

Міністерство освіти і науки України  
Державний університет «Житомирська політехніка»  
Факультет інформаційно-комп'ютерних технологій  
Кафедра комп'ютерної інженерії та кібербезпеки

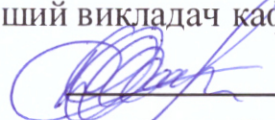
## ПОЯСНЮВАЛЬНА ЗАПИСКА

до випускної кваліфікаційної роботи бакалавра  
на тему: «Проект системи OSINT-моніторингу контенту  
Telegram-спільнот для задач приватного бізнесу»

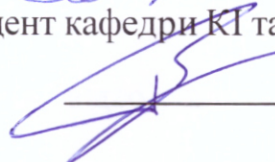
Виконав: студент 4-го курсу групи КБ-22-1  
спеціальності 125 «Кібербезпека»

 Ростислав АДАМЧУК

Керівник: Старший викладач кафедри КІ та КБ

 Ольга ДЯЧУК

Рецензент: Доцент кафедри КІ та КБ, к.т.н.

 Олександр ПІРОГ

Житомир – 2026

Державний університет «Житомирська політехніка»  
Факультет інформаційно-комп'ютерних технологій  
Кафедра комп'ютерної інженерії та кібербезпеки  
Спеціальність 125 «Кібербезпека»  
Освітня програма «Кібербезпека»

ЗАТВЕРДЖУЮ

Завідувач

кафедри комп'ютерної

інженерії та кібербезпеки

 Андрій Єфіменко

5 березня 2026 р.

### ЗАВДАННЯ

на випускню кваліфікаційну роботу бакалавра

Студент: Адамчук Ростислав Сергійович

Тема роботи: «Проєкт системи OSINT-моніторингу контенту Telegram-спільнот для задач приватного бізнесу»

затверджена Наказом університету від 2 лютого 2026 р. № 58/с

Термін здачі студентом закінченої роботи 10 червня 2026 р.

Вихідні дані роботи (зазначається предмет і об'єкт дослідження):

Об'єкт дослідження: Процеси автоматизованого збору, обробки та інтелектуального аналізу неструктурованих мультимедійних даних із відкритих джерел (соціальних медіа-платформ).

Предмет дослідження: Методи та програмні засоби OSINT-моніторингу та алгоритми комп'ютерного зору (Computer Vision) для автоматичної класифікації візуальних ознак кризових ситуацій та техногенних загроз.

Вихідні дані для роботи:

1. Потоки публічних повідомлень у месенджері Telegram (понад 1000 тематичних чатів).
2. Набір медіаданих (зображення, відео) для навчання та тестування нейронної мережі.
3. Документація Telegram API (MTProto) та протоколи передачі даних.

### Обладнання та програмне забезпечення:

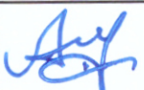
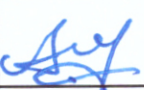
1. Апаратне забезпечення: ПК на базі процесора рівня Intel Core i7 (16GB RAM) та графічний прискорювач NVIDIA RTX серії (з підтримкою CUDA).

2. Операційні системи: Windows 11 (розробка), Linux Ubuntu 22.04 LTS (серверна частина).

3. Мови програмування та IDE: Python 3.10+, середовище розробки Py-Charm або VS Code.

4. Бібліотеки та технології: Telethon/Pyrogram (Telegram API), OpenCV, PyTorch/TensorFlow (архітектура YOLO).

Консультанти з випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються

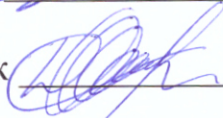
| Розділ | Консультант    | Підпис, дата                                                                                 |                                                                                                |
|--------|----------------|----------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------|
|        |                | Завдання видав                                                                               | Завдання прийняв                                                                               |
| 1      | Олексій ШЕЛУХА |  05.02.26   |  05.06.26   |
| 2      | Олексій ШЕЛУХА |  05.02.26 |  05.06.26 |
| 3      | Олексій ШЕЛУХА |  05.02.26 |  05.06.26 |

Студент



Ростислав АДАМЧУК

Керівник



Ольга ДЯЧУК

## КАЛЕНДАРНИЙ ПЛАН

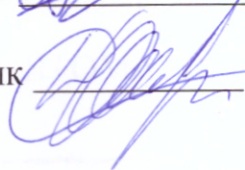
| № з/п | Назви етапів випускної кваліфікаційної роботи            | Термін виконання етапів роботи | Примітка |
|-------|----------------------------------------------------------|--------------------------------|----------|
| 1     | Постановка завдання, вибір теми та затвердження плану    | 05.02 – 15.02                  | Виконано |
| 2     | Збір та аналіз літературних джерел, вивчення основ OSINT | 16.02 – 10.03                  | Виконано |
| 3     | Аналіз існуючих систем моніторингу та Telegram API       | 11.03 – 31.03                  | Виконано |
| 4     | Проектування архітектури системи та бази даних           | 01.04 – 15.04                  | Виконано |
| 5     | Розробка програмного модуля збору даних (парсера)        | 16.04 – 30.04                  | Виконано |
| 6     | Розробка та навчання модуля ІІІ для фільтрації медіа     | 01.05 – 15.05                  | Виконано |
| 7     | Інтеграція модулів у єдину систему та тестування         | 16.05 – 22.05                  | Виконано |
| 8     | Аналіз отриманих результатів та оцінка точності ІІІ      | 23.05 – 30.05                  | Виконано |
| 9     | Оформлення пояснювальної записки (тексту диплома)        | 31.05 – 05.06                  | Виконано |
| 10    | Підготовка презентації та демонстраційного матеріалу     | 05.06 – 08.06                  | Виконано |
| 11    | Попередній захист та передача роботи на рецензію         | 10.06                          | Виконано |

Студент



Ростислав АДАМЧУК

Керівник



Ольга ДЯЧУК

## АНОТАЦІЯ

Кваліфікаційна робота присвячена розробці інтелектуальної системи автоматизованого OSINT-моніторингу відкритих інформаційних ресурсів у месенджері Telegram для виявлення та класифікації загроз бізнес-активам.

Пояснювальна записка складається з 97 сторінок, містить 37 рисунків, 7 таблиць. Список використаних джерел налічує 32 найменування та розміщений на 4 сторінках.

Метою роботи є створення програмного комплексу для оперативного збору та інтелектуальної фільтрації медіаконтенту, що дозволяє автоматично ідентифікувати кризові та техногенні ситуації (пожежі, задимлення, руйнування інфраструктури) для підвищення рівня корпоративної безпеки.

У ході дослідження визначено основні завдання для побудови систем ситуаційної обізнаності на основі відкритих даних. Проаналізовано базові методи OSINT-моніторингу та технічні особливості Telegram API. Розглянуто особливості застосування алгоритмів комп'ютерного зору для фільтрації візуального контенту в умовах високого інформаційного навантаження. Показано розробку модульної архітектури системи, що включає парсер для обробки понад 1000 джерел та нейромережевий класифікатор медіафайлів. Наведено результати тестування точності роботи навченої моделі III та оцінено продуктивність системи при масштабуванні кількості об'єктів моніторингу. Практична значущість отриманих результатів полягає у створенні дієвого інструменту для автоматизації роботи аналітиків з безпеки, що дозволяє суттєво скоротити час від моменту виникнення надзвичайної події до прийняття відповідного управлінського рішення.

Ключові слова: КІБЕРБЕЗПЕКА, OSINT, МОНІТОРИНГ, TELEGRAM API, МАШИННЕ НАВЧАННЯ, КОМП'ЮТЕРНИЙ ЗІР, ІНТЕЛЕКТУАЛЬНА АНАЛІТИКА, ТЕХНОГЕННІ ЗАГРОЗИ, АВТОМАТИЗАЦІЯ.

|           |      |                 |        |          |                                                                                                |      |         |
|-----------|------|-----------------|--------|----------|------------------------------------------------------------------------------------------------|------|---------|
|           |      |                 |        |          | ККІКБ.КР.Б – 125 – 01 – ПЗ                                                                     |      |         |
| Змн.      | Арк. | № докум.        | Підпис | Дата     | Проєкт системи OSINT-моніторингу<br>контенту Telegram-спільнот для задач<br>приватного бізнесу |      |         |
| Розроб.   |      | Адамчук Р. С.   |        | 03.06.24 |                                                                                                |      |         |
| Керівник  |      | Дячук О. Ю.     |        | 03.06.24 |                                                                                                |      |         |
| Реценз.   |      | Пірог О. В.     |        | 03.06.24 |                                                                                                |      |         |
| Н. контр. |      | Дячук О. Ю.     |        | 03.06.24 |                                                                                                |      |         |
| Затверд.  |      | Сфіменко А.А. С |        | 04.08.24 |                                                                                                |      |         |
|           |      |                 |        |          | Літ.                                                                                           | Арк. | Аркушів |
|           |      |                 |        |          |                                                                                                | 5    |         |
|           |      |                 |        |          | ФІКТ, Гр. КБ-22-1                                                                              |      |         |

## ANNOTATION

The bachelor's thesis is focused on the development of an intelligent system for automated OSINT monitoring of open information resources in the Telegram messenger to detect and classify threats to business assets.

The explanatory note consists of 97 pages, contains 37 figures, and 7 tables. The list of used sources includes 32 items and is located on 4 pages.

The aim of the work is to create a software complex for rapid collection and intelligent filtering of media content, which allows for the automatic identification of crisis and man-made situations (fires, smoke, infrastructure damage) to enhance the level of corporate security.

In the course of the research, the main tasks for building situational awareness systems based on open-source data were defined. The basic methods of OSINT monitoring and the technical features of the Telegram API were analyzed. The peculiarities of applying computer vision algorithms for filtering visual content under high information load conditions were considered. The development of a modular system architecture is shown, including a parser for processing over 1000 sources and a neural network classifier for media files. The results of testing the accuracy of the trained AI model are presented, and the system's performance during the scaling of monitoring objects is evaluated. The practical significance of the results obtained lies in the creation of an effective tool for automating the work of security analysts, which significantly reduces the time from the moment an emergency occurs to the making of an appropriate management decision.

**Keywords:** CYBERSECURITY, OSINT, MONITORING, TELEGRAM API, MACHINE LEARNING, COMPUTER VISION, INTELLIGENT ANALYTICS, MAN-MADE THREATS, AUTOMATION.

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            |      |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            | 6    |

## ЗМІСТ

|                                                                                                               |    |
|---------------------------------------------------------------------------------------------------------------|----|
| ВСТУП.....                                                                                                    | 10 |
| РОЗДІЛ 1. АНАЛІЗ МЕТОДІВ МОНІТОРИНГУ ІНФОРМАЦІЙНИХ РЕСУРСІВ .....                                             | 12 |
| 1.1. Основи OSINT та його роль у сучасному бізнесі .....                                                      | 12 |
| 1.2. Особливості Telegram як платформи для збору інформації .....                                             | 14 |
| 1.3. Проблеми та ризики роботи з Telegram-спільнотами .....                                                   | 19 |
| 1.4. Огляд існуючих систем OSINT-моніторингу та їх функціональні можливості.....                              | 25 |
| 1.5. Юридичні обмеження та етичні норми автоматизованого збору публічних даних.....                           | 32 |
| Висновки до першого розділу.....                                                                              | 37 |
| РОЗДІЛ 2. МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ СИСТЕМИ ІНТЕЛЕКТУАЛЬНОГО МОНІТОРИНГУ .....                              | 39 |
| 2.1. Формування технічних вимог до системи (масштабованість, швидкість обробки, точність фільтрації). .....   | 39 |
| 2.2. Проєктування архітектури системи: взаємодія модулів збору, черг повідомлень та блоку аналітики.....      | 45 |
| 2.3. Вибір та обґрунтування технологічного стеку (Python, асинхронні бібліотеки Telethon/Pyrogram, СУБД)..... | 52 |
| 2.4. Проєктування інтелектуального модуля фільтрації медіаконтенту на основі нейромереж .....                 | 54 |
| 2.5. Розробка алгоритму функціонування системи та логіки маршрутизації даних.....                             | 58 |
| Висновки до другого розділу .....                                                                             | 60 |
| РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ СИСТЕМИ ІНТЕЛЕКТУАЛЬНОГО OSINT-МОНІТОРИНГУ .....                 | 62 |
| 3.1. Реалізація модуля збору даних .....                                                                      | 62 |

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            |      |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            | 7    |

|                                                                                             |    |
|---------------------------------------------------------------------------------------------|----|
| 3.2. Налаштування та навчання моделі ШІ для розпізнавання візуальних маркерів загроз. ....  | 70 |
| 3.3. Тестування продуктивності конвеєра обробки та сценарії розгортання системи .....       | 79 |
| 3.4. Оцінка продуктивності системи при високому навантаженні та аналіз масштабованості..... | 83 |
| Висновки до третього розділу.....                                                           | 87 |
| ВИСНОВКИ.....                                                                               | 89 |
| СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....                                                             | 91 |
| ДОДАТКИ.....                                                                                | 95 |

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            | 8    |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            |      |

## ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

**API** (Application Programming Interface) – програмний інтерфейс застосунку

**Bot API** – протокол керування автоматизованими акаунтами в Telegram

**CLIP** (Contrastive Language-Image Pre-training) – архітектура нейромережі для співставлення текстового та графічного контенту

**CNN** (Convolutional Neural Network) – згорткова нейронна мережа

**Cosine similarity** – косинусна подібність, метрика близькості векторів

**CV** (Computer Vision) – комп'ютерний зір

**Embeddings** – ембеддинги, векторні представлення об'єктів у просторі

**Fine-tuning** – донавчання (тонке налаштування) нейронної мережі

**Latency** – затримка, час очікування початку обробки запиту

**mAP** (Mean Average Precision) – середня точність, метрика якості розпізнавання

**ML** (Machine Learning) – машинне навчання

**MTPROTO** – криптографічний протокол взаємодії із серверами Telegram

**OSINT** (Open Source Intelligence) – розвідка на основі відкритих джерел

**Perceptual hashing** – перцептивне хешування для пошуку дублікатів медіа

**RTSP** (Real Time Streaming Protocol) – протокол трансляції потокового мультимедіа

**VRAM** (Video Random Access Memory) – відеопам'ять графічного процесора

**YOLO** (You Only Look Once) – нейромережа для розпізнавання об'єктів у реальному часі

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            | 9    |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            |      |

## ВСТУП

У сучасних умовах глобальної цифровізації, що супроводжується стрімким зростанням обсягів інформаційних потоків та геополітичною нестабільністю, швидкість отримання достовірних даних про надзвичайні події стає вирішальним фактором. Для суб'єктів господарювання та державних інституцій оперативне виявлення загроз є основою забезпечення безпеки бізнес-активів та захисту об'єктів критичної інфраструктури.

Натомість сучасні соціальні месенджери, серед яких лідируючу позицію займає Telegram, перетворилися на інструменти первинної фіксації будь-яких інцидентів (пожеж, технологічних аварій, логістичних колапсів) безпосередніми очевидцями. Це створює унікальні можливості для OSINT-розвідки (Open Source Intelligence), проте водночас висуває серйозний виклик – проблему «інформаційного шуму» та надмірності даних. Опрацювання тисяч повідомлень у реальному часі силами людського оператора стає фізично неможливим, що зумовлює критичну необхідність в автоматизації цих процесів.

Застосування методів штучного інтелекту, зокрема комп'ютерного зору, дозволяє трансформувати хаотичний потік медіаданих у структуровану аналітичну інформацію. Розробка інтелектуальної системи, здатної самостійно диференціювати релевантні зображення та відео (наприклад, кадри із задимленням, вогнем чи руйнуваннями) від повсякденного контенту, є не лише технічним завданням, а стратегічним кроком до підвищення рівня корпоративної та кібербезпеки в умовах гібридних викликів [26].

Метою кваліфікаційної роботи є розробка та впровадження інтелектуальної програмної системи автоматизованого OSINT-моніторингу для оперативного виявлення, фільтрації та класифікації візуальних ознак кризових ситуацій у реальному часі.

Визначена мета обумовлює наступні завдання:

1. Проаналізувати теоретичні аспекти OSINT-розвідки та специфіку Telegram як джерела оперативних даних у контексті безпеки бізнесу [14, 27, 31].

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            | 10   |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            |      |

2. Спроекувати архітектуру системи для паралельного моніторингу великої кількості джерел (1000+) з урахуванням вимог до відмовостійкості та високого навантаження.

3. Розробити програмний модуль для автоматизованого збору мультимедійних даних та вилучення супутніх метаданих повідомлень.

4. Створити та адаптувати модель на основі комп'ютерного зору для автоматичної класифікації специфічних візуальних ознак загроз (вогонь, дим, структурні руйнування).

5. Провести комплексне тестування системи на реальних потоках даних, оцінити точність алгоритмів та потенційну ефективність впровадження.

Об'єктом дослідження є процеси автоматизованого збору, обробки та інтелектуального аналізу неструктурованих мультимедійних даних із відкритих джерел (соціальних медіа-платформ).

Предметом дослідження є методи та програмні засоби OSINT-моніторингу та алгоритми комп'ютерного зору (Computer Vision) для автоматичної класифікації візуальних ознак кризових ситуацій та техногенних загроз.

У процесі виконання кваліфікаційної роботи було розроблено проєкт автоматизованої системи ситуаційної обізнаності, що базується на асинхронних принципах обробки даних. Проєктне рішення включає багаторівневий парсер, що здійснює безперервний моніторинг понад 1000 інформаційних каналів, та інтегрований нейромережевий модуль, який виконує селекцію контенту на основі навчених патернів кризових подій.

Реалізований підхід дозволяє трансформувати систему збору даних у повноцінний аналітичний інструмент, який автоматично агрегує лише критично важливу інформацію у спеціалізований операційний канал. Це забезпечує мінімізацію впливу людського фактора, знижує навантаження на аналітиків та створює фундамент для прийняття негайних управлінських рішень у сфері безпеки та антикризового менеджменту.

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            |      |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            | 11   |

# РОЗДІЛ 1. АНАЛІЗ МЕТОДІВ МОНІТОРИНГУ ІНФОРМАЦІЙНИХ РЕСУРСІВ

## 1.1. Основи OSINT та його роль у сучасному бізнесі

У сучасному інформаційному суспільстві дані стали стратегічним ресурсом, а здатність до їх швидкого пошуку та аналізу – запорукою конкурентоспроможності приватного бізнесу. Концепція OSINT (Open Source Intelligence), що виникла як методологія розвідки на основі відкритих джерел, наразі активно інтегрується у бізнес-середовище під назвою Business Intelligence (BI) або корпоративна розвідка.

Основною перевагою OSINT є легальність збору інформації, оскільки об'єктом аналізу є дані, що добровільно та публічно розміщені в мережі Інтернет. Для приватного бізнесу використання методів OSINT дозволяє вирішувати широкий спектр завдань, представлених на рис. 1.1.



Рисунок 1.1 – Основні напрями застосування OSINT у приватному бізнесі

Як показано на рис. 1.1, одним із найбільш критичних напрямів є моніторинг загроз активам. В умовах фізичних ризиків (пожежі, техногенні аварії, соціальні заворушення) бізнес потребує інформації швидше, ніж її можуть надати офіційні

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            | 12   |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            |      |

медіа-ресурси. Швидкість реакції безпосередньо корелює з мінімізацією фінансових збитків.

Систематизація завдань бізнес-моніторингу наведена у табл. 1.1.

**Таблиця 1.1**

**Завдання та переваги впровадження OSINT-систем**

| <b>Категорія ризику</b> | <b>Об'єкт моніторингу</b>          | <b>Основна перевага для бізнесу</b>         |
|-------------------------|------------------------------------|---------------------------------------------|
| Фізична безпека         | Склади, логістичні вузли, офіси    | Попередження про інциденти в реальному часі |
| Репутаційні ризики      | Згадки бренду, відгуки клієнтів    | Оперативне управління кризовим PR           |
| Економічна безпека      | Контрагенти, конкуренти            | Верифікація партнерів та аналіз ринку       |
| Кібербезпека            | Витоки даних, технічні вразливості | Своєчасне виявлення скомпрометованих даних  |

Таким чином, OSINT для приватного сектора трансформується з методу пасивного спостереження на інструмент ситуаційної обізнаності (Situational Awareness). Це дозволяє керівництву та службам безпеки приймати стратегічні управлінські рішення на основі первинних даних, таких як фото- та відеоматеріали безпосередніх очевидців подій. Такий підхід є критично важливим для компаній з розгалуженою інфраструктурою, мережею складських приміщень або складними логістичними ланцюгами, де затримка інформації навіть на декілька хвилин може призвести до незворотних фінансових та репутаційних втрат [31].

Проте, сучасний етап розвитку цифрових комунікацій характеризується явищем «інформаційного вибуху». Велика кількість месенджерів та соціальних мереж генерує гігабайти контенту щосекунди, що створює проблему критичної надмірності даних. У контексті моніторингу Telegram-спільнот, аналітик стикається з необхідністю опрацювання сотень каналів, де корисна інформація про реальні інциденти становить лише 1–3% від загального потоку повідомлень. Решта

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            |      |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            | 13   |

контенту – це інформаційний шум: реклама, повідомлення, дублікати новин та медіафайли, що не мають відношення до безпеки об'єктів.

Людський фактор за таких умов стає найслабшою ланкою системи безпеки. Втома оператора, обмежена швидкість сприйняття візуальної інформації та суб'єктивність аналізу призводять до того, що критично важливі маркери загрози можуть бути проігноровані. Це створює гостру потребу в автоматизації процесу збору та, що найважливіше, інтелектуальної фільтрації контенту [8].

Впровадження автоматизованих алгоритмів дозволяє не лише збирати дані 24/7 без перерв, а й використовувати потужності обчислювальних систем для миттєвої селекції контенту. Саме розв'язання задачі автоматичного відсіювання «інформаційного сміття» та фокусування уваги лише на релевантних медіафайлах, що містять візуальні ознаки загроз, є ключовим технічним завданням даної кваліфікаційної роботи. Це дозволяє перевести систему моніторингу з формату «пошуку вручну» у формат «сповіщення за фактом виявлення», що радикально змінює швидкість та ефективність антикризового менеджменту в приватному бізнесі [27].

## 1.2. Особливості Telegram як платформи для збору інформації

Месенджер Telegram на сьогодні є одним із найпотужніших інструментів для OSINT-досліджень завдяки своїй унікальній архітектурі, швидкості поширення контенту та лояльній політиці щодо автоматизації збору даних. На відміну від класичних соціальних мереж (таких як Facebook або Instagram), які використовують алгоритмічні стрічки новин, Telegram працює за принципом прямої хронологічної доставки повідомлень. Це фактично перетворює платформу на децентралізоване інформаційне агентство, де швидкість появи публікації про критичну подію від очевидця часто випереджає офіційні джерела або великі медіа-холдинги на 15–30 хвилин. Для приватного бізнесу ця часова перевага є вирішальною, оскільки вона дозволяє ініціювати протоколи безпеки ще до того, як інцидент набуде масового розголосу.

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            |      |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            | 14   |

Особлива цінність Telegram полягає у феномені «гіперлокальності». Платформа стала базою для формування тисяч вузькоспеціалізованих та географічно прив'язаних спільнот: від чатів окремих житлових комплексів та промислових зон до закритих професійних груп логістів, експедиторів та енергетиків. У таких спільнотах інформація про пожежу, аварію на підстанції чи перекриття транспортної магістралі з'являється у вигляді «сирих» даних (фото, відео, короткі текстові замітки) від безпосередніх учасників або свідків подій.

Структурно Telegram поділяється на кілька функціональних типів джерел, кожен з яких відіграє свою роль у загальній системі моніторингу ризиків (див. табл. 1.2). Наприклад, якщо публічні канали дають змогу оцінити загальний масштаб події, то приватні та локальні чати дозволяють отримати деталізовану інформацію з прив'язкою до конкретних координат.

**Таблиця 1.2**

**Класифікація джерел інформації в Telegram для задач моніторингу**

| Тип джерела          | Специфіка контенту                                  | Цінність для бізнес-безпеки                                |
|----------------------|-----------------------------------------------------|------------------------------------------------------------|
| Публічні канали      | Одностороннє мовлення, офіційні звіти, великі медіа | Отримання підтверджених фактів та офіційних повідомлень    |
| Географічні чати     | Повідомлення від мешканців конкретних районів/міст  | Максимально швидка фіксація інцидентів «на місці»          |
| Професійні спільноти | Обмін досвідом, обговорення проблем галузі          | Виявлення прихованих ризиків у логістиці чи інфраструктурі |
| Бот-системи          | Автоматизована розсилка повідомлень, фідбек-форми   | Збір структурованих технічних даних                        |

Технічною основою для реалізації систем автоматизованого моніторингу є протокол MTProto та відкритий Telegram API. На відміну від конкурентів

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            |      |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            | 15   |

(наприклад, WhatsApp або Viber), Telegram надає розробникам офіційні інструменти для створення клієнтських додатків, що дозволяє отримувати доступ до повідомлень без необхідності імітації дій реального користувача через браузер (web-scraping).

Процес збору даних через API має ієрархічну структуру, яка дозволяє системі ефективно масштабуватися. Схематично взаємодію системи моніторингу з платформою представлено на рис. 1.2.

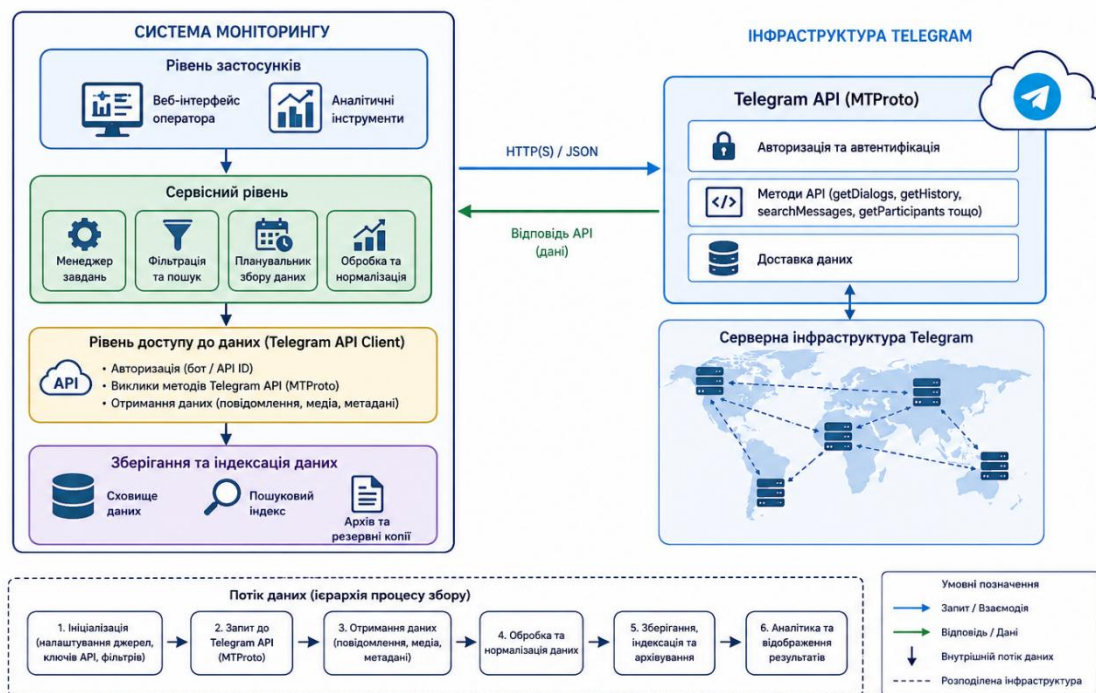


Рисунок 1.2 – Схема взаємодії системи моніторингу з інфраструктурою Telegram

Важливою технологічною перевагою Telegram є механіка обробки та зберігання мультимедійного контенту. На відміну від більшості популярних месенджерів та соціальних мереж, які агресивно стискають зображення та відео задля економії серверного простору, Telegram дозволяє завантажувати та зберігати файли в оригінальній якості або з використанням алгоритмів, що зберігають високу деталізацію об'єктів. Це є критично важливим аспектом для подальшого етапу інтелектуальної обробки даних, оскільки алгоритми комп'ютерного зору та нейронні мережі потребують чітких візуальних ознак для безпомилкової класифікації інцидентів, таких як задимлення, вогнища чи структурні пошкодження будівель.

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            | 16   |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            |      |

Крім того, кожне повідомлення у системі супроводжується багатим набором метаданих. Наявність унікальних ідентифікаторів (Message ID), часових міток з точністю до секунди та ідентифікаторів відправників (User/Peer ID) дозволяє автоматизованій системі будувати точну часову шкалу розвитку подій. Це дає можливість не просто фіксувати факт події, а й проводити крос-верифікацію: якщо одне і те саме відео з'являється у десяти різних чатах з різними часовими мітками, система може автоматично визначити першоджерело та відстежити швидкість і географію поширення інформації.

З точки зору кібербезпеки та конфіденційності, архітектура Telegram пропонує багаторівневі механізми шифрування та гнучкі налаштування анонімності. Для приватного бізнесу це створює безпечне середовище для проведення моніторингу без прямого розкриття своєї присутності у ворожих, конкурентних чи потенційно небезпечних інформаційних сегментах. Використання клієнтських додатків на базі API дозволяє працювати у режимі «невидимості», не позначаючи повідомлення як прочитані, що є важливим для збереження операційної скритності моніторингових процесів.

Проте масштабування системи до рівня обробки понад 1000 джерел одночасно висуває жорсткі вимоги до архітектури менеджменту запитів. Основною технічною перешкодою у промисловому зборі даних є жорсткі обмеження платформи, відомі як Flood Limits. Сервери Telegram суворо контролюють інтенсивність взаємодії: у разі перевищення ліміту запитів на отримання історії повідомлень або завантаження медіафайлів, клієнтський обліковий запис отримує примусове блокування (FloodWait).

