ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Кваліфікаційна наукова
праця на правах рукопису

РОМАНЕЦЬ ВЛАДИСЛАВ ОЛЕГОВИЧ

УДК 004.9+681.004

ДИСЕРТАЦІЯ
ІНТЕЛЕКТУАЛЬНА ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ АНАЛІЗУ ВІДЕОПОТОКУ ДОРОЖНЬОГО РУХУ

Спеціальність 126 – «Інформаційні системи та технології»
Галузь знань 12 – «Інформаційні технології»

Подається на здобуття ступеня доктора філософії

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

_____ В. О. Романець

Науковий керівник: Бісікало Олег Володимирович,
заслужений діяч науки і техніки України,
доктор технічних наук, професор

Вінниця – 2026

АНОТАЦІЯ

Романець В. О. Інтелектуальна інформаційна технологія аналізу відеопотоку дорожнього руху – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 126 – «Інформаційні системи та технології» галузі знань 12 – «Інформаційні технології», - Вінницький національний технічний університет (ВНТУ), Міністерство освіти і науки України, Вінниця, 2026.

Дисертаційна робота присвячена вирішенню актуальної науково-прикладної задачі підвищення ефективності інтелектуального аналізу відеопотоку дорожнього руху на основі розробки та інтеграції удосконаленої моделі детекції об’єктів та методів багатооб’єктного трекінгу й виявлення аномалій.

У дисертаційній роботі представлені результати наукових досліджень і практичних розробок, спрямованих на підвищення якості інтелектуального аналізу відеопотоку дорожнього руху. Це дозволило підвищити точність та швидкодію систем відеоаналітики за рахунок розроблення модифікованої моделі детекції об’єктів на основі архітектури YOLO, удосконалення методу багатооб’єктного трекінгу з урахуванням оклюзій та створення методу виявлення аномалій на основі аналізу траєкторій руху.

Метою дисертаційної роботи є підвищення якості виявлення аномалій у відеопотоці дорожнього руху шляхом розробки нової інформаційної технології, що базується на удосконалених моделях детекції об’єктів, багатооб’єктного трекінгу та аналізу траєкторій руху.

Для досягнення поставленої мети сформульовано та вирішено такі основні задачі: проведено аналіз сучасних методів комп’ютерного зору для задач детекції, трекінгу та виявлення аномалій; розроблено ефективну модель детекції об’єктів на основі модифікованої архітектури YOLO; удосконалено

метод багатооб’єктного трекінгу для підвищення стійкості до оклюзій; розроблено метод виявлення аномалій на основі аналізу траєкторій; створено архітектуру комплексної інформаційної технології з використанням Edge-Cloud підходу; реалізовано програмний комплекс та проведено експериментальну апробацію.

У **вступі** обґрунтовано актуальність теми дослідження, яка зумовлена стрімким зростанням кількості транспортних засобів, підвищенням вимог до безпеки дорожнього руху та необхідністю автоматизації процесів моніторингу транспортної інфраструктури. Показано, що традиційні підходи, засновані на ручному спостереженні, є неефективними через генерацію великого обсягу відеоданих і низьку частоту виникнення аномалій, що додатково ускладнює їх виявлення традиційними методами. Це зумовлює необхідність створення інтелектуальних інформаційних технологій, здатних працювати в реальному часі та забезпечувати високу точність аналізу.

У **першому розділі** проведено комплексний аналіз сучасних методів комп’ютерного зору для систем відеоспостереження дорожнього руху.
Розглянуто класичні підходи до детекції об’єктів, зокрема SIFT, HOG, Viola-Jones, LBP та GMM. Встановлено, що основними їх недоліками є залежність від вручну створених ознак, низька стійкість до змін умов зйомки та обмежена точність у складних сценах.

Детально проаналізовано сучасні методи на основі глибокого навчання, такі як R-CNN, Fast R-CNN, Faster R-CNN, Mask R-CNN, SSD та YOLO. Показано, що одноетапні детектори, зокрема YOLO, є найбільш придатними для задач реального часу завдяки оптимальному співвідношенню швидкодії та точності, що є критично важливим для задач реального часу.

Окремо розглянуто розвиток архітектур YOLO та їх застосування в інтелектуальних транспортних системах. Також проаналізовано методи багатооб’єктного трекінгу, включаючи класичні підходи та сучасні алгоритми, такі як DeepSORT і ByteTrack.

Досліджено методи виявлення аномалій, включаючи підходи на основі реконструкції, прогнозування та аналізу траєкторій.

У результаті аналізу визначено ключові проблеми сучасних систем:

- висока обчислювальна складність моделей;
- недостатня стійкість трекінгу до оклюзій;
- зниження точності в складних умовах;
- обмеженість спеціалізованих наборів даних (датасетів).

У **другому розділі** розроблено та удосконалено моделі та методи інтелектуального аналізу відеопотоку дорожнього руху. Запропоновано модель детекції об’єктів на основі модифікованої архітектури YOLO із використанням модулів C3Ghost-CBAM та GhostConv. Це дозволило зменшити кількість параметрів і обчислювальну складність моделі при збереженні високої точності.

Удосконалено метод багатооб’єктного трекінгу на основі алгоритму ByteTrack шляхом впровадження механізмів обробки оклюзій, зокрема захисту втрачених треків та покращеної асоціації об’єктів. Це забезпечило підвищення стабільності відстеження та зменшення кількості помилок ідентифікації.

Розроблено метод виявлення аномалій у відеопотоці на основі аналізу траєкторій руху об’єктів. Запропонований підхід дозволяє автоматично виявляти типові порушення дорожнього руху, а саме: проїзд на заборонений сигнал світлофора; перевищення швидкості; аварійне гальмування; тривале перебування об’єкта у визначеній зоні.

У **третьому розділі** розроблено архітектуру інтелектуальної інформаційної технології аналізу відеопотоку дорожнього руху. Запропоновано Edge-Cloud архітектуру, яка передбачає виконання обчислювально складних задач на периферійних пристроях із передачею до хмарної інфраструктури лише метаданих. Це дозволяє значно зменшити мережеве навантаження та забезпечити автономність роботи системи.

Побудовано UML-діаграми, що описують логіку функціонування системи на різних етапах. Обґрунтовано вибір програмного стеку, який включає Python, PyTorch, OpenCV та ONNX Runtime. Розроблено методику оцінювання загальної якості інформаційної технології, яка базується на використанні стандартних метрик якості детекції, трекінгу та швидкодії. Сформовано набори даних для навчання та тестування моделей.

У **четвертому розділі** здійснено програмну реалізацію та експериментальну апробацію розробленої інформаційної технології. Реалізовано програмний комплекс, що включає модулі детекції, трекінгу та виявлення аномалій у єдиний обчислювальний процес.

Експериментальні дослідження, проведені на реальних відеоданих, підтвердили ефективність запропонованих рішень. Отримані результати демонструють високу швидкість обробки у режимі реального часу, підвищення точності детекції та трекінгу, а також ефективне виявлення аномальних ситуацій. Порівняльний аналіз із існуючими та комерційними рішеннями підтвердив конкурентоспроможність розробленої технології.

Загальні висновки

Наукова новизна отриманих результатів полягає у розробці нової моделі детекції об’єктів на основі модифікованої YOLO-архітектури, удосконаленні методу багатооб’єктного трекінгу з урахуванням оклюзій, а також створенні формалізованого методу виявлення аномалій на основі аналізу траєкторій. Подальшого розвитку набула інформаційна технологія аналізу відеопотоку за рахунок інтеграції зазначених методів у єдину Edge-Cloud систему.

Практичне значення роботи полягає у створенні програмного комплексу для інтелектуального аналізу відеопотоку дорожнього руху, який може бути використаний у системах моніторингу дорожнього руху, автоматичної фіксації порушень, управління транспортними потоками та підвищення безпеки дорожньої інфраструктури.

У дисертаційній роботі досягнуто поставленої мети та вирішено всі поставлені задачі. Отримані результати мають наукову новизну та практичну цінність і можуть бути використані у системах моніторингу дорожнього руху, підвищенні безпеки, автоматичній фіксації порушень та аналізі транспортних потоків. Перспективами подальших досліджень є вдосконалення моделей для роботи у складних умовах, розширення функціональних можливостей системи, а також інтеграція з іншими компонентами розумної транспортної інфраструктури.

Ключові слова: згорткові нейронні мережі, машинне навчання, тренування моделей, комп’ютерний зір, обробка зображень, YOLO, інтелектуальні системи, інформаційні технології, математична модель, глибоке навчання, виявлення об‘єктів, Edge-Cloud архітектура, виявлення аномалій.

ABSTRACT

Romanets V. O. Intelligent information technology for analyzing video traffic flow – Qualification scientific work in the form of a manuscript.

Dissertation for the degree of Doctor of Philosophy in the specialty 126 – "Information Systems and Technologies" of the field of knowledge 12 – "Information Technologies", - Vinnytsia National Technical University (VNTU), Ministry of Education and Science of Ukraine, Vinnytsia, 2026.

The dissertation is dedicated to solving the current scientific and applied problem of increasing the efficiency of intelligent analysis of traffic video streams based on the development and integration of an improved object detection model and methods of multi-object tracking and anomaly detection.

The dissertation presents the results of scientific research and practical developments aimed at improving the quality of intelligent analysis of traffic video streams. This allowed to increase the accuracy and speed of video analytics systems by developing a modified object detection model based on the YOLO architecture, improving the multi-object tracking method taking into account occlusions and creating a method for detecting anomalies based on the analysis of motion trajectories.

The purpose of the dissertation is to improve the quality of detecting anomalies in traffic video streams by developing a new information technology based on improved object detection models, multi-object tracking and analysis of motion trajectories.

To achieve the goal, the following main tasks were formulated and solved: an analysis of modern computer vision methods for detection, tracking, and anomaly detection tasks was conducted; an effective object detection model was developed based on the modified YOLO architecture; a multi-object tracking method was improved to increase occlusion resistance; a method for anomaly detection based on trajectory analysis was developed; an architecture of a comprehensive information

technology was created using the Edge-Cloud approach; a software package was implemented and experimental testing was conducted.

The introduction substantiates the relevance of the research topic, which is due to the rapid growth in the number of vehicles, increased requirements for road safety and the need to automate the processes of monitoring transport infrastructure. It is shown that traditional approaches based on manual observation are ineffective due to the generation of a large amount of video data and the low frequency of anomalies, which further complicates their detection by traditional methods. This necessitates the creation of intelligent information technologies capable of working in real time and providing high accuracy of analysis.

The first section provides a comprehensive analysis of modern computer vision methods for road traffic video surveillance systems.

Classical approaches to object detection are considered, in particular SIFT, HOG, Viola-Jones, LBP and GMM. It is established that their main disadvantages are dependence on manually created features, low resistance to changes in shooting conditions and limited accuracy in complex scenes.

Modern methods based on deep learning, such as R-CNN, Fast R-CNN, Faster R-CNN, Mask R-CNN, SSD and YOLO, are analyzed in detail. It is shown that single-stage detectors, in particular YOLO, are most suitable for real-time tasks due to the optimal ratio of speed and accuracy, which is critically important for real-time tasks.

The development of YOLO architectures and their application in intelligent transport systems are separately considered. Multi-object tracking methods are also analyzed, including classical approaches and modern algorithms, such as DeepSORT and ByteTrack.

Anomaly detection methods are investigated, including approaches based on reconstruction, prediction and trajectory analysis.

As a result of the analysis, the key problems of modern systems are identified:

- high computational complexity of models;

- insufficient tracking resistance to occlusions;
- reduced accuracy in difficult conditions;
- Limited availability of specialized datasets.

In the second section, models and methods for intelligent analysis of traffic video streams are developed and improved. An object detection model based on a modified YOLO architecture using the C3Ghost-CBAM and GhostConv modules is proposed. This allowed us to reduce the number of parameters and computational complexity of the model while maintaining high accuracy.

The multi-object tracking method based on the ByteTrack algorithm is improved by implementing occlusion processing mechanisms, in particular, protection of lost tracks and improved object association. This ensured increased tracking stability and reduced identification errors.

A method for detecting anomalies in a video stream based on the analysis of object movement trajectories is developed. The proposed approach allows us to automatically detect typical traffic violations, namely: running a prohibited traffic light; speeding; emergency braking; prolonged stay of an object in a specified area.

In the third section, the architecture of intelligent information technology for traffic video stream analysis is developed. An Edge-Cloud architecture is proposed, which involves performing computationally complex tasks on peripheral devices with the transfer of only metadata to the cloud infrastructure. This allows significantly reducing the network load and ensuring the autonomy of the system.

UML diagrams are constructed that describe the logic of the system's functioning at different stages. The choice of a software stack that includes Python, PyTorch, OpenCV and ONNX Runtime is justified. A methodology for assessing the overall quality of information technology is developed, which is based on the use of standard metrics for detection quality, tracking and speed. Data sets for training and testing models are formed.

In the fourth section, the software implementation and experimental testing of the developed information technology were carried out. A software complex was

implemented that includes detection, tracking, and anomaly detection modules into a single computing process.

Experimental studies conducted on real video data confirmed the effectiveness of the proposed solutions. The results obtained demonstrate high processing speed in real time, increased detection and tracking accuracy, and effective detection of anomalous situations. Comparative analysis with existing and commercial solutions confirmed the competitiveness of the developed technology.

General conclusions

The scientific novelty of the results obtained lies in the development of a new object detection model based on the modified YOLO architecture, the improvement of the multi-object tracking method taking into account occlusions, and the creation of a formalized method for anomaly detection based on trajectory analysis. The information technology of video stream analysis was further developed by integrating the above methods into a single Edge-Cloud system.

The practical significance of the work lies in the creation of a software package for intelligent analysis of the video stream of road traffic, which can be used in systems for monitoring road traffic, automatic recording of violations, traffic flow management and improving the safety of road infrastructure.

The dissertation work has achieved the set goal and solved all the tasks set. The results obtained have scientific novelty and practical value and can be used in systems for monitoring road traffic, improving safety, automatic recording of violations and analyzing traffic flows. Prospects for further research are the improvement of models for operation in difficult conditions, expansion of the functional capabilities of the system, as well as integration with other components of smart transport infrastructure.

Keywords: convolutional neural networks, machine learning, model training, computer vision, image processing, YOLO, intelligent systems, information technology, mathematical model, deep learning, object detection, Edge-Cloud architecture, anomaly detection.

СПИСОК ОПУБЛІКОВАНИХ ПРАЦЬ ЗА ТЕМОЮ ДИСЕРТАЦІЇ

Наукові праці, в яких опубліковані основні наукові результати дисертації:

1. Романець В., Маслій Р. В. Виявлення аномалій у відеопотоці трафіку транспорту засобами комп'ютерного зору та глибинного навчання // Вісник ВПІ. 2025. Вип. 5. С. 146–155. https://doi.org/10.31649/1997-9266-2025-182-5-146-155. ***Фахове видання категорії Б***

2. Романець В., Бісікало О. Метод багатооб'єктного трекінгу з обробкою оклюзій для систем відеоспостереження // Measuring and Computing Devices in Technological Processes. 2025. Т. 84. № 4. С. 440–445. https://doi.org/10.31891/2219-9365-2025-84-53. ***Фахове видання категорії Б***

3. Романець В.О., Бісікало О.В. Інформаційна технологія інтелектуального аналізу відеопотоку дорожнього руху на основі Edge-Cloud архітектури // Вісник Херсонського національного технічного університету. 2026. №1. С.304-310. https://doi.org/10.35546/kntu2078-4481.2026.1.41. ***Фахове видання категорії Б***

Наукові праці, що засвідчують апробацію матеріалів дисертації:

4. Романець В., Бісікало О. Виявлення об'єктів дорожнього руху з камер відеоспостереження // Herald of Khmelnytskyi National University. Technical Sciences. 2025. Т. 355, № 4. С. 491–497. https://doi.org/10.31891/2307-5732-2025-355-70.

5. Романець В., Бісікало О. Information technology in customer service: use of computer vision and decision support systems // SWorldJournal. 2023. Т. 1, № 21-01. С. 65–72. https://doi.org/10.30888/2663-5712.2023-21-01-046.

6. Романець В., Бісікало О., Маслій Р. Адаптація сучасних моделей комп'ютерного зору (YOLOv8-nano) для роботи в реальному часі в системах відеоаналітики дорожнього руху. ВНТКП ВНТУ. Факультет інтелектуальних

інформаційних технологій та автоматизації. 2026. URL: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2026/paper/view/28227.

7. Романець В., Бісікало О. Навчання модифікованої моделі YOLOv8 для детекції транспортних засобів на комбінованому датасеті. Молодь в науці: дослідження, проблеми, перспективи (МН-2026). 2026. URL: https://conferences.vntu.edu.ua/index.php/mn/mn2026/paper/view/28837.

8. Романець В., Бісікало О. Edge-Cloud архітектура інформаційної технології аналізу відеопотоку дорожнього руху. // Scientific development in a changing world. Proceedings of the 5th International scientific and practical conference. SPC “Sci-conf.com.ua”. Lviv, Ukraine. 2026. Pp. 363–367. URL: https://sci-conf.com.ua/wp-content/uploads/2026/05/SCIENTIFIC-DEVELOPMENT-IN-A-CHANGING-WORLD-11-13.05.26.pdf#page=363.

ЗМІСТ

СПИСОК УМОВНИХ СКОРОЧЕНЬ 16

ВСТУП 18

РОЗДІЛ 1. АНАЛІЗ МЕТОДІВ КОМП'ЮТЕРНОГО ЗОРУ ДЛЯ СИСТЕМ ВІДЕОСПОСТЕРЕЖЕННЯ ДОРОЖНЬОГО РУХУ 27

1.1. Огляд методів детекції об'єктів дорожнього руху 27

1.1.1. Класичні методи детекції 27

1.1.2. Методи на основі глибокого навчання 31

1.1.3. Архітектури YOLO 36

1.1.4. Методи на основі трансформерів 40

1.2. Аналіз методів багатооб'єктного трекінгу 43

1.2.1. Класичні методи трекінгу 44

1.2.2. Методи на основі глибоких ознак 46

1.2.3. Методи трекінгу на основі трансформерів 50

1.3. Аналіз методів виявлення аномалій у відеопотоці трафіку 52

1.3.1. Методи на основі реконструкції 54

1.3.2. Методи прогнозування 56

1.3.3. Методи аналізу траєкторій 59

1.4. Проблеми застосування методів у реальному часі 62

1.5. Висновки до розділу 63

РОЗДІЛ 2. РОЗРОБКА МОДЕЛІ ТА МЕТОДІВ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ВІДЕОПОТОКУ ДОРОЖНЬОГО РУХУ 66

2.1 Модель детекції об'єктів дорожнього руху 66

2.2 Розробка трекера об'єктів дорожнього руху з обробкою оклюзій 79

2.3 Розробка методу виявлення аномалій у відео з камер спостереження за дорожнім рухом 85

2.4 Висновки до розділу 97

РОЗДІЛ 3. РОЗРОБКА ІНТЕЛЕКТУАЛЬНОЇ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ АНАЛІЗУ ВІДЕОПОТОКУ ДОРОЖНЬОГО РУХУ 100

3.1 Проектування загальної структури інформаційної технології 100

3.2 Розроблення UML-діаграм інформаційної технології 104

3.2.1 UML-діаграма послідовності інформаційної технології, етап ініціалізації та налаштування системи 105

3.2.2 UML-діаграма послідовності інформаційної технології на етапі обробки відеопотоку 111

3.2.3 UML-діаграма активності навчання моделі детекції 120

3.3 Вибір програмних засобів та технологічного стеку 122

3.3.1 Програмні засоби рівня Edge 123

3.3.2 Програмні засоби рівня Cloud 128

3.3.3 Інструменти розгортання та DevOps 130

3.4 Методика оцінювання якості інформаційної технології та набори даних 131

3.4.1 Часткові метрики оцінювання якості 131

3.4.2 Набори даних для навчання та тестування 135

3.5 Висновки до розділу 141

РОЗДІЛ 4. ПРОГРАМНА ІМПЛЕМЕНТАЦІЯ ТА АПРОБАЦІЯ РОЗРОБЛЕНОЇ ІНТЕЛЕКТУАЛЬНОЇ ТЕХНОЛОГІЇ 142

4.1 Реалізація комплексу програм для навчання детектора об’єктів 142

4.2 Розробка комплексу програм модулів інформаційної технології 151

4.2.1 Модуль детекції 151

4.2.2 Програмна реалізація трекеру об'єктів із підтримкою edge-інференсу 153

4.3 Експериментальна апробація інтелектуальної інформаційної технології 160

4.3.1 Експериментальне середовище та набори даних 161

4.3.2 Результати апробації за сценаріями виявлення аномалій 162

4.3.3 Порівняльний аналіз конфігурацій 166

4.4 Застосування розробленої інформаційної технології аналізу відеопотоку дорожнього руху 167

4.5 Порівняльний аналіз отриманих результатів з комерційними рішеннями для моніторингу дорожнього руху 175

4.6 Висновки до розділу 180

ЗАГАЛЬНІ ВИСНОВКИ 182

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 185

ДОДАТКИ 208

Додаток А. Приклади виявлення аномалій на перехрестях 209

Додаток Б. Відомості про наукові публікації та апробацію матеріалів дисертаційного дослідження 211

Додаток В. Акт про впровадження результатів дисертаційної роботи 213

Додаток Г. Акт про впровадження результатів у навчальний процес ВНТУ. 215

Додаток Д. Лістинг програми трекінгу об'єктів методу ByteTrackPlus 216

СПИСОК УМОВНИХ СКОРОЧЕНЬ

Edge-Cloud – розподілена архітектура обробки даних

YOLO (You Only Look Once) – модель детекції об’єктів

R-CNN / Fast R-CNN / Faster R-CNN – двоетапні моделі детекції

SSD (Single Shot Detector) – одноетапний детектор

MOT (Multiple Object Tracking) – багатооб’єктний трекінг

MOTA (Multiple Object Tracking Accuracy) – метрика точності трекінгу

IDSW (ID Switches) – кількість перемикань ідентичностей

GT (Ground Truth) – істинні (еталонні) значення

ReID (Re-Identification) – повторна ідентифікація об’єктів

IoU (Intersection over Union) – метрика перекриття областей

DIoU (Distance Intersection over Union) – метрика IoU з урахуванням відстані між центрами об’єктів

mAP (mean Average Precision) – середня точність детекції за всіма класами

AP (Average Precision) – точність детекції для окремого класу

MCC (Matthews Correlation Coefficient) – коефіцієнт кореляції Метьюза

Precision – частка правильних позитивних передбачень

Recall – частка правильно виявлених об’єктів

TP (True Positive) – істинно позитивні результати

FP (False Positive) – хибно позитивні результати

FN (False Negative) – хибно негативні результати

TN (True Negative) – істинно негативні результати

FPS (Frames Per Second) – кількість кадрів за секунду

CNN (Convolutional Neural Network) – згорткова нейронна мережа

CBAM (Convolutional Block Attention Module) – модуль механізму уваги

SPPF (Spatial Pyramid Pooling Fast) – модуль пірамідального пулінгу

FPN (Feature Pyramid Network) – пірамідальна мережа ознак

PAN (Path Aggregation Network) – мережа агрегації ознак

NMS (Non-Maximum Suppression) – алгоритм придушення немаксимумів

GFLOPs (Giga Floating Point Operations Per Second) – показник обчислювальної складності

UML (Unified Modeling Language) – мова моделювання систем

HSV (Hue, Saturation, Value) – колірна модель

Оклюзія (Occlusion) – часткове або повне перекриття об’єктів у сцені

LBP (Local Binary Patterns) – локальні бінарні ознаки

HOG (Histogram of Oriented Gradients) – гістограма орієнтованих градієнтів

SIFT (Scale-Invariant Feature Transform) – масштабно-інваріантне перетворення ознак

GMM (Gaussian Mixture Model) – модель змішування гауссіан

DeepSORT (Deep Simple Online and Realtime Tracking) – алгоритм багатооб’єктного трекінгу з використанням глибоких ознак

ONNX (Open Neural Network Exchange) – формат обміну нейронними мережами

OpenCV (Open Source Computer Vision Library) – бібліотека комп’ютерного зору

ResNet (Residual Network) – архітектура згорткової нейронної мережі з залишковими зв’язками

Mask R-CNN (Region-based Convolutional Neural Network) – модель сегментації та детекції об’єктів

Viola-Jones – алгоритм детекції обличчя на основі каскадних класифікаторів

DevOps – практика автоматизації процесів розробки та розгортання програмного забезпечення

ВСТУП

Актуальність теми. Виявлення об'єктів дорожнього руху за допомогою камер спостереження, а також відстеження й ідентифікація інцидентів є базовими технологіями в сучасних інтелектуальних транспортних системах (ITC). Із зростанням кількості транспортних засобів і підвищенням потреби в безпеці на дорогах автоматизовані системи аналізу відеопотоків стають невід'ємною складовою управління дорожнім рухом. За даними Всесвітньої організації охорони здоров'я, щорічно у світі внаслідок дорожньо-транспортних пригод гине близько 1,35 мільйона осіб, що робить питання підвищення безпеки дорожнього руху надзвичайно актуальним.

Системи відеоспостереження становлять основу сучасної інфраструктури забезпечення дорожньої безпеки, ефективного управління транспортними потоками та комплексного моніторингу транспортної мережі. Однак, значне зростання обсягів відеоданих, що надходять із розгалуженої мережі камер спостереження, створює значні труднощі для традиційних методів аналізу, що переважно базуються на спостереженні операторами. Аномальні події є досить рідкісними – за різними оцінками, вони відбуваються лише у 0,01% часу спостереження, що робить ручний моніторинг неефективним і ресурсовитратним процесом. Цей факт, разом із значним розвитком комп'ютерного зору та глибинного навчання, стимулює створення нових автоматизованих підходів до аналізу відеопотоку.

Найважливішими проблемами, з якими стикається галузь, є такі: (а) високі обчислювальні вимоги точних моделей детекції, що унеможливлює їх використання на пристроях з обмеженими ресурсами; (б) недостатня стійкість методів багатооб'єктного трекінгу до оклюзій та перемикань ідентичності без використання ресурсомістких моделей повторної ідентифікації; (в) погіршення якості розпізнавання в складних умовах зйомки (нічний час, погана погода,

низька роздільна здатність); (г) обмежена кількість спеціалізованих датасетів для виявлення аномалій у дорожньому русі.

Окремою актуальною проблемою є необхідність розгортання систем аналізу відеопотоку безпосередньо на Edge-пристроях, тобто обладнанні з обмеженими обчислювальними ресурсами, розташованому поблизу джерел даних (камер спостереження). Передача повного відеопотоку до централізованих серверів для обробки створює значне навантаження на мережеву інфраструктуру, збільшує затримку обробки та унеможливлює автономну роботу системи при втраті з'єднання. Тому актуальним є розроблення Edge-Cloud архітектури, яка забезпечує виконання обчислювально складного пайплайну обробки на Edge-пристрої з передачею до хмарного середовища лише результатів аналізу, що є необхідною умовою для практичного впровадження інтелектуальних транспортних систем у реальній інфраструктурі.

Питаннями, пов'язаними з інтелектуальними інформаційними технологіями аналізу відеопотоку дорожнього руху, займаються провідні наукові колективи та інженерні групи у сфері комп'ютерного зору, машинного навчання й аналітики даних. Серед українських установ варто відзначити Київський національний університет імені Тараса Шевченка, Національний технічний університет України «КПІ ім. Ігоря Сікорського», Національний університет «Львівська політехніка», Західноукраїнський національний університет, Вінницький національний технічний університет, Інститут кібернетики ім. В. М. Глушкова НАН України тощо. Серед провідних закордонних дослідників варто відзначити роботи Р. Гіршика (R-CNN), Дж. Редмона (YOLO), Ч. Ге (YOLOX), Н. Войтала (ByteTrack), З. Цао (OCSort), А. Досовицького (VIT), Н. Каріона (DETR), Ш. Чжао (RT-DETR), К. Хе (ResNet, Mask R-CNN), А. Васвані (Transformer), З. Лю (Swin Transformer), Й. Чжана (FairMOT), Г. Джочера (YOLOv5, YOLOv8) тощо.

Отже, вибір теми дисертаційного дослідження зумовлений як теоретичною, так і практичною значущістю. **Актуальною науковою задачею дисертаційної роботи** є обґрунтування та створення інтелектуальної інформаційної технології аналізу відеопотоку дорожнього руху. Розробка нових методів і технологій такого аналізу відкриває широкі перспективи для впровадження у сферах безпеки дорожнього руху, управління транспортними потоками та автоматичної фіксації порушень правил дорожнього руху.

Зв'язок роботи з науковими програмами, планами і темами.

Дослідження, результати яких представлено в дисертації, відповідають Закону України від 01.12.2022 № 2807-IX «Про Національну програму інформатизації» і проводились протягом 2021–2026 рр. згідно планів наукових досліджень кафедри автоматизації та інтелектуальних інформаційних технологій Вінницького національного технічного університету по науково-дослідній роботі № 41 К 5 «Методи, моделі і засоби обчислень та обробки даних і зображень в інформаційних системах», що відображено в щорічних звітах з науково-дослідної роботи.

Тематика дисертаційного дослідження відповідає пріоритетному напряму розвитку науки і техніки «Інформаційні та комунікаційні технології», визначеному Законом України «Про пріоритетні напрями розвитку науки і техніки» (№ 2623-III від 11.07.2001 р. зі змінами), а також пріоритетному тематичному напряму «Технології та засоби розроблення інтелектуальних інформаційних і інформаційно-аналітичних систем», затвердженому Постановою Кабінету Міністрів України № 942 від 07.09.2011 р. Результати дисертаційного дослідження відповідають:

- Стратегії розвитку сфери інноваційної діяльності на період до 2030 року, затвердженій розпорядженням Кабінету Міністрів України від 10.07.2019р. № 526-р, зокрема в частині розвитку інтелектуальних транспортних систем та технологій штучного інтелекту;

- Концепції розвитку штучного інтелекту в Україні, схваленій розпорядженням Кабінету Міністрів України від 02.12.2020 р. № 1556-р, в частині застосування технологій комп'ютерного зору та машинного навчання для підвищення безпеки дорожнього руху;
- Національній транспортній стратегії України на період до 2030 року, затвердженій розпорядженням Кабінету Міністрів України від 30.05.2018р. № 430-р, в частині впровадження інтелектуальних транспортних систем;
- 17 Цілям сталого розвитку України, зокрема Цілі 3 «Міцне здоров'я і благополуччя» (зменшення смертності та травматизму внаслідок ДТП) та Цілі 11 «Сталий розвиток міст і громад» (забезпечення доступу до безпечних та сталих транспортних систем).

Мета і задачі дослідження. Метою дисертаційної роботи є підвищення якості виявлення аномалій у відеопотоці дорожнього руху шляхом розробки нової інформаційної технології на основі удосконалених моделей і методів детекції об'єктів, багатооб'єктного трекінгу та аналізу траєкторій, придатних для розгортання на Edge-пристроях з обмеженими обчислювальними ресурсами.

У результаті проведеного аналізу для досягнення поставленої мети сформульовані наступні **завдання**:

1. Здійснити детальний аналіз сучасних підходів до детекції об'єктів дорожнього руху, багатооб'єктного трекінгу та виявлення аномалій у відеопотоці, приділивши особливу увагу методам на основі глибокого навчання, їхнім перевагам та недолікам.

2. Розробити ресурсоефективну модель детекції об'єктів дорожнього руху, що забезпечує прийнятну точність виявлення та можливість розгортання на Edge-пристроях з обмеженими обчислювальними ресурсами.

3. Удосконалити метод багатооб'єктного трекінгу для підвищення точності асоціації об'єктів та стійкості до оклюзій без використання ресурсомістких моделей повторної ідентифікації.

4. Розробити метод виявлення аномалій у відеопотоці дорожнього руху на основі аналізу траєкторій об'єктів з можливістю автоматичного виявлення різних типів порушень правил дорожнього руху.

5. Розробити архітектуру комплексної інформаційної технології аналізу відеопотоку дорожнього руху, що інтегрує створені моделі та методи детекції, трекінгу та виявлення аномалій у Edge-Cloud архітектурі.

6. Здійснити програмну реалізацію розроблених методів та провести експериментальні дослідження на реальних задачах і датасетах для підтвердження ефективності запропонованих рішень.

Об'єктом дослідження є процеси інтелектуального аналізу відеопотоку дорожнього руху для виявлення аномальних подій.

Предметом дослідження є інформаційна технологія, методи, моделі та алгоритми детекції об'єктів, багатооб'єктного трекінгу та виявлення аномалій у відеопотоці дорожнього руху.

Методи дослідження містять загальнонаукову методологію проведення досліджень і принципи системного підходу, а саме: аналіз літературних джерел та предметної області для визначення актуальних проблем та напрямків дослідження; огляд та порівняння існуючих моделей детекції, трекінгу та виявлення аномалій; методи глибокого навчання (згорткові нейронні мережі, механізми уваги) для розробки моделі детекції об'єктів; методи комп'ютерного зору та обробки зображень (гомографічне перетворення, аналіз кольорових каналів) для виявлення аномалій; методи оптимізації та фільтрації (фільтр Калмана, угорський алгоритм) для багатооб'єктного трекінгу; комп'ютерні експерименти для верифікації та валідації запропонованих рішень.

Наукова новизна одержаних результатів.

1. Уперше запропоновано модель детекції об'єктів дорожнього руху, яку, на відміну від існуючих, побудовано на основі модифікованої YOLO-архітектури із інтеграцією модулів C3Ghost-CBAM та полегшених згорток GhostConv. Це забезпечує зменшення кількості параметрів моделі на 23,3% (з 3,0 до 2,3 млн) та обчислювальної складності на 9,9% (з 8,1 до 7,3 GFLOPS) при незначному зниженні точності (значення метрики mAP@0,50:0,95 знизилось на 1,3%, з 0,715 до 0,706), що дозволяє використовувати модель на Edge-пристроях з обмеженими обчислювальними ресурсами.

2. Удосконалено метод багатооб'єктного трекінгу ByteTrack, який, на відміну від існуючих, поєднує три механізми: IoU-suppression для запобігання дублюванню треків при ініціалізації, механізм захисту втрачених треків (occlusion grace period) для обробки короткочасних оклюзій та метрику DIoU для покращення асоціації детекцій з треками. Це дозволило підвищити метрику MOTA на 0,5% (з 77,5% до 78,0%), IDF1 на 1,9% (з 75,8% до 77,7%) та зменшити кількість перемикань ідентичностей на 9,0% (з 210 до 191) при збереженні швидкості оброблення понад 29 FPS на датасеті MOT17-val.

3. Уперше запропоновано формальну модель виявлення аномалій у відеопотоці дорожнього руху на основі аналізу траєкторій, яка, на відміну від існуючих, враховує чотири типи порушень: проїзд на червоний сигнал світлофора, перевищення швидкості, аварійне гальмування та тривале перебування в зоні перехрестя – шляхом комбінування геометричного (функція перетину стоп-лінії), спектрального (HSV-аналіз стану світлофора), кінематичного (різниця швидкостей між послідовними кадрами) та топологічного (функція point-in-polygon) критеріїв, що дозволило описати складні умови виявлення кожного типу порушення.

4. Подальшого розвитку отримала інформаційна технологія аналізу відеопотоку дорожнього руху, яка, на відміну від існуючих, використовує Edge-Cloud-архітектуру з модульною структурою та забезпечує виконання повного пайплайну оброблення (детекція → трекінг → виявлення аномалій)

безпосередньо на Edge-пристрої, що дозволяє здійснювати виявлення аномалій у реальному часі з коефіцієнтом кореляції Меттьюза (MCC) від 0,515 до 0,861 для різних типів аномалій без залежності від пропускної здатності мережі та забезпечує автономну роботу системи при втраті з'єднання.

Практичне значення одержаних результатів.

1. Розроблено програмний комплекс мовою програмування Python для інтелектуального аналізу відеопотоку дорожнього руху, що реалізує функціонал усіх запропонованих моделей і методів детекції, трекінгу та виявлення аномалій з використанням фреймворків PyTorch, ONNX Runtime та OpenCV, та забезпечує швидкість оброблення понад 35 FPS на Edge-пристрої NVIDIA Jetson Orin Nano.

2. Створено комбінований набір даних для навчання моделі детекції, що містить 12000 зображень (2000 власних зображень з камер відеоспостереження трафіку та 10000 зображень з публічного датасету DETRAC) з анотаціями у форматі YOLO, що дозволяє навчати та порівнювати моделі детекції транспортних засобів в умовах, характерних для реальних систем відеоспостереження, без необхідності додаткової розмітки даних, та забезпечує відтворюваність результатів дослідження.

3. Сформовано датасет для апробації інформаційної технології, що містить 210 хвилин відеозаписів з трьох перехресть з 680 розміченими подіями (аномаліями та нормальними ситуаціями), що забезпечує репрезентативну оцінку ефективності системи.

4. Розроблено модуль калібрування камери на основі гомографічного перетворення з інтерактивним інтерфейсом для визначення точок відповідності, що дозволяє обчислювати швидкість транспортних засобів у реальних координатах.

5. Запропонована інформаційна технологія може бути використана для розв’язання таких задач: автоматичної фіксації порушень правил дорожнього руху (проїзд на червоний сигнал, перевищення швидкості); моніторингу

завантаженості транспортних потоків; підвищення безпеки на перехрестях та складних ділянках доріг; аналізу аварійних ситуацій (аварійне гальмування, тривале перебування в зоні перехрестя).

Особистий внесок здобувача. Усі наукові результати, викладені в дисертації, отримані автором особисто. У працях, опублікованих у співавторстві, здобувачеві належать: [1] – розробка моделі детекції об'єктів на основі модифікованої архітектури YOLOv8 з модулями C3Ghost-CBAM; [2] – розробка методу виявлення аномалій у відеопотоці трафіку; [3] – удосконалення методу багатооб'єктного трекінгу ByteTrack+; [4] – розробка архітектури інтелектуальної інформаційної технології аналізу відеопотоку дорожнього руху на основі Edge-Cloud моделі, проектування взаємодії компонентів системи; [5] – аналіз застосування комп'ютерного зору в системах обслуговування клієнтів.

Апробація матеріалів дисертації. Основні наукові результати та практичні розробки дисертаційної роботи пройшли апробацію, зокрема опубліковані 1 стаття у періодичному закордонному виданні, 1 стаття в іншому науковому фаховому виданні та взято участь з доповідями у роботі 1 всеукраїнської та 2 міжнародних наукових конференціях:

- Стаття «Information technology in customer service: use of computer vision and decision support systems» (SWorldJournal, 2023).
- Стаття «Виявлення об'єктів дорожнього руху з камер відеоспостереження» (Herald of Khmelnytskyi National University. Technical Sciences, 2025).
- LV Всеукраїнська науково-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації ВНТУ (Вінниця, ВНТУ, 2026).
- Міжнародній науково-практичній інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи» (Вінниця, ВНТУ, 2026).

- 5th International Scientific and Practical Conference «Scientific development in a changing world» (Lviv, SPC “Sci-conf.com.ua”, 2026).

Публікації. Основні результати дисертаційного дослідження викладені у 8 наукових працях (див. Додаток Б). Опубліковано 3 статті у фахових виданнях України категорії Б України зі спеціальності 126 [2], [3], [4]; 1 стаття, що входить до міжнародного періодичного журналу [5], 1 стаття в іншому науковому фаховому виданні [1] та 3 публікації в матеріалах всеукраїнської та міжнародних наукових конференцій [6], [7], [8].

Структура та обсяг дисертації. Дисертація складається зі вступу, чотирьох розділів, висновків, списку використаних джерел із 192 найменувань. Загальний обсяг дисертації становить 228 сторінок, з них 207 сторінок основного тексту, який містить 64 рисунка та 27 таблиць.

РОЗДІЛ 1. АНАЛІЗ МЕТОДІВ КОМП'ЮТЕРНОГО ЗОРУ ДЛЯ СИСТЕМ ВІДЕОСПОСТЕРЕЖЕННЯ ДОРОЖНЬОГО РУХУ

1.1. Огляд методів детекції об'єктів дорожнього руху

Детекція об'єктів є однією з фундаментальних задач комп'ютерного зору, яка полягає у виявленні та локалізації об'єктів певних класів у зображеннях або відеопотоці. У контексті інтелектуальних транспортних систем (ІТС) детекція об'єктів дорожнього руху включає розпізнавання транспортних засобів, пішоходів, велосипедистів, дорожніх знаків та інших учасників дорожнього руху. Точна та швидка детекція є дуже важливою для забезпечення безпеки руху, управління транспортними потоками та автоматизації транспортних систем.

Еволюція методів детекції об'єктів пройшла шлях від класичних підходів, заснованих на вручну створених ознаках, до сучасних методів глибокого навчання, що автоматично витягують ієрархічні представлення з даних.

1.1.1. Класичні методи детекції

У ранніх системах відеоспостереження дорожнього руху використовувалися класичні методи комп'ютерного зору, що базувалися на вручну створених ознаках (hand-crafted features) та традиційних алгоритмах машинного навчання.

Одним із найпоширеніших підходів було використання алгоритму Scale-Invariant Feature Transform (SIFT), запропонованого Lowe у 1999 році [9]. SIFT виявляє та описує локальні ознаки зображень, що є стійкими до змін масштабу, повороту та освітлення. Метод базується на пошуку екстремумів у просторі масштабів різницевих гаусіанів (Difference of Gaussians, DoG) та побудові дескрипторів на основі гістограм орієнтацій градієнтів у околицях ключових

точок [9]. Хоча SIFT демонструє високу інваріантність до геометричних перетворень, його обчислювальна складність та необхідність великої кількості ознак обмежують застосування в системах реального часу.

Альтернативним підходом став метод Histogram of Oriented Gradients (HOG), запропонований Dalal та Triggs у 2005 році [10]. HOG базується на обчисленні гістограм орієнтацій градієнтів у локалізованих ділянках зображення для представлення форми та структури об'єктів. Метод виявився особливо ефективним для детекції пішоходів завдяки здатності захоплювати загальну форму об'єкта. Типовий алгоритм детекції з використанням HOG включає: обчислення градієнтів зображення; розбиття на комірки (cells) та блоки (blocks); обчислення гістограм орієнтацій градієнтів; нормалізацію блоків; класифікацію за допомогою Support Vector Machine (SVM) [10].

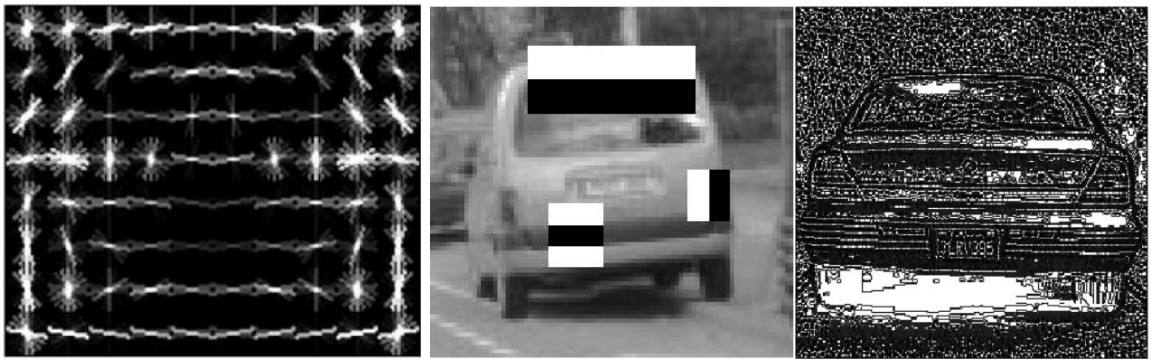
Одним із найвпливовіших проривів у задачі швидкої детекції об’єктів стала робота Viola та Jones (2001), в якій було запропоновано перший алгоритм, здатний виявляти обличчя в реальному часі на стандартному апаратному забезпеченні того часу [11, 12]. Метод базується на трьох основних компонентах: хаароподібних ознаках, інтегральному зображенні та каскадній архітектурі при використанні алгоритму Adaboost для навчання.

Незважаючи на високу швидкодію, метод має суттєві обмеження: чутливість до змін кута зору, освітлення, часткових закриттів об’єкта та високу частоту хибнопозитивних спрацювань на складних фонах [13]. Подальші модифікації, зокрема, використання попередньої обробки та альтернативних типів ознак, частково компенсували ці недоліки [14].

Іншим популярним класичним підходом стали методи на основі локальних бінарних шаблонів (Local Binary Patterns, LBP) [15, 16]. LBP-коди формуються шляхом порівняння яскравості центрального пікселя з яскравістю його сусідів, що забезпечує інваріантність до монотонних змін освітлення та високу обчислювальну ефективність. Варіанти LBP успішно застосовувались

як самостійні дескриптори [17], так і в комбінації з каскадними класифікаторами [14].

У роботі [18] LBP-ознаки використовуються для виявлення автомобілів. У роботі [19] поєднання HOG та LBP-ознак дозволяє досягти значного покращення якості виявлення людей у випадку часткових оклюзій. У роботі [20] здійснюється порівняння SIFT, Хаар та LBP-ознак для виявлення автомобілів для задачі автономного водіння. Найкращим з точки зору балансу якості/швидкості обчислення на думку авторів виявився підхід що використовує HOG-ознаки. Приклад ознак автомобілів наведений на рис. 1.1.



а) б) в)

Рис. 1.1. Порівняння ознак отриманих для виявлення задньої частини авто: а) HOG-ознаки; б) Хаар-ознаки; в) LBP-ознаки [20]

Ще одна група методів виявлення об'єктів базувалася на аналізі руху через оптичний потік або сегментацію фону. Алгоритми на основі Gaussian Mixture Model (GMM) використовувалися для відокремлення рухомих об'єктів від статичного тла. Зокрема у роботі [21] досліджувалися GMM-базовані методи для виявлення транспортних засобів у різних умовах спостереження. У роботі [22] автори застосували GMM для виявлення транспортних засобів та фільтр Калмана для їх відстеження. Хоча ці методи були економними з точки зору обчислювальних ресурсів, їхня ефективність знижувалася в умовах

щільного трафіку чи при наявності перешкод, таких як тіні чи погодні явища [21, 22].

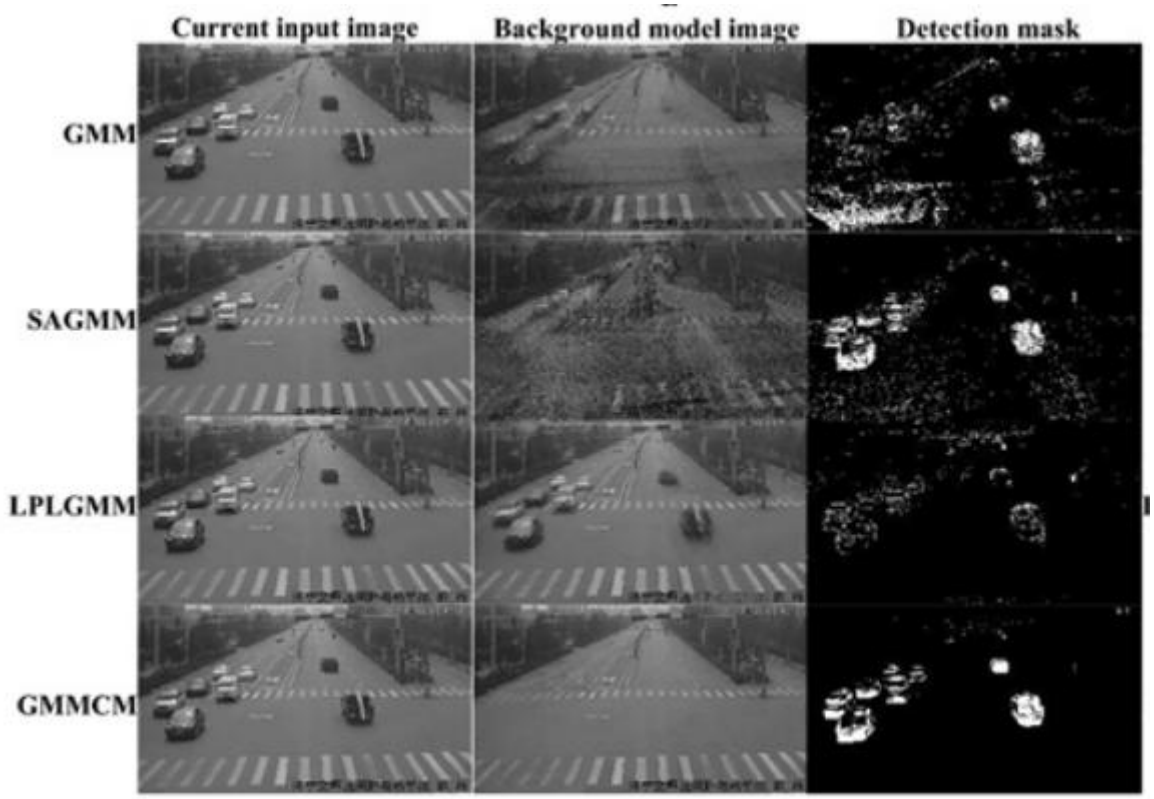


Рис. 1.2. Виявлення автомобілів за допомогою GMM підходів [21]

Таблиця 1.1 – Порівняння традиційних методів виявлення об'єктів

Метод	Рік	Особливості реалізації	Переваги	Недоліки	Застосування для ІТС
SIFT	1999	Difference of Gaussians (DoG), гістограми градієнтів	Інваріантність до масштабу, повороту, освітлення	Висока обчислювальна складність	[23]
HOG	2005	Гістограми орієнтацій градієнтів, нормалізація блоків	Ефективний для форм об'єктів (пішоходи)	Чутливий до оклюзій	[19, 20]

Продовження табл. 1.1

Метод	Рік	Особливості реалізації	Переваги	Недоліки	Застосування для ІТС
Viola-Jones	2001	Хаароподібні-ознаки, інтегральне зображення, каскад класифікаторів, AdaBoost алгоритм	Реальний час (15+ FPS)	Хибнопозитивні на складних фонах	[20, 24, 25]
LBP	2002	Бінарні шаблони пікселів	Інваріантність до освітлення, швидкість	Менш точний при варіаціях	[18-20]
GMM	1999	Моделі сумішей гаусіанів	Економний для сегментації руху	Погано в щільному трафіку/погоді	[21, 22]

Основними обмеженнями класичних методів детекції є [26]:

- залежність від вручну створених ознак, що потребує експертних знань предметної області;

- обмежена здатність до узагальнення на різноманітні умови зйомки (освітлення, погода, ракурс);

- необхідність окремого проектування ознак для різних класів об'єктів;

- чутливість до варіацій у зовнішньому вигляді об'єктів (оклюзії, деформації, зміни масштабу);

- недостатня точність для складних сценаріїв дорожнього руху з множинними об'єктами.

1.1.2. Методи на основі глибокого навчання

Поява глибокого навчання значно змінила парадигму детекції об’єктів. Згорткові нейронні мережі (Convolutional Neural Networks, CNN) замість використання вручну створених ознак (hand-crafted features) дозволили

автоматично навчати ієрархічні представлення ознак, значно перевершивши за точністю всі класичні методи [26].

Важливим етапом у розвитку методів виявлення об'єктів стала архітектура Region-based CNN (R-CNN), запропонована Girshick та колегами у 2014 році [27]. R-CNN поєднує генерацію регіональних пропозицій (region proposals) за допомогою алгоритму selective search [28] із класифікацією на основі згорткових нейронних мереж, реалізуючи двоетапний підхід (two-stage detector) (рис 1.3). Метод включає такі кроки: (1) генерація приблизно 2000 кандидатних регіонів за допомогою selective search; (2) витягування ознак для кожного регіону за допомогою CNN; (3) класифікація регіонів за допомогою SVM; (4) регресія обмежувальних рамок для уточнення локалізації [29].

