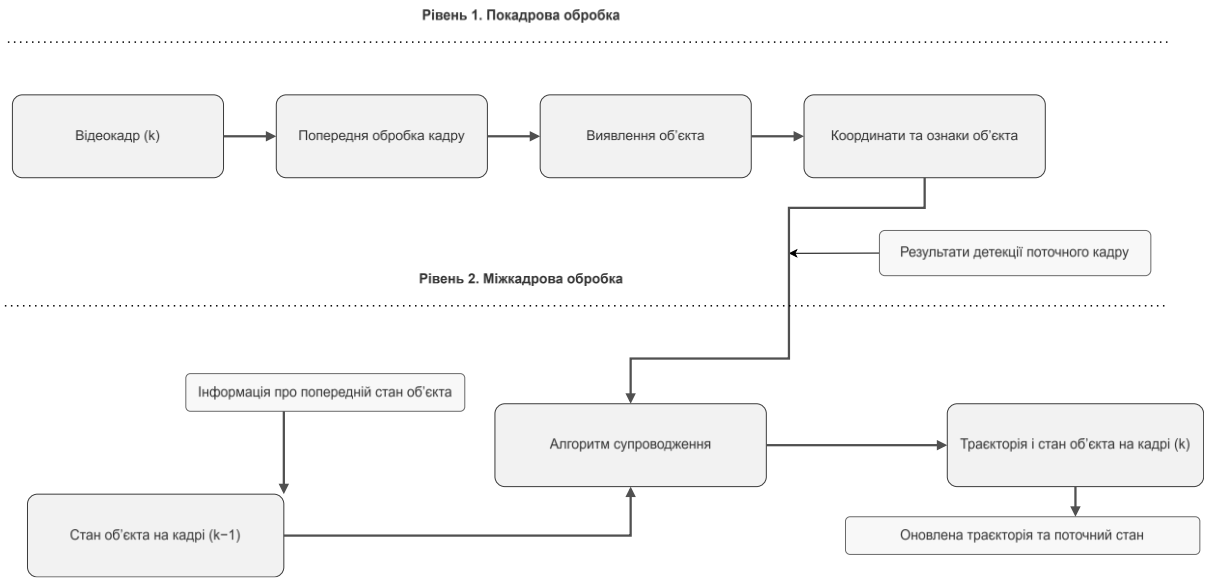



Рисунок 1.1 – Структура алгоритму «виявлення до супроводження» з розділенням покадрової та міжкадрової обробки

В практичних роботах ці сценарії конкретизуються: в задачі пасажиропідрахунку обробляється потік із верхньої fisheye-камери, у роботі для БПЛА — здійснюються детекція і трекінг цілей у реальному часі, а в роботі з відеоаномаліями — виявляються події, що відхиляються від нормального сценарію. Узагальнений процес створення та впровадження моделей штучного інтелекту для задач відеоаналітики наведено на рисунку 1.2.


Рисунок 1.2 –Процес розробки AI моделі

Особливістю є перехід від централізованої хмарної обробки до edge-підходу, коли значна частина аналітики виконується ближче до джерела даних. Це пояснюється тим, що передавання всього відеопотоку в хмару створює великі затримки, перевантажує мережу та ускладнює дотримання вимог конфіденційності. В роботах підкреслюється, що обробка біля камери або на спеціалізованому пристрої дозволяє зменшити latency, скоротити трафік і підвищити автономність системи. Дослідження [2]. також показує, що edge-архітектури можуть бути виділеними, спільними або ієрархічними, а вибір структури залежить від кількості камер, вимог до швидкості реакції та способу розподілу обчислень між камерою, edge-вузлом і хмарою. В статті [1] додатково показується, що гранична обробка фактично вимагає переосмислення алгоритмів: необхідні моделі меншого розміру, квантування, стиснення, адаптивне керування складністю та орієнтація на автономну роботу в умовах обмежених ресурсів.

Для потокової відеоаналітики критичними є обмеження реального часу. Сюди входить частота обробки кадрів, або FPS, від якої залежить плавність спостереження та здатність системи фіксувати швидкі зміни сцени. В роботі [4] наголошено, що висока швидкість обробки є необхідною для короткочасних подій, оскільки малі інтервали між кадрами дають більше даних для коректного реагування. Також показано, що оптимізація моделі через TensorRT і застосування DeepStream збільшили швидкість YOLOv4 на Nvidia Jetson Orin AGX приблизно з 9,6 до 30 FPS, одночасно зменшивши споживання GPU, CPU та пам’яті, що важливо для edge-пристроїв. Крім того на процес впливає затримка прийняття рішення. В роботі [5] для задачі виявлення аномалій показано, що реальний практичний сенс має висока точність та здатність приймати рішення за короткий інтервал часу. Запропонований метод досягав рішення за 6,4 с, тоді як конкуруючі підходи потребували до 273 с. Ще значення мають пропускна здатність мережі та енергоспоживання. За даними [1], потік від 4K-камери створює дуже великі обсяги даних, а для систем із багатьма камерами це стає критичним навантаженням на мережу. Тому для edge-систем важливо не лише виконати інференс, а s мінімізувати обсяг передаваних даних. Це підтверджується в роботі [6] де

фільтрація кадрів безпосередньо на стороні камери дала змогу відсіювати значну частину кадрів, економити смугу пропускання і знижувати затримку відповіді.

Тому завдання відеоаналітики поєднує задачі детекції, відстеження, класифікації та аналізу подій у безперервному відеопотоці з вимогами до реального часу, малої затримки, економного використання мережевих і енергетичних ресурсів та адаптації до edge-середовищ. Ці характеристики визначають напрям розвитку сучасних модулів комп'ютерного зору для потокового відео, зокрема рішень на Nvidia Jetson із прискореним інференсом і подальшою оптимізацією під обмежені апаратні ресурси.

## 1.2 Огляд і аналіз існуючих рішень

Існуючі рішення у сфері потокової відеоаналітики на edge можна поділити на кілька груп: готові edge-фреймворки для сервісної обробки відео, програмно-апаратні рішення на базі Nvidia Jetson, інструменти для перенесення моделей між фреймворками та засоби оптимізації інференсу.

Одним з аналогів є EdgeEye [7], що позиціонується як edge-сервісний фреймворк для відеоаналітики в реальному часі. Його перевага полягає в тому, що він надає високорівневий API і приховує деталі конкретних deep learning-фреймворків, даючи змогу прикладним сервісам працювати через узгоджений інтерфейс. В роботі описано використання WebSocket API, HTTP API, модельного репозиторію, оптимізатора моделей і конвеєрів обробки GStreamer. Такий підхід підходить для розширюваності, повторного використання компонентів та інтеграції кількох типів задач, таких як детекції, верифікації обличь або аналізу активності. Але недоліком є те, що для EdgeEye потрібен edge-сервер з високопродуктивними ресурсами, а реалізація окремих елементів залежить від конкретних засобів інференсу, таких як TensorRT, Крім того деколи потрібно реалізація додаткових plugin-шарів для непідтримуваних операцій. Компонентну структуру фреймворку EdgeEye та взаємодію його основних модулів наведено на рисунку 1.3.

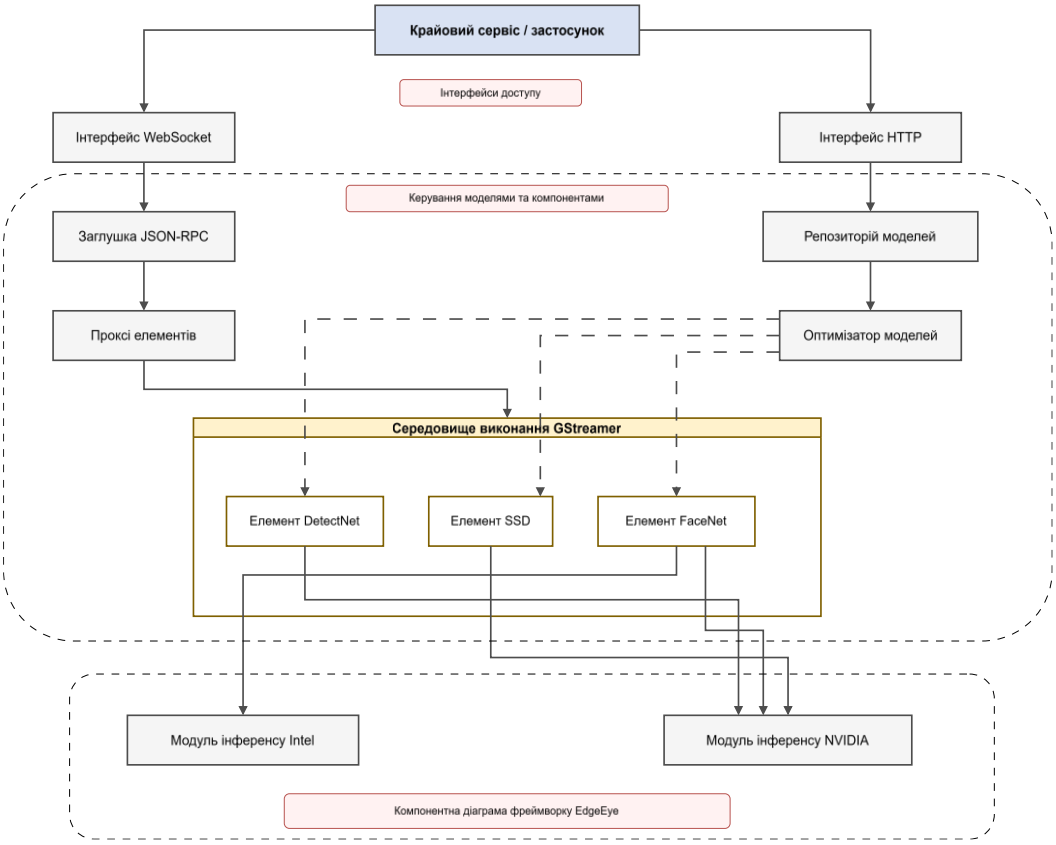



Рисунок 1.3 – Компонентна діаграма фреймворку EdgeEye

Інший клас рішень представлено в роботі [8], де використано NVIDIA DeepStream, Simple Realtime Server та Docker/Docker Compose на Jetson Xavier NX. Це вже ближче до практичної архітектури edge-відеоаналітики, орієнтованої на потокове відео і браузерний доступ до результатів. Перевагою такого рішення є використання готової схеми обробки DeepStream на базі GStreamer-плагінів: декодування потоку, мультиплексування, первинна детекція, трекінг, вторинна класифікація, відображення та передавання метаданих. Додатково контейнеризація полегшує розгортання і відтворюваність середовища. Крім того в роботі зазначається, що продуктивність і є близькою до запуску на фізичній машині, а WebRTC є кращим за HLS для передавання відео з малою затримкою. Крім того є недоліки: з ростом кількості одночасних підключень продуктивність погіршується, а WebRTC дає вищі вимоги до CPU, ніж HLS чи RTMP. Тобто, це рішення підходить для демонстрації того, як може виглядати повноцінний edge-сервіс з web-інтерфейсом, але має апаратні обмеження для масштабування. Загальну

архітектуру системи потокової відеоаналітики на основі NVIDIA DeepStream показано на рисунку 1.4.

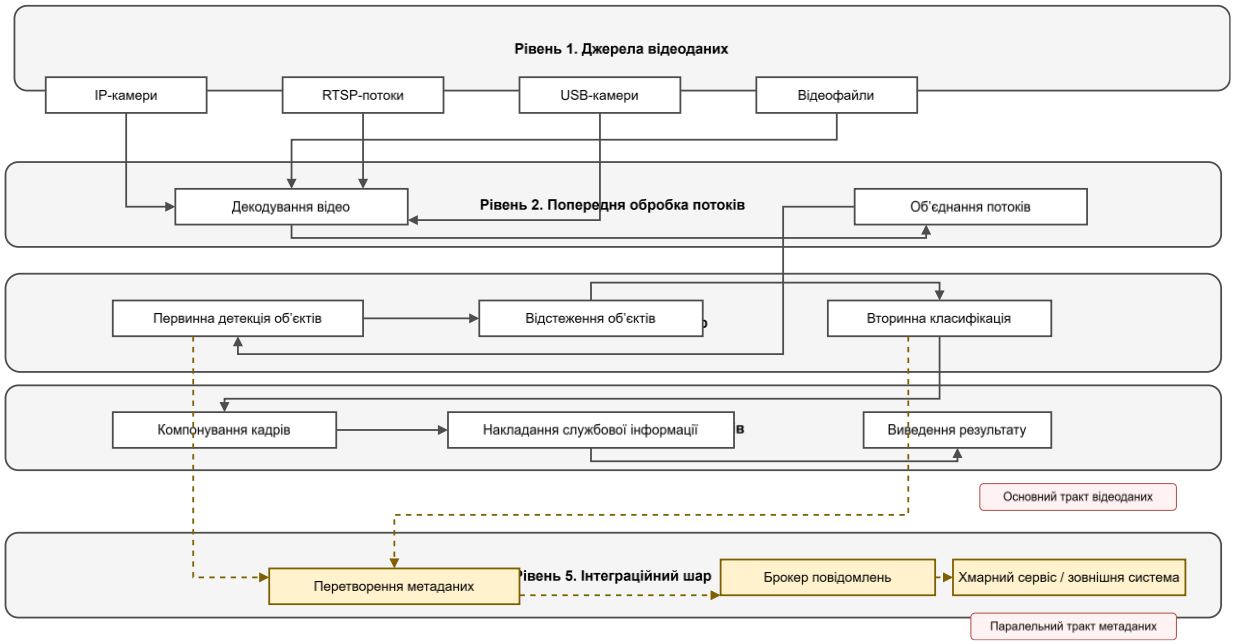



Рисунок 1.4 – Архітектура системи потокової відеоаналітики на базі NVIDIA DeepStream

Окрему групу становлять рішення, де головний акцент зроблено саме на прискоренні інференсу на платформах Nvidia Jetson. У дослідженні [9] продуктивності на Jetson AGX Xavier порівняно TensorFlow, TF-Lite, TF-TRT та чистий TensorRT для моделі YOLOv4. Отримані результати показують, що звичайний TensorFlow має помітно більшу затримку, тоді як TF-TRT і TensorRT забезпечують кращий баланс між точністю, латентністю та енергоспоживанням. Для TF-Lite зафіксовано найгіршу поведінку на Jetson, тому що він орієнтований на мобільні пристрої, а не на повноцінне використання GPU Jetson. Тобто для Nvidia Jetson як основні варіанти слід розглядати саме TF-TRT або TensorRT.

Для забезпечення переносимості моделей суттєве значення мають ONNX та суміжні формати. В роботах [10, 11] показано , що ONNX розв'язує проблему жорсткої прив'язки до початкового training-фреймворку та дає змогу будувати більш гнучкий ланцюг розгортання: навчання в одному середовищі, конвертація в ONNX, а потім запуск через відповідний inference engine. У цьому контексті ONNX

виступає не як готова система відеоаналітики, а як ключовий інфраструктурний компонент, який спрощує інтеграцію моделей у TensorRT, ONNX Runtime чи інші платформи. Його перевагами є сумісність, стандартизація та краща повторюваність розгортання; недоліком є те, що сам по собі він не вирішує задачі потокової обробки, стримінгу, трекінгу чи візуалізації, а лише полегшує шлях від навченої моделі до робочого inference-контуру. Типову схему перенесення та подальшого розгортання моделі з використанням формату ONNX наведено на рисунку 1.5.