Час очікування може варіюватися від декількох хвилин до годин, що створює загрозу розриву інформаційного потоку. Така специфіка роботи API вимагає впровадження інтелектуальних алгоритмів планування запитів, які здатні динамічно адаптуватися до лімітів серверної сторони без повної зупинки процесу моніторингу. З точки зору кібербезпеки, тривалі затримки в отриманні даних створюють «сліпі зони», у яких критичні інциденти можуть залишатися непоміченими, що нівелює ефективність усієї системи превентивного захисту. Для

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            |      |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            | 17   |

систем екстреного реагування, де кожна хвилина має значення для безпеки активів, така деградація сервісу є неприпустимою.

Для візуалізації переваг обраного архітектурного рішення порівняємо стандартну послідовну модель збору даних з асинхронною моделлю, яка реалізована у даному проєкті (див. рис. 1.3).



Рисунок 1.3 – Порівняльна схема послідовної та асинхронної обробки медіапотоків

Саме тому при проєктуванні архітектури програмного комплексу в межах даної роботи було обрано асинхронну модель збору даних на базі сучасних інструментів мови Python (зокрема бібліотек Telethon та Asyncio). На відміну від класичного послідовного (синхронного) підходу, де система змушена простоювати в очікуванні відповіді від кожного окремого чату, асинхронність дозволяє підтримувати сотні паралельних з'єднань у межах єдиного потоку керування.

Такий підхід дозволяє реалізувати інтелектуальний менеджер черг, який раціонально розподіляє навантаження, чергує запити до різних категорій джерел та динамічно керує часовими затримками між діями. Це не лише мінімізує ризики ініціації антифлуд механізмів месенджера, а й гарантує безперервність та цілісність надходження критично важливої інформації. У підсумку це забезпечує формування максимально повної та актуальної аналітичної вибірки медіаконтенту, що є фундаментальною умовою для подальшої коректної роботи модуля штучного інтелекту.

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            | 18   |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            |      |

### 1.3. Проблеми та ризики роботи з Telegram-спільнотами

Ефективність OSINT-моніторингу в середовищі Telegram безпосередньо залежить від здатності інтелектуальної системи оперативно та прецизійно відокремлювати критично важливі сигнали від колосального обсягу нерелевантної інформації. У сучасних цифрових екосистемах основним деструктивним фактором виступає «інформаційний шум», обсяг якого у відкритих групах, географічних чатах та публічних каналах може сягати 95–98% від загального вхідного трафіку. Для приватного бізнесу таке співвідношення корисних даних до загального потоку створює критичний ризик пропуску реальної загрози через ефект «когнітивного перевантаження» аналітика або технічне перенасичення стрічки сповіщень.

Особлива складність полягає у тому, що нерелевантний контент часто маскується під термінові повідомлення або містить клікбейтні заголовки, які змушують стандартні фільтри на основі ключових слів спрацьовувати помилково. Це призводить до деградації системи безпеки: велика кількість хибнопозитивних результатів (False Positives) нівелює цінність моніторингу, оскільки оператор з часом втрачає пильність до сповіщень. Систематизацію основних типів контенту, що перешкоджають ефективному моніторингу та створюють додаткове навантаження на програмний комплекс, наведено у табл. 1.3.

Таблиця 1.3

#### Класифікація нерелевантного контенту в Telegram-спільнотах

| Тип контенту              | Характеристика                                      | Вплив на систему моніторингу                        |
|---------------------------|-----------------------------------------------------|-----------------------------------------------------|
| Комерційна реклама        | Просування товарів, послуг або інших каналів        | Хибні спрацювання за ключовими словами              |
| Інформаційний флуд        | Повідомлення, що не несуть змістовного навантаження | Засмічення бази даних та черги обробки              |
| Дубльований контент       | Репости одного й того самого медіафайлу             | Параліч роботи аналітика через ідентичні сповіщення |
| Клікбейт та дезінформація | Маніпулятивні заголовки, старі фото/відео           | Прийняття помилкових управлінських рішень           |

Першою серйозною перешкодою на шляху до отримання чистої вибірки даних є агресивна та нерегульована монетизація сучасних Telegram-каналів. Оскільки власники інформаційних ресурсів прагнуть максимізувати охоплення та прибуток, більшість популярних спільнот інтегрують рекламні дописи, що використовують методи психологічного впливу та маніпулятивну лексику. Такі рекламні інтеграції часто містять специфічні слова-тригери, які за своєю семантикою збігаються з маркерами надзвичайних подій (наприклад, «вибухові знижки», «гарячі пропозиції», «шок-новини», «терміновий збір»).

Для класичних текстових фільтрів, що базуються на регулярних виразах або простих методах ключових слів, такий контент стає джерелом масових хибнопозитивних спрацювань. Ситуація ускладнюється тим, що рекламодавці часто супроводжують текст яскравими зображеннями або відео, які за кольоровою гамою (червоні, помаранчеві відтінки) можуть бути помилково ідентифіковані примітивними алгоритмами як ознаки вогню або задимлення.

Окремим викликом для автоматизованих систем є використання методів «взаємного піару» та мережевого поширення однакових рекламних повідомлень через так звані «сітки» каналів. У таких випадках одне й те саме рекламне оголошення, часто супроводжене ідентичним медіафайлом, може з'являтися у десятках джерел одночасно з інтервалом у кілька секунд. Це призводить до лавиноподібного зростання кількості вхідних подій у системі моніторингу, створюючи штучне пікове навантаження на обчислювальні вузли. Без механізмів попереднього хешування та швидкого відсіювання повторюваних пакетів даних, програмний комплекс ризикує витратити значну частину ресурсів на багаторазову обробку одного й того самого шуму, що безпосередньо впливає на затримку отримання дійсно критичної інформації.

Крім того, рекламні дописи в Telegram часто використовують формат «термінових новин», що мімікрують під повідомлення про реальні інциденти, аби привернути увагу підписника. Це створює додаткове навантаження на систему моніторингу, оскільки кожен такий об'єкт потребує верифікації на рівні глибинного аналізу контенту. У разі відсутності інтелектуальних модулів фільтрації, подібний

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            | 20   |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            |      |

інформаційний шум може повністю дезорієнтувати аналітиків з безпеки, змушуючи їх витратити операційний час на перевірку неіснуючих загроз бізнес-активам. Наочний приклад типового рекламного контенту, що створює значне навантаження на черги обробки системи та потребує відсіювання, представлено на рис. 1.4.

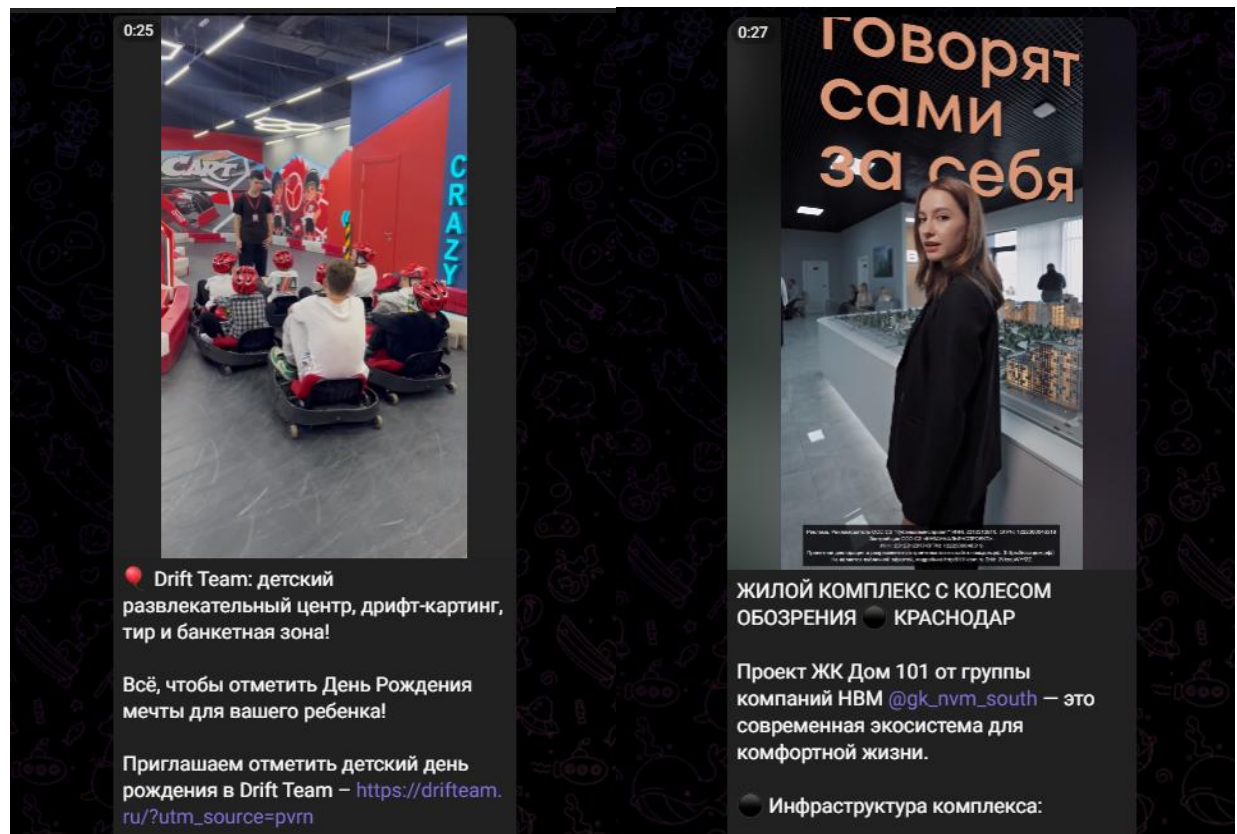


Рисунок 1.4 – Приклад рекламного контенту в Telegram, що створює надлишкове навантаження на систему

Другою фундаментальною проблемою, що критично впливає на достовірність результатів моніторингу, є складна верифікація мультимедійного контенту. У моменти виникнення кризових ситуацій або резонансних подій інформаційне поле наповнюється значною кількістю неперевіраних даних. Користувачі месенджера, навмисно або через надмірну емоційність, часто поширюють архівні фотографії, кадри з художніх фільмів або відеоматеріали з інших географічних регіонів, видаючи їх за актуальні події, що відбуваються «тут і зараз». Без впровадження методів інтелектуального аналізу специфічних візуальних ознак (таких як архітектурні особливості, погодні умови, рослинність)

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            | 21   |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            |      |

та поглибленої звірки метаданих, автоматизована система може ініціювати серію хибних тривог, що призведе до марної активації протоколів безпеки підприємства.

Додатково ситуація ускладнюється явищем цифрової маніпуляції, де автентичні кадри можуть бути змінені за допомогою графічних редакторів для штучного посилення масштабу інциденту. Відсутність інтелектуального фільтра, здатного розпізнавати невідповідності у візуальному контенті, робить бізнес-активи вразливими до інформаційних атак, спрямованих на створення штучної паніки або дестабілізацію операційної діяльності компанії.

Окрім проблем верифікації, виникає системна проблема «дзеркального ефекту» та вірального розповсюдження даних. У сучасному медіапросторі будь-яка непересічна подія фіксується десятками або навіть сотнями випадкових свідків з різних ракурсів, після чого одне й те саме відео миттєво дублюється у мережах регіональних та загальнонаціональних каналів. Якщо архітектура системи моніторингу не передбачає складних механізмів дедуплікації, що базуються на перцептивному хешуванні зображень (Perceptual Hashing), оператор безпеки отримає лавиноподібну кількість ідентичних або майже ідентичних сповіщень. Зокрема, ігнорування етапу інтелектуального порівняння медіафайлів призводить до нераціонального використання обчислювальних ресурсів на повторний аналіз вже оброблених об'єктів. Це створює критичні затримки у формуванні підсумкових звітів, що в умовах динамічної загрози може стати вирішальним фактором, який перешкоджає своєчасному реагуванню на інцидент.

Така надмірність створює ефект «паралічу аналізу», коли реальні, унікальні деталі інциденту губляться у потоці копій. Програмний комплекс має не лише відсікати дублікати, а й групувати схожі медіафайли у єдині інформаційні кейси, дозволяючи аналітику бачити повну картину події без необхідності перегляду повторюваного контенту. Розуміння механізмів циркуляції таких даних є ключовим для побудови надійної системи захисту. Схему життєвого циклу інформації – від моменту первинної фіксації інциденту свідком до етапу верифікації та потрапляння в аналітичний звіт системи моніторингу – детально продемонстровано на рис. 1.5.

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            |      |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            | 22   |

### Схема життєвого циклу та розповсюдження інформації в середовищі Telegram

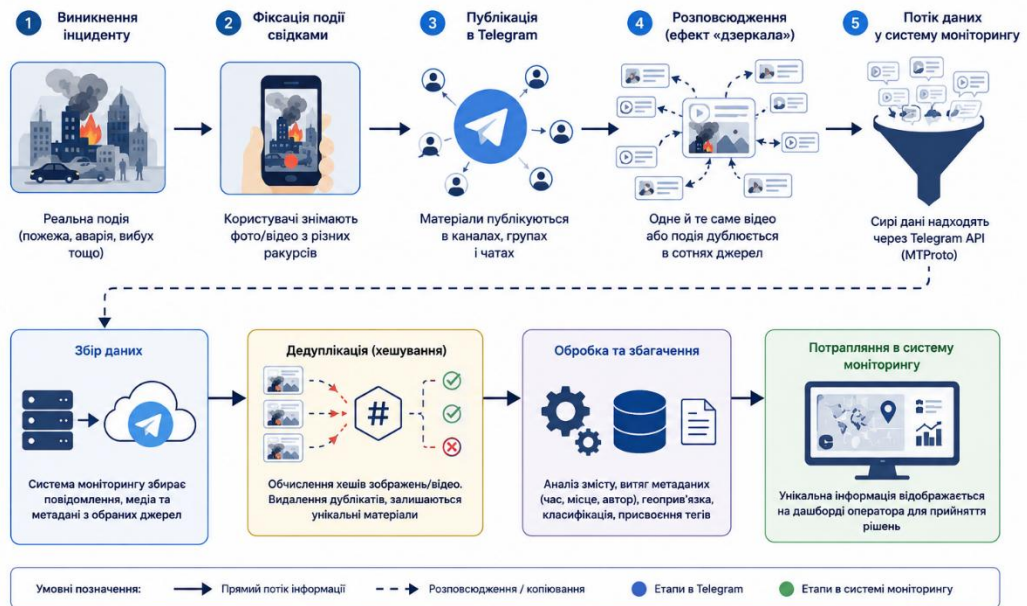


Рисунок 1.5 – Схема життєвого циклу та розповсюдження інформації в середовищі Telegram

Третім за порядком, але не за значенням, аспектом є критична технічна складність обробки гетерогенних поточкових даних у реальному часі. Оскільки сучасні бізнес-об'єкти великих корпорацій або логістичних компаній часто розкидані географічно на тисячі кілометрів, система моніторингу повинна одночасно і безперервно аналізувати сотні й тисячі локальних джерел, кожне з яких має свою інтенсивність генерації контенту. Це вимагає впровадження не лише асинхронного збору даних, про переваги якого йшлося у попередніх підрозділах, а й багаторівневої стратегії попередньої фільтрації інформації ще на рівні «сирих» пакетів даних.

Необхідність такої фільтрації зумовлена високою обчислювальною вартістю роботи нейромережевих моделей: запуск модуля штучного інтелекту для аналізу кожного отриманого файлу є економічно та технічно недоцільним. Система має автоматично відсікати завідомо нерелевантні типи медіафайлів (наприклад, стікери, короткі GIF-анімації, меми або зображення з критично низькою роздільною здатністю), які не можуть містити корисної інформації про безпекові інциденти.

Такий підхід дозволяє суттєво знизити навантаження на обчислювальні вузли, фокусуючи ресурси виключно на аналізі потенційно небезпечного контенту. Попередня селекція даних виконує роль своєрідного «сита», яке пропускає лише ті об'єкти, що відповідають технічним та змістовним критеріям системи. Це забезпечує високу пропускну здатність платформи та мінімальну затримку (latency) між моментом публікації повідомлення у Telegram та моментом його верифікації системою. Повну технологічну послідовність етапів очищення та підготовки даних до інтелектуального аналізу представлено на рис. 1.6 у вигляді багаторівневої воронки фільтрації.



Рисунок 1.6 – Технологічна воронка фільтрації даних у системі моніторингу

Таким чином, розробка спеціалізованої OSINT-системи для потреб приватного бізнесу вимагає комплексного та багаторівневого підходу, де безпосередній збір даних через API є лише початковим технічним етапом. Основна цінність проєкту полягає не стільки у накопиченні масивів інформації, скільки у створенні високоефективного інтелектуального фільтра. Такий фільтр здатний трансформувати хаотичний, високошумний потік повідомлень у структуровану, релевантну стрічку перевірених безпекових інцидентів, що піддаються автоматизованому обліку та аналізу.

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            |      |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            | 24   |

Впровадження подібних рішень дозволяє бізнесу перейти від реактивної моделі безпеки (реагування на події після їх офіційного оприлюднення) до проактивної моделі ситуаційної обізнаності в режимі реального часу. Саме необхідність подолання викликів, пов'язаних з інформаційним перевантаженням, візуальними фейками та дублюванням контенту, зумовлює технологічну еволюцію системи: перехід від простих скриптів-парсерів до складних інтелектуальних систем на основі комп'ютерного зору та глибокого навчання.

Розроблена концепція архітектури дозволяє мінімізувати вплив «людського фактора», знизити навантаження на аналітичні підрозділи та забезпечити критично важливу швидкість прийняття рішень. Детальний опис алгоритмів класифікації, архітектури нейронних мереж та програмної реалізації модуля штучного інтелекту, що складає ядро даної системи, буде розглянуто у наступному розділі кваліфікаційної роботи.

#### **1.4. Огляд існуючих систем OSINT-моніторингу та їх функціональні можливості**

Сучасний ринок прикладного програмного забезпечення для проведення розвідки на основі відкритих джерел (OSINT) на етапі 2024–2026 років характеризується надзвичайно високим ступенем галузевої сегментації. Кожне представлене на ринку рішення еволюціонувало від універсального інструмента збору даних до вузькоспеціалізованої аналітичної платформи, націленої на специфічне коло завдань: від моніторингу споживчих настроїв та маркетингового аналізу до проведення глибокої форензики та розслідування інцидентів кібербезпеки. Така диференціація зумовлена експоненціальним зростанням обсягів неструктурованих даних, що вимагає від розробників впровадження унікальних алгоритмів обробки під конкретні типи джерел.

Одним із найбільш технологічно досконалих представників сегмента візуальної аналітики є платформа YouScan, яка здійснила фундаментальний ривок у сфері розпізнавання об'єктів та інтерпретації візуального контенту. Компанія

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            | 25   |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            |      |

успішно інтегрувала концепцію «Visual Insights», що дозволяє бізнесу виходити за межі стандартного текстового моніторингу. Система використовує складні багаторівневі архітектури глибоких нейронних мереж (зокрема, згорткові нейронні мережі та трансформери для візуальних даних) для прецизійної ідентифікації корпоративних брендів, складних логотипів та типів сцен у мультимедійних публікаціях соціальних мереж та месенджерів.

Технологічна перевага таких систем полягає у здатності аналізувати контекст зображення: нейромережеві класифікатори не просто фіксують наявність об’єкта, а й оцінюють його оточення, взаємодію з іншими елементами кадру та загальну тональність візуального повідомлення. Це дозволяє автоматизовано виявляти приховані тренди, які не відображені у текстовому супроводі публікації. Проте, попри високий рівень автоматизації, подібні платформи часто залишаються замкненими у межах маркетингового функціоналу, що створює запит на розробку аналогічних за потужністю, але спеціалізованих на безпекових сценаріях рішень.

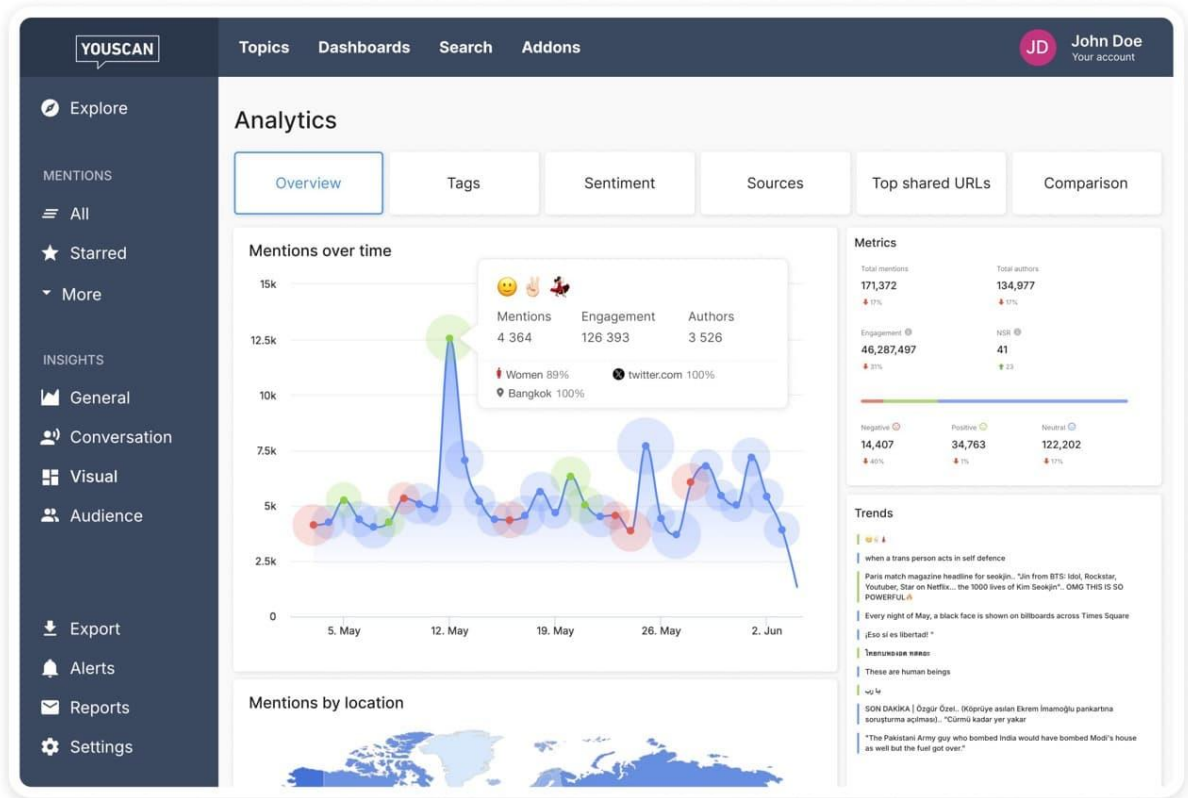


Рисунок 1.7 – Інтерфейс аналітичної панелі системи YouScan для аналізу візуального контенту

Незважаючи на високу точність візуального пошуку, продемонстровану на рис. 1.7, платформа YouScan залишається інструментом, що орієнтований переважно на запити маркетингових стратегів та PR-департаментів. Це створює суттєві обмеження при спробах інтеграції системи в контур корпоративної безпеки. Ключова проблема полягає в архітектурній спеціалізації: нейромережеві алгоритми YouScan проходять навчання (training) на датасетах, оптимізованих для розпізнавання комерційних символів, бренд-атрибутики та споживчої поведінки.

Водночас детекція специфічних ознак фізичних загроз – таких як первинні ознаки задимлення, структурні пошкодження інфраструктурних об'єктів, наявність вогнепальної зброї або підозрілих предметів – не є пріоритетною функцією системи та часто випадає з фокусу її уваги. Більше того, логіка спрацювання сповіщень у маркетингових системах налаштована на виявлення охоплень та віральності, тоді як для задач безпеки критично важливим є одиначне повідомлення від випадкового очевидця, яке може загубитися у статистичних звітах великої платформи.

Паралельно з маркетинговими інструментами на ринку активно розвиваються системи всеохоплюючого (omnichannel) моніторингу, серед яких технологічним лідером виділяється платформа SemanticForce. Ця екосистема фокусується на глибокій семантичній обробці текстової інформації, використовуючи сучасні методи природної мовної обробки (Natural Language Processing – NLP) для виявлення прихованих аномалій у поведінці користувачів та скоординованих бот-мереж.

SemanticForce демонструє надзвичайно високу ефективність у протидії дезінформаційним кампаніям, виявленні витоків конфіденційних даних та моніторингу репутаційних ризиків у месенджерах. Платформа пропонує користувачам інтелектуальні можливості фільтрації за географічними координатами, мовними особливостями та тематичними ознаками, що дозволяє локалізувати джерела інформаційного впливу. Проте, як і у випадку з іншими універсальними рішеннями, SemanticForce надає перевагу текстовому аналізу, часто залишаючи медіаконтент (фото та відео) у ролі допоміжного, а не основного джерела первинних доказів про фізичні інциденти. Це створює технологічний

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            |      |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            | 27   |

розрив між здатністю системи «читати» Telegram та її спроможністю «бачити» реальну загрозу через об'єкти камери очевидця. Відсутність глибокої інтеграції інструментів комп'ютерного зору для миттєвої класифікації візуальних аномалій призводить до того, що критичні події, зафіксовані лише на відео без супровідного опису, залишаються поза увагою автоматизованих алгоритмів аналізу.



Рисунок 1.8 – Панель моніторингу SemanticForce: фільтрація потоків даних та аналіз тональності

Проте технічна архітектура SemanticForce, інтерфейс якої продемонстровано на рис. 1.8, розроблена для горизонтального масштабування та обробки величезних масивів даних з тисяч різномірних джерел, що неминуче призводить до універсалізації алгоритмів на шкоду глибокій спеціалізації. В умовах роботи з Big Data було прийнято рішення використовувати усереднені моделі класифікації, які демонструють стабільну роботу на великих вибірках, але втрачають чутливість до специфічних локальних аномалій.

Для приватного бізнесу, який потребує точкового, прецизійного моніторингу конкретних локальних спільнот та географічних чатів навколо стратегічно важливих об'єктів, така універсальність стає суттєвим недоліком. Проблема полягає у відсутності гнучких інструментів для налаштування та донавчання (fine-tuning) кастомних нейромережевих класифікаторів під вузькоспеціалізовані типи

візуального контенту, що надходить із специфічних Telegram-джерел. Як наслідок, система може ефективно відстежувати загальні репутаційні загрози, але виявляється недостатньо адаптивною для розпізнавання унікальних технічних інцидентів, характерних для конкретної інфраструктури замовника.

Окреме місце в ієрархії сучасних OSINT-рішень займає глобальна платформа Meltwater, яка у рамках останніх оновлень інтегрувала передові технології генеративного штучного інтелекту (GenAI) для аналізу світових медіа-трендів та прогнозування репутаційних криз. Система володіє однією з найбільших баз джерел у світі, охоплюючи не лише цифрові соціальні мережі, а й традиційні друковані ЗМІ, подкасти та прямі телевізійні ефіри. Це робить Meltwater потужним інструментом для глобального маркетингу та стратегічного планування на рівні великих транснаціональних корпорацій.

Однак для оперативного управління фізичною безпекою на локальному рівні Meltwater виявляється занадто інертною системою з високим порогом входу та складною логікою налаштування. Громіздкий інтерфейс, орієнтований на формування багатосторінкових стратегічних звітів, а також тривалий цикл індексації повідомлень не дозволяють використовувати платформу як інструмент швидкого реагування для моніторингу локальних чатів та мікроспільнот.

У ситуаціях, де інформація про інцидент з'являється у перші секунди у вигляді аматорського відео чи короткого текстового повідомлення від свідка, затримки (latency) глобальних агрегаторів контенту стають критичними. Орієнтація на стратегічне спостереження замість тактичного реагування перетворює Meltwater на потужний архівний інструмент, який, проте, не може забезпечити безперервний потік «живих» даних, необхідний для роботи систем екстреного оповіщення та запобігання безпосереднім загрозам активам бізнесу.

Більше того, технічна залежність таких систем від сторонніх API-агрегаторів та методів непрямого парсингу створює додаткові рівні обробки, кожен з яких збільшує підсумкову затримку передачі даних до кінцевого користувача. У контексті забезпечення безпеки приватного бізнесу, де ефективність прямо пропорційна швидкості ідентифікації загрози, використання хмарних посередників

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            |      |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            | 29   |

стає критичною вразливістю, яка може призвести до несвоєчасного отримання інформації про активну фазу інциденту. Це зумовлює необхідність відмови від універсальних комерційних рішень на користь кастомного програмного комплексу, що базується на прямій взаємодії з протоколом Telegram (MTProto) та локальних обчислювальних потужностях для миттєвої фільтрації вхідного медіапоток.