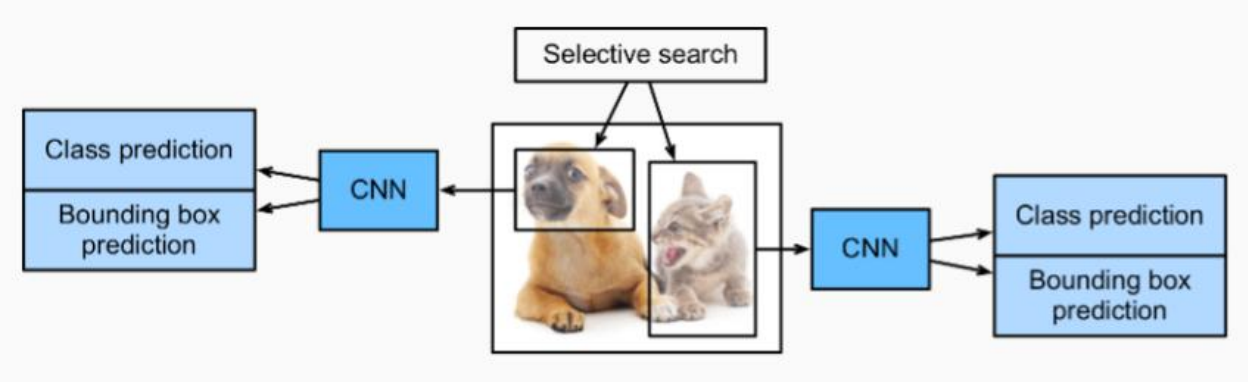


Рис. 1.3. Структура методу виявлення R-CNN [29]

Незважаючи на високу точність, R-CNN характеризується низькою швидкістю обробки через необхідність обробки кожного регіону окремо. Подальші вдосконалення, такі як Fast R-CNN [30] і Faster R-CNN [31], значно підвищили швидкість і точність. Fast R-CNN запровадив RoI pooling для спільної обробки всього зображення та окремих регіонів [30]. Faster R-CNN замінив selective search на Region Proposal Network (RPN), що генерує пропозиції безпосередньо з карт ознак [31].

Подальшим розвитком сімейства став Mask R-CNN (2017) [32] – розширення Faster R-CNN для одночасної детекції, класифікації та сегментації екземплярів (instance segmentation). Основними удосконаленнями Mask R-CNN є заміна RoI Pooling на RoI Align для точнішого вирівнювання за допомогою білінійної інтерполяції, для збереження просторової інформації та додавання повністю згорткової мережі для прогнозування бінарних масок об’єктів (рис. 1.4).

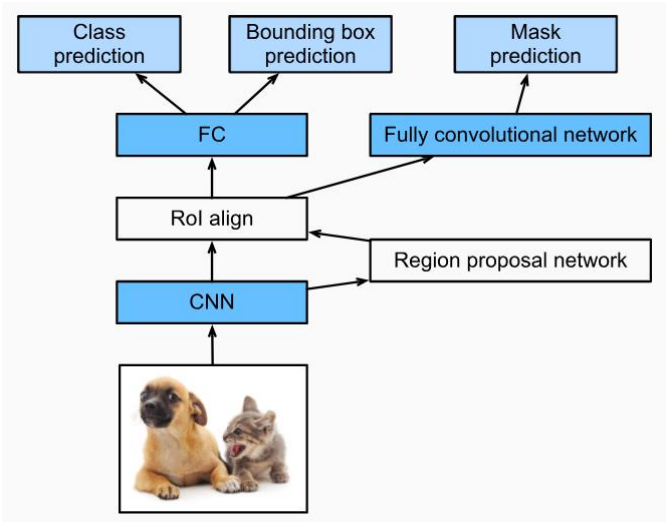


Рис. 1.4. Структура методу виявлення Mask R-CNN [29]

Однією з перших моделей, призначених для виявлення та класифікації об'єктів дорожнього руху з камер відеоспостереження в реальному часі, стала нейронна мережа DetectNet, розроблена компанією NVIDIA [33]. Ця мережа є адаптацією архітектури GoogLeNet [34] для задачі детекції об'єктів.

Паралельно з двоетапними розвивалися одноетапні методи виявлення об'єктів (single-stage detectors), такі як YOLO (You Only Look Once) [35] і SSD (Single Shot MultiBox Detector) [36]. Ці методи розглядають детекцію як задачу регресії, передбачаючи обмежувальні рамки та ймовірності класів безпосередньо з ознакових карт повного зображення в один прохід.

YOLO, запропонований Redmon та колегами у 2016 році, став революційним завдяки своїй здатності обробляти відеопотоки в реальному часі з високою точністю [35]. Базова архітектура YOLO розділяє вхідне зображення на сітку S×S клітинок, де кожна клітинка передбачає B обмежувальних рамок та їх довірчі оцінки, а також ймовірності класів для об'єктів, центри яких потрапляють у цю клітинку. Це дозволяє моделі “дивитися” на все зображення одразу (звідси назва “You Only Look Once”) та використовувати глобальний контекст для передбачення (рис. 1.5).

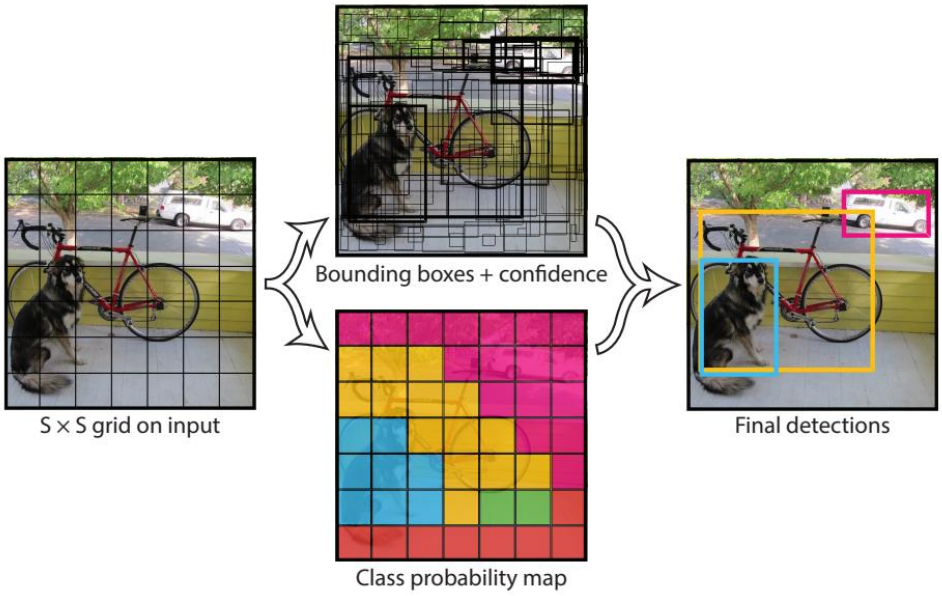


Рис. 1.5. Візуалізація роботи методу виявлення YOLO [35]

Подальший розвиток методів глибокого навчання в галузі детекції об’єктів призвів до появи сімейства моделей EfficientDet [37], які на момент публікації встановили новий стандарт співвідношення точності та обчислювальної ефективності.

На відміну від попередніх підходів, EfficientDet базується на принципово новій концепції compound scaling (складового масштабування), вперше запропонований в архітектурі EfficientNet [38]. Цей метод одночасно та збалансовано масштабує три ключові параметри нейронної мережі: глибину

(кількість шарів, depth), ширину (кількість каналів у шарах, width), роздільну здатність вхідного зображення (resolution).

Як backbone для вилучення ознак використовується одна з версій EfficientNet (від B0 до B7), оптимізована за допомогою нейронного пошуку архітектури (Neural Architecture Search, NAS).

Крім того для кращого використання ознак EfficientDet замінює традиційну Feature Pyramid Network (FPN) на weighted Bi-directional Feature Pyramid Network (BiFPN) (рис. 1.6).

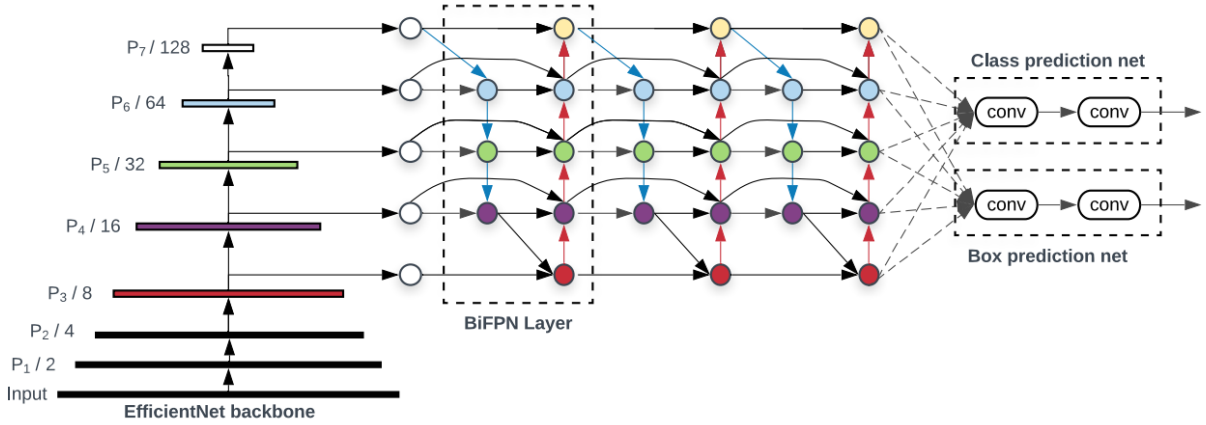


Рис. 1.6. Архітектура методу виявлення EfficientDet [37]

Таблиця 1.2 – Методи виявлення об'єктів на основі глибоких нейронних мереж

Метод	Рік	Тип	Точність (mAP)	Швидкість (FPS)	Застосування в ІТС
R-CNN	2014	Двохетапний CNN	53,3% (VOC 2012)	~0,07	-
Fast R-CNN	2015	Двохетапний CNN	70,0% (VOC 2012)	5	[39]
Faster R-CNN	2015	Двохетапний CNN	73,2% (VOC 2012)	5-17	[40-42]
DetectNet	2016	Одноетапний FCN	60-65% (KITTI)	~24	[33, 43, 44]

Продовження табл. 1.2

Метод	Рік	Тип	Точність (mAP)	Швидкість (FPS)	Застосування в ІТС
YOLO	2016	Одноетапний CNN	~63% (VOC)	45+	[35]
SSD	2016	Одноетапний CNN	~74% (VOC)	59	[45, 46]
Mask R-CNN	2017	Двохетапний CNN	39,8% box AP (COCO)	~5	[47-49]
EfficientDet	2019	Одноетапний CNN	55,1% AP (COCO)	26+	[50]

1.1.3. Архітектури YOLO

Сімейство архітектур YOLO заслуговує окремої уваги через їхню популярність у застосуваннях відеоспостереження дорожнього руху та здатність працювати в реальному часі [51]. Ілюстрація історії розвитку методів на основі архітектури YOLO наведена на рис. 1.7.

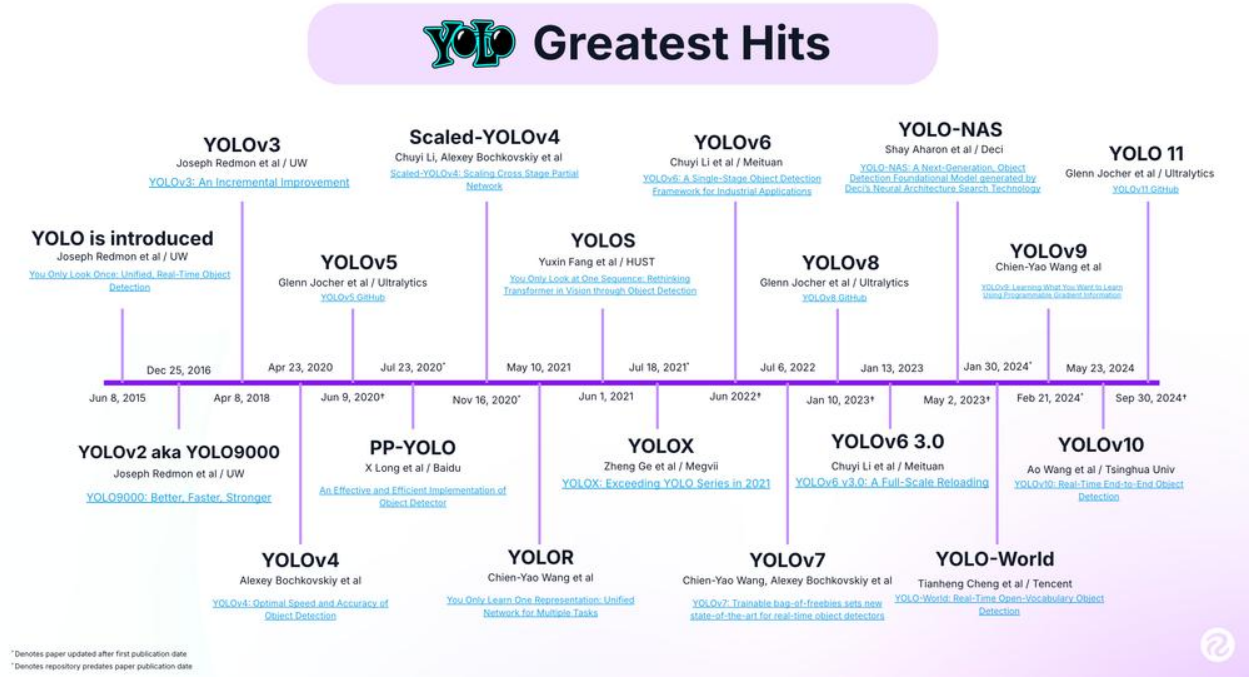


Рис. 1.7. Історія YOLO моделей [51]

YOLOv1 (2016) запровадив концепцію одноетапної детекції, розглядаючи задачу як регресію до обмежувальних рамок та ймовірностей класів. Модель використовує 24 згорткових шари для вилучення ознак та 2 повнозв'язних шари для передбачення.

YOLOv2 (2017) [52] покращив базову архітектуру шляхом введення batch normalization, anchor boxes, multi-scale training та кращої backbone-мережі Darknet-19. Це дозволило підвищити точність з 63,4% до 78,6% mAP на датасеті PASCAL VOC 2007 [52].

YOLOv3 (2018) [53] запровадив мультимасштабні передбачення на трьох різних рівнях ознакових карт, що покращило детекцію об'єктів малого розміру. Backbone було замінено на Darknet-53 (53 згорткових шари), що забезпечило кращу екстракцію ознак. YOLOv3 використовує Feature Pyramid Network (FPN)-подібну структуру для агрегації ознак різних масштабів.

YOLOv4 (2020) [54] інтегрував численні техніки покращення: CSPDarknet53 як backbone, SPP (Spatial Pyramid Pooling) та PAN (Path Aggregation Network) у neck-частині, а також різні тренувальні підходи (Mosaic augmentation, DropBlock, CIoU loss). Це дозволило досягти 43,5% AP на COCO при швидкості 65 FPS на Tesla V100 [54].

YOLOv5 (2020) [55], розроблений компанією Ultralytics, запровадив модульну архітектуру з різними варіантами моделей (YOLOv5n/s/m/l/x) для різних вимог до швидкості та точності. Модель використовує використовує CSP-структуру для покращеної екстракції ознак та автоматичне налаштування anchor boxes.

YOLOX (2021) [56], розроблений компанією Megvii, представив anchor-free архітектуру та decoupled head для розділення класифікації та локалізації об'єктів. Модель використовує Darknet53 як backbone з FPN та PAN у neck-частині, а також інтегрує Advanced Label Assignment Strategy та значні аугментації (Mosaic, MixUp). На бенчмарку COCO YOLOX досягає 47,3% AP

при швидкості 77 FPS на Tesla V100, забезпечуючи значне покращення як у точності, так і швидкості порівняно з YOLOv4 та YOLOv5 [56].

YOLOv6 (2022) [57] від Meituan зосередився на промисловому застосуванні, запровадивши RepVGG-style backbone, efficient decoupled head та безякірну парадигму. У дослідженні [66] для відстеження транспортних засобів у відео автори застосували алгоритм DeepSORT, використавши PP-YOLO як детектор.

YOLOv7 (2022) [58] досяг state-of-the-art результатів завдяки extended efficient layer aggregation networks (E-ELAN), model scaling for concatenation-based models та плановаому re-parametrization.

YOLOv8 (2023) [59] від Ultralytics це безякірна (anchor-free) модель, яка прогнозує центри об'єктів безпосередньо, замість зміщень від якорів. Включає покращений CSPDarknet бекбон, C2f модулі та Task-Aligned Assigner для динамічного призначення міток. Підтримує детекцію, сегментацію, класифікацію, оцінку поз та OBB. Модель оптимізовано для edge-пристроїв з mosaic/mixup аугментаціями для уникнення overfitting.

YOLOv9 (2024) [60]: вводить модуль Programmable Gradient Information (PGI) для збереження інформації через шари та бекбон Generalized Efficient Layer Aggregation Network (GELAN) для кращої ефективності. Легкі моделі для мобільних пристроїв, досягає вищої mAP на COCO порівняно з YOLOv8/v7/v5. Фокус на балансі точності та ресурсів.

YOLOv10 (2024, Tsinghua University) [61]: NMS-free навчання з consistent dual-label assignments (one-to-many/one-to-one). Покращений CSPNet бекбон, PAN neck, large-kernel convolutions та partial self-attention. Зменшує затримку на 46% та параметри на 25% порівняно з YOLOv9-C при подібній продуктивності. Варіант YOLOv10b для балансу.

YOLOv11 (2024) [62] від Ultralytics, в якій оптимізований в порівнянні з yolov8 backbone та neck для кращого витягування ознак. Як і yolov8 підтримує детекцію, сегментацію, класифікацію, keypoints та OBB. YOLOv11m має вищу

mAP на COCO, ніж YOLOv8m, з 22% меншою кількістю параметрів. Сумісний з YOLOv8 workflows.

Таблиця 1.3 – Yolo-базовані методи виявлення об'єктів

Метод	Рік	Бекбон	Об'єднання ознак	Точність (mAP) (COCO)	Швидкість (FPS) (V100)	Застосування в ІТС
yolov3	2018	Darknet-53	Декілька масштабів	33%	45	[63, 64, 65]
yolov4	2020	CSPDarknet-53	PANet + SPP	43,5 %	65	[66-68]
yolov5	2020	CSPDarknet (Pytorch)	PANet	28 % (n) 50,7 % (x)	142 (n) 26 (x)	[69-72, 73-74]
yoloX	2021	CSPDarknet-53	FPN+PANet	50,0 (L) 51,1 (X)	70 (L) 58,0 (X)	[75, 76]
yolo6	2022	EfficientRepNet	RepPAN	37 % 51 %	484 85	[77, 78]
yolov7	2022	E-ELAN	PANet + E-ELAN	37,2 % 56,8	161 8	[79, 80]
yolov8	2023	Cf2 модулі	PANet	37,3% 53,9 %	80,4 20,1	[81-86, 87]
yolov9	2024	GELAN	GELAN-FPN	38,3 % (n) 55,6 % (x)	183 (n) 55 (x)	[88, 89]
yolov10	2024	CSPNet (Enhanced)	PANet (NMS-free dual head)	38,5 % (n) 54,4 % (x)	120 (n) 25 (x)	[90, 91]
yolov11	2024	CSPNet + C3K2 + C2PSA	PANet + C3K2 (FPN/PAN)	39,5 % (n) 54,7 % (x)	666 (n) 88 (x)	[92-94]

1.1.4. Методи на основі трансформерів

Останнім часом активно розвиваються методи на основі трансформерів для задач комп'ютерного зору [95-97]. Vision Transformer (ViT) [95] та їхні похідні, такі як DETR (DEtection TRansformer) [98], RT-DETR [99] показали високу ефективність у задачах виявлення об'єктів із складними взаємозв'язками, наприклад, при аналізі щільних дорожніх сцен із численними учасниками руху. DETR формулює детекцію об'єктів як задачу передбачення множини (set prediction), використовуючи механізм уваги для безпосереднього передбачення фіксованої кількості детекцій без потреби в anchor boxes або non-maximum suppression (NMS) [98].

Архітектура DETR [98] складається з трьох основних компонентів:

1) CNN Backbone (зазвичай ResNet-50/101 [2]): екстрагує просторові ознаки з вхідного зображення H×W×3, після згорткових шарів з stride=32 отримуючи карту ознак розміром H/32 × W/32 × C, які через 1×1 згортку зменшується до H/32 × W/32 × d_model (d_model=256);

2) траснформерний кодер (6 шарів): обробляє просторові ознаки, перетворені у послідовність (flattened) розміром (HW/1024) × d_model, доповнені 2D позиційним кодуванням для збереження просторової структури. Multi-head self-attention дозволяє моделювати глобальні залежності між всіма просторовими частинами зображення;

3) трансформерний декодер (6 шарів): приймає N=100 навчаних object queries (ембеденгів розміром N × d_model) та encoder outputs. Через механізм самоуваги (self-attention) між queries та cross-attention з ознак зображень, декодер генерує N передбачень. Кожен вихід декодера передається через загальну мережу прямого поширення (shared feed-forward network - FFN), яка передбачає клас об'єкта (разом зі спеціальним класом “no object“) та нормалізовані обмежувальні прямокутники (center_x, center_y, width, height) $\in [0,1]^4$.

Під час етапу навчання DETR використовує угорський алгоритм для знаходження оптимального зіставлення $\hat{\sigma}$ між N передбаченнями та M справжніми (ground truth) об’єктами, мінімізуючи помилку зіставлення (matching cost). Після зіставлення обчислюється функція втрат:

$$L = \Sigma_{i=1}^{N} [-\log \hat{p}(c_i) + \mathbb{1}\{c_i \neq \emptyset\} L_{box}(b_i, \hat{b}_i)],$$

де $L_{box} = \lambda_{L1} \cdot ||b - \hat{b}||_1 + \lambda_{giou} \cdot L_{GIoU}$.

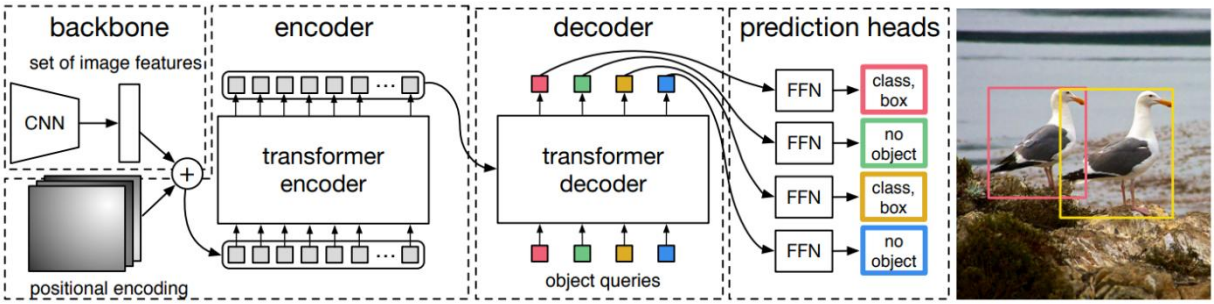


Рис. 1.8. Архітектура методу виявлення DETR [98]

У методі RT-DETR [99] багатомасштабні ознаки з останніх трьох рівнів backbone - P3 (H/8×W/8), P4 (H/16×W/16), P5 (H/32×W/32) - передаються в ефективний гібридний кодер, що комбінує CNN та трансформер компоненти:

- AIFI (Attention-based Intra-scale Feature Interaction) модуль використовує окремий механізм самоуваги для кожного масштабу (P3, P4, P5) для захоплення довгострокових залежностей всередині масштабу з лінійною складністю $O(HW/4^i)$;

- CCFM (CNN-based Cross-scale Feature Fusion) модуль здійснює CNN операції для швидкого міжмасштабного об'єднання ознак, зберігаючи просторову структуру.

- IoU-aware query selection модуль замість фіксованих навчених queries відбирає top-N найінформативніших ознак (N=300 для навчання, N=100 для інференсу) на основі показника невизначеності (uncertainty score) u = IoU_pred max(class_conf), де вищий показник означає нижчу невизначеність та вищу

ймовірність містити об'єкт. Це забезпечує адаптивну контент-базовану ініціалізацію queries.

- трансформерний декодер (6 шарів) який використовуючи самоувагу, крос-увагу (cross-attention) та FFN ітеративно уточнює queries. Допоміжні головки передбачень (auxiliary prediction heads) (class, bbox, IoU) після кожного шару декодера забезпечують проміжний нагляд, оптимізуючи комбіновану функцію втрат для генерації фінальних передбачень класів та обмежувальних рамок:

$$L = \Sigma_l [L_{cls} + \lambda_{L1} \cdot L_{L1} + \lambda_{GIoU} \cdot L_{GIoU} + \lambda_{IoU} \cdot L_{IoU}]$$

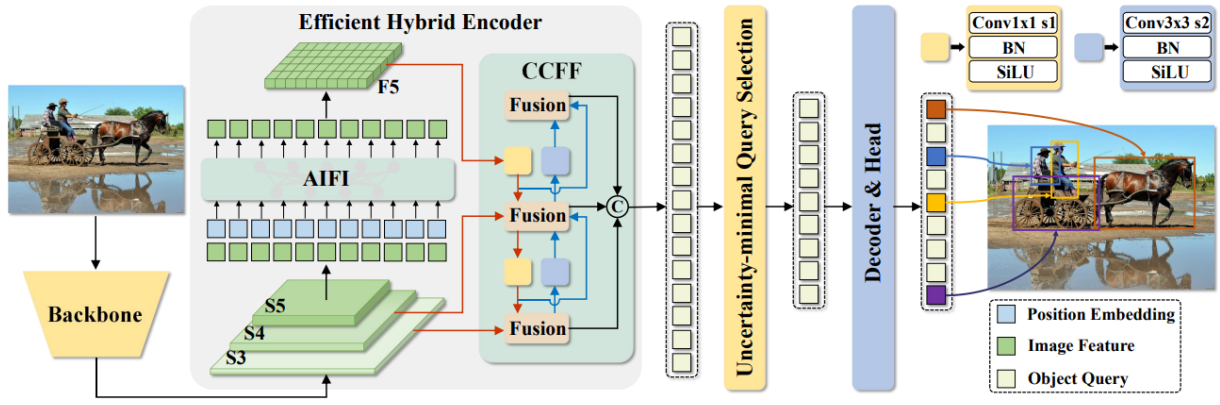


Рис. 1.9. Архітектура методу виявлення RT-DETR [99]

Таблиця 1.4 – Методи виявлення об'єктів на основі трансформерів

Метод	Рік	Тип	Точність (mAP на COCO val17)	Швидкість (FPS) (V100)	Застосування в ІТС
DETR	2020	Трансформер	~42 (ResNet-50)	~20	[100]
RT-DETR	2023	Трансформер	53,1 (ResNet-50)	50+	[101]

Традиційні методи, хоча й поступаються за точністю, досі використовуються в системах із обмеженими обчислювальними ресурсами, наприклад, у вбудованих пристроях. Натомість методи глибокого навчання

домінують у високопродуктивних ІТС, де доступні потужні GPU чи TPU. Одними із головних задач залишається адаптація моделей до різних умов – нічного часу, поганої погоди чи низької якості зображень, а також адаптація моделей глибоких нейронних мереж до застосування в системах із обмеженими обчислювальними ресурсами.

1.2. Аналіз методів багатооб'єктного трекінгу

Багатооб'єктний трекінг (Multiple Object Tracking, MOT) є важливою задачею для систем відеоспостереження, що полягає у відстеженні траєкторій множини об'єктів у відеопослідовності [102, 103-107]. На відміну від однооб'єктного трекінгу [108], МОТ вимагає ефективного управління ідентифікаторами об'єктів у разі часткових або повних перекриттів (оклюзій), коректного розв'язання конфліктів траєкторій та масштабованості для сцен з великою кількістю цілей [109, 110].

Актуальність МОТ зумовлена практичними потребами різних галузей: автономні транспортні системи потребують затримки менше 100 мс для безпечної навігації; системи масового відеоспостереження вимагають обробки понад 30 кадрів за секунду на камеру; промислові роботизовані комплекси потребують стабільного трекінгу на пристроях з обмеженими ресурсами [102].

Рис. 1.10. Приклади складних сценаріїв для багатооб'єктного трекінгу: оклюзії, щільний трафік, зміни освітлення

Сучасні підходи до MOT класифікуються на три основні категорії: трекінг на основі детекцій (tracking-by-detection) [111], спільна детекція та трекінг (joint detection and tracking) [112] та методи на основі трансформерів [113].

Парадигма tracking-by-detection є найпоширенішою, де детектор спочатку виявляє об'єкти в кожному кадрі, а потім алгоритм трекінгу асоціює детекції між послідовними кадрами для формування траєкторій [111]. Ця парадигма дозволяє використовувати найновіші досягнення в детекції об'єктів незалежно від алгоритму трекінгу.

1.2.1. Класичні методи трекінгу

Класичний метод SORT (Simple Online and Realtime Tracking) [111] став основою для більшості наступних розробок, використовуючи фільтр Калмана з моделлю постійної швидкості (constant velocity) для прогнозування положення об'єктів та угорський алгоритм для асоціації детекцій з треками (рис. 1.11). SORT моделює рух кожного об'єкта як лінійну систему, використовуючи 7-вимірний вектор стану [x, y, s, r, ẋ, ẏ, ṡ], де (x, y) – центр обмежувальної рамки у пікселях, s – площа (scale), r – співвідношення сторін (aspect ratio), а крапки позначають швидкості зміни відповідних параметрів.

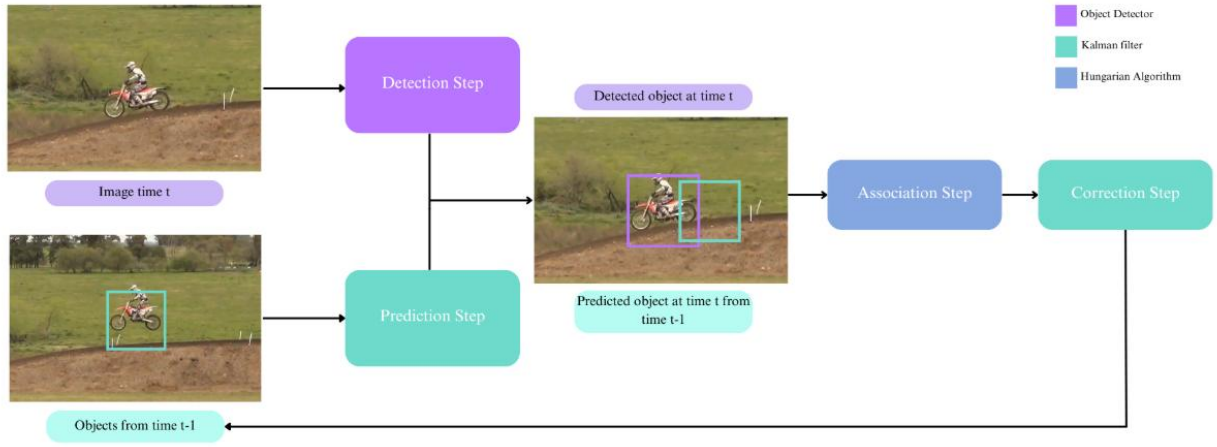


Рис. 1.11. Схематичне представлення методу трекінгу SORT [114]

Процес трекінгу в SORT включає наступні кроки:

- прогнозування нових позицій існуючих треків за допомогою фільтра Калмана;
- обчислення IoU-based cost matrix між детекціями та прогнозованими треками;
- використання угорського алгоритму для оптимального bipartite matching (зазвичай з IoU threshold $\geq 0{,}3$);
- оновлення Kalman фільтру для успішно асоційованих треків;
- ініціалізація нових треків для неасоційованих детекцій (після min_hits $\approx$ 3 успішних детекцій);
- видалення треків без детекцій протягом max_age кадрів (зазвичай 1-30).

Завдяки простоті та ефективності, SORT досягає швидкості 30-100 FPS, проте обмежується motion-only асоціацією, що призводить до високої кількості ID switches при оклюзіях [111].

DeepSORT [115] вдосконалює SORT через інтеграцію глибоких ознак вигляду (appearance features) для повторної ідентифікації (ReID), суттєво покращуючи обробку оклюзій та зменшуючи перемикань ідентичностей (ID switches). Метод використовує згорткову нейронну мережу для екстракції ембедингів вигляду та комбінує їх з передбаченням, що базується на русі через

гібридну метрику асоціації: відстань Махаланобіса для руху та косинусну відстань для вигляду [115].

Стратегія каскадного порівняння надає вищий пріоритет новим трекам, дозволяючи підтримувати ідентичність об'єктів навіть після тривалих періодів відсутності детекцій. Однак використання ReID моделі значно підвищує обчислювальну вартість (20-40 FPS), вимагає попереднього навчання на спеціалізованих датасетах та чутливе до змін освітлення [115].

1.2.2. Методи на основі глибоких ознак

Метод FairMOT [116], запропонований у 2020 році, продемонстрував ефективність спільного end-to-end навчання детекції та ReID, об'єднавши їх у єдину нейронну архітектуру. На відміну від двоетапних підходів (детекція + окремий ReID), FairMOT використовує безякірний (anchor-free) підхід для одночасного передбачення обмежувальних рамок, класів об'єктів та ембеденгів вигляду для кожного об'єкта в одному прямому проході (forward pass) [116].

Це дозволяє уникнути проблем з неузгодженістю між якірним (anchor-based) детектором і ReID-моделлю, де центр обмежувальної рамки може не збігатися з центром об'єкта, а також покращує швидкість обробки до 25-30 FPS при досягненні 73,7% MOTA на датасеті MOT17 [116].

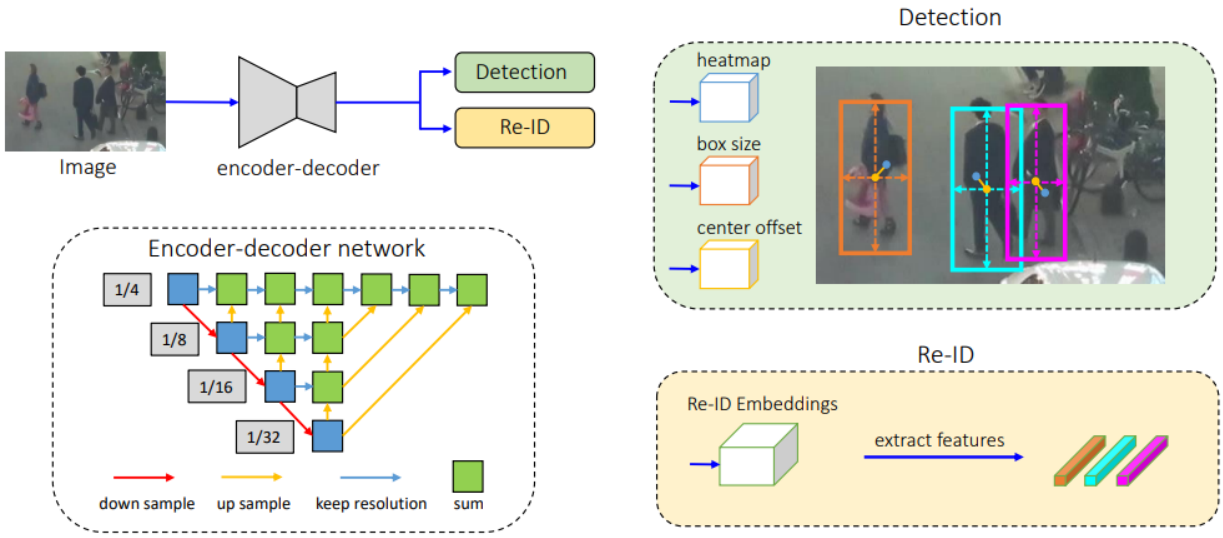


Рис. 1.12. Архітектура методу трекінгу FairMOT [116]

Подальший розвиток tracking-by-detection підходу отримав у ByteTrack [117], який запроваджує принципово новий підхід до обробки детекцій з різним рівнем достовірності. На відміну від традиційних методів, що відкидають низькопорогові детекції як хибнопозитивні, ByteTrack використовує їх для асоціації, виходячи з припущення, що низькі оцінки часто відповідають реальним об'єктам в умовах оклюзій або поганої видимості [117]. ByteTrack застосовує двоетапну стратегію асоціації на основі IoU та угорського алгоритму: (1) високопорогові детекції (score > $\theta_{high} \approx 0{,}6$) спочатку асоціюються з активними треками; (2) низькопорогові детекції ($\theta_{low} < score < \theta_{high}$, де $\theta_{low} \approx 0{,}1$) асоціюються з неасоційованими треками з першого етапу; (3) решта високопорогових детекцій ініціалізують нові треки. Фільтр Калмана використовується для прогнозування позицій треків між кадрами. Завдяки простоті та відсутності потреби у важких ReID моделях, ByteTrack досягає 80.3% MOTA на датасеті MOT17 при швидкості понад 100 FPS (на NVIDIA V100 з YOLOX-X детектором), встановивши новий стандарт балансу між точністю та ефективністю для реального часу [117].

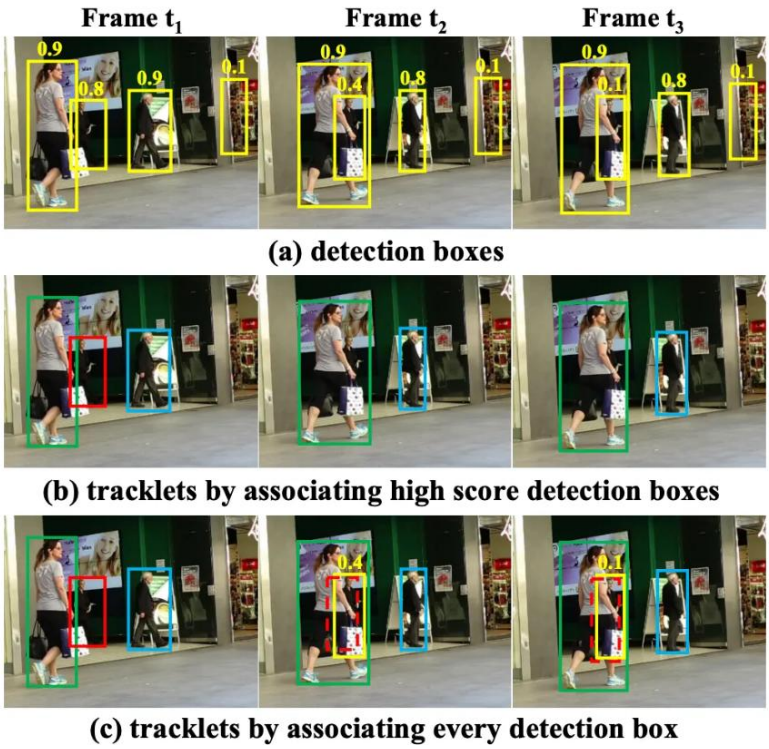


Рис. 1.13. Використання низькопорогових детекцій методом трекінга ByteTrack [117]

Подальший розвиток tracking-by-detection підходу спрямований на вирішення обмежень ByteTrack у складних сценах. OC-SORT [118] запроваджує observation-centric парадигму для адаптації до нелінійного руху, тоді як BoT-SORT [119] інтегрує ReID-ознаки та компенсацію руху камери для підвищення стабільності траєкторій у динамічних умовах.

BoT-SORT (SORT with Bag of Tricks) [119] розширює ByteTrack шляхом інтеграції компенсації руху камери (Camera Motion Compensation, CMC) та ReID-ознак на базі FastReID, досягаючи 80.5% IDF1 на MOT17 при швидкості близько 40 FPS [119]. Метод використовує техніку Bag of Tricks для покращення ключових компонентів трекінгу: гібридну асоціацію на основі IoU та близькості вигляду (appearance similarity), адаптивні пороги IoU залежно від швидкості об'єктів, компенсацію руху камери через гомографічне перетворення, фільтрацію шумних детекцій та GSI (Gaussian-smoothed interpolation) для заповнення прогалин у треках [119].

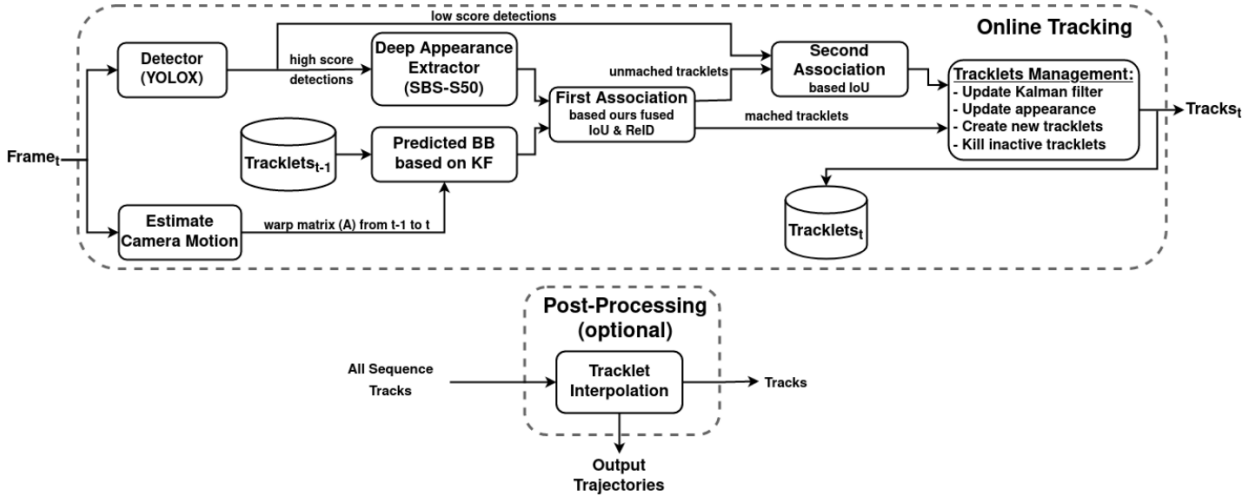


Рис. 1.14. Архітектура методу трекінгу Bot-SORT [119]

OC-SORT [118] запроваджує observation-centric парадигму відстеження, де пріоритет надається реальним детекціям над прогнозованими положеннями фільтра Калмана. На відміну від традиційних motion-centric підходів (SORT, DeepSORT), які покладаються на прогнозування стану об'єктів та порівнюють детекції з передбаченими позиціями, OC-SORT використовує Observation-Centric Momentum (OCM) - механізм корекції швидкості треку на основі реальних спостережень.

OCM обчислює швидкість безпосередньо з послідовних детекцій $v_{obs} = (bbox_t - bbox_{\{t-\Delta t\}}) / \Delta t$, що дозволяє адаптуватися до нерегулярних рухів, різких змін напрямку та різних динамік об'єктів. Для підтримки треків під час коротких оклюзій метод генерує віртуальні траєкторії на основі спостереженої швидкості та постійності швидкості (velocity consistency), що запобігає передчасному видаленню активних треків [118].

Додатково OC-SORT використовує двонаправлене зіставлення (bi-directional matching) та механізм відновлення треків (OCR - Observation-Centric Recovery) для з'єднання фрагментованих траєкторій. На датасеті MOT17 метод досягає 63,2% HOTA, 78,5% MOTA та 77,5% IDF1 при швидкості 30-40 FPS,

демонструючи особливу ефективність у сценах з нелінійним рухом та частими оклюзіями [118].

1.2.3. Методи трекінгу на основі трансформерів

Методи на основі трансформерів, які використовують механізм уваги для одночасного моделювання детекції й трекінгу [120-122].

Метод TransTrack [120] є одним із перших підходів, що адаптує архітектуру DETR для задачі multi-object tracking, формулюючи трекінг як задачу tracking-by-attention. На відміну від класичних методів, які виконують детекцію та асоціацію послідовно, TransTrack обробляє пари кадрів (поточний і попередній) через dual-encoder архітектуру і об'єднує їх ознаки в unified attention space. Decoder з Query Interaction Module (QIM) виконує одночасне передбачення нових об'єктів та уточнення існуючих траєкторій через механізм object queries, що дозволяє безпосередньо асоціювати детекції між кадрами без пост-обробки.

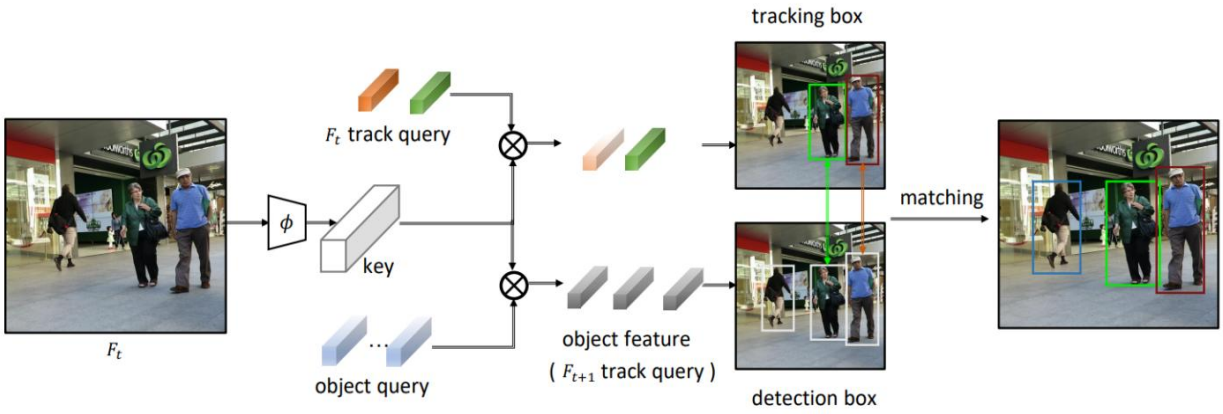


Рис. 1.15. Пайплайн методу трекінгу TransTrack [120]

Методи MOTR (MOtion TRansformer) [121] та TrackFormer [122] розвивають підхід базований на DETR для БОТ через моделювання темпоральних залежностей на рівні object queries.

MOTR [121] вводить концепцію track queries - навчуваних вбудовувань, що представляють окремі траєкторії об'єктів. Метод використовує механізм temporal query passing для поширення track queries між послідовними кадрами через encoder-decoder архітектуру, що дозволяє явно моделювати траєкторії у часі. Для детекції нових об'єктів MOTR використовує додаткові detect queries, які ініціалізуються для кожного кадру.

TrackFormer [122] застосовує автогресивний підхід до трекінгу, безпосередньо використовуючи object queries детектора з кадру t як track queries для кадру t+1 (рис. 1.16). Така стратегія query propagation забезпечує природне збереження ідентичності об'єктів без додаткових механізмів асоціації. Метод розширює DETR-архітектуру, інтегруючи темпоральну інформацію через послідовне оновлення queries між кадрами.

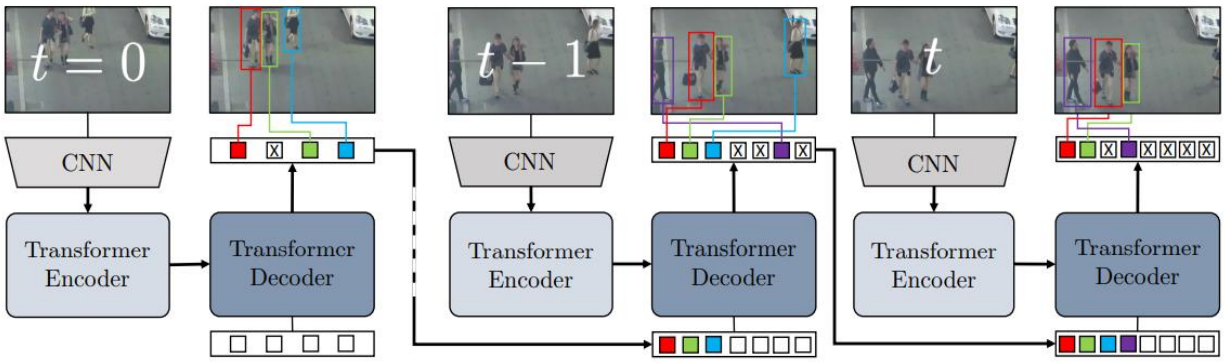


Рис. 1.16. Пайплайн методу трекінгу TrackFormer

Водночас обмеження методів на основі трансформерів включають квадратичну складність обчислень відносно кількості об'єктів та низьку швидкість (15-20 FPS), що ускладнює застосування на edge-пристроях.

1.3. Аналіз методів виявлення аномалій у відеопотоці трафіку

Виявлення аномалій у відеопотоці спрямоване на ідентифікацію подій або закономірностей, що відхиляються від типової чи очікуваної поведінки [123]. У контексті відеоспостереження за дорожнім рухом аномалії можуть включати дорожньо-транспортні пригоди, порушення правил руху (проїзд на червоний сигнал світлофора, перевищення швидкості), нетипову поведінку учасників руху та виникнення потенційно небезпечних ситуацій, зокрема появу перешкод на проїжджій частині, аварійне гальмування та інші нестандартні ситуації (рис. 1.17).

Автоматичне виявлення таких подій дозволяє підвищити безпеку дорожнього руху та ефективність управління транспортними потоками. Однак це є складною задачею через рідкісність аномальних подій (за оцінками, аномалії складають лише 0,01% часу спостереження), високу варіативність їх проявів та необхідність роботи в реальному часі [125].

Рис. 1.17. Приклади аномалій: дорожньо-транспортна пригода, перешкоджання руху пішоходів, рух забороненому напрямку, проїзд на червоний сигнал світлофору, стоянка поза зоною парковки, небезпечне маневрування [124]

Вибір досліджуваних типів аномалій ґрунтується на їхній задокументованій ролі в структурі дорожньо-транспортного травматизму. За даними Європейської ради з безпеки на транспорті (ETSC) [126], перевищення швидкості є чинником приблизно третини смертельних ДТП у Європі. А в Україні порушення швидкості руху займають перше місце (40,9% від усіх зареєстрованих ДТП з потерпілими) згідно з статистикою патрульної поліції за 2025 рік [127]. За даними Страхового інституту безпеки дорожнього руху (IIHS) [128], проїзд на червоний сигнал спричиняє 28% смертельних аварій на регульованих перехрестях. Аварійне гальмування є безпосереднім передвісником зіткнень ззаду, які становлять 32% усіх зафіксованих поліцією аварій [129]. Тривале перебування транспортного засобу в зоні перехрестя, у свою чергу, є індикатором інциденту або блокування, що порушує безперебійність руху та підвищує ризик вторинних зіткнень (Федеральне управління автомобільних доріг США, FHWA [130]). Таким чином, сукупність обраних аномалій охоплює основні категорії небезпечних подій на регульованих перехрестях.

Методи виявлення аномалій так як і методи комп'ютерного зору історично можна класифікувати на три категорії:

- традиційні методи;
- методи, що базуються на машинному навчанні;
- методи глибинного навчання.

Традиційні методи виявлення аномалій у відеопотоці спираються на низькорівневі ознаки зображення, як текстура, рух чи форма об’єктів. Поширений підхід – аналіз оптичного потоку, що оцінює вектори руху пікселів між кадрами для виявлення незвичайних траєкторій, наприклад, зворотного руху транспортних засобів як-от у роботі [131]. Недоліками цих методів є чутливість до шуму, змін освітлення та оклюзій.

Методи на основі класичного машинного навчання використовували кластеризацію чи класифікацію. У методі на основі опорних векторів (Support

Vector Machine, SVM) [132] здійснено класифікацію траєкторій на нормальні та аномальні на основі таких ознак, як швидкість, напрямок та тривалість [133]. У методі на основі кластеризації DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [134] здійснено групування траєкторій для виявлення відхилень, що дозволило ефективно ідентифікувати незвичні шляхи руху [135]. Хоча вказані методи показують хороші результати в контрольованих умовах, вони залежні від ручних ознак і погано адаптуються до нових типів аномалій.