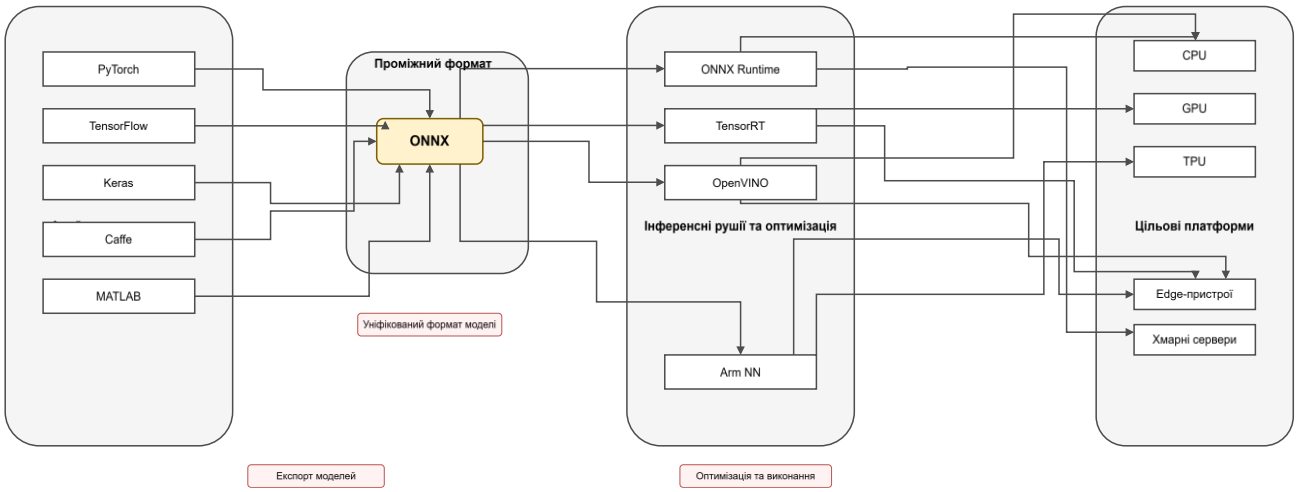



Рисунок 1.5 – Схема перенесення та розгортання моделі через ONNX

В роботі [10]. показано, що Jetson Nano є зручним для розробки, а TensorRT є критично важливим компонентом для оптимізації продуктивності. В іншій роботі [12] показано, що для застосунку з YOLOv4 на Jetson Xavier NX запуск через OpenCV давав лише близько 5 FPS, тоді як перехід до TensorRT дозволив досягти режиму реального часу. Це показує, що для edge-відеоаналітики основним моментом є вибір моделі, формату моделі та засобів апаратного прискорення.

Отже аналіз існуючих рішень показує, що для побудови високопродуктивного модуля комп’ютерного зору для потокового відео на Nvidia Jetson найбільш простим способом є поєднання підходів використання обробки на периферії мережі замість хмарної схеми. Порівняно з універсальними сервісними фреймворками, такими як EdgeEye, рішення на Jetson із DeepStream мають кращу прикладну придатність до для потокового передавання відео та інтеграції з edge-

обладнанням. Порівняно з мобільними платформами, Jetson дає кращу продуктивність і доступ до GPU-прискорення.

## 1.3 Постановка задачі

Сучасні системи потокової відеоаналітики мають працювати в умовах де є обмеження до яких входять великий обсяг відеоданих, вимоги до близького до реального часу реагування, обмежених ресурсів граничних пристроїв і необхідність збереження прийнятної точності виявлення об’єктів. В проаналізованих роботах показано, що тільки хмарний підхід не завжди забезпечує потрібний рівень швидкодії, тому що передавання відеопотоку до віддалених серверів збільшує затримку та навантаження на мережу. Водночас гранична обробка дає змогу наблизити виконання моделі до джерела даних, але вимагає ефективного використання обчислювальних ресурсів, пам’яті та засобів апаратного прискорення. Для платформ Nvidia Jetson це важливо, тому що практична придатність системи визначається точністю моделі, частотою обробки кадрів, затримкою та стабільністю роботи при тривалому навантаженні.

Крім того проблема ще полягає в тому, що для потокового відео необхідно побудувати такий програмний модуль, який поєднуватиме приймання відеопотоку, декодування, попередню обробку кадрів, виконання моделі виявлення об’єктів, формування результатів і візуалізацію або передавання цих результатів, зберігаючи придатність до роботи на edge-пристрої. Аналіз архітектурних і прикладних рішень показує, що система має будуватись на моделі детекції та правильно налаштованому конвеєрі обробки. Крім того мають враховуватись механізми сумісності форматів моделі та засоби оптимізації її виконання.

Отже загальна концепція розв’язання задачі полягає в створенні високопродуктивного модуля комп’ютерного зору для потокового відео на платформі Nvidia Jetson, в якому основна обробка виконується безпосередньо на граничному пристрої. Джерелом даних має бути відеопотік з камери або з іншого джерела, а внутрішня структура рішення повинна забезпечувати послідовне

проходження етапів декодування, підготовки кадру, запуску моделі виявлення об’єктів, отримання координат рамок і класів, відображення результатів та формування технічної статистики. Для підвищення швидкодії буде використано проміжний формат ONNX для перенесення моделі між середовищем навчання та середовищем розгортання, а також TensorRT як засіб оптимізації виконання моделі на GPU Nvidia.

До функціональних вимог розроблюваного модуля належать:

– підключення одного або кількох джерел відео;

– приймання та декодування відеопотоку;

– попередня обробка кадрів;

– запуск моделі виявлення об’єктів;

– формування координат рамок, класів і довірчих оцінок;

– відображення результатів у відеопотоці;

– формування статистики роботи системи;

– можливість використання оптимізованої моделі, придатної до виконання на Nvidia Jetson;

– передавання результатів до зовнішнього сервісу або інтерфейсу за потреби.

До нефункціональних вимог належать:

– робота в режимі, близькому до реального часу;

– обмежене використання CPU, GPU та оперативної пам’яті;

– стійкість до тривалої безперервної роботи;

– відтворюваність середовища розгортання;

– зручність налаштування та супроводу;

– можливість адаптації до різних джерел відео та конфігурацій моделі.

Ще для таких систем важливими є стабільність сервісу, можливість автоматичного відновлення та підтримка потокового передавання результатів з малою затримкою.

Також розроблюваний модуль має забезпечувати таку швидкість обробки, при якій відеоаналітика залишається придатною для практичного використання в

потоковому режимі. Це означає, що оцінювання має проводитися за сукупністю показників. До таких показників входять: а)завантаження CPU, б)завантаження GPU, в)точність, г)затримка та енергоспоживанням.

Тому, основними критеріями оцінювання розроблюваного модуля будуть:

– FPS — для оцінки швидкості обробки відеопотоку;

– затримка — для визначення часу, потрібного на обробку кадру або отримання результату;

– точність — для оцінки коректності виявлення об’єктів;

– використання GPU, CPU та RAM — для оцінки ефективності використання апаратних ресурсів платформи;

– стабільність роботи — для оцінки придатності модуля до тривалого використання.

Для досягнення поставленої мети необхідно виконати наступні завдання:

1. Дослідити існуючі підходи до побудови систем потокової відеоаналітики та проаналізувати їхні переваги та недоліки.

2. Розробити архітектуру високопродуктивного модуля комп’ютерного зору для обробки потокового відео на платформі Nvidia Jetson.

3. Обґрунтувати вибір програмних і апаратних засобів для реалізації модуля.

4. Реалізувати модуль приймання, попередньої обробки та аналізу відеопотоку з використанням моделі виявлення об’єктів.

5. Реалізувати оптимізацію виконання моделі за допомогою TensorRT для підвищення швидкодії системи.

6. Забезпечити відображення результатів аналізу відеопотоку та формування вихідних даних для подальшого використання.

7. Провести тестування розробленого модуля та оцінити його продуктивність.

# 2 АЛГОРИТМІЧНЕ ТА ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ МОДУЛЯ

## 2.1 Загальна структура проектованого модуля

Виходячи з результатів аналізу предметної області, постановки задачі та огляду існуючих рішень загальна структура проектованого модуля має будуватися як гранична система потокової відеоаналітики, в якій основна обробка виконується безпосередньо на платформі Nvidia Jetson.

Такий підхід відповідає останнім напрямкам в області edge-рішень, коли обчислення переносяться ближче до джерела даних, що дає змогу зменшити затримку, скоротити навантаження на мережу та зберегти працездатність системи навіть при втратах зв’язку. Для потокового відео основну роль відіграє не тільки модель детекції, а і правильно організований конвеєр обробки, в якому працюють модулі приймання потоку, перетворення кадрів, виконання моделі, постобробки та передавання результатів.

Така модульна побудова є найбільш прийнятною для розроблюваного рішення. На рисунку 2.1 наведено загальну структуру проектованого модуля потокової відеоаналітики на платформі Nvidia Jetson.

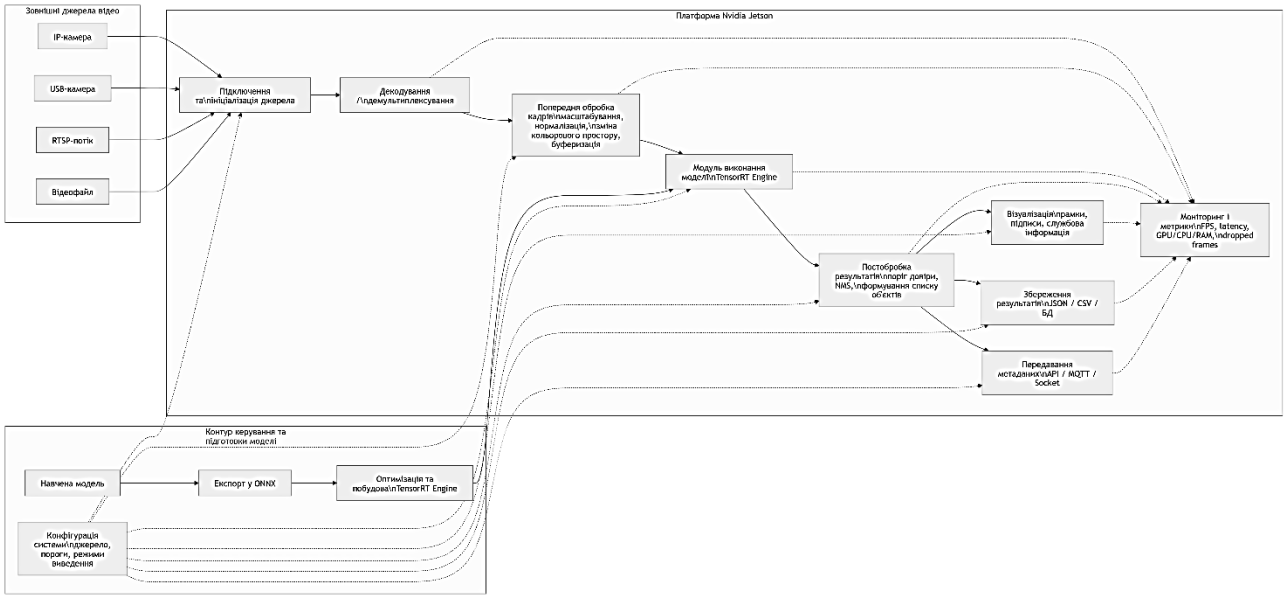



Рисунок 2.1 – Загальна структура проектованого модуля потокової відеоаналітики на платформі Nvidia Jetson

Першим компонентом архітектури є джерело відеопотоку. В проектованому модулі ним може бути IP-камера, звичайна USB-камера, локальний відеофайл або мережеве джерело потоку RTSP. Це дає можливість закласти в архітектуру універсальний вхідний інтерфейс, який не прив’язує систему до одного конкретного способу отримання відео. Тобто, на вході модуля має бути передбачено підсистему підключення та ініціалізації джерела, що забезпечує захоплення потоку, перевірку доступності джерела та подальшу передачу даних до наступного етапу обробки.

Другим компонентом є модуль декодування кадрів. Його призначення полягає у перетворенні вхідного відеопотоку в послідовність кадрів, придатних для подальшої аналітики. Цей етап реалізується через модулі декодування і злиття потоків, що готують відеодані до паралельної обробки. Для проектованого модуля це означає, що декодування має бути виокремлене в самостійний блок, який приймає вхідний потік, розпаковує його, за потреби змінює формат кадру і передає дані далі безпосередньо до підсистеми попередньої обробки. Такий поділ спрощує заміну джерела потоку і дає змогу незалежно оптимізувати роботу з відеовходом.

Після декодування кадри надходять у модуль попередньої обробки. Його функція полягає у приведенні кадрів до формату, потрібного для моделі: масштабуванні, нормалізації, зміні кольорового простору, узгодженні роздільної здатності та буферизації даних. Крім того час попередньої обробки є окремою складовою загального часу проходження кадру і повинен враховуватися в архітектурі окремо від часу виконання самої моделі. Тому, модуль попередньої обробки має працювати так, щоб мінімізувати зайві копіювання пам’яті, готувати дані в форматі, сумісному з GPU-обробкою, та забезпечувати рівномірне надходження кадрів до наступного етапу.

Центральним елементом системи є модуль виконання моделі. В ньому виконується обчислення результатів нейромережі для кожного кадру або пакету кадрів. Аналіз попередніх робіт показує, що краще відокремлювати середовище навчання моделі від середовища її виконання, тому ONNX використовується як проміжний формат сумісності, що дає змогу перенести модель з навчального

фреймворку в середовище розгортання, а ONNX Runtime і TensorRT виступають як спеціалізовані модулі виконання.

Для Nvidia Jetson найбільш підхлдящим є TensorRT, тому що він орієнтований на GPU Nvidia, підтримує різні типи оптимізацій, повторного використання пам’яті, змішаних режимів точності та квантування. Тому в архітектурі проєктованого модуля блок виконання моделі має приймати підготовлений кадр, завантажувати оптимізований файл моделі та повертати первинні результати детекції. Крім того, цей модуль виконує ініціалізацію середовища виконання, завантаження моделі, створення контексту виконання та керування GPU-ресурсами.

Після отримання сирих виходів моделі їх має обробляти модуль постобробки результатів. Отже, постобробка в проектованому модулі має включати відбір результатів за порогом довіри, придушення дубльованих рамок, формування остаточного списку виявлених об’єктів, та за потреби — передавання координат в модуль відстеження. В цьому блоці результати нейромережі перетворюються з внутрішнього представлення моделі на дані, придатні для відображення, збереження або подальшої інтеграції.

Останнім є модуль візуалізації, збереження або передавання результатів. Він робить  виокремлення та компонування кадрів, накладання службової інформації та виведення результату, а також окремий канал для перетворення метаданих і надсилання їх у зовнішні сервіси. Тобто цей блок повинен підтримувати як мінімум три режими, а саме накладання рамок і підписів на відео, збереження технічних результатів у структурованому вигляді та передавання даних до зовнішнього інтерфейсу або сервісу. Це забезпечить гнучкість використання модуля як в локальному, так і в мережевому сценарії.

В загальному взаємодія між компонентами модуля має працювати, як послідовний, але модульний конвеєр. Джерело відеопотоку передає дані в модуль декодування, після чого кадри проходять попередню обробку і надходять до модуля виконання моделі. Результати виконання передаються в постобробку, де формуються фінальні дані про виявлені об’єкти. Далі ці дані або візуалізуються у

відеопотоці, або зберігаються, або надсилаються до зовнішнього сервісу. Крім того працює канал керування моделлю та конфігурацією, де вибір моделі, формат її подання і спосіб оптимізації узгоджуються ще до запуску аналітичного конвеєра. Така структура відповідає практиці сучасних edge-систем: вона поєднує модульність EdgeEye [13], потоковий характер DeepStream [4], сумісність ONNX [3] і апаратно-орієнтовану оптимізацію TensorRT [14], що разом створює основу для розробки високопродуктивного модуля комп'ютерного зору для потокового відео на платформі Nvidia Jetson [15].