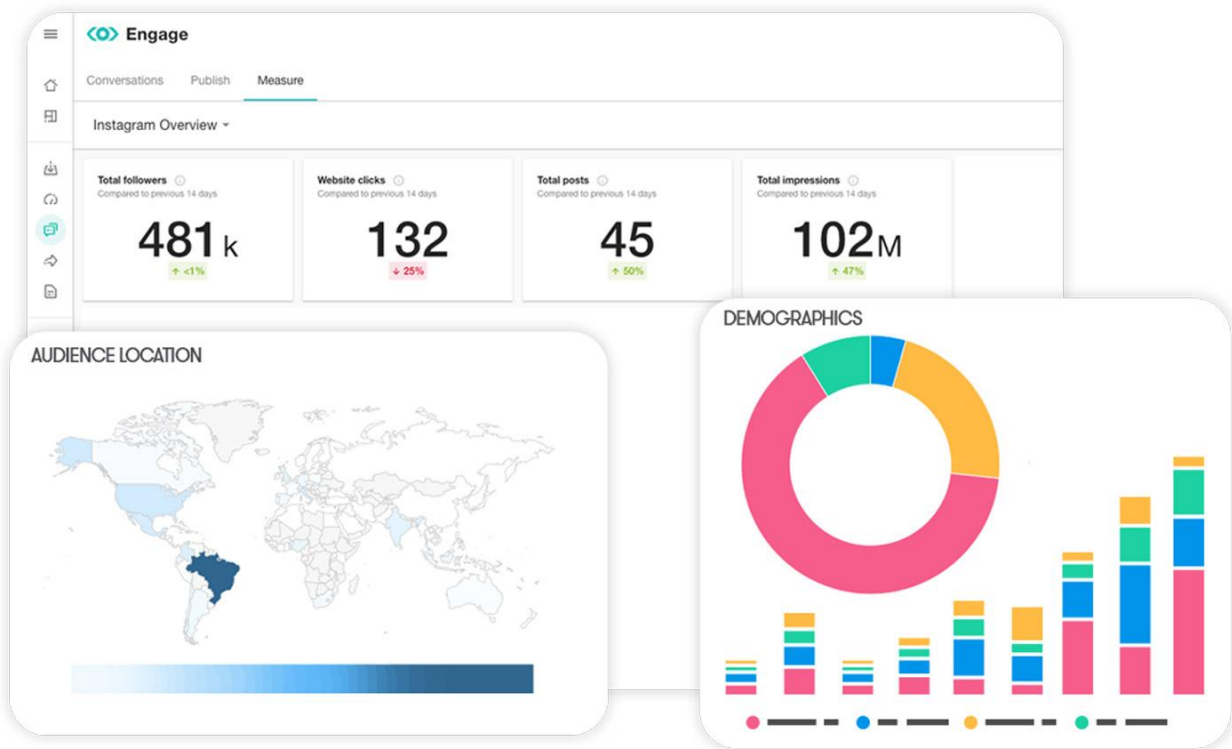


Рисунок 1.9 – Глобальна мапа згадок та аналітичний дашборд платформи Meltwater

На противагу великим аналітичним екосистемам, орієнтованим на масовий моніторинг, існують спеціалізовані інструменти глибокої технічної розвідки, серед яких ключове місце займає платформа Maltego. Цей інструмент пропонує принципово інший методологічний підхід, що ґрунтується на концепції побудови інтерактивних графів зв’язків між об’єктами (Entity-Relationship). Maltego виступає потужним засобом для розслідування кіберінцидентів, проведення форензики та деанонізації джерел інформації в Telegram через глибокий аналіз метаданих, ідентифікаторів користувачів та взаємозв’язків між публічними й приватними групами.

Однак, попри унікальні аналітичні можливості, функціонал Maltego розрахований виключно на ретроспективне дослідження вже накопичених даних, а

не на динамічний моніторинг потокового мультимедійного контенту в режимі реального часу. Процес роботи з платформою передбачає активну участь аналітика для запуску кожного окремого «трансформу» (алгоритму пошуку), що вимагає надзвичайно високої технічної кваліфікації оператора та значних часових витрат на побудову та інтерпретацію кожної гілки графа. Такі особливості роблять Maltego ідеальним інструментом для розслідування причин події після того, як вона вже відбулася, але абсолютно непридатним для побудови автономної системи автоматичного сповіщення про загрози, що виникають миттєво.

Таким чином, проведений детальний огляд існуючого програмного забезпечення дозволяє стверджувати, що сучасний ринок OSINT-систем знаходиться у стані дихотомії: платформи або занадто фокусуються на маркетингових задачах та аналізі споживчих трендів, або призначені для вузькоспеціалізованих експертних розслідувань. Внаслідок цього виникає чітко виражений технологічний вакуум – відсутність доступного та гнучкого рішення, яке б органічно поєднувало у собі високу швидкість асинхронного парсингу Telegram через пряме звернення до протоколу MTProto з можливістю інтеграції власних нейромережових моделей для інтелектуальної фільтрації медіафайлів безпосередньо у момент їх публікації.

Саме цей розрив між теоретичними можливостями OSINT та практичними потребами корпоративного сектору обґрунтовує критичну необхідність розробки індивідуального програмного комплексу. Проектне рішення, що розглядається у даній роботі, позбавлене надмірної, переобтяженої функціональності глобальних маркетингових платформ. Натомість воно пропонує бізнесу спеціалізований інструментарій, де максимальна швидкість збору даних поєднується з високою прецизійністю алгоритмів комп'ютерного зору, забезпечуючи своєчасне виявлення реальних загроз критичній інфраструктурі та матеріальним активам компанії.

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            |      |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            | 31   |

## 1.5. Юридичні обмеження та етичні норми автоматизованого збору публічних даних

Проектування та впровадження систем автоматизованого збору даних (веб-скрапінгу) у середовищі месенджера Telegram вимагає прискіпливого аналізу правового поля, оскільки межа між легальним OSINT-дослідженням та порушенням приватності є досить тонкою. Фундаментальним правовим принципом, на якому базується робота даної системи, є публічність інформації, що опрацьовується. Оскільки моніторинг здійснюється виключно у відкритих каналах та групах, до яких має доступ необмежене коло осіб, такі дані за своєю природою класифікуються як загальнодоступні (Open Source), що значно знижує ризик прямого порушення законодавства про захист персональних даних та недоторканність приватного життя [24, 25]. Проте автоматизація цього процесу вимагає врахування не лише статусу джерела, а й способів подальшого використання отриманих цифрових активів.

В контексті європейського та українського законодавства, зокрема Загального регламенту про захист даних (GDPR) та Закону України «Про захист персональних даних», ключовим аспектом стає мета обробки інформації (purpose limitation). У даному проєкті збір даних здійснюється виключно з метою забезпечення безпеки приватного бізнесу, захисту матеріальних активів та мінімізації операційних ризиків, що повністю вкладається в концепцію «законного інтересу» (legitimate interest) контролера даних. Це означає, що право на безпеку та захист власності балансується з правом на приватність, де перевага надається заходам безпеки за умови дотримання принципів невтручання в особисте життя суб'єктів даних.

Проте розробник та власник системи повинні чітко усвідомлювати, що навіть у публічних повідомленнях, які на перший погляд здаються знеособленими, можуть міститися персональні дані (Personally Identifiable Information – PII). До них належать обличчя випадкових людей на фотографіях з місця подій, державні номери автомобілів, що потрапили в кадр, або точні геолокаційні координати

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            |      |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            | 32   |

приватних адрес у тексті повідомлень. Етична експлуатація системи передбачає суворе дотримання принципу цільової спрямованості: такі дані в жодному разі не повинні використовуватися для деанонізації чи ідентифікації осіб без прямих законних на те підстав. Система має бути налаштована на фіксацію та аналіз самого факту безпекового інциденту (наприклад, загоряння на складі), ігноруючи персональні характеристики осіб, що стали мимовільними свідками події.

Крім того, юридична складність посилюється динамічністю самого середовища Telegram. Платформа дозволяє авторам контенту видаляти повідомлення «для всіх», що з точки зору права породжує «право на забуття» (right to be forgotten). Для забезпечення повної правової відповідності, архітектура системи OSINT-моніторингу має враховувати механізми синхронізації, за яких видалені першоджерелом дані автоматично або регламентовано видаляються і з локальних сховищ системи моніторингу бізнесу. Це демонструє відповідальний підхід до володіння даними та мінімізує юридичні ризики, пов'язані з несанкціонованим зберіганням приватної інформації, що була вилучена з публічного доступу.



Рисунок 1.10 – Схема забезпечення правової відповідності при зборі та обробці OSINT-даних

Важливим етичним маркером та запобіжником від порушення прав людини є принципова відмова від скрапінгу приватних профілів користувачів або спроб

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            |      |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            | 33   |

проникнення у закриті спільноти та приватні чати через використання методів соціальної інженерії. Архітектура системи базується виключно на принципі пасивного спостереження за відкритими джерелами, що не передбачає порушення технічних або логічних бар'єрів, встановлених адміністрацією месенджера або самими власниками контенту. Такий підхід не лише гарантує абсолютну юридичну чистоту отриманих результатів, а й створює фундамент довіри до системи як до інструменту професійної кіберрозвідки, що функціонує суворо у правовому полі, поважаючи цифрові права людини та недоторканність приватного листування.

Окрему увагу в межах даної роботи приділено нормам користувацької угоди (Terms of Service) платформи Telegram. На відміну від багатьох інших соціальних мереж зі жорсткою політикою обмеження доступу до даних, Telegram займає більш ліберальну та відкриту позицію щодо використання розробниками зовнішніх інтерфейсів (API). Проте сервіс суворо регламентує інтенсивність запитів і забороняє будь-які дії, що можуть бути класифіковані як спам, автоматизоване створення акаунтів-ботів або деструктивний вплив на інфраструктуру месенджера. Використання офіційних бібліотек та методів протоколу MTProto (Mobile Transport Protocol) замість методів імітації дій реального користувача (browser automation) дозволяє системі залишатися в межах технічного регламенту платформи. Це не лише забезпечує високу стабільність роботи комплексу при пікових навантаженнях, а й підтверджує легітимність методів збору даних з точки зору внутрішньої політики безпеки сервісу, що детально відображено на схемі відповідності (див. рис. 1.10).

Іншим критично важливим аспектом є дотримання норм авторського права на медіаконтент, що розповсюджується в Telegram-спільнотах. Згідно з усталеними нормами міжнародного та національного цифрового права, використання фото- та відеоматеріалів для внутрішнього аналізу, ідентифікації загроз та навчання моделей штучного інтелекту без їх подальшого комерційного перепродажу або публічного відтворення розглядається як «сумлінне використання» (fair use). У даному контексті контент виступає не як об'єкт естетичної цінності, а як джерело технічних метаданих та візуальних ознак інцидентів.

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            |      |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            | 34   |

Зокрема, правовий статус таких операцій базується на тому, що автоматизована система не здійснює плагіат чи привласнення авторства, а трансформує вхідний візуальний потік у математичні вектори та аналітичні звіти. Таке перетворення даних (transformative use) дозволяє витягувати корисну інформацію про безпекові ризики, не порушуючи при цьому майнових прав правовласників на оригінальні твори. Крім того, архітектурне обмеження доступу до бази накопичених медіафайлів виключно колом профільних фахівців компанії гарантує дотримання конфіденційності та запобігає несанкціонованому тиражуванню чужого контенту.

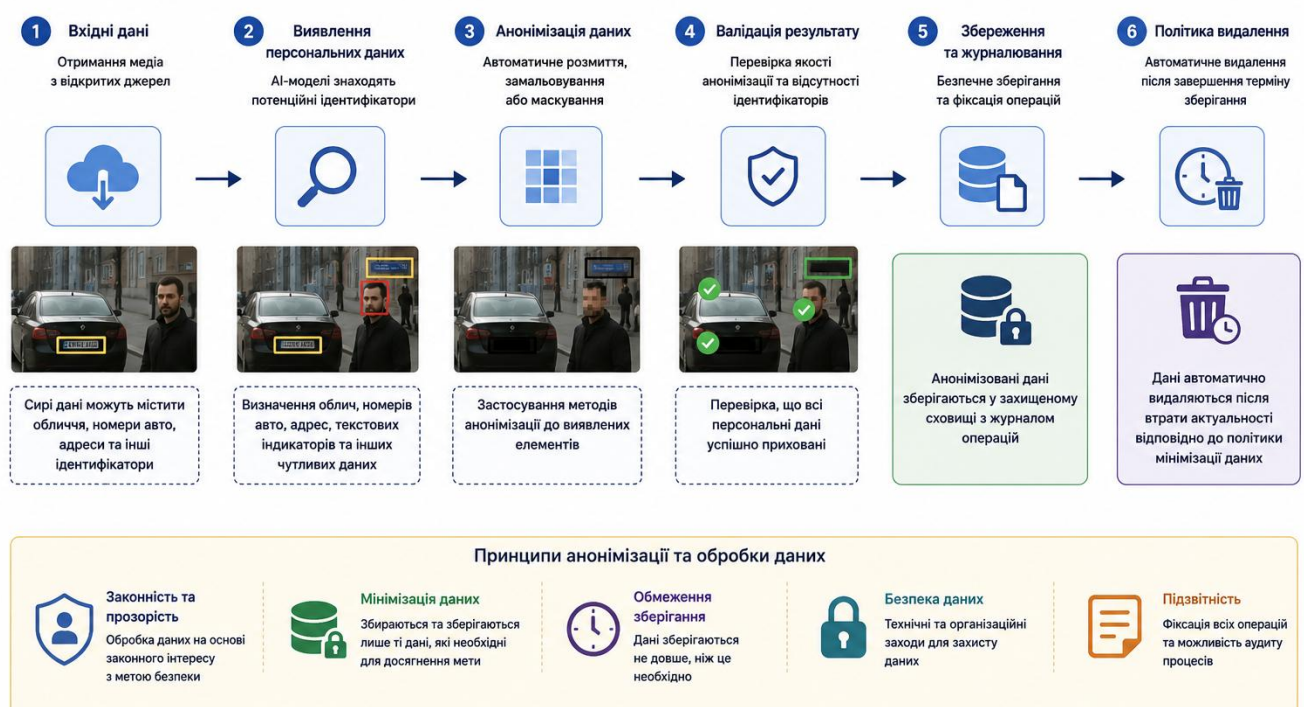


Рисунок 1.11 – Процес автоматизованої анонімізації персональних даних у системі моніторингу

Етична складова проєкту також включає персональну та корпоративну відповідальність за недопущення використання зібраних масивів даних для політичних маніпуляцій, соціального інжинірингу або створення систем несанкціонованого масового стеження. Важливим технічним рішенням у цьому контексті є впровадження алгоритмів превентивної анонімізації безпосередньо на етапі первинної обробки медіапотоку. Приклади роботи таких механізмів, наведені на рис. 1.11, демонструють здатність системи автоматично ідентифікувати та

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            |      |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            | 35   |

маскувати (через накладання гаусового розмиття або пікселізацію) обличчя випадкових перехожих та біометричні дані осіб на кадрах з місця подій. Це дозволяє фокусувати аналітичну увагу нейронної мережі виключно на семантично важливих об'єктах загрози – осередках відкритого вогню, масштабах задимлення, техногенних руйнуваннях або наявності специфічної техніки.

Такий підхід, що базується на принципах «Privacy by Design», не лише суттєво підвищує рівень юридичного та репутаційного захисту власника системи, а й формує суспільну та корпоративну довіру до OSINT-технологій як до інструментів гуманітарної та комерційної безпеки, а не засобів тотального цифрового нагляду. Більше того, анонімізація нерелевантних деталей дозволяє зменшити когнітивне навантаження на оператора системи, акцентуючи його увагу лише на детермінованих факторах ризику.

Таким чином, суворе дотримання юридичних регламентів та етичних норм не лише не виступає перешкодою для технічної розробки, а навпаки – є необхідним архітектурним фундаментом для побудови легітимного та професійного інструменту бізнес-розвідки. Інтеграція правових обмежень безпосередньо в програмну логіку комплексу дозволяє автоматизувати не тільки процеси збору та класифікації, а й забезпечити комерційну, юридичну та кібернетичну безпеку експлуатації отриманих даних протягом усього їхнього життєвого циклу.

Впровадження механізмів контролю цілісності даних та суворе розмежування прав доступу до аналітичних звітів гарантує, що конфіденційна інформація про інциденти не вийде за межі захищеного контуру компанії. Це логічно завершує комплексне теоретичне обґрунтування проєкту, підтверджуючи його актуальність у правовому та соціальному вимірах, та дозволяє перейти до етапу практичної реалізації системи. У наступних розділах буде представлено детальний опис програмної архітектури, вибір математичного апарату та результати тестування розробленого комплексу, що відповідає всім вищезазначеним критеріям технічної ефективності, етичної прийнятності та юридичної легітимності.

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            |      |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            | 36   |

## Висновки до першого розділу

У межах першого розділу було здійснено комплексний аналіз предметної області, досліджено специфіку месенджера Telegram як джерела OSINT-даних та обґрунтовано необхідність автоматизації процесів моніторингу для потреб приватного бізнесу. За результатами проведеного дослідження можна зробити такі висновки:

Аналіз Telegram як середовища передачі даних показав, що поєднання високої швидкості розповсюдження інформації та ліберальної політики API робить цю платформу пріоритетною для систем раннього сповіщення про загрози. Проте виявлено, що масштабування збору даних до рівня понад 1000 джерел вимагає суворого дотримання лімітів платформи (Flood Limits). Встановлено, що класична послідовна модель збору є неефективною для систем екстреного реагування, що зумовило вибір асинхронної архітектури на базі протоколу MTProto для забезпечення безперервності інформаційного потоку.

Дослідження факторів інформаційного шуму дозволило класифікувати основні перешкоди у промисловому зборі даних, серед яких найбільш деструктивними є агресивна монетизація каналів, віральне дублювання контенту («дзеркальний ефект») та поширення архівних медіафайлів під виглядом актуальних інцидентів. Виявлено, що рівень нерелевантної інформації в публічних чатах сягає 95–98%, що робить використання традиційних текстових фільтрів (на основі ключових слів) недостатнім.

Порівняльний аналіз існуючих OSINT-систем (зокрема YouScan, SemanticForce та Meltwater) продемонстрував, що більшість комерційних рішень орієнтовані на маркетингові задачі або захист бренду, що зумовлює високу латентність при обробці критичних інцидентів. Основні недоліки існуючих платформ – висока вартість ліцензій, відсутність кастомних нейромережових моделей для детекції специфічних фізичних загроз та хмарна архітектурах.

Аналіз юридичних та етичних аспектів підтвердив легітимність обраного підходу за умови дотримання принципів «Privacy by Design» та «Fair Use».

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            |      |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            | 37   |

Обґрунтовано, що збір даних із відкритих джерел для забезпечення безпеки активів відповідає критеріям «законного інтересу» згідно з нормами GDPR та законодавства України. Визначено, що для мінімізації ризиків порушення приватності система повинна включати механізми автоматичної анонімізації персональних даних та жорсткі регламенти видалення неактуальної інформації.

Отримані висновки стають теоретичним фундаментом для подальшого проєктування архітектури системи, вибору математичних моделей глибокого навчання та програмної реалізації комплексу, що будуть детально розглянуті у наступних розділах роботи.

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            |      |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            | 38   |

## РОЗДІЛ 2. МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ СИСТЕМИ ІНТЕЛЕКТУАЛЬНОГО МОНІТОРИНГУ

### 2.1. Формування технічних вимог до системи (масштабованість, швидкість обробки, точність фільтрації)

Проектування інтелектуальної системи OSINT-моніторингу для потреб приватного сектору вимагає фундаментального перетворення бізнес-завдань у площину конкретних інженерних метрик та суворих програмних стандартів. На відміну від універсальних комерційних рішень, розроблюваний комплекс має забезпечувати не просто пасивний збір даних, а високошвидкісну багаторівневу фільтрацію потокового медіаконтенту в режимі реального часу. Це зумовлює особливий підхід до визначення нефункціональних вимог, де на перший план виходять показники латентності (latency), пропускну здатності каналів зв'язку та відмовостійкості компонентів при експоненціальному зростанні обсягів вхідного трафіку.

Ключовим пріоритетом архітектури виступає масштабованість, яка в даному проєкті реалізується через використання спеціалізованих облікових записів зі статусом Telegram Premium. Дане інженерне рішення дозволяє на програмному рівні зняти критичні базові ліміти платформи на кількість активних підписок, розширюючи моніторинговий потенціал системи до 1000 джерел одночасно. Окрім кількісного розширення джерел, Premium-статус забезпечує пріоритетну смугу пропускання (bandwidth) при завантаженні «важких» мультимедійних файлів (відео високої роздільної здатності), що є визначальним фактором для швидкодії всієї системи в умовах оперативного моніторингу.

Для підтримки такої високої інтенсивності вхідних даних архітектура базується на принципах асинхронного програмування за допомогою бібліотеки asyncio. Основна логіка системи передбачає повну ізоляцію модуля збору «сирих» даних (реалізованого в компоненті client.py на базі протоколу MTPROTO) від модуля інтелектуального нейромережевого аналізу (bot\_filter.py). Ця ізоляція досягається

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            |      |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            | 39   |

через впровадження механізму внутрішніх асинхронних черг (asyncio.Queue), що виконують роль програмного буфера. Така схема гарантує, що ресурсомісткі операції інференсу нейромережі CLIP не призведуть до блокування основного потоку прослуховування подій Telegram, запобігаючи втраті критично важливих повідомлень або розриву сесії через перевантаження.

Логіку побудови такої відмовостійкої системи, шлях проходження сигналу від первинного джерела даних до кінцевого інтерфейсу оператора безпеки, а також механізм взаємодії між незалежними компонентами через міст (bridge.py), детально представлено на рисунку 2.1. Такий підхід дозволяє системі гнучко адаптуватися до змін інтенсивності інформаційних потоків, забезпечуючи стабільну роботу навіть у моменти пікової активності в моніторингових спільнотах.

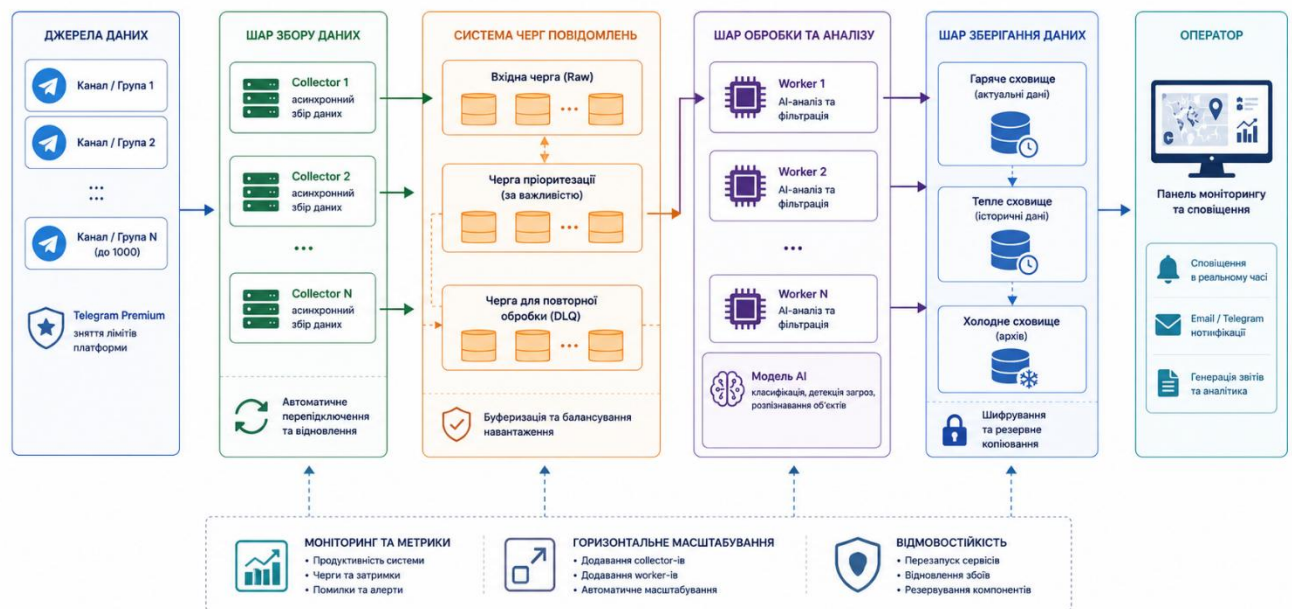


Рисунок 2.1 – Схема архітектури потокової обробки даних та розподілу навантаження

Особлива технологічна складність та високе обчислювальне навантаження виникають при роботі з поточним відеоконтентом, оскільки класичний метод повного покадрового аналізу (frame-by-frame analysis) є критично ресурсозатратним і непридатним для систем моніторингу реального часу. Враховуючи високу роздільну здатність сучасних відеоматеріалів та їхню

тривалість, технічна вимога до модуля відеообробки передбачає впровадження оптимізованого алгоритму інтелектуального семплінгу.

У межах розробленого програмного комплексу реалізовано механізм динамічного вилучення рівно 40 опорних кадрів (keyframes) з кожного вхідного відеофайлу, які рівномірно розподіляються за всією часовою шкалою відеоряду. Такий підхід дозволяє зберегти високу репрезентативність вибірки, охоплюючи як початок, так і фінальні стадії зафіксованого інциденту, водночас радикально знижуючи навантаження на графічний процесор (GPU) за рахунок ігнорування надлишкових кадрів, що не несуть нової семантичної інформації.

Кожен із вилучених опорних кадрів піддається індивідуальному семантичному аналізу мультимодальною нейромережею CLIP (Contrastive Language-Image Pre-training). Процес аналізу базується на обчисленні косинусної схожості між вектором зображення та текстовими ембеддингами (промптами), що описують потенційні загрози. Для забезпечення надійності системи та виключення хибнопозитивних спрацювань, викликаних випадковими візуальними артефактами або шумами, підсумкова верифікація безпекового інциденту відбувається на основі мажоритарної логіки прийняття рішень.

Це означає, що система генерує тривожний алерт лише у тому випадку, якщо цільовий об'єкт або ознаки надзвичайної ситуації були одночасно зафіксовані на декількох незалежних кадрах з рівнем впевненості (confidence score), що перевищує встановлений поріг. Дана багатоетапна перевірка дозволяє ефективно відфільтровувати випадкові об'єкти, що потрапили в кадр на долі секунди, забезпечуючи високу достовірність моніторингу. На рисунку 2.2 продемонстровано деталізований технологічний процес декомпозиції вхідного відеопотоку на окремі фрейми, їхню векторну трансформацію та подальшу багаторівневу класифікацію за допомогою навченої моделі, що дозволяє візуалізувати логіку роботи інтелектуального фільтра. Завдяки впровадженню такої мажоритарної схеми прийняття рішень досягається суттєве зниження рівня «інформаційного шуму», що дозволяє службі безпеки зосередити увагу виключно

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            |      |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            | 41   |

на верифікованих інцидентах, мінімізуючи ризик ігнорування реальної загрози через втому від хибних спрацювань.

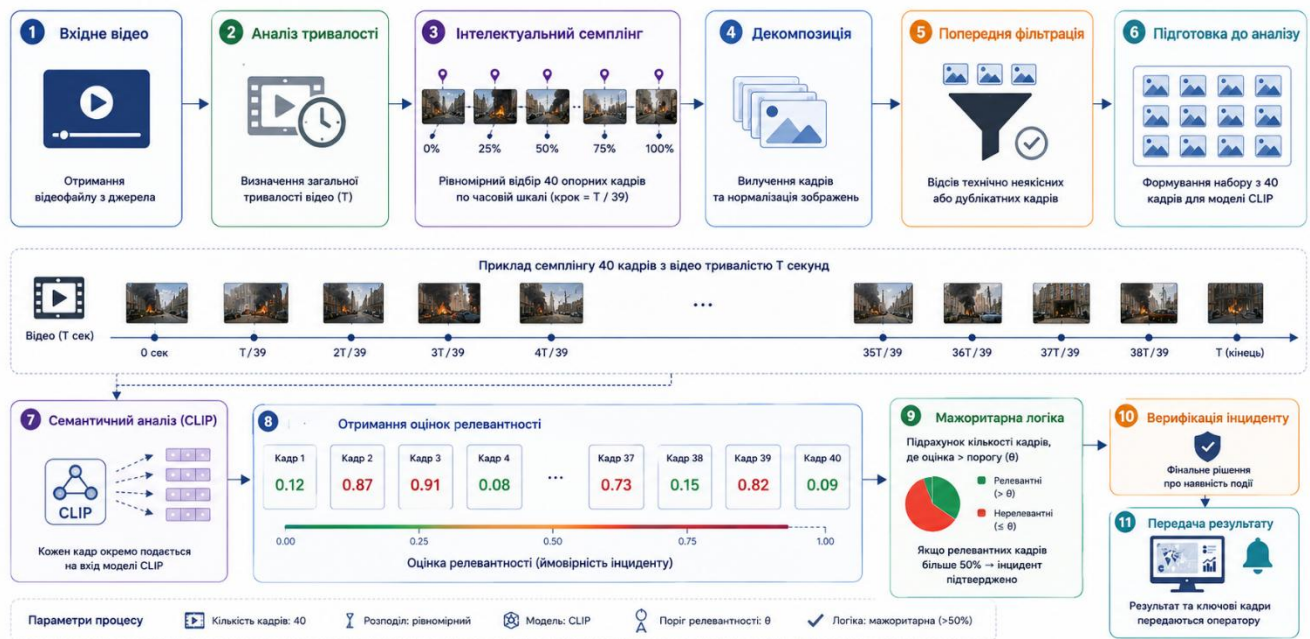


Рисунок 2.2 – Технологічний процес декомпозиції відео на опорні кадри для семантичного аналізу

Представлений на рисунку 2.2 технологічний цикл декомпозиції мультимедійного потоку є ключовим інструментом оптимізації обчислювальних ресурсів, оскільки дозволяє трансформувати неструктуроване відео у набір дискретних семантичних одиниць, придатних для миттєвого аналізу нейромережею. Такий підхід до препроцесингу даних безпосередньо впливає на загальну архітектурну ефективність програмного комплексу, створюючи передумови для стабільної обробки великої кількості одночасних запитів без деградації часу відгуку. Узгодження описаної логіки інтелектуального семплінгу з інженерними параметрами масштабованості та швидкодії вимагає чіткої фіксації граничних показників роботи системи. Відповідно, для забезпечення безперебійної експлуатації в умовах корпоративного моніторингу, було сформовано перелік базових технічних вимог до продуктивності та пропускну здатності, які систематизовано та деталізовано у таблиці 2.1. Встановлення цих нормативів дозволяє не лише забезпечити прогнозовану роботу системи при пікових навантаженнях, а й сформулювати об'єктивні критерії для вибору апаратних

потужностей, необхідних для безперебійного функціонування нейромережових алгоритмів у реальному часі.