Методи виявлення аномалій на основі глибокого навчання класифікуються на підходи, що базуються на: (1) реконструкції; (2) прогнозуванні; (3) аналізі траєкторій; (4) контрольованому навчанні класифікаторів [123, 125, 136-138].

1.3.1. Методи на основі реконструкції

Методи виявлення аномалій на основі автоенкодерів [139] базуються на припущенні, що нормальні дані лежать на низькомірному підпросторі, який модель може навчитися представляти. Автоенкодер навчається виключно на нормальних зразках, мінімізуючи помилку реконструкції $L = ||X - \hat{X}||^2$, де X - вхідні дані, $\hat{X}$ - реконструйовані дані [139].

Базова архітектура складається з двох компонентів: (1) encoder $\varphi: X \rightarrow h$, що стискає вхідні дані до низькорозмірного латентного представлення $h \in \mathbb{R}^d$ (де $d << \dim(X)$); (2) decoder $\psi: h \rightarrow \hat{X}$, що відновлює вхідні дані з латентного представлення. Bottleneck архітектура змушує модель вивчати компактні представлення нормальних шаблонів.

Під час інференсу аномалії виявляються за критерієм великої помилки реконструкції: якщо $r(x) = ||x - \psi(\varphi(x))||^2$ перевищує поріг θ (зазвичай визначається як $\mu + k \cdot \sigma$ на валідаційних даних), зразок класифікується як аномальний [139]. Припускається, що аномальні зразки, які лежать поза

вивченим підпростором нормальних даних, будуть погано реконструйовані моделлю.

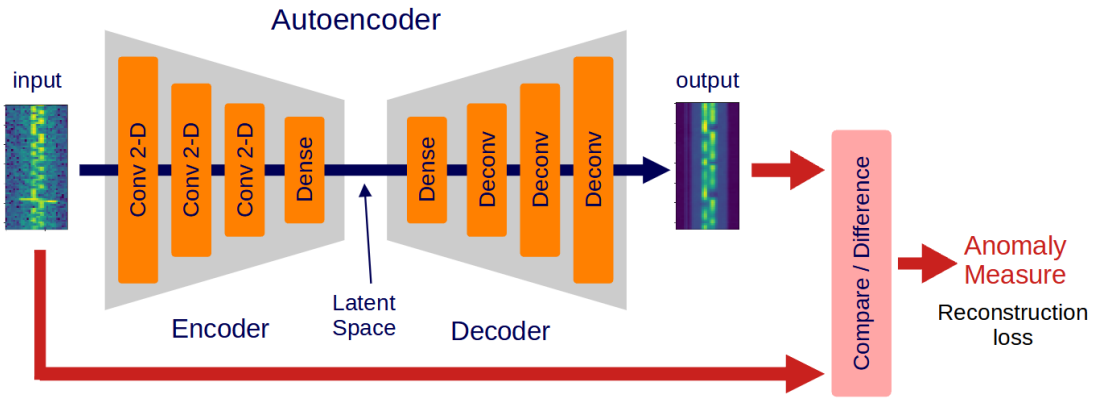


Рис. 1.18. Автоенкодер для виявлення аномалій [140]

Розширені варіанти автоенкодерів демонструють покращення над базовою архітектурою через специфічні модифікації латентного простору або архітектури. Варіаційні автоенкодери (Variational Autoencoders, VAE) [141] вводять стохастичний латентний простір з регуляризацією через KL-дивергенцію ($L = L_{rec} + \beta \cdot KL(q(z|x)||p(z))$), що забезпечує кращу узагальнюючу здатність та зменшує overfitting на тренувальних даних. Розріджені автоенкодери (Sparse Autoencoders) [142] застосовують L1-регуляризацію на активації прихованих шарів($L = L_{rec} + \lambda \cdot \Sigma|h_i|$), змушуючи модель вивчати більш селективні та інтерпретовані представлення.

Згорткові автоенкодери (Convolutional Autoencoders, CAE) [139] використовують згорткові шари в encoder (conv + pooling) та деконволюційні/upsampling шари в decoder для ефективної обробки просторової інформації у відеокадрах, зберігаючи локальні структури та паттерни (рис. 1.19).

Основною перевагою реконструкційних методів є можливість навчання без розмітки аномальних зразків - достатньо лише нормальних даних для тренування (semi-supervised підхід) [125].

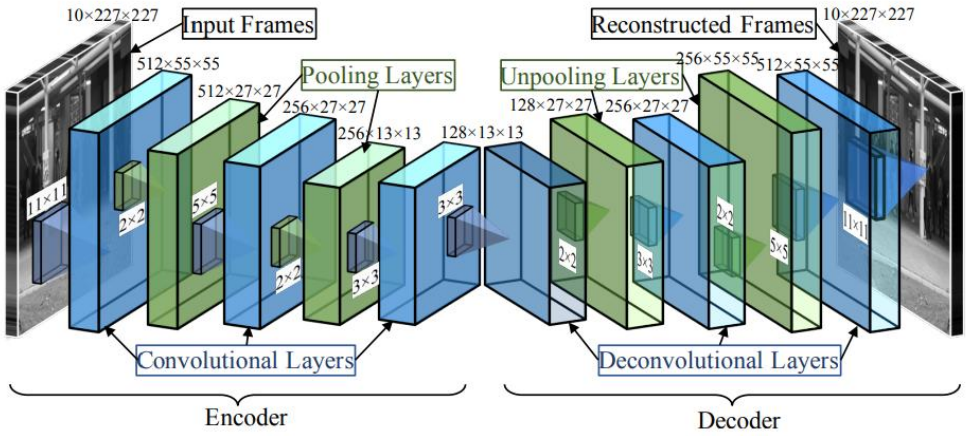


Рис. 1.19. Візуалізація структури згорткового автоенкодера [139]

Однак основним недоліком є ефект переузагальнення (overgeneralization), коли при великій складності моделі або різноманітності тренувальних даних вона може навчитися реконструювати рідкісні варіації нормальних даних, що призводить до високої кількості хибнонегативних спрацювань [125]. Для вирішення цієї проблеми використовуються архітектури з доповненою пам'яттю, змаганнєве навчання (AnoGAN)[143] або ансамблеві підходи [144].

1.3.2. Методи прогнозування

З метою усунення недоліку перегенералізації, який властивий методам реконструкції [125], запропоновані методи прогнозування, які базуються на прогнозуванні майбутніх кадрів або оптичному потоці замість реконструкції вхідних даних. Основна ідея полягає в тому, що нормальні події мають передбачувані просторово-часові патерни, тоді як аномалії порушують цю передбачуваність, призводячи до великої помилки прогнозування [139].

Формально, модель f навчається на нормальних відеопослідовностях прогнозувати наступний кадр $\hat{x}_{\{t+1\}} = f(x_{\{1:t\}})$, і аномалія виявляється, якщо помилка прогнзування error $S = \left|\left|\hat{x}_{\{t+1\}} - x_{\{t+1\}}\right|\right|$перевищує поріг θ [145].

Одним з ефективних підходів є використання Generative Adversarial Networks (GAN) для прогнозування майбутніх кадрів [146]. GAN складається з двох компонентів:

1) generator G, що створює синтетичний кадр $\hat{x}_{\{t+1\}} = G(x_{\{1:t\}}, z)$ на основі попередніх кадрів та noise vector z;

2) discriminator D, що намагається відрізнити реальні кадри від згенерованих через бінарну класифікацію D(x) → [0,1].

Навчання відбувається на нормальних відеопослідовностях через adversarial game: генератор навчається створювати реалістичні майбутні кадри, обдурюючи дискримінатор, а дискримінатор навчається розрізняти реальні та синтетичні кадри [147]. Мінімізується комбінована функція втрат:

$$L = L_{adv} + \lambda L_{rec} + \beta \cdot L_{feat}$$

де L_{adv} - adversarial loss (GAN loss), L_{rec}- reconstruction loss (MSE між передбаченим та реальним кадром), L_{feat}- feature matching loss у просторі дискримінатора [148].

Під час інференсу аномалія виявляється за критерієм високої помилки передбачення, що складається з кількох компонентів:

$$S = \alpha||x_{\{t+1\}} - G(x_{\{1:t\}})||^2 + \beta(1 - D(G(x_{\{1:t\}}))) + \gamma||\varphi(x_{\{t+1\}}) - \varphi(G(x_{\{1:t\}}))||^2$$

де φ(·) - features з проміжних шарів дискримінатора.

Специфічні GAN архітектури для детекцій аномалій у відео включають Future Frame Prediction GAN [149], що моделює як зовнішній вигляд, так і рух через оптичний потік (рис. 1.20) та GANomaly [148], що використовує кодер-декодер-кодер структуру для кращого видобування ознак [125].

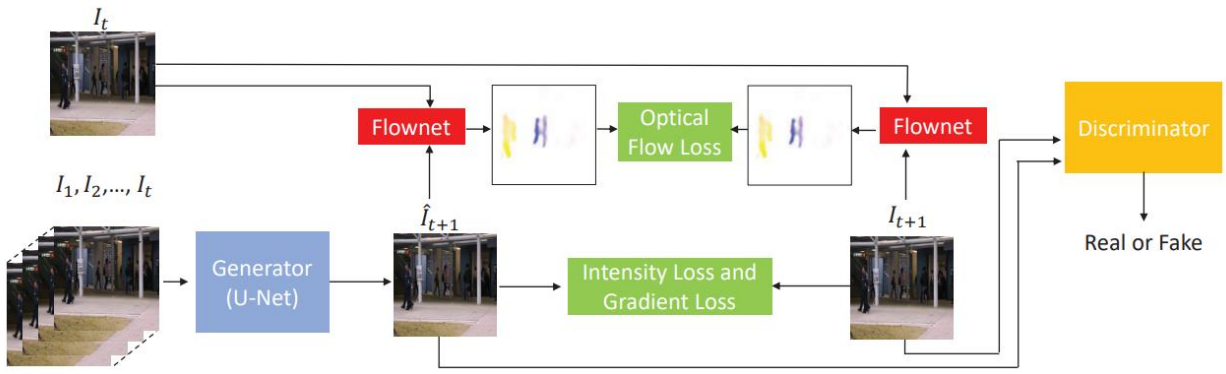


Рис. 1.20. Пайплайн методу виявлення аномалій на основі прогнозування майбутніх кадрів

Альтернативним підходом є використання рекурентних нейронних мереж (LSTM, GRU) для моделювання темпоральних залежностей [150]. Convolutional LSTM (ConvLSTM) [151-153] комбінує згорткові шари для просторової екстракції ознак з LSTM для темпорального моделювання, дозволяючи прогнозувати майбутні кадри або оптичний потік на основі послідовності попередніх кадрів.

Попри переваги у виявленні темпоральних аномалій, методи прогнозування мають кілька обмежень [125]:

1. Чутливість до руху камери: дрібні зсуви або вібрації камери призводять до великих помилок передбачення навіть за відсутності справжніх аномалій. Потрібна стабілізація відео або механізм компенсації руху камери [154].

2. “М'які” аномалії: повільні або градуальні зміни (наприклад, поступове сповільнення перед зупинкою) важко виявити, оскільки модель може адаптуватися до них через ковзне вікно [155].

3. Обчислювальна складність: тренування GAN є нестабільним (mode collapse, затухання градієнтів), а передбачення для кожного кадру вимагає значних ресурсів порівняно з простою реконструкцією [146].

4. Чутливість до зміни розподілу: зміни погодних умов, освітлення або часу доби призводять до погіршення якості прогнозування навіть для нормальних подій, вимагаючи адаптації домену або fine-tuning [156].

5. Компроміс часового вікна: вибір кількості попередніх кадрів для передбачення (розмір часового вікна) впливає на баланс між довготривалим контекстом та обчислювальною вартістю [157].

Порівняно з методами реконструкції, методи прогнозування краще виявляють швидкі темпоральні аномалії (раптові зміни руху, колізії), але є більш чутливими до нестаціонарності середовища та вимагають більших обчислювальних ресурсів [145]. Для практичних систем відеоспостереження за дорожнім рухом часто використовують гібридні підходи, що комбінують прогнозування разом з аналізом траєкторій для балансу між точністю та робастністю [158].

1.3.3. Методи аналізу траєкторій

Методи, що базуються на аналізі траєкторій об'єктів, використовують детектор для виявлення об'єктів, трекер для побудови траєкторій та аналізатор для виявлення аномальних патернів поведінки [159]. На відміну від reconstruction-based та prediction-based підходів, які працюють безпосередньо з пікселями, trajectory-based методи оперують високорівневими представленнями руху об'єктів у просторі та часі, що забезпечує кращу інтерпретовність та можливість роботи в реальному часі [160].

Загальний алгоритм таких методів включає:

1. Детекцію об'єктів дорожнього руху (транспортних засобів, пішоходів) у кожному кадрі;

2. Багатооб'єктний трекінг для формування траєкторій з унікальними ідентифікаторами;

3. Нормалізацію координат через гомографічне перетворення H: $\mathbb{R}^2 \rightarrow \mathbb{R}^2$ для переходу від координат зображення (u,v) до світової системи координат (X,Y): $[X, Y, 1]^T = H \cdot [u, v, 1]^T$, де H - матриця 3×3 з 8 ступенями свободи [161];

4. Обчислення кінематичних характеристик руху: швидкість v = Δd/Δt (де Δd обчислюється у світових координатах), прискорення a = Δv/Δt, напрямок θ = atan2(Δy, Δx), кривизна траєкторії κ;

5. Виявлення аномалій на основі правил або статистичного аналізу [162].

Деякі дослідження використовують швидкі детектори (часто з YOLO-серії) та багатооб'єктний трекінг, як-от [163-165]. Це дозволяє здійснювати виявлення аномалій у реальному часі. Перевагами цих методів є інтерпретованість на рівні маневрів (наприклад, заборонений поворот, рух у зворотному напрямку, різке гальмування тощо). Недоліками є погіршення якості трекінгу в щільному транспортному потоці та проблеми з асоціацією об'єктів при оклюзіях.

Для виявлення аномалій застосовуються різні підходи:

- Виявлення статистичних відхилень від нормальних траєкторій за допомогою методів кластеризації або навчання словника патернів [166];

- Детекція порушень дорожніх правил (проїзд на червоний, неправильний поворот, рух у зворотному напрямку) на основі експертних правил [166];

- Оцінка швидкості транспортних засобів та виявлення перевищення швидкості або аварійного гальмування [159].

Для точного вимірювання швидкості необхідне калібрування камери, що включає визначення внутрішніх параметрів (фокусна відстань, головна точка) та зовнішніх параметрів (положення та орієнтація камери відносно дорожньої площини) [167, 168].

Гомографічне перетворення дозволяє перетворити координати з площини зображення на площину дороги, що є необхідним для коректного обчислення реальних відстаней та швидкостей [168]. При ручному калібруванні користувач позначає на зображенні кілька точок на дорозі (зазвичай, 4-8 точок) з відомими реальними координатами [169] (рис. 1.21). При автоматичному калібруванні можуть використовуватися методи, що базуються на виявленні точок сходу та оцінці масштабу на основі розмірів транспортних засобів [170,

171], або на основі ключових точок транспортних засобів (колеса, бампери) [172] або їх обмежувальних прямокутників з припущенням відомої форми та розмірів транспортних засобів [173].

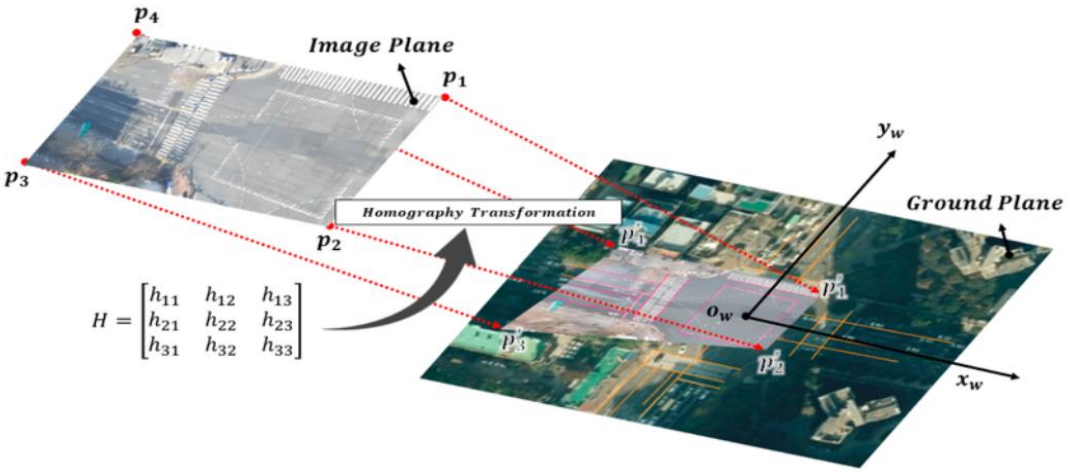


Рис. 1.21. Калібрування камери та гомографічне перетворення [174]

Перевагами методів на основі аналізу траєкторій є:

- інтерпретованість на рівні маневрів та конкретних порушень правил;

- можливість роботи в реальному часі при використанні ефективних детекторів та трекерів;

- не потрібні великі обсяги розмічених даних з аномаліями для навчання, достатньо нормальних траєкторій для статистичних методів або експертних правил для підходів базованих на правилах;

- гнучкість та zero-shot властивості: можливість швидкого додавання нових типів аномалій через налаштування правил без перенавчання моделей;

- модульність: можна окремо покращувати детектор, трекер та аналізатор, що спрощує розробку та зневадження.

Недоліками є:

- залежність від якості детекції та трекінгу об'єктів: помилки детекції призводять до помилок трекінгу, що в свою чергу призводить до неправильного виявлення аномалій;

- погіршення якості трекінгу в щільному транспортному потоці та при оклюзіях: збільшується кількість перемикань ідентичностей;

- необхідність калібрування камери для точного вимірювання швидкості;

- чутливість до якості відеопотоку, кута огляду та стабільності камери.

1.4. Проблеми застосування методів у реальному часі

Незважаючи на значні досягнення в розробці методів детекції, трекінгу та виявлення аномалій, їх практичне застосування в системах відеоспостереження дорожнього руху стикається з низкою проблем [125].

Обчислювальна складність. Більшість високоточних моделей детекції (Faster R-CNN, EfficientDet) та трекінгу (з ReID-модулями) характеризуються значною обчислювальною складністю, що робить їх непридатними для використання на пристроях з обмеженими ресурсами, таких як edge-камери відеоспостереження. Водночас, легковагі моделі часто демонструють недостатню якість детекції в складних умовах (нічний час, погана погода, низька якість зображень).

Обробка оклюзій. Методи багатооб'єктного трекінгу недостатньо ефективно обробляють ситуації часткових та повних оклюзій об'єктів, що призводить до втрати ідентичності (ID switches) та фрагментації траєкторій. Використання глибоких ознак повторної ідентифікації (ReID) покращує ситуацію, але суттєво збільшує обчислювальні витрати.

Адаптація до умов зйомки. Моделі, навчені на стандартних датасетах з хорошою якістю зображень та освітленням, часто показують погіршення якості при застосуванні до відео з реальних камер відеоспостереження, що характеризуються низькою роздільною здатністю, поганим освітленням, розфокусуванням та іншими артефактами.

Виявлення аномалій. Існуючі методи виявлення аномалій у відеопотоці трафіку або базуються на ресурсомістких моделях реконструкції/прогнозування, або вимагають значних обсягів розмічених даних для навчання класифікаторів. Методи на основі аналізу траєкторій є більш інтерпретованими та ефективними, але потребують надійної детекції та трекінгу об'єктів.

Датасети. На сьогодні існує кілька доступних датасетів виявлення аномалій, створених на основі даних із камер спостереження за дорожнім рухом, таких як CADP [175] і TU-DAT [176]. Однак ці датасети переважно зосереджені на виявленні дорожньо-транспортних пригод (ДТП). Крім того, якість відео в таких наборах даних, зокрема в CADP, часто є недостатньою (нижче 720p), що ускладнює ефективне застосування методів, заснованих на детекції об'єктів.

Балансування точності та швидкості. Основний виклик полягає в досягненні оптимального балансу між точністю детекції/трекінгу та швидкістю обробки, що є важливим для систем реального часу. Необхідні підходи, що забезпечують високу точність при збереженні можливості роботи на обмежених обчислювальних ресурсах.

1.5. Висновки до розділу

На основі проведеного аналізу літературних джерел можна зробити наступні висновки:

1. У сфері детекції об'єктів дорожнього руху спостерігається перехід від класичних методів (SIFT, HOG, Haar Cascades) до підходів на основі глибокого навчання (R-CNN, YOLO, SSD, трансформери). Сімейство архітектур YOLO демонструє найкращий баланс між точністю та швидкістю для застосувань реального часу, проте існує потреба в подальшій оптимізації для роботи на пристроях з обмеженими ресурсами.

2. Методи багатооб'єктного трекінгу еволюціонували від простих підходів (SORT) до складніших алгоритмів з використанням ReID-ознак (DeepSORT, BoT-SORT) та спостереження-центричної парадигми (OC-SORT). ByteTrack показав можливість ефективного трекінгу без ReID-моделей, проте проблема обробки оклюзій та перемикань ідентичності залишається актуальною.

3. Для виявлення аномалій у відеопотоці трафіку найбільш перспективними є методи на основі аналізу траєкторій об'єктів, що поєднують детектор, трекер та аналізатор. Такі методи забезпечують інтерпретованість результатів та можливість роботи в реальному часі, але вимагають надійної детекції та трекінгу.

4. Основними проблемами залишаються: (а) високі обчислювальні вимоги точних моделей; (б) недостатня стійкість до оклюзій без ReID; (в) погіршення якості в складних умовах зйомки; (г) обмежена кількість спеціалізованих датасетів.

На основі проведеного аналізу сформульовано такі задачі дослідження:

1. Здійснити детальний аналіз сучасних підходів до детекції об'єктів дорожнього руху, багатооб'єктного трекінгу та виявлення аномалій у відеопотоці, приділивши особливу увагу методам на основі глибокого навчання, їхнім перевагам та недолікам.

2. Розробити ресурсоефективну модель детекції об'єктів дорожнього руху, що забезпечує прийнятну точність виявлення та можливість розгортання на Edge-пристроях з обмеженими обчислювальними ресурсами.

3. Удосконалити метод багатооб'єктного трекінгу для підвищення точності асоціації об'єктів та стійкості до оклюзій без використання ресурсомістких моделей повторної ідентифікації.

4. Розробити метод виявлення аномалій у відеопотоці дорожнього руху на основі аналізу траєкторій об'єктів з можливістю автоматичного виявлення різних типів порушень правил дорожнього руху.

5. Розробити архітектуру комплексної інформаційної технології аналізу відеопотоку дорожнього руху, що інтегрує створені моделі та методи детекції, трекінгу та виявлення аномалій у Edge-Cloud архітектурі.

6. Здійснити програмну реалізацію розроблених методів та провести експериментальні дослідження на реальних задачах і датасетах для підтвердження ефективності запропонованих рішень.

Основні результати розділу опубліковані автором у роботах [1-3, 5].

РОЗДІЛ 2. РОЗРОБКА МОДЕЛІ ТА МЕТОДІВ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ВІДЕОПОТОКУ ДОРОЖНЬОГО РУХУ

2.1 Модель детекції об'єктів дорожнього руху

За основу для удосконалення у роботі використана архітектура моделі YOLOv8. Вона є сучасною одноетапною моделлю детекції об'єктів, що поєднує ефективність та високу точність завдяки використанню вдосконаленої модульної архітектури, що ділиться на три блоки: backbone – відповідальний за вилучення ознак з зображення, neck – відповідальний за об'єднання ознак, та head відповідальний за генерацію обрамляючих прямокутників навколо об’єктів різних розмірів (рис. 2.1).

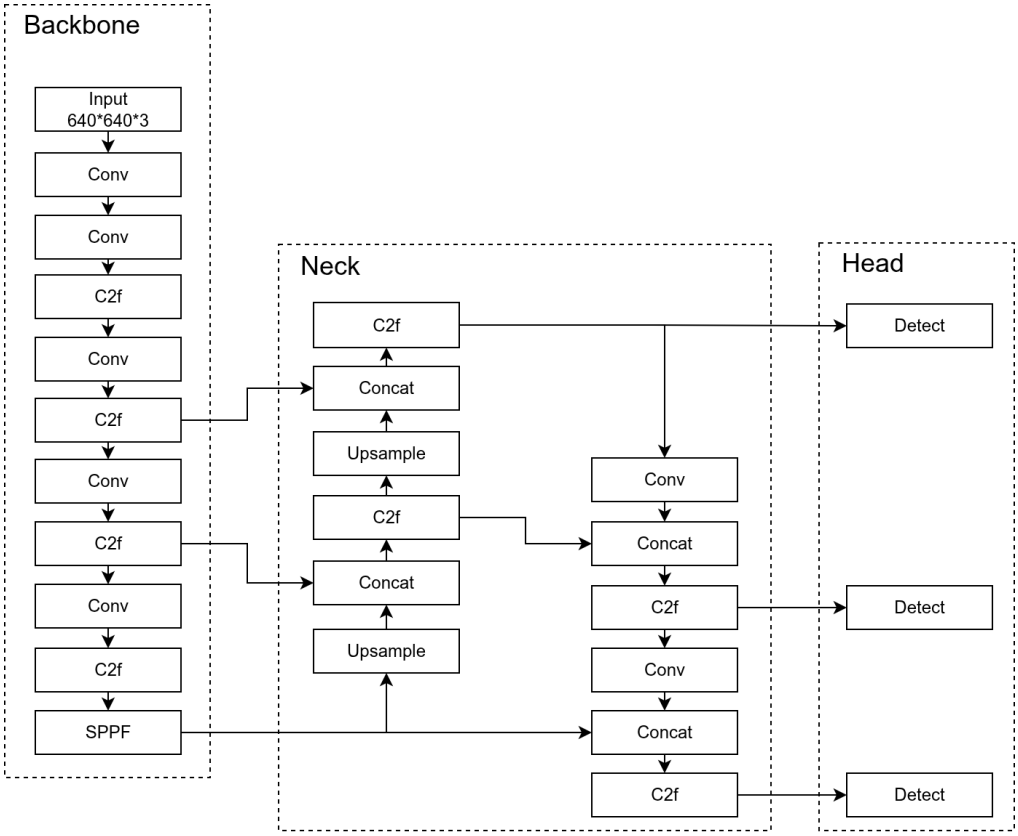


Рис. 2.1. Базова архітектура мережі YOLOv8

Neck-частина моделі YOLOv8 базується на комбінації FPN (Feature Pyramide Network) [177] і PANet (Path Aggregation Network), [178] що дозволяє ефективно агрегувати ознаки з різних рівнів. У FPN об'єкти виявляються окремо та незалежно на різних рівнях масштабу. Це, на думку авторів PANet, може призвести до дублювання прогнозів та не використання інформації з карт ознак іншого масштабу. Натомість PAN об'єднує дані з усіх шарів, що призводить до покращення виявлення об'єктів. Схематичне представлення FPN та PANet наведено на рисунку 2.2.

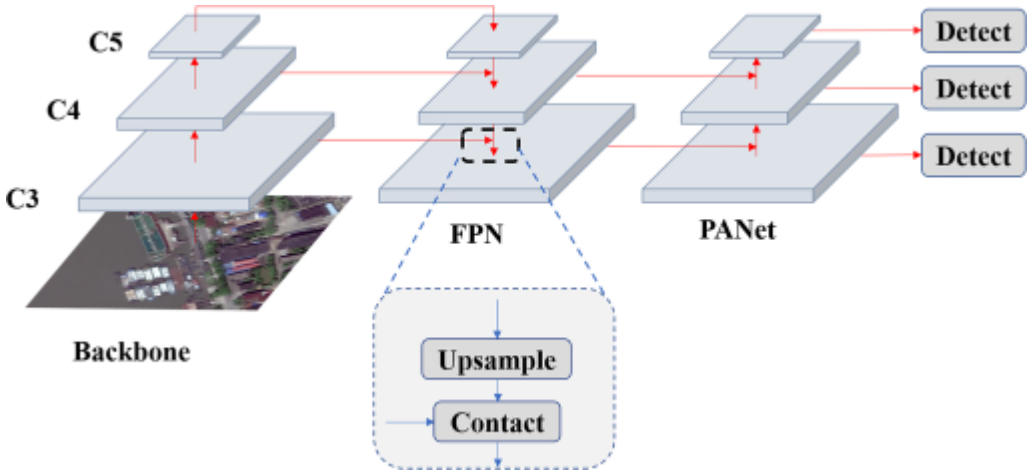


Рис. 2.2. FPN та PANet [179]

Основою мережі є C2f-блоки, які забезпечують компактність та глибину представлення ознак. C2f блок побудований на основі двох паралельних потоків: першого, що скорочує розмірність (Conv 1×1), та другого, що проводить глибоку послідовну обробку через серію залишкових (Bottleneck) блоків. Обидва потоки об'єднуються конкатенацією, забезпечуючи багатомасштабне представлення ознак (рис. 2.3).

Така архітектура дозволяє ефективно: збереження достатнього рівня деталей ознак для точної детекції об'єктів малого розміру; скорочення

обчислювального навантаження порівняно з глибшими мережами; поліпшення поширення градієнтів завдяки залишковим з'єднанням.

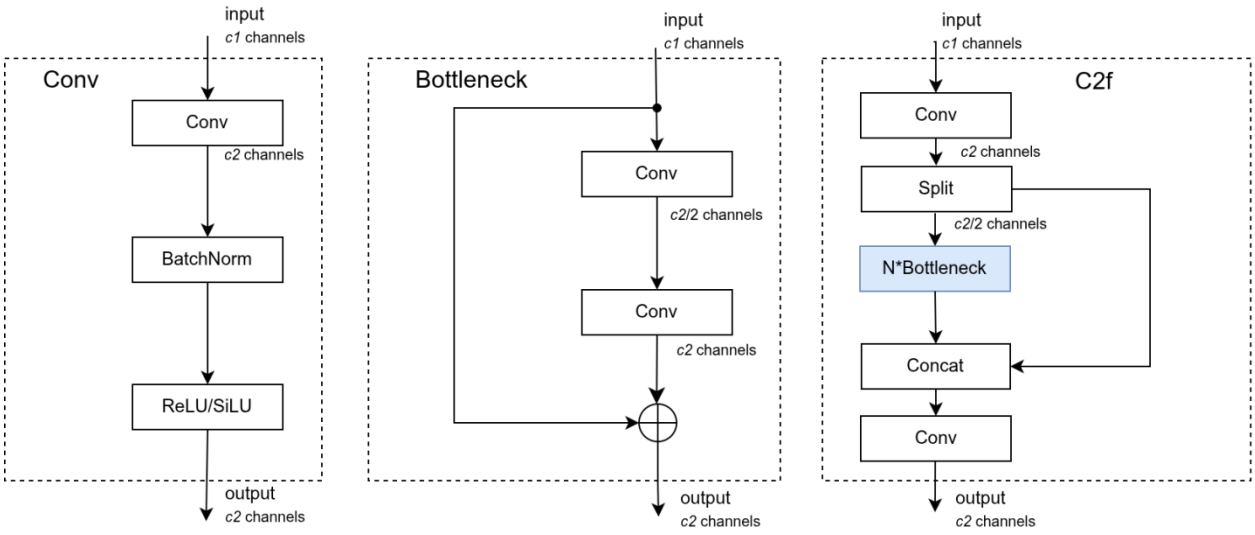


Рис. 2.3. Структура модуля C2f

У частині head застосовується безякірний (anchor-free) підхід, який у порівнянні з традиційними методами виявлення, що використовують попередньо визначені якорі, спрощує процес навчання та підвищує узагальнювальну здатність моделі.

На відміну від YOLOv5 і попередніх версій, що розділяють зображення на сітку та для кожної комірки генерують множину якорів (зазвичай 3-9) різних масштабів і аспектних співвідношень, YOLOv8 прямо передбачає локалізацію об'єктів без посилань на попередньо визначені шаблони. Для кожної комірки сітки модель одночасно передбачає:

- впевненість у наявності об'єкта у цій комірці p_{obj};
- абсолютні координати обмежувального прямокутника [x,y,w,h] без масштабування якорів;
- логіти класів s_k для кожного з K класів об'єктів.

Це забезпечує кілька переваг. По-перше, відсутня проблема невідповідності якорів – явище, коли об'єкт значно відрізняється від доступних якорів, що призводить до гіршого навчання моделі. По-друге, anchor-free підхід зменшує кількість гіперпараметрів (немає необхідності вручну настройки масштабів і аспектних співвідношень якорів), що сприяє більш стабільному навчанню. По-третє, модель показує вищу узагальнювальну здатність на нових датасетах, особливо для об'єктів нестандартних розмірів.

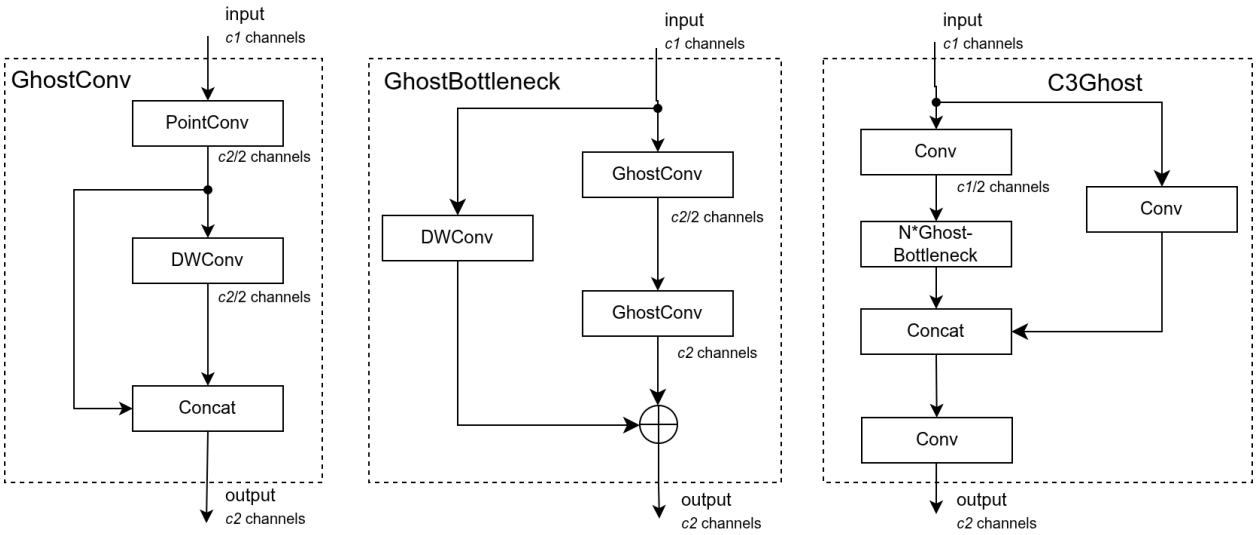
Крім того, YOLOv8 використовує вдосконалену функцію втрат CIoU та підтримує масштабування архітектури під різні ресурси, що робить її придатною для широкого спектру застосувань у реальному часі.

Втрати локалізації визначаються через функція втрат CIoU визначається як:

$$L_{box} = 1 - IoU + \left(\frac{\rho^2(c^{pred}, c^{gt})}{c^2}\right) + \alpha \times v, \quad (2.1)$$

де IoU – Intersection over Union; $\rho^2(c^{pred}, c^{gt})$ – евклідова відстань між центрами обмежуючих прямокутників; c – діагональ найменшого прямокутника, що охоплює обидва обмежуючих прямокутника; v – міра подібності співвідношень сторін; α – ваговий параметр.

Для зменшення обчислювальної складності та кількості параметрів моделі YOLOv8 при збереженні продуктивності запропоновано використати модуль C3Ghost, що є модифікацією блоку C3, використаного в YOLOv5 [56]. C3Ghost базується на Ghost-згортках [180], які зменшують надлишковість ознакових карт. Замість повної згортки створюється обмежена кількість первинних ознакових карт стандартною згорткою, які потім трансформуються у більше “ghost“-ознак за допомогою економних глибинних згорток (3×3 або 5×5). Це дозволяє генерувати більше ознак із меншими витратами ресурсів. У C3Ghost стандартні Bottleneck-блоки замінено на GhostBottleneck. Графічне представлення Ghost-згорток, Bottleneck-блоків та модуля C3Ghost наведено на рисунку 2.4.



а) б) в)

Рис. 2.4. Ghost-згортки - а, GhostBottleneck-блоки - б та модуль C3Ghost

Для покращення розпізнавання об’єктів дорожнього руху в складних умовах (ніч, дощ, низька якість зображень) до C3Ghost пропонується інтегрувати модуль уваги CBAM [181]. Модуль CBAM призначений для посилення як канальної, так і просторової уваги. Канальна увага дозволяє моделі визначати найбільш інформативні канали та надавати їм більшу вагу, тоді як просторова увага фокусується на ключових областях зображення, ігноруючи менш релевантні. Зокрема, канальна увага може допомогти краще розпізнавати важливі ознаки транспортних засобів, такі як форма чи текстура. Просторова увага, в свою чергу, може зменшити вплив фону (наприклад, дерев або неба), концентруючись на самих автомобілях, що буде сприяти підвищенню якості виявлення.

Модуль CBAM приймає вхідну карту ознак $F \in R^{C\times H\times W}$і генерує уточнену карту ознак $F'' \in R^{C\times H\times W}$.

Процес обробки виглядає наступним чином:

$$F' = M_c(F) \otimes F$$

$$F'' = M_c(F') \otimes F', \quad (2.2)$$

де $M_c \in R^{C\times1\times1}$ – маска канальної уваги; $M_s \in R^{1\times H\times W}$– маска просторової уваги; - $\otimes$ – поелементне множення.

Маски $M_c(F)$ та $M_c(F')$ нормуються через сигмоїдну функцію, щоб значення маски були в межах [0, 1].

Маска канальної уваги визначається як:

$$M_c(F) = \sigma(MLP(AvgPool(F)) + MLP(MaxPool(F))) \quad (2.3)$$

де $AvgPool(F) \in R^{C\times1\times1}$ – глобальне усереднене об'єднання (pooling); $MaxPool(F) \in R^{C\times1\times1}$– глобальне максимальне об'єднання (pooling); MLP – багатошаровий перцептрон зі спільними вагами для обох шляхів; σ – сигмоїдна функція активації.

MLP складається з двох повнозв'язних шарів з коефіцієнтом зменшення r (зазвичай r=16):

$$MLP(x) = W_1(ReLU(W_0) \quad (2.4)$$

де$W_0 \in R^{C/r\times C}$, $W_1. \in R^{C\times C/r}$,

ReLU (Rectified Linear Unit) – функція активації при якій негативні значення обнуляються, а позитивні пропускаються лінійно [109]: ReLU(z)= max(0,z).

Маска просторової уваги визначається як:

$$M_s(F') = \sigma(Conv^{7\times7}([AvgPool(F'); MaxPool(F')]) \quad (2.5)$$

де $AvgPool(F') \in R^{1\times H\times W}$– усереднення вздовж каналів; $MaxPool(F') \in R^{1\times H\times W}$– максимум вздовж каналів; [·; ·] – конкатенація вздовж каналів; $Conv^{7\times7}$– згортка з ядром 7×7; σ – сигмоїдна функція активації.

Структура CBAM модуля представлена на рис 2.5.

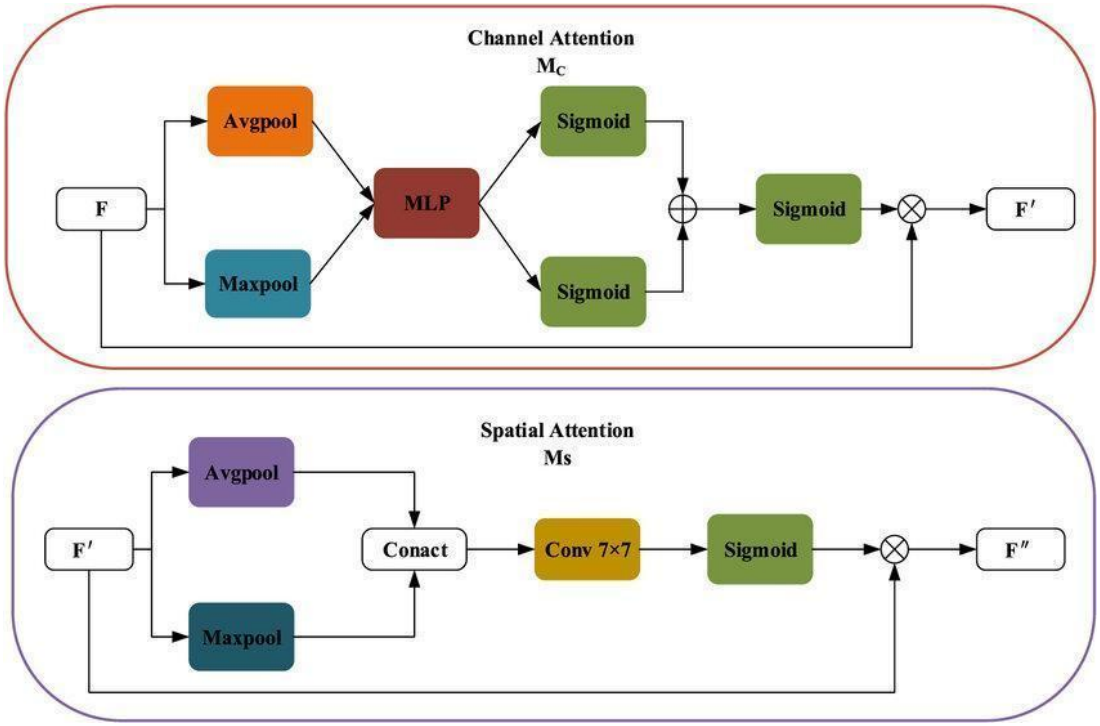


Рис. 2.5. Структура CBAM модуля

Пропонується додати модуль CBAM у C3Ghost після конкатенації перед останньою згорткою. Це дозволить фільтрувати надлишкову інформацію після об’єднання ознакових карт та компенсувати втрати від економних Ghost-згорток, підсилюючи релевантні ознаки. Графічне представлення запропонованого модуля C3Ghost-CBAM наведено на рисунку 2.6.

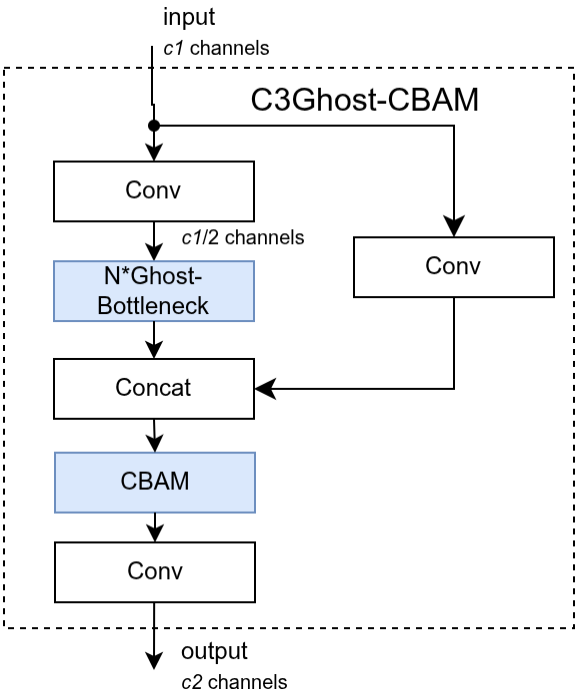


Рис. 2.6. C3Ghost-CBAM модуль

Таблиця 2.1 – Порівняння модулів нейромережевої моделі виявлення

Модулі моделі\ Характеристика	C2f	C3	C3Ghost	C3Ghost-CBAM
Параметри моделі (для 64 каналів)	~2,5K	~2,5K	~2,5K	~2,5K
FLOPS	~500	~420	~150	~180
Придатність для edge	Середня	Хороша	Відмінна	Відмінна
Стійкість до шуму	Хороша	Хороша	Задовільна	Відмінна

Пропонується модифікувати базову архітектуру YOLOv8 шляхом додавання блоків C3Ghost-CBAM замість C2f у Neck частину а також полегшеної згортки GhostConv. Крім того для спрощення архітектури пропонується замінити один модуль C2f у Backbone частину перед модулем SPPF. Запропонована модифікація YOLOv8 представлена на рис. 2.7.

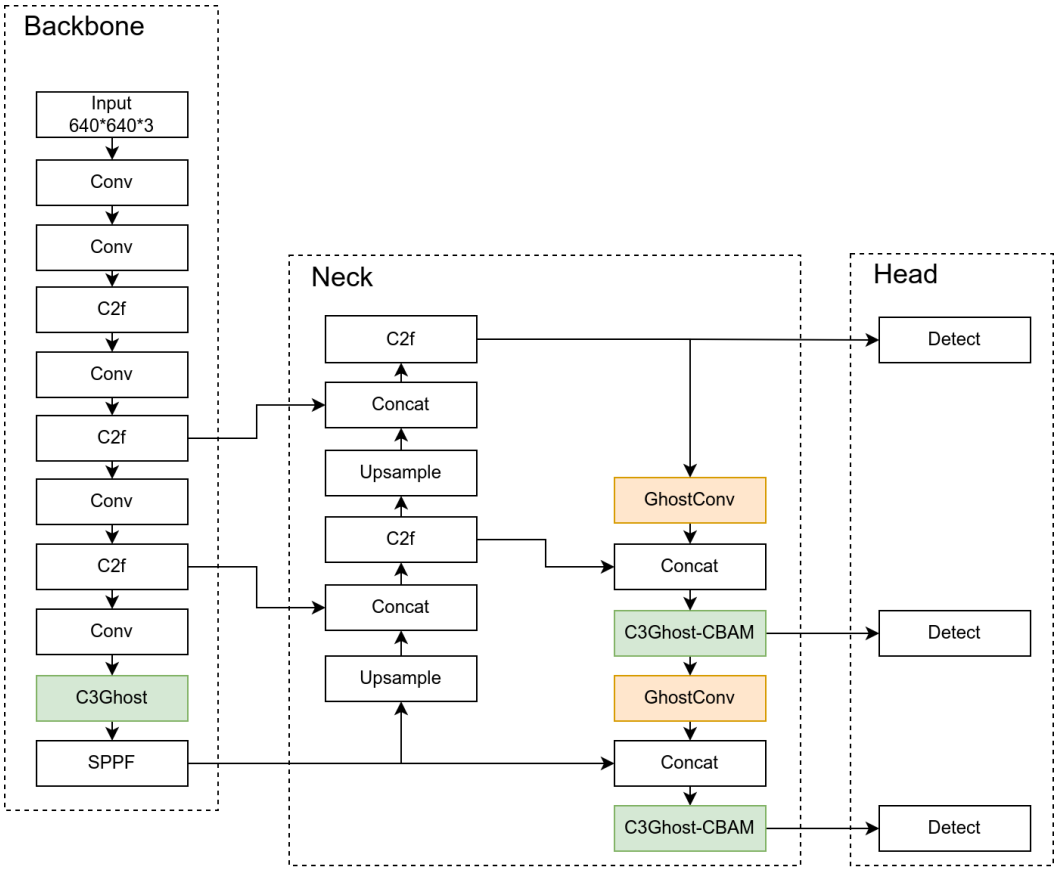


Рис. 2.7 – Запропонована архітектура нейронної мережі детектора об'єктів

Для оцінки ефективності запропонованого методу було створено набір зображень з камер відеоспостереження трафіку, які отримані з відкритих джерел. Датасет складає 2000 зображень, які були поділені на тренувальні, валідаційні та тестові у співвідношенні - 7:2:1. Велика частина зображень характеризуються складними умовами, зокрема, отриманими в різний час доби, за умов поганої погоди чи характеризуються низькою якістю. Типові приклади зображень отриманого датасету наведені на рис. 2.8.

Рис. 2.8. Приклади зображень створеного датасету

Для експериментів була використана така апаратна конфігурація – графічна карта NVIDIA GeForce GTX 1080 з 8 ГБ, фреймворк глибинного навчання PyTorch версії 2.5.1, Python версії 3.11, CUDA версії 11.0 та операційна система Ubuntu 20.04.

Максимальна кількість епох для тренування набору даних становить 100, з першими трьома епохами для розігріву. Для регулювання швидкості навчання використовується оптимізатор AdamW з початковою швидкістю навчання 0,02. Через наявність об'єктів малого розміру у наборі даних, а також для забезпечення балансу між якістю та швидкістю виявлення у пристроях з обмеженими ресурсами розмір кожного зображення пропонується нормалізувати до 640×640 пікселів.

Для оцінки ефективності моделі виявлення транспортних засобів в цій роботі використовувалася метрика mean Average Precision (mAP). Обчислювальну складність моделі оцінювали за такими показниками, як кількість параметрів і обсяг операцій з плаваючою комою (FLOPs). Оскільки в дослідженні розглядався лише один клас (транспортний засіб), значення mAP еквівалентне значенню Average Precision (AP) для цього класу.

Метрика AP визначається як площа під кривою Precision-Recall і розраховується наступним чином:

$$\mathrm{AP} = \int_0^1 \mathrm{Precision}(\mathrm{Recall})\mathrm{d}(\mathrm{Recall}), \quad (2.6)$$

Precision відображає частку правильних детекцій серед усіх позитивних передбачень моделі, а Recall – частку правильних детекцій серед усіх реальних об’єктів класу “транспортний засіб“. Рівняння для обчислення Precision та Recall представлені на у формулах (2) та (3) відповідно:

$$\mathrm{Precision} = \frac{\mathrm{TP}}{\mathrm{TP+FP}}, \quad (2.7)$$

$$\mathrm{Recall} = \frac{\mathrm{TP}}{\mathrm{TP+FN}}, \quad (2.8)$$

де TP позначає кількість правильно виявлених позитивних зразків, FN позначає кількість неправильно виявлених негативних зразків, а FP вказує на кількість неправильно виявлених позитивних зразків.

У дослідженні використовувалися два показники mAP: mAP@0,5, що відповідає значенню mAP при порозі IoU (Intersection over Union), рівному 0,5, та mAP@0,5:0,95, яке є середнім значенням mAP, розрахованим для порогів IoU від 0,5 до 0,95 з кроком 0,05.

На основі створеного набору зображень, були проведені експерименти з виявлення транспортних засобів у зображеннях з камер відеоспостереження трафіку, для дослідження впливу кожного доданого модуля на загальну продуктивність моделі. За базову взята оригінальна модель YOLOv8n, модель 1 включала модуль C3Ghost замість модуля C2f як у backbone так і у neck відповідно до рис. 2.6. У моделі 2 окрім C3Ghost модулів з моделі 1 замість двох Conv блоків у neck були використані GhostConv блоки, у моделі 3 були використані GhostConv у neck та доданий CBAM у C3Ghost модулі лише у neck. Зміни показників оцінювання для чотирьох моделей були кількісно досліджені та порівнянні. Результати оцінювання нових моделей на створеному наборі

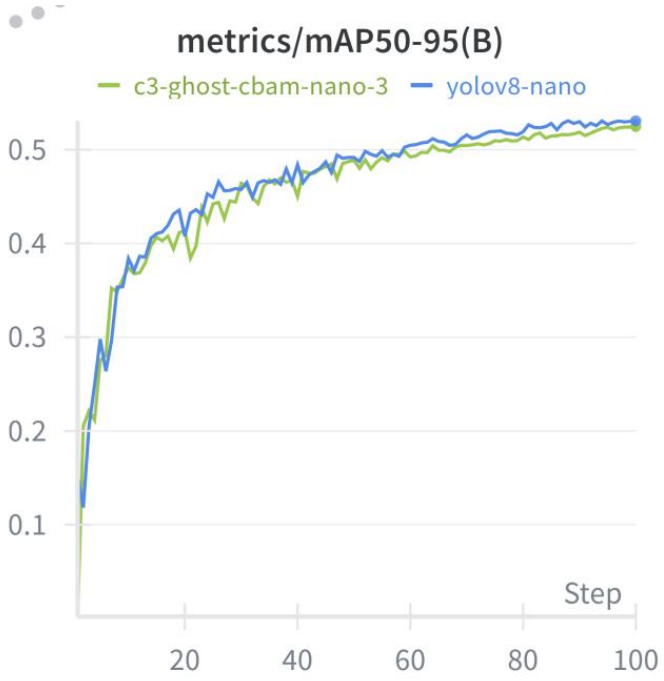
даних наведені в табл. 2.2, де найкращі результати для кожного показника оцінювання були позначені жирним.