## 2.2 Алгоритмічне забезпечення модуля

Узагальнений алгоритм функціонування проектованого модуля (рисунок 2.2) має конвеєрний характер і охоплює весь шлях даних від джерела потоку до кінцевого представлення результатів. На першому етапі виконується ініціалізація системи: зчитуються параметри конфігурації, обирається джерело відео, перевіряється доступність камери або RTSP-потоку, ініціалізується середовище виконання на платформі Nvidia Jetson і завантажується підготовлений файл моделі.

Після цього запускається безперервний цикл приймання відеоданих. Кожен прийнятий фрагмент потоку декодується в кадри, проходить попередню обробку, передається в модуль інференсу, а потім в модуль постобробки.

На завершальному етапі результати або накладаються на зображення, або зберігаються у структурованому форматі, або передаються до зовнішнього сервісу. Така організація відповідає практиці сучасних систем потокової відеоаналітики, де обробка будується як безперервний конвеєр обробки кадрів з чітким розмежуванням ролей між блоками.

Узагальнений цикл системи можна описати наступним чином. Спочатку виконується підготовка моделі та параметрів виконання. Далі запускається конвеєр обробки, який у циклі читає нові дані з джерела, перетворює їх у придатне для моделі представлення, обчислює виходи нейромережі та переводить їх у прикладний формат.

Після цього результати маршрутизуються до вихідних каналів. Якщо є втрата потоку або помилки декодування виконується повторна спроба підключення або пропуск проблемного фрагмента.

Паралельно працює підсистема моніторингу, яка збирає часові показники та інформацію про завантаження ресурсів. Такий підхід дозволяє поєднати безперервність роботи, відмовостійкість і можливість подальшої оптимізації окремих етапів.

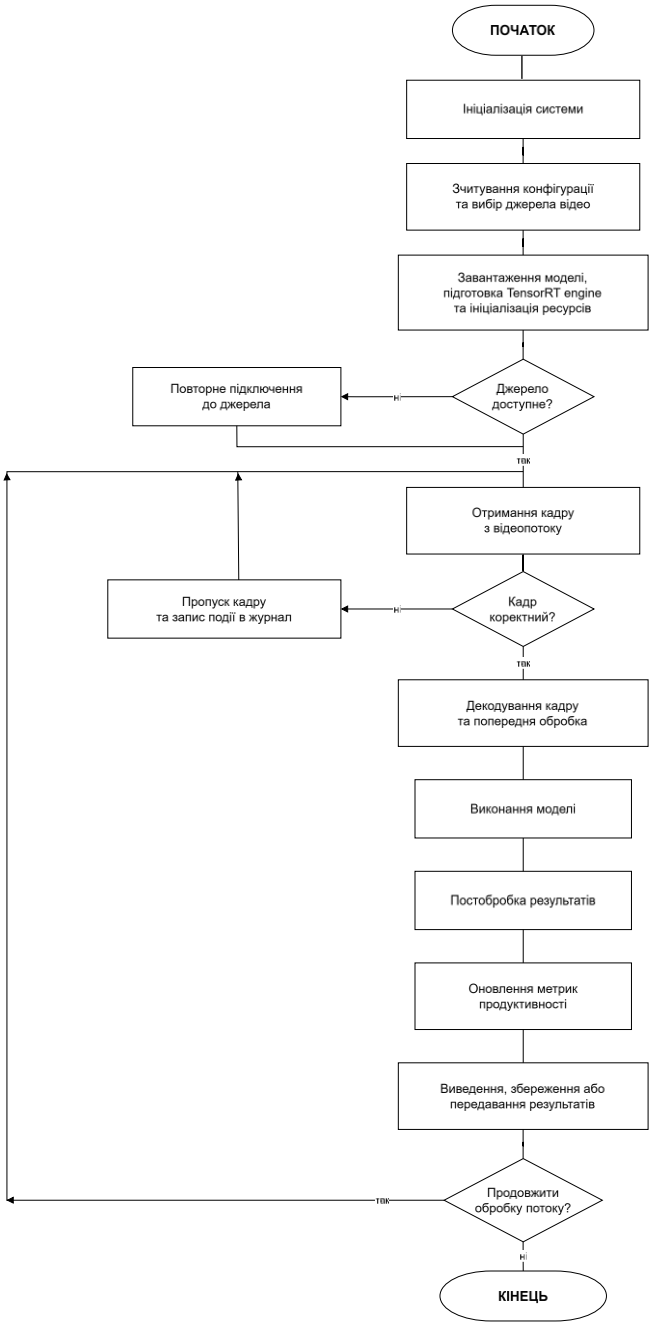



Рисунок 2.2 – Схема узагальненого алгоритму функціонування модуля потокової відеоаналітики

### 2.2.1 Алгоритм обробки одного кадру

Базовою одиницею обробки в проектованому модулі є окремий кадр. Саме тому алгоритм обробки одного кадру (рисунок 2.3) повинен бути прописаний чітко, оскільки на його основі далі реалізуються пакетний і асинхронний режими. Після отримання кадру з декодера виконується перевірка його коректності. Якщо кадр пошкоджений або порожній, він відкидається, а в журнал записується відповідна подія. Якщо кадр непошкоджений, він передається в блок попередньої обробки, де здійснюється масштабування до розміру вхідного тензора моделі, перетворення кольорового простору, нормалізація значень та вирівнювання пам’яті під GPU-обробку.

Наступним кроком є подання підготовленого тензора до TensorRT engine та отримання сирих результатів моделі. Після завершення інференсу результати передаються у постобробку. Тут відкидаються слабкі спрацювання за порогом довіри, здійснюється відбір найкращих рамок методом придушення немаксимумів для усунення дубльованих рамок, а потім формується остаточний список виявлених об’єктів з координатами, класами та рівнями впевненості моделі.

Якщо в системі активовано режим відстеження, результати детекції додатково передаються у модуль трекінгу, де кожному об’єкту призначається ідентифікатор, а його стан уточнюється за послідовністю кадрів. Такий підхід дозволяє ухвалювати рішення по кожному кадру окремо, без накопичення великого буфера попередніх кадрів. В результаті це зменшує затримку в формуванні результату.

Завершується алгоритм одного кадру формуванням вихідних артефактів. Якщо активний режим візуалізації, на кадр накладаються рамки, підписи класів, службова інформація та значення FPS. Якщо обрано режим накопичення статистики, результати записуються у таблицю або журнал. Якщо передбачено інтеграцію із зовнішнім сервісом, формується структуроване повідомлення з метаданими. Окремо фіксується час проходження кадру через усі етапи конвеєра.

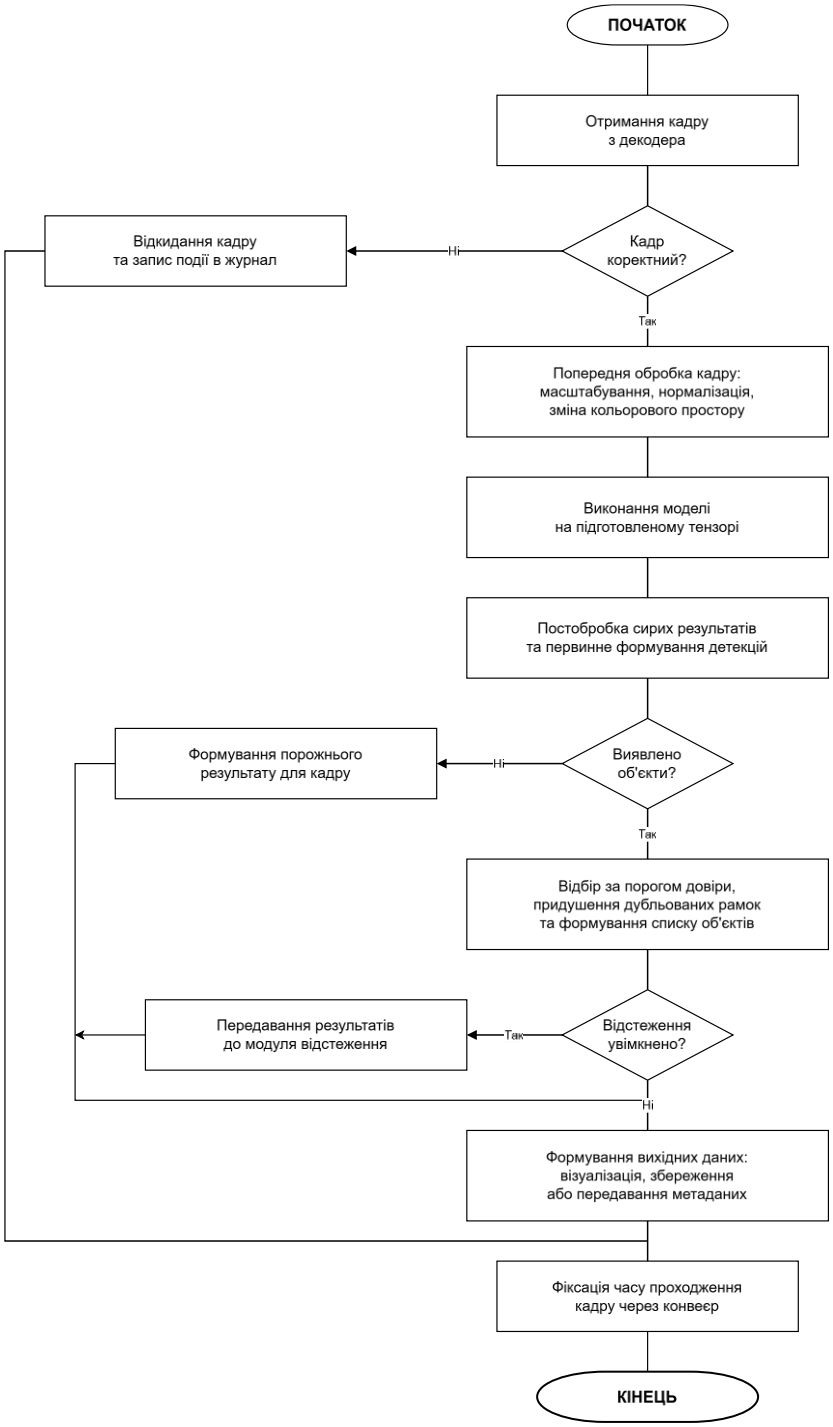



Рисунок 2.3 – Схема алгоритму обробки одного кадру в модулі потокової відеоаналітики

### 2.2.2 Алгоритм пакетної та асинхронної обробки

Для системи потокової відеоаналітики недостатньо лише покадрового алгоритму. В реальних умовах обробка має підтримувати пакетний або асинхронний режим, томущо різні етапи конвеєра мають різну тривалість і можуть виконуватися паралельно. В модулі найкращим варіантом буде використання

асинхронного конвеєра з буферами між блоками декодування, попередньої обробки, інференсу та постобробки. Тобто поки один кадр перебуває на інференсі, наступний може декодуватися і проходити підготовку. Така організація дозволяє підвищити загальну пропускну здатність системи без зміни логіки кожного окремого етапу.

При пакетній обробці вхідні кадри акумулюються в невеликі пакети фіксованого або адаптивного розміру. Після накопичення пакета створюється пакетний тензор, який подається до модуля виконання моделі. Перевага цього режиму полягає в кращому завантаженні GPU та зменшенні накладних витрат на запуск інференсу.

Асинхронна організація потрібна коли йде багатопотокове або багатокамерне захоплення і обробка відео. Це означає, що черги між етапами мають бути обмеженими, щоб не накопичувати надмірну затримку, а політика переповнення буфера повинна або відкидати найстаріші кадри, або пропускати частину кадрів у режимі деградації сервісу. Такий механізм дозволяє зберігати актуальність відеоаналітики навіть при тимчасовому перевантаженні системи.

### 2.2.3 Алгоритм оптимізації виконання моделі через TensorRT

Важливою частиною алгоритмічного забезпечення є алгоритм оптимізації моделі перед розгортанням (рисунок 2.4). Його призначення полягає у тому, щоб перетворити вихідну модель навчання в форму, придатну для високопродуктивного виконання на Nvidia Jetson.

На першому кроці підготовлена модель експортується в проміжний формат сумісності, після чого для неї виконується побудова TensorRT engine. На другому кроці задаються параметри оптимізації, такі як цільова точність обчислень, доступний обсяг робочої пам’яті та дозволені оптимізаційні перетворення. На третьому кроці виконується побудова engine з оптимізованим графом, де здійснюються злиття шарів, скорочення дубльованих операцій, повторне використання пам’яті та інші апаратно-орієнтовані перетворення. На четвертому

кроці сформований виконуваний модуль моделі зберігається у придатному для завантаження вигляді та використовується в робочому конвеєрі обробки.

TensorRT використовується для зменшення розміру моделі та ефективнішого використання апаратних ресурсів за рахунок об’єднання й усунення зайвих шарів, а також зменшення обсягу зайнятої пам’яті та повторного використання робочої пам’яті.

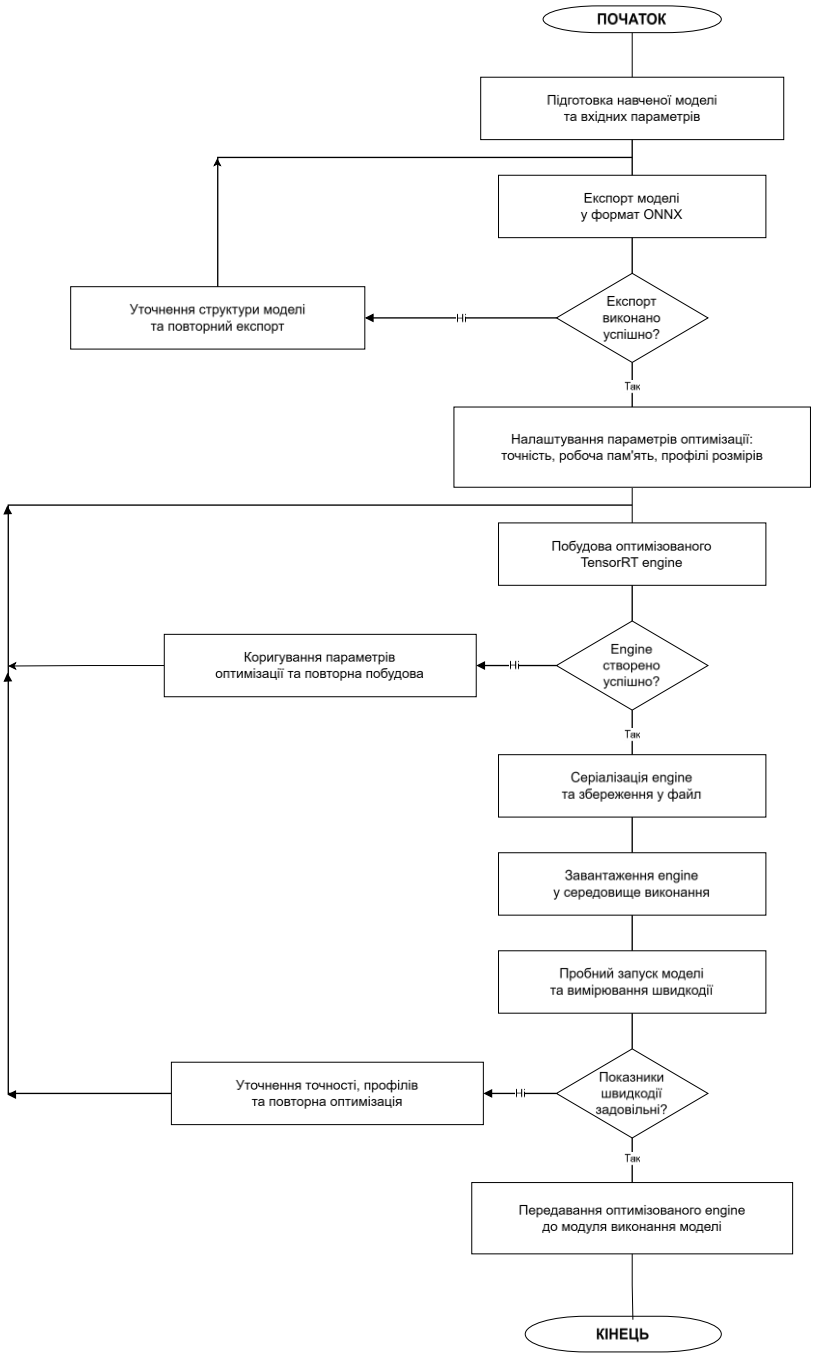



Рисунок 2.4 – Схема алгоритму оптимізації виконання моделі засобами TensorRT

### 2.2.4 Алгоритм вимірювання продуктивності

Завершальним елементом є алгоритм вимірювання продуктивності. Так як модуль має виконувати детекцію та забезпечувати заданий рівень швидкодії на edge-платформі. Тому в системі доцільно окремо вимірювати час декодування, час попередньої обробки, час виконання моделі, час постобробки, загальна наскрізна затримка, середній FPS, а також використання CPU, GPU та оперативної пам’яті.