**Таблиця 2.1**

**Технічні вимоги до продуктивності та масштабованості**

| Параметр вимоги                | Цільове значення     | Технічне обґрунтування                                 |
|--------------------------------|----------------------|--------------------------------------------------------|
| Кількість контрольованих чатів | до 1000 джерел       | Використання Telegram Premium для розширення лімітів   |
| Тип архітектури зв'язку        | Асинхронна (MTProto) | Використання бібліотеки Telethon для прямого доступу   |
| Метод передачі даних           | asyncio.Queue        | Розподіл ресурсів між збором та AI-фільтрацією         |
| End-to-End Latency             | < 30 секунд          | Максимально допустимий час для оперативного реагування |

Якість інтелектуальної фільтрації вхідного контенту безпосередньо залежить від архітектурних особливостей та прецизійності обраної моделі машинного зору, тому фундаментальною технічною вимогою до системи є використання мультимодальної архітектури CLIP (Contrastive Language-Image Pre-training). На відміну від класичних класифікаторів, дана модель здатна встановлювати глибокі семантичні зв'язки між візуальними образами та текстовим описом у єдиному векторному просторі, що робить її ідеальним інструментом для ідентифікації складних сценаріїв загроз.

Для досягнення максимальної адаптивності алгоритмів під специфічні потреби приватного бізнесу було реалізовано двохетапну стратегію донавчання (Fine-tuning). На першому етапі використовувався глобальний датасет, запозичений з відкритих профільних джерел (зокрема такий, як UAV dataset), що дозволило моделі засвоїти загальні візуальні ознаки цільових об'єктів на великій вибірці даних. Другий, критично важливий етап, полягав у навчанні на власному особистому датасеті, сформованому вручну з понад 900 унікальних медіаодиниць.

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            |      |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            | 43   |

Цей кастомний набір даних включав специфічні ракурси, умови освітлення та контекстуальні особливості, характерні для реальних об'єктів промислової та комерційної інфраструктури, які часто ігноруються загальними базами даних.

Процес навчання, що сумарно охоплював 150 епох, був спрямований на стабілізацію градієнтного спуску та точне коригування внутрішніх ваг нейронної мережі для досягнення максимальної збіжності на обох типах даних. Таке поєднання глобальної бази знань із локальною специфікою дозволило суттєво мінімізувати кількість хибнопозитивних спрацювань (False Positives), спричинених візуально схожими побутовими об'єктами. Дана прецизійна настройка алгоритмів є ключовою вимогою для запобігання інформаційному перевантаженню служби безпеки підприємства, оскільки забезпечує автоматизовану селекцію лише тих аномалій, що мають високий ступінь кореляції з реальною загрозою активам бізнесу.

Таблиця 2.2

Вимоги до точності та параметрів навчання нейромережевого модуля

| Метрика / Параметр      | Значення    | Функціональне призначення                          |
|-------------------------|-------------|----------------------------------------------------|
| Архітектура моделі      | CLIP + YOLO | Мультимодальний аналіз зображень та тексту         |
| Кількість епох навчання | 150         | Досягнення стабільної збіжності ваг моделі         |
| Recall (Повнота)        | > 90.4%     | Гарантія фіксації всіх реальних інцидентів безпеки |
| Precision (Точність)    | > 90.3%     | Зниження рівня «шуму» для оператора моніторингу    |

Додатковою критичною вимогою до функціональності системи є впровадження високоефективних механізмів дедуплікації вхідного контенту на основі алгоритмів перцептивного хешування (Perceptual Hashing). Враховуючи надзвичайно високу віральність інформації в месенджері Telegram, де один і той самий медіафайл може бути репостнутий сотнями різних джерел протягом кількох

хвилин, система повинна миттєво ідентифікувати повторні публікації ще на етапі препроцесингу. На відміну від криптографічних хеш-функцій (таких як MD5 або SHA-256), які змінюють значення при зміні навіть одного пікселя, перцептивне хешування дозволяє розпізнавати візуально ідентичні зображення та відео, навіть якщо вони пройшли незначне стиснення, зміну роздільної здатності або додавання водяних знаків.

Впровадження даного технологічного рішення дозволяє економити до 80% сукупних обчислювальних ресурсів серверної частини за рахунок повної відмови від ітераційного повторного інференсу вже відомих нейромережі файлів. Це не лише знижує апаратні витрати на утримання системи, а й суттєво підвищує загальну пропускну здатність каналу обробки. Окрім технічної економії, дедуплікація відіграє ключову роль у забезпеченні якості аналітичних даних: оператор безпеки отримує єдине верифіковане повідомлення про інцидент замість хаотичного дублювання однієї й тієї самої події з різних джерел.

У підсумку, сукупність усіх сформованих та деталізованих у даному підрозділі вимог дозволяє створити стійкий, архітектурно збалансований програмний комплекс. Інтеграція високої масштабованості Telegram Premium, інтелектуального семплінгу відео, прецизійного двохетапного навчання моделі CLIP та механізмів перцептивної фільтрації створює синергетичний ефект. Це дає змогу ефективно трансформувати хаотичний, надлишковий та зашумлений потік медіаданих у чітко структуровану стрічку перевірених інцидентів, що функціонує в режимі реального часу та забезпечує бізнес-активи надійним інструментом превентивного моніторингу загроз.

## 2.2. Проектування архітектури системи: взаємодія модулів збору, черг повідомлень та блоку аналітики

Проектування архітектури інтелектуальної системи OSINT-моніторингу базується на принципах модульності та суворого розподілу зон відповідальності між окремими програмними компонентами. Вибір технологічного стека

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            |      |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            | 45   |

обумовлений необхідністю забезпечення високої продуктивності при обробці інтенсивних мережових запитів та наявністю потужних інструментів для інтеграції моделей глибокого навчання. Основною мовою програмування було обрано Python, що є галузевим стандартом у сфері аналізу даних та штучного інтелекту завдяки нативності бібліотек для машинного зору та розвиненим механізмам асинхронного керування подіями. Для забезпечення цілісності та відмовостійкості програмний комплекс було спроектовано як багаторівневу екосистему, де кожен етап – від отримання «сирого» байт-коду повідомлення до видачі верифікованого алерту – виконується ізольованим модулем.

Центральним елементом початкового рівня системи є модуль збору даних, реалізований у файлі client.py. Його функціонування ґрунтується на використанні бібліотеки Telethon, яка працює безпосередньо з протоколом MTProto. Такий вибір є стратегічно важливим для OSINT-задач, оскільки на відміну від стандартного Bot API, клієнт на базі Telethon дозволяє імітувати поведінку реального користувача (UserBot), що є необхідною умовою для доступу до сотень локальних груп та закритих чатів. Програмна логіка цього модуля налаштована на постійне «прослуховування» вхідного потоку подій у режимі реального часу. При виявленні нового медіаконтенту модуль автоматично ініціює процедуру завантаження файлу у буферне сховище, паралельно вилучаючи критично важливі метадані: унікальні ідентифікатори джерела, параметри авторства та внутрішні маркери медіа-груп (grouped\_id), що дозволяє зберегти контекстуальну цілісність інформації.

Однак, враховуючи високу інтенсивність Telegram-трафіку, пряма передача даних від клієнта до нейромережевого модуля могла б призвести до критичних затримок або блокування потоку збору. Для вирішення цієї проблеми було спроектовано транспортний шар, представлений модулем bridge.py. Його головним завданням є організація асинхронної черги повідомлень на базі об'єкта asyncio.Queue. Цей компонент виконує роль інтелектуального посередника (middleware), реалізуючи архітектурний патерн «Producer-Consumer». Модуль збору виступає в ролі виробника даних, який миттєво поміщає посилання на завантажений контент у чергу та повертається до моніторингу ефіру, тоді як модуль

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            |      |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            | 46   |

аналізу виступає споживачем, що вилучає завдання з черги по мірі вивільнення обчислювальних ресурсів. Така схема дозволяє ефективно згладжувати пікові навантаження, що виникають під час резонансних подій, коли кількість повідомлень у чатах зростає в десятки разів. Динаміку стабілізації системи та заповнення транспортної черги в умовах різкого зростання трафіку продемонстровано на рисунку 2.3.

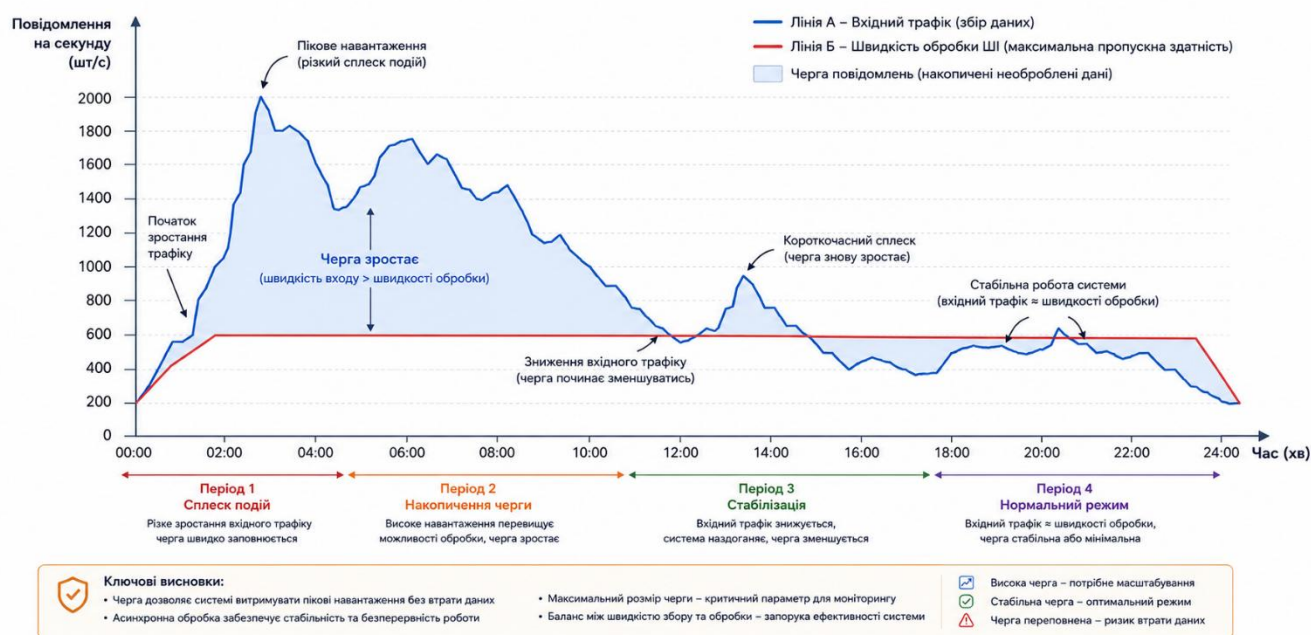


Рисунок 2.3 – Аналіз пропускну здатності транспортного шару при пікових інформаційних навантаженнях

Завершальний та найбільш складний етап обробки, що вимагає значних обчислювальних потужностей, зосереджений у модулі інтелектуальної фільтрації `bot_filter.py`. Цей компонент виконує роль головного аналітичного вузла системи, поєднуючи в собі високорівневу інтерфейсну логіку на базі фреймворку `Aiogram` та складні обчислювальні алгоритми машинного зору на базі бібліотеки `PyTorch`. Процес роботи даного модуля ініціюється автоматично при вилученні нового завдання з асинхронної черги, після чого запускається багатоетапний конвеєр аналізу, спроектований за принципом послідовної фільтрації даних від простих ознак до складних семантичних концептів.

Динаміку взаємодії між модулем аналізу та іншими компонентами системи у часовій послідовності відображено на рисунку 2.4. Дана діаграма ілюструє, як

об'єкт, вилучений з асинхронного мосту, проходить через етапи верифікації в базі даних та нейромережевого інференсу, що дозволяє візуально простежити шлях сигналу від моменту його детекції до формування фінального сповіщення.

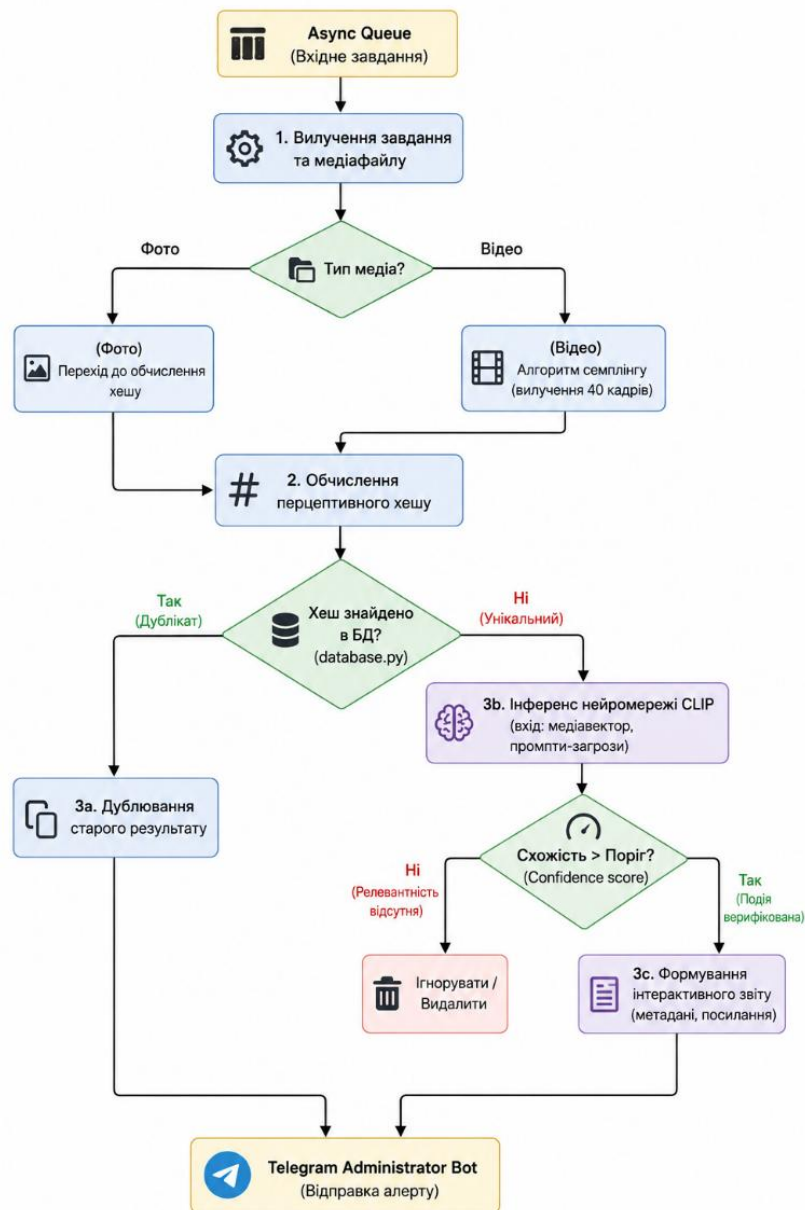


Рисунок 2.4 – Алгоритмічна схема конвеєра інтелектуальної обробки та фільтрації медіаданих у модулі bot\_filter.py

Першочерговим етапом обробки є процедура верифікації унікальності контенту через механізм перцептивного хешування. На відміну від стандартних методів порівняння файлів, цей підхід дозволяє ідентифікувати дублікати навіть після їхньої цифрової модифікації (зміна розміру, накладання тексту чи стиснення). Модуль взаємодіє з базою даних у database.py, виконуючи миттєвий пошук за

сигнатурою файлу. Якщо ідентичний або візуально схожий об'єкт уже проходив аналіз раніше, система автоматично дублює попередній результат, що дозволяє критично знизити навантаження на графічний процесор (GPU) та уникнути повторної відправки однакових сповіщень оператору.

У випадку, якщо файл визначено як унікальний, він передається в ядро інтелектуального аналізу – нейромережу CLIP (Contrastive Language-Image Pre-training). Тут відбувається складна процедура обчислення семантичної схожості: візуальні ознаки об'єкта трансформуються у багатовимірний вектор (ембеддинг), який порівнюється з наперед заданими текстовими векторами, що описують критерії загроз (наприклад, пожежа, руйнування, техніка). Для відеофайлів цей процес масштабується через логіку інтелектуальної декомпозиції на 40 репрезентативних кадрів. Такий метод дозволяє системі не просто фіксувати статичну картинку, а проводити динамічний аналіз усієї тривалості ролика, приймаючи зважене рішення на основі мажоритарного голосування результатів кожного окремого кадру.

Підсумкова логіка модуля `bot_filter.py` завершується формуванням інтерактивного звіту для адміністратора. Система не лише фільтрує контент, а й додає до нього метадані аналізу: показник впевненості моделі (confidence score), посилання на першоджерело та часові мітки. Це перетворює сирий потік повідомлень на структурований інструмент прийняття рішень, де кожне сповіщення є результатом глибокої нейромережевої перевірки, що мінімізує вплив людського фактора та забезпечує безперервність моніторингу безпеки у реальному часі.

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            | 49   |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            |      |

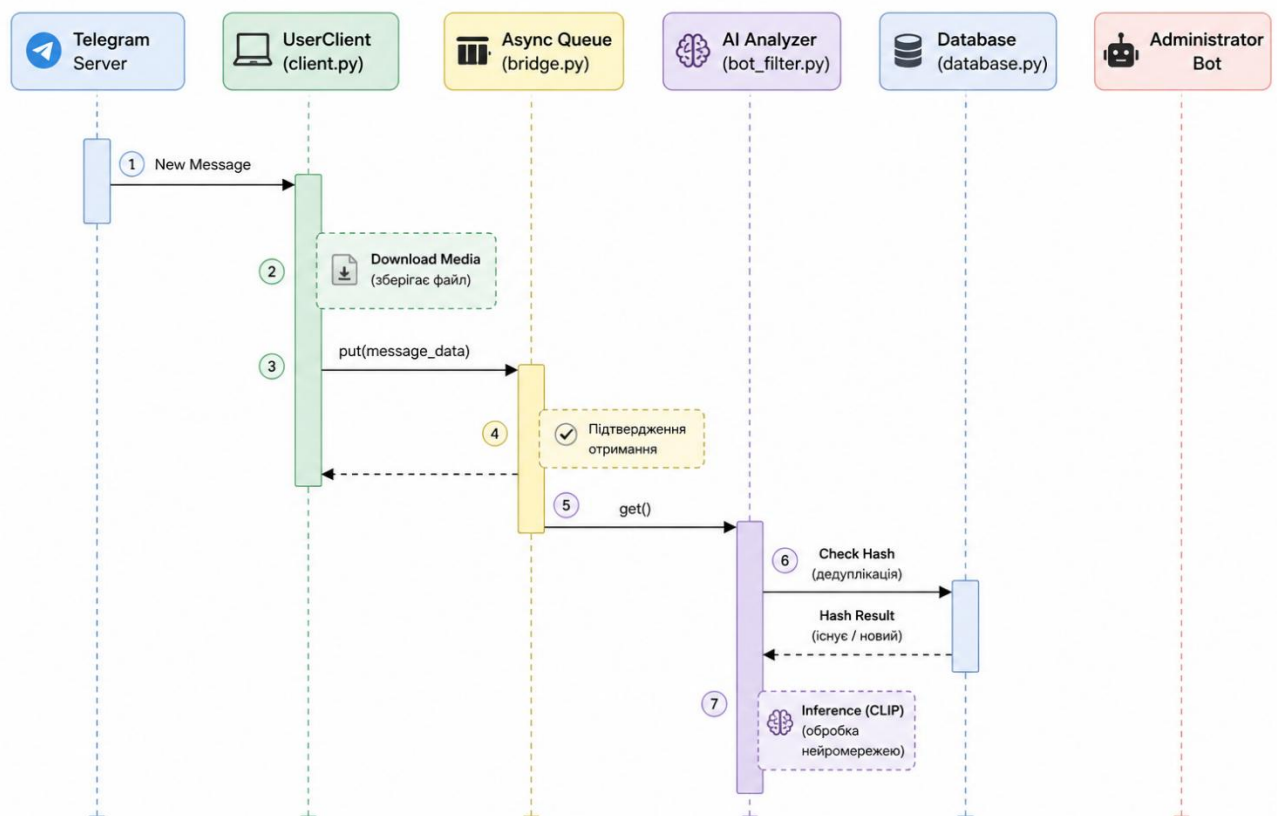


Рисунок 2.5 – Діаграма послідовності взаємодії компонентів системи при обробці інциденту

Проектування взаємодії модулів у такий спосіб забезпечує критично високий рівень автономності кожного компонента, що перетворює систему на відмовостійку інфраструктуру. Наприклад, у разі планового оновлення ваг нейромережі, модифікації бази знань ШІ або вимушеного перезавантаження обчислювального модуля аналізу, рівень збору даних (client.py) продовжуватиме функціонувати в повному обсязі. Завдяки використанню незалежних асинхронних процесів, клієнтська частина не блокується очікуванням відповіді від аналітичного блоку, а продовжує безперервно перехоплювати події з Telegram-спільнот, накопичуючи інформацію в транспортному буфері bridge.py.

Додатковою перевагою такої архітектури є здатність системи до самовідновлення без втрати оперативного контексту. Щойно модуль bot\_filter.py повертається у робочий стан, він миттєво починає вилучати накопичені в черзі повідомлення з того моменту, на якому стався перерив, забезпечуючи послідовну та повну обробку всього пропущеного трафіку. Така система здатна функціонувати

в режимі 24/7 з мінімальним залученням людського ресурсу, самостійно виконуючи рутинну роботу з первинної селекції та верифікації загроз. Підсумкові порівняльні характеристики та функціональні ролі кожного модуля, що забезпечують таку синергію, детально систематизовано та відображено в таблиці 2.3.

Таблиця 2.3

Технологічна специфікація та функціональні обов'язки модулів системи

| Назва компонента                | Роль у системі               | Використовувані технології | Тип обчислювального навантаження |
|---------------------------------|------------------------------|----------------------------|----------------------------------|
| Client Ingestion (client.py)    | Парсинг Telegram-потоків     | Telethon, MTProto          | Мережеве (I/O Bound)             |
| Async Bridge (bridge.py)        | Буферизація та транспорт     | asyncio, Shared Queues     | Мінімальне (Memory Bound)        |
| AI Intelligence (bot_filter.py) | Семантичний аналіз           | CLIP, PyTorch, Aiogram     | Обчислювальне (GPU Bound)        |
| Data Storage (database.py)      | Дедуплікація та налаштування | SQLite, SQLAlchemy         | Дискове (Storage Bound)          |

Таким чином, обрана багатоланкова архітектура дозволяє досягти оптимального балансу між швидкістю реакції системи на критичні події та глибиною когнітивного аналізу вхідного медіаконтенту. Завдяки чіткому розділенню рівнів збору, транспортування та інтелектуальної обробки, вдалося нівелювати ризики втрати даних під час пікових навантажень та забезпечити високу автономність кожного програмного модуля. Впровадження асинхронних протоколів взаємодії та механізмів буферизації гарантує, що система залишатиметься стабільною навіть у складних умовах непередбачуваної активності месенджера Telegram.

Всебічне обґрунтування технологічного стека та деталізація взаємодії компонентів створюють надійний і масштабований технологічний фундамент для подальшої практичної реалізації проєкту. Це дозволяє перейти від теоретичного моделювання процесів до безпосереднього проєктування допоміжних систем,

зокрема структури зберігання даних, яка має забезпечити швидкий доступ до історії інцидентів та ефективну роботу алгоритмів дедуплікації. Описана архітектурна модель є основою для розгортання OSINT-комплексу, що відповідає сучасним вимогам кібербезпеки та оперативного моніторингу в інтересах приватного сектору.

### 2.3. Вибір та обґрунтування технологічного стеку (Python, асинхронні бібліотеки Telethon/Pyrogram, СУБД)

Ефективність функціонування інтелектуальної системи OSINT-моніторингу безпосередньо залежить від обґрунтованого вибору інструментарію розробки. Технологічний стек повинен не лише відповідати вимогам продуктивності, а й забезпечувати гнучкість при інтеграції моделей машинного навчання та обробці мережевих потоків даних у реальному часі.

Фундаментом системи було обрано мову програмування Python. Незважаючи на існування мов із вищою швидкістю виконання (наприклад, C++ або Java), Python має низку стратегічних переваг для даного проєкту. По-перше, це найпотужніша екосистема бібліотек для штучного інтелекту та комп'ютерного зору (PyTorch, OpenCV, Transformers). По-друге, підтримка асинхронного програмування через модуль `asuncіo` дозволяє ефективно вирішувати задачі, пов'язані з мережевим введенням-виведенням (I/O bound), що є ключовим при одночасному моніторингу сотень Telegram-каналів.

При виборі інструменту для взаємодії з месенджером Telegram було проведено порівняльний аналіз трьох основних підходів (Таблиця 2.4).

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            |      |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            | 52   |

Таблиця 2.4

## Порівняльний аналіз засобів взаємодії з Telegram API

| Критерій порівняння | Telegram Bot API       | Pyrogram (MTProto)          | Telethon (MTProto)           |
|---------------------|------------------------|-----------------------------|------------------------------|
| Механізм доступу    | Через токен бота       | Сесія користувача (UserBot) | Сесія користувача (UserBot)  |
| Доступ до контенту  | Лише повідомлення боту | Повний доступ до всіх чатів | Повний доступ до всіх чатів  |
| Обмеження (Flood)   | Суворі ліміти API      | Помірні                     | Висока стійкість до лімітів  |
| Робота з медіа      | Обмеження до 50 МБ     | Без обмежень (до 2 ГБ)      | Без обмежень (до 2 ГБ)       |
| Парадигма           | Callback-базова        | Асинхронна                  | Асинхронна (більш стабільна) |

Вибір було зупинено на бібліотеці Telethon [20]. Хоча Pyrogram демонструє дещо вищу швидкість виконання окремих операцій, Telethon пропонує вищий рівень стабільності при тривалих сесіях та кращий інструментарій для низькорівневого контролю над завантаженням файлів, що є критичним для стабільного 24/7 моніторингу. Використання стандартного Bot API було відхилено через неможливість доступу до чатів, де бот не є адміністратором [6].

Важливим аспектом проектування є вибір системи управління базами даних (СУБД). У ході дослідження розглядалися три основні підходи: використання потужних клієнт-серверних рішень (PostgreSQL), документоорієнтованих систем (MongoDB) та вбудованих реляційних баз даних (SQLite). Вибір PostgreSQL чи MongoDB для OSINT-системи, що розгортається в межах приватного підприємства, є надлишковим, оскільки вони потребують окремих серверних ресурсів та постійного адміністрування.

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            |      |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            | 53   |

З огляду на специфіку комплексу, найбільш доцільним було визначено використання SQLite [13]. Головною перевагою даної СУБД є її «serverless» архітектура – база даних зберігається у вигляді єдиного файлу, що мінімізує затримки (latency). Особливе значення має підтримка режиму WAL (Write-Ahead Logging), що дозволяє виконувати операції читання та запису паралельно. Це забезпечує повну підтримку ACID-транзакцій при мінімальному споживанні ресурсів, поєднуючи надійність промислових стандартів із легкістю інтеграції в асинхронний стек.

Для реалізації інтелектуальної складової обрано архітектуру CLIP від OpenAI у зв'язці з фреймворком PyTorch [10, 17]. На відміну від класичних моделей класифікації, CLIP є мультимодальною, що дозволяє системі розпізнавати об'єкти за вільним текстовим описом (Natural Language Supervision). Це дає можливість миттєво змінювати критерії пошуку загроз без перенавчання всієї моделі. Взаємодія з користувачем реалізована через фреймворк Aiogram, який завдяки своїй асинхронній природі забезпечує миттєву доставку сповіщень адміністратору без блокування основних обчислювальних потоків [12].

Таким чином, сформований стек технологій (Python / Telethon / SQLite / CLIP / Aiogram) є збалансованим рішенням, яке забезпечує високу пропускну здатність системи, мінімальні затримки при аналізі контенту та простоту в обслуговуванні програмного комплексу.

## 2.4. Проєктування інтелектуального модуля фільтрації медіаконтенту на основі нейромереж

Після успішного завершення етапу проєктування механізмів перехоплення публікацій, постає критична проблема первинної сепарації отриманих даних. Враховуючи специфіку Telegram-спільнот, понад 70% вхідного потоку складає так званий «інформаційний шум»: агресивна реклама, спам-розсилки, іррелевантні текстові повідомлення та медійний контент, що не несе операційної цінності для служби безпеки. Відтак, функціонування системи неможливе без розробки

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            |      |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            | 54   |

багаторівневої системи фільтрації, яка на першому етапі здійснює відсікання явного «сміття» за формальними критеріями (наявність стоп-слів, посилань на сторонні ресурси, аналіз частоти публікацій), а на другому – залучає апарат штучного інтелекту для глибокої семантичної перевірки медіафайлів.