Таблиця 2.2 – Результати експериментів з вилученням модулів

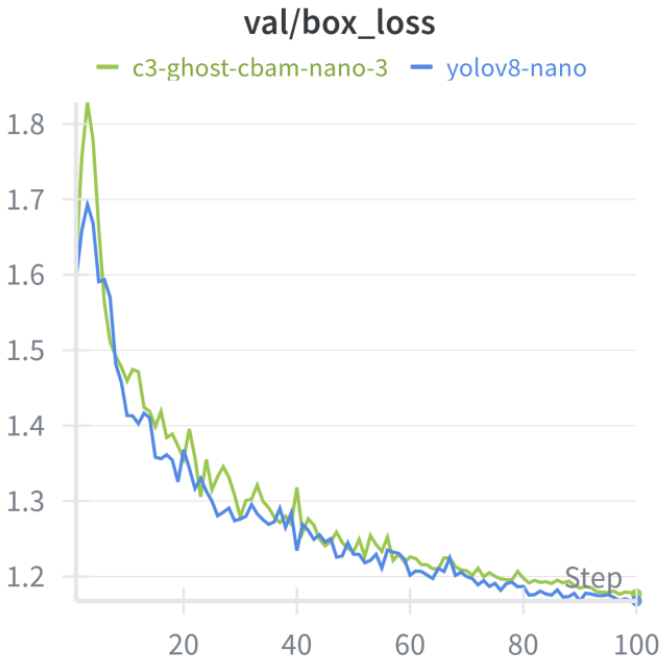
Модель	C3Ghost	GhostConv	CBAM	mAP@(0,50-0,95)	mAP@0,50	Кількість параметрів, млн	GFLOPs
YOLOv8-nano	-	-	-	**0,531**	**0,817**	3,0	8,1
1	+	-	-	0,518	0,809	2,3	7,3
2	+	+	-	0,515	0,805	**2,2**	**7,2**
3	+	+	+	0,525	0,810	2,3	7,3

Згідно з табл. 2.2, базова модель YOLOv8-nano показала найвищу якість виявлення: mAP@(0,50:0,95) – 0,531, mAP@0,50 – 0,817, при 3,0 млн параметрів і 8,1 GFLOPs. Модель 2 досягла найменшої обчислювальної складності: 2,2 млн параметрів, 7,2 GFLOPs, але з нижчим mAP@(0,50:0,95) – 0,515 і mAP@0,50 – 0,805. Модель 3 забезпечила оптимальний баланс: mAP@(0,50:0.95) – 0,525, mAP@0,50 – 0,810, 2,3 млн параметрів і 7,3 GFLOPs.

Візуалізація процесу навчання базової моделі та отриманої моделі 3 за критеріями mAP@(0,50:0,95) та втрат для обмежувального прямокутника (box_loss) наведені на рисунку 2.9.



а)



б)

Рис. 2.9. Візуалізація процесу навчання базової моделі та моделі 3:

а) критерій mAP50-95; б) критерій box_loss

Результати роботи моделі 3 на тестових зображеннях наведені на рисунку 2.10.

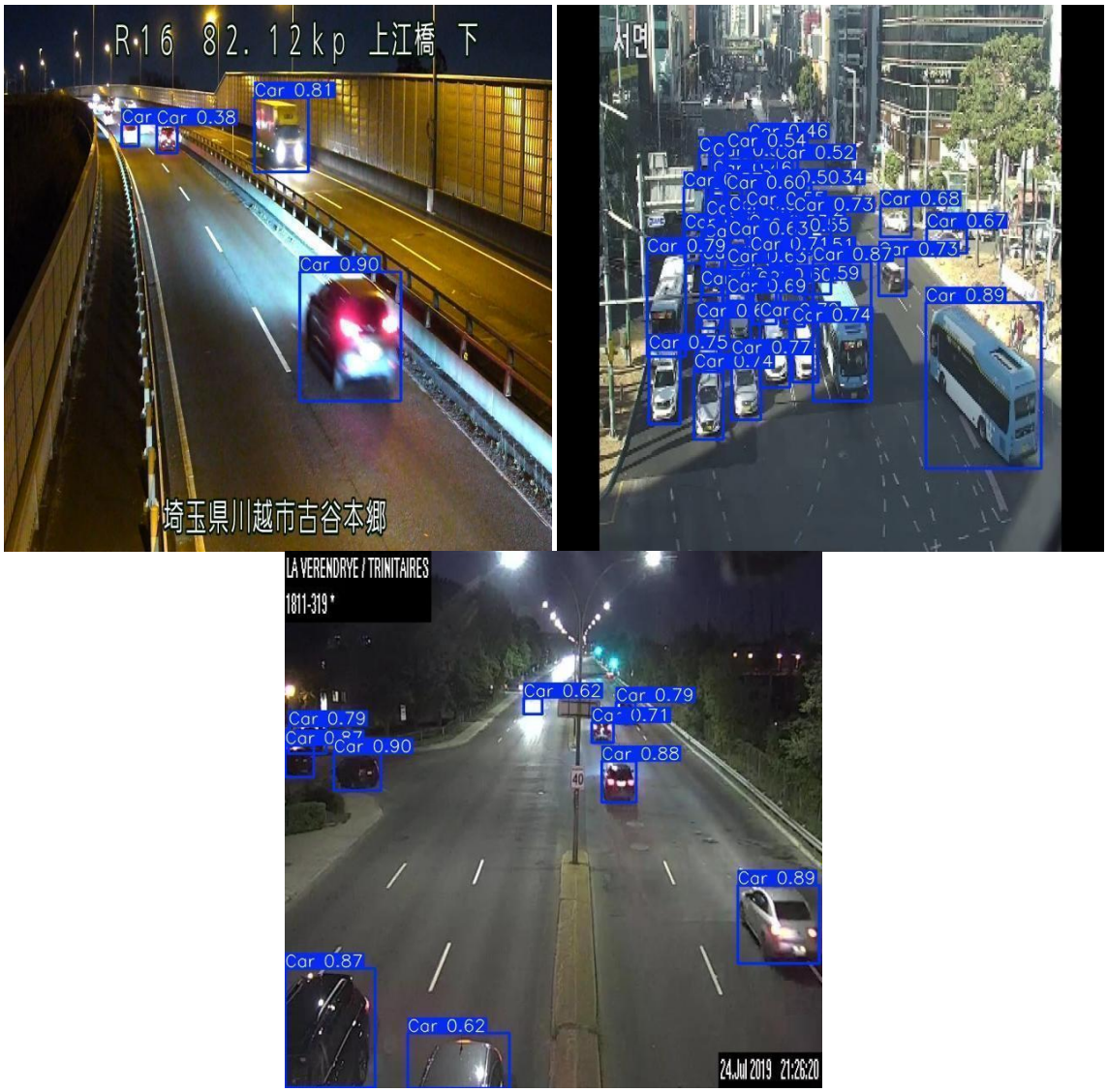


Рис. 2.10. Приклади детекцій об'єктів дорожнього руху моделлю 3

2.2 Розробка трекера об'єктів дорожнього руху з обробкою оклюзій

Задача БОТ на основі детекцій полягає в асоціації детекцій об'єктів між послідовними кадрами відео для побудови безперервних траєкторій об'єктів. Основна мета – точно відстежувати множину об'єктів у реальному часі, мінімізуючи помилки асоціації, пропуски та перемикання ідентичностей.

Нехай $D_t = d_1, d_2, \ldots, d_M$ – множина детекцій у кадрі t, де кожна детекція $d_i = (b_i, s_i)$ складається з:

- обмежувального прямокутника $\mathrm{b_i} = [\mathrm{x_1^i}, \mathrm{y_1^i}, \mathrm{x_2^i}, \mathrm{y_2^i}]$, де $(\mathrm{x_1^i}, \mathrm{y_1^i})$ та $(\mathrm{x_2^i}, \mathrm{y_2^i})$ – координати лівого верхнього та правого нижнього кутів.
- оцінки довіри $\mathrm{s_i} \in [0,1]$) отримана від детектора (наприклад, YOLOX як у методі ByteTrack).

Множина активних треків $\mathrm{T_t} = \tau_1, \tau_2, \ldots, \tau_\mathrm{M}$, де кожен трек τ_j характеризується станом фільтра Калмана $\mathrm{x_j} = [\mathrm{c_x}, \mathrm{c_y}, \mathrm{a}, \mathrm{h}, \mathrm{v_x}, \mathrm{v_y}, \mathrm{v_a}, \mathrm{v_h}]^\mathrm{T}$, де $(\mathrm{c_x}, \mathrm{c_y})$ – координати центру прямокутника; $a = w/h$ – співвідношення ширини до висоти (aspect ratio); h – висота прямокутника; $(\mathrm{v_x}, \mathrm{v_y}, \mathrm{v_a}, \mathrm{v_h})$ – швидкості відповідних параметрів.

Задача БОТ полягає у знаходженні відображення $\phi: D_t \to T_t \cup \{\varnothing\}$, де кожна детекція d_i асоціюється максимум з одним треком $\tau_j \in T_t$ або відхиляється $\phi(d_i) = \varnothing$. Кожен трек τ_j асоціюється максимум з однією детекцією або залишається без асоціації (оклюзія). При цьому потрібно мінімізувати функцію втрат:

$$\mathrm{L} = \sum_{i=1}^{M} \sum_{j=1}^{N} c_{ij}\, 1[\phi(d_i) = \tau_j] + \lambda_{\mathrm{FP}} \cdot \mathrm{FP} + \lambda_{\mathrm{FN}} \cdot \mathrm{FN} + \lambda_{\mathrm{IDs}} \cdot \mathrm{IDs}, \quad (2.9)$$

де c_{ij} – вартість асоціації між детекцією d_i та треком τ_j, визначається як $c_{ij} = 1 - \mathrm{IoU}(b_i, b_j^{\mathrm{track}})$; *FP* – кількість хибнопозитивних детекцій (асоційованих із неіснуючими об'єктами); *FN* – кількість пропущених об'єктів (треків без асоціації); *IDs* – кількість перемикань ідентичностей (коли трек асоціюється з іншим об'єктом); $\lambda_{\mathrm{FP}}, \lambda_{\mathrm{FN}}, \lambda_{\mathrm{IDs}}$ – вагові коефіцієнти, налаштовані емпірично.

Оновлення треків здійснюється у декілька етапів:

- *передбачення:* $x_j^{t+1} = F x_j^t + \omega_t$, де F – матриця переходу фільтра Калмана, $\omega_t \sim N(0, Q)$.
- на етапі *асоціації* використовується угорський алгоритм для мінімізації $\sum c_{ij}\, 1[\phi(d_i) = \tau_j]$.
- *оновлення*: якщо $\phi(d_i) = \tau_j$, стан x_j оновлюється за b_i інакше трек вважається втраченим.

- *ініціалізація та видалення*: нові треки створюються для $s_i \geq \theta_{high}$. Треки видаляються якщо не асоціюються протягом T_{max} кадрів.

Для підвищення точності ініціалізації треків у щільних сценах пропонується механізм **IoU-suppression**, адаптований із відомого в комп'ютерному зорі підходу Non-Maximum Suppression (NMS). У задачах виявлення об’єктів NMS використовується для видалення надлишкових обмежувальних прямокутників, що відповідають одному об’єкту, шляхом порівняння їхнього перекриття за метрикою IoU.

У контексті багатооб’єктного трекінгу IoU-suppression застосовується на етапі ініціалізації нових треків, щоб запобігти створенню дублікатів, коли кілька детекцій із високою оцінкою довіри $s_i \geq \theta_{high}$ відповідають одному об’єкту. Новий трек для детекції d_i створюється за умови:

$$\text{Init}(d_i) \Leftrightarrow \{s_i \geq \theta_{high}\} \wedge \{max_{j \in T_t} \text{IoU}(b_i, b_j^{\text{track}}) < \theta_{\text{init}}\} \qquad (2.10)$$

де T_t – множина активних треків, b_j^{track} – передбачений прямокутник треку τ_j, а θ_{init} – поріг IoU для придушення надлишкових треків.

Цей механізм особливо ефективний у щільних сценах, таких як відеоспостереження натовпу, де детектор може генерувати кілька детекцій для одного об’єкта через шум або часткове перекриття. IoU-suppression може зменшити кількість хибнопозитивних детекцій та перемикань ідентичностей.

Пропонується додати механізм **захисту втрачених треків**. Для цього для кожного треку окрім стану фільтра Калмана пропонується додати статус оклюзії $t_j^{\text{occ}} \in \{\text{True}, \text{False}\}$ та лічильник часу останньої асоціації t_j^{last}. Тоді трек буде видалятися якщо:

$$\text{Remove}(\tau_j) \Leftrightarrow (\text{t} - t_j^{\text{last}}. > T_{max}) \wedge (\text{t} - t_j^{\text{last}} > \tau_{\text{protect}}). \qquad (2.11)$$

Використання механізму захисту втрачених треків дозволить зберігати треки під час короткочасних оклюзій, зменшуючи кількість перемикань ідентичностей у сценах з частковими перекриттям об’єктів.

Пропонується вартість асоціації c_{ij} замість стандартного IOU визначати через **Distance IoU** (DIoU) [18]:

$$c_{ij} = 1 - \mathrm{DIoU}(b_i, \mathrm{b}_j^{\mathrm{track}}) \leq 1 - \theta_{\mathrm{assoc}}, \quad (2.12)$$

$$\mathrm{DIoU}(b_i, \mathrm{b}_j^{\mathrm{track}}) = \mathrm{IoU}(b_i, \mathrm{b}_j^{\mathrm{track}}) - \varrho^2(\mathrm{c}_i, \mathrm{c}_j^{track})/\mathrm{c}^2, \quad (2.13)$$

де $\varrho^2(\mathrm{c}_i, \mathrm{c}_j^{track})/\mathrm{c}^2$ – евклідова відстань між центрами прямокутників, c – діагональ найменшого прямокутника, що їх охоплює, θ_{assoc} – поріг асоціації.

DIoU дозволяє врахувати не лише перекриття, а й відстань між центрами, що може покращити асоціацію при малому чи нульовому перекритті. Це може зменшити кількість помилкових асоціацій і перемикань ідентичностей, особливо для об'єктів, що швидко рухаються.

Дослідження проводилися при таких параметрах $\theta_{init} = 0{,}65$, $\theta_{high} = 0{,}6$, $\tau_{protect} = 25$, $T_{max} = 30$, $\theta_{assoc} = 0{,}9$. Для YOLOX-Tiny використовувався вхідний розмір зображення 608×1088. Результати абляційних досліджень, які демонструють інкрементальний внесок кожного модуля, наведено в табл. 2.3.

Таблиця 2.3 – Абляційні дослідження методу на MOT-17-val для моделі YOLOX-Tiny

Метод	**Модулі**			**Критерії**					
	IoU-suppression	Захист втрачених треків	DIoU	MOTA (%)↑	IDF1 (%)↑	IDs ↓	FP ↓	FN ↓	FPS ↑
ByteTrack (базовий)	-	-	-	77,5	75,8	210	2240	9687	**29,6**
1	+	-	-	77,8	77,6	201	2059	**9677**	29,5
2	+	+	-	77,7	**78,3**	**189**	2087	9723	29,4
3	+	+	+	**78,0**	77,7	191	**1982**	9688	29,1

Згідно табл. 2.3 комбінація з DIoU демонструє найкращі результати: MOTA 78,0% (+0,5%), IDF1 77,7% (+1,9%), зменшення IDs на 19, FP на 258, при збільшенні FN на одиницю порівняно з базовим ByteTrack (MOTA 77,5%, IDF1 75,8%, IDs 210, FP 2240, FN 9687).

Використання IoU-suppression модуля підвищує MOTA (+0,3%) та IDF1 (+1,8%), знижуючи FP (-181) та IDs (-9), але не впливає на FN. Додавання модулю захисту втрачених треків забезпечує найбільше зниження IDs (-21) та зростання IDF1 (+2,5%), що вказує на ефективність обробки оклюзій, але MOTA знижується на 0,1%. Додаткове використання DIoU оптимізує всі метрики, зокрема FP (-258) та MOTA (+0,5%), зберігаючи високу швидкість (>29 FPS) при розмірі кадру 608×1088.

Усі модифікації відповідають вимогам реального часу, а варіант із DIoU є найперспективнішим завдяки балансу між точністю, стійкістю до оклюзій та обчислювальною ефективністю.

Для оцінки практичної значущості отриманих результатів обчислено розмір ефекту за Коеном [182, 183]: для метрики MOTA приріст +0,5 % відповідає d = 0,056 (малий ефект), для IDF1 приріст +1,9 % – d = 0.238 (малий–середній ефект). Водночас зменшення кількості хибнопозитивних спрацювань на 11,5 % (−258) та перемикань ідентифікаторів на 9,0 % (−19) є практично значущим з точки зору застосування в системах реального часу, оскільки безпосередньо знижує навантаження на модуль виявлення аномалій. Знаковий тест узгодженості підтверджує, що всі чотири ключові метрики (MOTA, IDF1, IDs, FP) покращились одночасно (p = 0,063), що свідчить про системний, а не випадковий характер покращень. Слід зазначити, що модуль захисту втрачених треків вносить основний внесок у зниження IDs (−12), тоді як IoU-suppression забезпечує найбільше зменшення FP (−181) та приріст IDF1 (+1,8 %).

Дослідження впливу типу сцени на результат трекінгу для найкращої комбінації модулів (IoU-supression + Захист втрачених треків + DIoU) наведено у табл. 2.4.

Таблиця 2.4 – Дослідження методу для різних типів сцени на MOT-17-val для моделі YOLOX-Tiny

Тип сцени	**Послідовності**	**MOTA, %** ByteTrack	**MOTA, %** метод з DIoU	**△MOTA**	**IDF1, %** ByteTrack	**IDF1, %** метод з DIoU	**△IDF1**
Висока щільність	MOT17-02, 09	64,3	64,7	+ 0,4	61,1	65,1	+ 4,0
Довгі оклюзії	MOT17-04	91,1	91,6	+ 0,5	87,0	88,4	+ 1,4
Рухома камера	MOT17-05, 10, 13	72,7	73,6	+ 0,9	72,9	74,2	+ 1,3
Низьке освітлення	MOT17-11	76,7	76,7	+ 0,0	72,9	77,3	+ 4,4

Запропонований метод БОТ, протестований на наборі MOT17-val із детектором YOLOX-Tiny, демонструє стабільне покращення метрик у різних сценаріях: MOTA зросла на 0,4% у сценах високої щільності (з 64,3% до 64,7%), на 0,5% при довгих оклюзіях (з 91,1% до 91,6%) і на 0,9% при рухомій камері (з 72,7% до 73,6%), залишившись незмінною в умовах низького освітлення (76,7%). Значне зростання IDF1 на 1,3–4,4% у всіх типах сцен, зокрема на 4,0% при високій щільності (з 61,1% до 65,1%) і 4,4% при низькому освітленні (з 72,9% до 77,3%), підтверджує ефективність DIoU, IoU-suppression та механізмів захисту треків у зменшенні перемикань ідентичностей. Незмінність MOTA в умовах низького освітлення вказує на обмеження YOLOX-Tiny, що потребує потужнішого детектора. Метод забезпечує надійну основу для трекінгу в реальному часі в складних умовах, таких як висока щільність об’єктів, довгі оклюзії та динамічні сцени, з потенціалом для подальшого вдосконалення шляхом інтеграції з потужнішими детекторами або ReID-модулями.

2.3 Розробка методу виявлення аномалій у відео з камер спостереження за дорожнім рухом

Розглянемо формальну модель виявлення аномалій, які можливо визначити за аналізом відео з камер трафіку в режимі реального часу.

Для виявлення аномалій перевищення швидкості та аварійного гальмування необхідно визначати швидкість транспортного засобу. Швидкість транспортного засобу з ідентифікатором tid у момент часу t пропонується обчислювати за формулою:

$$v(t, tid) = \|P_{world}(t, tid) - P_{world}(t - \Delta t, tid)\| \frac{3{,}6}{\Delta t_{sec}} , \qquad (2.14)$$

де P_{world} – позиція транспортного засобу в реальних світових координатах, отримана через гомографічне перетворення: P_{world}(t,tid)=H*P_{camera}(t,tid), де H – матриця гомографії, що перетворює координати з простору камери $P_{camera}(t, tid)$ у світові координати; ||Δ|| – евклідова відстань між двома точками $||\Delta|| = \sqrt{[(x_2 - x_1)^2 + (y_2 - y_1)^2]}$, Δt_{frames} – кількість кадрів $\Delta t_{sec} = \Delta t_{frames}$ / fps – часовий інтервал у секундах, між моментами t і t-Δt, а fps – частота кадрів відео; 3,6 – коефіцієнт для переведення швидкості з м/с у км/год.

Аномалія перевищення швидкості виявляється, якщо транспортний засіб із ідентифікатором tid перевищує максимальну дозволену швидкість (v_{max}) протягом трьох послідовних кадрів у межах області інтересу (ROI). Ця аномалія обчислюється за формулою:

$$A_{speed}(t, tid) = \begin{cases} 1, \text{якщо } C_{violations}(t, tid) \geq 3 \ \wedge \ ROI(t, tid), \\ 0, \text{інакше} \end{cases} \qquad (2.15)$$

де $C_{violations}(t, tid)$ – лічильник порушень, який обчислюється як: $C_{violations}(t, tid)$ = { $C_{violations}(t - 1, tid)$ + 1, якщо $v(t, tid) > v_{max}$, 0, інакше}; $ROI(t, tid)$ – індикатор перебування транспортного засобу в області інтересу: $ROI(t, tid)$ = { 1, якщо транспортний засіб у ROI, 0, інакше }; v_{max}– максимальна дозволена швидкість;

Аномалія аварійного гальмування виявляється, якщо швидкість транспортного засобу з ідентифікатором tid різко зменшується на величину, що перевищує поріг, за умови достатньої початкової швидкості. Ця аномалія обчислюється за формулою:

$$A_{brake}(t, tid) = \{1, \text{якщо} \Delta v > \Delta v_{threshold} \wedge v(t-1, tid) > v_{min}, 0, \text{інакше}\}, (2.16)$$

де $\Delta v = v(t-1, tid) - v(t, tid)$ – зменшення швидкості між сусідніми кадрами; $\Delta v_{threshold} = 15$ км/год – поріг різкого гальмування; $v_{min} = 10$ км/год – мінімальна початкова швидкість для фіксації аномалії.

Виявлення аномалії перетину стоп-лінії на червоний сигнал світлофора ґрунтується на одночасному виконанні трьох умов, реалізованих через відповідні функції: перетину стоп-лінії, перевірки стану світлофора та фільтрації одиночних спрацьовувань. Загальну умову виявлення аномалії задамо таким формульним виразом:

$$A_{red}(t, tid) = I(t, tid) \wedge R(t) \wedge V(t, tid), \quad (2.17)$$

де $I(t, tid)$ – функція перетину стоп-лінії транспортним засобом із ідентифікатором треку tid у момент часу t; $R(t)$ – функція перевірки стану світлофора (червоний сигнал); $V(t, tid)$ – функція фільтрації одиночних спрацьовувань на основі швидкості руху.

Функція перетину стоп-лінії $I(t, tid)$ обчислюється як:

$$I(t, tid) = \{ 1, [P(t-\Delta t), P(t)] \cap [L_{start}, L_{end}] \neq \emptyset, 0, \text{інакше} \}, (2.18)$$

де $[L_{start}, L_{end}]$ – координати початкової та кінцевої точок стоп-лінії; $P(t)$ – позиція транспортного засобу в момент часу t, визначена як центр нижньої межі обмежувального прямокутника об'єкта; $[P(t-\Delta t), P(t)]$ – траєкторія руху транспортного засобу за інтервал часу Δt, ідентифікована за tid.

Функція перевірки стану світлофора у момент часу t обчислюється як:

$$R_t = \{1, \text{якщо} \frac{(N_{red}^{(1)} + N_{red}^{(2)})}{N_{pixels}} > 0{,}02 ; 0, \text{інакше}, \quad (2.19)$$

де $N_{red}^{(1)}$ – кількість пікселів, що задовольняють умову першого діапазону відтінку червоного в просторі HSV:$H \in [0°, 10°] \wedge S \in [100,255] \wedge V \in [100,255]$; $N_{red}^{(2)}$ – кількість пікселів другого діапазону: $H \in [160°, 180°] \wedge S \in [100,255] \wedge V \in [100,255]$; N_{pixels} – загальна кількість пікселів у зоні аналізу сигналу.

На відміну від RGB-підходу, де порівнюються абсолютні значення каналів (R > 100, R > 1,5G, R > 1,5B), аналіз у просторі HSV (Hue–Saturation–Value) спирається на хроматичні характеристики кольору. Перетворення вхідного кадру з BGR у HSV виконується функцією cv2.cvtColor(roi, cv2.COLOR_BGR2HSV), після чого для ROI-ділянки формуються дві бінарні маски відповідно до двох діапазонів відтінку.

Необхідність двох діапазонів зумовлена цикличністю шкали відтінку: червоний колір у просторі HSV охоплює кути $H \in [0°, 10°]$(низькі значення) та $H \in [160°, 180°]$(високі значення), тоді як проміжні кути відповідають жовтому, зеленому, блакитному та синьому. Умови S≥100 та V≥100 відфільтровують ненасичені (білі, сірі) та темні пікселі, що дозволяє виключити хибні спрацювання від відблисків і тіней, характерних для вуличних умов зйомки. Бінарні маски об'єднуються порозрядною диз'юнкцією:

$$M = M_1 \vee M_2, M_i(p) = 1\left[H_p \in [h_i^-, h_i^+] \wedge S_p \geq 100 \wedge V_p \geq 100\right], \quad (2.20)$$

де p – піксель зони аналізу, $1[\cdot]$ – індикаторна функція.

Порогове значення 0,02 є нижчим порівняно з 0,05 у RGB-підході, оскільки HSV-маскування є більш точним: насичені червоні пікселі власне сигналу класифікуються коректно, тоді як фонові пікселі відтінку червоного при недостатній насиченості або яскравості відфільтровуються умовами $S \geq 100, V \geq 100$ ще на етапі формування масок.

Функція фільтрації $V(t, tid)$ зменшує кількість хибнопозитивних спрацьовувань, у випадку оклюзій з іншими транспортними засобами або/та ефекту мерехтіння обмежувальної рамки у випадку зупинки перед стоп-лінією:

$$V(t, tid) = \left\{ 1, \text{якщо } v(t, tid) > v_{\underset{crossing}{min}}, 0, \text{інакше} \right\} \quad (2.21)$$

де $V(t, tid)$ – швидкість транспортного засобу з ідентифікатором tid у момент часу t; $v_{\underset{crossing}{min}}$ – мінімальна порогова швидкість, яка вказує на рух транспортного засобу через стоп-лінію.

Аномалія довгого перебування в ROI визначається за формулою:

$$A_{roi}(t, tid) = \{1, \text{якщо } t_{roi}(t, tid) > t_{max} \wedge ROI(t, tid) = 1, 0, \text{інакше }\}, \quad (2.22)$$

де $t_{roi}(t, tid)$ – час перебування транспортного засобу з ідентифікатором tid у межах ROI у момент часу t; t_{max} – максимальний допустимий час перебування в ROI; $ROI(t, tid)$ – функція, що вказує, чи перебуває транспортний засіб у межах ROI в момент часу t.

Тривалість перебування транспортного засобу в ROI обчислюється як:

$$t_{roi}(t, tid) = (t_{current} - t_{entry}(tid))/fps \quad (2.23)$$

де $t_{current}$ – поточний момент часу; ; fps– частота кадрів відеопотоку; $t_{entry}(tid) = min\{t' : ROI(t', tid) = 1 \wedge ROI(t' - 1, tid) = 0\}$ – момент часу, коли транспортний засіб із ідентифікатором tid увійшов у ROI, тобто коли $ROI(t', tid)$ змінилося з 0 на 1.

Функція належності до ROI визначається за допомогою перевірки, чи перебуває транспортний засіб у межах заданої області інтересу:

$$ROI(t, tid) = point_in_polygon(P_{vehicle}(t, tid), ROI_{vertices}), \quad (2.24)$$

де $P_{vehicle}(t, tid)$ – позиція транспортного засобу з ідентифікатором tid у момент часу t, визначена як центр нижньої межі обмежувального прямокутника; $ROI_{vertices}$ – набір вершин, що визначають багатокутник області інтересу.

Функція point_in_polygon перевіряє, чи точка $P_{vehicle}$(t, tid) лежить усередині багатокутника, заданого вершинами $ROI_{vertices}$, використовуючи, наприклад, функцію contains_point бібліотеки matplotlib.

На основі розглянутої формальної моделі пропонується блок-схема алгоритму методу виявлення аномалій у відео з камер трафіку в режимі реального часу, що зображена на рисунку 2.11.

Початковий етап алгоритму передбачає налаштування камери через графічний інтерфейс користувача. На цьому етапі визначаються області та лінії інтересу, такі як: лінії зупинки транспортних засобів перед світлофором; зони перехрестя; зони розташування світлофорів. Ці елементи можуть бути задані безпосередньо через інтерфейс камери або програмне забезпечення, що забезпечує гнучкість у виборі областей для моніторингу. Етап калібрування камери є важливим для точного вимірювання швидкості транспортних засобів, оскільки оцінка швидкості чутлива до якості калібрування. Цей процес включає визначення внутрішніх і зовнішніх параметрів камери, таких як: фокусна відстань та 3D-положення камери (переміщення та обертання відносно сцени).

Наступний етап включає налаштування системи детекції, трекінгу та виявлення аномалій. Цей процес охоплює: вибір моделі детекції об’єктів (наприклад, нейронної мережі для виявлення транспортних засобів); налаштування параметрів моделі детекції (чутливість, порогові значення); вибір і налаштування алгоритму трекінгу для відстеження об’єктів за унікальними ідентифікаторами треків (Track ID, або tid); визначення параметрів для виявлення аномалій, наприклад, встановлення максимального дозволеного часу перебування в зоні перехрестя для задачі обмеження тривалості перебування.

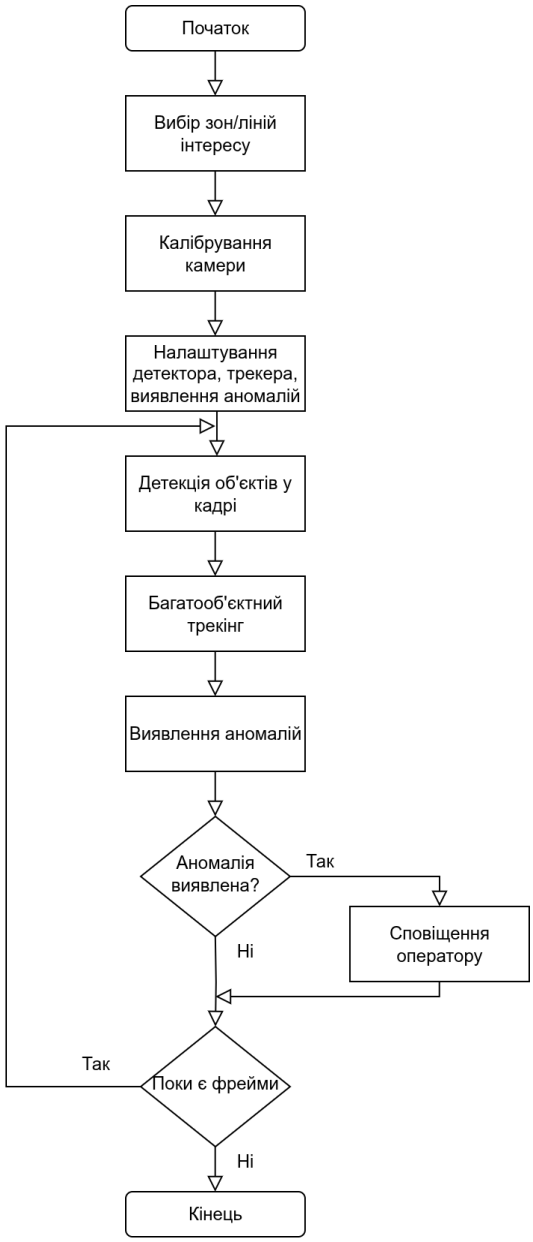


Рис. 2.11. Блок-схема алгоритму виявлення аномалій у відео з камер відеоспостереження трафіку

Після етапів налаштувань йдуть етапи обробки відеопотоку, а саме, детекції, відстеження та аналізу аномалій. На етапі детекції на кожному кадрі відео виконується виявлення об’єктів дорожнього руху (транспортних засобів, пішоходів тощо) з використанням обраної моделі детекції. На етапі багатооб’єктного трекінгу на основі детекцій формується послідовність треків, де кожному об’єкту присвоюється унікальний tid для відстеження його руху в

часі. На етапі виявлення аномалій на основі отриманих детекцій і треків виконується аналіз для виявлення аномальних подій. У разі виявлення аномалії система генерує сповіщення для оператора, що дозволяє оперативно реагувати на порушення.

Калібрація камери проводилася за допомогою гомографічного перетворення подібним чином як у роботі [169]. Процедура калібрування включала такі етапи: за допомогою користувацького інтерфейсу задавалися чотири або більше точок на 2D-зображенні з камери; для кожної з цих точок визначалися відповідні їм реальні координати у світовій системі координат, отримані за допомогою інструменту вимірювання відстаней Google Maps; на основі отриманих пар точок обчислювалась матриця гомографічного перетворення.

Перетворення реалізовано з використанням функції cv2.findHomography бібліотеки OpenCV та методу RANSAC для підвищення стійкості до викидів. Приклад вимірювань відстаней наведено на рис. 2.12. Приклад задання точок на 2D-зображенні камери наведено на рис. 2.13.

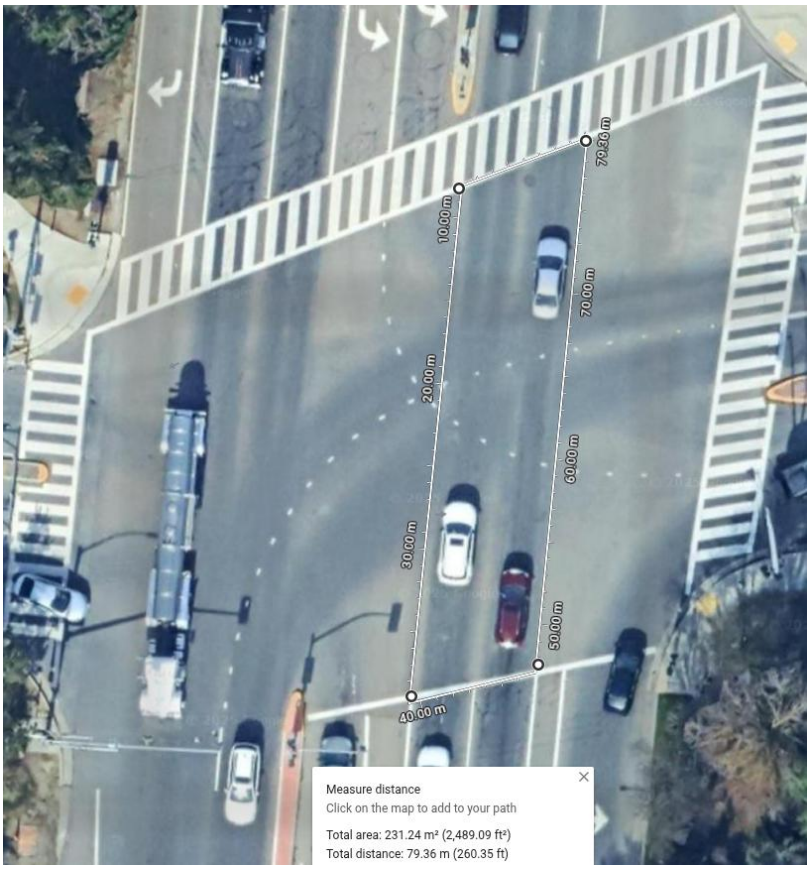


Рис. 2.12. Вимір відстаней на перехресті Friant Roulette

а)

```
Точка 0 координати зображення (379,541) -> світові x,y (через кому): 100,100
Точка 1 координати зображення (717,455) -> світові x,y (через кому): 100,130
Точка 2 координати зображення (1035,462) -> світові x,y (через кому): 108,130
Точка 3 координати зображення (898,537) -> світові x,y (через кому): 108,100
```

б)

Рис. 2.13. Задання гомографічних перетворень:

а - задання точок у кадрі камери, *б* - задання відносних відстаней

Виявлення аномалій у дослідженні сформульовано як задачу бінарної класифікації, де подія класифікується як “аномалія“ або “нормальна“. Оскільки аномалії є рідкісними подіями, для оцінки якості класифікації використано коефіцієнт кореляції Меттьюза (MCC), який враховує всі елементи матриці помилок і є стійким до нерівномірного розподілу класів у вибірці. Значення MCC перебуває в діапазоні [-1; +1], де +1 відповідає ідеальній класифікації, і обчислюється як:

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}, \quad (2.25)$$

де TP – кількість правильно виявлених аномалій, FP – кількість помилкових спрацювань, FN – кількість пропущених аномалій, TN – кількість правильно виявлених нормальних подій.

Для реалізації детектора в реальному часі на камері обрано модель YOLOv8-nano через її компактність і високу швидкість обробки. Для відстеження траєкторій об'єктів використано трекер OC-SORT з такими параметрами: функція асоціації: IoU на основі GIoU метрики; часовий інтервал між кадрами: delta_t = 1; поріг впевненості для обмежувальних рамок: conf_thresh = 0,5; поріг IoU для асоціації треків: 0,5; коефіцієнт інерції траєкторії: 0,39; максимальна кількість кадрів без асоціацій: max_age = 50; мінімальна кількість успішних детекцій для створення треку: min_hits = 1; використання byte-асоціації для покращення стабільності: True. Для порівняння також досліджено класичний трекер SORT з параметрами: max_age = 50, min_hits = 1.

Для оцінки впливу трекера OC-SORT порівняно з SORT на обраних типах аномалій на підготовленому датасеті було проаналізовано комбінації YOLOv8n + SORT та YOLOv8n + OC-SORT. Результати подано в табл. 2.5.

Таблиця 2.5 – Результати дослідження виявлення аномальних подій у відеопотоці з камер трафіку

Тип аномалії/ метод	Критерій	Проїзд на червоний	Перебування в зоні перехрестя	Аварійне гальмування	Перевищення швидкості
YOLOv8n + SORT	MCC	0,819	0,833	0,459	0,740
YOLOv8n + OC-SORT	MCC	**0,909**	**0,881**	**0,505**	**0,815**

Приклад виявлення аномалій на перехресті Friant Roulette наведено на рис. 2.14-2.17.

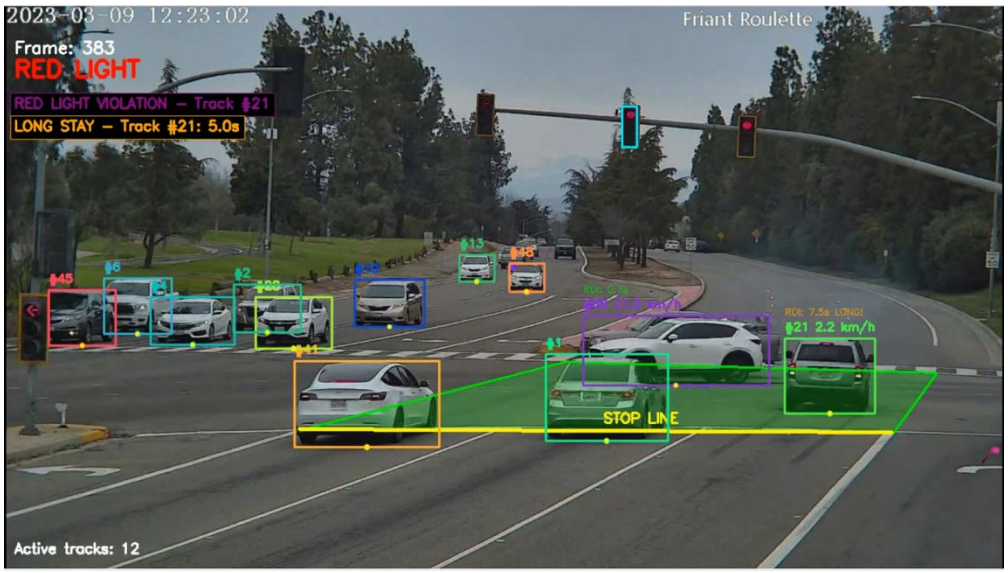


Рис. 2.14. Проїзд на червоний і перебування в зоні більше 5с

Рис. 2.15. Проїзд на червоний, перевищення швидкості 100 км/г та аварійне гальмування

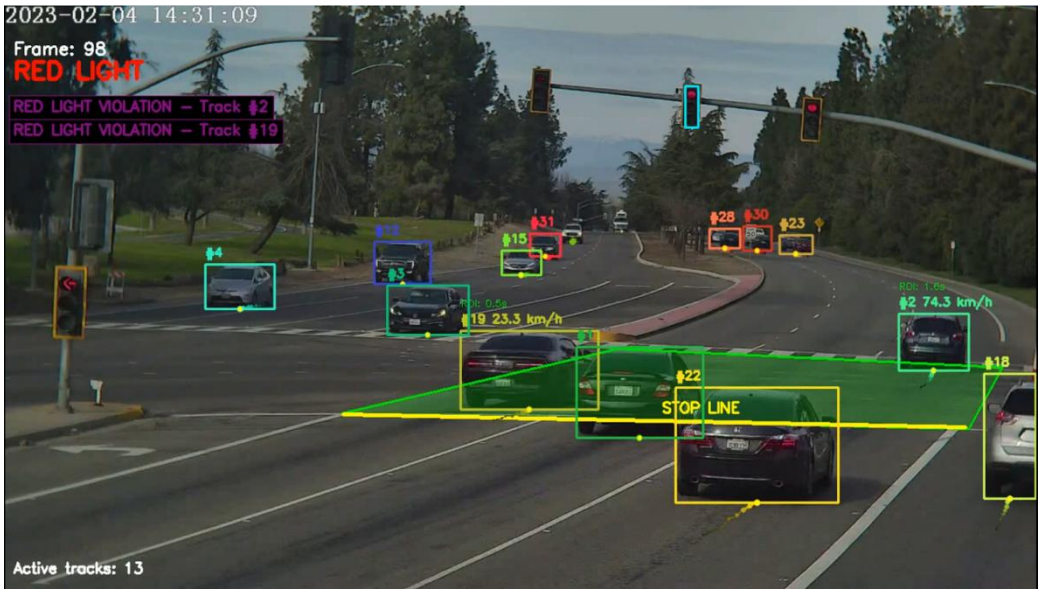


Рис. 2.16. Проїзд на червоний

Рис. 2.17. Проїзд на червоний та перевищення швидкості 100 км/г

Результати, наведені в таблиці 2.6, підтверджують вищу точність комбінації YOLOv8n + OC-SORT порівняно з YOLOv8n + SORT завдяки кращій стійкості OC-SORT до короткочасних оклюзій і меншій кількості помилкових асоціацій треків (перемикань tid). Найвищі значення коефіцієнта MCC отримано для аномалій «проїзд на червоне світло» (0,909) і «перебування в зоні перехрестя» (0,881). Нижчий результат зафіксовано для «перевищення швидкості» (0,815), а найнижчий – для «аварійного гальмування» (0,505). Висока точність виявлення проїзду на червоне світло пояснюється чіткими умовами класифікації (виявлення сигналу світлофора, перетину стоп-лінії та фільтрація хибних спрацювань). Нижчі результати для оцінки швидкості зумовлені обмеженим кутом огляду камери, її нестабільністю під впливом вітру, змінами розмірів обмежувальних рамок в умовах поганої видимості чи оклюзій.

Перспективи вдосконалення включають використання складніших моделей детектора (наприклад, YOLOv8s), донавчання YOLOv8n на специфічних даних камер, інтеграцію модулів реідентифікації для зменшення помилок трекінгу та покращення алгоритмів оцінки швидкості й прискорення. Перевагою підходу «детектор–трекер–аналізатор аномалій» є його гнучкість і

можливість розширення для нових сценаріїв, як-от виявлення пішоходів на проїжджій частині чи забороненого паркування.

Основні обмеження підходу пов’язані із залежністю від кута встановлення камери, її стабільності та якості відеопотоку (мін. 720p). Сильні оклюзії, нестабільне освітлення чи невдалий ракурс суттєво знижують точність, особливо для виявлення аварійного гальмування.

2.4 Висновки до розділу

У другому розділі дисертаційної роботи розроблено набір методів інтелектуального аналізу відеопотоку дорожнього руху, що включає удосконалену модель детекції об'єктів, метод багатооб'єктного трекінгу з обробкою оклюзій та метод виявлення аномалій на основі аналізу траєкторій. На основі проведених досліджень можна сформулювати наступні висновки:

1. Розроблено модель детекції об'єктів дорожнього руху на основі модифікованої архітектури YOLO із застосуванням модулів C3Ghost-CBAM та полегшених згорток GhostConv. Запропонована модифікація дозволила зменшити кількість параметрів моделі на 23,3% (з 3,0 млн до 2,3 млн) та обчислювальну складність на 9,9% (з 8,1 до 7,3 GFLOPs) при незначному зниженні точності (mAP@0,50:0,95 знизився з 0,531 до 0,525, що становить лише 1,1%). Інтеграція модуля уваги CBAM дозволила компенсувати втрати від застосування Ghost-згорток, підвищивши mAP@0,50 з 0,805 до 0,810. Це підтверджує ефективність запропонованого підходу для досягнення балансу між точністю та обчислювальною ефективністю в умовах обмежених ресурсів.

2. Удосконалено метод багатооб'єктного трекінгу ByteTrack шляхом впровадження трьох механізмів: IoU-suppression для запобігання дублюванню треків при ініціалізації, механізму захисту втрачених треків для обробки короткочасних оклюзій та метрики DIoU для покращення асоціації детекцій з

треками. Експериментальні дослідження на датасеті MOT17-val з детектором YOLOX-Tiny показали підвищення метрики MOTA на 0,5% (з 77,5% до 78,0%), IDF1 на 1,9% (з 75,8% до 77,7%), зменшення кількості перемикань ідентичностей (ID switches) на 9,0% (з 210 до 191) та хибнопозитивних детекцій на 11,5% (з 2240 до 1982) при збереженні швидкості обробки понад 29 FPS.

3. Досліджено ефективність запропонованого методу трекінгу для різних типів сцен. Найбільше покращення метрики IDF1 досягнуто для сцен з низьким освітленням (+4,4%) та високою щільністю об'єктів (+4,0%), що свідчить про ефективність механізмів обробки оклюзій. Метрика MOTA зросла на 0,9% для сцен з рухомою камерою та на 0,5% для сцен з довгими оклюзіями, що підтверджує універсальність запропонованих удосконалень.

4. Розроблено метод виявлення аномалій у відеопотоці дорожнього руху на основі аналізу траєкторій об'єктів, який дозволяє виявляти чотири типи порушень: проїзд на червоний сигнал світлофора, перевищення швидкості, аварійне гальмування та тривале перебування в зоні перехрестя. Метод базується на формальній моделі, що включає: обчислення швидкості транспортного засобу через гомографічне перетворення координат, функцію перетину стоп-лінії, RGB-аналіз стану світлофора та функцію належності до області інтересу.

5. Для оцінки якості виявлення аномалій застосовано метрику MCC (коефіцієнт кореляції Меттьюза), яка є стійкою до нерівномірного розподілу класів. Експериментальні дослідження показали, що комбінація YOLOv8n + OC-SORT забезпечує вищу точність порівняно з YOLOv8n + SORT для всіх типів аномалій: MCC для проїзду на червоний сигнал становить 0,909 (+0,090), для перебування в зоні перехрестя – 0,881 (+0,048), для перевищення швидкості – 0,815 (+0,075), для аварійного гальмування – 0,505 (+0,046).

6. Виявлено, що найвища точність досягається для аномалій з чіткими умовами класифікації (проїзд на червоний сигнал), тоді як нижчі результати для оцінки швидкості та аварійного гальмування зумовлені чутливістю до якості

трекінгу, кута огляду камери та стабільності відеопотоку. Це визначає напрямки подальшого вдосконалення методу.

Отже, у розділі розроблено три взаємопов'язані методи, що утворюють комплексне рішення для інтелектуального аналізу відеопотоку дорожнього руху. Запропоновані методи забезпечують можливість роботи в реальному часі на пристроях з обмеженими обчислювальними ресурсами при збереженні високої точності виявлення та відстеження об'єктів, а також виявлення аномальних подій.

Основні результати розділу опубліковані автором у роботах [1-3, 6].

РОЗДІЛ 3. РОЗРОБКА ІНТЕЛЕКТУАЛЬНОЇ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ АНАЛІЗУ ВІДЕОПОТОКУ ДОРОЖНЬОГО РУХУ

Відповідно до поставлених задач на основі запропонованих у розділі 2 методів розробимо інтелектуальну технологію аналізу відеопотоку дорожнього руху.

3.1 Проектування загальної структури інформаційної технології

У загальному вигляді інформаційна технологія аналізу відеопотоку дорожнього руху реалізується як сукупність взаємопов’язаних модулів, згрупованих у функціональні блоки (рис. 3.1), кожен із яких виконує чітко окреслені завдання та оперує визначеними вхідними й вихідними даними. Розрізнення блоків забезпечує масштабованість і гнучкість системи: обчислювальна частина на Edge-рівні може незалежно еволюціонувати від інтерфейсу користувача та хмарних шарів зберігання й аналітики. Нижче наведено докладний опис кожного блоку та його модулів.

1. Блок «Інтерфейс користувача»

Модуль налаштування ROI та калібрації камери:

– Вхідні дані: координати зон інтересу (ROI), параметри калібрації (фокусна відстань, висота монтажу), URL-адреси камер.

– Функції: інтерактивне малювання ROI на відеопотоці, автоматичне або ручне введення калібраційних параметрів, валідація геометричної коректності.

– Вихідні дані: конфігураційний файл ROI та калібрації, переданий до Edge-рівня.

Модуль візуалізації треків, теплових карт, статистики:

– Вхідні дані: агреговані треки, теплові карти, статистичні звіти з Cloud-рівня.

– Функції: побудова адаптивного веб-інтерфейсу, відображення треків на карті/відео, генерація теплових карт щільності руху, інтерактивних графіків (інтенсивність, швидкість, класифікація).

– Вихідні дані: візуальні елементи для користувача, експортовані звіти (PDF/CSV).