Алгоритм вимірювання продуктивності працює наступним чином. Перед входом кадру в кожний ключовий блок фіксується часовий штамп. Після завершення етапу записується другий штамп, а різниця між ними додається у відповідний лічильник. Через певний часовий інтервал або після обробки визначеної кількості кадрів обчислюються середні, медіанні та пікові значення. Для FPS визначається кількість оброблених кадрів за секунду. Для повної затримки — інтервал між моментом отримання кадру від декодера і моментом формування фінального результату.

Тому алгоритм проектованого модуля повинно поєднувати безперервний конвеєр роботи системи, детальний покадровий алгоритм, асинхронну або пакетну організацію обробки, окремий етап TensorRT-оптимізації та підсистему вимірювання продуктивності.

## 2.3 Інформаційне забезпечення модуля

В цьому підрозділі подано, які саме дані надходять до системи, як вони перетворюються в процесі обробки та в якому вигляді формуються результати. Це є важливим для edge-орієнтованих рішень, тому що в них необхідно враховувати точність виявлення, обсяг передаваних даних, затримку обробки, обмеження пам’яті та енергоспоживання. Дослідження показують, що перенесення обробки ближче до джерела даних дає змогу зменшити накладні витрати на передавання та виконання обчислень.

Для проектованого модуля інформаційне забезпечення повинно охоплювати як відеодані, так і службову інформацію, що супроводжує обробку кадрів. Це пов’язано з тим, що сучасні системи відеоаналітики працюють не як одиничний алгоритм детекції, а як цілісний конвеєр, в якому кожний кадр проходить декодування, попередню обробку, виконання моделі, постобробку та формування вихідних результатів. Крім того, важливу роль відіграє ресурсне керування, де ключовими обмеженнями виступають точність, затримка та енергоспоживання. Тому в модулі мають бути передбачені окремі інформаційні структури для відеокадрів, результатів детекції, параметрів моделі, налаштувань потоку та статистики продуктивності.

Вхідна інформація проектованого модуля формується з кількох основних груп даних. Першою і базовою групою є відеодані, які надходять у вигляді потоку від IP-камери, RTSP-джерела, USB-камери або локального відеофайлу. В подібних edge-рішеннях джерело потоку часто конфігурується через RTSP-адресу або інші параметри підключення, що дає змогу відтворювати однаковий відеопотік для тестування та експлуатації. Це означає, що для вхідного потоку мають фіксуватися тип джерела, адреса або шлях до файлу, частота кадрів, роздільна здатність, формат кодування та часові мітки кадрів.

Другою групою вхідної інформації є параметри моделі. До них належать назва або тип моделі, шлях до файлу моделі, формат подання, розмір вхідного тензора, допустимі режими точності, поріг довіри, поріг придушення дубльованих рамок, а також параметри використання TensorRT. Так як в проектованому модулі модель проходить підготовку до виконання і далі працює у вигляді оптимізованого виконуваного модуля, інформаційне забезпечення повинно зберігати відомості про модель та параметри її розгортання.

Третьою групою є конфігурація потоку та режимів роботи системи. Сюди належать режим виведення результату, використання або невикористання модуля відстеження, максимальний розмір буфера кадрів, режим пакетної чи асинхронної обробки, а також інтервали оновлення статистики продуктивності. Ці параметри забезпечують керування загальною поведінкою модуля.

Вихідна інформація модуля формується як результат обробки кадрів. До неї належать координати виявлених об’єктів у вигляді обмежувальних рамок, ідентифікатори класів, текстові назви класів, оцінка достовірності виявлення, а також ідентифікатори відстежуваних об’єктів. Крім результатів виявлення, система повинна формувати службову статистику, до якої входить частота обробки кадрів, час декодування, час попередньої обробки, час виконання моделі, час постобробки, загальна наскрізна затримка, використання CPU, GPU та оперативної пам’яті. Ці показники використовуються для оцінювання ефективності. В таблиці 2.1 наведено склад вхідної та вихідної інформації проектованого модуля.

Таблиця 2.1 – Вхідна та вихідна інформація проектованого модуля

| Група | Найменування даних | Зміст | Форма подання |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| Вхідна | Джерело відеопотоку | Тип джерела: IP-камера, USB-камера, RTSP-потік або відеофайл | Текстовий параметр |
| Вхідна | Ідентифікатор джерела | RTSP-адреса, шлях до файлу або назва пристрою | Рядок |
| Вхідна | Параметри відеопотоку | Роздільна здатність, частота кадрів, кодек, формат потоку | Числові та текстові значення |
| Вхідна | Відеокадри | Послідовність кадрів, отриманих після декодування потоку | Масив піксельних даних |
| Вхідна | Часові мітки кадрів | Час отримання або номер кадру в потоці | Числове значення |
| Вхідна | Параметри моделі | Назва моделі, формат, шлях до файлу, розмір входу | Структурований набір параметрів |
| Вхідна | Параметри виконання моделі | Режим точності, поріг довіри, поріг придушення немаксимумів, пакетний режим | Числові та логічні значення |
| Вхідна | Конфігурація обробки | Режим візуалізації, збереження, передавання, використання відстеження | Логічні та текстові параметри |

Продовження Таблиця 2.1

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| Вхідна | Параметри буферизації | Розмір черги кадрів, режим асинхронної або пакетної обробки | Числові параметри |
| Вихідна | Координати об’єктів | Координати обмежувальних рамок: x_min, y_min, x_max, y_max | Числові значення |
| Вихідна | Клас об’єкта | Ідентифікатор і назва виявленого класу | Числове і текстове значення |
| Вихідна | Рівень достовірності | Оцінка впевненості моделі у виявленні об’єкта | Дійсне число |
| Вихідна | Ідентифікатор відстеження | Номер траєкторії об’єкта при увімкненому відстеженні | Ціле число |
| Вихідна | Анотований кадр | Кадр з рамками, підписами та службовою інформацією | Зображення або відеопотік |
| Вихідна | Метадані результатів | Структурований опис результатів детекції для зовнішніх сервісів | JSON / CSV / SQLite |
| Вихідна | Статистика продуктивності | FPS, час декодування, попередньої обробки, виконання моделі, постобробки, загальна затримка, CPU, GPU, RAM | Числові значення |

Після визначення складу вхідних і вихідних даних можна перейти до узагальненого опису інформаційних об’єктів, з якими працює система. На концептуальному рівні інформаційне забезпечення можна подати у вигляді сукупності взаємопов’язаних сутностей. Для проектованого модуля такими сутностями є VideoStream, Frame, Detection, TrackedObject, ModelConfig і PerformanceMetrics.

Сутність VideoStream описує джерело відеопотоку. Вона містить відомості про тип джерела, спосіб підключення, параметри відео та поточний стан потоку. Одному потоку відповідає множина кадрів, тому між сутностями VideoStream і Frame існує зв’язок «один до багатьох». Сутність Frame описує окремий кадр, що має ідентифікатор, часову мітку, розмір, формат і службові ознаки проходження через етапи конвеєра. Кадр є базовою одиницею обробки в модулі.

Сутність Detection відображає результат виявлення об’єкта в межах конкретного кадру. Вона містить координати рамки, клас об’єкта, назву класу,

оцінку достовірності та службові поля, пов’язані з постобробкою. Для одного кадру може бути отримано багато результатів виявлення, тому між Frame і Detection також існує зв’язок «один до багатьох». Якщо в системі ввімкнено відстеження, тоді окремі виявлення пов’язуються із сутністю TrackedObject, яка зберігає ідентифікатор траєкторії, часові межі існування об’єкта, його останню відому позицію та узагальнені характеристики руху.

Сутність ModelConfig описує параметри моделі та середовища її виконання. Вона містить назву моделі, формат зберігання, розмір входу, пороги детекції, режим точності та інші параметри, які впливають на якість і швидкодію. Сутність PerformanceMetrics описує статистику функціонування системи: FPS, часові витрати на етапах конвеєра, завантаження процесора, графічного прискорювача та пам’яті. На рисунку 2.5 наведено концептуальну інфологічну модель даних проектованого модуля.

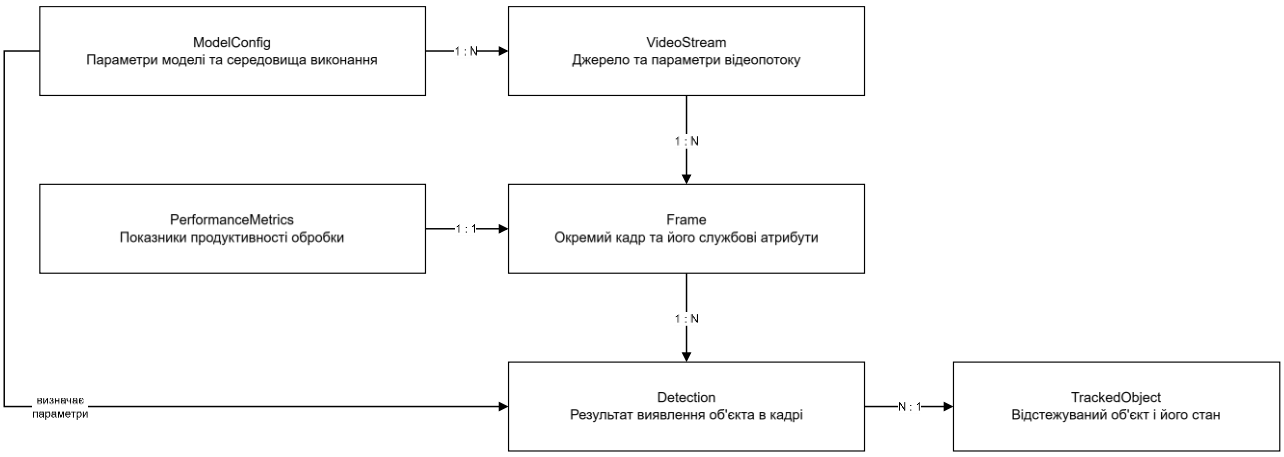



Рисунок 2.5 – Концептуальна інфологічна модель даних проектованого модуля

Подальша деталізація інформаційного забезпечення виконується на даталогічному рівні, де визначаються основні таблиці та їх атрибути. В проектованому модулі такими таблицями або класами є VideoStream, Frame, Detection, TrackedObject, ModelConfig і PerformanceMetrics.

Для VideoStream потрібно передбачити поля: stream_id, source_type, source_uri, codec, width, height, fps, status, created_at.

Для Frame — frame_id, stream_id, timestamp, frame_number, width, height, pixel_format, preprocess_status.

Для Detection — detection_id, frame_id, class_id, class_name, confidence, x_min, y_min, x_max, y_max, track_id.

Для TrackedObject — track_id, stream_id, first_seen, last_seen, last_x, last_y, class_id, state.

Для ModelConfig — config_id, model_name, model_format, engine_path, input_width, input_height, confidence_threshold, nms_threshold, precision_mode, batch_mode.

Для PerformanceMetrics — metric_id, frame_id, decode_time_ms, preprocess_time_ms, model_exec_time_ms, postprocess_time_ms, total_latency_ms, fps_value, cpu_usage, gpu_usage, memory_usage_mb.

Такий склад атрибутів відображає основну логіку та практику реальних edge-систем, де обробка кадру супроводжується веденням журналу подій, параметрів і метрик. На рисунку 2.6 наведено ER-діаграму даталогічної моделі даних проектованого модуля.

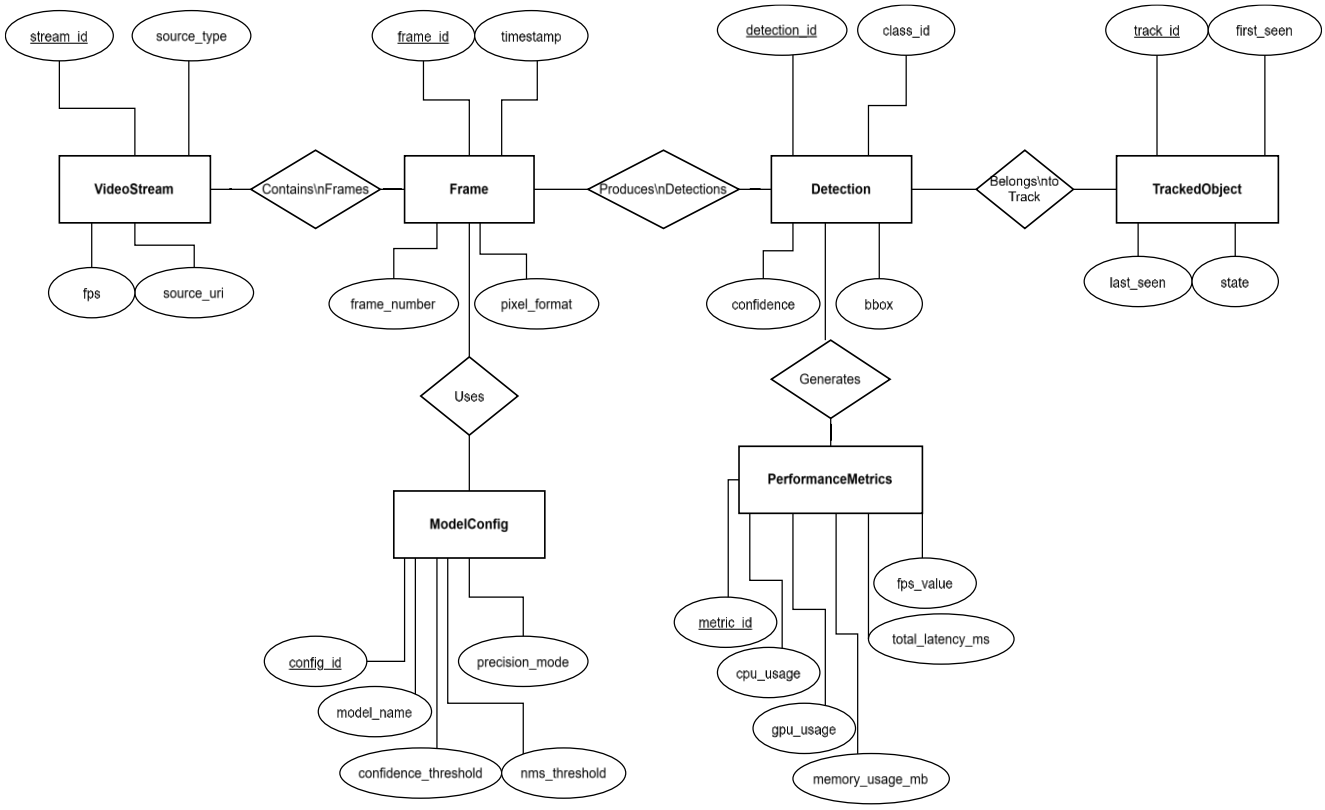



Рисунок 2.6 – ER-діаграма даталогічної моделі даних проектованого модуля

Завершальним рівнем подання є фізична модель даних, яка повинна бути орієнтована на практичне використання модуля на платформі Nvidia Jetson. Для передавання результатів у зовнішні сервіси найбільш простим і поширеним є формат JSON, тому що він дає змогу компактно описати кадр, список виявлень, параметри моделі та показники продуктивності у вигляді структурованого повідомлення.