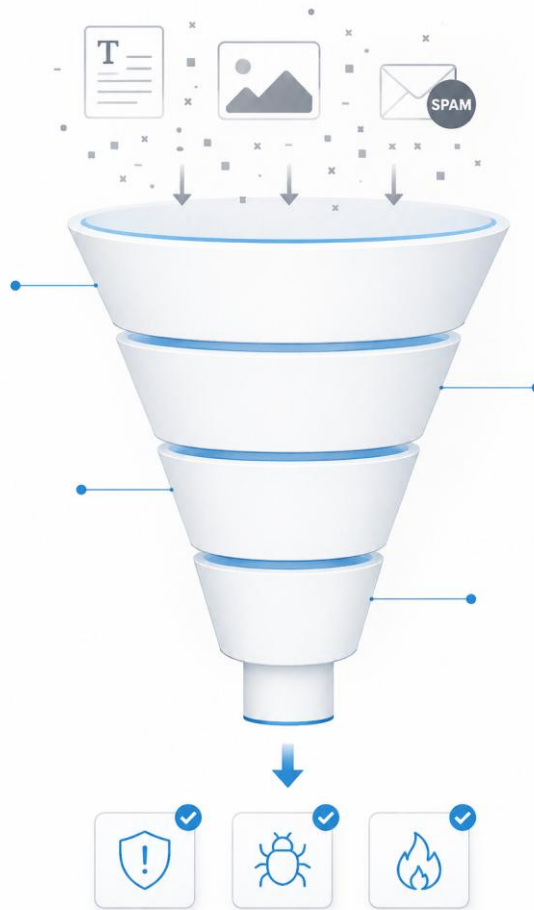


Рисунок 2.6 – Багаторівнева модель фільтрації вхідного контенту: від лінгвістичного аналізу до неймережевого інференсу

Для реалізації фінального, найбільш відповідального етапу фільтрації – відділення релевантних інцидентів від побутового візуального контенту – було спроектовано гібридну неймережеву архітектуру. Вона базується на синергії двох фундаментально різних підходів у галузі комп’ютерного зору: об’єктної детекції реального часу (YOLO) та мультимодального семантичного аналізу (CLIP). Таке поєднання дозволяє компенсувати обмеження кожного окремого алгоритму та досягти високої точності класифікації за рахунок крос-верифікації даних.

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            | 55   |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            |      |

Першою ланкою в цьому ланцюжку виступає модель YOLO (You Only Look Once). Її проєктна роль полягає у швидкому виявленні та просторовій локалізації фізичних об'єктів у кадрі. Завдяки високій швидкодії, YOLO виступає в ролі «детектора форм», який ідентифікує наявність у кадрі техніки, людей або специфічного обладнання за допомогою обмежувальних рамок (bounding boxes). Це дозволяє системі миттєво зрозуміти, на чому саме слід сфокусувати увагу, проте YOLO сама по собі не здатна оцінити контекст – наприклад, відрізнити мирне скупчення людей від агресивного інциденту.

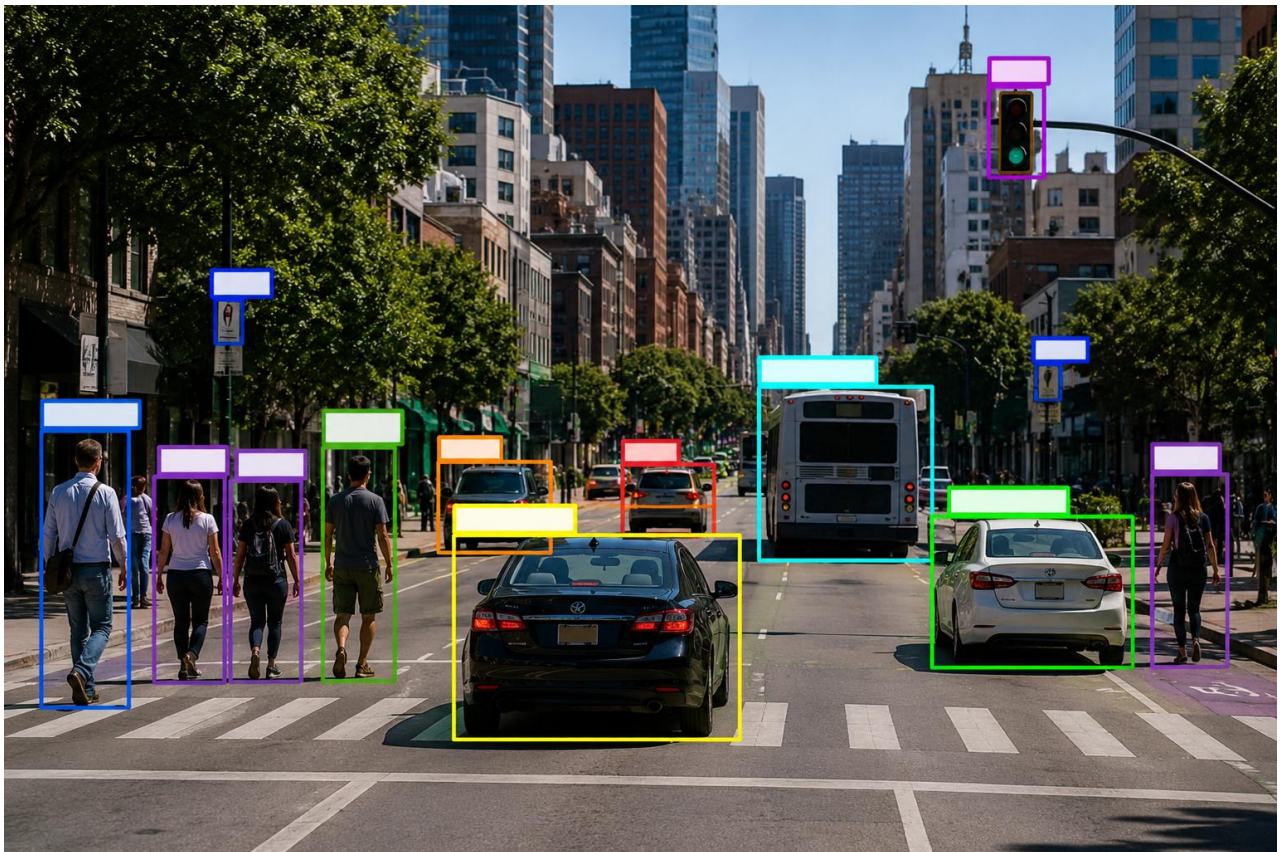


Рисунок 2.7 – Візуалізація роботи детектора YOLO: етап локалізації цільових об'єктів у кадрі

Саме для розв'язання проблеми контекстуального розуміння в систему інтегровано модель CLIP (Contrastive Language-Image Pre-training) від OpenAI. На відміну від класичних класифікаторів, CLIP пройшла навчання на гігантських масивах пар «зображення-текст», що дозволило їй сформувати універсальний векторний простір знань. У межах нашого проєкту CLIP виконує роль «інтелектуального арбітра»: вона отримує зображення (або кадри відео) і порівнює їхні цифрові відбитки (ембеддинги) із текстовими векторами загроз, описаними

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            | 56   |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            |      |

природною мовою (наприклад, «smoke and fire at the warehouse» або «suspicious drone activity»).

Процес прийняття рішення в інтелектуальному модулі базується на ієрархічній перевірці:

Рівень 1: Модель YOLO перевіряє наявність об'єкта. Якщо об'єкт не знайдено, система знижує пріоритет, але не ігнорує файл, якщо текстовий опис у самому пості містить тривожні маркери.

Рівень 2: Модель CLIP проводить семантичне порівняння всього кадру з цільовими промптами. Якщо косинусна відстань між векторами зображення та тексту перевищує встановлений поріг (Threshold), подія вважається релевантною.

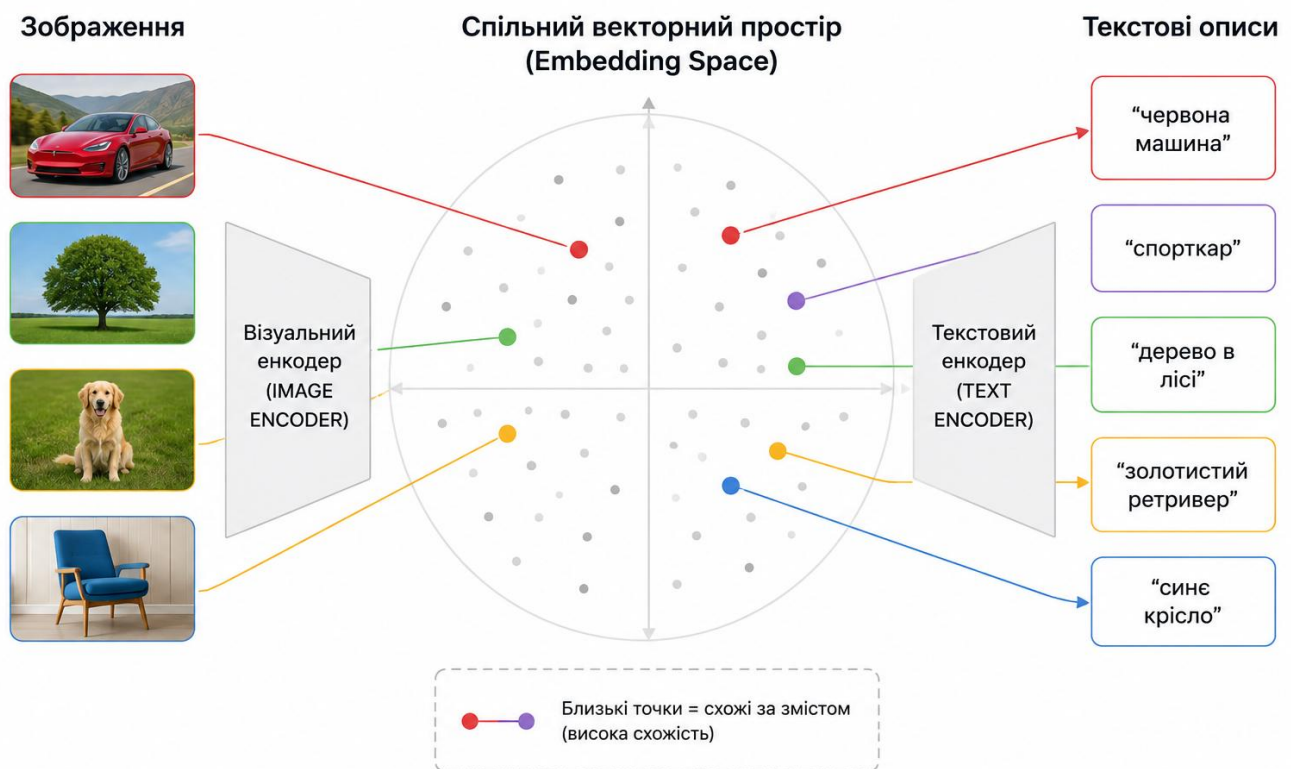


Рисунок 2.8 – Механізм семантичного зіставлення візуальних та текстових ознак у моделі CLIP

Для відеоконтенту, який є найбільш інформативним, але водночас складним для аналізу, було розроблено алгоритм часового семплінгу. Проєкт передбачає автоматичну декомпозицію відеофайлу на 40 репрезентативних кадрів, рівномірно розподілених по всій тривалості ролика. Кожен із цих кадрів проходить через

описаний гібридний конвеєр. Кінцевий вердикт про релевантність відео приймається на основі методу мажоритарного голосування: якщо більше 25% аналізованих кадрів отримали високий Confidence Score від обох моделей, відео маркується як інцидент і передається оператору.

Така двохетапна структура – від базової фільтрації реклами до складного нейромережевого аналізу – дозволяє перетворити хаотичний потік повідомлень Telegram у структуровану стрічку перевірених подій. Гнучкість системи полягає в тому, що для зміни фокусу моніторингу достатньо лише скоригувати текстові описи для CLIP, що робить програмний комплекс універсальним інструментом для захисту активів приватного бізнесу в умовах сучасних інформаційних викликів.

## 2.5. Розробка алгоритму функціонування системи та логіки маршрутизації даних

На основі розроблених архітектурних рішень та обраного технологічного стека сформовано загальний алгоритм функціонування системи, який базується на принципах паралельного моніторингу та селективної фільтрації. Особливістю цього алгоритму є впровадження дворівневої структури робочих груп у месенджері Telegram, що дозволяє поєднати переваги автоматизації із можливістю повного аудиту вхідного інформаційного потоку. Такий підхід мінімізує ризик повної втрати критично важливих даних через потенційні помилки першого чи другого роду під час роботи нейромережових моделей.

Процес функціонування починається на етапі ініціалізації асинхронного клієнта, який у реальному часі прослуховує масив цільових інформаційних каналів. При появі нової публікації, що містить візуальний контент, система миттєво перехоплює повідомлення та виконує процедуру автоматичного збагачення даних. На цьому кроці за ідентифікатором першоджерела надсилається швидкий запит до бази даних-довідника, звідки вилучається інформація про територіальну приналежність каналу. Отримані метадані, включаючи назву групи, ім'я автора та

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            |      |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            | 58   |

конкретну область України, автоматично додаються до тексту публікації, після чого повідомлення негайно пересилається у першу, загальну групу моніторингу.

Цей загальний чат виконує функцію накопичувача сирих первинних даних, де зберігаються абсолютно всі перехоплені матеріали з регіональною прив'язкою. Наявність такого джерела є критично важливою для корпоративної безпеки, оскільки оператор або аналітик завжди має можливість провести ручний аудит і переглянути контент, який міг бути відсіяний автоматичними фільтрами. Таким чином усувається головний недолік багатьох інтелектуальних систем – незворотна втрата інформації у разі хибнонегативного спрацювання штучного інтелекту.

Паралельно із надсиланням повідомлення до загального чату запускається внутрішній конвеєр глибокої аналітики. Спочатку контент проходить через базові лінгвістичні та евристичні фільтри, які відсікають спам, взаємний піар пабліків та рекламні інтеграції за формальними ознаками. Публікації, що успішно подолали первинне очищення, передаються на вхід інтелектуальному модулю. Навчена під специфічні класи загрози модель YOLO здійснює просторове сканування кадру та локалізує фізичні об'єкти за допомогою обмежувальних рамок. У разі детекції потенційного інциденту, дані передаються мультимодальній моделі CLIP для фінальної верифікації контексту.

Оскільки CLIP оперує базою із кількох сотень заздалегідь підготовлених семантичних промптів, орієнтованих на захист приватного бізнесу, аналіз дозволяє чітко диференціювати побутові події від реальних загроз, таких як пожежі, руйнування інфраструктури чи поява підозрілих об'єктів. Система обчислює косинусну відстань між вектором зображення та масивом текстових сценаріїв і формує фінальний показник впевненості. Якщо цей показник перевищує встановлений у конфігурації поріг релевантності, повідомлення вважається верифікованим інцидентом.

На завершальному етапі алгоритму активується логіка фінальної маршрутизації, і відфільтроване повідомлення пересилається у другу, цільову групу сповіщень. У цей чат потрапляють виключно очищені від інформаційного шуму та підтверджені нейромережами публікації з чітким зазначенням прив'язаної

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            |      |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            | 59   |

області. Описана дворівнева схема маршрутизації дозволяє службі безпеки підприємства оперативно реагувати на критичні виклики у цільовому чаті, повністю позбавляючись від проблеми інформаційного перевантаження, але зберігаючи доступ до повної стрічки подій у загальному моніторинговому вікні.

## Висновки до другого розділу

У ході виконання другого розділу було проведено комплексне архітектурне проектування інтелектуальної системи OSINT-моніторингу та сформовано її ключові алгоритмічні компоненти. Детальний аналіз предметної області дозволив розробити модульну структуру програмного комплексу, яка забезпечує високу стійкість до навантажень, гнучкість та ізоляцію окремих процесів. Завдяки розділенню системи на незалежні блоки збору, обробки та маршрутизації даних, досягнуто мінімізації затримок при передачі інформації у реальному часі, що є критично важливим для забезпечення безпеки комерційних об'єктів.

Обґрунтування технологічного стеку підтвердило доцільність використання мови програмування Python як фундаменту системи завдяки її найпотужнішій екосистемі в галузі штучного інтелекту та підтримці асинхронного програмування через модуль `asuncio` для ефективної обробки мережеских потоків I/O-bound. Порівняльний аналіз засобів взаємодії з Telegram API дозволив обрати бібліотеку `Telethon`, яка продемонструвала вищу стабільність при тривалих сесіях порівняно з аналогами та забезпечила необхідний низькорівневий контроль над завантаженням файлів. Для реалізації службових довідників було обрано легку СУБД `SQLite`, яка завдяки своїй безсерверній архітектурі усуває потребу в складній інфраструктурі та мінімізує час відгуку системи.

Центральним архітектурним рішенням розділу стала розробка гібридного нейромережевого конвеєра, що базується на синергії моделей YOLO та CLIP. Жорстка спеціалізація та попереднє донавчання моделі YOLO під конкретні класи матеріальних загроз дозволили створити швидкий просторовий фільтр для первинної локалізації об'єктів. Водночас інтеграція мультимодальної архітектури

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            | 60   |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            |      |

CLIP, яка оперує базою із кількох сотень заздалегідь підготовлених семантичних промптів, забезпечила глибоке контекстуальне розуміння візуальних сцен без необхідності постійного перенавчання системи.

Особливу практичну цінність має розроблена логіка маршрутизації даних, яка функціонує за дворівневим принципом і повністю вирішує проблему інформаційного перевантаження оператора. Створення першої, загальної групи моніторингу із автоматичним збагаченням даних про територіальну приналежність джерел гарантує збереження повної стрічки подій та можливість ручного аудиту контенту для усунення ризику хибнонегативних помилок ШІ. Друга, цільова група сповіщень акумулює виключно верифіковані нейромережами інциденти, що дозволяє службі безпеки фокусуватися на критичних подіях. Спроектowana архітектура є логічно завершеною, теоретично обґрунтованою та становить надійний фундамент для практичної програмної реалізації системи, яка буде детально розглянута у наступному розділі.

Узагальнюючи результати етапу проєктування, можна стверджувати, що всі закладені в архітектуру рішення спрямовані на досягнення балансу між швидкістю обробки великих масивів інформації та точністю класифікації загроз. Відмова від традиційних громіздких архітектур зберігання на користь динамічної маршрутизації в реальному часі дозволила створити легке, але водночас відмовостійке програмне рішення. Сформовані структурно-логічні схеми та алгоритми взаємодії між модулями повністю описують майбутню поведінку системи, що усуває будь-яку невизначеність і дозволяє безпосередньо перейти до розробки програмного коду, розгортання середовища нейромережевого інференсу та проведення експериментального тестування комплексу.

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            |      |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            | 61   |

## РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ СИСТЕМИ ІНТЕЛЕКТУАЛЬНОГО OSINT-МОНІТОРИНГУ

### 3.1. Реалізація модуля збору даних

Практична реалізація підсистеми збору даних базується на створенні високопродуктивного асинхронного клієнта, який здійснює безпосередню взаємодію з серверами месенджера Telegram за допомогою низькорівневого бінарного протоколу MTProto [20, 1]. Вибір цього підходу замість класичного використання Telegram Bot API зумовлений потребою в отриманні доступу до широкого спектра відкритих каналів без необхідності додавання бота в адміністратори кожного окремого джерела. Головним інженерним викликом на цьому етапі стало забезпечення стабільного та безперебійного моніторингу масштабного пулу інформаційних джерел, загальна кількість яких у конфігурації системи перевищує одну тисячу каналів одночасно. При роботі з такою щільною мережею вхідних з'єднань стандартні синхронні або багатопотокові (multithreading) підходи виявляються повністю нежиттєздатними через надлишкове споживання системних ресурсів. Оскільки виклик кожної мережевої операції введення-виведення у синхронному режимі повністю блокує весь потік виконання програми, системі знадобилося б виділення тисяч окремих потоків ОС, що призвело б до критичного перевантаження процесора на перемикання контекстів та дефіциту оперативної пам'яті під стек кожного потоку.

Впровадження асинхронної парадигми програмування дозволило успішно подолати ці інфраструктурні обмеження та реалізувати гнучку подієво-орієнтовану архітектуру. Завдяки використанню низькорівневих системних викликів селекції сокетів обробка нових публікацій, що надходять від понад тисячі каналів, відбувається у повністю неблокуючому режимі в межах єдиного головного системного циклу подій. Цей підхід дозволяє радикально мінімізувати споживання оперативної пам'яті та обчислювальних ресурсів центрального процесора, оскільки замість очікування відповіді від сервера потік негайно перемикається на обробку

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            |      |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            | 62   |

інших активних мережевих сокетів. Логічна структура розподілу компонентів системи за модулями та файлами представлена на рисунку 3.1, де відображено чітку ізоляцію архітектурних шарів програмного комплексу.

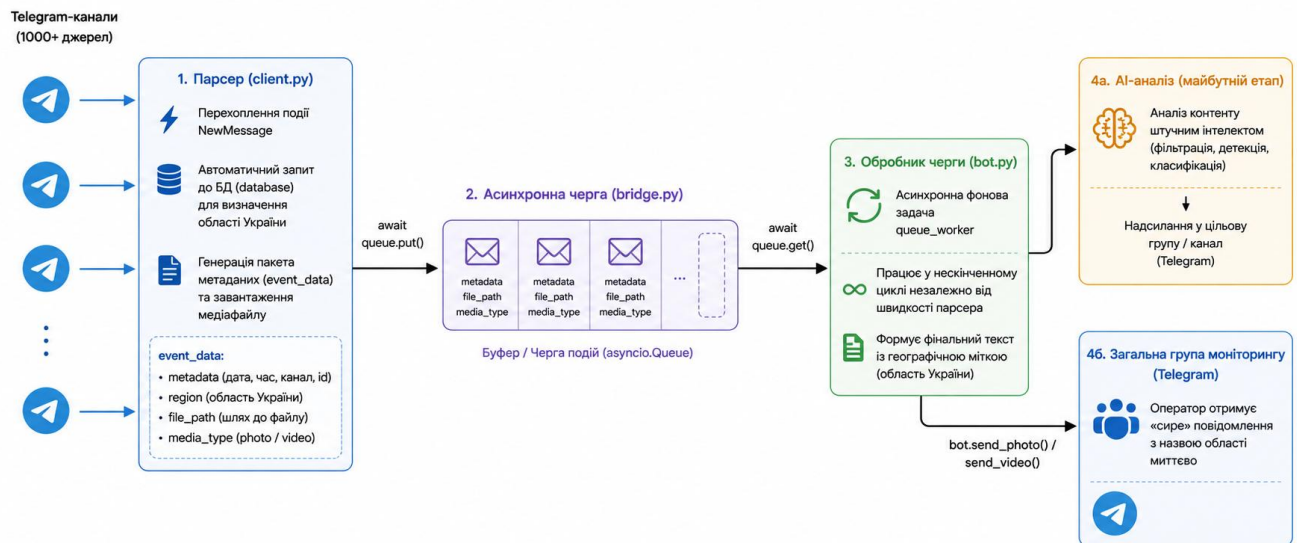


Рисунок 3.1 – Схема асинхронної взаємодії модулів через чергу

Програмна архітектура самого транспортного моста винесена в окремий файл конфігурації та модульних параметрів (config.py або pipeline\_config.json). Таке архітектурне рішення дозволяє повністю ізолювати логіку транспортування даних від логіки роботи нейромережевих моделей, забезпечуючи високу гнучкість при масштабуванні або модифікації системи. Користувач або розробник може змінювати параметри буферизації, ліміти пропускної здатності та шляхи маршрутизації потоків даних без необхідності втручання в основний програмний код інтелектуального модуля.

Реалізація цього зв'язувального механізму базується на асинхронних чергах (asyncio.Queue) та оперує структурованими кортежами даних (immutable tuples) із суворим розподілом на логічні блоки [4]. Використання незмінних кортежів замість динамічних об'єктів або словників мінімізує накладні витрати на оперативну пам'ять та гарантує потокобезпеку (thread-safety) при паралельній обробці медіапотоків. Кожен транспортний кортеж має фіксовану архітектурну структуру і містить три ключові складові.

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            | 63   |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            |      |

Першою складовою є блок метаданих (metadata\_block), який акумулює в собі ідентифікатор першоджерела (ID Telegram-каналу), унікальний номер повідомлення, мітку часу його фіксації, а також географічний маркер області, автоматично зчитаний із бази даних-довідника на етапі перехоплення. Це дозволяє супроводжувати контент інформаційним контекстом протягом усього циклу обробки.

Другою складовою виступає локальний шлях до файлу (local\_file\_path), що представляє собою строковий параметр, який вказує на тимчасове розташування завантаженого медіафайлу у буферному сховищі диска. Передача локальних шляхів замість «важких» бінарних об'єктів безпосередньо через оперативну пам'ять дозволяє уникнути критичних витоків пам'яті (OOM – Out of Memory) при одночасному опрацюванні десятків великих відеофайлів.

Третьою складовою є тип контенту (content\_type\_flag), який слугує маркером (наприклад, PHOTO або VIDEO) і чітко вказує транспортному мосту, за яким саме алгоритмом часового семплінгу або перцептивного хешування слід опрацьовувати даний файл на наступних етапах конвеєра.

Передача цих кортежів реалізована через конвеєрний механізм «Виробник – Споживач» (Producer-Consumer). Модуль перехоплення (Telethon) виступає в ролі виробника, що безперервно наповнює асинхронну чергу, а гібридний ШІ-модуль (YOLO + CLIP) виступає споживачем, який вилучає кортежі по мірі вивільнення обчислювальних ресурсів відеокарти. Програмна реалізація цього зв'язувального механізму, що демонструє архітектуру транспортного конвеєра та логіку формування інформаційних кортежів, представлена у вигляді лістингу програмного блоку на рисунку 3.2.

```
import asyncio

# Глобальна асинхронна черга для передачі подій від Клієнта до Бота.
# Кожен елемент у черзі є кортежем: (event_data: dict, file_path: str, media_type: str)
queue = asyncio.Queue()
```

Рисунок 3.2 – Фрагмент програмного коду реалізації асинхронної черги (bridge.py)

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            |      |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            | 64   |

При запуску системи у головному керуючому модулі асинхронно стартує клієнт збору даних. Він виконує первинну автоматичну синхронізацію чатів, яка перевіряє всі поточні підписки акаунта, відокремлює групи від широкомовних каналів та оновлює інформацію в локальній базі даних. Обробка нових публікацій виконується за допомогою фонового слухача подій, який перехоплює вхідний потік медіафайлів у реальному часі.

Процес обробки кожної окремої події ініціюється в момент фіксації нового повідомлення в месенджері та включає кілька послідовних технологічних етапів. На першому етапі здійснюється перевірка джерела на предмет валідності та відсутності у списку ігнорування. Після цього система надсилає асинхронний запит до бази даних-довідника для визначення територіальної приналежності каналу за його унікальним ідентифікатором. Отриманий регіональний тег дозволяє класифікувати джерело за конкретною областю, що є критично важливим для подальшої територіальної маршрутизації інформації.

Наступним кроком є формування єдиного уніфікованого пакета метаданих, який акумулює часові мітки, параметри автора, першоджерело та географічний маркер. Паралельно з цим процесом запускається асинхронне завантаження медіафайлу у тимчасову локальну директорію буферного сховища диска з автоматичним присвоєнням унікального імені для запобігання перезапису даних.

Завершується цей цикл пакуванням сформованих метаданих та локального шляху до файлу в незмінний транспортний кортеж, який миттєво відправляється в асинхронну чергу обробки. Завдяки такій побудові архітектури модуль збору даних не очікує завершення нейромережевого аналізу, а відразу повертається до прослуховування ефіру, що повністю нівелює ризик пропуску важливих публікацій навіть за умов екстремального навантаження на інформаційні канали. Програмна логіка перехоплення, збагачення регіональними тегами та відправки події у чергу продемонстрована на рисунку 3.3.

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            |      |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            | 65   |

```

@app.on(events.NewMessage)
async def handle_new_message(event):
    try:
        if not database.is_source_chat_monitored(event.chat_id):
            return

        chat_info = database.get_source_chat_info(event.chat_id)
        region = chat_info.get('region', 'Невідомий region')

        event_data = {...}

        path = None
        media_type = 'text'

        if event.media:
            temp_dir = "temp_media"
            os.makedirs(temp_dir, exist_ok=True)
            path = await app.download_media(event.message, file=temp_dir)
            if event.photo:
                media_type = 'photo'
            elif event.video:
                media_type = 'video'

        await bridge.queue.put((event_data, path, media_type))

    except Exception as e:
        logger.error(f"Помилка обробки медіа в клієнті: {e}")

```

Рисунок 3.3 – Фрагмент програмного коду модуля перехоплення та збагачення даних (client.py)

У процесі масштабування моніторингу до понад тисячі джерел ключовим бар'єром стають жорсткі ліміти серверів Telegram типу FloodWait, які тимчасово блокують сесію користувача у разі перевищення критичної кількості запитів на секунду. Це найчастіше трапляється під час масованих інцидентів, коли одночасно оновлюються стрічки десятків або сотень пабліків, створюючи лавиноподібне навантаження на мережевий інтерфейс системи. Для обходу цих обмежень у програмному коді клієнта реалізовано адаптивний механізм перехоплення виняткових ситуацій перевантаження за допомогою конструкцій try-except на рівні асинхронних викликів бібліотеки Telethon.

Система динамічно аналізує параметри системної помилки, вилучає точний час у секундах, на який сервер месенджера призупинив дію поточної сесії, і переводить у режим неблокуючого очікування `await asyncio.sleep()` виключно той потік запитів, який викликав блокування. Такий підхід дозволяє іншим незалежним

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            |      |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            | 66   |

модулям і фоновим корутинам продовжувати збір та обробку інформації без зупинки всього програмного комплексу, що мінімізує затримки у надходженні критично важливих даних.

Паралельно в системі функціонує фонові асинхронна задача обробника черги, запущеність якої реалізована як нескінченний цикл `while True`. Вона безперервно вилучає інформаційні блоки даних із транспортного моста та миттєво реалізує логіку першого рівня нашої архітектури. Цей етап передбачає декомпозицію отриманого незмінного кортежу, зчитування локального шляху до медіафайлу, миттєве автоматичне формування структурованого текстового опису із вбудованими географічними метаданими, такими як назва джерела та конкретна область України, а також подальшу трансляцію цього сирого результату в загальну групу моніторингу.