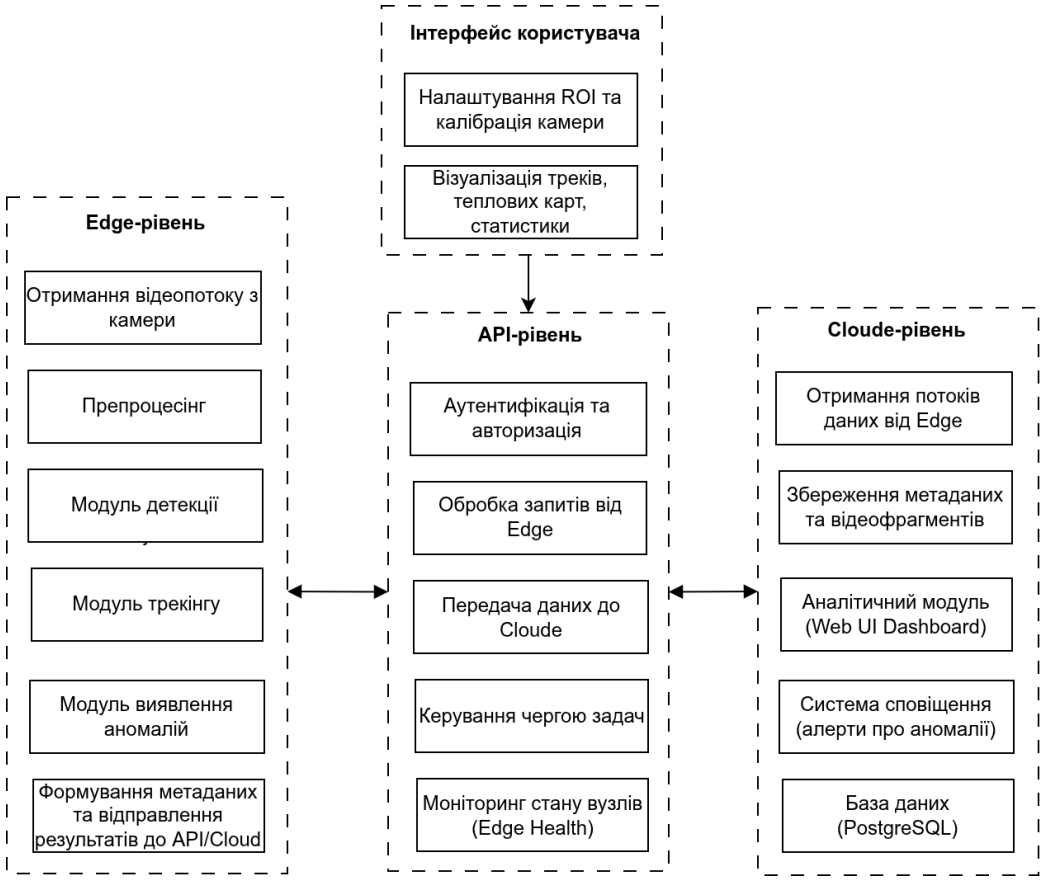


Рис. 3.1. Загальна структурна схема розробленої інформаційної технології аналізу відеопотоку дорожнього руху

2. Блок «API-рівень»

Модуль аутентифікації та авторизації:

– Вхідні дані: облікові дані (логін/пароль, OAuth-токен).

– Функції: перевірка токенів, видача JWT з ролями (оператор, аналітик, адміністратор), логування сесій.

– Вихідні дані: валідний токен доступу, статус авторизації.

Модуль обробки запитів від Edge:

– Вхідні дані: метадані треків, детекцій, аномалій з Edge-рівня (через gRPC/REST).

– Функції: валідація, фільтрація, нормалізація даних, маршрутизація до черги завдань або бази.

– Вихідні дані: підтвердження прийому, перенаправлені пакети до Cloud.

Модуль передачі даних до Cloud:

– Вхідні дані: агреговані треки, метадані, статистика з Edge.

– Функції: пакетна передача через захищений канал, стиснення, шифрування, повторні спроби при втраті зв’язку.

– Вихідні дані: підтвердження доставки, статус синхронізації.

Модуль керування чергою завдань:

– Вхідні дані: завдання на аналітику (агрегація, виявлення аномалій).

– Функції: публікація в брокер (Kafka/RabbitMQ), пріоритизація, масштабування воркерів.

– Вихідні дані: статус виконання, сповіщення про завершення.

Модуль моніторингу стану вузлів (Edge Health):

– Вхідні дані: телеметрія з Edge (CPU, GPU, мережа, затримки).

– Функції: агрегація метрик, виявлення аномалій, сповіщення.

– Вихідні дані: звіти про здоров’я вузлів, алерти.

3. Блок «Edge-рівень»

Модуль отримання відеопотоку з камери:

– Вхідні дані: RTSP/RTMP-потоки, локальні файли.

– Функції: буферизація, декодування, синхронізація часових міток.

– Вихідні дані: сирі кадри з метаданими (timestamp, camera_id).

Модуль препроцесінгу:

– Вхідні дані: сирі кадри.

– Функції: корекція дисторсії, стабілізація, адаптивна фільтрація шуму, нормалізація освітленості.

– Вихідні дані: чисті кадри, готові до детекції.

Модуль детекції:

– Вхідні дані: препроцесовані кадри, ROI.

– Функції: детекція транспортних засобів/пішоходів (модифікований YOLOv8), фільтрація за ROI.

– Вихідні дані: bounding boxes, класи, впевненість.

Модуль трекінгу:

– Вхідні дані: детекції послідовних кадрів.

– Функції: асоціація об’єктів (DeepSORT/ByteTrack), присвоєння track_id, фільтрація шуму.

– Вихідні дані: треки з координатами, швидкістю, напрямком.

Модуль виявлення аномалій:

– Вхідні дані: треки, параметри калібрації.

– Функції: аналіз траєкторій (зупинки, різкі повороти), виявлення скупчень, обчислення швидкості/прискорення.

– Вихідні дані: події аномалій з типом, рівнем небезпеки, timestamp.

Модуль формування метаданих та відправлення результатів до API/Cloud:

– Вхідні дані: треки, аномалії, статистика.

– Функції: агрегація за інтервалом, стиснення, пакетування в JSON/Protobuf.

– Вихідні дані: метадані, готові до передачі.

4. Блок «Cloud-рівень»

Модуль отримання потоків даних від Edge:

– Вхідні дані: пакети метаданих через API.

– Функції: розпакування, валідація, тимчасове буферизування.

– Вихідні дані: нормалізовані записи для зберігання.

Модуль збереження метаданих та відеофрагментів:

– Вхідні дані: метадані, ключові кадри/фрагменти (опціонально).

– Функції: індексація за часом/камерою, стиснення відео, довготривале архівування.

– Вихідні дані: доступні записи в object storage (S3-подібне).

Аналітичний модуль (Web UI Dashboard):

– Вхідні дані: збережені метадані, запити користувача.

– Функції: агрегація статистики (інтенсивність, середня швидкість, класи), генерація теплових карт, виявлення довготривалих аномалій.

– Вихідні дані: звіти, дашборди, API-відповіді.

Система сповіщення (алерти про аномалії):

– Вхідні дані: події аномалій.

– Функції: фільтрація за порогами, відправка через email/SMS/Slack, ескалація.

– Вихідні дані: сповіщення, лог подій.

База даних (PostgreSQL):

– Вхідні дані: метадані, треки, конфігурації.

– Функції: CRUD-операції, індексація (PostGIS для геопросторових запитів), реплікація, бекапи.

– Вихідні дані: структуровані дані для аналітики та UI.

Кожен із блоків взаємодіє через стандартизовані інтерфейси (REST/gRPC, WebSocket, SQL, Protobuf), що забезпечує високу модульність, відмовостійкість та можливість горизонтального масштабування інформаційної технології аналізу відеопотоку дорожнього руху.

3.2 Розроблення UML-діаграм інформаційної технології

Функціонування інформаційної технології інтелектуального аналізу відеопотоку дорожнього руху складається з двох послідовних етапів: етапу ініціалізації та налаштування (виконується одноразово) та етапу безперервної

обробки відеопотоку в режимі реального часу (виконується циклічно). Розділення на два етапи забезпечує гнучкість системи та можливість адаптації до різних сцен спостереження без зміни програмного коду.

3.2.1 UML-діаграма послідовності інформаційної технології, етап ініціалізації та налаштування системи

Етап ініціалізації виконується адміністратором системи один раз при встановленні камери відеоспостереження або при зміні умов сцени (зміна положення камери, реконфігурація перехрестя, зміна правил дорожнього руху тощо). Мета цього етапу – визначити геометричні параметри сцени, провести калібрування камери та встановити пороги для виявлення аномальних подій. Результатом етапу є конфігураційний файл, що містить всю необхідну інформацію для автоматичної обробки відеопотоку в режимі реального часу. UML-діаграма послідовності інформаційної технології етапу ініціалізації та налаштування системи зображена на рисунку 3.2.

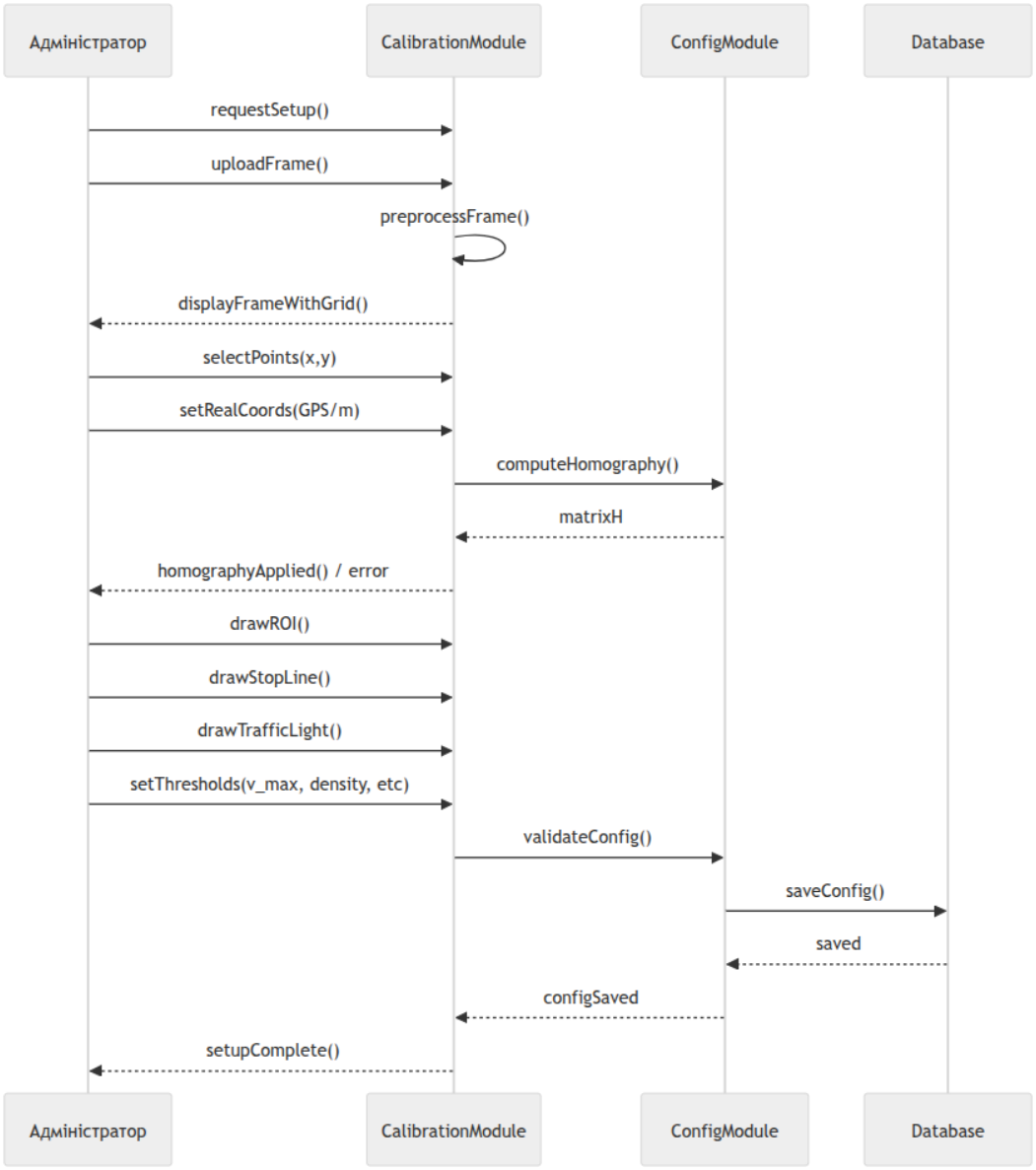


Рис. 3.2. UML-діаграма послідовності інформаційної технології на етапі ініціалізації та налаштування системи

Процес починається з ініціювання адміністратором запиту на налаштування системи через веб-інтерфейс або спеціалізований інструмент конфігурації. Модуль калібрування (CalibrationModule) отримує команду requestSetup() та переходить у режим очікування вхідних даних від адміністратора.

Адміністратор завантажує репрезентативний кадр з камери відеоспостереження, який відображає типову сцену в нормальних умовах

освітлення. Рекомендується використовувати кадр, знятий у денний час за ясної погоди, оскільки він забезпечує найкращу видимість дорожньої розмітки та інших орієнтирів. Завантажений кадр передається до модуля калібрування методом uploadFrame(image).

Модуль калібрування відображає завантажений кадр адміністратору за допомогою методу displayFrame(image), надаючи інтерактивний інтерфейс для подальшої розмітки та вибору опорних точок.

Калібрування камери є важливим етапом так як забезпечує точне перетворення координат з піксельного простору зображення у світовий простір (метричні координати на дорозі). Цей процес базується на методі гомографічного перетворення, яке встановлює математичне відображення між площиною зображення та площиною дорожнього полотна.

Адміністратор вибирає мінімум чотири опорні точки на зображенні, координати яких можна точно визначити як на кадрі, так і в реальному світі. Рекомендується вибирати точки, що утворюють чотирикутник максимально можливої площі в межах зони інтересу (наприклад, кути перехрестя, розмітка доріг, краї тротуарів). Для підвищення точності калібрування рекомендовано використовувати 6-8 опорних точок замість мінімальних чотирьох.

Для кожної обраної опорні точки адміністратор визначає її реальні координати у світовій системі координат. Найпоширенішими методами визначення є:

1. Використання інструментів вимірювання відстаней у Google Maps або інших картографічних сервісах, що дозволяє отримати координати у метричній системі або географічні координати (широта, довгота).

2. Використання GPS-приймача для безпосереднього вимірювання координат на місцевості.

3. Використання існуючої технічної документації перехрестя з точними розмірами.

На основі отриманих пар відповідностей між точками зображення та світовими координатами модуль калібрування обчислює матрицю гомографічного перетворення H розміром 3×3. Для обчислення використовується метод найменших квадратів у поєднанні з алгоритмом RANSAC (RANdom SAmple Consensus) для усунення впливу викидів та підвищення стійкості до помилок вимірювання.

Ця матриця дозволяє перетворювати будь-яку точку з піксельних координат (x_pixel, y_pixel) у світові координати (x_world, y_world).

Обчислена матриця H зберігається як основний результат етапу калібрування та буде використовуватися на етапі runtime обробки для перетворення координат обмежувальних рамок детекцій у світові координати, що необхідно для точного обчислення відстаней, швидкості та прискорення транспортних засобів.

Після завершення калібрування камери адміністратор переходить до визначення ключових зон та елементів сцени, що використовуються алгоритмами виявлення аномальних подій.

Стоп-лінія – це відрізок прямої на дорожньому полотні, що позначає межу дозволеної зупинки транспортного засобу перед перехрестям при червоному сигналі світлофора. Адміністратор визначає положення стоп-лінії методом drawStopLine(line), вказуючи дві точки на зображенні.

Стоп-лінія використовується алгоритмом виявлення проїзду на червоне світло, детально описаним у підрозділі 2.3, формули (2.17-2.21). Система перевіряє, чи перетинає траєкторія руху транспортного засобу цю лінію в момент, коли світлофор показує червоний сигнал.

ROI (Region Of Interest) перехрестя – це полігональна область, що охоплює безпосередньо зону перехрестя, де транспортні засоби повинні швидко рухатись і не затримуватись. Адміністратор визначає цю зону методом drawROI(polygon), послідовно вказуючи вершини багатокутника, що обмежує область перехрестя.

ROI використовується алгоритмом виявлення надмірно тривалого перебування транспортного засобу в зоні перехрестя (підрозділ 2.3, формули 2.22-2.24). Система відстежує час перебування кожного об'єкта всередині цієї зони та генерує аномалію, якщо цей час перевищує встановлений поріг.

Зона світлофора – це прямокутна область на зображенні, що містить зображення світлофора. Адміністратор визначає цю зону методом defineTrafficLightROI(bbox), вказуючи координати прямокутника.

Ця зона використовується модулем детекції сигналу світлофора, який застосовує метод аналізу кольору (HSV аналіз, підрозділ 2.3, формули 2.19-2.20) для визначення поточного стану світлофора (червоний, жовтий або зелений сигнал). Точне визначення зони світлофора важливе для надійної роботи алгоритму виявлення проїзду на червоне світло.

Адміністратор встановлює числові пороги, що використовуються алгоритмами виявлення різних типів аномальних подій. Ці пороги передаються методом setThresholds(params):

1. Максимально дозволена швидкість (v_{max}). Граничне значення швидкості руху транспортного засобу, виражене у кілометрах на годину (км/г). Транспортні засоби, швидкість яких перевищує це значення, класифікуються як такі, що порушують швидкісний режим. Типові значення: 50 км/г для міських вулиць, 60-80 км/г для проспектів, 100-130 км/г для автострад.

2. Поріг аварійного гальмування (a_emergency). Граничне значення від'ємного прискорення (уповільнення), виражене у метрах на секунду в квадраті (м/с²), що вказує на аварійне гальмування. Від'ємні значення позначають уповільнення. Типове значення: від -4 до -6 м/с². Транспортні засоби з прискоренням менше цього порогу (більше за абсолютним значенням) класифікуються як такі, що здійснюють екстрене гальмування, що може вказувати на небезпечну ситуацію.

3. Максимальний час перебування у ROI (t_roi_max). Граничний час, виражений у секундах, протягом якого транспортний засіб може перебувати в

зоні перехрестя. Перевищення цього часу може вказувати на затор, поломку транспортного засобу або порушення правил дорожнього руху. Типове значення: 3-7 секунд залежно від розміру перехрестя.

Після завершення всіх кроків налаштування модуль калібрування передає зібрану інформацію до модуля конфігурації методом generateConfig(H, zones, thresholds). Модуль конфігурації формує структуровану конфігурацію сцени у форматі JSON, що включає:

1. Ідентифікатор камери.
2. Параметри калібрування (матриця H, опорні точки).
3. Визначення зон сцени (стоп-лінія, ROI, зона світлофора).
4. Пороги виявлення аномалій.

Приклад структури конфігураційного файлу у форматі json наведений на рисунку 3.3.

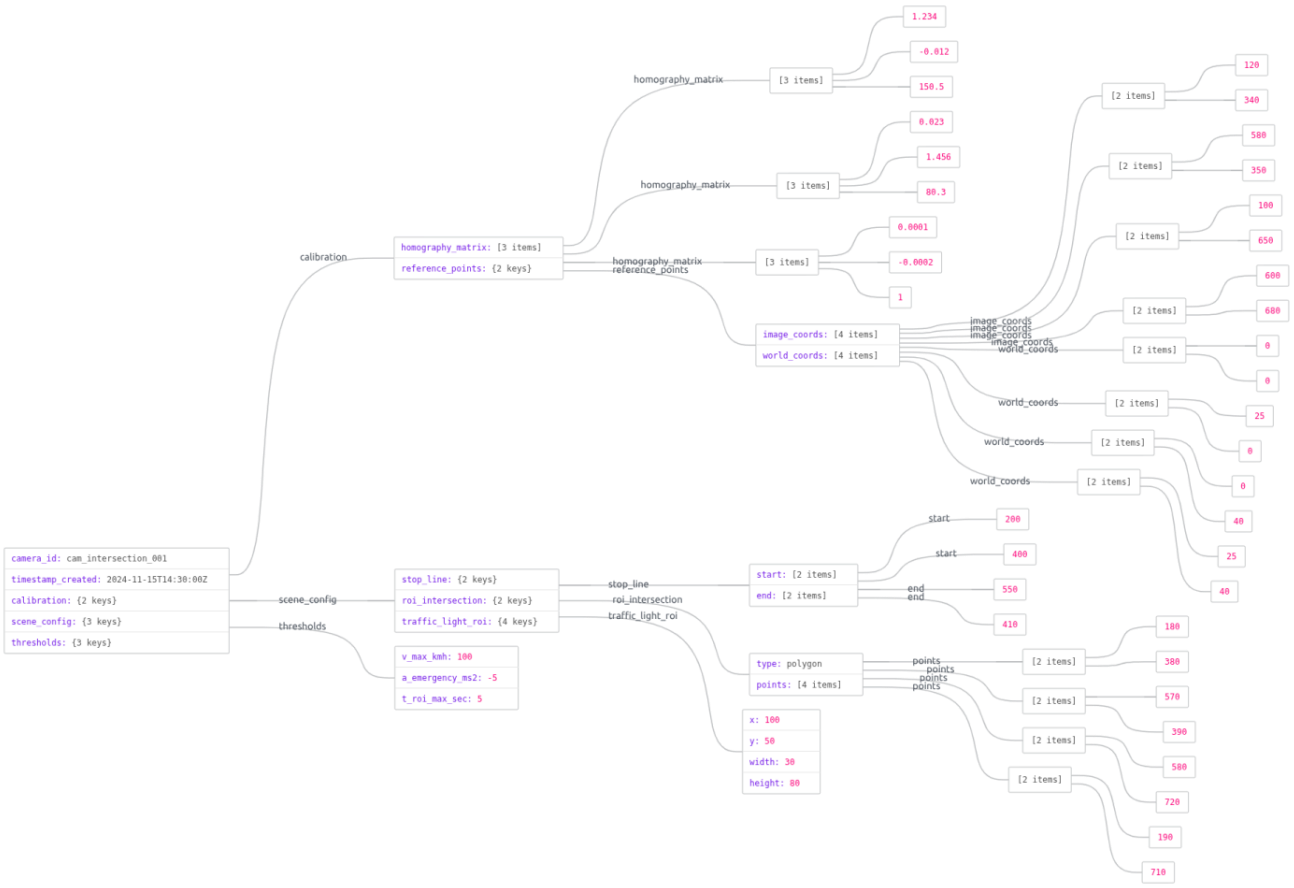


Рис. 3.3. Структура файлу json-конфігурації

Сформована конфігурація зберігається у базі даних методом saveConfig(config.json) з прив'язкою до унікального ідентифікатора камери. Це дозволяє системі автоматично завантажувати відповідну конфігурацію при ініціалізації обробки відеопотоку з конкретної камери.

По завершенні збереження конфігурації модуль калібрування повідомляє адміністратора про успішне завершення налаштування методом setupComplete(status). Система переходить у стан готовності до запуску runtime обробки.

Особливості етапу ініціалізації такі:

1. Одноразове виконання. Етап виконується один раз при встановленні камери. Повторне налаштування потрібне тільки при зміні положення камери або реконфігурації сцени.

2. Участь людини: Етап потребує активної участі кваліфікованого адміністратора, оскільки точність калібрування та правильність визначення зон впливають на якість виявлення аномалій.

3. Адаптивність: Можливість налаштування окремих параметрів для кожної камери забезпечує адаптацію системи до різноманітних умов спостереження (різні перехрестя, кути огляду, специфіка дорожнього руху).

4. Незалежність від обробки: Результати етапу ініціалізації (конфігураційний файл) зберігаються окремо від коду обробки, що дозволяє легко масштабувати систему на велику кількість камер без модифікації програмного забезпечення.

3.2.2 UML-діаграма послідовності інформаційної технології на етапі обробки відеопотоку

Етап обробки відеопотоку виконується автоматично системою без участі адміністратора після завершення етапу ініціалізації. Це безперервний

циклічний процес, що включає послідовну обробку кожного кадру відеопотоку з камери спостереження в режимі реального часу. Мета цього етапу – автоматично виявляти транспортні засоби, відстежувати їх траєкторії, обчислювати кінематичні параметри руху та ідентифікувати аномальні події, застосовуючи параметри калібрування та конфігурацію сцени, визначені на етапі ініціалізації. UML-діаграма послідовності інформаційної технології етапу обробки відеопотоку зображена на рисунку 3.4.

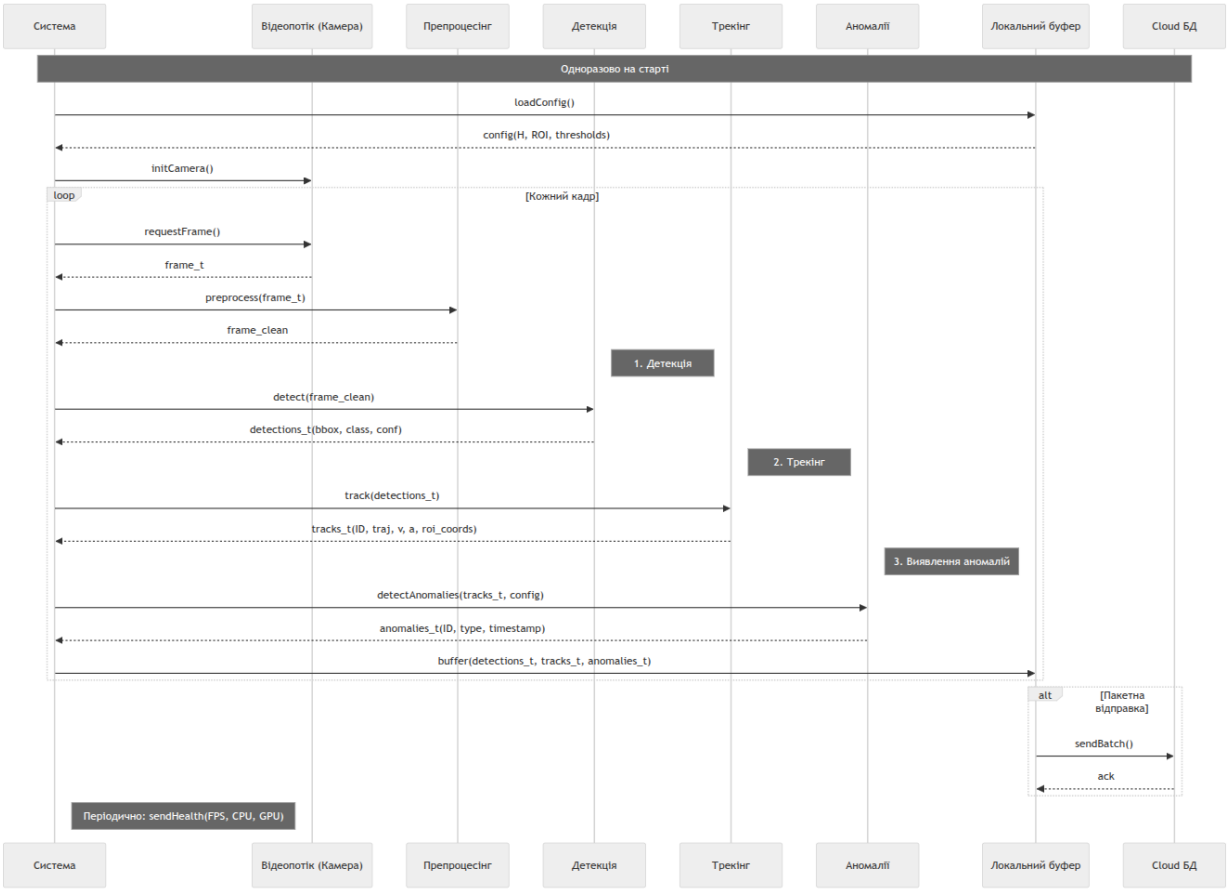


Рис. 3.4. UML-діаграма послідовності роботи інформаційної технології на етапі обробки відеопотоку

Перед початком циклічної обробки кадрів система виконує одноразову ініціалізацію своїх компонентів. Система звертається до бази даних методом loadConfig(camera_id), де camera_id – унікальний ідентифікатор камери,

відеопотік якої обробляється. База даних повертає збережену конфігурацію сцени, створену на першому етапі (Рис. 3.5):

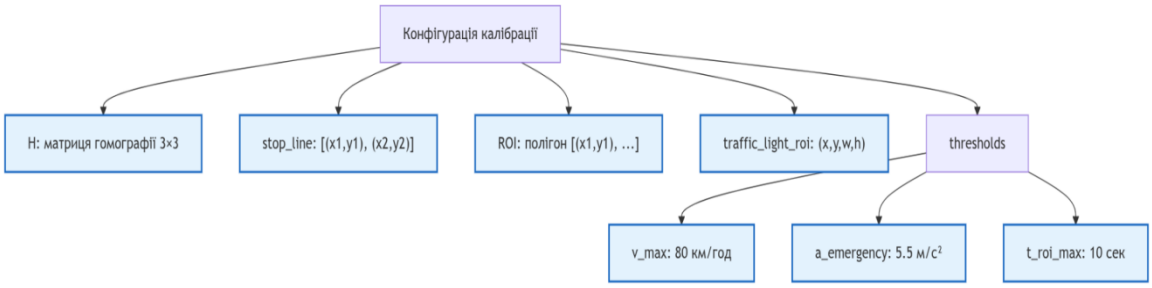


Рис. 3.5. Загальна схема json-конфігурації сцени на етапі ініціалізації камери

Ці параметри завантажуються в оперативну пам'ять системи та будуть використовуватися протягом всього часу обробки відеопотоку.

Система завантажує попередньо навчену модель нейронної мережі для детекції об'єктів, розроблену у підрозділі 2.1 (модифікована YOLOv8 з модулями C3Ghost-CBAM). Модель завантажується методом loadModel(model_path), де model_path вказує на файл моделі у форматі .pt (PyTorch), .onnx (ONNX Runtime), .tflite (TensorFlow Lite) або .engine (TensorRT) залежно від платформи виконання.

Завантажена модель ініціалізується на доступному обчислювальному пристрої (GPU, NPU або CPU) та переводиться у режим інференсу (evaluation mode), готовий до обробки вхідних кадрів.

Система ініціалізує модуль трекінгу об'єктів методом initialize(tracker_params), використовуючи алгоритм удосконаленого ByteTrack, розроблений у підрозділі 2.2.

Трекер створює порожні структури даних для зберігання активних треків, втрачених треків та історії траєкторій.

Після завершення ініціалізації система переходить до основного циклу обробки відеопотоку. Цей цикл виконується безперервно з частотою, визначеною швидкістю надходження кадрів (зазвичай 25-30 кадрів на секунду для відеопотоків реального часу).

Система запитує черговий кадр з джерела відео методом getFrame(). Джерелом відео може бути:

- RTSP-потік з IP-камери;

- локальний відеофайл (для тестування або обробки архівних записів);

- HTTP-потік.

Джерело повертає кадр frame_t у форматі RGB або BGR з роздільністю H×W×3 (висота × ширина × 3 канали кольору), де індекс t позначає номер кадру в послідовності.

Далі отриманий кадр передається до модуля детекції методом detect(frame_t). Модуль детекції виконує наступні операції:

1. Попередня обробка кадру: масштабування до розміру 640×640 пікселів (вхідний розмір моделі), нормалізація значень пікселів до діапазону [0, 1].

2. Інференс нейронної мережі: обробка кадру модифікованою моделлю YOLOv8 + C3Ghost-CBAM (детально описаною у підрозділі 2.1). Мережа виконує:

- вилучення ознак через backbone з модулями C3Ghost;

- агрегацію багатомасштабних ознак через neck з FPN/PAN;

- посилення важливих ознак через CBAM-модулі;

- генерацію передбачень через head (безякірний підхід).

3. Постобробка: застосування Non-Maximum Suppression (NMS) для усунення дублюючих детекцій з порогом IoU = 0,45, фільтрація детекцій з низькою впевненістю (confidence < 0,3).

Модуль детекції повертає список детекцій для поточного кадру:

detections_t = [

{bbox: (x, y, w, h), class: “car“, confidence: 0.92},

{bbox: (x, y, w, h), class: "truck", confidence: 0.87},

...

]

де bbox – обмежувальна рамка (bounding box) у форматі (x_center, y_center, width, height).

Детекції поточного кадру передаються до модуля трекінгу методом update(detections_t). Модуль трекінгу виконує алгоритм удосконаленого ByteTrack (підрозділ 2.2):

1. Розділення детекцій: детекції розділяються на високопорогові (confidence ≥ 0.5) та низькопорогові (0,1 ≤ confidence < 0,5).

2. Перша асоціація: високопорогові детекції асоціюються з існуючими активними треками за допомогою IoU-метрики та Угорського алгоритму (Hungarian algorithm).

3. Друга асоціація: низькопорогові детекції асоціюються з треками, що не отримали асоціацій на першому етапі. Це дозволяє зберігати треки об'єктів з тимчасово низькою впевненістю детекції (через оклюзії, погане освітлення).

4. Застосування удосконалень з підрозділу 2.2:

- IoU-suppression: зменшення конфліктів між близькими треками
- Захист втрачених треків: збереження треків у буфері протягом N кадрів (N=30)
- Уточнення з DIoU: використання Distance-IoU замість звичайного IoU для кращої локалізації

5. Управління життєвим циклом: створення нових треків для неасоційованих детекцій, видалення треків без оновлень протягом max_age кадрів. Для кожного активного треку система застосовує матрицю гомографії H (отриману на етапі ініціалізації) для перетворення координат центру обмежувальної рамки з піксельних координат у світові (метричні) координати. Це виконується внутрішнім методом трекера applyHomography(tracks, H).

Для кожного треку з центром bbox (x_pixel, y_pixel):

$x_world = (h_{11} \cdot x_pixel + h_{12} \cdot y_pixel + h_{13}) / (h_{31} \cdot x_pixel + h_{32} \cdot y_pixel + h_{33})$

$y_world = (h_{21} \cdot x_pixel + h_{22} \cdot y_pixel + h_{23}) / (h_{31} \cdot x_pixel + h_{32} \cdot y_pixel + h_{33})$

Отримані світові координати додаються до історії траєкторії треку.

На основі світових координат у різні моменти часу система обчислює кінематичні параметри руху для кожного треку методом computeKinematics(tracks):

1. Швидкість обчислюється як відстань, пройдена між двома послідовними кадрами, поділена на проміжок часу:

$$v(t) = \sqrt{[(x(t) - x(t-1))^2 + (y(t) - y(t-1))^2]} / \Delta t$$

де $\Delta t = 1/fps$ (для fps=30, $\Delta t \approx 0.033$ с). Швидкість виражається в м/с та конвертується у км/г множенням на 3.6.

2. Прискорення обчислюється як зміна швидкості за одиницю часу:

$$a(t) = [v(t) - v(t-1)] / \Delta t$$

Від'ємне прискорення вказує на уповільнення (гальмування).

3. Згладжування: для зменшення впливу шумів вимірювань застосовується ковзне середнє або фільтр Калмана до обчислених значень швидкості та прискорення.

Модуль трекінгу повертає оновлені треки з усією необхідною інформацією (рис 3.6).

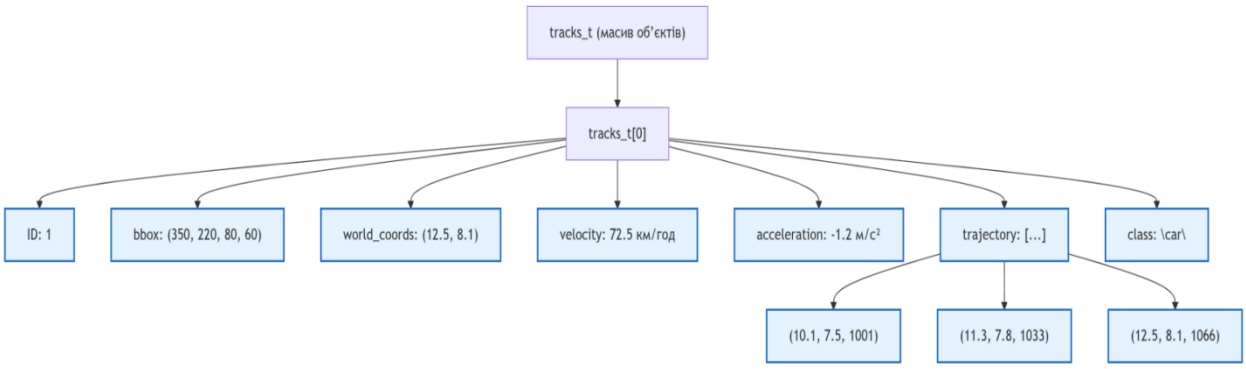


Рис. 3.6. Схема словника який повертається модулем трекінгу

Система передає оновлені треки до модуля виявлення аномалій, який послідовно застосовує чотири алгоритми виявлення, кожен з яких реалізує відповідний метод з підрозділу 2.3.

Метод детекції проїзду на червоне світло detectRedLight(tracks, traffic_light_roi, stop_line) виконує наступні кроки:

1. Визначення стану світлофора: система аналізує область зображення, визначену параметром traffic_light_roi (отриманим з конфігурації), застосовуючи HSV-аналіз згідно з формулами (2.19-2.20) з підрозділу 2.3. Алгоритм обчислює співвідношення інтенсивності червоного каналу до інших каналів та частку червоних пікселів у зоні світлофора, визначаючи поточний стан (RED, YELLOW, GREEN).

2. Перевірка перетину стоп-лінії: для кожного треку перевіряється, чи перетинає його траєкторія стоп-лінію (визначену параметром stop_line). Перетин визначається шляхом аналізу, чи змінюється знак векторного добутку для послідовних позицій треку відносно вектора стоп-лінії.

3. Фільтрація за швидкістю: застосовується додаткова умова (формула 2.21 з підрозділу 2.3) – транспортний засіб повинен мати швидкість вище мінімального порогу (v > $v_{\underset{crossing}{min}}$, наприклад 5 км/г) в момент перетину. Це усуває хибні спрацювання для зупинених транспортних засобів біля стоп-лінії.

4. Генерація аномалії: якщо всі умови виконані (світлофор показує RED, трек перетинає стоп-лінію, швидкість достатня), система генерує подію аномалії типу “RED_LIGHT_VIOLATION“.

Метод детекції перевищення швидкості detectSpeeding(tracks, v_{max}) порівнює поточну швидкість кожного треку з максимально дозволеною швидкістю v_{max} (отриманою з конфігурації):

IF track.velocity > v_{max} THEN

generate_anomaly(“SPEEDING“, track.ID, track.velocity)

Згідно з формулою (2.14 підрозділу 2.3), швидкість обчислюється на основі світових координат, отриманих через гомографічне перетворення, що забезпечує точність вимірювання незалежно від положення об'єкта на зображенні.

Метод детекції аварійного гальмування detectEmergencyBrake(tracks, a_emergency) аналізує прискорення кожного треку:

IF track.acceleration < a_emergency THEN

generate_anomaly("EMERGENCY_BRAKE", track.ID, track.acceleration)

Поріг a_emergency (наприклад, -5 м/с²) визначає граничне від'ємне прискорення, що класифікується як екстрене гальмування. Виявлення таких подій може вказувати на небезпечні дорожні ситуації.

Метод детекції надмірно тривалого перебування у ROI detectROIOverflow(tracks, ROI, t_roi_max) відстежує час перебування кожного треку всередині зони перехрестя, визначеної полігоном ROI:

1. Визначення належності до ROI: для кожного треку застосовується алгоритм point-in-polygon для перевірки, чи знаходиться центр обмежувальної рамки всередині ROI (формула 2.24 з підрозділу 2.3).

2. Вимірювання тривалості перебування: якщо трек входить у ROI, система фіксує час входу t_{entry}. При кожному наступному кадрі обчислюється тривалість перебування $t_{duration} = (t_{current} - t_{entry}) / fps$ (формула 2.23 з підрозділу 2.3).

3. Генерація аномалії: якщо тривалість перевищує поріг t_roi_max (наприклад, 5 секунд), генерується аномалія "ROI_OVERFLOW".

Модуль виявлення аномалій повертає список виявлених аномалій для поточного кадру (рис. 3.7).

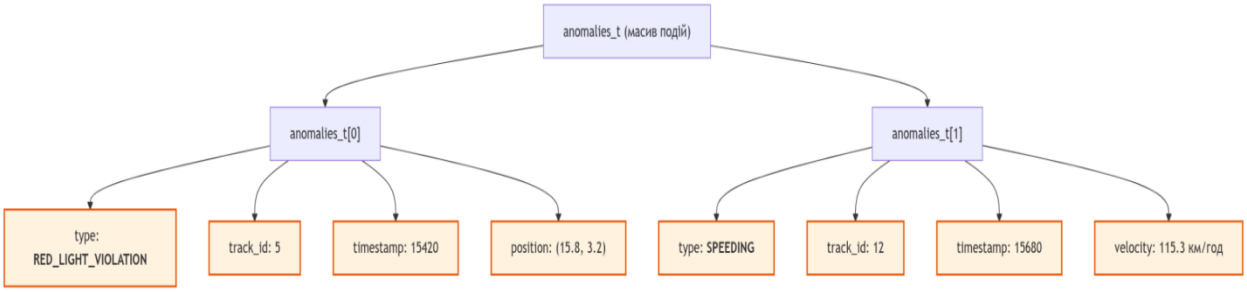


Рис. 3.7. Схема словника який повертається модулем виявлення аномалій

Усі результати обробки поточного кадру – детекції, оновлені треки та виявлені аномалії – передаються до модуля зберігання методом store(detections_t, tracks_t, anomalies_t). Модуль зберігання виконує наступні операції:

1. Запис у базу даних: результати зберігаються у реляційній базі даних (PostgreSQL) або базі часових рядів (TimescaleDB) з прив'язкою до часової мітки та ідентифікатора камери.

2. Індексування: створюються індекси за часовою міткою, ID треку та типом аномалії для швидкого пошуку.

3. Публікація подій: у разі виявлення аномалій система може публікувати події у черги повідомлень (MQTT, Kafka) або відправляти сповіщення (email, SMS, webhook) відповідним службам або адміністраторам.

4. Архівування: періодично старі дані архівуються або агрегуються для зменшення обсягу зберігання.

Після завершення обробки поточного кадру система повертається на початок циклу для захоплення та обробки наступного кадру. Цикл продовжується безперервно протягом всього часу роботи системи.

3.2.3 UML-діаграма активності навчання моделі детекції

Ця діаграма відображає повний цикл розробки моделі детекції об'єктів на основі YOLOv8 для дорожної інфраструктури (рис. 3.8).

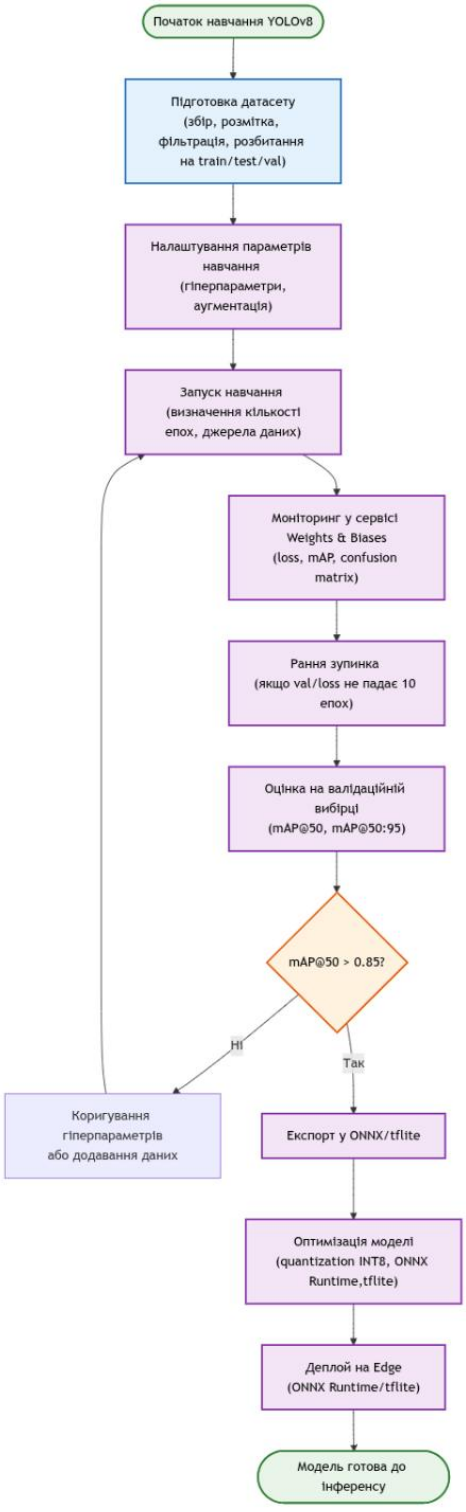


Рис. 3.8. UML-діаграма активності навчання моделі детекції

Етапи процесу

1. Підготовка датасету

Процес починається зі збору датасету дорожного середовища, подальшої анотації об'єктів за допомогою спеціалізованих інструментів (CVAT, LabelStudio). Визначаються класи, наприклад, легкові авто, вантажівки, автобуси, мотоцикли, пішоходи, велосипеди та світлофори. Розмітка валідується за строгими критеріями (IoU > 0,8, мінімум 2 анотатори). Датасет розбивається на навчальну (70%), валідаційну (20%) та тестову (10%) вибірки.

3. Налаштування параметрів навчання

Створюється конфіг-файл data.yaml з гіперпараметрами (кількість класів, розмір зображення 640×640, batch size 16). Налаштовується аугментація даних – мозаїка, повороти, додавання шуму, трансформація HSV для підвищення робастності моделі.

4. Етап навчання

Запускається навчання YOLOv8 nano на 100 епохах з моніторингом метрик у Weights & Biases (loss, mAP, confusion matrix). Активується зупинка – якщо валідаційна втрата не покращується 10 епох поспіль. Модель оцінюється на валідаційній вибірці за метриками mAP@50 та mAP@50:95.

5. Рішення

Якщо mAP@50 > 0,85:

Ні → коригування гіперпараметрів або додавання даних, повернення до навчання

Так → перехід до деплойменту

6. Оптимізація та деплой

Модель експортується у ONNX формат з половинною точністю (half precision). Проводиться оптимізація: INT8 квантизація та ONNX Runtime. Фіналізація на edge-пристрої через Docker контейнер.

3.3 Вибір програмних засобів та технологічного стеку

Підбір програмних інструментів для реалізації інформаційної технології аналізу відеопотоку дорожнього руху здійснювався з урахуванням комплексу вимог, що випливають з особливостей Edge-Cloud архітектури та необхідності обробки даних у реальному часі. Ключовою особливістю запропонованої архітектури є виконання всіх обчислювально-інтенсивних операцій – детекції об'єктів, багатооб'єктного трекінгу та виявлення аномалій – безпосередньо на Edge-рівні, що дозволяє мінімізувати мережевий трафік та забезпечити автономність роботи при тимчасовій втраті з'єднання з сервером.

Функціональні вимоги до програмних засобів:

- продуктивність – здатність виконувати повний пайплайн обробки (детекція → трекінг → аналіз аномалій) у реальному часі без втрати кадрів (мінімум 25 FPS на Edge-рівні);

- автономність – можливість локального виявлення та фіксації аномалій на Edge-пристрої без постійного з'єднання з Cloud;

- масштабованість – можливість безболісно додавати нові камери чи обчислювальні вузли без перебудови системи;

- модульність – незалежність компонентів для спрощення розробки, тестування та підтримки;

- відмовостійкість – збереження працездатності системи при тимчасових збоях окремих компонентів або втраті мережевого з'єднання.

Технічні вимоги:

- підтримка GPU-прискорення для нейромережевих обчислень як на Edge-рівні (NVIDIA Jetson), так і на Cloud-рівні (для навчання моделей);

- сумісність з edge-пристроями обмеженої потужності (NVIDIA Jetson Orin Nano);

- підтримка сучасних архітектур нейронних мереж та можливість їх оптимізації (квантизація INT8, pruning);

- ефективне використання пам'яті для одночасної роботи детектора, трекера та аналізатора аномалій на Edge-пристрої;

- наявність засобів для серіалізації та ефективної передачі метаданих між рівнями архітектури.

Організаційні вимоги:

- популярність і підтримка спільноти – наявність детальної документації, активного розвитку та бібліотек розширень;

- вартість – орієнтація на безкоштовні або open-source рішення з ліцензіями, що дозволяють комерційне використання;

- стабільність – використання перевірених технологій з довготривалою підтримкою.

Для реалізації системи було обрано двошарову архітектуру – Edge (повна обробка відеопотоку включно з виявленням аномалій) та Cloud (агрегація даних, навчання моделей, візуалізація та довготривале зберігання), що дозволяє оптимально розподілити обчислювальне навантаження та мінімізувати мережевий трафік до передачі лише метаданих про виявлені події.

3.3.1 Програмні засоби рівня Edge

Рівень Edge є основним компонентом розробленої інформаційної технології, оскільки саме на ньому виконується весь обчислювальний пайплайн обробки відеопотоку: захоплення кадрів з камери, детекція транспортних засобів та світлофорів, багатооб'єктний трекінг з присвоєнням унікальних ідентифікаторів, аналіз траєкторій та виявлення аномалій (проїзд на червоний сигнал, перевищення швидкості, аварійне гальмування, тривале перебування в ROI). Використання Edge-Cloud архітектури обґрунтоване необхідністю забезпечення автономності системи. Основні обчислення виконуються на периферійних пристроях, що дозволяє системі працювати навіть при втраті мережевого з’єднання, а також зменшує залежність від пропускної здатності

каналів зв’язку. До Cloud-рівня передаються лише метадані про виявлені події та агрегована статистика, що забезпечує економію мережевого трафіку на 99,6% порівняно з передачею повного відеопотоку.

Основною мовою програмування для Edge-рівня обрано Python версії 3.8 і вище. Вибір обґрунтовано наступними факторами: гнучкість та швидкість розробки завдяки динамічній типізації та інтерпретованості; широкий вибір бібліотек для комп'ютерного зору та машинного навчання; повна сумісність з платформами NVIDIA Jetson через JetPack SDK; можливість швидкого розгортання з підтримкою hot-reload без перезапуску системи. Незважаючи на те, що Python поступається компільованим мовам (C++, Rust) у швидкодії, важливі операції – інференс нейронних мереж, обробка зображень та матричні обчислення для трекінгу – делегуються оптимізованим бібліотекам, написаним на C++ з CUDA-прискоренням, що нівелює цей недолік.

Фреймворки для інференсу нейронних мереж. Оскільки детекція об'єктів виконується на Edge-пристроях з обмеженими ресурсами, вибір фреймворку інференсу є важливим для досягнення необхідної продуктивності. Було проаналізовано кілька альтернативних рішень:

- PyTorch Mobile – оптимізована версія фреймворку PyTorch для пристроїв з обмеженими ресурсами, що забезпечує повну сумісність з моделями, розробленими в PyTorch, проте характеризується відносно великим розміром бібліотеки (~45 МБ) та вищим споживанням пам'яті;

- ONNX Runtime – універсальний рушій для виконання моделей у форматі ONNX (Open Neural Network Exchange), що озабезпечує широку сумісність з різним апаратним забезпеченням та компактний розмір (~12 МБ);

- TensorFlow Lite – оптимізовано для мобільних пристроїв та мікроконтролерів, проте вимагає конвертації моделей з інших форматів та має обмежену підтримку деяких операцій;

- TensorRT – забезпечує максимальну продуктивність на GPU NVIDIA завдяки глибокій оптимізації графа обчислень, проте обмежений апаратною платформою та має більший розмір.

Порівняльний аналіз фреймворків інференсу для Edge-рівня (Jetson Orin Nano) для моделі YOLOv8-nano наведено в таблиці 3.1.