Для експорту статистики та подальшого аналізу можна застосовувати CSV, тому що цей формат є простим у формуванні та придатним для подальшої обробки в електронних таблицях, скриптах або засобах візуалізації. Для локального збереження журналів, конфігурацій, виявлень і метрик можна використовувати SQLite, так як вона не потребує окремого серверного середовища, є легкою для edge-пристрою та підходить для автономного режиму роботи.

# 3 ПРОГРАМНО ТЕХНОЛОГІЧНЕ ЗАБЕЗПЕЧЕННЯ

## 3.1 Вибір програмних і апаратних засобів

Для реалізації модуля було обрано програмні й апаратні засоби, які дають змогу відтворити основну логіку потокової відеоаналітики та водночас забезпечують подальше перенесення рішення на цільову платформу Nvidia Jetson [16] (рисунок 3.1).

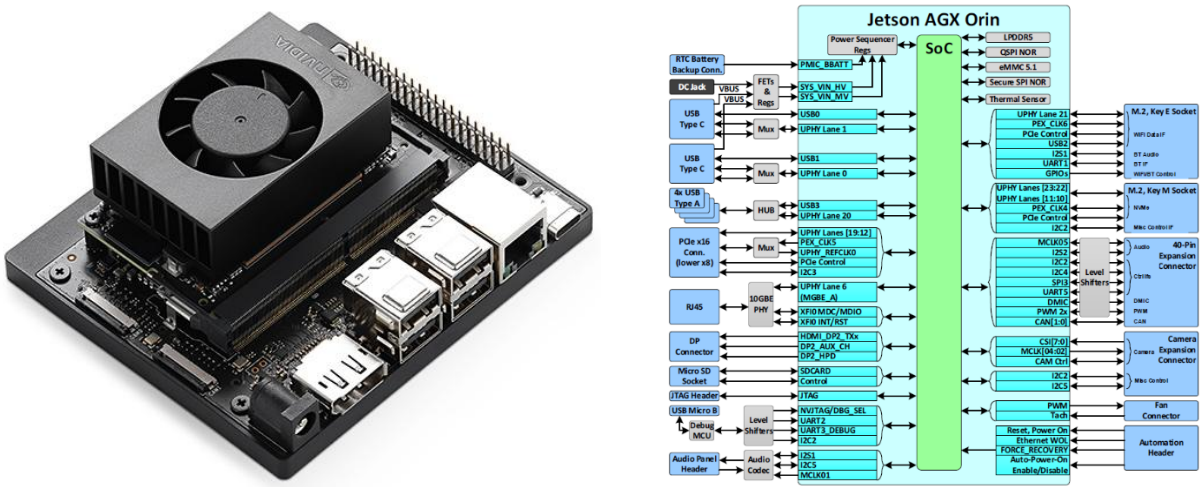



Рисунок 3.1 – Високопродуктивна платформа Nvidia Jetson [17]

Як операційну платформу реалізації було використано Linux-середовище [18]. Вибір такого варіанта аргументується тим, що Linux є типовим середовищем для розгортання edge-рішень і систем комп’ютерного зору. По-друге, більшість програмних бібліотек, потрібних для побудови потокової відеоаналітики, мають добру сумісність саме з Linux.

Основною мовою реалізації було обрано Python [19] тому що Python забезпечує достатню швидкість розробки, має прості засоби інтеграції між компонентами системи та широко використовується в сфері комп’ютерного зору та прикладних систем відеоаналітики. Крім того він дозволяє швидко реалізувати модульну структуру програми, забезпечити завантаження конфігурації, роботу з відеопотоком, збереження результатів та побудову web-інтерфейсу.

Для реалізації web-інтерфейсу було використано фреймворк Flask [20], так як він є нескладний, простим в налаштуванні та достатністю функціональних можливостей для побудови інтерфейсу керування модулем. За допомогою Flask реалізовано приймання HTTP-запитів, завантаження відеофайлів, запуск обробки та формування сторінки з відображенням конфігурації, параметрів відео й попереднього результату аналізу.

Для роботи з відеоданими було використано бібліотеку OpenCV [21] у варіанті opencv-python-headless, тому що графічне середовище не буде використовуватись, а сам модуль орієнтований на серверний або edge-режим виконання. OpenCV забезпечує відкриття відеофайлів, зчитування кадрів, визначення параметрів потоку, побудову анотованих кадрів і запис результатів у вихідний відеофайл. Використання headless-варіанта бібліотеки дозволило уникнути графічних залежностей і зробити реалізацію більш наближеною до типових умов розгортання у безмоніторному середовищі.

В таблиці 3.1 наведено вибрані програмні й апаратні засоби реалізації модуля та їх призначення.

Таблиця 3.1 – Вибрані програмні й апаратні засоби реалізації модуля та їх призначення

| Засіб | Тип засобу | Призначення в модулі |
|---|---|---|
| 1 | 2 | 3 |
| Nvidia Jetson | апаратний засіб | Цільова edge-платформа для подальшого розгортання модуля потокової відеоаналітики |
| Linux | операційне середовище | Базове середовище виконання програмного модуля, придатне для серверного та edge-режиму роботи |
| Python | мова програмування | Реалізація логіки модуля, інтеграція компонентів системи та організація обробки відеоданих |

Продовження Таблиця 3.1

| 1 | 2 | 3 |
|---|---|---|
| Flask | web-фреймворк | Побудова web-інтерфейсу користувача, обробка HTTP-запитів, завантаження відео та запуск обробки |
| OpenCV | бібліотека комп'ютерного зору | Зчитування відеофайлів, отримання кадрів, визначення параметрів потоку, побудова анотованих кадрів і запис вихідного відео |
| NumPy | бібліотека числових обчислень | Опрацювання числових даних та підтримка операцій, пов'язаних із представленням і передаванням кадрів |
| PyYAML | бібліотека роботи з конфігурацією | Зчитування параметрів модуля з конфігураційного файла та відокремлення налаштувань від програмного коду |
| mock_backend.py | програмний модуль | Виконання моделі виявлення об'єктів у Linux- середовищі |
| tensorrt_backend.py | програмний модуль | Підготовлений цільовий backend з використанням TensorRT |
| MP4 | формат вихідних даних | Збереження анотованого відео з результатами обробки |
| JSON | формат вихідних даних | Збереження структурованого опису виявлень для подальшого аналізу та інтеграції |
| CSV | формат вихідних даних | Збереження метрик продуктивності для тестування та оцінювання ефективності роботи модуля |

Для роботи з числовими масивами використано бібліотеку NumPy, яка є стандартним засобом організації обчислень на Python-системах комп'ютерного зору. Для збереження конфігураційного файла обрано формат YAML і бібліотеку PyYAML. Це рішення дозволяє відокремити службові параметри модуля від програмного коду, задавати тип джерела відео, шляхи до вхідних і вихідних файлів, розміри входу моделі, порогові значення та параметри обробки в зовнішньому конфігураційному файлі. Це підвищує гнучкість реалізації і полегшує адаптацію системи до нових умов експлуатації.

Так як цільовою платформою роботи є Nvidia Jetson, під час проектування програмної структури було передбачено модульний backend виконання моделі. У

поточній реалізації використано backend mock_backend.py, який забезпечує повернення структурованого результату виявлення об’єктів і дозволяє відпрацювати всі інші компоненти системи.

Для збереження результатів роботи модуля було використано кілька форматів. Анотоване відео зберігається у форматі MP4, що дозволяє переглядати результати обробки у звичному вигляді. Структурований опис виявлень зберігається у JSON, що є зручним для інтеграції з іншими програмними компонентами та подальшого аналізу. Метрики продуктивності зберігаються у CSV, що полегшує їх подальше використання в електронних таблицях і під час підготовки матеріалів до оцінювання результатів.

Обрані програмні й апаратні засоби забезпечили можливість реалізувати функціонально повний стенд високопродуктивного модуля комп’ютерного зору для потокового відео.

## 3.2 Реалізація ключових модулів

Реалізація модуля виконувалася у вигляді стенда потокової відеоаналітики, який за своєю структурою наближений до цільового варіанта розгортання на платформі Nvidia Jetson. Основною метою цього етапу була побудова цілісної модульної системи, в якій кожен компонент відповідає певному етапу конвеєра обробки відеопотоку. В результаті було реалізовано ядро системи, що включає web-інтерфейс, модуль завантаження відео, модуль читання потоку, модуль виконання моделі, засоби візуалізації результатів, а також модулі збереження вихідних даних і метрик.

Загальна структура програмної реалізації організована у вигляді окремих Python-модулів, кожен із яких виконує чітко визначену функцію. Такий підхід відповідає раніше розробленій архітектурі і дозволяє ізолювати окремі задачі одна від одної. Центральним вхідним файлом програмного рішення є main.py, в якому описано маршрути web-застосунку, обробку подій завантаження відео, запуск попереднього аналізу та виконання обробки короткого відеофрагмента. Цей файл

виконує координацію роботи всіх інших модулів і забезпечує взаємодію між інтерфейсом користувача та внутрішньою логікою системи.

На рисунку 3.2 наведено фрагмент програмної реалізації модуля main.py, який забезпечує завантаження відеофайлу та запуск попереднього аналізу.

```python
@app.route("/upload", methods=["POST"])
def upload():
    file = request.files.get("video_file")
    if not file or file.filename == "":
        return redirect(url_for("index", message="Файл не вибрано."))

    save_path = UPLOADS_DIR / "input.mp4"
    file.save(save_path)
    try:
        info = get_video_info(str(save_path))
        preview = run_preview_inference(
            str(save_path),
            config["model"]["classes"]
        )
        message = (
            f"Відеофайл успішно завантажено. "
            f"Розмір кадру: {info['width']}×{info['height']}, "
            f"FPS: {info['fps']}, "
            f"кадрів: {info['frame_count']}, "
            f"попередньо виявлено об'єктів: {preview['detection_count']}."
        )
    except (VideoSourceError, PipelineError) as e:
        message = f"Файл завантажено, але під час аналізу виникла помилка: {e}"
    return redirect(url_for("index", message=message))
```

Рисунок 3.2 – Лістинг, фрагмент main.py

На рисунку 3.3 наведено структуру програмної реалізації модуля потокової відеоаналітики та взаємозв’язки між його основними компонентами.

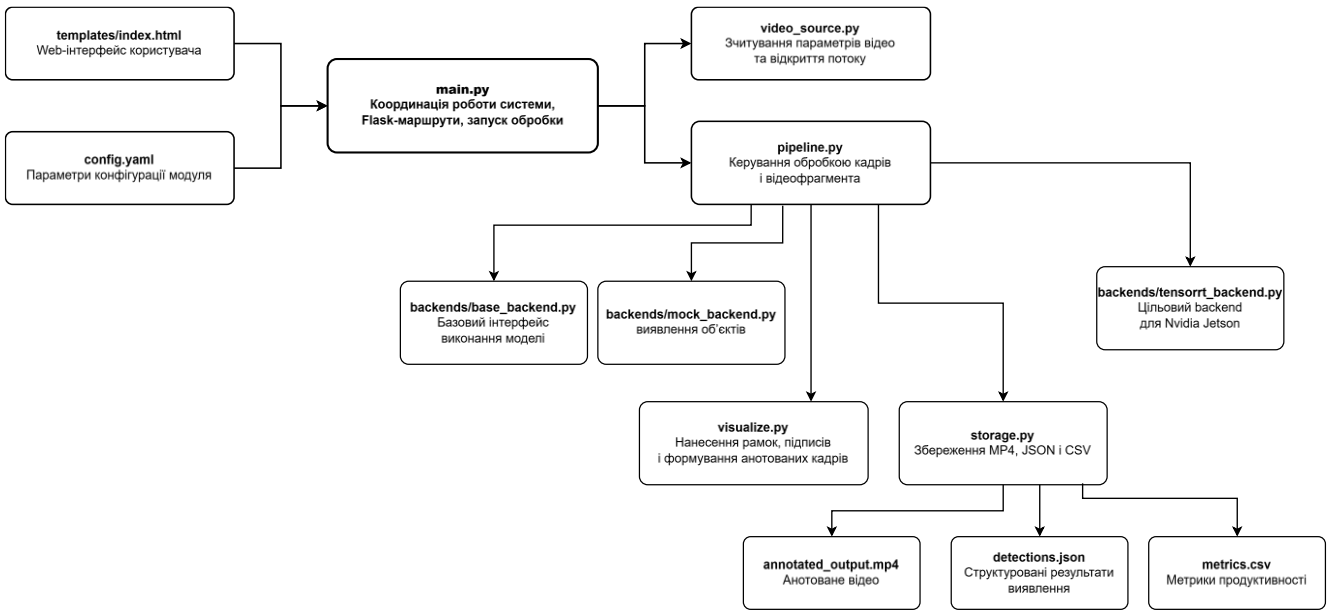



Рисунок 3.3 – Структура програмної реалізації модуля

Одним з базових елементів реалізації є модуль роботи з відеоджерелом video_source.py. В ньому реалізовано перевірку існування відеофайлу, відкриття потоку через OpenCV і визначення основних характеристик відео, зокрема розміру кадру, частоти кадрів і загальної кількості кадрів. Цей модуль використовується як під час завантаження відеофайлу, так і під час відображення службової інформації в інтерфейсі користувача.

Для відтворення логіки виконання моделі було реалізовано окремий backend-модуль. Його базовий інтерфейс описано у файлі base_backend.py, де визначено загальні методи завантаження моделі та обробки кадру. На поточному етапі фактичне виконання реалізовано у файлі mock_backend.py. Кожне виявлення містить ідентифікатор класу, назву класу, рівень достовірності та координати обмежувальної рамки. Окремо у структурі проекту передбачено файл tensorrt_backend.py, який закладає можливість подальшої інтеграції TensorRT як цільового середовища виконання моделі.