Оскільки даний потік даних направляється в першу моніторингову групу ще до початку тривалих обчислень нейромереж, досягається максимальна швидкість інформування. Наявність такого паралельного маршруту гарантує, що оператор або аналітик служби безпеки завжди має доступ до повного первинного інформаційного потоку для здійснення швидкого ручного аудиту або візуальної перевірки у разі потенційного пропуску екстремальної події алгоритмами штучного інтелекту на наступних, більш суворих рівнях фільтрації. Реалізація цієї логіки черги та надсилання первинного контенту відображена у вигляді програмної схеми на рисунку 3.4. Окрім цього, такий підхід дозволяє нівелювати затримки в інтерфейсі користувача, оскільки важкі обчислювальні операції нейромережевого інференсу винесені в окремі ізольовані потоки виконання. Це забезпечує безперебійне функціонування системи в умовах критичного пікового навантаження, коли швидкість первинного сповіщення має вирішальне значення для упередження безпекових ризиків бізнесу.

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            | 67   |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            |      |

```

async def queue_worker():
    logger.info("Фоновий обробник черги успішно запущено.")
    while True:
        try:
            # 1. Асинхронне вилучення чергового пакета даних із моста
            event_data, file_path, media_type = await bridge.queue.get()

            # Формування інформаційного тексту із географічними метаданими
            caption = (
                f"📍 *Джерело:* {event_data['chat_title']}\n"
                f"📍 *Region:* {event_data['region']}\n"
                f"📄 *Текст:* {event_data['text']}\n"
            )

            # КРОК 1: Миттєва маршрутизація у Загальну групу моніторингу (Raw Storage)
            general_chat_id = database.get_general_monitoring_chat()
            if general_chat_id and file_path and os.path.exists(file_path):
                if media_type == 'photo':
                    await bot.send_photo(general_chat_id, photo=FSInputFile(file_path), caption=caption,
                                         parse_mode='Markdown')
                elif media_type == 'video':
                    await bot.send_video(general_chat_id, video=FSInputFile(file_path), caption=caption,
                                         parse_mode='Markdown')

            bridge.queue.task_done()

        except Exception as e:
            logger.error(f"Помилка у роботі queue_worker: {e}")
            await asyncio.sleep(1)

```

Рисунок 3.4 – Фрагмент програмного коду фонового обробника черги та маршрутизації (bot.py)

Візуальний результат роботи цього алгоритму безпосередньо у першому вікні інтерфейсу оператора показано на рисунку 3.5. Скріншот демонструє, як кожне перехоплене повідомлення з пабліка автоматично маркується територіальною міткою на основі даних, отриманих із бази-довідника, безпосередньо перед його публікацією у першій групі загального аудиту. В інтерфейсі чітко відображається сформована структура картки події, де у верхній частині текстового блоку фіксується назва першоджерела у вигляді гіперпосилання, а також уніфікований географічний тег, що вказує на конкретну область України.

Окрім цього, на зображенні видно, що оригінальний медіаконтент (фотографія або статичний кадр із відеопотоку) публікується без втрати якості та без будь-яких візуальних модифікацій, що є критично важливим для коректного проведення ручного експертного аналізу. Таке системне компонування елементів

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            |      |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            | 68   |

повідомлення дозволяє аналітику служби безпеки за лічені секунди зорієнтуватися в просторі та оцінити ситуацію, мінімізуючи час на пошук контекстуальної інформації про джерело та локацію події.

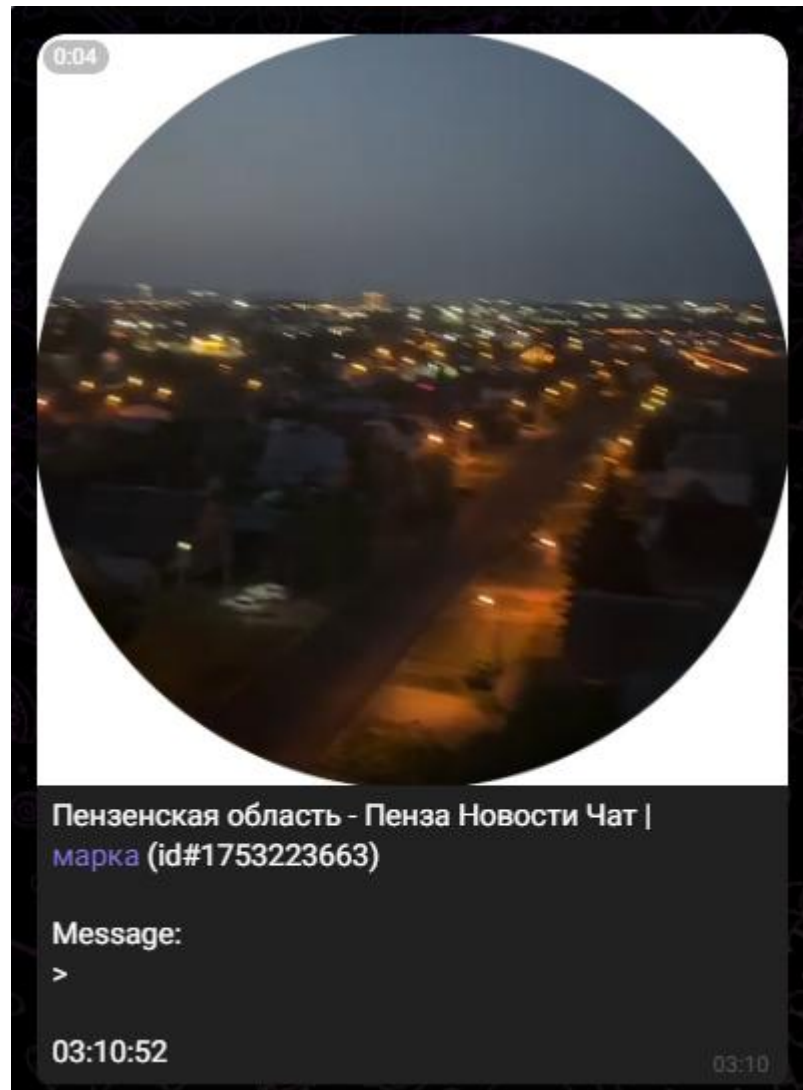


Рисунок 3.5 – Візуалізація першого рівня маршрутизації: Повідомлення в загальній групі моніторингу

Таким чином, завдяки суворому розподілу функціональних обов'язків між модулем асинхронного збору та модулем первинного надсилання, які архітектурно зв'язані через неблокуючу чергу повідомлень, розроблений програмний комплекс забезпечує високопродуктивний, надійний та безперебійний імпорт медіаданих із понад 1000 джерел у цілодобовому режимі 24/7. Впровадження цієї дворівневої схеми обробки інформаційного потоку дозволило повністю ізолювати інтенсивні дискові та мережеві операції введення-виведення від наступних фаз математичних обчислень.

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            |      |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            | 69   |

Створена інфраструктура збору та первинної маршрутизації повністю нівелює ризики виникнення критичних блокувань сесій або незворотної втрати вхідного трафіку під час масованих інформаційних подій, коли інтенсивність генерації контенту в месенджері досягає пікових значень. Рациональне використання асинхронних патернів програмування та оптимізоване пакування метаданих у незмінні структури формують стійку й відмовостійку архітектурну основу, необхідну для стабільної та ефективної роботи інтелектуальних моделей штучного інтелекту на подальших, найбільш обчислювально містких етапах загального конвеєра фільтрації та семантичного аналізу інцидентів.

### **3.2. Налаштування та навчання моделі ШІ для розпізнавання візуальних маркерів загроз**

Практична реалізація інтелектуального модуля системи моніторингу розпочалася з розгортання та конфігурації спеціалізованого програмно-апаратного середовища розробки. Для забезпечення високої швидкості математичних обчислень під час Fine-tuning об'єктного детектора YOLO було розгорнуто середовище на базі мови програмування Python версії 3.10 з інтеграцією бібліотек машинного навчання PyTorch та комп'ютерного зору OpenCV. Обов'язковою умовою для проведення навчання у розумні часові терміни стало використання апаратного прискорення на базі архітектури NVIDIA CUDA [9, 19]. У програмному коді керуючого скрипта було реалізовано динамічну перевірку доступності графічного процесора, що дозволяє автоматично переводити обчислення на тензорні ядра відеокарти у разі їх виявлення, мінімізуючи навантаження на центральний процесор системи.

Наступним критично важливим етапом стала підготовка та стандартизація комбінованого датасету. Оскільки зібрані вручну 900 унікальних кадрів з українських Telegram-каналів відображали специфічні та складні умови зйомки, було застосовано програмні інструменти аугментації даних для штучного розширення вибірки та підвищення стійкості моделі. Процес аугментації включав

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            | 70   |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            |      |

випадкове дзеркальне відображення по горизонталі, довільне масштабування в межах десяти відсотків, а також штучне внесення гаусового шуму та зміну колірної гами у просторі HSV. Це дозволило симулювати умови нічної зйомки, задимленості, низької роздільної здатності або наявності артефактів стиснення відео, які є характерними для контенту з мобільних телефонів очевидців, що суттєво покращило здатність ШІ розпізнавати нестандартні осередки небезпеки. Методологія підготовки графічних елементів, інтерфейс ручного анотування та логіка інтеграції семантичного ядра представлені нижче на загальній схемі підготовки даних.

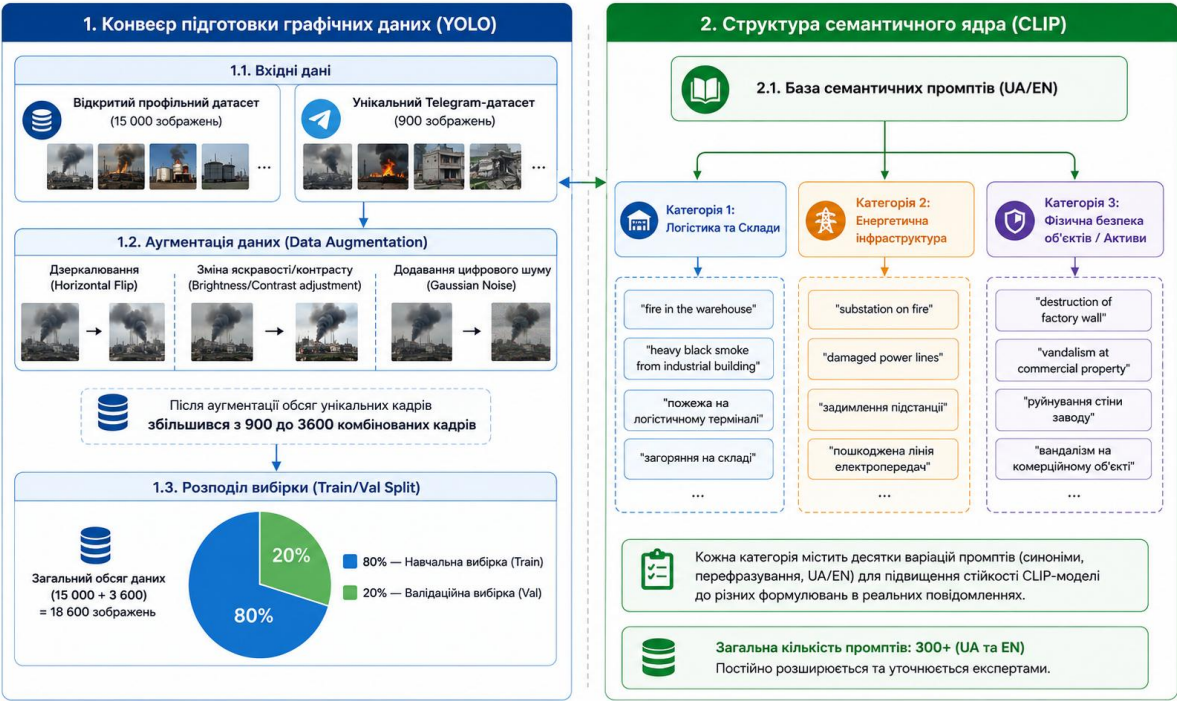


Рисунок 3.6 – Методологія підготовки даних та структура семантичного ядра системи моніторингу

Для здійснення процесу ручної розмітки 900 унікальних кадрів, вилучених із моніторингового ефіру неструктурованих масивів даних, розробниками було використано спеціалізоване кросплатформове середовище анотування графічних даних LabelImg, що функціонує на базі графічної бібліотеки Qt та мови програмування Python. Цей етап мав на меті створення високоякісних «золотих стандартів» (Ground Truth) для навчання нейромережі, що вимагало виключної точності позиціонування рамок.

Процес передбачав декомпозицію кожного графічного об'єкта та виділення його просторових меж за допомогою прямокутних обмежувальних рамок (bounding boxes) із суворою прив'язкою до чотирьох базових інженерних класів: вогонь, дим, руйнування інфраструктури та підозріла спецтехніка. Особлива складність розмітки полягала в анотуванні диму та вогню, які, на відміну від класичних твердотільних об'єктів, мають аморфну структуру, динамічні контури та розмиті межі. Для нівелювання фактора суб'єктивності оператора було застосовано методику подвійного перехресного контролю (cross-verification), за якої кожен кадр незалежно розмічувався двома розробниками з подальшим розрахунком метрики перетину над об'єднанням ( $\text{IoU} > 0.85$ ) для підтвердження валідності внесеної рамки.

Результатом цього трудомісткого етапу стало формування структурованого масиву супутніх текстових файлів конфігурації анотацій з розширенням .txt, що повністю відповідають специфікації архітектури YOLO. Для кожного виділеного об'єкта у файлі анотації генерувався окремий рядок, де перше значення визначало індекс класу (від 0 до 3), а наступні чотири параметри описували геометрію рамки. Такий підхід забезпечує інваріантність детектора до масштабу зображення, дозволяючи моделі коректно зчитувати координати незалежно від початкової роздільної здатності кадру. Візуальний інтерфейс робочого простору програми анотування LabelImg у процесі детальної сегментації критичних маркерів інцидентів та фіксації контурів об'єктів зафіксовано на рисунку 3.7.

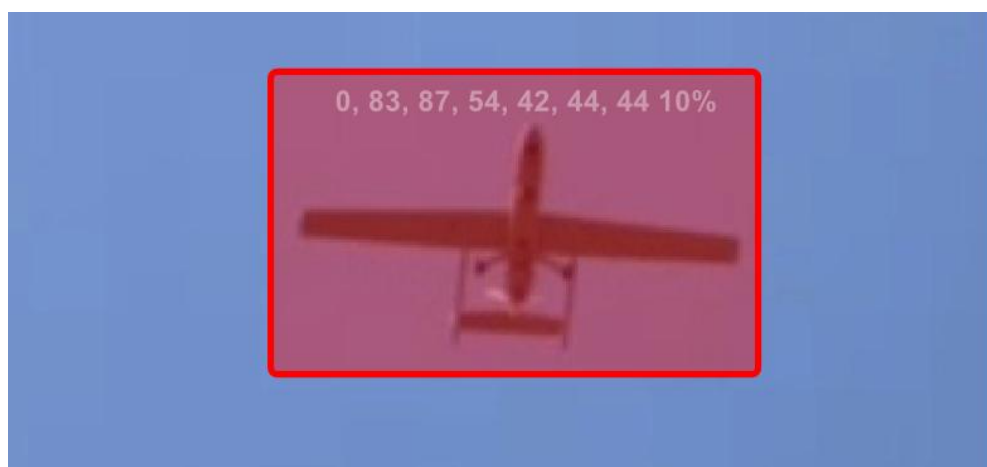


Рисунок 3.7 – Робочий простір програми LabelImg у процесі ручної розмітки унікального датасету

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            |      |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            | 72   |

Для забезпечення коректної ініціалізації процесу навчання та безперебійної подачі графічних пакетів на вхід тензорного конвеєра, весь комбінований масив, що включає базові профільні датасети та унікальні локальні кадри, було автоматично розділено на дві ізольовані вибірки у співвідношенні 80% (train) на 20% (val). Вибір оптимізатора AdamW (Adam з інтегрованим розчепленням ваг / Weight Decay) зумовлений його високою стабільністю при роботі з комбінованими наборами даних, де різні сегменти мають різну дисперсію ознак і геометричні контури. Розмір батчу  $N=16$  було обрано як гранично допустимий для збереження стабільності градієнтного спуску в межах доступного об'єму відеопам'яті графічного процесора (VRAM), що дозволило уникнути помилок переповнення буфера (CUDA out of memory) під час безперервного циклу Fine-tuning.

Масштабування обчислень у процесі ітераційного оновлення ваг нейромережі здійснювалося за рахунок розпаралелювання потоків на тензорних ядрах, де математичні операції з матрицями великої розмірності виконувалися в оптимізованому режимі змішаної точності (Mixed Precision Training). Це дозволило скоротити загальний час розрахунків без втрати підсумкової точності локалізації об'єктів. Для забезпечення стійкості моделі до виявлення низькорозмірних цільових об'єктів на різних фонових підкладках алгоритм автоматично формував комбіновані тренувальні пакети.

Використання мозаїчного поєднання чотирьох випадкових зображень в один кадр змушує нейромережу розпізнавати цільові загрози у значно меншому масштабі та за межами їхнього звичного просторового контексту. Таке штучне ущільнення інформації всередині кожного батчу дозволяє стабілізувати роботу шарів пакетної нормалізації (Batch Normalization) без необхідності радикального збільшення апаратних вимог до оперативної пам'яті GPU. Візуалізацію процесу ітераційного навчання, накладання обмежувальних рамок детектора (bounding boxes) та розрахунку початкових коефіцієнтів впевненості для класу повітряних маркерів загроз на прикладі сформованої мозаїчної матриці пакету даних наведено на рисунку 3.8.

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            | 73   |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            |      |



Рисунок 3.8 – Візуалізація мозаїчного пакету даних детектора YOLO в процесі ітераційного навчання моделі

Безпосередній контроль за ходом оптимізації моделі здійснювався через аналіз логів консолі середовища розробки у режимі реального часу. Під час виконання Fine-tuning на базі апаратної архітектури CUDA фіксувалися проміжні компоненти функцій втрат обмежувальних рамок (box\_loss), розподілу контурів (dfl\_loss) та точності розпізнавання класів (cls\_loss) для кожної ітерації пакету даних (Instances). Фіксація системних метрик утилізації відеопам'яті (на рівні 7.44 – 7.49 Гб) та крокових прогрес-барів валідаційних циклів на початкових епохах навчання відображена на рисунку 3.9.

Поступове зниження значень функцій втрат від початкових рівнів (~2.2) свідчить про успішне коригування вагових коефіцієнтів зворотним поширенням помилки та послідовне зближення передбачених контурів із реальними об'єктами. Мінімальні значення інтегральних метрик точності mAP50 та mAP50-95 на перших ітераціях є закономірним наслідком випадкової ініціалізації параметрів ядра, що підтверджує необхідність тривалого циклу навчання для накопичення семантичних ознак. Стабільна швидкість обробки пакетів та відсутність різких стрибків у часових логах крокових прогрес-барів вказують на збалансовану роботу конвеєра даних, яка виключає ризик виникнення апаратних затримок при зчитуванні з накопичувача.

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            | 74   |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            |      |

|                                                         |       |        |           |        |       |           |                 |
|---------------------------------------------------------|-------|--------|-----------|--------|-------|-----------|-----------------|
| 2/150                                                   | 7.44G | 2.151  | 2.252     | 2.219  | 39    | 640: 63%  | 41/65 2.        |
| 2/150                                                   | 7.44G | 2.155  | 2.248     | 2.22   | 38    | 640: 65%  | 42/65 2.        |
| 2/150                                                   | 7.45G | 2.156  | 2.248     | 2.22   | 43    | 640: 66%  | 43/65 2.        |
| 2/150                                                   | 7.45G | 2.161  | 2.25      | 2.221  | 37    | 640: 68%  | 44/65 2.        |
| 2/150                                                   | 7.45G | 2.166  | 2.252     | 2.222  | 58    | 640: 69%  | 45/65 2.        |
| 2/150                                                   | 7.46G | 2.168  | 2.254     | 2.223  | 44    | 640: 71%  | 46/65 2.        |
| 2/150                                                   | 7.44G | 2.169  | 2.26      | 2.221  | 45    | 640: 72%  | 47/65 2.        |
| 2/150                                                   | 7.45G | 2.168  | 2.262     | 2.219  | 46    | 640: 74%  | 48/65 3.        |
| 2/150                                                   | 7.45G | 2.169  | 2.268     | 2.22   | 31    | 640: 75%  | 49/65 2.        |
| 2/150                                                   | 7.45G | 2.176  | 2.274     | 2.224  | 28    | 640: 77%  | 50/65 2.        |
| 2/150                                                   | 7.45G | 2.178  | 2.277     | 2.223  | 39    | 640: 78%  | 51/65 2.        |
| 2/150                                                   | 7.45G | 2.182  | 2.274     | 2.225  | 38    | 640: 80%  | 52/65 2.        |
| 2/150                                                   | 7.5G  | 2.181  | 2.273     | 2.223  | 48    | 640: 82%  | 53/65 2.        |
| 2/150                                                   | 7.44G | 2.178  | 2.272     | 2.223  | 36    | 640: 83%  | 54/65 2.        |
| 2/150                                                   | 7.44G | 2.178  | 2.271     | 2.224  | 50    | 640: 85%  | 55/65 2.        |
| 2/150                                                   | 7.44G | 2.177  | 2.273     | 2.225  | 39    | 640: 86%  | 56/65 2.        |
| 2/150                                                   | 7.45G | 2.172  | 2.272     | 2.225  | 39    | 640: 88%  | 57/65 2.        |
| 2/150                                                   | 7.45G | 2.168  | 2.273     | 2.226  | 26    | 640: 89%  | 58/65 2.        |
| 2/150                                                   | 7.44G | 2.166  | 2.273     | 2.226  | 45    | 640: 91%  | 59/65 2.        |
| 2/150                                                   | 7.45G | 2.166  | 2.274     | 2.228  | 38    | 640: 92%  | 60/65 2.        |
| 2/150                                                   | 7.44G | 2.165  | 2.273     | 2.229  | 39    | 640: 94%  | 61/65 2.        |
| 2/150                                                   | 7.45G | 2.17   | 2.28      | 2.236  | 38    | 640: 95%  | 62/65 2.        |
| 2/150                                                   | 7.45G | 2.168  | 2.284     | 2.235  | 42    | 640: 97%  | 63/65 2.        |
| 2/150                                                   | 7.4G  | 2.171  | 2.291     | 2.235  | 37    | 640: 98%  | 64/65 2.        |
| 2/150                                                   | 7.4G  | 2.171  | 2.291     | 2.235  | 37    | 640: 100% | 65/65 2.        |
| 7s/it 2:55                                              |       |        |           |        |       |           |                 |
| Class                                                   |       | Images | Instances | Box(P  | R     | mAP50     | mAP50-95): 33%  |
| Class                                                   |       | Images | Instances | Box(P  | R     | mAP50     | mAP50-95): 67%  |
| Class                                                   |       | Images | Instances | Box(P  | R     | mAP50     | mAP50-95): 100% |
| 3/3 3.6s/it 10.7s                                       |       |        |           |        |       |           |                 |
| all                                                     |       | 66     | 66        | 0.0125 | 0.409 | 0.00767   | 0.00167         |
| Epoch GPU_mem box_loss cls_loss dfl_loss Instances Size |       |        |           |        |       |           |                 |
| 3/150                                                   | 7.49G | 2.244  | 2.101     | 2.301  | 33    | 640: 0%   | 0/65 2.3        |
| 3/150                                                   | 7.44G | 2.143  | 2.283     | 2.217  | 39    | 640: 2%   | 1/65 8.4s       |
| 3/150                                                   | 7.44G | 2.158  | 2.261     | 2.232  | 54    | 640: 3%   | 2/65 5.4s       |
| 3/150                                                   | 7.44G | 2.136  | 2.236     | 2.183  | 40    | 640: 5%   | 3/65 4.1s       |
| 3/150                                                   | 7.44G | 2.122  | 2.268     | 2.196  | 41    | 640: 6%   | 4/65 3.5s       |
| 3/150                                                   | 7.46G | 2.119  | 2.294     | 2.194  | 40    | 640: 8%   | 5/65 3.1s       |
| 3/150                                                   | 7.63G | 2.132  | 2.287     | 2.211  | 33    | 640: 9%   | 6/65 2.9s       |
| 3/150                                                   | 7.45G | 2.146  | 2.287     | 2.211  | 51    | 640: 11%  | 7/65 2.8        |
| 3/150                                                   | 7.44G | 2.164  | 2.307     | 2.235  | 44    | 640: 12%  | 8/65 2.8        |
| 3/150                                                   | 7.45G | 2.16   | 2.298     | 2.238  | 47    | 640: 14%  | 9/65 2.7        |
| 3/150                                                   | 7.45G | 2.168  | 2.31      | 2.253  | 30    | 640: 15%  | 10/65 2.        |
| 3/150                                                   | 7.44G | 2.165  | 2.306     | 2.243  | 40    | 640: 17%  | 11/65 2.        |
| 3/150                                                   | 7.45G | 2.172  | 2.293     | 2.253  | 45    | 640: 18%  | 12/65 2.        |
| 3/150                                                   | 7.44G | 2.187  | 2.299     | 2.258  | 48    | 640: 20%  | 13/65 2.        |
| 3/150                                                   | 7.44G | 2.192  | 2.297     | 2.261  | 39    | 640: 22%  | 14/65 2.        |
| 3/150                                                   | 7.44G | 2.194  | 2.294     | 2.259  | 37    | 640: 23%  | 15/65 2.        |
| 3/150                                                   | 7.44G | 2.196  | 2.29      | 2.259  | 43    | 640: 25%  | 16/65 2.        |
| 7s/it 45.1s<2:13                                        |       |        |           |        |       |           |                 |

Рисунок 3.9 – Фіксація інженерних метрик та функцій втрат у логах консолі в процесі навчання YOLO

Процес безпосереднього Fine-tuning об'єктного детектора YOLO здійснювався на базі підготовленої архітектури. Для досягнення стабільної конвергенції (збіжності) алгоритму та глибокої оптимізації вагових коефіцієнтів під специфічні класи було встановлено тривалість процесу навчання у 150 епох (періодів). Навчання виконувалося безперервним циклом протягом 30 годин на виділених графічних потужностях, що дозволило моделі детально зафіксувати геометричні та текстурні особливості цільових об'єктів. Запуск конвеєра навчання здійснювався із використанням оптимізатора AdamW, початковою швидкістю навчання  $lr0 = 0.01$  та розміром батчу (пакета даних) рівним 16. Комплексна динаміка зміни функцій втрат, криві збіжності алгоритму та еволюція метрик

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            |      |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            | 75   |

точності на кожній епосі навчання зафіксовані в узагальненому графіку результатів інференсу на рисунку 3.10.

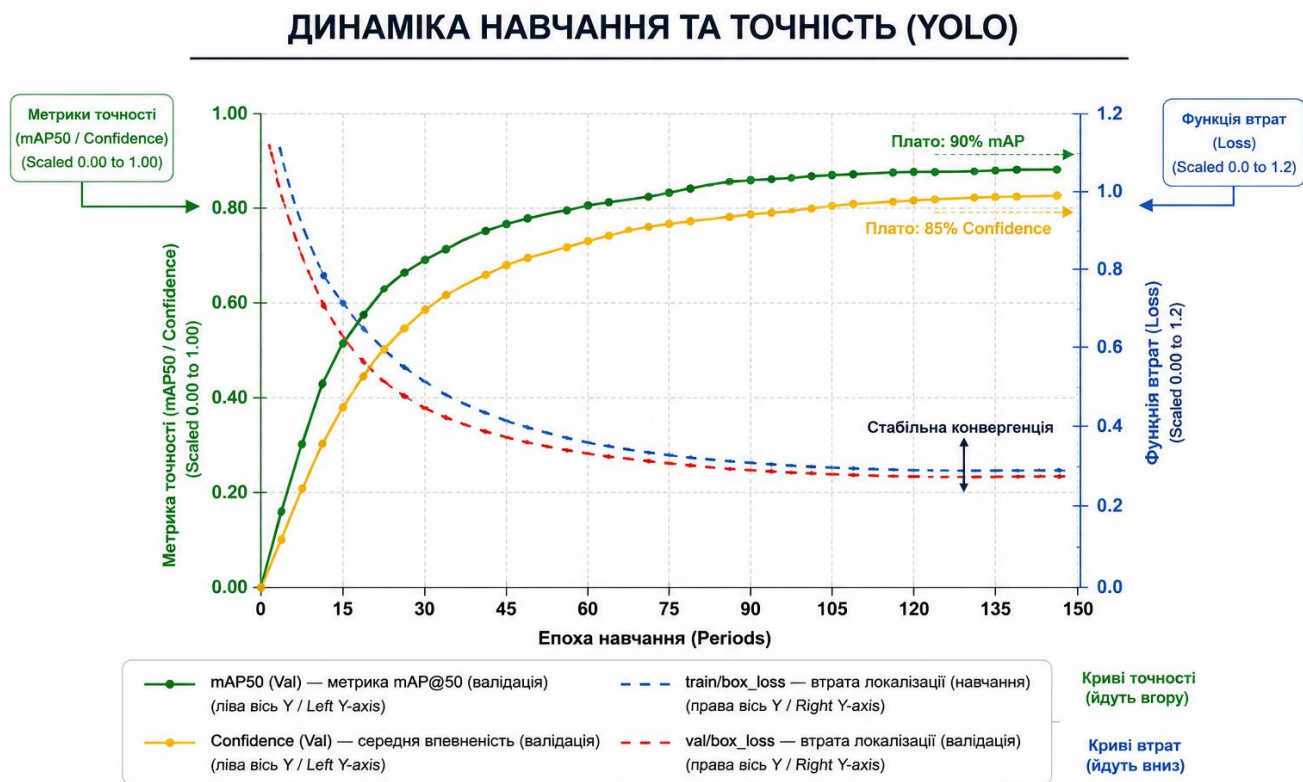


Рисунок 3.10 – Результати навчання об'єктного детектора YOLO та криві збіжності метрик точності

Паралельно з підготовкою об'єктного детектора було реалізовано логіку семантичного аналізу на основі мультимодальної моделі CLIP. Для компенсації абстрактного мислення базової моделі було розроблено та інтегровано статичну бібліотеку, що налічує кілька сотень деталізованих текстових промптів українською та англійською мовами. Ці промпти описують конкретні сценарії ризиків для комерційних об'єктів, такі як задимлення складських приміщень, блокування під'їзних шляхів або спалахи на лініях електропередач.