Таблиця 3.1 – Порівняння фреймворків інференсу для Edge-пристроїв

Фреймворк	FPS	Розмір бібліотеки	RAM (інференс)	Формати моделей
Pytorch Mobile	40-50	45 Мб	1-1.3 Гб	.pt
ONNX Runtime	50-60	12 Мб	0.7-0.9 Гб	.onnx
TensorFlow Lite	25-35	8 Мб	0.6-0.8 Гб	.tflite
TensorRT	65-100+	80-150 Мб	0.6-1.0 Гб	.engine

За результатами порівняльного аналізу фреймворків інференсу для edge-пристроїв перевагу надано TensorRT як основному рушію виконання нейронної мережі. Таке рішення зумовлене такими перевагами цього фреймворку на апаратному забезпеченні NVIDIA Jetson Orin Nano:

- максимальна продуктивність – завдяки глибокій оптимізації під GPU (включаючи FP16 та INT8-квантування) забезпечується найвища швидкість інференсу порівняно з іншими фреймворками (ONNX Runtime, TensorFlow Lite, PyTorch Mobile);

- ефективне використання апаратних ресурсів – повна утилізація Tensor Core та CUDA-ядер, що особливо важливо для обмежених edge-пристроїв;

- підтримка оптимізованої моделі – експорт YOLOv8n у формат .engine з можливістю статичної та динамічної побудови engine для найкращої швидкодії;

- енергоефективність – краще співвідношення FPS/ват порівняно з універсальними рішеннями.

Для забезпечення універсальності та спрощення розробки передбачено можливість резервного використання ONNX Runtime (з TensorRT Execution Provider), що дозволяє запускати модель у форматі .onnx без повної перебудови engine. Однак у штатному режимі роботи системи пріоритетом є TensorRT, оскільки він забезпечує найвищу частоту кадрів та найкраще використання обмеженого обсягу пам’яті (4 ГБ) при одночасній роботі трекера об’єктів та модуля аналізу аномалій.

Бібліотека обробки зображень OpenCV. Для обробки зображень використовується бібліотека OpenCV версії 4.8 і вище у мінімальній збірці. OpenCV надає необхідний набір функцій для: читання відеопотоків з різних джерел (RTSP, HTTP, локальні файли); попередньої обробки кадрів (масштабування до 640×640 пікселів методом letterbox, нормалізація значень пікселів до діапазону [0, 1] та конвертація BGR→RGB для модуля детекції; конвертація BGR→HSV для модуля HSV-аналізу стану сигналу світлофора); постобробки результатів детекції (відображення bounding boxes, траєкторій); роботи з калібрувальними даними камери та гомографічним перетворенням для обчислення швидкості транспортних засобів у реальних координатах.

Для мінімізації використання ресурсів на Edge-пристроях OpenCV збирається з обмеженим набором модулів (core, imgproc, videoio, calib3d).

Реалізація трекера та аналізатора аномалій. Розроблений трекер ByteTrack+ та модуль виявлення аномалій реалізовані на Python з використанням бібліотек NumPy та SciPy для матричних обчислень. Фільтр Калмана для прогнозування позицій об'єктів реалізовано з використанням оптимізованих NumPy-операцій, що забезпечує швидкість обробки понад 1000 треків на кадр без суттєвого навантаження на CPU. Угорський алгоритм для асоціації детекцій з треками використовує реалізацію з бібліотеки SciPy (scipy.optimize.linear_sum_assignment), яка є оптимізованою версією на C.

Модуль виявлення аномалій працює безпосередньо з результатами трекера, аналізуючи траєкторії об'єктів у реальному часі. Для кожного кадру

виконуються перевірки: перетин стоп-лінії при червоному сигналі світлофора (через аналіз HSV-каналів детектованого світлофора), обчислення швидкості через гомографічне перетворення координат, детекція аварійного гальмування (різке зменшення швидкості), перевірка тривалості перебування в ROI. Обчислювальна складність цих операцій є незначною порівняно з детекцією, що дозволяє виконувати їх на Edge без втрати продуктивності.

Протокол передачі даних MQTT. Для передачі метаданих про виявлені аномалії та агрегованої статистики між Edge та Cloud рівнями обрано протокол MQTT (Message Queuing Telemetry Transport) з використанням бібліотеки Paho MQTT Client. Оскільки на Cloud передаються лише метадані (ідентифікатор камери, час події, тип аномалії, координати об'єкта, швидкість), а не повний відеопотік, вимоги до пропускної здатності каналу є мінімальними. Вибір MQTT обґрунтовано наступними характеристиками: легка бібліотека розміром близько 50 КБ; підтримка трьох рівнів якості обслуговування (QoS 0, 1, 2) для гарантованої доставки важливих повідомлень про аномалії; автоматична повторна доставка повідомлень при відновленні з'єднання після збоїв; локальне буферування повідомлень на Edge при втраті зв'язку з сервером.

Порівняння протоколів передачі даних наведено в таблиці 3.2.

Таблиця 3.2 – Порівняння протоколів передачі даних

Протокол	Затримки	Накладні витрати	Робота при втраті з'єднання	Складність
MQTT	5-15 мс	низькі	Буферування + QoS	низька
HTTP REST	50-200 мс	середні	Втрата даних	низька
qRPC	10-30 мс	низькі	Потребує повторного з'єднання	висока
WebSocket	5-20 мс	середні	Втрата з'єднання	середня

Структура повідомлення MQTT включає: camera_id (ідентифікатор камери), timestamp (час події в UTC), event_type (тип аномалії: red_light, speeding, emergency_brake, roi_violation), track_id (ідентифікатор об'єкта), bbox (координати обмежувального прямокутника), speed_kmh (швидкість у км/год, якщо застосовно), confidence (впевненість детекції). Середній розмір повідомлення становить 200-500 байт.

3.3.2 Програмні засоби рівня Cloud

Рівень Cloud виконує допоміжні функції, що не потребують обробки в реальному часі: агрегація та зберігання даних від множини Edge-пристроїв, навчання та оновлення моделей детекції, візуалізація статистики через веб-інтерфейс, формування звітів та сповіщень. На відміну від Edge-рівня, Cloud не виконує детекцію, трекінг чи виявлення аномалій – він отримує вже готові результати у вигляді метаданих.

Фреймворк глибокого навчання PyTorch. На серверному рівні використовується PyTorch версії 2.0 і вище як основний фреймворк для навчання моделей детекції. PyTorch забезпечує: імперативне програмування з

інтуїтивним налагодженням та швидким прототипуванням; автоматичну диференціацію через механізм autograd для обчислення градієнтів; повну підтримку CUDA для прискорення навчання на GPU NVIDIA (RTX 4090, A100); можливість розподіленого навчання через DistributedDataParallel для multi-GPU конфігурацій; компіляцію моделей через torch.compile (PyTorch 2.0+) для автоматичної оптимізації.

Бібліотека Ultralytics YOLO. Для навчання та експорту моделей сімейства YOLO використовується бібліотека Ultralytics YOLOv8, яка надає: уніфікований API для навчання, валідації та експорту моделей; підтримку кастомних архітектур через YAML-конфігурації (для інтеграції модулів C3Ghost-CBAM); автоматичний експорт у формати ONNX, TensorRT, TFLite для розгортання на Edge; інтеграцію з платформами логування експериментів (Weights & Biases, MLflow).

Система управління базами даних PostgreSQL. Для зберігання структурованих даних використовується PostgreSQL версії 15 і вище з розширенням TimescaleDB для ефективної роботи з часовими рядами. PostgreSQL забезпечує: ACID-гарантії для надійності транзакцій; підтримку JSON/JSONB для зберігання конфігурацій камер та параметрів калібрування; розширення PostGIS для геопросторових запитів. TimescaleDB забезпечує: автоматичне партиціонування за часом через механізм hypertables; стиснення даних до 90% від початкового розміру; оптимізовані запити по часових діапазонах для формування звітів.

Кешування та черги повідомлень Redis. Для кешування та асинхронної обробки вхідних повідомлень від Edge-пристроїв використовується Redis версії 7.0 і вище. Redis забезпечує: буферування вхідних MQTT-повідомлень перед записом у базу даних; механізм Pub/Sub для real-time сповіщень операторів про важливі аномалії; кешування агрегованої статистики для швидкого відображення у веб-інтерфейсі.

REST API та веб-фреймворк Flask. Серверна частина API реалізована на Flask версії 3.0 і вище з розширенням Flask-SocketIO для двостороннього зв'язку в реальному часі. API забезпечує: отримання метаданих від Edge-пристроїв; надання даних для веб-інтерфейсу візуалізації; управління конфігураціями камер та параметрами калібрування; WebSocket-з'єднання для real-time оновлення дашбордів.

3.3.3 Інструменти розгортання та DevOps

Контейнеризація Docker. Розгортання всіх компонентів системи відбувається у контейнерах Docker, що забезпечує: ізоляцію сервісів один від одного; відтворюваність середовища на будь-якій платформі; версіонування образів для можливості rollback; оптимізацію розміру образів через multi-stage builds.

Для Edge-рівня використовуються базові образи NVIDIA L4T (Linux for Tegra) з попередньо встановленими CUDA та cuDNN, що забезпечує роботу на платформах Jetson Nano, Jetson Xavier NX. Docker-образ Edge-модуля включає: ONNX Runtime з CUDA Execution Provider, мінімальну збірку OpenCV, модулі трекера ByteTrack+ та аналізатора аномалій, Paho MQTT Client. Розмір образу оптимізовано до ~1.5 ГБ.

Для Cloud-рівня використовуються офіційні образи Python та PostgreSQL. Оркестрація багатокомпонентних сервісів забезпечується Docker Compose. Типова конфігурація Cloud включає сервіси: api-service (REST API та WebSocket), postgres (база даних з TimescaleDB), redis (кешування та черги), mosquitto (MQTT брокер для прийому даних від Edge).

Безперервна інтеграція GitLab CI. Безперервна інтеграція та розгортання реалізовані через GitLab CI/CD, що дозволяє: автоматизувати запуск юніт-тестів та інтеграційних тестів при кожному merge request; виконувати статичний аналіз коду (flake8, mypy) для контролю якості;

автоматично збирати Docker-образи для Edge та Cloud компонентів; здійснювати автоматичний деплой на staging та production середовища.

3.4 Методика оцінювання якості інформаційної технології та набори даних

Для об'єктивної оцінки ефективності розробленої інформаційної технології аналізу відеопотоку дорожнього руху необхідно визначити критерії оцінювання та сформувати репрезентативні набори даних для проведення експериментів. У цьому підрозділі систематизовано часткові метрики якості для кожного компонента системи (детекція, трекінг, виявлення аномалій) з посиланнями на відповідні формули з розділу 2, а також описано набори даних, використані для навчання моделей та апробації інформаційної технології.

3.4.1 Часткові метрики оцінювання якості

Метрики якості детекції об'єктів.

Для оцінки якості детекції транспортних засобів використовуються метрики, визначені в підрозділі 2.1.

Precision (точність) – частка правильних детекцій серед усіх позитивних передбачень моделі, визначається формулою (2.7) підрозділу 2.1:

Recall (повнота) – частка правильних детекцій серед усіх реальних об'єктів, визначається формулою (2.8) підрозділу 2.1:

F1-score (F-міра) – гармонійне середнє між Precision та Recall, що забезпечує збалансовану оцінку якості детекції:

$$F1 = 2 \times (Precision \times Recall) / (Precision + Recall). \quad (3.1)$$

F1-score приймає значення від 0 до 1, де 1 відповідає ідеальній детекції. Ця метрика є особливо корисною, коли необхідно знайти компроміс між точністю та повнотою.

Average Precision (AP) – площа під кривою Precision-Recall, що характеризує якість детекції для одного класу об'єктів при різних порогах впевненості.

mean Average Precision (mAP) – усереднене значення AP за всіма класами об'єктів. Відповідно до підрозділу 2.1, у дослідженні використовуються два варіанти метрики:

- mAP@0,50 – значення mAP при порозі IoU (Intersection over Union) рівному 0.50, що є стандартним порогом для задач детекції;

- mAP@0,50:0,95 – усереднене значення mAP, розраховане для порогів IoU від 0,50 до 0,95 з кроком 0,05, що забезпечує більш суворе оцінювання точності локалізації об'єктів.

Метрики якості багатооб'єктного трекінгу.

Для оцінки якості трекінгу використовуються стандартні метрики бенчмарку MOT (Multiple Object Tracking).

MOTA (Multiple Object Tracking Accuracy) – загальна метрика трекінгу, що враховує три типи помилок: хибнопозитивні детекції, пропуски та перемикання ідентичностей. MOTA обчислюється за формулою:

$$MOTA = 1 - (FN + FP + IDSW) / GT, \quad (3.2)$$

де FN – кількість пропущених об'єктів (False Negatives); FP – кількість хибнопозитивних детекцій (False Positives); IDSW – кількість перемикань ідентичностей (ID Switches), коли об'єкту помилково присвоюється інший ідентифікатор; GT – загальна кількість істинних об'єктів у всіх кадрах (Ground Truth).

MOTA може приймати значення від -∞ до 1, де 1 відповідає ідеальному трекінгу без жодних помилок. Від'ємні значення свідчать про те, що кількість помилок перевищує кількість істинних об'єктів.

IDF1 (ID F1 Score) – метрика, що оцінює якість збереження ідентичностей об'єктів протягом усієї траєкторії. IDF1 є гармонійним середнім між точністю та повнотою ідентифікації:

$$IDF1 = 2 \times IDTP / (2 \times IDTP + IDFP + IDFN), \quad (3.3)$$

де IDTP – кількість правильно ідентифікованих детекцій (коли детекція асоційована з правильним треком); IDFP – кількість хибнопозитивних ідентифікацій; IDFN – кількість пропущених ідентифікацій.

IDF1 приймає значення від 0 до 1 і є особливо важливою метрикою для задач, де необхідно зберігати ідентичність об'єкта протягом тривалого часу, як у випадку виявлення аномалій дорожнього руху.

ID Switches (IDSW) – абсолютна кількість випадків, коли об'єкту присвоюється новий ідентифікатор замість попереднього. Мінімізація ID Switches є необхідною для коректного виявлення аномалій, оскільки перемикання ідентичностей може призвести до розриву траєкторії та втрати інформації про поведінку об'єкта.

Метрики якості виявлення аномалій.

Виявлення аномалій у дослідженні сформульовано як задачу бінарної класифікації, де кожна подія класифікується як “аномалія“ або “норма“. Оскільки аномалії є рідкісними подіями, класи є суттєво незбалансованими, що вимагає використання метрик, стійких до такого дисбалансу.

MCC (Matthews Correlation Coefficient) – коефіцієнт кореляції Меттьюза, визначений формулою (2.25) підрозділу 2.3, є основною метрикою для оцінки якості виявлення аномалій

MCC приймає значення від -1 до +1, де +1 відповідає ідеальній класифікації, 0 – випадковому вгадуванню, -1 – повній інверсії класифікації. На

відміну від accuracy, MCC враховує всі чотири елементи матриці помилок та є стійким до незбалансованості класів.

Матриця помилок (Confusion Matrix) – таблиця, що містить значення TP, TN, FP, FN для візуалізації результатів класифікації та аналізу типів помилок.

Метрики продуктивності системи.

Для оцінки продуктивності інформаційної технології на Edge-пристрої використовуються наступні метрики:

FPS (Frames Per Second) – кількість кадрів, оброблених за секунду повним пайплайном (детекція → трекінг → виявлення аномалій). Цільове значення: ≥ 25 FPS для обробки в реальному часі.

Повна затримка – час від захоплення кадру до формування результату виявлення аномалії, вимірюється в мілісекундах. Цільове значення: < 100 мс.

Завантаження GPU/CPU – відсоток завантаження обчислювальних ресурсів під час обробки відеопотоку.

Використання RAM – обсяг оперативної пам'яті, що споживається системою.

Зведену інформацію про критерії оцінювання якості на основі відібраних часткових метрик наведено в таблиці 3.3.

Таблиця 3.3 – Критерії оцінювання ефективності ІТ

Компонент	Метрика	Діапазон значень	Цільове значення
Детекція	mAP@0.50	0–1	> 0,80
Детекція	mAP@0,50:0,95	0–1	> 0,50
Детекція	F1-score	0–1	> 0,80
Трекінг	MOTA	-∞ – 1	> 0,70

Продовження табл. 3.3

Компонент	Метрика	Діапазон значень	Цільове значення
Трекінг	IDF1	0–1	> 0,70
Трекінг	ID Switches	0–∞	мінімізація
Аномалії	MCC	-1 – +1	> 0,75
Продуктивність	FPS (Edge)	0–∞	≥ 25
Продуктивність	Затримка	0–∞ мс	< 100 мс

3.4.2 Набори даних для навчання та тестування

Набір даних для навчання детектора.

Для навчання модифікованої моделі YOLOv8 з модулями C3Ghost-CBAM (підрозділ 2.1) сформовано комбінований набір даних, що складається з двох джерел:

Власний датасет містить 2000 зображень, отриманих з відеокамер спостереження за дорожнім рухом з різних джерел (YouTube, публічні архіви відеозаписів). Зображення характеризуються різноманітними умовами зйомки:

- час доби: день (60%), сутінки (20%), ніч (20%);
- погодні умови: ясно (50%), хмарно (30%), дощ (15%), туман (5%);
- ракурс камери: фронтальний, бічний, діагональний;
- щільність трафіку: низька, середня, висока.

Датасет DETRAC (Detection and Tracking Benchmark for Autonomous Vehicles) – публічний набір даних для детекції та трекінгу транспортних засобів, з якого відібрано 10000 зображень. DETRAC містить відеозаписи з камер спостереження за дорожнім рухом у Китаї та забезпечує додаткове різноманіття умов освітлення, типів транспортних засобів та сцен.

Загальний обсяг комбінованого датасету: 12000 зображень.

Анотування виконано у форматі YOLO з використанням інструменту CVAT (Computer Vision Annotation Tool). Для кожного зображення визначено обмежувальні прямокутники транспортних засобів з класом “автомобіль“ (vehicle).

Розподіл даних здійснено у співвідношенні 70:20:10 для тренувальної, валідаційної та тестової вибірок відповідно:

- тренувальна вибірка: 8400 зображень;
- валідаційна вибірка: 2400 зображень;
- тестова вибірка: 1200 зображень.

Характеристики комбінованого датасету наведено в таблиці 3.4.

Таблиця 3.4 – Характеристики датасету для навчання детектора

Параметр	Значення
Загальна кількість зображень	12000
- Власний датасет	2000
- DETRAC	10000
Кількість анотованих об'єктів	~85000
Середня кількість об'єктів на зображенні	7,3
Роздільна здатність	640 * 640
Формат анотацій	YOLO

Набір даних для тестування трекінгу

Для порівняльної оцінки методів трекінгу використовується стандартний бенчмарк MOT17 (Multiple Object Tracking Benchmark 2017). MOT17 містить 14 відеопослідовностей (7 для навчання та 7 для тестування) з різними умовами зйомки: статична та рухома камера, різна щільність об'єктів, різне освітлення.

Для валідації трекера ByteTrack+ використовується підмножина MOT17-val, що дозволяє порівняти результати з опублікованими показниками інших методів (SORT, ByteTrack, OC-SORT) та забезпечує об'єктивність оцінки.

Набір даних для апробації інформаційної технології.

Для апробації інформаційної технології (детекція + трекінг + виявлення аномалій) сформовано набір відеозаписів з камер спостереження за дорожнім рухом з трьох різних перехресть.

Перехрестя 1: Friant Road / Roulette Drive (Fresno, California, USA) [184].

Відеозаписи з камери спостереження за регульованим перехрестям у передмісті Фресно. Камера встановлена під кутом ~45° до напрямку руху, що забезпечує оптимальний огляд стоп-лінії та світлофора. Характеристики:

- роздільна здатність: 1920×1080;
- частота кадрів: 30 FPS;
- тривалість відеозаписів: 90 хвилин;
- умови: різний час доби (день, вечір, ніч), пульсуючий трафік.

Для даного перехрестя виконано калібрування камери з використанням гомографічного перетворення (підрозділ 2.3, рис. 2.9, 2.10).

Перехрестя 2: Jackson Hole Wyoming USA Town Square Live Cam. [185].

Відеозаписи з камери дорожнього руху на міському перехресті. Характеристики:

- роздільна здатність: 1920×1080;
- частота кадрів: 30 FPS;
- тривалість відеозаписів: 60 хвилин;
- умови: інтенсивний трафік, складні умови освітлення.

Перехрестя 3: 4 Corners Camera Downtown City of Coldwater MI [186].

Відеозаписи з камери дорожнього руху на перехресті Jackson Street. Характеристики:

- роздільна здатність: 1280×720;

- частота кадрів: 25 FPS;
- тривалість відеозаписів: 60 хвилин;
- умови: різний час доби (день, вечір), помірна інтенсивність трафіку.

Загальні характеристики датасету для апробації наведено в таблиці 3.5.

Таблиця 3.5 – Характеристики датасету для апробації ІТ

Параметр	Перехрестя 1	Перехрестя 2	Перехрестя 3	Загалом
Тривалість відео, хв	90	60	60	210
Кількість кадрів	162000	90000	108000	360000
Роздільна здатність	1920×1080	1280×720	1920×1080	
Частота кадрів	30 FPS	25 FPS	30 FPS	

Приклади зображень отриманих з камер досліджуваних перехресть наведено на рисунках 3.9-3.11.

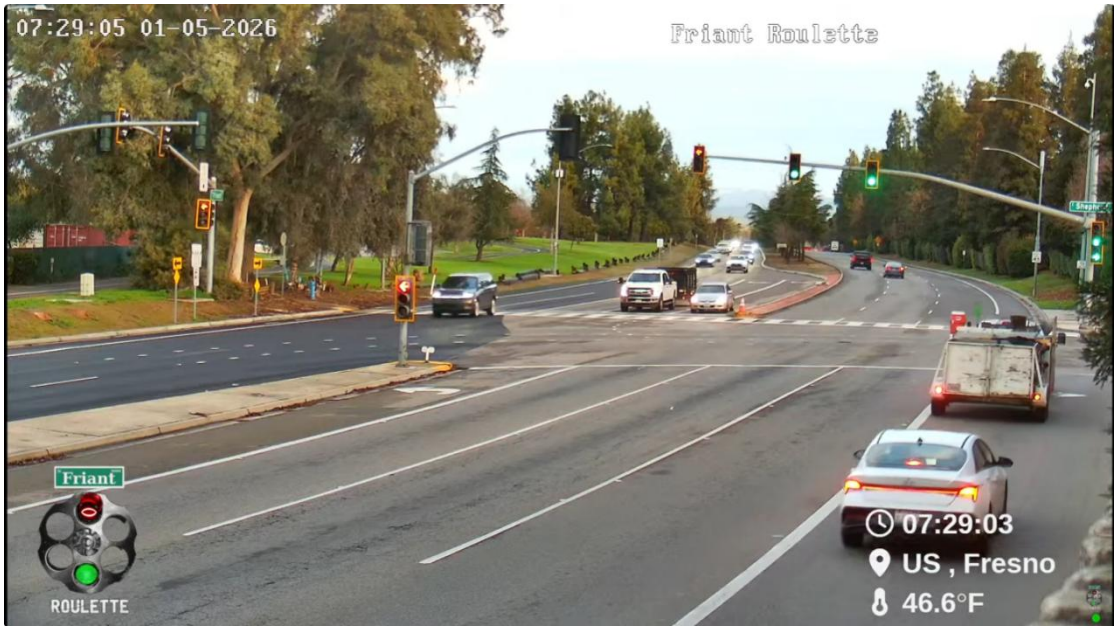


Рис. 3.9. Кадр з камери на перехресті 1

Рис. 3.10. Кадр з камери на перехресті 2

Рис. 3.11. Кадр з камери на перехресті 3

Розмітка аномалій (Ground Truth)

Для кожного відеозапису виконано ручну розмітку аномальних подій. Ground truth формувався за участю двох незалежних анотаторів з подальшим узгодженням спірних випадків.

Для кожної аномалії фіксувалися:

- тип аномалії (RED_LIGHT, SPEEDING, EMERGENCY_BRAKE, ROI_OVERFLOW);

- час початку та закінчення події (номер кадру);

- ідентифікатор транспортного засобу;

- додаткові параметри (швидкість для SPEEDING, тривалість для ROI_OVERFLOW).

Розподіл аномалій за типами наведено в таблиці 3.6.

Таблиця 3.6 – Розподіл аномалій у датасеті для апробації

Тип аномалії	Перехрестя 1	Перехрестя 2	Перехрестя 3	Загалом	%
Проїзд на червоне	60	8	11	79	34
Перевищення швидкості	28	10	8	46	20
Аварійне гальмування	28	5	4	37	16
Тривале перебування в ROI	23	21	24	68	30
Загалом	139	44	47	230	100

Датасет також містить відеозаписи без аномалій (нормальний рух) для формування негативних прикладів та оцінки кількості хибнопозитивних спрацювань.

3.5 Висновки до розділу

У третьому розділі розроблено інтелектуальну інформаційну технологію аналізу відеопотоку дорожнього руху, що інтегрує моделі та методи детекції, трекінгу та виявлення аномалій, запропоновані у розділі 2.

1. Спроектовано загальну структуру інформаційної технології на основі Edge-Cloud архітектури, що забезпечує розподіл обчислювального навантаження між edge-пристроями (детекція, трекінг, виявлення аномалій) та cloud-сервером (агрегація даних, навчання моделей, візуалізація). Ключовою перевагою є виконання повного пайплайну обробки на edge-рівні, що забезпечує автономність роботи при втраті з'єднання з сервером та економію мережевого трафіку на 99,6%.

2. Розроблено UML-діаграми інформаційної технології, що формалізують взаємодію компонентів системи: діаграма послідовності етапу ініціалізації та налаштування системи; діаграма послідовності етапу обробки відеопотоку; діаграма активності навчання моделі детекції.

3. Обґрунтовано вибір програмних засобів та технологічного стеку. Для edge-рівня обрано: Python 3.8+, ONNX Runtime (35 FPS на Jetson Nano), мінімальну збірку OpenCV, MQTT-протокол. Для cloud-рівня: PyTorch, PostgreSQL з TimescaleDB, Flask. Усі компоненти мають відкриті ліцензії.

4. Визначено методику оцінювання ефективності ІТ з метриками для кожного компонента: mAP для детекції; MOTA, IDF1 для трекінгу; MCC для аномалій. Сформовано набори даних: 12000 зображень для детектора, MOT17-val для трекінгу, 3 перехрестя (360000 кадрів, 230 аномалій) для апробації.

Основні результати розділу опубліковані автором у роботах [4, 8].

РОЗДІЛ 4. ПРОГРАМНА ІМПЛЕМЕНТАЦІЯ ТА АПРОБАЦІЯ РОЗРОБЛЕНОЇ ІНТЕЛЕКТУАЛЬНОЇ ТЕХНОЛОГІЇ

4.1 Реалізація комплексу програм для навчання детектора об’єктів

Для використання детектора об'єктів у модулі детекції інформаційної технології потрібно виконати два етапи, а саме:

− провести навчання моделі детектора;

− здійснити оптимізацію моделі детектора для використання на edge-пристроях.

Для виконання цих етапів було використано інтегроване середовище розробки PyCharm від компанії JetBrains.

Навчання модифікованої версії YOLOv8 структура якої наведена у розділі 2 проводилося за допомогою структурованих Python-скриптів, організованих у вигляді окремих модульних програм. Усі експерименти, включаючи модифікацію архітектури (заміна C3 на C3Ghost_CBAM, використання GhostConv, SPPF тощо), налаштування гіперпараметрів, навчання та валідацію, виконувалися безпосередньо в PyCharm.

Використання PyCharm забезпечило наступні переваги при розробці та навчанні детектора на основі YOLO:

− Професійне редагування коду та автодоповнення: інтелектуальна підказка типів, швидка навігація по проєкту та рефакторинг спрощували роботу з великою кодовою базою Ultralytics YOLO.

− Інтегроване налагодження: вбудований дебагер дозволяв покроково відстежувати потік даних через кастомні шари (C3Ghost_CBAM, GhostBottleneck), виявляти помилки в розмірностях тензорів та контролювати поведінку під час forward-проходу.

− Підтримка віртуальних середовищ і залежностей: зручне керування requirements.txt, автоматична активація venv та інтеграція з pip забезпечували стабільність експериментів.

− Інтеграція з системою контролю версій: вбудована підтримка Git дозволяла відстежувати зміни в YAML-конфігураціях моделей, скриптах навчання та результатах експериментів.

− Локальний та віддалений запуск: PyCharm давав можливість запускати навчання як на локальній машині з GPU Nvidia, так і через SSH на віддалених серверах, що забезпечувало гнучкість у використанні обчислювальних ресурсів.

− Візуалізація результатів у реальному часі: завдяки інтеграції з Weights & Biases, Matplotlib та TensorBoard, графіки втрат, метрик mAP, приклади детекції, а також динаміка оновлення ваг і зсувів відображалися безпосередньо в IDE під час навчання.

Особливо важливим було використання платформи Weights & Biases для:

− ітеративного відстеження експериментів: кожна ітерація навчання фіксувалася з повним набором гіперпараметрів, метрик та конфігурацій;

− візуалізації градієнтів і розподілу ваг: аналіз гістограм параметрів дозволяв виявляти проблеми з навчанням (наприклад, vanishing gradients);

− порівняння моделей: зручне порівняння різних архітектур (YOLOv8n, YOLOv8-ghost, з/без CBAM) за метриками mAP@0.5:0.95, швидкістю та розміром;

− відтворюваності: автоматичне збереження чекпоінтів, логів та артефактів забезпечувало повну відтворюваність результатів.

Незважаючи на те, що реалізація була проведена в PyCharm, код легко портується в інші середовища, такі як:

− AWS SageMaker – для масштабованого хмарного навчання;

− Google Colab – для швидкого прототипування з безкоштовним GPU;

− Kaggle – для роботи з публічними датасетами та змаганнями;

− Microsoft Azure ML – для інтеграції з хмарними рішеннями.

Для реалізації комплексу програм була використана мова програмування Python. Вибір Python обґрунтовано його простотою, інтерпретованою природою, динамічною типізацією, кросплатформністю, розвиненою стандартною бібліотекою та потужною екосистемою для машинного навчання (PyTorch, Ultralytics, OpenCV, NumPy, Matplotlib). Особливо важливу роль відіграла бібліотека Ultralytics YOLOv8, яка забезпечує зручний API для навчання, валідації, експорту моделей та інференсу, що дозволило швидко інтегрувати кастомні шари (C3Ghost_CBAM, GhostConv) у стандартний пайплайн YOLO.

Таким чином, використання PyCharm як основного середовища розробки забезпечило професійний, контрольований та відтворюваний процес створення, модифікації та навчання детектора об’єктів. На основі схеми алгоритму детектора об'єктів з розділу 2 був сформований yaml файл зі структурою нейронної мережі (табл. 4.1).

Таблиця 4.1 – Структура запропонованої нейронної мережі

Шар	Тип	Канали	Особливості
0-2	Conv + C2f	64 → 128	P1 → P2
3-4	Conv + C2f	256	P3/8
5-6	Conv + C2f	512	P4/16
7-9	Conv + C3Ghost + SPPF	1024	P5/32
10-12	Upsample + Concat + C2f	512	FPN: P5 → P4
13-15	Upsample + Concat + C2f	256	FPN: P4 → P3
16-18	GhostConv + Concat + C3Ghost_CBAM	512	PAN: P3 → P4
19-21	GhostConv + Concat + C3Ghost_CBAM	1024	PAN: P4 → P5
Detect	Detect	nc=1	Використовує P3, P4, P5

Для реалізації запропонованих блоків були розроблені класи CBAM та C3Ghost_CBAM. UML-діаграма класів з залежними зв'язками наведена на рис. 4.1.

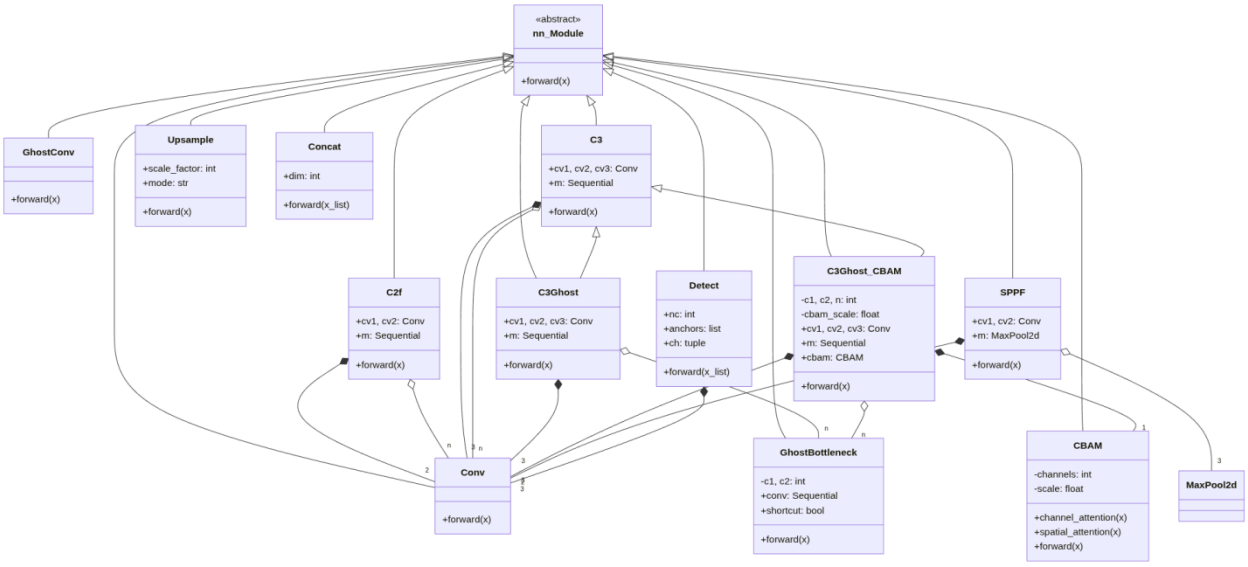


Рис. 4.1. UML-діаграма класів модулю детектора

Програмна реалізація класу CBAM представлена на рисунку 4.2.

```
3016    class CBAM(nn.Module):  Roman
3017        """Convolutional Block Attention Module (CBAM)"""
3018        def __init__(self, channels, kernel_size=7, scale=1.0):  Roman
3019            super().__init__()
3020            self.channel_attention = nn.Sequential(
3021                nn.AdaptiveAvgPool2d(1),
3022                nn.Conv2d(channels, channels, 1, bias=True),
3023                nn.Sigmoid()
3024            )
3025            self.spatial_attention = nn.Sequential(
3026                nn.Conv2d(2, 1, kernel_size, padding=kernel_size // 2, bias=False),
3027                nn.Sigmoid()
3028            )
3029            self.scale = scale
3030        def forward(self, x):  Roman
3031            ca = self.channel_attention(x)  # Канальна увага
3032            sa = self.spatial_attention(torch.cat([...], dim=1))  # Просторова увага
3034            return x * (1 + self.scale * ca) * (1 + self.scale * sa)  # Пом'якшена версія CBAM
```

Рис. 4.2. Програмна реалізація класу CBAM

Клас CBAM, що наслідує torch.nn.Module реалізує механізм уваги Convolutional Block Attention Module, який покращує представлення тензора шляхом фокусування на важливих каналах і просторових регіонах. Містить такі методи:

- __init__(self, channels, kernel_size=7, scale=1.0) створює два підмодулі: канальну увагу через послідовність глобального середнього пулингу, 1×1 згортки та сигмоїди, і просторову увагу через конкатенацію середнього та максимального значення по каналах, згортку з ядром розміру kernel_size (з відповідним паддінгом) та сигмоїду. Зберігає параметр scale для регулювання сили уваги.
- forward(self, x) спочатку обчислює канальну увагу, застосовуючи глобальне середнє по просторовим вимірам, потім формує вхід для просторової уваги шляхом конкатенації середнього та максимального значення тензора по каналах, пропускає його через просторову гілку, і нарешті повертає зважений вхід за формулою □ ∗ (1 + □□□□□ ∗ □□) ∗ (1 + □□□□□ ∗ □□), де ca – канальна увага, sa – просторова. Такий підхід не блокує, а м’яко підсилює важливі ознаки.

Програмна реалізація класу C3Ghost_CBAM наведена на рисунку 4.3.

```
class C3Ghost_CBAM(C3):
    """C3 module with GhostBottleneck() + CBAM (правильний розмір каналів)."""

    def __init__(self, c1, c2, n=1, shortcut=True, g=1, e=0.5, cbam_scale=1.0):
        super().__init__(c1, c2, n, shortcut, g, e)
        c_ = int(c2 * e)  # hidden channels
        self.m = nn.Sequential(*(GhostBottleneck(c_, c_, 3, 1) for _ in range(n)))
        self.cbam = CBAM(c2, scale=cbam_scale)  # ✅ Додаємо CBAM на вихідний тензор
        # self.cbam = LSKA(c2, c2)

    def forward(self, x):
        """Forward pass through the CSP bottleneck with 2 convolutions."""
        # return self.cbam(self.cv3(torch.cat((self.m(self.cv1(x)), self.cv2(x)), 1)))
        return self.cv3(self.cbam(torch.cat((self.m(self.cv1(x)), self.cv2(x)), 1)))
```

Рис. 4.3. Програмна реалізація класу C3Ghost_CBAM

Структурна схема класу C3Ghost_CBAM представлена на рисунку 4.4.

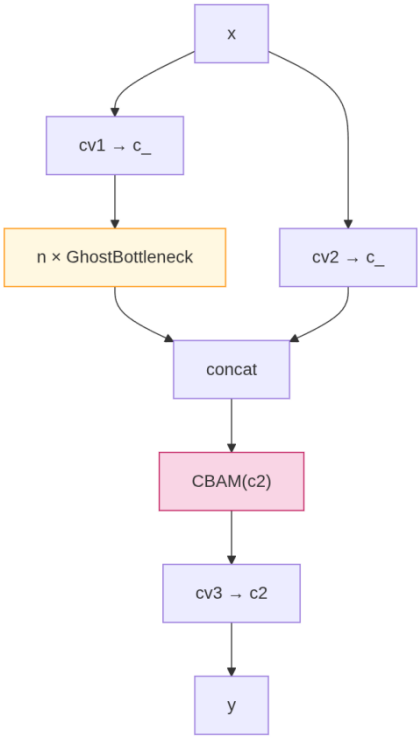


Рис. 4.4. Структурна схема класу C3Ghost_CBAM

Клас C3Ghost_CBAM, що успадковується від C3 є модифікованою версією CSP-блоку, де стандартні Bottleneck замінено на легкі GhostBottleneck, а після конкатенації додано CBAM для покращення представлення. Містить такі методи:

- __init__(self, c1, c2, n=1, shortcut=True, g=1, e=0.5, cbam_scale=1.0) викликає ініціалізацію базового класу C3, обчислює приховану кількість каналів c_ = int(c2 * e), створює послідовність із n екземплярів GhostBottleneck з ядром 3×3 і кроком 1, та додає модуль CBAM з вихідною кількістю каналів c2 і заданим cbam_scale.
- forward(self, x) виконує стандартний CSP-потік: розбиває вхід через cv1 і cv2, пропускає одну гілку через послідовність GhostBottleneck, об’єднує виходи конкатенацією по каналах, застосовує CBAM до об’єднаного тензора, і нарешті пропускає результат через cv3. Повертає тензор розміру [B, c2, H, W].

Таким чином, C3Ghost_CBAM поєднує економію параметрів за рахунок Ghost-операцій, стабільність CSP-структури та підвищення якості за рахунок уваги після повного контексту блоку.

Після завершення навчання модифікованої моделі детектора проводилася її оптимізація для розгортання на платформі Jetson Orin Nano, з фокусом на максимальну продуктивність GPU та мінімальну затримку інференсу в умовах обробки відеопотоку в реальному часі.

Оптимізація виконувалася в PyCharm за допомогою Ultralytics YOLOv8 та TensorRT з підтримкою FP16-квантування. Основним цільовим форматом було обрано TensorRT Engine (.engine), що забезпечує максимальну продуктивність завдяки апаратним оптимізаціям Tensor Core архітектури Ampere, вбудованої у Jetson Orin Nano.

Процес складався з таких етапів:

Експорт моделі в проміжний формат. Навчена модель у форматі .pt (PyTorch) експортувалася в ONNX за допомогою вбудованого API Ultralytics. Цей крок є необхідним для подальшої компіляції засобами TensorRT.

Компіляція TensorRT Engine з FP16-квантуванням:

а) використовувався TensorRT Builder з режимом FP16 (half-precision), що вдвічі зменшує обсяг обчислень порівняно з FP32 без суттєвої втрати точності детекції;

б) вхідні дані: нормалізовані зображення у форматі NCHW [1, 3, 640, 640] з діапазоном значень [0, 1] у форматі FP16;

в) усі ваги та проміжні активації конвертовані у формат FP16 з автоматичним визначенням оптимальних масштабів засобами TensorRT;

г) виконувалася оптимізація графа обчислень: злиття шарів (layer fusion), усунення надлишкових операцій та адаптація під архітектуру GPU Jetson Orin Nano;

д) для підвищення точності оптимізації використовувався репрезентативний калібрувальний датасет (100–200 зображень з тренувальної вибірки).

Результат – скомпільований файл .engine, специфічний для апаратної платформи Jetson Orin Nano, розмір якого зменшився приблизно вдвічі порівняно з FP32 ONNX-версією. Швидкість інференсу після компіляції становить більше 60 FPS, що забезпечує обробку відеопотоку без пропуску кадрів.

Результати навчання детектора об'єктів на повному датасеті описаному у підрозділі 3.4 наведені у табл. 4.2.

Таблиця 4.2 – Результати навчання детектора об'єктів на повному датасеті

Метод/Критерії	mAP@(0,50-0,95)	mAP@0,50	Кількість параметрів, млн	GFLOPs
yolov8-nano	**71,5**	**91,5**	3,0	8,1
Запропонований	70,6	90,8	**2,3**	**7,3**

На основі табл. 2.2 (об'єднаний датасет Detrac + власний) та табл. 4.2 (власний датасет 2000 зображень зі складними умовами) проведено порівняння ефективності запропонованого методу.

Запропонована конфігурація (YOLOv8-nano + C3Ghost + GhostConv + CBAM) демонструє незначне зниження якості порівняно з базовою моделлю: mAP@(0,50-0,95) зменшується з 0,715 до 0,706 (різниця -1,3%). Однак цей результат досягається при суттєвому скороченні обчислювальної складності: кількість параметрів зменшена на 23,3% (з 3,0 до 2,3 млн), а GFLOPs зменшені на 9,9% (з 8,1 до 7,3).

На власному датасеті зі складними умовами (низька якість, різний час доби, погані погодні умови) запропонований метод показує: mAP@(0,50-0,95) знижується з 0,531 до 0,525 (різниця -1,1%), тоді як скорочення

обчислювальних ресурсів залишається тим же (23,3% параметрів, 9,9% GFLOPs).

Запропонований метод демонструє послідовну поведінку на обох датасетах, що свідчить про його робастність. Зниження якості становить близько 1% незалежно від складності датасету.

Власний датасет характеризується нижчими абсолютними значеннями mAP (0,531 проти 0,715), що на 25,7% менше порівняно з об'єднаним датасетом. Це логічно пояснюється тим, що складні умови зйомки (ніч, дощ, туман, низька якість) суттєво утруднюють виявлення об'єктів. На об'єднаному датасеті переважають зображення з простішими умовами (10000 зображень з Detrac з добрими умовами освітлення та якості), що призводить до вищих показників якості детекції.

Запропонований метод забезпечує баланс між якістю виявлення та обчислювальною ефективністю. Зниження якості на 1–1,3% є прийнятною ціною за скорочення обчислювального навантаження на ~10%, що важливим для розгортання моделей на пристроях з обмеженими ресурсами (edge-пристрої, мобільні девайси). Послідовність результатів на датасетах з різними характеристиками вказує на хорошу узагальнюючу здатність запропонованого підходу.

Результати тестування отриманих моделей на валідаційному датасеті за співдношенням F1/Confidence наведені на рис.4.5.

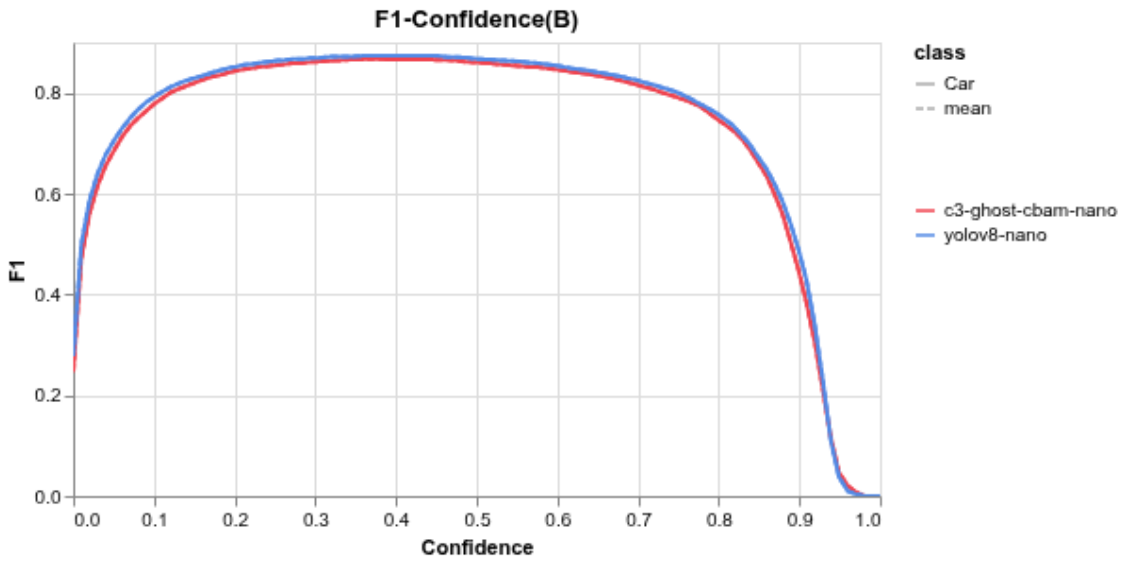


Рис. 4.5 – Результати тестування детекторів об'єктів на повному датасеті

4.2 Розробка комплексу програм модулів інформаційної технології

4.2.1 Модуль детекції

Для інференсу моделей детекції на платформі Jetson Orin Nano реалізовано клас YOLOv8TensorRT, що використовує скомпільований TensorRT Engine (.engine) для досягнення максимальної продуктивності на GPU. Основними функціями класу є:

– __init__(engine_path, conf_thresh=0.25, iou_thresh=0.45, input_size=(640, 640)) – завантаження скомпільованого TensorRT Engine через tensorrt.Runtime та deserialize_cuda_engine(). Створюється контекст виконання IExecutionContext, виділяється пам'ять на GPU через cuda.mem_alloc() для вхідних та вихідних буферів. Встановлюються пороги впевненості та IoU для NMS;

– preprocess(image) – підготовка зображення до інференсу: (1) масштабування з збереженням aspect ratio та letterbox padding (заповнення сірим кольором 114 пікселів); (2) нормалізація на [0, 1] з конвертацією у FP16; (3) транспонування у формат NCHW [1, 3, 640, 640]; (4) асинхронне копіювання у GPU-пам'ять через cuda.memcpy_htod_async();

– detect(image) – основний метод інференсу: (1) препроцесування з розміщенням даних у GPU-буфері; (2) запуск інференсу через context.execute_async_v2() з прив'язкою буферів; (3) синхронізація CUDA-потоку через stream.synchronize(); (4) асинхронне копіювання результатів з GPU у CPU-пам'ять; (5) постпроцесування;

– postprocess(outputs, scale, pad) – постпроцесування вихідних тензорів: (1) перевірка формату [1, 8400, 84] або [1, 84, 8400], трансформація при необхідності; (2) екстракція bounding box [x, y, w, h] та впевненості класів; (3) вибір класу з максимальною впевненістю; (4) фільтрація за conf_thresh; (5) конвертація з xywh на xyxy; (6) адаптація координат до оригінального розміру кадру; (7) застосування Non-Maximum Suppression (NMS);

– nms(detections) – реалізація жадної NMS: (1) сортування детекцій за впевненістю; (2) ітеративне видалення детекцій з IoU > iou_thresh до остаточного набору; (3) відбір детекцій тільки якщо вони мають різні класи об'єктів;

– release() – звільнення GPU-ресурсів: видалення CUDA-контексту, буферів та Engine для коректного завершення роботи модуля.

Результат detect() повертає масив формату [[x1, y1, x2, y2, score, class_id], ...] у піксельних координатах оригінального кадру, що може безпосередньо передаватися у модуль трекінгу. Завдяки асинхронному виконанню на GPU та FP16-оптимізації TensorRT, затримка одного інференсу становить ~22 мс, що відповідає продуктивності 45 FPS на платформі Jetson Orin Nano.

4.2.2 Програмна реалізація трекеру об'єктів із підтримкою edge-інференсу

Багатооб'єктний трекінг (Multi-Object Tracking, MOT) організовано як послідовна асоціація детекцій між послідовними кадрами відеопослідовності для збереження тотожності об'єктів. На кожному кадрі система виконує три основні операції: (1) прогнозування позицій існуючих трекінг-об'єктів на основі їхніх попередніх траєкторій за допомогою розширеного Калманівського фільтру; (2) асоціація вновиявлених детекцій з існуючими треками через матрицю схожості на основі IoU та опціональних механізмів подавлення; (3) управління життєвим циклом трекінг-об'єктів (створення нових трекінг-об'єктів, позначення як втраченого, видалення давно втрачених).

Запропонований у розділі 2.2 удосконалений варіант базового ByteTrack алгоритму, позначимо його як ByteTrack+, має три опційні компонентами для підвищення якості трекінгу:

1. IoU-suppression – технологія для уникнення помилкових ініціалізацій нових треків шляхом перевірки перекриття з активними треками на рівні□□□ < *0,65*. При порушенні умови новий трек не активується, що зменшує фрагментацію траєкторій;

2. Occlusion grace period – механізм для збереження втрачених треків протягом періоду очікування (25 кадрів) при тимчасовій оклюзії, що підвищує MOTA метрику при обробці сцен з частою оклюзією;

3. DIoU-matching – альтернативна метрика асоціації, що враховує не тільки перекриття (IoU), але й центральну відстань між bounding box, забезпечуючи кращу розрізнення близьких об'єктів.

Враховуючи розроблені в третьому розділі загальну структуру інформаційної технології та UML-діаграми послідовностей було розроблено програмне забезпечення модулю трекінгу. Воно представлене у вигляді UML-діаграми класів на рис. 4.6.