Ключова логіка обробки відео зосереджена у модулі pipeline.py. В ньому реалізовано два основні режими роботи. Перший режим — це попередній аналіз першого кадру, який використовується для швидкої перевірки працездатності системи. Після завантаження відео модуль зчитує перший кадр, передає його до backend-модуля, отримує список виявлень і формує службову інформацію для відображення у web-інтерфейсі. Другий режим — це повна обробка короткого відеофрагмента. В цьому випадку модуль послідовно зчитує кадри відео, виконує для кожного з них виявлення об’єктів, наносить анотації на кадр і записує результат у вихідний відеофайл. Додатково в процесі роботи накопичуються зведені дані про кількість оброблених кадрів, загальну кількість виявлень, час виконання та ефективну частоту кадрів.

На рисунку 3.4 наведено фрагмент програмної реалізації модуля pipeline.py, який виконує послідовну обробку кадрів відеопотоку.

```
while processed_frames < max_frames:
    ok, frame = cap.read()
    if not ok or frame is None:
        break
    detections = backend.infer(frame)
    annotated_frame = draw_detections(frame, detections)
    writer.write(annotated_frame)
    frames_data.append(
        {
            "frame_index": processed_frames,
            "detection_count": len(detections),
            "detections": detections,
        }
    )
    processed_frames += 1
    total_detections += len(detections)
```

Рисунок 3.4 – Лістинг, фрагмент pipeline.py

Візуальне оформлення результатів виконується у модулі visualize.py. В ньому реалізовано нанесення прямокутних рамок, підписів класів і значень достовірності на зображення. Окремо передбачено збереження анотованого першого кадру у вигляді графічного файла, який надалі відображається у web-інтерфейсі.

На рисунку 3.5 наведено фрагмент програмної реалізації модуля visualize.py, який забезпечує нанесення рамок і підписів на кадр.

```
def draw_detections(frame, detections):
    output = frame.copy()
    for det in detections:
        x1, y1, x2, y2 = det["bbox"]
        class_name = det["class_name"]
        confidence = det["confidence"]
        cv2.rectangle(output, (x1, y1), (x2, y2), (0, 255, 0), 2)
        label = f"{class_name} {confidence:.2f}"
        cv2.putText(
            output,
            label,
            (x1, max(y1 - 10, 20)),
            cv2.FONT_HERSHEY_SIMPLEX,
            0.6,
            (0, 255, 0),
            2,
            cv2.LINE_AA
        )
    return output
```

Рисунок 3.5 – Лістинг, фрагмент visualize.py

Цей компонент надає користувачу результат роботи модуля, оскільки дозволяє побачити, як система інтерпретує вхідне відео.

Для формування вихідних файлів реалізовано модуль storage.py. Його функції охоплюють створення записувача відео, збереження структурованих результатів у JSON та збереження зведених метрик у CSV. Використання кількох форматів дає змогу задовольнити різні сценарії подальшого використання результатів. Відеофайл потрібний для візуальної перевірки роботи системи, JSON — для інтеграції з іншими програмними компонентами, а CSV — для аналізу продуктивності та підготовки експериментальних результатів.

Інтерфейс користувача реалізовано у вигляді web-сторінки на Flask із HTML-шаблоном index.html. На сторінці відображаються поточна конфігурація системи, форма завантаження відеофайлу, параметри завантаженого потоку, результати попереднього аналізу першого кадру та повідомлення про завершення повної обробки відео. Після запуску обробки користувач отримує зведену інформацію про кількість оброблених кадрів, загальну кількість виявлень, час роботи модуля та ефективну частоту кадрів.

Окремого значення в реалізації набуває конфігураційний файл config.yaml. В ньому зосереджено основні параметри програми: тип backend-модуля, шлях до вхідного відео, шляхи до вихідних файлів, розмір входу моделі, порогові значення та обмеження на кількість кадрів. Завдяки цьому відокремлюються параметри налаштування від програмного коду, що спрощує подальше масштабування рішення та його адаптацію до інших умов використання.

Реалізовані програмні модулі утворюють працездатний прототип системи потокової відеоаналітики, який відтворює основні функціональні етапи цільового рішення приймання відео, обробку кадрів, отримання структурованих результатів, візуалізацію виявлень і збереження вихідних даних.

## 3.3 Інтерфейс взаємодії з користувачем та оцінювання продуктивності

Для забезпечення зручної взаємодії з модулем було реалізовано web-інтерфейс користувача. Такий підхід пояснюється тим, що програмна реалізація

виконувалася у Linux середовищі без графічної оболонки, тому використання браузера як клієнтського середовища є найбільш доцільним рішенням.

Інтерфейс побудовано у вигляді web-застосунку, який працює через Flask. Його основне призначення полягає в тому, щоб надати користувачеві прості засоби для завантаження відеофайлу, перегляду службової інформації про потік, запуску обробки та ознайомлення з попередніми результатами аналізу. Завдяки цьому навіть без прямого доступу до внутрішніх програмних модулів користувач може виконати повний цикл взаємодії із системою через браузер.

В верхній частині сторінки розміщено заголовок модуля та короткий опис призначення системи. Основний зміст сторінки організовано у вигляді двох інформаційних блоків. Лівий блок відповідає за керування обробкою відео, а правий — за відображення поточної конфігурації системи. Такий поділ є зручним, оскільки відокремлює операції користувача від службових характеристик програмного середовища. Початковий вигляд розробленого web-інтерфейсу із блоками керування та відображення конфігурації наведено на рисунку 3.6.

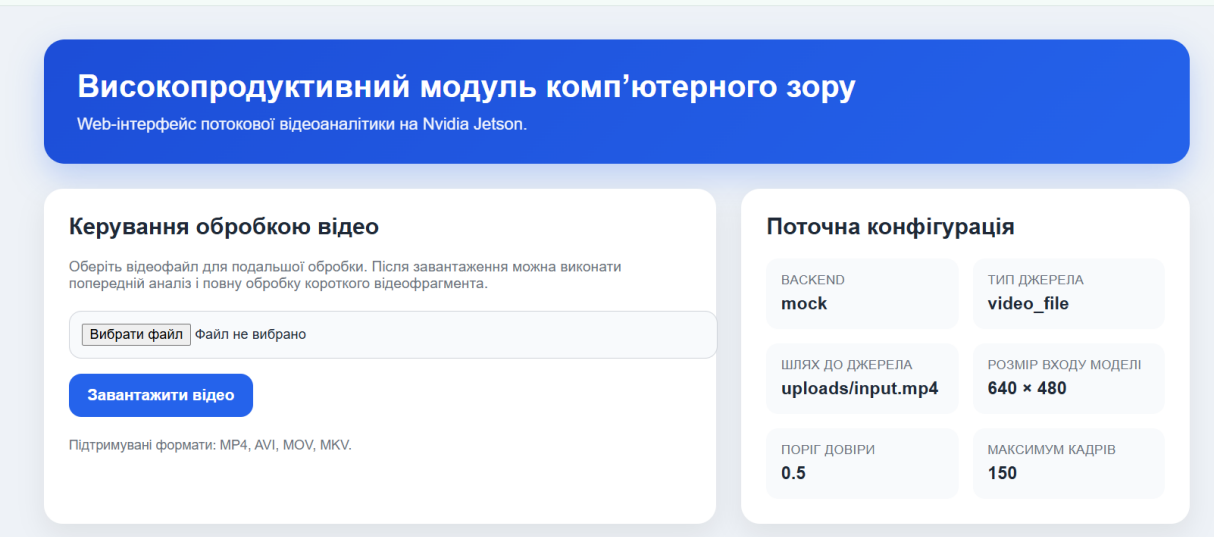



Рисунок 3.6 – Початковий вигляд web-інтерфейсу модуля потокової відеоаналітики

В блоці керування реалізовано форму завантаження відеофайлу. Користувач може вибрати файл одного з підтримуваних форматів і передати його до системи. Після цього модуль зберігає відео у внутрішньому каталозі, відкриває його

засобами OpenCV і визначає основні параметри потоку. Якщо завантаження виконано успішно, інтерфейс відображає повідомлення про результат операції, а також службову інформацію про відео, зокрема шлях до файла, розмір кадру, частоту кадрів і загальну кількість кадрів. Такий підхід дозволяє ще до запуску повної обробки переконатися, що система коректно прийняла і розпізнала вхідні дані. Вигляд інтерфейсу після успішного завантаження відеофайлу та отримання його основних характеристик показано на рисунку 3.7.

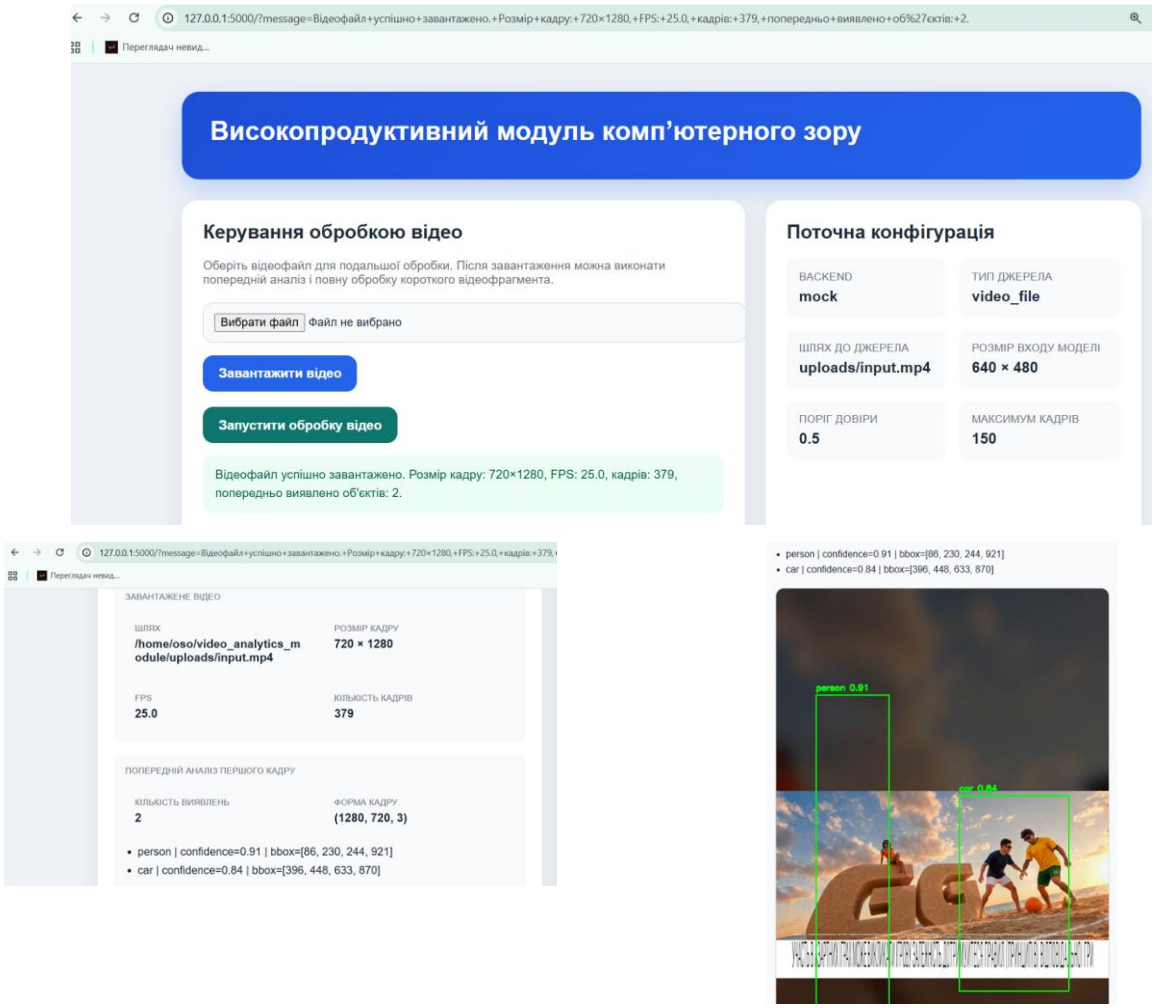



Рисунок 3.7 – Web-інтерфейс після завантаження відеофайлу

Після завантаження відео інтерфейс відкриває доступ до наступної дії — запуску обробки. Для цього реалізовано окрему кнопку запуску, натискання якої ініціює обробку короткого відеофрагмента відповідно до параметрів, заданих у конфігураційному файлі. За підсумками виконання користувач одержує зведене

текстове повідомлення, у якому відображаються кількість оброблених кадрів, загальна кількість виявлень, час виконання та ефективна частота кадрів.

Окремим елементом сторінки є блок попереднього аналізу першого кадру. Його призначення полягає в тому, щоб ще до запуску повної обробки показати користувачеві, як саме система інтерпретує вхідне відео. У цьому блоці виводяться кількість виявлень, форма кадру та перелік знайдених об’єктів із зазначенням назви класу, рівня достовірності й координат рамки. Крім текстового списку, на сторінці відображається збережене зображення першого кадру з нанесеними прямокутними рамками та підписами. Результат роботи інтерфейсу після запуску обробки відео та виконання попереднього аналізу наведено на рисунку 3.8.

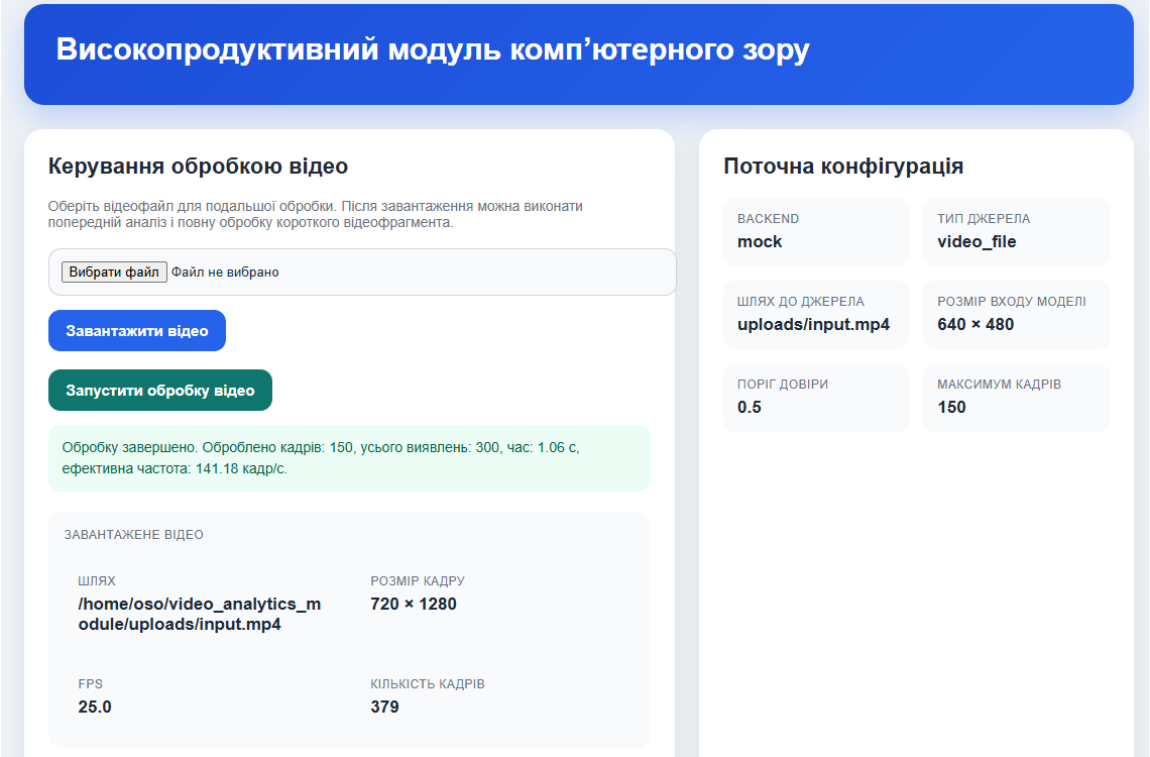



Рисунок 3.8 – Web-інтерфейс після запуску обробки відео

В правій частині сторінки розташовано блок поточної конфігурації. У ньому відображаються ключові параметри системи, зокрема тип backend-модуля, тип джерела, шлях до відеоджерела, розмір входу моделі, поріг довіри та максимальна кількість кадрів для обробки. Наявність такого блоку є важливою з практичної точки зору, оскільки дозволяє контролювати активні налаштування без відкриття

конфігураційного файла. Це особливо корисно під час демонстрації роботи модуля та його тестування.