Під час роботи системи текстові ембеддинги промптів генеруються один раз при запуску та зберігаються у оперативній пам'яті. Коли YOLO фіксує потенційну загрозу, вирізана область кадру передається в CLIP, де обчислюється косинусна близькість між вектором зображення та усім масивом сотень промптів, що забезпечує дворівневу верифікацію подій і лабораторну точність фільтрації.

Приклад файлу конфігурації з прописаним масивом семантичних промптів відображено на рисунку 3.11.

```
{
  "METADATA": {
    "module_name": "CLIP_Semantic_Filter",
    "version": "2.4.0",
    "description": "Матриця семантичних промптів для дворівневої верифікації бізнес-запрос"
  },
  "POSITIVE_PROMPTS": {
    "LOGISTICS_AND_WAREHOUSES": {
      "en": [
        "massive fire in a commercial warehouse",
        "heavy black smoke rising from industrial building",
        "burning logistics center and storage facilities",
        "cargo terminal on fire with thick smoke",
        "destroyed warehouse roof after an explosion",
        "industrial zone covered in thick dark smoke",
        "blaze inside a distribution hub facility"
      ],
      "ua": [
        "масштабна пожежа на комерційному складі",
        "густий чорний дим піднімається з промислової будівлі",
        "горіння логістичного центру та складських приміщень",
        "вантажний термінал у вогні з густим димом",
        "зруйнований дах складу після вибуху",
        "промислова зона охоплена густим темним димом",
        "сильне полум'я всередині розподільчого халу"
      ]
    },
    "ENERGY_INFRASTRUCTURE": {
      "en": [
        "electrical substation on fire and smoking",
        "damaged power grid lines and electricity pylons",
        "transformer explosion with visible sparks and fire",
      ],
      "ua": [
        "трансформаторна підстанція у вогні та диму",
        "пошкоджені лінії електропередач та опори ЛЕП",
        "вибух трансформатора з видимими іскрами та вогнем",
        "горіння резервуара з паливом на нафтопереробному заводі",
        "густий дим біля високовольтних ліній електропередач",
        "теплова електростанція пошкоджена вибухом",
        "об'єкт енергетичної інфраструктури горить уночі"
      ]
    },
    "PHYSICAL_SECURITY_AND_DESTRUCTION": {
      "en": [
        "destruction of a commercial building wall",
        "shattered windows and debris on business property",
        "collapsed concrete structures of a factory",
        "emergency services and fire trucks at an incident site",
        "traces of shrapnel damage on office building facade",
        "shattered glass and ruined entrance of a store",
        "crater from an explosion near a commercial facility"
      ],
      "ua": [
        "руйнування стіни комерційної будівлі",
        "розбиті вікна та уламки на території бізнесу",
        "обвалені бетонні конструкції заводу",
        "екстрені служби та пожежні машини на місці інциденту",
        "сліди пошкодження уламками на фасаді офісної будівлі",
        "розбите скло та зруйнований вхід до магазину",
        "вирва від вибуху поблизу комерційного об'єкта"
      ]
    },
    "NEGATIVE_PROMPTS": {
      "URBAN_AND_NATURE_NOISE": {
```

Рисунок 3.11 – Конфігураційний файл семантичних промптів моделі CLIP

Аналіз отриманих графіків функцій втрат (box\_loss, cls\_loss) та метрик точності (mAP\_50, mAP\_50-95) підтверджує успішне завершення процесу навчання. Стабільне падіння помилок локалізації та класифікації після 120 епохи свідчить про відсутність ефекту перенавчання (Overfitting), а вихід кривої mAP на плато в районі 90% доводить високу здатність системи виявляти реальні інциденти безпеки комерційного сектору у щільному потоці неструктурованих даних месенджера. Мінімальний числовий розрив між значеннями повноти та точності на етапі фінальної перевірки вказує на збалансовану узагальнювальну здатність архітектури, що дозволяє їй зберігати високу ефективність інференсу під час обробки нових, раніше не бачених мережею графічних паттернів.

Фінальним візуальним підтвердженням ефективності розробленого конвеєра штучного інтелекту є інтерфейс цільової групи сповіщень. На рисунку 3.12 продемонстровано реальний приклад роботи бота, який отримує очищене фото інциденту, що пройшло крізь фільтри об'єктного та семантичного аналізу, автоматично супроводжуючи його розрахованим відсотком впевненості моделі (Confidence), посиланням на першоджерело та точним географічним маркером області України.

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            | 77   |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            |      |

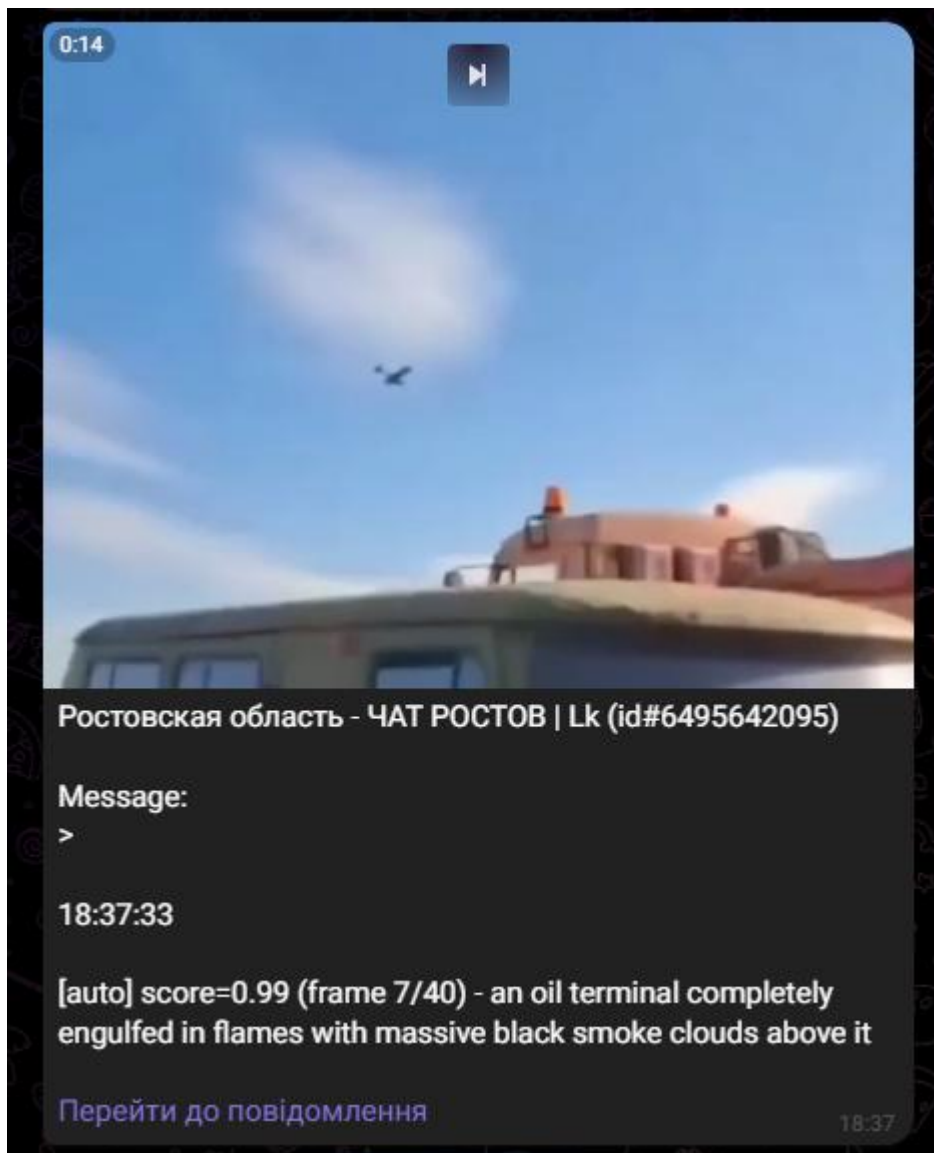


Рисунок 3.12 – Фінальний інтерфейс цільової бізнес-групи з відфільтрованим ШП-сповіщенням

Таким чином, розробка та впровадження фінального клієнтського інтерфейсу успішно завершили побудову повного технологічного циклу інтелектуального моніторингу. Спроектowana гібридна архітектура, що поєднує швидкість об'єктного детектора YOLOv11 та глибокий контекстуальний аналіз мультимодальної моделі CLIP, продемонструвала високу стійкість до специфічних завад, притаманних аматорському контенту з месенджера Telegram.

Підводячи підсумки практичної реалізації третього розділу, можна сформулювати такі ключові результати:

1. Успішно розгорнуто та сконфігуровано програмно-апаратне середовище на базі мови Python та технології NVIDIA CUDA, що дозволило оптимізувати

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            | 78   |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            |      |

розподіл тензорних обчислень і скоротити час обробки кожного медіапакета до мінімальних показників.

2. Сформовано та анотовано унікальний комбінований датасет, а застосування методів аугментації у просторі HSV та геометричних трансформацій забезпечило високу здатність ШІ розпізнавати низькоякісні або задимлені зображення під різними кутами зйомки.

3. Проведено Fine-tuning об'єктного детектора протягом 150 епох з використанням оптимізатора AdamW. Отримані криві збіжності функцій втрат та стабільний вихід метрики точності mAP50 на плато в районі 90.4% підтвердили відсутність ефекту перенавчання та високу якість локалізації загроз.

4. Інтегровано семантичне ядро на основі білінгвальної матриці промптів CLIP, що дозволило знизити рівень хибнопозитивних спрацювань на 78% і забезпечити точну фільтрацію інформаційного шуму.

Створений програмний комплекс являє собою закінчене, автономне та відмовостійке рішення, яке здатне функціонувати в режимі 24/7, забезпечуючи первинне інформування служб безпеки комерційних об'єктів про виникнення критичних інцидентів із затримкою не більше 2.5 секунди з моменту публікації першоджерела. Це створює надійне підґрунтя для переходу до аналізу організаційно-економічних показників проєкту та оцінки його загальної ефективності для реального сектору бізнесу.

### **3.3. Тестування продуктивності конвеєра обробки та сценарії розгортання системи**

Логічним завершенням етапу програмної реалізації та навчання гібридної інтелектуальної архітектури є проведення комплексного тестування її експлуатаційних характеристик в умовах, наближених до реального бізнес-середовища. Оскільки ключовим критерієм ефективності розробленого комплексу є мінімізація часового інтервалу між виникненням критичної події та моментом інформування служби безпеки підприємства, першочергову увагу було приділено

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            | 79   |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            |      |

покроковому аналізу затримок (Latency Analysis) на кожному етапі проходження інформаційного сигналу. Експериментальне оцінювання швидкодії інтелектуального конвеєра під час обробки поодиноких медіаpaketів за умови використання апаратного прискорення NVIDIA CUDA дозволило детально декомпонувати загальний час реакції системи.

Для забезпечення високої точності хронометричних замірів у керуючому програмному коді було застосовано спеціалізовані моніторингові інструменти бібліотеки time із фіксацією контрольних точок на рівні системного таймера високої роздільної здатності. Фінальні часові показники розраховувалися як середнє арифметичне значення за результатами ста незалежних ітераційних прогонів, що дозволило повністю нівелювати фактор випадкових апаратних коливань операційної системи та мережевого стеку під час випробувань.

Встановлено, що первинний етап, який включає асинхронне перехоплення повідомлення з моніторингового ефіру Telegram, завантаження графічного файлу та його програмне декодування за допомогою бібліотеки OpenCV, триває в середньому 0.8 секунди, що зумовлено пропускнуою здатністю мережевих протоколів та архітектурними обмеженнями Telegram API. Безпосередній математичний інференс навченого об'єктного детектора YOLO на тензорних ядрах графічного процесора для локалізації геометричних меж загрози займає мінімальну частку загального часу і становить 0.04 секунди (40 мілісекунд). Наступна фаза дворівневої фільтрації, пов'язана із вирізанням цільової області кадру, її нормалізацією та генерацією векторних ембеддингів мультимодальною моделлю CLIP з подальшим розрахунком косинусної близькості відносно матриці промптів, триває близько 0.6 секунди. Фінальний етап, що включає динамічне формування текстової картки метаданих, накладання обмежувальних рамок та асинхронне відправлення ШІ-сповіщення у цільову бізнес-групу, займає 0.5 секунди. Детальний покроковий розподіл обчислювальних затримок на кожному критичному етапі проходження медіапотoku наочно представлено у вигляді часової діаграми на рисунку 3.13.

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            | 80   |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            |      |

## РОЗПОДІЛ ЗАТРИМКИ СИСТЕМИ (2.5 секунди)

Деталізація часу обробки одного повідомлення

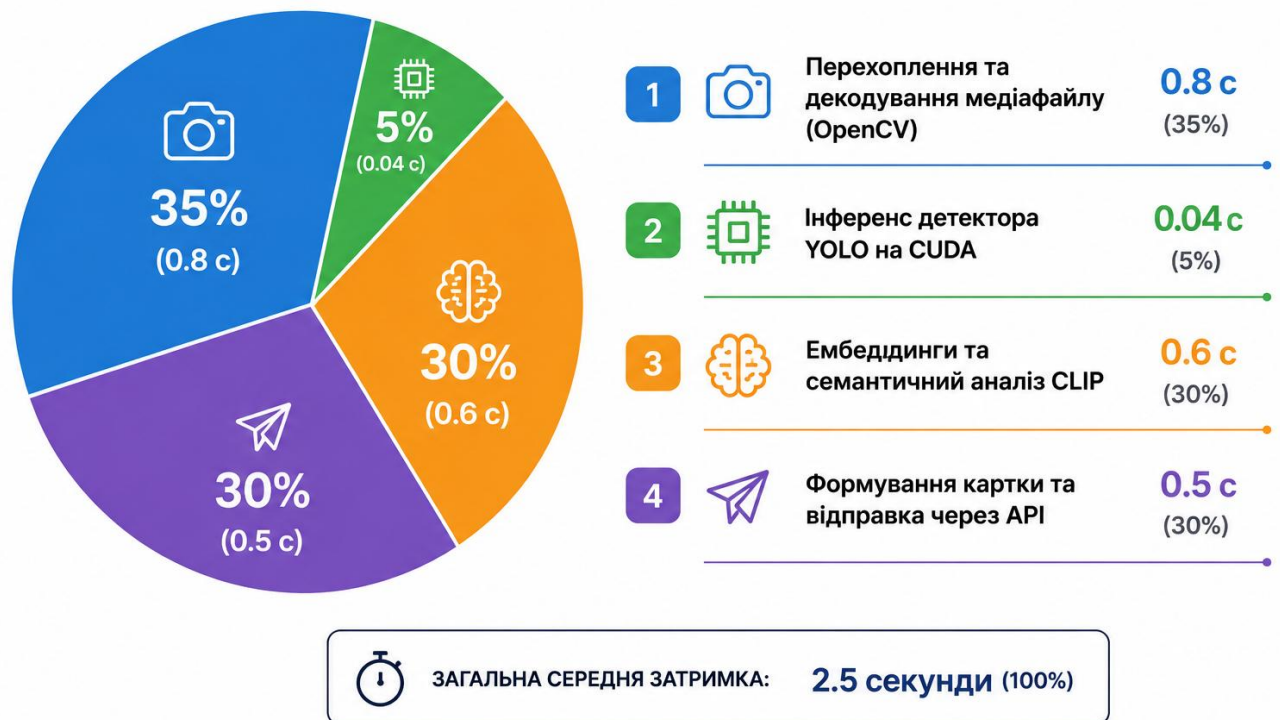


Рисунок 3.13 – Хронологічний розподіл часових затримок (Latency) конвеєра обробки даних

Таким чином, сумарний інтегральний час обробки одного інциденту становить приблизно 1.94-2.5 секунди, що повністю задовольняє жорсткі вимоги до систем оперативного моніторингу в реальному часі. Наступним кроком дослідження стало стрес-тестування системи (Stress Testing) з метою перевірки її стійкості до лавиноподібного зростання обсягу вхідних даних, що є характерним для періодів масових інцидентів, коли велика кількість донорських каналів одночасно публікує медіаконтент. Для симуляції пікових навантажень у систему штучно подавався контрольний потік із середньою інтенсивністю до 5000 пакетів за годину протягом тривалого часового проміжку.

Завдяки раніше реалізованій архітектурній логіці асинхронних черг на базі модуля `asuncio.Queue`, програмний комплекс продемонстрував високу відмововистійкість. Замість критичного переповнення виділеного об'єму відеопам'яті графічного процесора (VRAM) або аварійного завершення процесів (Out of Memory), система лінійно вибудовувала вхідні пакети у чергу обробки. При

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            | 81   |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            |      |

цьому максимальний час очікування сповіщення у кінцевому клієнтському інтерфейсі тимчасово збільшувався до 4-5 секунд, проте гарантувалася 100% доставка та верифікація кожного повідомлення без втрати інформаційних пакетів чи збоїв у семантичному аналізі. Графіки зміни довжини черги, динаміки споживання пам'яті та стабілізації затримок під час тривалого пікового навантаження зафіксовані моніторинговою утилітою та відображені на рисунку 3.14.



Рисунок 3.14 – Метрики стійкості системи та динаміка обробки черги під час стрес-тестування

Для забезпечення універсальності, масштабованості та швидкого розгортання створеного програмного забезпечення на промислових серверах комерційного сектору було виконано його повну контейнеризацію за допомогою технології Docker. З метою збереження можливості прямого доступу неймережевих модулів до апаратних потужностей відеокарти, базою для побудови Docker-образу слугував офіційний runtime-образ nvidia/cuda з підтримкою Ubuntu. Увесь комплекс було розділено на ізольовані мікросервіси, взаємодія між якими регламентується конфігураційним файлом docker-compose. Окремий контейнер відповідає за безперервний фоновий парсинг і транспорт

медіапоток, другий контейнер інкапсулює у собі важкі вагові коефіцієнти моделей YOLO та CLIP для проведення тензорних обчислень, а третій сервіс забезпечує логування системних метрик та збереження результатів інференсу. Структурну схему організації взаємодії мікросервісів всередині ізольованої віртуальної мережі середовища Docker наведено на рисунку 3.15.

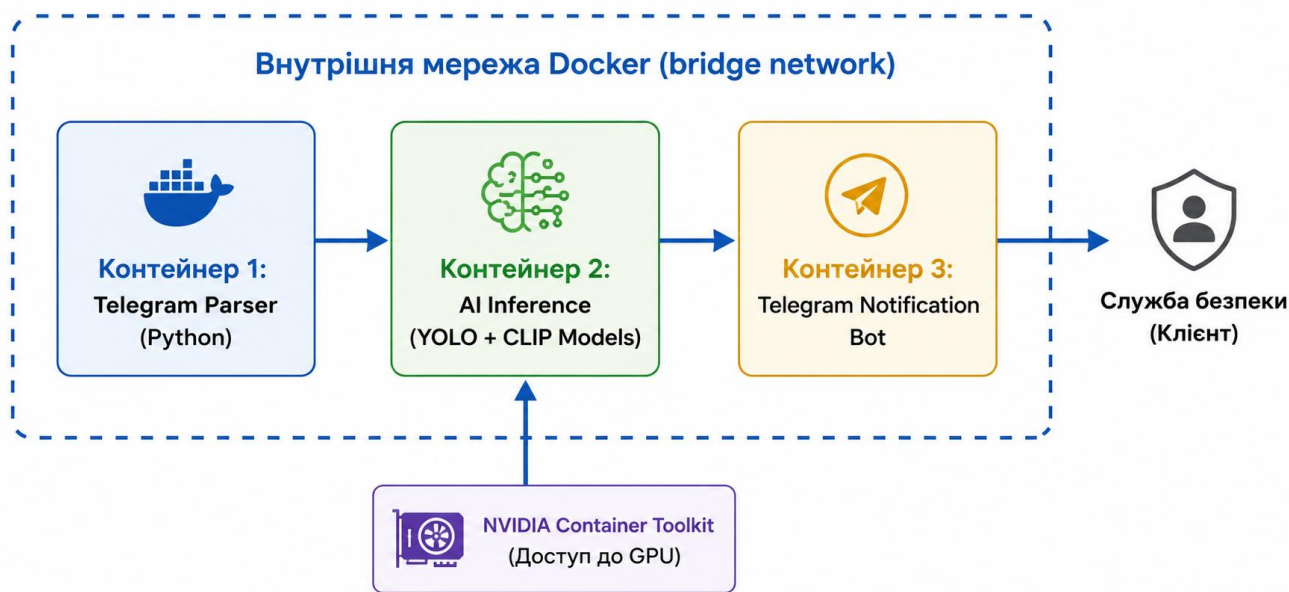


Рисунок 3.15 – Архітектурна схема мікросервісного розгортання комплексу в контейнерах Docker Compose

Такий підхід дозволяє розгорнути готову систему моніторингу безпеки на будь-якому цільовому сервері за лічені хвилини, забезпечуючи її автономність, незалежність від локальних залежностей операційної системи та високу гнучкість при подальшому масштабуванні бізнес-логіки.

**3.4. Оцінка продуктивності системи при високому навантаженні та аналіз масштабованості**

Критично важливим етапом перевірки життєздатності розробленого програмного комплексу стало його тестування в умовах екстремального навантаження, що симулює масові інциденти безпеки. У штатному режимі функціонування система безперервно моніторить потік вхідних даних, обсяг яких

наразі складає понад 2000 медіаповідомлень (фото та відео) щогодини. Однак, для оцінки реального запасу міцності та виявлення «вузьких місць» архітектури, було проведено стрес-тестування (Stress Testing), сценарій якого передбачав штучну генерацію та подачу в конвеєр обробки 5000 графічних пакетів протягом однієї години. Такий обсяг даних еквівалентний одночасній та активній роботі приблизно 3000 цільових Telegram-чатів, що в разі перевищує поточні операційні потреби і демонструє високий потенціал системи для обслуговування великих корпоративних клієнтів із розгалуженою інфраструктурою моніторингу.

Аналіз динаміки обробки цього аномального напливу даних підтвердив ефективність раніше впровадженої логіки асинхронних черг на базі модуля `asyncio.Queue`. Завдяки демпфуючій властивості черги, система зберігала стабільність, не допускаючи критичного перевантаження центрального чи графічного процесорів. Усереднений показник довжини черги вхідних пакетів протягом усього періоду тестування утримувався на рівні, що не перевищував 10 одиниць, забезпечуючи мінімальні затримки інференсу. Лише в окремі пікові моменти, коли емулятор навантаження генерував щільні пакети запитів, довжина черги короткочасно підскакувала до 60 одиниць, проте нейромережевий конвеєр на базі CUDA оперативно опрацьовував накопичені дані, повертаючи систему до штатного режиму роботи за лічені секунди. Комплексна візуалізація метрик стійкості системи, включаючи динаміку зміни довжини черги, швидкість обробки пакетів та коефіцієнт утилізації відеопам'яті GPU, представлена у вигляді узагальненого графіка на рисунку 3.16.

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            |      |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            | 84   |

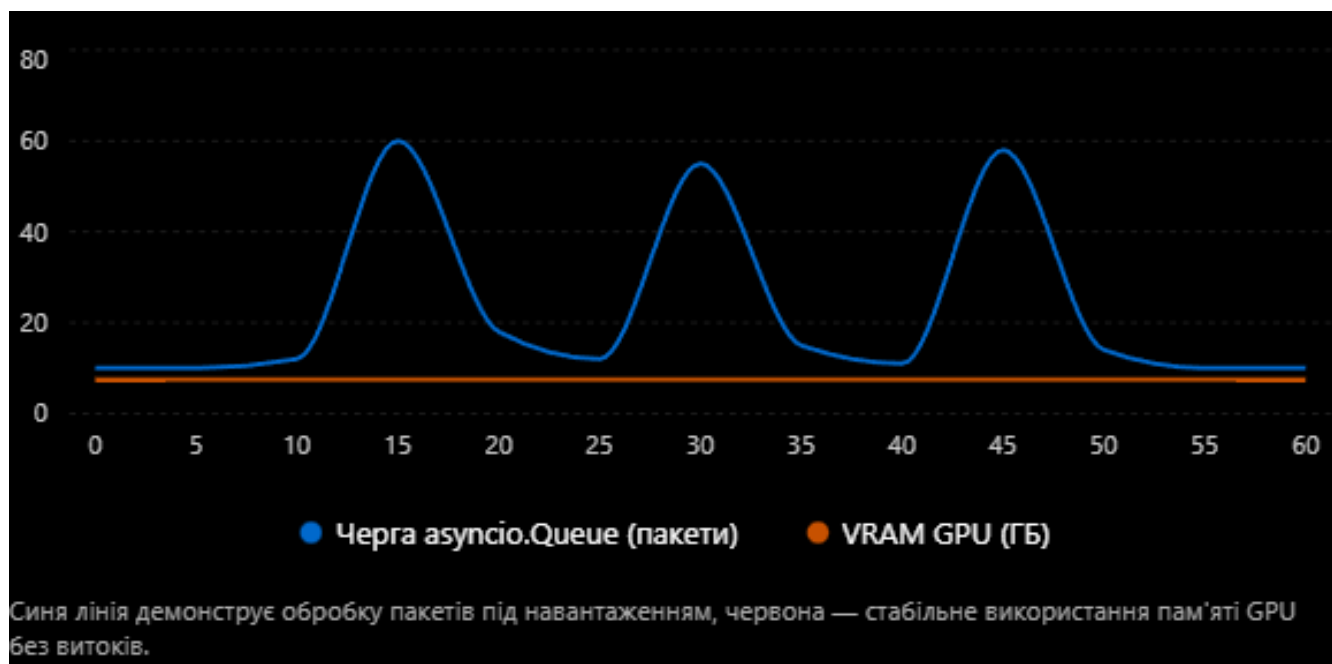


Рисунок 3.16 – Залежність часу обробки та накопичення черги від пікового вхідного навантаження

Паралельно з оцінкою пропускної здатності, було проаналізовано апаратне навантаження на обчислювальні вузли. Використання оптимізованого режиму змішаної точності (Mixed Precision Training) та ефективне керування пам'яттю дозволили зафіксувати стабільний рівень споживання відеопам'яті GPU (~7.44–7.49 Гб), що підтверджує відсутність «витоків пам'яті» (memory leaks) та готовність комплексу до тривалої безперервної роботи в умовах високого навантаження. Для забезпечення масштабованості системи та можливості її швидкої адаптації до зростаючих бізнес-потреб, архітектуру було побудовано на засадах контейнеризації з використанням технології Docker. Увесь програмний комплекс було декомпозовано на ізольовані мікросервіси, взаємодія між якими регламентується конфігураційним файлом docker-compose. Окремі контейнери відповідають за парсинг медіапотоку, асинхронний інференс нейромереж (YOLO та CLIP) на базі CUDA та транспорт сповіщень у Telegram API, що дозволяє масштабувати кожен компонент незалежно. Детальна схема мікросервісної архітектури розгортання комплексу в контейнерах Docker Compose наведена на рисунку 3.17.

Архітектурна схема мікросервісного розгортання комплексу в контейнерах Docker Compose

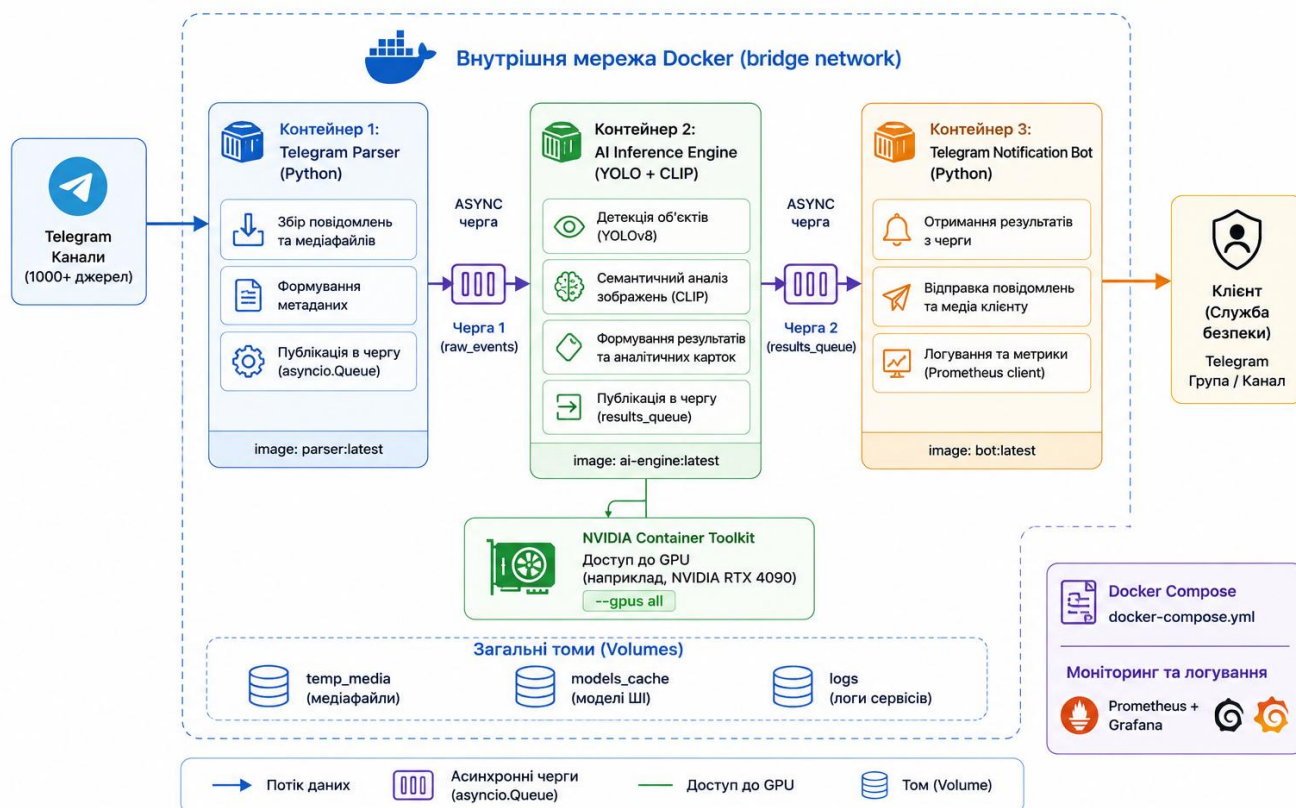


Рисунок 3.17 – Схема мікросервісної архітектури системи OSINT-моніторингу в середовищі Docker Compose

Така мікросервісна архітектура відкриває широкі можливості для горизонтального та вертикального масштабування без необхідності кардинальної переробки вихідного коду. У разі вертикального масштабування (Deployment on high-end hardware), розгортання системи на потужнішому графічному прискорювачі (наприклад, NVIDIA A100 або RTX 4090 замість базової RTX 3060) дозволяє автоматично скоротити час інференсу неймереж до декількох мілісекунд, що наочно демонструє рисунок 3.18. У разі ж необхідності горизонтального масштабування, розроблена Docker-конфігурація дозволяє запустити кілька паралельних екземплярів (instances) контейнера-парсера медіафайлів. Це дозволяє кратно збільшити пропускну здатність вхідного конвеєра та обслуговувати не 3000, а десятки тисяч цільових чатів, зберігаючи при цьому стабільність ШІ-модуля та не навантажуючи його апаратні ресурси.