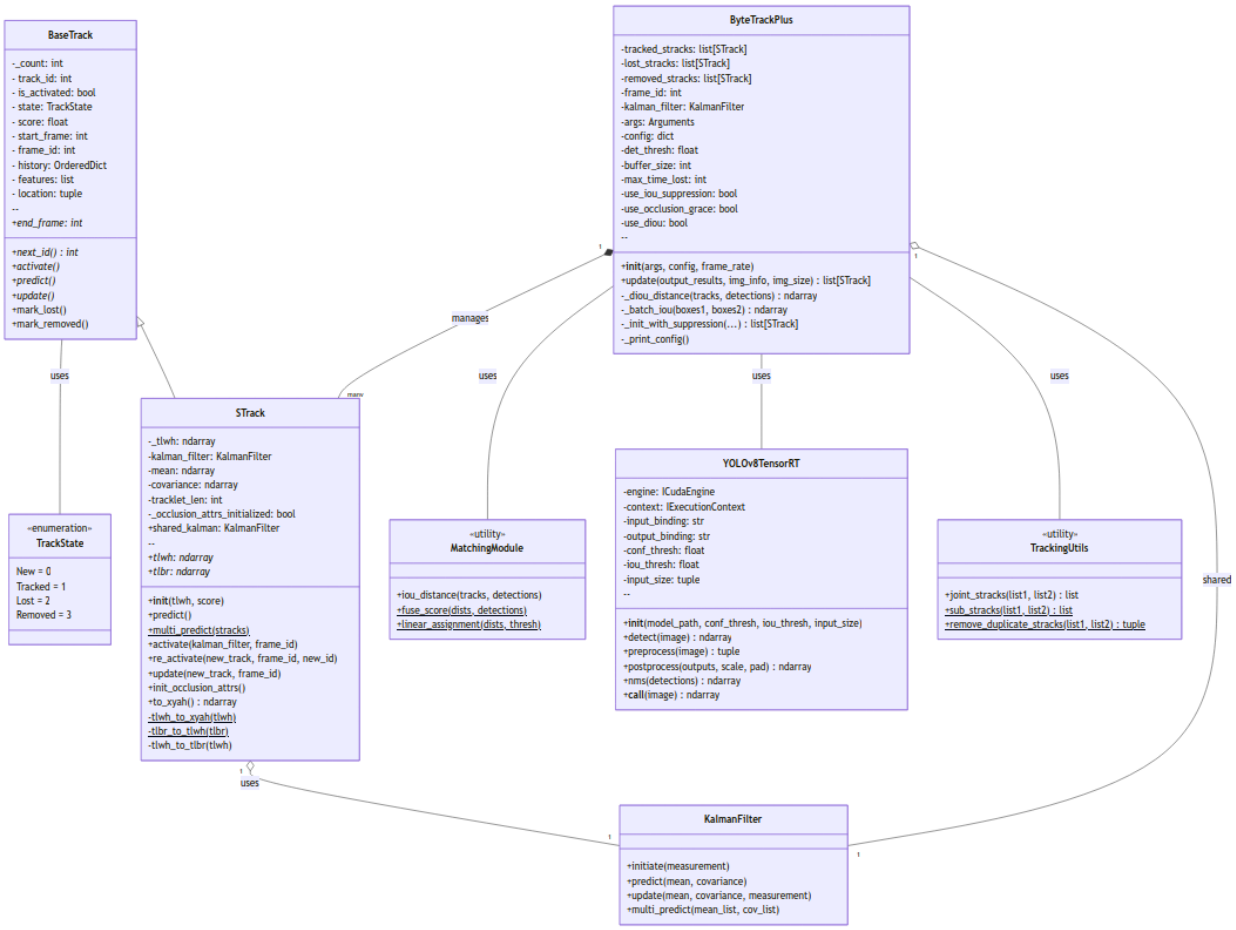


Рис. 4.6. UML-діаграма класів модулю трекінга

Клас BaseTrack служить абстрактною основою для всіх трекінг-об'єктів у системі та визначає спільний інтерфейс і стан. Основні атрибути класу:

– TrackState – перелічення (enum) для представлення стану трека з чотирма значеннями: New (новостворений трек, ще не активований), Tracked (активний трек з послідовними детекціями), Lost (втрачений трек без детекцій протягом останніх кадрів), Removed (видалений трек, що більше не розглядається);

– track_id – унікальний числовий ідентифікатор, присвоюється методом next_id() при активації трека та залишається постійним протягом всього його життєвого циклу;

Методи базового класу є абстрактними та повинні бути реалізовані у похідних класах:

– activate(...) – ініціалізація нового трека з розподілом унікального ID та початковим станом;

– predict() – прогнозування позиції трека на наступному кадрі;

– update(...) – оновлення трека новою детекцією;

– mark_lost() та mark_removed() – переведення трека у стан Lost або Removed.

Клас STrack розширює BaseTrack та реалізує конкретний функціонал для трекінгу одного об'єкту у межах однієї камери. Основними методами є:

– __init__(tlwh, score) – конструктор приймає координати обмежуючого прямокутника у форматі [top, left, width, height] та впевненість детекції. Ініціалізуються: массив _tlwh для зберігання координат, об'єкт Калманівського фільтру (лінива ініціалізація), логічний прапор is_activated, tracklet_len для лічення послідовних детекцій, та спеціальний прапор _occlusion_attrs_initialized для лінивої ініціалізації атрибутів при оклюзії;

– predict() – виконує прогнозування стану через Kalman filter. При не-Tracked стані компонент швидкості (mean[7]) встановлюється на нуль для уникнення завищених прогнозів при оклюзії;

– activate(kalman_filter, frame_id) – активація новоствореного трека, розподіл унікального ID, ініціалізація Калманівського фільтру за допомогою методу initiate() з initial observation у форматі xyah (x_center, y_center, aspect_ratio, height);

– re_activate(new_track, frame_id, new_id=False) – переактивація втраченого трека із новою детекцією. При new_id=False зберігається старий track_id, що дозволяє розглядати його як продовження попередньої траєкторії;

– update(new_track, frame_id) – оновлення активного трека новою детекцією, включаючи: інкремент tracklet_len, оновлення Калманівського фільтру методом update(), встановлення стану на Tracked та активацію трека;

Клас ByteTrackPlus управляє станом всіх активних, втрачених та видалених треків на відеопослідовності. Конструктор приймає аргументи та конфігурацію:

– args – об'єкт ArgumentParser з параметрами: track_thresh (поріг впевненості для high-confidence детекцій, типово 0,5), track_buffer (максимальний час для утримання втрачених треків в кадрах, типово 30), match_thresh (поріг для угорського алгоритму асоціації, типово 0,8), mot20 (прапор для набору даних MOT20 з низькою якістю детекцій);

– config – словник конфігурації з опціональними компонентами: use_iou_suppression (bool), init_iou_suppress (float, 0,65), use_occlusion_grace (bool), grace_period (int, 25 кадрів), use_diou (bool);

– frame_rate – частота кадрів відеопослідовності для масштабування track_buffer;

Основні атрибути стану:

– tracked_stracks – список активних треків, що мають послідовні детекції;

– lost_stracks – список втрачених треків, що можуть бути переактивовані;

– removed_stracks – список видалених треків;

– frame_id – глобальний лічильник кадрів для синхронізації.

Основний метод update(output_results, img_info, img_size) виконує трекінг за один кадр:

1. Препроцесування детекцій:

а) вихід детектора YOLOv8 (або TensorRT) містить bounding box [x1, y1, x2, y2] та впевненість;

б) масштабування координат від розміру моделі (640×640) до оригінального розміру кадру через factor scale = min(img_size[0] / img_h, img_size[1] / img_w);

в) розділення на high-confidence (score > track_thresh) та low-confidence (0.1 < score < track_thresh) для диференційованої обробки;

2. Прогнозування:

а) Об'єднання активних (tracked_stracks) та втрачених (lost_stracks) треків у пул;

б) Векторизовані прогнозування всіх треків за допомогою STrack.multi_predict();

в) Масив mean містить стан [x, y, a, h, vx, vy, va, vh] у форматі Kalman filter.

3. Двостадійна асоціація детекцій з треками:

Стадія 1 – High-confidence детекції:

а) Обчислення матриці відстаней: якщо use_diou, застосовується метод _diou_distance(), інакше – matching.iou_distance();

б) Матриця заповнюється значеннями від 0 (ідеальне перекриття) до ∞ (жодного перекриття);

в) Угорський алгоритм (linear_assignment з порогом match_thresh) знаходить оптимальну біпартитну відповідність;

г) Спарені пари (itracked, idet) оновлюють активні треки методом update();

д) Неспарені треки переводяться в стан Lost, неспарені детекції переходять на стадію 2;

Стадія 2 – Low-confidence детекції:

а) Асоціація залишившихся неспарених активних треків з low-confidence детекціями;

б) Поріг асоціації знижується до 0,5 для більшої толерантності до низькоточних детекцій;

в) Спарені треки оновлюються, що дозволяє відстежити об'єкти навіть при тимчасовій зниженій впевненості;

4. Опціональна обробка оклюзії:

а) Якщо use_occlusion_grace увімкнено, для неспарених активних треків ініціалізуються атрибути оклюзії;

б) При наслідуючому виклику update() треки, позначені як недавно окклюовані, дозволяються залишатись у lost_stracks протягом grace_period кадрів;

5. Управління новими треками:

а) Якщо use_iou_suppression увімкнено, метод _init_with_suppression() фільтрує нові детекції: новий трек активується лише якщо його IoU з активними треками < init_iou_suppress (0,65), що уникає дублювання;

б) Інакше всі неспарені high-confidence детекції (score > det_thresh) активуються як нові треки;

6. Видалення застарілих треків:

а) Для кожного трека в lost_stracks перевіряється: frame_id - track.end_frame > max_time_lost;

б) Якщо умова виконана, трек переводиться в Removed стан та видаляється;

в) max_time_lost = buffer_size, масштабований від частоти кадрів;

7. Агрегація результатів:

а) Об'єднання активних (tracked), переактивованих (refind) та новостворених (new_tracks) треків;

б) Видалення дублікатів через remove_duplicate_stracks(), що порівнює IoU між Tracked і Lost список та залишає лише найдовші траєкторії;

в) Повернення масиву output_stracks, що містить лише активовані треки (is_activated=True).

Допоміжні методи:

– _diou_distance(tracks, detections) – обчислення матриці DIoU (Distance IoU) для всіх пар трек-детекція. Формула: DIoU = IoU - (d_center / d_diag), де d_center – відстань між центрами, d_diag – діагональ обмежуючого прямокутника. Такий підхід краще розрізнює близькі об'єкти та змушує пристосуватись до обставин;

– _batch_iou(boxes1, boxes2) – векторизована обчислення IoU для всіх пар bounding box, результат має форму (len(boxes1), len(boxes2));

– _init_with_suppression(detections, u_detection, active_tracks) – фільтрація нових трекінг-об'єктів за IoU з активними: новий трек активується лише якщо max_iou_to_active < init_iou_suppress.

Модуль реалізує три утилітарні функції для роботи зі списками STrack:

– joint_stracks(tlista, tlistb) – об'єднання двох списків треків з усуненням дублікатів за track_id. Результат: список унікальних треків на основі першого появлення;

– sub_stracks(tlista, tlistb) – видалення всіх треків з tlista, які знаходяться в tlistb (на основі track_id). Результат: different set;

– remove_duplicate_stracks(stracksa, stracksb) – усунення дублікатів між двома списками на основі IoU < 0,15. При виявленні дублікату залишається трек з більшою тривалістю (frame_id - start_frame).

Ці функції використовуються на кожному кадрі для оптимізації пам'яті та попередження ID switches.

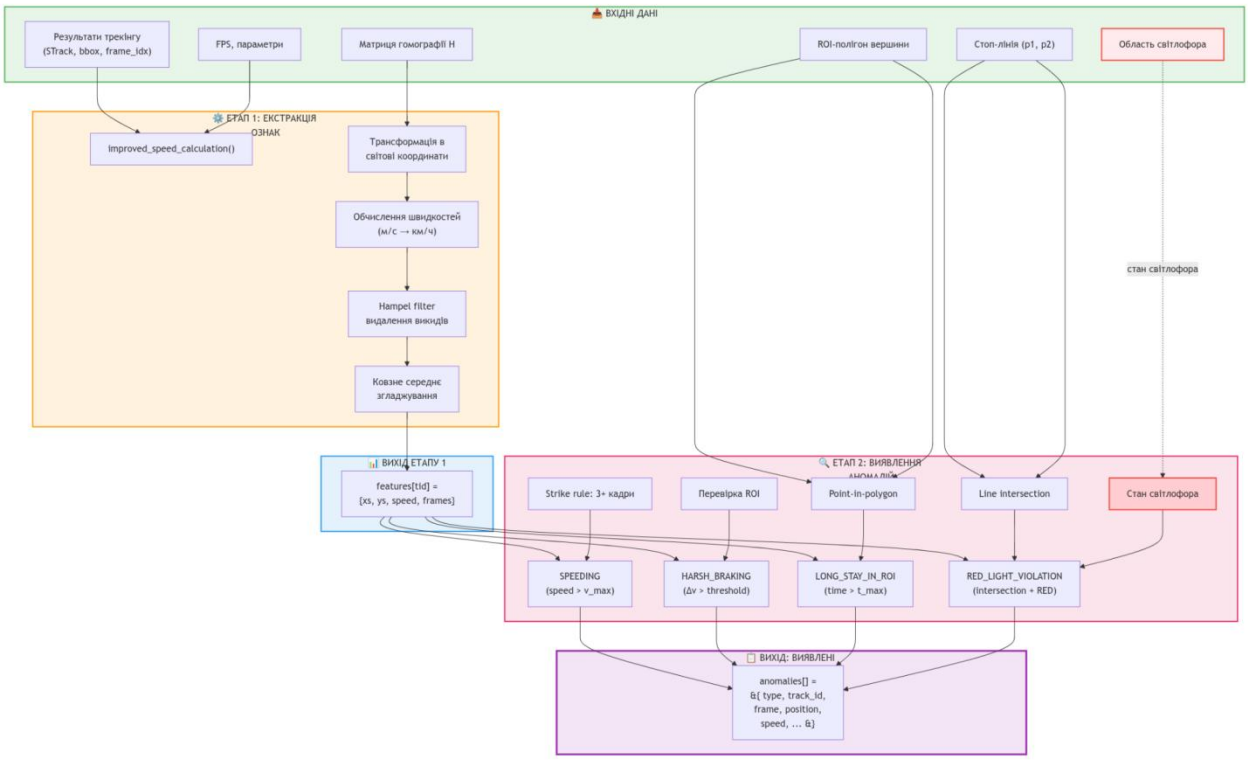


Рис. 4.7. Архітектура модуля виявлення аномалій

4.3 Експериментальна апробація інтелектуальної інформаційної технології

Для підтвердження ефективності розробленої інформаційної технології аналізу відеопотоку дорожнього руху проведено комплексне експериментальне дослідження, що охоплює всі етапи обробки: детекцію транспортних засобів, багатооб'єктний трекінг, виявлення аномалій та оцінку продуктивності системи на Edge-пристрої. Експерименти проводилися відповідно до методики, викладеної в підрозділі 3.4, з використанням наборів даних та апаратного забезпечення, описаних у підрозділах 3.4.2, 4.1 та 4.2 відповідно.

4.3.1 Експериментальне середовище та набори даних

Експериментальні дослідження проводилися на двох апаратних платформах відповідно до Edge-Cloud архітектури розробленої інформаційної технології.

Edge-платформа (для апробації повного пайплайну обробки в реальному часі):

- пристрій: NVIDIA Jetson Orin Nano Developer Kit;
- процесор: 6-ядерний Arm Cortex-A78AE @ 1.5 GHz;
- графічний процесор: 1024-core NVIDIA Ampere з 32 тензорними ядрами;
- оперативна пам'ять: 8 GB LPDDR5;
- накопичувач: microSD 64 GB / NVMe SSD;
- продуктивність AI: до 40 TOPS.

Cloud-платформа (для навчання моделей та порівняльних експериментів):

- процесор: Intel Xeon W-2245 @ 3.9 GHz (8 cores / 16 threads);
- графічний процесор: NVIDIA GeForce RTX 4090 (16384 CUDA cores, 24 GB VRAM);
- оперативна пам'ять: 64 GB DDR4-3200;
- накопичувач: Samsung 980 PRO NVMe SSD 1 TB.

Для апробації інформаційної технології було сформовано власний датасет, сформований з камер відеоспостереження трьох перехресть [184-186]. В загальному датасет містить 210 хвилин відеозаписів, у яких виділено 230 аномальних подій та 450 нормальних подій. Більш детальна інформація про датасет наведена у підрозділі 3.4.

Розмітка ground truth виконувалася вручну двома незалежними анотаторами з подальшим узгодженням спірних випадків. Для кожної аномалії фіксувалися: тип аномалії, час початку та закінчення події, ідентифікатор

транспортного засобу, додаткові параметри (швидкість для перевищення, тривалість для ROI).

Додатково для порівняльних експериментів з трекінгу використовувався стандартний бенчмарк MOT17-val, а для детекції – власний датасет з 2000 зображень, описаний у підрозділі 2.1.

Для об'єктивної оцінки ефективності запропонованих удосконалень визначено базову конфігурацію системи:

- детектор: YOLOv8-nano (стандартна архітектура без модифікацій);
- трекер: SORT (Simple Online and Realtime Tracking);
- пороги детекції: conf_threshold = 0.25, iou_threshold = 0.45;
- параметри трекінгу: max_age = 30, min_hits = 3.

Запропонована конфігурація включає:

- детектор: YOLOv8-nano + C3Ghost-CBAM (модифікована архітектура з підрозділу 2.1);
- трекер: ByteTrack+ (удосконалений метод з підрозділу 2.2);
- ті самі пороги детекції та трекінгу для забезпечення порівнянності.

4.3.2 Результати апробації за сценаріями виявлення аномалій

Апробація інформаційної технології проводилася за чотирма сценаріями, що відповідають типам аномалій, визначеним у підрозділі 2.3.

Сценарій 1: Виявлення проїзду на червоний сигнал світлофора.

Умови експерименту. Тестування проводилося на 529 відеозаписах, що містять 79 випадків проїзду на червоний сигнал. Для кожного перехрестя попередньо виконано калібрування камери та визначено: координати стоп-лінії, зону розташування світлофора для HSV-аналізу, параметри гомографічного перетворення. Матрицю помилок для виявлення проїзду на червоний сигнал наведено в таблиці 4.3.

Таблиця 4.3 – Матриця помилок для сценарію “проїзд на червоний сигнал“

	Прогнозоване позитивне	Прогнозоване негативне
Актуальне позитивне	TP = 74	FN = 5
Актуальне негативне	FP = 15	TN = 435

На основі матриці помилок обчислено метрики якості:

- MCC = 0.861;
- Precision = 0.832;
- Recall = 0.937;
- F1-score = 0.881.

Аналіз помилок. Хибнонегативні випадки (FN = 5) переважно зумовлені: неправильним визначенням стану світлофора через засвічення або відблиски (2 випадки); втратою треку через оклюзію іншим транспортним засобом (3 випадки). Хибнопозитивні випадки (FP = 15) виникали через: помилкове визначення червоного сигналу при жовтому (10 випадків); мерехтіння обмежувальної рамки при зупинці (5 випадків).

Сценарій 2: Виявлення перевищення швидкості.

Умови експерименту. Тестування проводилося на 496 відеозаписах з прямолінійними ділянками доріг, що містять 46 випадків перевищення швидкості. Максимальна дозволена швидкість v_{max} = 100 км/год. Матрицю помилок для виявлення перевищення швидкості наведено в таблиці 4.4.

Таблиця 4.4 – Матриця помилок для сценарію “перевищення швидкості“

	Прогнозоване позитивне	Прогнозоване негативне
Актуальне позитивне	TP = 61	FN = 7
Актуальне негативне	FP = 26	TN = 424

Метрики якості:

- MCC = 0.758;
- Precision = 0.701;
- Recall = 0.897;
- F1-score = 0.787.

Аналіз помилок. Нижча точність порівняно зі сценарієм 1 зумовлена чутливістю оцінки швидкості до: якості калібрування камери та точності гомографічного перетворення; стабільності обмежувальних рамок при русі об'єкта; частоти кадрів та інтервалу обчислення швидкості. Основні джерела помилок: коливання розміру обмежувальної рамки box через оклюзії (7 випадки FN); похибка калібрування на краях зони спостереження (26 випадків FP).

Сценарій 3: Виявлення аварійного гальмування.

Умови експерименту. Аварійне гальмування визначалося за критерієм різкого зменшення швидкості: $\Delta v > 15$ км/год за 1 секунду при початковій швидкості $v > 10$ км/год. Датасет містить 37 випадків аварійного гальмування. Результати виявлення аномалій занесено у матрицю помилок, яка наведена в таблиці 4.5.

Таблиця 4.5 – Матриця помилок для сценарію “аварійне гальмування“

	Прогнозоване позитивне	Прогнозоване негативне
Актуальне позитивне	TP = 28	FN = 9
Актуальне негативне	FP = 39	TN = 411

Метрики якості:

- MCC = 0.515;
- Precision = 0.418

- Recall = 0.757;

- F1-score = 0.538.

Аналіз помилок. Найнижча точність серед усіх сценаріїв зумовлена: високою чутливістю до шуму в оцінці швидкості та прискорення; короткою тривалістю події (1-2 секунди); складністю розмежування нормального та аварійного гальмування на межі порогу. Можливими покращеннями можуть бути: використання згладжування швидкості (moving average); врахування контексту (відстань до попереднього автомобіля); інтеграцію з даними про стан гальмівних вогнів.

Сценарій 4: Виявлення тривалого перебування в ROI.

Умови експерименту. ROI визначено як зону перехрестя. Аномалією вважається перебування транспортного засобу в ROI довше t_{max} = 5 секунд (без зупинки на світлофорі). Датасет містить 46 таких випадків.

Матрицю помилок наведено в таблиці 4.6.

Таблиця 4.6 – Матриця помилок для сценарію “тривале перебування в ROI”

	Прогнозоване позитивне	Прогнозоване негативне
Актуальне позитивне	TP = 42	FN = 4
Актуальне негативне	FP = 11	TN = 439

Метрики якості:

- MCC = 0.834;

- Precision = 0.792;

- Recall = 0.913;

- F1-score = 0.849.

Аналіз помилок. Висока точність зумовлена простотою критерію (час перебування) та надійністю функції point-in-polygon. Помилки виникали

переважно через: втрату треку та присвоєння нового ID (4 випадків FN); помилкове визначення меж ROI на краях зони (11 випадків FP).

4.3.3 Порівняльний аналіз конфігурацій

Для оцінки внеску запропонованих удосконалень проведено порівняльний експеримент з різними комбінаціями детектор + трекер. Результати наведено в табл. 4.7.

Таблиця 4.7 – Порівняння конфігурацій за метрикою MCC

Конфігурація	Проїзд на червоний	Перевищення швидкості	Аварійне гальмування	Довге перебування в POI	Середнє MCC
YOLOv8n+ SORT (базова)	0,824	0,717	0,449	0,748	0,684
YOLOv8n+ Bytrack	0,843	0,727	0,501	0,792	0,716
YOLOv8n+ OCSort	0,849	0,738	0,511	0,800	0,725
YOLOv8n+ ByteTrack+	0,875	0,764	0,533	0,843	0,754
Запропонована IT	*0,861*	*0,758*	*0,515*	*0,834*	*0,742*

Аналіз результатів. Запропонована конфігурація (YOLOv8n + C3Ghost-CBAM + ByteTrack+) демонструє значне покращення показників MCC порівняно з базовою конфігурацією:

- покращення порівняно з baseline: +4,5% для Red Light, +5,7% для Speeding, +14,7% для Emergency Brake, +11,5% для ROI Overflow;
- середнє покращення MCC: +8,4% (з 0,684 до 0,742);

- найбільший приріст спостерігається для аномалії "аварійне гальмування" (+14,7%), що свідчить про ефективність удосконаленого трекера в умовах швидкої зміни динаміки об'єкта.

Порівняння ByteTrack+ з OC-SORT показує незначну перевагу запропонованого методу (+0,6% середнього MCC) при суттєво вищій швидкості обробки.

4.4 Застосування розробленої інформаційної технології аналізу відеопотоку дорожнього руху

У рамках апробації, проведеної у 2026 році на замовлення Вінницької міської ради, розроблену інтелектуальну інформаційну технологію застосовано для автоматичного моніторингу дорожнього руху на вулицях міста Вінниці. Тестування виконано на відеозаписах з трьох стаціонарних IP-камер різних виробників, встановлених у попередньо визначених зонах спостереження. Записи велись в різні дні у ранішній (07:30–09:30) та обідній (13:00–14:00) час доби, що забезпечило 6 годин відеоматеріалу на камеру. Відеозаписи нарізано на фрагменти тривалістю 30 секунд: 6 год × 3600 с ÷ 30 с = 720 фрагментів на камеру, або 2160 фрагментів загалом, що відповідає 1080 хвилинам тестового матеріалу. Характеристику зон наведено в таблиці 4.8.

Таблиця 4.8 – Характеристика зон спостереження та обсяг тестових даних

№	Локація	Задача/Аномалія	Відеофрагментів	Тривалість, хв	Умови зйомки
1	Перехрестя вул. Келецька × вул. 600-річчя	Проїзд на червоний сигнал	720	360	Ранок / обід; ясно / хмарно / засвітлення
2	Перехрестя вул. Келецька × вул. Шевченка	Перевищення швидкості; тривале перебування у зоні перехрестя	720	360	Ранок / обід; пряме засвітлення у ранкові години
3	Пішохідний перехід до Лісопарку / Хмельницьке шосе (пр. Космонавтів)	Розподіл швидкостей потоку (<30, 30–50, 50–70, >70 км/год)	720	360	Ранок / обід; різне освітлення
Разом			2 160	1 080	

Загальна процедура підготовки зон. Для кожної зони до початку тестування виконувався єдиний підготовчий цикл: (1) калібрування камери методом гомографічного перетворення (6 опорних точок, cv2.findHomography + RANSAC); (2) задання стоп-лінії, ROI-полігона зони перехрестя та ділянки для HSV-аналізу стану сигналу світлофора; (3) встановлення чисельних порогів (v_{max}, t_roi_max); (4) збереження параметрів у конфігураційному файлі зони. При обчисленні MCC враховувалися як аномальні, так і нормальні події (TP, FP, FN, TN), що забезпечило об'єктивну оцінку якості класифікатора при суттєво нерівномірному розподілі класів у тестовій вибірці.

Зона 1. Перехрестя вул. Келецька × вул. 600-річчя: виявлення проїзду на червоний сигнал

Камера охоплює підхід до перехрестя з боку проспекту, стоп-лінію та прилеглу ділянку проїжджої частини (рис. 4.8).

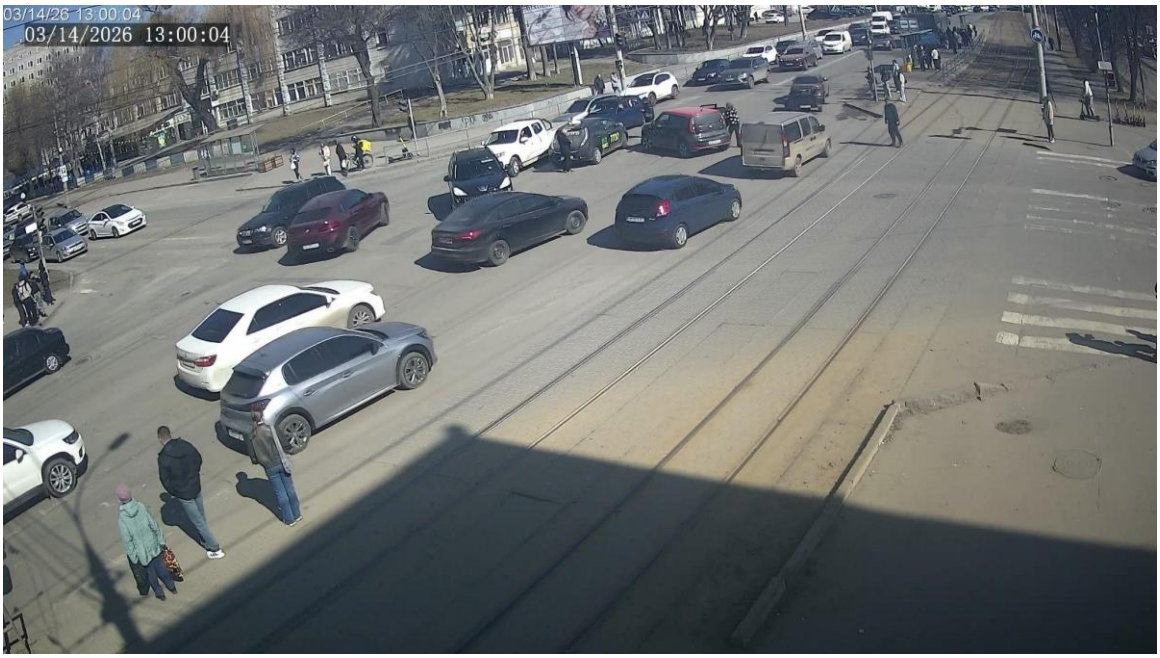


Рис. 4.8. Сцена з перехрестя вул. Келецька × вул. 600-річчя

Серед 720 відеофрагментів розмічено 25 аномальних подій. Більшість з них (близько 70 %) відбулися у момент одночасного горіння жовтого та червоного сигналів, тобто на кінці заборонної фази циклу, коли водії помилково сприймають увімкнення жовтого сигналу як дозвільний. Результати наведено в таблиці 4.9.

Таблиця 4.9 – Результати виявлення проїзду на червоний сигнал (вул. Келецька × вул. 600-річчя)

Тип аномалії	TP	FP	FN	TN	MCC
Проїзд на червоний сигнал (вул. Келецька × вул. 600-річчя)	22	8	3	687	0,796

Значення MCC = 0.796 відображає задовільну якість класифікатора з урахуванням значного дисбалансу класів (25 аномалій на 720 фрагментів). Хибнопозитивні випадки (FP = 8) переважно пов'язані з помилковою класифікацією ситуацій зупинки транспортного засобу безпосередньо перед стоп-лінією при жовтому сигналі. Хибнонегативні (FN = 3) – з оклюзією об'єкта маршрутним транспортом у момент перетину стоп-лінії.

Складнощі та обмеження. Камера розташована під кутом відносно кожного зі світлофорних пристроїв, що суттєво ускладнює HSV-аналіз стану сигналу. У низці ранкових сесій при прямому засвіченні сонця пересвітленість кадру робила неможливим розпізнавання кольору сигналу навіть візуально оператором. Для усунення цього обмеження як альтернативне джерело інформації залучено циклограми роботи світлофорного об'єкта, отримані від служби організації дорожнього руху. Водночас така залежність від зовнішнього джерела є обмеженням технології у сценарії автономного розгортання. Приклад такої ситуації наведена на рис 4.8, приближення зображень світлофорів без візуально видимого стану сигналу світлофора наведений на рис. 4.9.

Рис. 4.9. Приклад відсутності візуально видимого стану сигналу світлофора з перехрестя вул. Келецька × вул. 600-річчя

Трамваї та тролейбуси при проходженні через поле зору камери частково або повністю перекривали стоп-лінію та прилеглу ділянку, що призводило до втрати трекером об'єктів позаду і було найбільш відчутним у ранкові години-пік. Ці фактори у сукупності зумовили деяке зниження MCC (0,796) порівняно зі значенням, отриманим на стандартизованому датасеті (0,861).

Зазначені труднощі формують вимоги до положення камери: фронтальна або близька до фронтальної орієнтація відносно світлофорних пристроїв; відсутність у полі зору постійних джерел прямого засвітлення; забезпечення огляду стоп-лінії без перекриття великогабаритним транспортом.

Зона 2. Перехрестя вул. Келецька × вул. Шевченка: перевищення швидкості та тривале перебування в зоні перехрестя

Для спостереження використано IP-камеру типу «риб'яче око» (fisheye), що дозволяє охопити всю площу перехрестя одним пристроєм (рис. 4.10).

Рис. 4.10. Сцена з перехрестя вул. Келецька × вул. Шевченка

Перед гомографічним калібруванням виконано корекцію радіальної дисторсії об'єктива: без цього кроку залишкова дисторсія fisheye вносить

похибку у перетворення координат до 10–25 % на периферії поля зору, що безпосередньо позначається на точності вимірювання швидкості. Серед 720 фрагментів розмічено 16 випадків перевищення швидкості та 49 випадків тривалого перебування в зоні перехрестя (t > 10 с за відсутності червоного сигналу). Результати наведено в таблиці 4.10.

Таблиця 4.10 – Результати виявлення перевищення швидкості і тривалого перебування в зоні перехрестя (вул. Келецька × вул. Шевченка)

Тип аномалії	TP	FP	FN	TN	MCC
Перевищення швидкості	12	9	4	695	0,646
Тривале перебування в зоні перехрестя	44	12	5	659	0,828

Для аномалії «тривале перебування в зоні перехрестя» отримано $MCC = 0{,}828$, що пояснюється стійкістю просторово-часового критерію та повним охопленням зони ROI fisheye-камерою. Нижче значення $MCC = 0{,}646$ для перевищення швидкості відображає, крім малої кількості аномальних подій, чутливість оцінки швидкості до залишкової похибки гомографічного перетворення після корекції дисторсії.

Складнощі та обмеження. У ранішні години, коли сонце сходить у напрямку, що збігається з оптичною віссю камери, виникало значне пряме засвітлення, яке унеможливлювало аналіз зображення: пересвітлення охоплювало значну частину кадру і робило неможливим не лише автоматичне розпізнавання сигналів та транспортних засобів, а й їх візуальну ідентифікацію оператором (рис. 4.10). Такі фрагменти виключалися з розрахунку метрик або оброблялися із залученням циклограм. Широкий кут огляду та пов'язане з ним стиснення об'єктів у периферійних ділянках кадру можуть знижувати точність детектора об’єктів, навченого на зображеннях з перспективних камер, що

потенційно потребує донавчання моделі на зображеннях саме цієї камери. Водночас принципова перевага зони 2 полягає у фронтальному розташуванні світлофорного пристрою відносно камери, що забезпечує оптимальні умови для HSV-аналізу стану сигналу та усуває проблему кутового засвітлення, характерну для зони 1.

Зона 3. Пішохідний перехід до Лісопарку / Хмельницьке шосе: аналіз розподілу швидкостей

Камера орієнтована безпосередньо на нерегульований пішохідний перехід при вході до Лісопарку, а рух транспортних засобів по Хмельницькому шосе спостерігається на краю її поля зору (рис. 4.11).

Рис. 4.11. Сцена з пішохідного переходу до Лісопарку

Для кожного транспортного засобу, що послідовно перетинав дві вимірювальні лінії, перпендикулярні до осі руху, обчислювалась швидкість на основі часового інтервалу між перетинами та відомої метричної відстані між лініями. Загалом оброблено 3210 проїздів. Розподіл за діапазонами швидкостей наведено в таблиці 4.11.

Таблиця 4.11 – Розподіл транспортних засобів за діапазонами швидкостей (Хмельницьке шосе)

< 30 км/год	30–50 км/год	50–70 км/год	> 70 км/год	Примітка
22 %	45 %	27 %	6 %	Нерегульований пішохідний перехід; ліміт 50 км/год

Характерною особливістю розподілу є підвищена частка транспортних засобів, що рухалися зі швидкістю менше 30 км/год (22 %). Це пояснюється наявністю нерегульованого пішохідного переходу в зоні огляду: значна частина водіїв знижує швидкість при наближенні до нього незалежно від фактичної присутності пішоходів. Разом з тим 33 % транспортних засобів рухалися із перевищенням ліміту 50 км/год.

Складнощі та обмеження. Периферійне положення транспортного потоку у полі зору підвищує ймовірність неповних обмежувальних рамок при детекції та збільшує чутливість гомографічного перетворення до похибок калібрування на краях кадру. Для досягнення якості, порівнянної з результатами на стандартизованому датасеті, рекомендується або встановити додаткову камеру з фронтальним оглядом проїжджої частини, або переорієнтувати наявну так, щоб транспортний потік знаходився у центральній частині кадру.

Результати апробації підтверджують працездатність розробленої інформаційної технології в умовах реального вуличного відеоспостереження. Для аномалії з чіткими просторово-часовими критеріями – тривалого перебування в зоні перехрестя – отримано MCC = 0,828, що є порівнянним з результатом підрозділу 4.3 (0,834). Нижчі значення MCC для проїзду на червоний сигнал (0,796) та перевищення швидкості (0,646) відображають вплив специфічних умов встановлення камер: кутового розташування відносно світлофорних пристроїв і перекриття стоп-лінії маршрутним транспортом у

зоні 1; залишкової похибки гомографії fisheye-камери та малої кількості аномалій у тестовій вибірці у зоні 2. Проведена апробація дозволила сформулювати конкретні вимоги до положення камер, дотримання яких є необхідною передумовою для досягнення якості, порівнянної з результатами на стандартизованому датасеті.

Аналіз розподілу швидкостей на ділянці Хмельницького шосе підтверджує придатність технології для збору агрегованої статистики транспортного потоку, що є підставою для її інтеграції у системи управління дорожнім рухом міського рівня. Акт впровадження результатів дисертаційної роботи наведено у Додатку В.

4.5 Порівняльний аналіз отриманих результатів з комерційними рішеннями для моніторингу дорожнього руху

Серед сучасних комерційних рішень для аналізу дорожнього руху та виявлення аномальних подій можна виокремити низку інформаційних систем, які реалізують повний або частковий функціонал, подібний до розробленої інформаційної технології. Дані рішення орієнтовані на автоматизовану обробку відеопотоків, детекцію транспортних засобів, аналіз траєкторій та фіксацію порушень правил дорожнього руху.

Hikvision IVS (Intelligent Video Surveillance) [187] є інтегрованою системою відеоаналітики, що реалізується безпосередньо на рівні IP-камер виробника. Система використовує вбудовані модулі відеоаналітики для виявлення транспортних засобів, підрахунку інтенсивності руху та детекції окремих типів порушень, зокрема проїзду на заборонений сигнал світлофора та перевищення швидкості.

Алгоритмічна реалізація базується на сучасних згорткових нейронних мережах для детекції об’єктів та класичних методах багатооб’єктного трекінгу, оптимізованих для роботи на edge-пристроях.

Перевагами системи є швидке розгортання завдяки інтеграції аналітики безпосередньо у камери, прийнятна точність для базових сценаріїв моніторингу та відносно невисока вартість (орієнтовно $500–1000 за камеру для стандартних конфігурацій).

Обмеженнями є знижена стійкість трекінгу в умовах щільного трафіку та оклюзій, відсутність повноцінної мультикамерної кореляції подій, а також обмежена можливість глибокої кастомізації алгоритмів, що зумовлює орієнтацію системи на сценарії типу plug-and-play.

Hikvision ITS (Intelligent Traffic System) [188] є комплексною платформою інтелектуальних транспортних систем, призначеною для масштабного моніторингу дорожнього руху на рівні міської інфраструктури. На відміну від IVS, дана система підтримує централізоване управління великою кількістю перехресть, поєднуючи edge-обробку з серверною або хмарною аналітикою.

Архітектура системи включає:

- edge-модулі відеоаналітики для локальної детекції та трекінгу;
- централізовану платформу для зберігання, аналізу та візуалізації даних;
- веб-інтерфейс для конфігурації зон, калібрування камер та управління подіями.

Функціональність системи охоплює виявлення порушень сигналів світлофора, перевищення швидкості, несанкціонованого паркування, аналіз заторів та формування статистики трафіку в реальному часі.

Перевагами є масштабованість, підтримка розподіленої обробки, інтеграція з системами “розумного міста“ та можливість обробки потоків з сотень камер.

Обмеженнями залишаються висока складність первинного налаштування (калібрування, визначення зон, порогів), обмежена підтримка складних динамічних аномалій (зокрема аварійного гальмування), а також залежність від пропрієтарної екосистеми виробника.

Axis Communications (AXIS People Counter та суміжні рішення) [189] орієнтовані переважно на аналіз потоків людей, проте можуть використовуватися для обмежених сценаріїв моніторингу вуличного руху. Система базується на 2D-детекції об’єктів та простих механізмах трекінгу з використанням віртуальних ліній та зон інтересу.

Перевагами є простота інтеграції, підтримка REST API та можливість розгортання без додаткового серверного обладнання.

Обмеженнями є відсутність геометричного калібрування камер, що унеможливлює точне вимірювання швидкості та відстаней, а також відсутність механізмів аналізу складних аномальних сценаріїв дорожнього руху.

Genetec AutoVu [190] є спеціалізованим рішенням для автоматичного розпізнавання номерних знаків (ALPR) та фіксації порушень правил дорожнього руху. Система поєднує відеоаналітику, оптичне розпізнавання символів та трекінг транспортних засобів.

Перевагами є висока точність фіксації проїзду на заборонений сигнал світлофора, інтеграція з державними та муніципальними реєстрами, а також можливість масштабування на великі міські мережі.

Обмеженнями є висока вартість впровадження та експлуатації, а також залежність ефективності системи від якості розпізнавання номерних знаків, що знижується за несприятливих умов освітлення або при пошкоджених номерних табличках.

Viisights (AI Video Analytics) [191] є хмарною платформою відеоаналітики, орієнтованою на виявлення аномальних подій у різних сценаріях, включаючи дорожній рух. Система поєднує глибоке навчання з rule-based підходами для аналізу поведінкових патернів транспортних засобів.

Перевагами є підтримка широкого спектра аномалій, висока масштабованість та можливість конфігурації користувацьких правил.

Обмеженнями є значна вартість підписки, залежність від стабільності мережевого з'єднання та ризики, пов'язані з передачею відеоданих у хмарну інфраструктуру.

Sensys Networks / Citilog AID [192] є спеціалізованою системою автоматичного виявлення дорожніх інцидентів, що використовується на автомагістралях та швидкісних дорогах. Система підтримує виявлення зупинених транспортних засобів, неправильного напрямку руху, заторів та сторонніх об'єктів на проїзній частині.

Перевагами є орієнтація на інцидент-детекцію, підтримка edge- та централізованої обробки, а також зменшення кількості хибних спрацювань завдяки спеціалізованим алгоритмам.

Таблиця 4.12 – Порівняльна характеристика аналогів із розробленою ІТ

Система	Тип системи	Трекінг	Виявлення аномалій	Орієнтовна вартість	Якість аналітики
Hikvision IVS [185]	Edge (камера)	Чутливий до оклюзій	Проїзд на червоне	$500-1000 на камеру	Середня
Hikvision ITS [186]	Edge + Server/Cloud	Середній рівень	Проїзд на червоне, перевищення швидкості, паркування у забороненому місці	$1200-2000 на камеру	Висока
AXIS People Counter [187]	Edge (камера)	Базовий	Підрахунок пішоходів	$600-1200 на камеру	Низька-висока
Genetec AutoVu [188]	Edge+ cloud	Високий рівень	Проїзд на червоне, перевищення швидкості, паркування у забороненому місці	$2000-3000$ перехрестя + $200-400 в місяць	Висока

Продовження табл. 4.12

Система	Тип системи	Трекінг	Виявлення аномалій	Орієнтовна вартість	Якість аналітики
Viisights [189]	Cloud	Високий рівень	Виявлення ДТП, тривала зупинка авто у зоні, рух авто по тратуару, рух авто ривками	$3000-5000 в місяць	Дуже висока
Sensys Networks / Citilog AID [190]	Edge + Server	Високий рівень	Повільний рух, тривала зупинка авто, пішоходи на дорозі, сміття на дорозі, дим, затори	$2500+ на систему	Висока
Розроблена ІТ-технологія	Edge+ Server	Середній рівень	Проїзд на червоне, перевищення швидкості, стоянка в зоні перехрестя, аварійне гальмування	Витрати на камеру та встановленн я локального сервера	Середня

Проведений огляд комерційних систем моніторингу дорожнього руху показав, що сучасні рішення забезпечують достатній рівень автоматизації базових задач відеоаналітики, зокрема детекції транспортних засобів, підрахунку інтенсивності руху та фіксації окремих типів порушень правил дорожнього руху. Разом з тим, більшість проаналізованих систем орієнтовані на заздалегідь визначений перелік сценаріїв та використовують пропрієтарні алгоритми, що істотно обмежує можливості їх адаптації до специфічних умов дорожньої інфраструктури.

Комерційні рішення, реалізовані на рівні edge-пристроїв, характеризуються обмеженою стійкістю багатооб’єктного трекінгу в умовах щільного трафіку, значної кількості оклюзій та складних маневрів транспортних засобів. Хмарні платформи, навпаки, забезпечують ширший спектр аналітичних можливостей, однак супроводжуються високими експлуатаційними витратами, підвищеною латентністю обробки та ризиками, пов’язаними з передачею відеоданих у віддалені дата-центри.

У порівнянні з розглянутими комерційними рішеннями, розроблена інформаційна технологія орієнтована на усунення виявлених обмежень за рахунок поєднання локальної відеообробки з можливістю мультикамерної кореляції подій, розширеного аналізу траєкторій та виявлення динамічних аномалій дорожнього руху, таких як аварійне гальмування, тривале перебування транспортного засобу в зоні інтересу та нехарактерні поведінкові патерни.

Крім того, на відміну від пропрієтарних комерційних систем, запропонована технологія передбачає відкриту модульну архітектуру, що забезпечує можливість гнучкого налаштування алгоритмів, інтеграції нових методів аналізу та адаптації до різних типів дорожніх сцен без значних додаткових фінансових витрат. Це дозволяє розглядати розроблену інформаційну технологію як перспективне рішення для впровадження у системах інтелектуального моніторингу дорожнього руху з підвищеними вимогами до точності, масштабованості та економічної ефективності.

4.6 Висновки до розділу

У четвертому розділі здійснено програмну реалізацію та експериментальну апробацію розробленої інтелектуальної інформаційної технології аналізу відеопотоку дорожнього руху.

1. Реалізовано комплекс програм для навчання детектора на основі YOLOv8 з C3Ghost-CBAM мовою Python (PyTorch, Ultralytics). Проведено оптимізацію для edge-пристроїв через експорт в ONNX/TensorRT з INT8-квантуванням (зменшення розміру в 4 рази).

2. Розроблено програмні модулі: детекції (YOLOv8TensorRT), трекінгу (ByteTrackPlus з IoU-suppression, захистом втрачених треків, DIoU-matching),

виявлення аномалій для 4 типів порушень. Архітектуру представлено UML-діаграмою класів.

3. Апробація на датасеті з 3 перехресть показала: проїзд на червоне MCC=0,860; перевищення швидкості MCC=0,758; аварійне гальмування MCC=0,515; тривале перебування в ROI MCC=0,834. Запропонована конфігурація забезпечує середнє MCC=0,742 (+8,4% від базового методу).

4. Порівняння з комерційними рішеннями (Hikvision, Genetec, Viisights) показало вагомі переваги розробленої ІТ: відкрита архітектура, підтримка динамічних аномалій, нижча вартість.

Основні результати розділу опубліковані автором у роботах [4, 7, 8].

ЗАГАЛЬНІ ВИСНОВКИ

У дисертаційній роботі вирішено актуальну науково-практичну задачу підвищення якості виявлення аномалій у відеопотоці дорожнього руху шляхом розробки нової інформаційної технології на основі удосконалених методів детекції об'єктів, багатооб'єктного трекінгу та аналізу траєкторій. Основні наукові та практичні результати роботи полягають у наступному:

1. Проведено аналіз сучасних підходів до детекції об'єктів дорожнього руху, багатооб'єктного трекінгу та виявлення аномалій у відеопотоці. Встановлено, що існуючі методи характеризуються високими обчислювальними вимогами, недостатньою стійкістю до оклюзій та обмеженою адаптивністю до різних умов зйомки, що обґрунтовує необхідність розробки нових підходів, орієнтованих на роботу в реальному часі на пристроях з обмеженими ресурсами.

2. Уперше запропоновано модель детекції об'єктів дорожнього руху, яку, на відміну від існуючих, побудовано на основі модифікованої YOLO-архітектури із інтеграцією модулів C3Ghost-CBAM та полегшених згорток GhostConv. Це забезпечує зменшення кількості параметрів моделі на 23,3% (з 3,0 до 2,3 млн) та обчислювальної складності на 9,9% (з 8,1 до 7,3 GFLOPs) при незначному зниженні точності (значення метрики mAP@0,50:0,95 знизилось на 1,3%, з 0,715 до 0,706), що дозволяє використовувати модель на Edge-пристроях з обмеженими обчислювальними ресурсами.

3. Удосконалено метод багатооб'єктного трекінгу ByteTrack, який, на відміну від існуючих, поєднує три механізми: IoU-suppression для запобігання дублюванню треків при ініціалізації, механізм захисту втрачених треків (occlusion grace period) для обробки короткочасних оклюзій та метрику DIoU для покращення асоціації детекцій з треками. Це дозволило підвищити метрику MOTA на 0,5% (з 77,5% до 78,0%), IDF1 на 1,9% (з 75,8% до 77,7%) та

зменшити кількість перемикань ідентичностей на 9,0% (з 210 до 191) при збереженні швидкості оброблення понад 29 FPS на датасеті MOT17-val.

4. Уперше запропоновано формальну модель виявлення аномалій у відеопотоці дорожнього руху на основі аналізу траєкторій, яка, на відміну від існуючих, враховує чотири типи порушень: проїзд на червоний сигнал світлофора, перевищення швидкості, аварійне гальмування та тривале перебування в зоні перехрестя – шляхом комбінування геометричного (функція перетину стоп-лінії), спектрального (HSV-аналіз стану світлофора), кінематичного (різниця швидкостей між послідовними кадрами) та топологічного (функція point-in-polygon) критеріїв, що дозволило описати складні умови виявлення кожного типу порушення.

5. Розроблено архітектуру інтелектуальної інформаційної технології аналізу відеопотоку дорожнього руху, яка, на відміну від існуючих, використовує Edge-Cloud архітектуру з модульною структурою та забезпечує виконання повного пайплайну оброблення (детекція → трекінг → виявлення аномалій) безпосередньо на Edge-пристрої, що дозволяє здійснювати виявлення аномалій у реальному часі з передачею до Cloud лише метаданих. Така архітектура забезпечує економію мережевого трафіку на 99,6%, автономну роботу при втраті з'єднання та масштабованість системи. Розроблено комплект UML-діаграм, що формалізують структуру та взаємодію компонентів технології.

6. Виконано програмну реалізацію розроблених методів та проведено експериментальні дослідження на реальних датасетах. Апробація на власному датасеті (210 хвилин відео з трьох перехресть, 680 розмічених подій) підтвердила ефективність запропонованої інформаційної технології: середнє значення MCC = 0,742, що на 8,4% вище за базову конфігурацію. Порівняльний аналіз з комерційними рішеннями (Hikvision, Genetec, Viisights) показав конкурентоспроможність розробленої технології за функціональністю при суттєво нижчій вартості впровадження.

Результати дисертаційної роботи були використані в експерименті з метою аналізу відеозаписів з камер дорожнього руху на двох перехрестях та пішохідному переході міста Вінниці. Виявлені аномальні події були використані для аналізу аварійності перехресть, кількості порушень, що відбувались на цих перехрестях під час спостереження, а також для дослідження розподілу швидкостей транспортного потоку на пішохідному переході.

Результати роботи також можуть бути використані для створення систем автоматичного моніторингу дорожнього руху, фіксації порушень правил дорожнього руху, аналізу транспортних потоків та підвищення безпеки на дорогах. Перспективи подальших досліджень включають: інтеграцію модулів повторної ідентифікації (ReID) для мультикамерного трекінгу, розширення переліку типів аномалій (виявлення пішоходів на проїжджій частині, несанкціонованого паркування), покращення алгоритмів оцінки швидкості з використанням методів глибокого навчання та адаптацію системи до різних типів дорожньої інфраструктури.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Романець В., Бісікало О. Виявлення об'єктів дорожнього руху з камер відеоспостереження // Herald of Khmelnytskyi National University. Technical Sciences. 2025. Т. 355, № 4. С. 491–497. https://doi.org/10.31891/2307-5732-2025-355-70.

2. Романець В., Маслій Р. В. Виявлення аномалій у відеопотоці трафіку транспорту засобами комп'ютерного зору та глибинного навчання // Вісник ВПІ. 2025. Вип. 5. С. 146–155. https://doi.org/10.31649/1997-9266-2025-182-5-146-155.

3. Романець В., Бісікало О. Метод багатооб'єктного трекінгу з обробкою оклюзій для систем відеоспостереження // Measuring and Computing Devices in Technological Processes. 2025. Т. 84. № 4. С. 440–445. https://doi.org/10.31891/2219-9365-2025-84-53.

4. Романець В.О., Бісікало О.В. Інформаційна технологія інтелектуального аналізу відеопотоку дорожнього руху на основі Edge-Cloud архітектури // Вісник Херсонського національного технічного університету. 2026. №1. С.304-310. https://doi.org/10.35546/kntu2078-4481.2026.1.41.