Окремо перевірялося, що в каталозі результатів створюються всі потрібні вихідні файли: анотоване відео у форматі MP4, структурований файл detections.json і файл metrics.csv із метриками продуктивності. Усі ці компоненти були успішно сформовані, що свідчить про коректну роботу основного конвеєра системи.

Оцінювання продуктивності виконувалося за базовими показниками, які безпосередньо пов’язані з логікою проектованого модуля. До них належали кількість оброблених кадрів, загальна кількість виявлень, повний час проходження процедури обробки і ефективна частота кадрів. Додатково до CSV-файлу зберігалися параметри вхідного відео, зокрема розмір кадру та частота вхідного потоку. Це дозволило відокремити характеристики самого вхідного відео від результатів, які були отримані в процесі виконання модуля.

В таблиці 3.2 наведено результати оцінювання продуктивності реалізованого модуля за даними, сформованими у файлі metrics.csv

Таблиця 3.2 – Результати оцінювання продуктивності реалізованого модуля

| Показник | Позначення у файлі | Значення | Одиниця вимірювання | Характеристика |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 |
| Кількість оброблених кадрів | processed_frames | 150 | кадрів | Загальна кількість кадрів, оброблених у межах одного запуску |
| Загальна кількість виявлень | total_detections | 300 | об’єктів | Сумарна кількість виявлених об’єктів у всіх оброблених кадрах |
| Повний час обробки | elapsed_time_sec | 1,06 | с | Час виконання процедури обробки відеофрагмента |
| Ефективна частота кадрів | effective_fps | 141,18 | кадр/с | Фактична швидкість обробки кадрів модулем |

Продовження Таблиця 3.2

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| Ширина вхідного кадру | input_width | 720 | пікселів | Горизонтальний розмір кадру вхідного відео |
| Висота вхідного кадру | input_height | 1280 | пікселів | Вертикальний розмір кадру вхідного відео |
| Частота вхідного потоку | input_fps | 25,0 | кадр/с | Номінальна частота кадрів вхідного відео |

Реалізований web-інтерфейс забезпечує зручний спосіб взаємодії з модулем потокової відеоаналітики в середовищі без локальної графічної оболонки.

Наведені результати підтверджують, що модуль забезпечує коректну обробку відеопотоку, формування структурованих результатів і збереження базових метрик продуктивності.

# ВИСНОВКИ

Під час підготовки кваліфікаційної роботи було розроблено високопродуктивний модуль комп’ютерного зору для потокового відео, орієнтований на використання в edge-середовищі та подальше розгортання на платформі Nvidia Jetson.

Було отримано такі основні результати:

1. Проведено аналіз предметної області потокової відеоаналітики.
2. Визначено, що сучасні системи цього класу повинні поєднувати задачі виявлення, класифікації, відстеження об’єктів та аналізу подій у безперервному відеопотоці.
3. Встановлено, що перспективним напрямом є виконання основної обробки безпосередньо на граничному пристрої.
4. Виконано огляд і аналіз існуючих рішень у сфері edge-відеоаналітики.
5. Розглянуто сервісні фреймворки, програмно-апаратні рішення на базі Nvidia Jetson.
6. За результатами аналізу встановлено, що для побудови високопродуктивного модуля доцільно використовувати платформу Nvidia Jetson, проміжний формат ONNX і TensorRT як засіб апаратно-орієнтованої оптимізації.
7. Сформульовано постановку задачі та визначено основні функціональні й нефункціональні вимоги до розроблюваного модуля.
8. Визначено, що система повинна забезпечувати приймання та декодування відеопотоку, попередню обробку кадрів, виконання моделі виявлення об’єктів, формування вихідних результатів, їх візуалізацію, збереження або передавання до зовнішніх сервісів.
9. Розроблено загальну структуру проектованого модуля як граничної системи потокової відеоаналітики.
10. Сформовано алгоритмічне забезпечення модуля. Описано узагальнений алгоритм функціонування системи, алгоритм обробки одного кадру, особливості

пакетної та асинхронної обробки, алгоритм оптимізації виконання моделі через TensorRT і алгоритм вимірювання продуктивності.

11. Розроблено інформаційне забезпечення проектованого модуля. Визначено склад вхідної та вихідної інформації.

12. Обґрунтовано вибір програмних і апаратних засобів реалізації модуля. Для практичної реалізації використано Linux-середовище, мову Python, фреймворк Flask, бібліотеки OpenCV, NumPy і PyYAML, а також модульний підхід до організації backend-компонентів.

13. Реалізовано програмно-технологічний стенд потокової відеоаналітики, який включає web-інтерфейс, модуль завантаження відео, модуль зчитування параметрів потоку, модуль обробки кадрів, засоби візуалізації результатів і модулі збереження вихідних даних..

14. Реалізовано web-інтерфейс взаємодії з користувачем, який забезпечує завантаження відеофайлу, перегляд службових характеристик потоку, запуск обробки, відображення попереднього аналізу першого кадру та одержання підсумкової інформації про результати роботи модуля.

15. Проведено тестування працездатності розробленого рішення.

16. Виконано оцінювання продуктивності реалізованого стенда за базовими показниками.

17. Встановлено, що розроблений модуль забезпечує коректну обробку відеопотоку, формування структурованих результатів, візуалізацію виявлень і збереження базових метрик продуктивності, а отже може розглядатися як працездатний прототип системи потокової відеоаналітики.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Головін О. М. Алгоритмічно-програмні аспекти граничної відеоаналітики: від теорії до практичного впровадження. Проблеми керування та інформатики. 2025. № 4. С. 50–84.

2. Hu M., Luo Z., Pasdar A., Lee Y. C., Zhou Y., Wu D. Edge-Based Video Analytics: A Survey. arXiv preprint. 2023. arXiv:2303.14329.

3. Маленчик Т. В., Мирончук О. Ю., Неуймін О. С. Аналіз алгоритмів виявлення та супроводження точкових об’єктів у відеопотоці. Вісник Вінницького політехнічного інституту. 2022. № 6. С. 48–56. DOI: 10.31649/1997-9266-2022-165-6-48-56.

4. Ghaziamin P. A Privacy-Preserving Edge Computing Solution for Real-Time Passenger Counting at Bus Stops using Overhead Fisheye Camera : Master of Applied Science thesis. Montreal : Concordia University, 2023.

5. Karim H., Doshi K., Yilmaz Y. Real-Time Weakly Supervised Video Anomaly Detection. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV). 2024. P. 6848–6856. DOI: 10.1109/WACV57701.2024.00670.

6. Li Y., Padmanabhan A., Zhao P., Wang Y., Xu G. H., Netravali R. Reducto: On-Camera Filtering for Resource-Efficient Real-Time Video Analytics. In: Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM ’20). 2020. DOI: 10.1145/3387514.3405874.

7. Liu P., Qi B., Banerjee S. EdgeEye: An Edge Service Framework for Real-Time Intelligent Video Analytics. In: Proceedings of the 1st International Workshop on Edge Systems, Analytics and Networking (EdgeSys). 2018. P. 1–6. DOI: 10.1145/3213344.3213345.

8. Shih W.-C., Wang Z.-Y., Kristiani E., Hsieh Y.-J., Sung Y.-H., Li C.-H., Yang C.-T. The Construction of a Stream Service Application with DeepStream and Simple

Realtime Server Using Containerization for Edge Computing. Sensors. 2025. Vol. 25, No. 1. Art. 259. DOI: 10.3390/s25010259.

9. Shin D.-J., Kim J.-J. A Deep Learning Framework Performance Evaluation to Use YOLO in Nvidia Jetson Platform. Applied Sciences. 2022. Vol. 12, No. 8. Art. 3734. DOI: 10.3390/app12083734.

10. Oranen L. Utilizing Deep Learning on Embedded Devices : master's thesis. Tampere : Tampere University, 2021.

11. Trinh Gia H. On the Evaluation of Neural Network Deployment Options : bachelor's thesis. Tampere : Tampere University, 2023.

12. Puumala J. Monitoring Dangerous Goods on Roads Using Computer Vision : master's thesis. Tampere : Tampere University, 2023.

13. Hu M., Luo Z., Pasdar A., Lee Y. C., Zhou Y., Wu D. Edge-Based Video Analytics: A Survey. arXiv preprint. 2023. arXiv:2303.14329.

14. NVIDIA TensorRT. Quick Start Guide. URL: https://docs.nvidia.com/deeplearning/tensorrt/latest/getting-started/quick-start-guide.html (дата звернення: 21.02.2026).

15. Hossain S., Lee D.-j. Deep Learning-Based Real-Time Multiple-Object Detection and Tracking from Aerial Imagery via a Flying Robot with GPU-Based Embedded Devices. Sensors. 2019. Vol. 19, No. 15. Art. 3371. DOI: 10.3390/s19153371.

16. Shih, W.-C., Wang, Z.-Y., Kristiani, E., Hsieh, Y.-J., Sung, Y.-H., Li, C.-H., & Yang, C.-T. (2025). The Construction of a Stream Service Application with DeepStream and Simple Realtime Server Using Containerization for Edge Computing. Sensors, 25(1), 259. https://doi.org/10.3390/s25010259

17. Mapping of pin no, module pin name with Jetson AGX Orin Carrier Board Block Diagram [Електронний ресурс]. NVIDIA Developer Forums. 2023. URL: https://forums.developer.nvidia.com/t/mapping-of-pin-no-module-pin-name-with-jetson-agx-orin-carrier-board-block-diagram/276842 (дата звернення: 05.04.2026).

18. NVIDIA Jetson Linux Developer Guide. Quick Start. URL: https://docs.nvidia.com/jetson/archives/r35.6.1/DeveloperGuide/IN/QuickStart.html (дата звернення: 05.05.2026)

19. NVIDIA DeepStream documentation. Service Maker for Python Developers. URL: https://docs.nvidia.com/metropolis/deepstream/dev-guide/text/DS_service_maker_python.html (дата звернення: 05.05.2026)

20. dusty-nv. Flask + REST : вебзастосунок для Jetson // jetson-inference. GitHub. URL: https://github.com/dusty-nv/jetson-inference/blob/master/docs/webrtc-flask.md (дата звернення: 05.06.2026)

21. A. Basulto-Lantsova, J. A. Padilla-Medina, F. J. Perez-Pinal and A. I. Barranco-Gutierrez, "Performance comparative of OpenCV Template Matching method on Jetson TX2 and Jetson Nano developer kits," 2020 10th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 2020, pp. 0812-0816, doi: 10.1109/CCWC47524.2020.9031166.

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

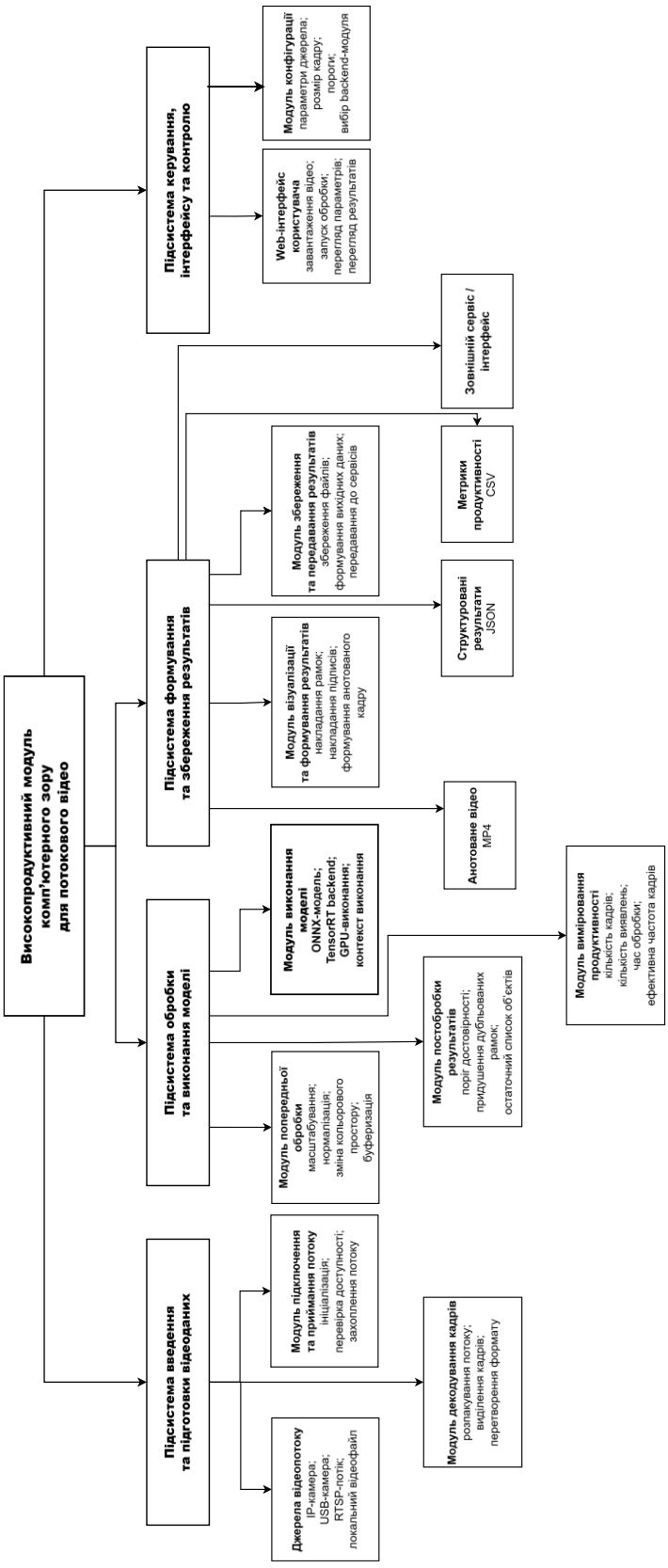
23. Островерхов В. М., Біловус Л. І., Возьний К. З., Луцишин О. О., Монастирський Г. Л., Надвиничний С. А., Питель С. В., Шандрук С. К. Загальні методичні рекомендації з підготовки, оформлення, захисту та оцінювання кваліфікаційних робіт здобувачів вищої освіти першого (бакалаврського) і другого (магістерського)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

24. ДСТУ 3008:2015. Національний стандарт України. Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання. [Чинний від 2016-07-01]. Київ: КР «УкрНДНЦ», 2016. 25 с.

25. Наконечний М. В. Високопродуктивний модуль комп’ютерного зору для потокового відео // Інтелектуальні комп’ютерні системи та мережі : матеріали V Всеукраїнської науково-практичної конференції, 20 травня, Тернопіль. Тернопіль : Західноукраїнський національний університет, факультет комп’ютерних інформаційних технологій, кафедра комп’ютерної інженерії, 2026. С.