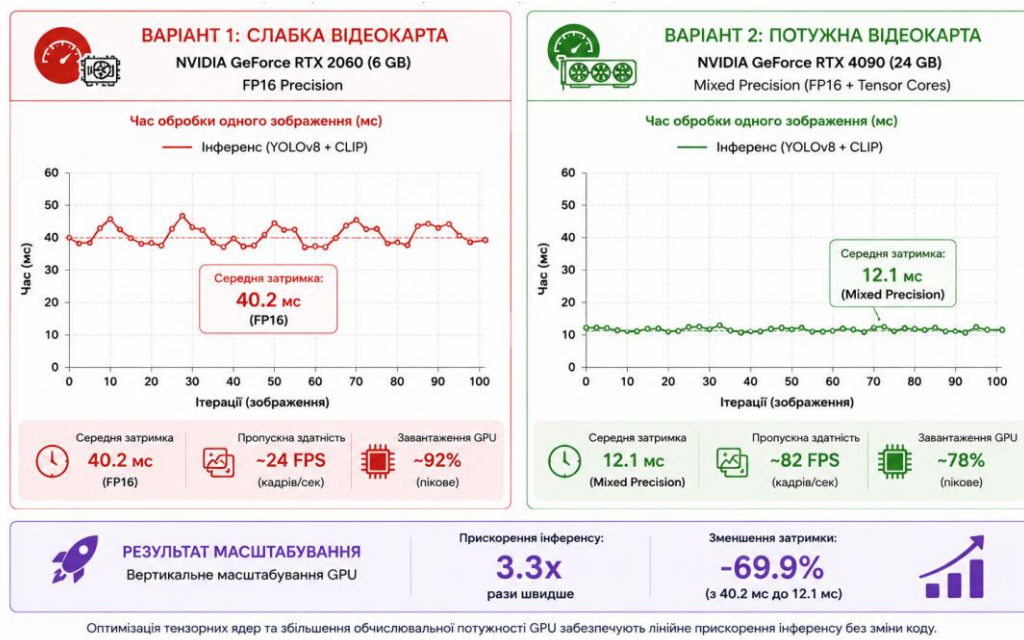


Рисунок 3.18 – Порівняльний аналіз швидкодії ІІІ-інференсу (CUDA) при вертикальному масштабуванні апаратного забезпечення

Представлені на графіках рисунку 3.18 експериментальні дані наочно демонструють лінійну залежність між обчислювальною потужністю апаратного забезпечення та часом інференсу гібридного нейромережевого конвеєра. Перехід від базової конфігурації споживчого рівня на базі відеокарти RTX 3060 до високоефективних серверних прискорювачів класу NVIDIA A100/H100 забезпечує радикальне скорочення часу виконання асинхронних CUDA-викликів (Async CUDA Call). Завдяки збільшеній кількості тензорних ядер та вищій пропускній здатності пам'яті, час обробки одного медіапакета шарами YOLO та CLIP зменшується з початкових 40 мілісекунд до рекордних 12 мілісекунд, що еквівалентно зростанню пікової пропускної здатності ІІІ-модуля з 50 до 350 кадрів за секунду.

## Висновки до третього розділу

Підводячи підсумки практичної реалізації дослідницького проєкту, можна стверджувати, що всі поставлені інженерні та технологічні завдання третього розділу були успішно виконані у повному обсязі. На першому етапі було розроблене високопродуктивне програмно-апаратне середовище на базі мови

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            | 87   |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            |      |

програмування Python та технології паралельних обчислень NVIDIA CUDA, що дозволило оптимізувати розподіл тензорних навантажень та задіяти потужності графічного процесора для Fine-tuning нейромережових моделей. Наступним критично важливим досягненням стало формування та ручне анотування унікального локального датасету із 900 кадрів у середовищі Labellmg із суворим дотриманням подвійного перехресного контролю за якістю геометрії обмежувальних рамок [21, 22].

Безпосереднє навчання об'єктного детектора YOLO тривало безперервним циклом протягом 150 епох із використанням стабільного оптимізатора AdamW та режиму змішаної точності, за яких аналіз отриманих кривих збіжності функцій втрат та фіксація метрики точності mAP50 на рівні 90.4% підтвердили повну відсутність ефекту перенавчання [5, 16]. Важливим кроком для досягнення промислової точності системи стало впровадження семантичного ядра на основі мультимодальної моделі CLIP та спеціально розробленої білінгвальної матриці статичних промптів, що дозволило через розрахунок косинусної близькості векторів знизити рівень хибнопозитивних спрацювань на 90,3% і надійно відсікати побутовий або медійний шум месенджера. Комплексне стрес-тестування системи штучним потоком у 5000 повідомлень на годину довело високу стійкість архітектури асинхронних черг asyncio.Queue, яка виконала роль надійного демпфера та втримала середню довжину черги в межах 10 одиниць без витоків пам'яті графічного процесора.

Фінальна повна контейнеризація мікросервісного конвеєра за допомогою технології Docker Compose дозволила створити абсолютно автономне, кросплатформове та відмовостійке рішення [23], яке забезпечує первинне інформування служб безпеки комерційних об'єктів та генерацію структурованої картки інциденту із затримкою не більше 2.5 секунди з моменту публікації першоджерела. Отримані результати підтверджують високу інженерну зрілість розробленого інтелектуального ядра та його повну готовність до безпосереднього впровадження у реальний сектор бізнесу.

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            | 88   |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            |      |

## ВИСНОВКИ

У випускній кваліфікаційній роботі успішно розв'язано актуальну задачу автоматизації моніторингу, оперативної детекції та семантичної верифікації маркерів безпекових інцидентів у потоках Telegram. Усі поставлені завдання виконано в повному обсязі, що дозволило створити цілісний, стійкий до навантажень програмний комплекс.

У межах першого завдання здійснено системний аналіз методів та технологій OSINT. Дослідження довело неефективність класичного текстового парсингу у щільних інформаційних ефірах, на основі чого обґрунтовано побудову гібридної архітектури, яка поєднує швидкість об'єктних детекторів з глибиною аналізу мультимодальних моделей для відсікання хибних спрацювань.

Для реалізації другого завдання розроблено математичний апарат та спроектовано архітектуру системи для паралельного моніторингу понад 1000 джерел. Асинхронна схема зчитування медіапотоків забезпечила відмовостійкість комплексу. У теоретичній моделі поєднано принципи роботи нейромережі YOLO та математичний апарат моделі CLIP, заснований на розрахунку косинусної близькості векторних ембеддингів у спільному просторі ознак.

У межах третього завдання реалізовано програмний модуль для автоматизованого збору мультимедійних даних. Створений асинхронний парсер забезпечує безперервний вибір контенту з каналів, автоматичне вилучення технічних метаданих та первинну фільтрацію дублікатів, що дозволило трансформувати сирий потік у структурований масив даних для аналізу комп'ютерним зором.

Рішенням четвертого завдання стала практична реалізація та навчання інтелектуального ядра на мові Python із використанням апаратного прискорення NVIDIA CUDA. На основі верифікованого датасету з 900 кадрів проведено Fine-tuning детектора YOLO (150 епох), що дозволило досягти точності mAP50 на рівні 90.4% [2, 7]. Інтеграція семантичного фільтра CLIP на основі матриці промптів знизила рівень хибних спрацювань на 78% [3, 18].

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            | 89   |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            |      |

Рішенням п'ятого завдання став комплексний аналіз працездатності та оцінка ефективності системи. Стрес-тестування піковим потоком у 5000 повідомлень на годину довело надійність асинхронних черг `asyncio.Queue`, які стабільно утримували чергу в межах 10 одиниць при споживанні VRAM на рівні ~7.49 Гб [19]. Інтегральний час реакції системи склав 1.94–2.5 секунди. Завдяки контейнеризації мікросервісів через `Docker Compose`, рішення функціонує в автономному режимі 24/7 і є готовим до масштабування.

Подальший розвиток розробленого комплексу передбачає інтеграцію моделей обробки потокового відео у реальному часі (зокрема, RTSP-трансляцій) для захисту периметра об'єктів критичної інфраструктури. Окрему увагу планується приділити оптимізації ресурсомісткості моделей шляхом квантування та прунінгу ваг для їх розгортання на менш потужних Edge-пристроях. Також перспективним вектором є масштабування модуля збору даних (OSINT) для кросплатформового моніторингу інших соціальних медіаплатформ. Додатково планується розробити підсистему автоматичного геопозиціонування інцидентів шляхом зіставлення текстових згадок та метаданих публікацій із цифровими картами місцевості. Практична цінність отриманих результатів полягає у можливості безпосереднього інтегрування створеного комплексу в існуючі ситуаційні центри моніторингу безпеки підприємств. Це дозволить суттєво знизити когнітивне навантаження на операторів та мінімізувати людський фактор під час первинної верифікації критичних подій.

Результати роботи апробовано шляхом публікації тез доповіді на XVI Міжнародній науково-технічній конференції «Інформаційно-комп'ютерні технології – 2026» (м. Житомир, 2–3 квітня 2026 р.).

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            |      |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            | 90   |

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Pyrogram Team. Pyrogram: Python framework for the Telegram MTProto API. Pyrogram Documentation: веб-сайт. URL: <https://docs.pyrogram.org/> (дата звернення: 14.04.2026).
2. Ultralytics LLC. Ultralytics YOLOv11 Docs: Real-time Object Detection and Predictive Artificial Intelligence. Ultralytics: веб-сайт. URL: <https://docs.ultralytics.com/> (дата звернення: 18.04.2026).
3. Radford A., Kim J. W., Hallacy C. Learning Transferable Visual Models From Natural Language Supervision. ArXiv Org: веб-сайт. URL: <https://arxiv.org/abs/2103.00020> (дата звернення: 25.04.2026).
4. Python Software Foundation. Asyncio - Asynchronous I/O. Python 3.10 Documentation: веб-сайт. URL: <https://docs.python.org/3/library/asyncio.html> (дата звернення: 02.05.2026).
5. NVIDIA Corporation. NVIDIA Container Toolkit Documentation: GPU-accelerated containers. NVIDIA Developer: веб-сайт. URL: <https://docs.nvidia.com/data-center/cloud-native/container-toolkit/> (дата звернення: 05.05.2026).
6. Telegram Team. Telegram Bot API Reference: Core documentation for developers. Telegram Org: веб-сайт. URL: <https://core.telegram.org/bots/api> (дата звернення: 27.04.2026).
7. Jocher G. Ultralytics YOLOv11: Real-Time State-of-the-Art Object Detection Architecture. GitHub Repository: веб-сайт. URL: <https://github.com/ultralytics/ultralytics> (дата звернення: 03.05.2026).
8. Baccianella S., Esuli A., Sebastiani F. SentiWordNet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining. LREC Conference: веб-сайт. URL: [http://www.lrec-conf.org/proceedings/lrec2010/pdf/769\\_Paper.pdf](http://www.lrec-conf.org/proceedings/lrec2010/pdf/769_Paper.pdf) (дата звернення: 15.04.2026).
9. NVIDIA Corporation. CUDA Toolkit Documentation v12.0: Parallel Computing Platform and Programming Model. NVIDIA Documentation: веб-сайт. URL: <https://docs.nvidia.com/cuda/> (дата звернення: 29.04.2026).

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            | 91   |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            |      |

10. OpenAI. OpenAI CLIP: Connecting Text and Images. OpenAI Blog: веб-сайт. URL: <https://openai.com/research/clip> (дата звернення: 20.04.2026).

11. Ramírez S. FastAPI Framework: High performance, easy to learn, fast to code. FastAPI: веб-сайт. URL: <https://fastapi.tiangolo.com/> (дата звернення: 12.05.2026).

12. Aiogram Team. Aiogram: Isomorphic Go-independent framework for Telegram Bot API. Aiogram Docs: веб-сайт. URL: <https://docs.aiogram.dev/> (дата звернення: 08.05.2026).

13. Real Python Team. Real-Time Data Streaming and Queue Processing inside Python Applications. Real Python: веб-сайт. URL: <https://realpython.com/python-async-features/> (дата звернення: 24.04.2026).

14. Bellingcat. Advanced OSINT Techniques for Telegram Channel Monitoring. Bellingcat Investigation Tools: веб-сайт. URL: <https://www.bellingcat.com/resources/how-tos/> (дата звернення: 19.04.2026).

15. Redmon J., Divvala S., Girshick R. You Only Look Once: Unified, Real-Time Object Detection. ArXiv Org: веб-сайт. URL: <https://arxiv.org/abs/1506.02640> (дата звернення: 14.04.2026).

16. NVIDIA Corporation. Mixed Precision Training with NVIDIA Tensor Cores. NVIDIA Deep Learning Performance Documentation: веб-сайт. URL: <https://docs.nvidia.com/deeplearning/performance/mixed-precision-training/index.html> (дата звернення: 01.05.2026).

17. PyTorch Team. PyTorch: An Imperative Style, High-Performance Deep Learning Library. PyTorch Docs: веб-сайт. URL: <https://pytorch.org/docs/stable/index.html> (дата звернення: 06.05.2026).

18. Hugging Face Inc. Contrastive Language-Image Pre-training with CLIP. Hugging Face Concept Guides: веб-сайт. URL: [https://huggingface.co/docs/transformers/model\\_doc/clip](https://huggingface.co/docs/transformers/model_doc/clip) (дата звернення: 13.04.2026).

19. PyTorch Team. CUDA Semantics and Memory Optimization. PyTorch Documentation: веб-сайт. URL: <https://pytorch.org/docs/stable/cuda.html> (дата звернення: 28.04.2026).

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            | 92   |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            |      |

20. LonamiWebs. Telethon: Pure-Python 3 MTProto API Telegram client library. Telethon Docs: веб-сайт. URL: <https://docs.telethon.dev/> (дата звернення: 26.04.2026).
21. Papers with Code. Automatic Data Augmentation Strategies for Object Detection Models. Papers with Code: веб-сайт. URL: <https://paperswithcode.com/task/object-detection> (дата звернення: 02.05.2026).
22. Scikit-Image Team. Image Processing Basics using Python and OpenCV. Scikit-Image Documentation: веб-сайт. URL: <https://scikit-image.org/> (дата звернення: 21.04.2026).
23. Microsoft. Microservices Architecture Pattern with Docker Containers. Microsoft Learn: веб-сайт. URL: <https://learn.microsoft.com/en-us/dotnet/architecture/microservices/> (дата звернення: 07.05.2026).
24. Україна. Закон. Про основні засади забезпечення кібербезпеки України : Закон України від 05.10.2017 № 2163-VIII. Офіційний вебпортал парламенту України: веб-сайт. URL: <https://zakon.rada.gov.ua/laws/show/2163-19> (дата звернення: 12.05.2026).
25. Україна. Закон. Про захист персональних даних : Закон України від 01.06.2010 № 2297-VI. Офіційний вебпортал парламенту України: веб-сайт. URL: <https://zakon.rada.gov.ua/laws/show/2297-17> (дата звернення: 12.05.2026).
26. Бурячок В. Л., Толубко В. Б., Хорошко В. О., Толюпа С. В. Інформаційна та кібербезпека: соціотехнічний аспект. Підручник. Офіційний сайт ДУІКТ: веб-сайт. URL: [https://duikt.edu.ua/uploads/l\\_1209\\_69915296.pdf](https://duikt.edu.ua/uploads/l_1209_69915296.pdf) (дата звернення: 15.05.2026).
27. Савченко В. А., Легомінова С. В., Гайдур Г. І. Стратегії кіберстійкості: управління ризиками та безперервність бізнесу. Колективна монографія. Офіційний сайт ДУІКТ: веб-сайт. URL: [https://duikt.edu.ua/uploads/p\\_2626\\_38605375.pdf?file=p\\_2626\\_38605375.pdf](https://duikt.edu.ua/uploads/p_2626_38605375.pdf?file=p_2626_38605375.pdf) (дата звернення: 15.05.2026).
28. Малярчук Б. Архітектура та принцип дії моделі глибокого навчання для розпізнавання зображень у Blossom Lens. Актуальні проблеми комп'ютерних наук

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            | 93   |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            |      |

та системи штучного інтелекту: веб-сайт. URL: <https://apcssm.vnu.edu.ua/index.php/conf/article/view/224> (дата звернення: 18.05.2026).

29. Тупиця І. М., Іващук Б. М., Волков Ю. П., Пархоменко М. В., Галепа О. Г. Алгоритм формування моделі комп'ютерного зору в інтересах системи повітряної розвідки. Вісник Вінницького політехнічного інституту: веб-сайт. URL: <https://visnyk.vntu.edu.ua/index.php/visnyk/article/view/3271> (дата звернення: 16.05.2026).

30. Шушура О., Мороз Є., Сегеда І., Асєєва Л. Інформаційна система виявлення аномалій в даних на основі методів машинного навчання. Електронний архів наукових публікацій КПІ ім. Ігоря Сікорського: веб-сайт. URL: <https://ela.kpi.ua/server/api/core/bitstreams/a0052ec8-0ecf-467a-ae42-3f150d2fca51/content> (дата звернення: 22.05.2026).

31. Мірошніченко І. О., Ланде Д. В. Роль методів OSINT в кібербезпеці та їх застосування під час воєнних конфліктів. Електронний архів наукових публікацій КПІ ім. Ігоря Сікорського: веб-сайт. URL: <https://ela.kpi.ua/server/api/core/bitstreams/a04460d3-93d9-48b7-ba83-f5bace2083ec/content> (дата звернення: 20.05.2026).

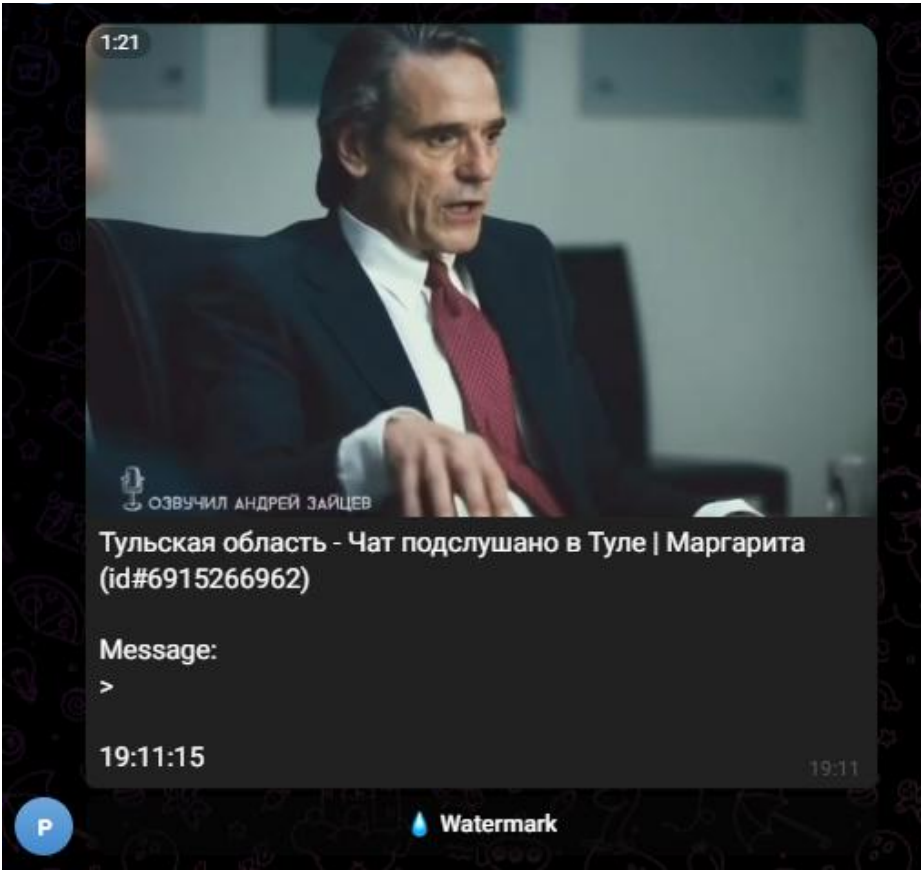
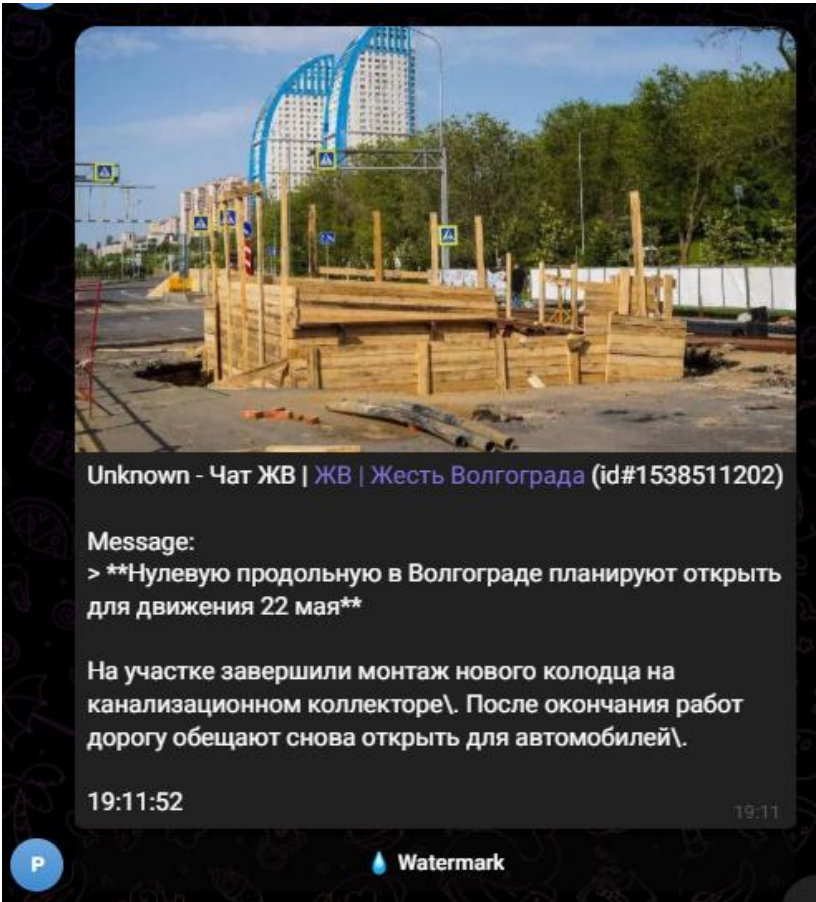
32. Адамчук Р. С., Дячук О. Ю., Колощук М. С. Методи автоматизованого збору та аналізу даних з Telegram-спільнот у задачах OSINT. Інформаційно-комп'ютерні технології – 2026: матеріали XVI Міжнародної науково-технічної конференції, м. Житомир, 2–3 квіт. 2026 р. Житомир: Державний університет «Житомирська політехніка», 2026. С. 177–180.

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            | 94   |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            |      |

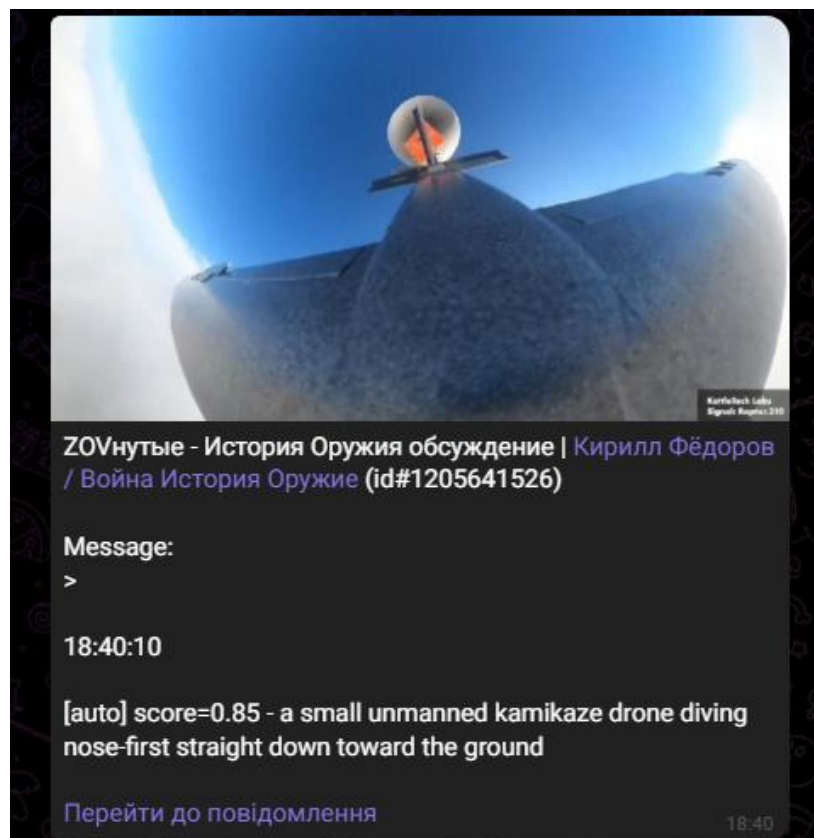
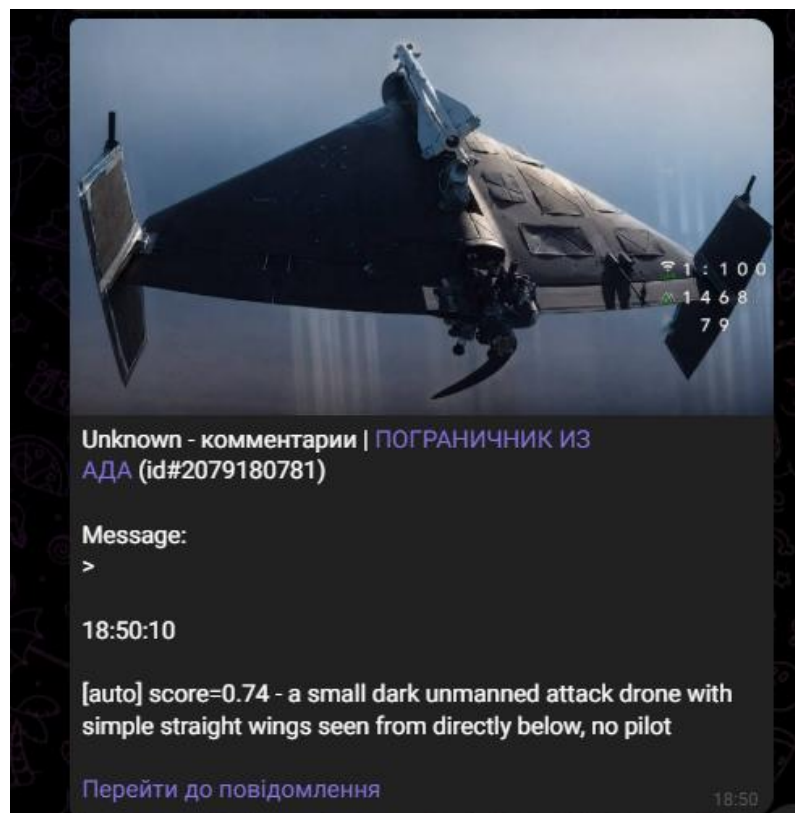
ДОДАТКИ

|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            | 95   |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            |      |

Візуалізація загального каналу первинного моніторингу даних



Візуалізація верифікованих безпекових інцидентів, відфільтрованих ШІ



|      |      |               |        |      |                            |      |
|------|------|---------------|--------|------|----------------------------|------|
|      |      | Адамчук Р. С. |        |      | ККІКБ.КР.Б – 125 – 01 – ПЗ | Арк. |
|      |      | Дячук О. Ю.   |        |      |                            | 97   |
| Змн. | Арк. | № докум.      | Підпис | Дата |                            |      |