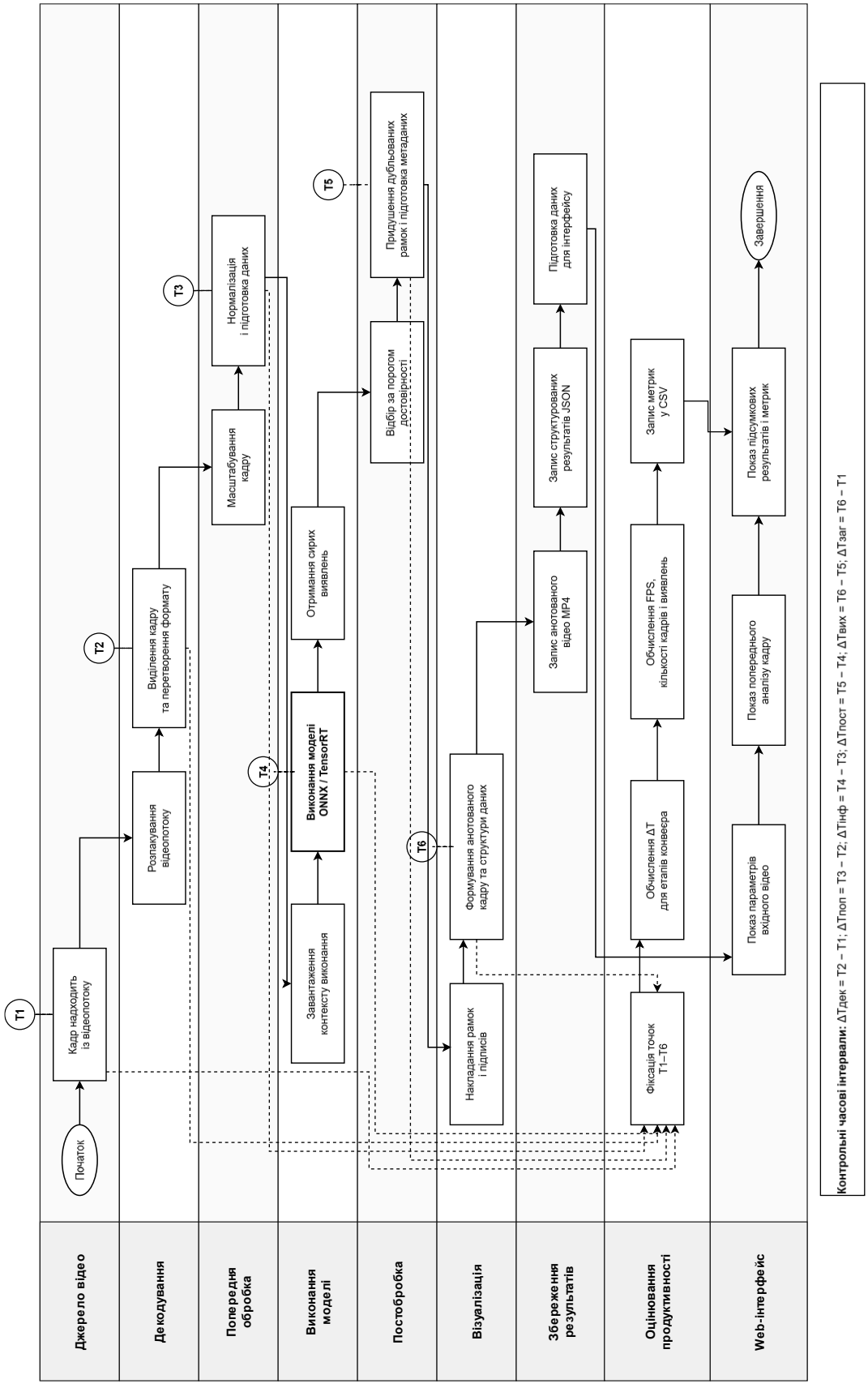
Додаток А

Розгорнута структурна схема високопродуктивного модуля комп’ютерного зору для потокового відео



Додаток Б

Діаграма проходження кадру та службових даних через модуль потокової відеоаналітики



## Додаток В

## Структура конфігураційного файла та форматів вихідних даних реалізованого модуля

Налаштування файлу config.yaml.

```
app:
  host: "0.0.0.0"
  port: 5000
  debug: true

pipeline:
  backend: "mock"
  source_type: "video_file"
  source_path: "uploads/input.mp4"
  output_video_path: "outputs/annotated_output.mp4"
  output_json_path: "outputs/detections.json"
  output_csv_path: "outputs/metrics.csv"

model:
  input_width: 640
  input_height: 480
  confidence_threshold: 0.50
  nms_threshold: 0.45
  classes:
    - "person"
    - "car"
    - "bicycle"

visualization:
  draw_boxes: true
  draw_labels: true
  draw_confidence: true
  draw_fps: true

processing:
  max_frames: 150
  save_video: true
  save_json: true
  save_csv: true
```

Файл detections.json для збереження структурованих результатів виявлення об’єктів.

```
[
  {
    "frame_index": 0,
    "detection_count": 2,
    "detections": [
      {
        "class_id": 0,
        "class_name": "person",
        "confidence": 0.93,
        "bbox": [118, 84, 262, 401]
      },
      {
```

```
        "class_id": 1,
        "class_name": "car",
        "confidence": 0.87,
        "bbox": [356, 210, 612, 428]
      }
    ]
  },
  {
    "frame_index": 1,
    "detection_count": 1,
    "detections": [
      {
        "class_id": 0,
        "class_name": "person",
        "confidence": 0.91,
        "bbox": [123, 88, 267, 405]
      }
    ]
  }
]
```

Файл metrics.csv для збереження підсумкових метрик продуктивності модуля.

```
metric_name,metric_value
processed_frames,150
total_detections,300
elapsed_time_sec,1.06
effective_fps,141.18
input_width,720
input_height,1280
input_fps,25.0
```

Приклади відповідності між окремими параметрами конфігураційного файла та результатами роботи системи.

**Таблиця В.1 – Вплив конфігураційних параметрів на роботу модуля**

| Параметр | Де використовується | На що впливає |
|---|---|---|
| backend | модуль pipeline.py і backend-модулі | визначає спосіб виконання моделі |
| source_path | video_source.py | задає шлях до вхідного відеофайлу |
| input_width, input_height | підготовка кадру | визначають розмір подання кадру для моделі |
| confidence_threshold | постобробка | впливає на кількість прийнятих виявлень |
| nms_threshold | постобробка | впливає на усунення дубльованих рамок |
| max_frames | pipeline.py | обмежує кількість кадрів під час тестового запуску |
| output_video_path | storage.py | задає шлях до файла анотованого відео |
| output_json_path | storage.py | задає шлях до файла структурованих результатів |
| output_csv_path | storage.py | задає шлях до файла метрик продуктивності |
| draw_boxes, draw_labels, draw_confidence | visualize.py | визначають склад графічних анотацій на кадрі |

Додаток Г
Апробація отриманих результатів

**Почесні співголови конференції:**

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

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

**Голова конференції:**

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

**Заступники голови конференції:**

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

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

## ПРОГРАМНИЙ КОМІТЕТ КОНФЕРЕНЦІЇ

**Склад програмного комітету:**

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

## ОРГАНІЗАЦІЙНИЙ КОМІТЕТ КОНФЕРЕНЦІЇ

**Склад організаційного комітету:**

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

**Технічна підтримка:**

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

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

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

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

**Тези доповідей подаються за оригіналом рукопису**

Наконечний М.В.
Студент 4 курсу ФКІТ ЗУНУ
Науковий керівник к.т.н., доцент Осолінський О.Р., кафедра ІОСУ ЗУНУ

# ВИСОКОПРОДУКТИВНИЙ МОДУЛЬ КОМП'ЮТЕРНОГО ЗОРУ ДЛЯ ПОТОКОВОГО ВІДЕО

**Вступ.** Сучасні системи відеоспостереження, транспортного моніторингу, виробничого контролю та робототехніки формують значні обсяги відеоданих, які потрібно аналізувати безпосередньо під час надходження. Передавання повного потоку до віддаленого сервера не завжди є доцільним, тому що це збільшує затримку, навантажує мережу та ускладнює автономну роботу системи. Тому все більше звертають увагу на граничну відеоаналітика, в якій основні етапи обробки виконуються поблизу джерела даних [1]. Для таких задач доцільно використовувати компактні апаратні платформи з графічним прискоренням, зокрема Nvidia Jetson, а також засоби перенесення та оптимізації моделей, такі як ONNX і TensorRT [2, 3].

Актуальність роботи полягає в потребі створення компактного модуля комп'ютерного зору, здатного приймати відеопотік, виконувати підготовку кадрів, запускати модель виявлення об'єктів, формувати результати та подавати їх користувачу в зручному вигляді..

**Постанова задачі.** Метою роботи є розробка високопродуктивного модуля комп'ютерного зору для потокового відео, орієнтованого на використання в edge-середовищі та подальше розгортання на платформі Nvidia Jetson. Об'єктом дослідження є процес потокової відеоаналітики в граничних системах комп'ютерного зору. Предметом дослідження є методи, алгоритми та програмні засоби приймання, обробки, аналізу й представлення результатів відеопотоку.

**Основний матеріал.** В процесі дослідження розглянуто сервісні фреймворки для edge-відеоаналітики, рішення на базі Nvidia Jetson і засоби апаратно-орієнтованого прискорення інференсу. Аналіз показав, що практична ефективність системи залежить не лише від вибору моделі, але також від організації повного конвеєра обробки [4].

Розроблена структура модуля передбачає роботу з кількома типами джерел: IP-камерою, USB-камерою, RTSP-потоком або локальним відеофайлом.

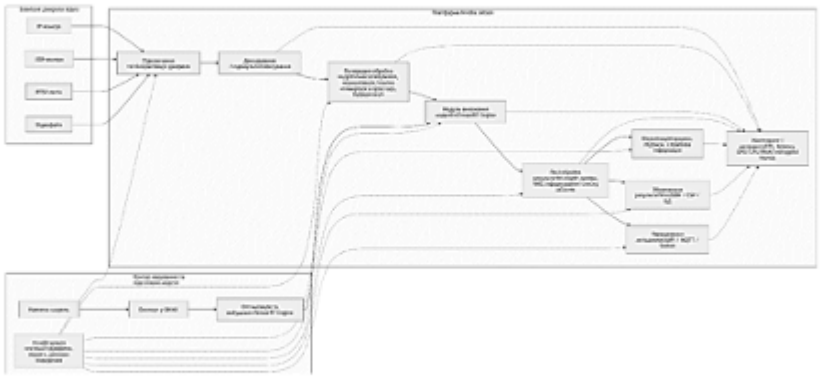


Рисунок 1 – Загальна структура модуля потокової відеоаналітики на платформі Nvidia Jetson

Після підключення джерела кадри проходять декодування, масштабування, нормалізацію та перетворення у формат, придатний для подання на модель виявлення об'єктів. Центральним елементом є backend виконання моделі. В проекті передбачено модульний підхід: поточний стенд використовує mock_backend.py для відпрацювання повного конвеєра, а окремий tensorrt_backend.py закладає можливість подальшого виконання оптимізованої моделі через TensorRT.

Програмна реалізація виконана в Linux-середовищі мовою Python. Для побудови веб-інтерфейсу використано Flask, для роботи з відеоданими - OpenCV, для числових операцій - NumPy, а для винесення налаштувань у зовнішній файл - PyYAML.

Функціональна логіка реалізована у вигляді окремих модулів. Файл main.py відповідає за маршрути веб-застосунку, завантаження відео та запуск обробки. Модуль video_source.py відкриває відеопотік і визначає його параметри. У pipeline.py зосереджено основну послідовність обробки кадрів: читання, виконання backend-модуля, нанесення рамок і формування підсумкової статистики. Модуль visualize.py відповідає за побудову анотованих кадрів, а storage.py зберігає вихідне відео, структуровані результати у JSON і метрики продуктивності у CSV.

Для взаємодії з користувачем створено тестову веб-сторінку, на якій можна завантажити відеофайл, переглянути службові параметри потоку, виконати попередній аналіз першого кадру та запустити повну обробку. Після завантаження відео система відображає розмір кадру, частоту кадрів, загальну кількість кадрів, кількість попередньо виявлених об'єктів і анотований кадр з рамками та підписами. Приклад інтерфейсу після завантаження тестового відео наведено на рисунку 2.

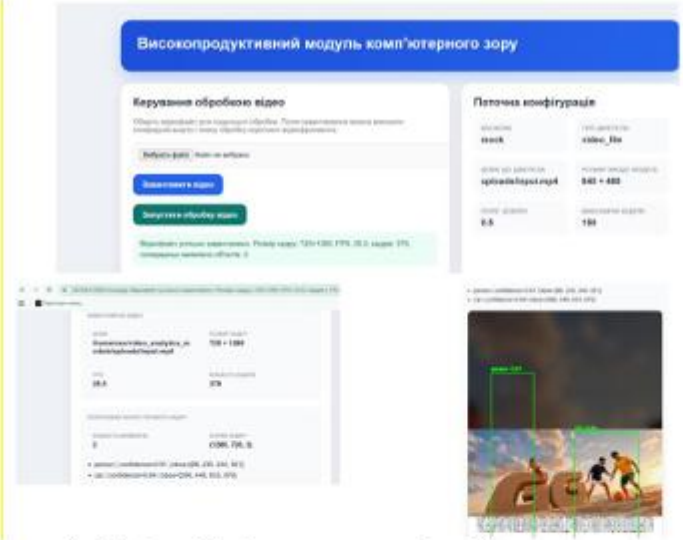



Рисунок 2 – Web-інтерфейс після завантаження відеофайлу та попереднього аналізу кадру

Під час тестування перевірено створення всіх вихідних файлів: анотованого відео у форматі MP4, файла detections.json зі структурованими результатами та файла metrics.csv з числовими показниками. В межах одного запуску було оброблено 150 кадрів, сформовано 300 виявлень, а повний час обробки становив 1,06 с.

**Висновки.** В результаті було розроблено програмний стенд високопродуктивного модуля комп'ютерного зору для потокового відео. Створений веб-інтерфейс забезпечує завантаження відеофайлу, запуск аналізу, відображення попередніх результатів і перегляд активної конфігурації. Перевірка показала, що модуль коректно обробляє тестовий відеофрагмент, формує анотоване відео, JSON-файл виявлень і CSV-файл метрик. Розроблене рішення може бути використане в задачах відеоспостереження, транспортного моніторингу, промислового контролю та навчальних досліджень з граничних обчислень.

**Список літератури**


1. Hu M., Luo Z., Pasdar A., Lee Y. C., Zhou Y., Wu D. Edge-Based Video Analytics: A Survey. arXiv preprint. 2023. arXiv:2303.14329.
2. NVIDIA Jetson Linux Developer Guide. Quick Start. URL: https://docs.nvidia.com/jetson/archives/r35.6.1/DeveloperGuide/IN/QuickStart.html (дата звернення: 05.04.2026).
3. NVIDIA TensorRT. Quick Start Guide. URL: https://docs.nvidia.com/deeplearning/tensorrt/latest/getting-started/quick-start-guide.html (дата звернення: 21.02.2026).
4. Liu P., Qi B., Banerjee S. EdgeEye: An Edge Service Framework for Real-Time Intelligent Video Analytics. EdgeSys. 2018. P. 1-6. DOI: 10.1145/3213344.3213345.
5. Basulto-Lantsova A., Padilla-Medina J. A., Perez-Pinal F. J., Barranco-Gutierrez A. I. Performance comparative of OpenCV Template Matching method on Jetson TX2 and Jetson Nano developer kits. CCWC. 2020. P. 0812-0816. DOI:10.1109/CCWC47524.2020.9031166.