5. Романець В., Бісікало О. Information technology in customer service: use of computer vision and decision support systems // SWorldJournal. 2023. Т. 1, № 21-01. С. 65–72. https://doi.org/10.30888/2663-5712.2023-21-01-046.

6. Романець В., Бісікало О., Маслій Р. Адаптація сучасних моделей комп'ютерного зору (YOLOv8-nano) для роботи в реальному часі в системах відеоаналітики дорожнього руху. ВНТКП ВНТУ. Факультет інтелектуальних інформаційних технологій та автоматизації. 2026. URL: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2026/paper/view/28227 (дата звернення: 25.04.2026).

7. Романець В., Бісікало О. Навчання модифікованої моделі YOLOv8 для детекції транспортних засобів на комбінованому датасеті. Молодь в науці:

дослідження, проблеми, перспективи (МН-2026). 2026. URL: https://conferences.vntu.edu.ua/index.php/mn/mn2026/paper/view/28837 (дата звернення: 30.04.2026).

8. Романець В., Бісікало О. Edge-Cloud архітектура інформаційної технології аналізу відеопотоку дорожнього руху. // Scientific development in a changing world. Proceedings of the 5th International scientific and practical conference. SPC “Sci-conf.com.ua”. Lviv, Ukraine. 2026. Pp. 363–367. URL: https://sci-conf.com.ua/wp-content/uploads/2026/05/SCIENTIFIC-DEVELOPMENT-IN-A-CHANGING-WORLD-11-13.05.26.pdf#page=363 (дата звернення: 13.05.2026).

9. Lowe D.G. Distinctive image features from scale-invariant keypoints // International Journal of Computer Vision. – 2004. – Vol. 60, № 2. – P. 91–110.

10. Dalal N., Triggs B. Histograms of oriented gradients for human detection // 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) : proceedings. – IEEE, 2005. – Vol. 1. – P. 886–893.

11. Viola P., Jones M. Rapid object detection using a boosted cascade of simple features // Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001. – IEEE, 2001. – Vol. 1. – P. I–I. – URL: https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf.

12. Viola P., Jones M.J. Robust real-time face detection // International Journal of Computer Vision. – 2004. – Vol. 57, № 2. – P. 137–154. – URL: http://dx.doi.org/10.1023/B:VISI.0000013087.49260.fb.

13. Wang Y.Q. An analysis of the Viola-Jones face detection algorithm // Image Processing On Line. – 2014. – Vol. 4. – P. 128–148. – URL: https://www.ipol.im/pub/art/2014/104/article_lr.pdf.

14. Kvyetnyy R., Maslii R., Harmash V., Bogach I., Kotyra A., Grądz Ż., Zhanpeisova A., Askarova N. Object detection in images with low light condition // Photonics Applications in Astronomy, Communications, Industry, and High Energy

Physics Experiments 2017 : proceedings of SPIE. – 2017. – Vol. 10445. – P. 104450W. – DOI: https://doi.org/10.1117/12.2281001.

15. Pietikäinen M. Image analysis with local binary patterns // Scandinavian Conference on Image Analysis : proceedings. – Berlin ; Heidelberg : Springer, 2005. – P. 115–118.

16. Ojala T., Pietikainen M., Maenpaa M. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns // Pattern Analysis and Machine Intelligence, IEEE Transactions on. – 2002. – Vol. 24, № 7. – P. 971–987. – URL: https://pdfs.semanticscholar.org/33fa/d977a6b317cfd6ecd43d978687e0df8a7338.pdf.

17. Маслій Р.В. Використання локальних бінарних шаблонів для розпізнавання облич на півтонових зображеннях // Електронні праці ВНТУ. – 2008. – Т. 4. – С. 6. – URL: http://masliy.vk.vntu.edu.ua/file/5569c524a1c4d8d28806db7443af037f.pdf.

18. Kenk M.A., Hassaballah M., Elhenawy I.M. On-road vehicles detection based on LBP derivatives fusion // Proceedings of the 11th International Conference on Informatics & Systems (INFOS 2018). – 2018. – URL: http://dx.doi.org/10.2139/ssrn.3389787.

19. Wang X., Han T.X., Yan S. An HOG-LBP human detector with partial occlusion handling // 2009 IEEE 12th International Conference on Computer Vision : proceedings. – IEEE, 2009. – P. 32–39.

20. Neumann D., Langner T., Ulbrich F., Spitta D., Goehring D. Online vehicle detection using Haar-like, LBP and HOG feature based image classifiers with stereo vision preselection // 2017 IEEE Intelligent Vehicles Symposium (IV) : proceedings. – IEEE, 2017. – P. 773–778.

21. Zhang Y., Zhao C., He J., Chen A. Vehicles detection in complex urban traffic scenes using Gaussian mixture model with confidence measurement // IET Intelligent Transport Systems. – 2016. – Vol. 10, № 6. – P. 445–452.

22. Bakti Rizki Yusliana, Areni Intan Sari, Prayogi A. Ais. Vehicle detection and tracking using gaussian mixture model and kalman filter // 2016 International Conference on Computational Intelligence and Cybernetics : proceedings. – IEEE, 2016.

23. Meher S.K., Murty M.N. Efficient method of moving shadow detection and vehicle classification // AEU, International Journal of Electronics and Communications. – 2013. – Vol. 67, № 8. – P. 665–670. – URL: https://2024.sci-hub.se/2356/63327772dcfec0a8827d0368bfb2fcb2/meher2013.pdf.

24. Choudhury S., Chattopadhyay S.P., Hazra T.K. Vehicle detection and counting using haar feature-based classifier // 2017 8th Annual Industrial Automation and Electromechanical Engineering Conference (IEMECON) : proceedings. – IEEE, 2017. – P. 106–109.

25. Ramadhani M.I., Minarno A.E., Cahyono E.B. Vehicle classification using haar cascade classifier method in traffic surveillance system // Kinetik. – 2018. – Vol. 3, № 1. – P. 57–64.

26. Ghahremannezhad H., Shi H., Liu C. Object detection in traffic videos: A survey // IEEE Transactions on Intelligent Transportation Systems. – 2023. – Vol. 24, № 7. – P. 6780–6799.

27. Girshick R., Donahue J., Darrell T., Malik J. Rich feature hierarchies for accurate object detection and semantic segmentation // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. – 2014. – P. 580–587.

28. Uijlings J.R., Van De Sande K.E., Gevers T., Smeulders A.W. Selective search for object recognition // International Journal of Computer Vision. – 2013. – Vol. 104, № 2. – P. 154–171.

29. Region-based CNNs (R-CNNs) // Dive into Deep Learning : электрон. ресурс. – Режим доступу: https://d2l.ai/chapter_computer-vision/rcnn.html.

30. Girshick R. Fast R-CNN // Proceedings of the IEEE International Conference on Computer Vision. – 2015. – P. 1440–1448.

31. Ren S., He K., Girshick R., Sun J. Faster R-CNN: Towards real-time object detection with region proposal networks // IEEE Transactions on Pattern Analysis and Machine Intelligence. – 2016. – Vol. 39, № 6. – P. 1137–1149.

32. He K., Gkioxari G., Dollár P., Girshick R. Mask R-CNN // Proceedings of the IEEE International Conference on Computer Vision. – 2017. – P. 2961–2969.

33. NVIDIA DIGITS : електрон. ресурс. – Режим доступу: https://developer.nvidia.com/digits.

34. Szegedy C., Liu W., Jia Y., Sermanet P., Reed S., Anguelov D., Rabinovich A. Going deeper with convolutions // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. – 2015. – P. 1–9.

35. Redmon J., Divvala S., Girshick R., Farhadi A. You only look once: Unified, real-time object detection // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. – 2016. – P. 779–788.

36. Liu W., Anguelov D., Erhan D., Szegedy C., Reed S., Fu C.Y., Berg A.C. SSD: Single shot multibox detector // European Conference on Computer Vision : proceedings. – Cham : Springer International Publishing, 2016. – P. 21–37.

37. Tan M., Pang R., Le Q.V. EfficientDet: Scalable and efficient object detection // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. – 2020. – P. 10781–10790.

38. Tan M., Le Q. EfficientNet: Rethinking model scaling for convolutional neural networks // International Conference on Machine Learning : proceedings. – PMLR, 2019. – P. 6105–6114.

39. Rashad R.A., Al-Hamdi H.S., George L.E. Vehicle detection using simplified Fast R-CNN // Proceedings of the 2018 First International Scientific Conference on Engineering Science (ISCES). – 2018. – P. 164–169.

40. Chaudhuri A. Smart Traffic Management of Vehicles using Faster R-CNN based Deep Learning Method // arXiv. – 2023. – URL: https://arxiv.org/pdf/2311.10099.

41. Li X., Pang Y., Jiang Y., Pan J. On-road vehicle detection in varying weather conditions using faster R-CNN with several region proposal networks // Multimedia Tools and Applications. – 2021. – Vol. 80. – P. 10511–10527.

42. Faruque M.O., Ghahremannezhad H., Liu C. Vehicle classification in video using deep learning // Machine Learning and Data Mining in Pattern Recognition, MLDM. – Leipzig : ibai publishing, 2019. – P. 117–131.

43. Коцюбинський В., Квєтний Р., Кириленко О., Бісікало О., Маслій Р. Розв'язання задачі підрахунку об'єктів при складних умовах отримання зображень // Інформаційні технології та комп'ютерне моделювання : матеріали міжнародної науково-практичної конференції. – Івано-Франківськ, 2018. – С. 44–47.

44. Квєтний Р.Н., Маслій Р.В., Кириленко О.М. Виявлення та класифікація об'єктів дорожнього руху при використанні середовища DIGITS // Оптико-електронні інформаційно-енергетичні технології. – 2020. – № 39(1). – С. 14–20.

45. Dai Z., Song H., Wang X., Fang Y., Yun X., Zhang Z., Li H. Video-based vehicle counting framework // IEEE Access. – 2019. – Vol. 7. – P. 64460–64470.

46. Tondelli D., Tosi F., Poggi M., Mattoccia S. Dual-view single-shot multibox detector at urban intersections: settings and performance evaluation // Sensors. – 2023. – Vol. 23, № 7. – P. 3483.

47. Onishchenko D., Liubchenko N., Podorozhniak A. License plate recognition system based on mask R-CNN // Automation of Technological and Business Processes. – 2023. – Vol. 15, № 3. – P. 37–43. – DOI: https://doi.org/10.15673/atbp.v15i3.2623.

48. Korablyov M., Axak N., Ivanisenko I. Traffic sign recognition using the mask R-CNN // Proceedings of the Modern Machine Learning Technologies Workshop (MoMLeT 2024). – Lviv, Ukraine, 2024. – P. 28–39. – URL: https://ceur-ws.org/Vol-3711/paper3.pdf.

49. Intelligent traffic management system using mask regions-convolutional neural network // Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics and Control. – 2025. – Vol. 10, № 3. – DOI: https://doi.org/10.22219/kinetik.v10i3.2233.

50. Lv C., Mittal U., Madaan V., Agrawal P. Vehicle detection and classification using an ensemble of EfficientDet and YOLOv8 // PeerJ Computer Science. – 2024. – Vol. 10. – P. e2233. – URL: https://peerj.com/articles/cs-2233/.

51. Guide to YOLO Models : електрон. ресурс / Roboflow. – URL: https://blog.roboflow.com/guide-to-yolo-models/.

52. Redmon J., Farhadi A. YOLO9000: Better, faster, stronger // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. – 2017. – P. 7263–7271.

53. Redmon J., Farhadi A. Yolov3: An incremental improvement // arXiv preprint arXiv:1804.02767. – 2018.

54. Bochkovskiy A., Wang C.Y., Liao H.Y.M. Yolov4: Optimal speed and accuracy of object detection // arXiv preprint arXiv:2004.10934. – 2020.

55. Ge Z., Liu S., Wang F., Li Z., Sun J. YOLOX: Exceeding YOLO series in 2021 // arXiv preprint arXiv:2107.08430. – 2021.

56. Jocher G. et al. YOLOv5 by Ultralytics (v7.0) // GitHub, 2020–2025. – URL: https://github.com/ultralytics/yolov5.

57. Li C., Li L., Jiang H., Weng K., Geng Y., Li L., Wei X. YOLOv6: A single-stage object detection framework for industrial applications // arXiv preprint arXiv:2209.02976. – 2022.

58. Wang C.Y., Bochkovskiy A., Liao H.Y.M. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. – 2023. – P. 7464–7475.

59. Jocher G. et al. Ultralytics YOLOv8 // GitHub, 2023–2025. – URL: https://github.com/ultralytics/ultralytics.

60. Wang C.Y., Yeh I.H., Mark Liao H.Y. Yolov9: Learning what you want to learn using programmable gradient information // European Conference on Computer Vision : proceedings. – Cham : Springer Nature Switzerland, 2024. – P. 1–21.

61. Wang A., Chen H., Liu L., Chen K., Lin Z., Han J. Yolov10: Real-time end-to-end object detection // Advances in Neural Information Processing Systems. – 2024. – Vol. 37. – P. 107984–108011.

62. Jocher G. et al. YOLOv11 by Ultralytics // GitHub, 2025. – URL: https://github.com/ultralytics/ultralytics.

63. Shekade A., Mahale R., Shetage R., Singh A., Gadakh P., Kumar S. Vehicle classification in traffic surveillance system using YOLOv3 model // Proceedings of the 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC). – Coimbatore, India : IEEE, 2020. – P. 1015–1019. – DOI: https://doi.org/10.1109/ICESC48915.2020.9155702.

64. Bourja O., Derrouz H., Abdelali H.A., Maach A., Thami R.O.H., Bourzeix F. Real time vehicle detection, tracking, and inter-vehicle distance estimation based on stereovision and deep learning using YOLOv3 // International Journal of Advanced Computer Science and Applications. – 2021. – Vol. 12, № 8. – P. 915–923. – DOI: https://doi.org/10.14569/IJACSA.2021.01208101.

65. Марчук Д. Аналіз сучасних алгоритмів виявлення і розпізнавання об'єктів з відеопотоку для систем управління паркуванням в реальному часі. Вісник Хмельницького національного університету. Технічні науки. 2023. № 3 (321). С. 17–23. https://doi.org/10.31891/2307-5732-2023-321-3-17-23.

66. Zhang H., Wang K., Liu Z. Real-time vehicle target detection in inclement weather conditions based on YOLOv4 // Frontiers in Neurorobotics. – 2023. – Vol. 17. – P. 1058723.

67. Can V.X., Vu P.X., Rui-fang M., Thuat V.T., Van Duy V., Noi N.D. Vehicle detection and counting under mixed traffic conditions in Vietnam using YOLOv4 // International Journal of Advanced Research in Engineering and Technology. – 2021. – Vol. 2, № 3. – P. 34–48.

68. Optimizing YOLOv4 hyperparameters for enhanced vehicle detection in intelligent transportation systems // International Journal of Intelligent Transportation Systems Research. – 2025. – Vol. 23, № 3. – DOI: https://doi.org/10.1007/s13177-025-00519-3.

69. Tang Y., Jiang H., Wang J. Enhancing urban traffic management through YOLOv5 and DeepSORT algorithms within digital twin frameworks // Mechatronics and Intelligent Transportation Systems. – 2024. – Vol. 3, № 1. – P. 35–51.

70. Wang Y., Lin J., Zhang H. A high-precision vehicle detection and tracking method based on the attention mechanism // Sensors. – 2023. – Vol. 23, № 1. – P. 478.

71. Zheng Z., Sun L., Wang Y. Vehicle detection for traffic management systems based on YOLOv5 with spatial attention // Applied Sciences. – 2022. – Vol. 12, № 6. – P. 2876.

72. Kumar S., Singh S.K., Varshney S., Singh S., Kumar P., Kim B.-G., Ra I.-H. Fusion of deep SORT and YOLOv5 for effective vehicle detection and tracking scheme in real-time traffic management sustainable system // Sustainability. – 2023. – Vol. 15, № 24. – P. 16869.

73. Жеребух О., Фармага І. Використання нейронних мереж для визначення об'єктів на зображенні. Computer Design Systems. Theory and Practice. 2024. Т. 6, № 1. С. 232–240. https://doi.org/10.23939/cds2024.01.232.

74. Шматко О. В., Гамаюн І. П., Коломійцев О. В., Третяк В. Ф., Рудаков І. С., Бердочник А. Д. Дослідження та оцінка підсистеми виявлення та класифікації об'єктів у відеопотоці. Системи обробки інформації. 2024. № 4 (179). С. 70–80. https://doi.org/10.30748/soi.2024.179.08.

75. Xiong H., Leng J., Liu Y. Research on vehicle detection based on improved YOLOX_S // Scientific Reports. – 2023. – Vol. 13. – P. 22638.

76. Xiong C. et al. Vehicle detection algorithm based on lightweight YOLOX // Signal, Image and Video Processing. – 2023. – Vol. 17, № 5. – P. 1793–1800.

77. Ling H., Zhao T., Zhang Y. Engineering vehicle detection based on improved YOLOv6 // Applied Sciences. – 2024. – Vol. 14, № 17. – P. 8054. – DOI: https://doi.org/10.3390/app14178054.

78. Kulkarni S., Mittal N., Gupta R.R., Ponnalagu R.N. Investigation of YOLO models in the detection and classification of multiple negative road anomalies // Proceedings of the 2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT). – Delhi, India : IEEE, 2023. – P. 1–7. – DOI: https://doi.org/10.1109/ICCCNT56998.2023.10306347.

79. Hu Z.Y. Research on small target detection in optical remote sensing based on YOLOv7 // Proceedings of the 2023 IEEE International Conference on Sensors, Electronics and Computer Engineering (ICSECE). – Jinzhou, China : IEEE, 2023. – P. 804–809. – DOI: https://doi.org/10.1109/ICSECE58870.2023.10263365.

80. Wibowo A., Trilaksono B.R., Hidayat E.M.I., Munir R. Object detection in dense and mixed traffic for autonomous vehicles with modified YOLO // IEEE Access. – 2023. – Vol. 11. – P. 134866–134877. – DOI: https://doi.org/10.1109/ACCESS.2023.3335826.

81. Bakirci M. Utilizing YOLOv8 for enhanced traffic monitoring in intelligent transportation systems (ITS) applications // Digital Signal Processing. – 2024. – Vol. 152, № 2. – P. 104594. – DOI: https://doi.org/10.1016/j.dsp.2024.104594.

82. Doria Usta D., Hundelshaussen R., López Martínez C., Coimbra Leite Costa J.F., Machado Marques D. Intelligent traffic management: Comparative evaluation of YOLOv3, YOLOv5, and YOLOv8 for vehicle detection in urban environments in Montería, Colombia // Future Transportation. – 2025. – Vol. 5, № 4. – P. 191. – DOI: https://doi.org/10.3390/futuretransp5040191.

83. Liu Q., Liu Y., Lin D. Revolutionizing target detection in intelligent traffic systems: YOLOv8-SnakeVision // Electronics. – 2023. – Vol. 12, № 24. – P. 4970. – DOI: https://doi.org/10.3390/electronics12244970.

84. Enhancing YOLOv8 for vehicle detection in intelligent traffic management // Metallurgical and Materials Engineering. – 2025. – Vol. 31, № 4. – P. 190–200.

85. Bakirci M. Real-time vehicle detection using YOLOv8-Nano for intelligent transportation systems // Traitement du Signal. – 2024. – Vol. 41, № 4. – P. 1727.

86. Zhang H., Liang M., Wang Y. YOLO-BS: a traffic sign detection algorithm based on YOLOv8 // Scientific Reports. – 2025. – Vol. 15, № 1. – P. 7558. – DOI: https://doi.org/10.1038/s41598-025-88184-0.

87. Назаркевич М. А., Олексів Н. Т. Система розпізнавання об'єктів на основі моделі YOLO. Український журнал інформаційних технологій. 2024. Т. 6, № 1. С. 120–126. https://doi.org/10.23939/ujit2024.01.120.

88. Kumar M.R., Sendur G.J., Kumar S.P., Annamalai R. YOLOv9 in action: A futuristic approach to real-time license plate recognition and analysis // Proceedings of the 2024 IEEE International Conference on Intelligent Signal Processing and Effective Communication Technologies (INSPECT). – Gwalior, India : IEEE, 2024. – P. 1–6. – DOI: https://doi.org/10.1109/INSPECT63485.2024.10896171.

89. Smart traffic monitoring with YOLOv9 object detection algorithm // Akıllı Ulaşım Sistemleri ve Uygulamaları Dergisi. – 2025. – Vol. 8, № 1.

90. Zhang H. Real-time pedestrian detection algorithm of YOLOv10 under complex lighting and occlusion conditions // Proceedings of the 2024 International Conference on Advances in Electrical Engineering and Computer Applications (AEECA). – Dalian, China : IEEE, 2024. – P. 645–650. – DOI: https://doi.org/10.1109/AEECA62331.2024.00114.

91. Wang S., Han C., Xiao M., Zhou W., Zhang Y. Design of a road traffic classification accident detection and identification system based on YOLOv10 // International Conference of Pioneering Computer Scientists, Engineers and Educators. – Singapore : Springer Nature Singapore, 2025.

92. Alif M.A.R. YOLOv11 for vehicle detection: Advancements, performance, and applications in intelligent transportation systems // arXiv preprint arXiv:2410.22898. – 2024.

93. Gao H. Vehicle detection and tracking based on YOLOv11 // Proceedings of the 2nd International Conference on Data Science and Engineering (ICDSE 2025). – 2025. – P. 481–486. – DOI: https://doi.org/10.5220/0013699700004670.

94. Zhang Z., Xu Y., Huang J., Gao W., Li S. ZZ-YOLOv11: A lightweight vehicle detection model based on improved YOLOv11 // Sensors. – 2025. – Vol. 25, № 11. – P. 3399. – DOI: https://doi.org/10.3390/s25113399.

95. Dosovitskiy A., Beyer L., Kolesnikov A., Weissenborn D., Zhai X., Unterthiner T., Dehghani M., Minderer M., Heigold G., Gelly S., Uszkoreit J., Houlsby N. An image is worth 16x16 words: Transformers for image recognition at scale // arXiv preprint arXiv:2010.11929. – 2020. – DOI: https://doi.org/10.48550/arXiv.2010.11929.

96. Liu Z., Lin Y., Cao Y., Hu H., Wei Y., Zhang Z., Lin S., Guo B. Swin transformer: Hierarchical vision transformer using shifted windows // Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). – Montreal, Canada : IEEE, 2021. – P. 9992–10002. – DOI: https://doi.org/10.1109/ICCV48922.2021.00986.

97. Touvron H., Cord M., Douze M., Massa F., Sablayrolles A., Jégou H. Training data-efficient image transformers & distillation through attention // International Conference on Machine Learning. – PMLR, 2021. – P. 10347–10357.

98. Carion N., Massa F., Synnaeve G., Usunier N., Kirillov A., Zagoruyko S. End-to-end object detection with transformers // European Conference on Computer Vision. – Cham : Springer International Publishing, 2020. – P. 213–229.

99. Zhao Y., Lv W., Xu S., Wei J., Wang G., Dang Q., Chen J. DETRs beat YOLOs on real-time object detection // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. – 2024. – P. 16965–16974.

100. Ahmed M., El-Sheimy N., Leung H. Enhancing object tracking in smart city intelligent transportation systems: A track-by-detection approach utilizing satellite video monitoring // Proceedings of the 2024 IEEE International Conference

on Smart Mobility (SM). – IEEE, 2024. – P. 17–24. – DOI: https://doi.org/10.1109/SM63044.2024.10733409.

101. Sadik M.N., Hossain T., Sayeed F. Real-time detection and analysis of vehicles and pedestrians using deep learning // arXiv preprint arXiv:2404.08081. – 2024. DOI: https://doi.org/10.48550/arXiv.2404.08081.

102. Dendorfer P., Osep A., Milan A., et al. MOT20: A benchmark for multi object tracking // arXiv preprint arXiv:2003.09003. – 2020. – DOI: https://doi.org/10.48550/arXiv.2003.09003.

103. Кривенчук Ю., Марко О. Система розпізнавання та трекінгу об'єктів на відео. Вісник Хмельницького національного університету. Технічні науки. 2022. № 3 (309). С. 61–65. https://doi.org/10.31891/2307-5732-2022-309-3-61-65.

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

105. Кисіль Т. М., Зінченко О. В. Згорткові нейронні мережі для аналізу рухомих об'єктів у відеопотоці. Зв'язок. 2025. № 4. С. 48–57. https://doi.org/10.31673/2412-9070.2025.042042.

106. Кушнір Д. О. Методи відстеження довільної кількості об'єктів у реальному часі на мобільній платформі. Computer Systems and Networks. 2023. Т. 5, № 1. https://doi.org/10.23939/csn2023.01.050.

107. Назар О., Назаркевич М., Мішковський Ю., Наконечний Н., Романчук Р. Побудова системи ідентифікації рухомих об'єктів на основі технологій машинного навчання. Кібербезпека: освіта, наука, техніка. 2024. № 1 (25). С. 410–433. https://doi.org/10.28925/2663-4023.2024.25.410433.

108. Щербатюк М., Маслій Р. Аналіз методів однооб'єктного трекінгу, що незалежні від класу // Measuring and Computing Devices in Technological Processes. – 2025. – № 3. – P. 369–375. – DOI: https://doi.org/10.31891/2219-9365-2025-83-45.

109. Luo W., Xing J., Milan A., Zhang X., Liu W., Kim T.K. Multiple object tracking: A literature review // Artificial Intelligence. – 2021. – Vol. 293. – P. 103448. – DOI: https://doi.org/10.1016/j.artint.2020.103448.

110. Kristan M., Leonardis A., Matas J., et al. The sixth visual object tracking VOT2018 challenge results // Proceedings of ECCV Workshops. – 2019. – P. 1–21. – DOI: https://doi.org/10.1007/978-3-030-11009-3_1.

111. Bewley A., Ge Z., Ott L., Ramos F., Upcroft B. Simple online and realtime tracking // Proceedings of IEEE ICIP. – 2016. – P. 3464–3468. – DOI: https://doi.org/10.1109/ICIP.2016.7533003.

112. Zhou X., Koltun V., Krähenbühl P. Tracking objects as points // Proceedings of ECCV. – 2020. – P. 474–490. – DOI: https://doi.org/10.1007/978-3-030-58548-8_28.

113. Pang J., Qiu L., Li X., Chen H., Li Q., Darrell T., Yu F. Quasi-dense similarity learning for multiple object tracking // Proceedings of IEEE/CVF CVPR. – 2021. – P. 164–173. – DOI: https://doi.org/10.1109/CVPR46437.2021.00023.

114. Beginner's guide to SORT: Simple online and realtime tracking in Python // Roboflow Blog. – URL: https://blog.roboflow.com/sort-explained-real-time-object-tracking-in-python/.

115. Wojke N., Bewley A., Paulus D. Deep SORT: Simple online and realtime tracking with a deep association metric // Proceedings of IEEE ICIP. – 2017. – P. 3645–3649. – DOI: https://doi.org/10.1109/ICIP.2017.8296962.

116. Zhang Y., Wang C., Wang X., Zeng W., Liu W. FairMOT: On the fairness of detection and re-identification in multiple object tracking // Proceedings of IEEE/CVF CVPR. – 2020. – P. 12239–12248. – arXiv preprint arXiv:2004.01888.

117. Zhang Y., Sun P., Jiang Y., Yu D., Weng F., Yuan Z., Wang X. ByteTrack: Multi-object tracking by associating every detection box // Proceedings of ECCV. – 2022. – P. 1–21. – DOI: https://doi.org/10.1007/978-3-031-20047-2_1.

118. Cao J., Pang J., Weng X., Khirodkar R., Kitani K. Observation-centric SORT: Rethinking SORT for robust multi-object tracking // Proceedings of

IEEE/CVF CVPR. – 2023. – P. 9686–9696. – DOI: https://doi.org/10.1109/CVPR52729.2023.00934.

119. Aharon N., Orfaig R., Bobrovsky B.Z. BoT-SORT: Robust associations multi-pedestrian tracking // arXiv preprint arXiv:2206.14651. – 2022. – DOI: https://doi.org/10.48550/arXiv.2206.14651.

120. Sun P., Cao J., Jiang Y., et al. TransTrack: Multiple-object tracking with transformer // Proceedings of IEEE/CVF CVPR. – 2021. – P. 8824–8833. – arXiv preprint https://arxiv.org/abs/2012.15460.

121. Zeng F., Dong B., Zhang Y., Wang T., Zhang X., Wei Y. MOTR: End-to-end multiple-object tracking with transformer // Proceedings of ECCV. – 2022. – P. 659–675. – DOI: https://doi.org/10.1007/978-3-031-19812-0_38.

122. Meinhardt T., Kirillov A., Leal-Taixé L., Feichtenholer C. TrackFormer: Multi-object tracking with transformers // Proceedings of IEEE/CVF CVPR. – 2022. – P. 8844–8854. – DOI: https://doi.org/10.1109/CVPR52688.2022.00864.

123. Wu P., Pan C., Yan Y., Pang G., Wang P., Zhang Y. Deep learning for video anomaly detection: A review // arXiv preprint arXiv:2409.05383. – 2024. – URL: https://arxiv.org/pdf/2409.05383 (дата звернення: 06.09.2025).

124. HEIMDALL: An AI-based infrastructure for traffic monitoring and anomalies detection // ResearchGate. – URL: https://www.researchgate.net/publication/348390012_HEIMDALL_an_AI-based_infrastructure_for_traffic_monitoring_and_anomalies_detection (дата звернення: 06.09.2025).

125. Duong H.-T., Le V.-T., Hoang V.T. Deep learning-based anomaly detection in video surveillance: A survey // Sensors. – 2023. – Vol. 23, № 11. – Art. 5024. – DOI: https://doi.org/10.3390/s23115024.

126. Adminaité-Fodor D., Jost G. Reducing Speeding in Europe. PIN Flash Report 36. Brussels : European Transport Safety Council (ETSC), 2019. URL: https://etsc.eu/wp-content/uploads/PIN-Flash-36-Reducing-speeding-in-Europe-Final.pdf (дата звернення: 25.04.2026).

127. Статистика. Патрульна поліція України : веб-сайт. URL: https://patrolpolice.gov.ua/statystyka/ (дата звернення: 23.01.2026).

128. Insurance Institute for Highway Safety (IIHS). Red Light Running. IIHS–HLDI, 2024. URL: https://www.iihs.org/research-areas/red-light-running (дата звернення: 25.04.2026).

129. Cicchino J. B. Effectiveness of forward collision warning and autonomous emergency braking systems in reducing front-to-rear crash rates. Accident Analysis & Prevention. 2017. Vol. 99. P. 142–152. DOI: https://doi.org/10.1016/j.aap.2016.11.009.

130. Federal Highway Administration (FHWA). About Intersection Safety. U.S. Department of Transportation, 2024. URL: https://highways.dot.gov/safety/intersection-safety/about (дата звернення: 25.04.2026).

131. Adam A., Rivlin E., Shimshoni I., Reinitz D. Robust real-time unusual event detection using multiple fixed-location monitors // IEEE Transactions on Pattern Analysis and Machine Intelligence. – 2008. – Vol. 30, № 3. – P. 555–560.

132. Cortes C., Vapnik V. Support-vector networks // Machine Learning. – 1995. – Vol. 20, № 3. – P. 273–297. – URL: https://link.springer.com/content/pdf/10.1007/bf00994018.pdf.

133. Morris B.T., Trivedi M.M. Learning trajectory patterns by clustering: Experimental studies and comparative evaluation // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. – 2011. – P. 312–319.

134. Ester M., Kriegel H.P., Sander J., Xu X. A density-based algorithm for discovering clusters in large spatial databases with noise // KDD. – 1996. – Vol. 96, № 34. – P. 226–231.

135. Hu W., Xiao X., Fu Z., Xie D., Tan T., Maybank S. A system for learning statistical motion patterns // IEEE Transactions on Pattern Analysis and Machine Intelligence. – 2006. – Vol. 28, № 9. – P. 1450–1464.

136. Choudhry N., Abawajy J., Huda S., Rao I. A comprehensive survey of machine learning methods for surveillance videos anomaly detection // IEEE Access. – 2023. – Vol. 11. – P. 114680–114713. – DOI: https://doi.org/10.1109/ACCESS.2023.3321800.

137. Маслій Р.В., Гармаш В.В., Ковальчук П.П., Кабачій В.В. Аналіз методів знаходження аномалій у зображеннях // Інформаційні технології та комп'ютерна інженерія. – 2024. – Вип. 59, № 1. – С. 13–22. – DOI: https://doi.org/10.31649/1999-9941-2024-59-1-13-22 (дата звернення: 06.09.2025).

138. Симбирский Г.Д. Аналіз методів виявлення аномалій у відеоряді камер відеоспостереження на транспорті // Вісник Харківського національного автомобільно-дорожного університету. – 2022. – № 98. – С. 26–36. – DOI: https://doi.org/10.30977/BUL.2219-5548.2022.98.0.26 (дата звернення: 06.09.2025).

139. Hasan M., Choi J., Neumann J., Roy-Chowdhury A.K., Davis L.S. Learning temporal regularity in video sequences // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. – 2016. – P. 733–742.

140. Anomaly detection for radio signals with neural networks // PanoRadio-SDR. – URL: https://panoradio-sdr.de/anomaly-detection-for-radio-signals-with-neural-networks/ (дата звернення: 06.09.2025).

141. Kingma D.P., Welling M. Auto-encoding variational Bayes // arXiv preprint arXiv:1312.6114. – 2013. – URL: https://arxiv.org/abs/1312.6114.

142. Sabokrou M., Fathy M., Hoseini M., Klette R. Real-time anomaly detection and localization in crowded scenes // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. – 2015. – P. 56–62.

143. Schlegl T., Seeböck P., Waldstein S.M., et al. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery // International Conference on Information Processing in Medical Imaging (IPMI). – 2019. – P. 146–156.

144. Gong D., Liu L., Le V., Saha B., Mansour M.R., Venkatesh S., van den Hengel A. Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection // IEEE International Conference on Computer Vision (ICCV). – 2019. – P. 1705–1714. – DOI: https://doi.org/10.1109/ICCV.2019.00179.

145. Liu W., Luo W., Lian D., Gao S. Future frame prediction for anomaly detection–a new baseline // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. – 2018. – P. 6536–6545. – URL: https://arxiv.org/pdf/1712.09867 (дата звернення: 06.09.2025).

146. Goodfellow I., Pouget-Abadie J., Mirza M., Xu B., Warde-Farley D., Ozair S., Courville A., Bengio Y. Generative adversarial nets // Advances in Neural Information Processing Systems (NIPS). – 2014. – P. 2672–2680

147. Schlegl T., Seebock P., Waldstein S.M., Schmidt-Erfurth U., Langs G. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery // International Conference on Information Processing in Medical Imaging (IPMI). – 2017. – P. 146–157.

148. Akcay S., Atapour-Abarghouei A., Breckon T.P. GANomaly: Semi-supervised anomaly detection via adversarial training // IEEE International Conference on Computer Vision (ICCV). – 2018. – URL: https://arxiv.org/abs/1805.06725.

149. Nguyen K.-T., Dinh D.-T., Do M.-N., Tran M.-T. Anomaly detection in traffic surveillance videos with GAN-based future frame prediction // Proceedings of the 2020 International Conference on Multimedia Retrieval (ICMR). – 2020. – P. 457–463. – DOI: https://doi.org/10.1145/3372278.3390701.

150. Shi X., Chen Z., Wang H., Yeung D.-Y., Wong W.-K., Woo W.-C. Convolutional LSTM network: A machine learning approach for precipitation forecasting // Advances in Neural Information Processing Systems (NIPS). – 2015. – P. 802–810.

151. Lu C., Shi J., Jia J. Abnormal event detection at 150 fps in MATLAB // IEEE International Conference on Computer Vision (ICCV). – 2013. – P. 2720–2727.

152. Yong S.C., Tay Y.H. Abnormal event detection in videos using spatiotemporal autoencoder // International Symposium on Neural Networks. – 2017. – P. 189–196. – Springer.

153. Medel, J. R., & Savakis, A. Anomaly detection in video using predictive convolutional long short-term memory networks // arXiv. – 2016. – URL: https://arxiv.org/pdf/1612.00390.

154. Karpukhin V., Shalumov B., Zhang X., Schmeink A. Video stabilization using optical flow // IEEE International Symposium on Mixed and Augmented Reality (ISMAR). – 2019.

155. Luo W., Liu W., Gao S. A revisit of sparse coding based anomaly detection in stacked RNN framework // IEEE International Conference on Computer Vision (ICCV). – 2017. – P. 341–349.

156. Tuzel O., Porikli F., Meer P. Region covariance: A fast descriptor for detection and classification // European Conference on Computer Vision (ECCV). – 2006. – P. 589–600. – Springer.

157. Luc P., Couprie C., Chintala S., LeCun Y. Predicting deeper into the future of semantic segmentation // IEEE International Conference on Computer Vision (ICCV). – 2017. – P. 648–656.

158. Yao R., Lin G., Qi F., Zhang D. A novel trajectory clustering algorithm based on multi-agent reinforcement learning // IEEE Transactions on Vehicular Technology. – 2020. – Vol. 69, № 1. – P. 1– 12. – DOI: https://doi.org/10.1109/TVT.2019.2954472.

159. Fernández-Rodríguez J.D., García-González J., Benítez-Rochel R., Molina-Cabello M.A., Ramos-Jiménez G., López-Rubio E. Anomalous trajectory detection for automated traffic video surveillance // International Work-Conference on the Interplay Between Natural and Artificial Computation (IWNC). – 2022. – P. 173–182.

160. Duong H.-T., Le V.-T., Hoang V.T. Deep learning-based anomaly detection in video surveillance: A survey // Sensors. – 2023. – Vol. 23, № 11. – Art. 5024. – DOI: https://doi.org/10.3390/s23115024.

161. OpenCV: Basic concepts of the homography explained with code // OpenCV Documentation. – URL: https://docs.opencv.org/4.x/d9/dab/tutorial_homography.html.

162. Isaloo M.A., Azimifar Z. Anomaly detection on traffic videos based on trajectory simplification // IEEE Conference Publication. – 2013. – URL: https://ieeexplore.ieee.org/document/6779978/.

163. Колесницький О. К., Кукунін С. В., Дерев'янко М. Ю., Преподобний Мендеш Да Майа А. А. Розпізнавання нетипових ситуацій на дорозі за допомогою згорткової нейронної мережі // Оптико-електронні інформаційно-енергетичні технології. – 2020. – № 38(2). – С. 38–44. – DOI: https://doi.org/10.31649/1681-7893-2019-38-2-38-44.

164. Rezaei M., Azarmi M., Mir F. M. P. Traffic-net: 3d traffic monitoring using a single camera [preprint]. – 2021. – URL: https://arxiv.org/pdf/2109.09165 (дата звернення: 06.09.2025).

165. D'amicantonio G., Bondarau E. uTRAND: Unsupervised Anomaly Detection in Traffic Trajectories // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. – 2024. – P. 7638–7645. – URL: https://arxiv.org/pdf/2404.12712 (дата звернення: 06.09.2025).

166. Cai Y., Wang H., Chen X., Jiang H. Trajectory-based anomalous behaviour detection for intelligent traffic surveillance // IET Intelligent Transport Systems. – 2015. – Vol. 9, № 10. – P. 810–816. – DOI: https://doi.org/10.1049/iet-its.2014.0137.

167. Forsyth D.A., Ponce J. Computer Vision: A Modern Approach. – 2nd ed. – Pearson, 2011.

168. Wu J. Camera Calibration Using Homography Estimation // Galliot. – 2023. – URL: https://galliot.us/blog/camera-calibration-using-homography-estimation/.

169. Luvizon D. C., Nassu B. T., Minetto R. A video-based system for vehicle speed measurement in urban roadways // IEEE Transactions on Intelligent Transportation Systems. – 2017. – Vol. 18, № 6. – P. 1393–1404.

170. Dubská M., Herout A., Sochor J. Automatic camera calibration for traffic understanding // Proceedings of the British Machine Vision Conference. – 2014. – Vol. 4. – P. 8.

171. Sochor J., Juránek R., Herout A. Traffic surveillance camera calibration by 3d model bounding box alignment for accurate vehicle speed measurement // Computer Vision and Image Understanding. – 2017. – Vol. 161. – P. 87–98.

172. Bartl V., Špaňhel J., Dobeš P., Juránek R., Herout A. Automatic camera calibration by landmarks on rigid objects // Machine Vision and Applications. – 2021. – Vol. 32, № 1. – P. 1–13.

173. Revaud J., Humenberger M. Robust automatic monocular vehicle speed estimation for traffic surveillance // Proceedings of the IEEE/CVF International Conference on Computer Vision. – 2021. – P. 4551–4561.

174. D'Amicantonio G., Bondarev E., De With P.H.N. Homography estimation for camera calibration in complex topological scenes // IEEE Intelligent Vehicles Symposium (IV). – 2023. – P. 1–8. – DOI: https://doi.org/10.1109/IV55152.2023.10186786.

175. A. P. Shah, J. B. Lamare, T. Nguyen-Anh, and A. Hauptmann, “CADP: A novel dataset for CCTV traffic camera based accident analysis,“ 15th IEEE International Conference on Advanced Video and Signal Based Surveillance(AVSS), 2018, pp. 1-9. [Online]. Available: https://arxiv.org/pdf/1809.05782 . Accessed: 06.09.2025.

176. P. Pradeep Kumar and K. Kant, “TU-DAT: A computer vision dataset on road traffic anomalies,“ Sensors, vol. 25, no. 11, Art. 3259, 2025. [Online]. Available: https://www.mdpi.com/1424-8220/25/11/3259 . Accessed: 06.09.2025.

177. Lin T.-Y., Dollár P., Girshick R., He K., Hariharan B., Belongie S. Feature pyramid networks for object detection // IEEE Conference on Computer Vision and Pattern Recognition (CVPR). – 2017. – Vol. 30. – P. 2117–2125. – DOI: https://doi.org/10.1109/CVPR.2017.106.

178. Liu S., Qi L., Qin H., Shi J., Jia J. Path aggregation network for instance segmentation // IEEE Conference on Computer Vision and Pattern Recognition (CVPR). – 2018. – Vol. 31. – P. 8759–8768. – DOI: https://doi.org/10.1109/CVPR.2018.00913.

179. Zeng W., He X., Xiong S., Chen H., Qing L. DRYDet: decoupling YOLOv5 detector and refining the tasks in remote sensing scenario // Journal of Applied Remote Sensing. – 2022. – Vol. 16, № 4. – Art. 046513. – DOI: https://doi.org/10.1117/1.JRS.16.046513.

180. Han K., Wang Y., Tian Q., Guo J., Xu C., Xu C. GhostNet: More features from cheap operations // IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). – 2020. – Vol. 33. – P. 1580–1589. – DOI: https://doi.org/10.1109/CVPR42600.2020.00165.

181. Woo S., Park J., Lee J. Y., Kweon I. S. CBAM: Convolutional block attention module // European Conference on Computer Vision (ECCV). – 2018. – Vol. 15. – P. 3–19. – DOI: https://doi.org/10.1007/978-3-030-01234-2_1.

182. Luiten J., Osep A., Dendorfer P., Torr P., Geiger A., Leal-Taixé L., Leibe B. HOTA: A Higher Order Metric for Evaluating Multi-Object Tracking. International Journal of Computer Vision. 2021. Vol. 129, № 2. P. 548–578. DOI: https://doi.org/10.1007/s11263-020-01375-2.

183. Cohen J. Statistical Power Analysis for the Behavioral Sciences. 2nd ed. Hillsdale : Lawrence Erlbaum Associates, 1988. 567 p.

184. Friant Roulette : YouTube channel [Electronic resource]. – URL: https://www.youtube.com/@friantroulette. – Accessed: 06.09.2025.

185. Jackson Hole Wyoming USA Town Square Live Cam : YouTube [Electronic resource]. – URL: https://www.youtube.com/watch?v=1EiC9bvVGnk. – Accessed: 06.09.2025.

186. 4 Corners Camera Downtown City of Coldwater MI : YouTube [Electronic resource]. – URL: https://www.youtube.com/watch?v=ByED80IKdIU. – Accessed: 06.09.2025.

187. Hikvision. Intelligent Video Surveillance (IVS) Camera Products [Electronic resource]. – URL: https://www.hikvision.com/en/search/?q=ivs+camera&active=Products. – Accessed: 15.10.2025.

188. Hikvision. Intelligent Traffic Systems (ITS) Products [Electronic resource]. – URL: https://www.hikvision.com/europe/products/ITS-Products/. – Accessed: 15.10.2025.

189. Axis Communications. AXIS People Counter [Electronic resource]. – URL: https://www.axis.com/products/axis-people-counter. – Accessed: 15.10.2025.

190. Genetec. AutoVu : Unified Security Solution [Electronic resource]. – URL: https://www.genetec.com/products/unified-security/autovu. – Accessed: 15.10.2025.

191. Viisights. Traffic Monitoring and Urban Mobility : WISE Platform [Electronic resource]. – URL: https://www.viisights.com/products/wise/traffic-monitoring-and-urban-mobility/. – Accessed: 15.10.2025.

192. Citilog. Intelligent Incident Detection [Electronic resource]. – URL: https://www.citilog.com/. – Accessed: 15.10.2025.

ДОДАТКИ

Додаток А. Приклади виявлення аномалій на перехрестях

Рис. А.1. Приклад перевищення швидкості на перехресті вул. Шевченко-Келецька

Рис. А.2. Приклад тривалого стояння у зоні інтересу на перехресті вул. Шевченко-Келецька

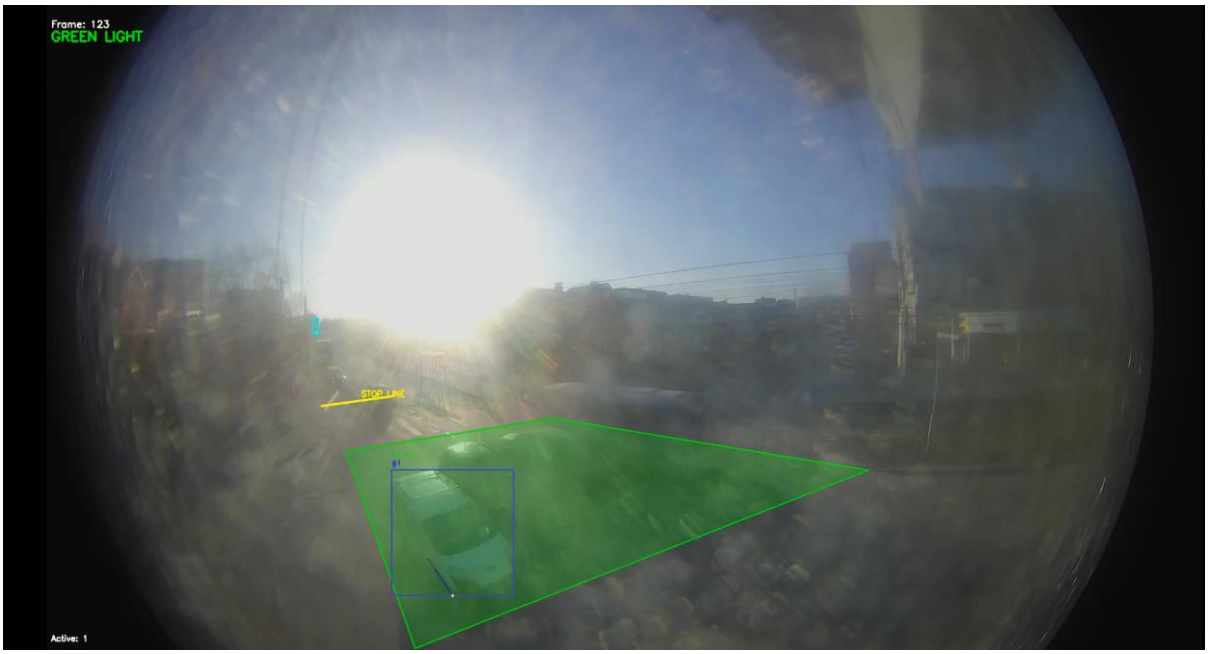


Рис. А.3. Приклад складних умов освітлення на перехресті вул. Шевченко-Келецька

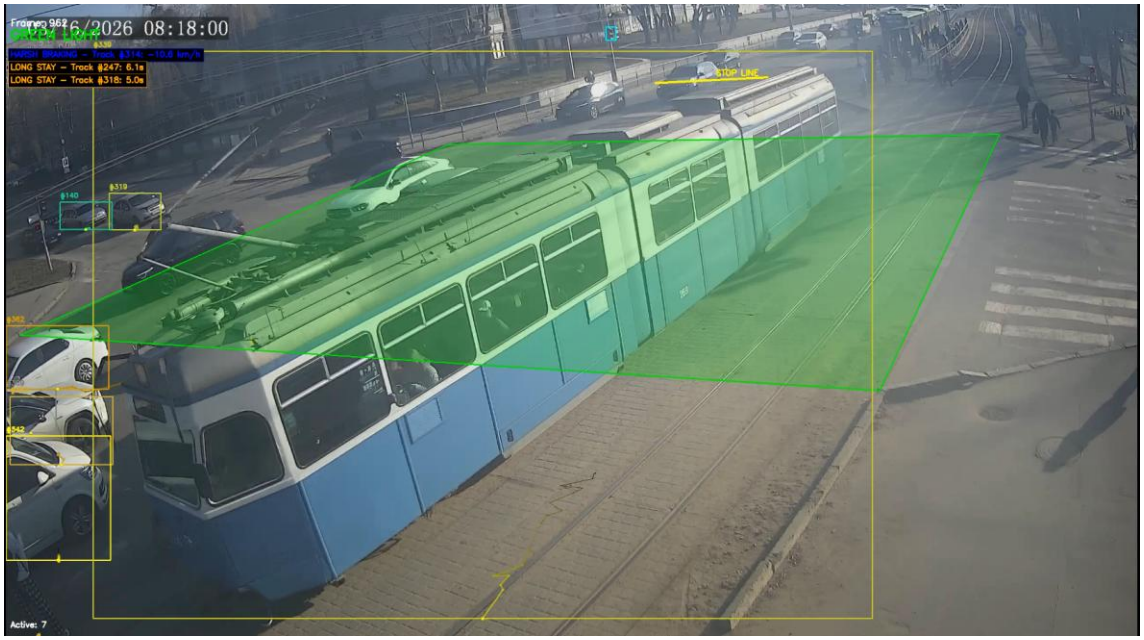


Рис. А.4. Приклад наявності декількох складних умов - засвітлення вранішнім сонцем, кутом огляду камери, перекриттям трамваєм зони інтересу на перехресті вул. Келецької-600-річчя

Додаток Б. Відомості про наукові публікації та апробацію матеріалів дисертаційного дослідження

Наукові праці, в яких опубліковані основні наукові результати дисертації:

1. Романець В., Маслій Р. В. Виявлення аномалій у відеопотоці трафіку транспорту засобами комп'ютерного зору та глибинного навчання // Вісник ВПІ. 2025. Вип. 5. С. 146–155. https://doi.org/10.31649/1997-9266-2025-182-5-146-155. ***Фахове видання категорії Б***

2. Романець В., Бісікало О. Метод багатооб'єктного трекінгу з обробкою оклюзій для систем відеоспостереження // Measuring and Computing Devices in Technological Processes. 2025. Т. 84. № 4. С. 440–445. https://doi.org/10.31891/2219-9365-2025-84-53. ***Фахове видання категорії Б***

3. Романець В.О., Бісікало О.В. Інформаційна технологія інтелектуального аналізу відеопотоку дорожнього руху на основі Edge-Cloud архітектури // Вісник Херсонського національного технічного університету. 2026. №1. С.304-310. https://doi.org/10.35546/kntu2078-4481.2026.1.41. ***Фахове видання категорії Б***

Наукові праці, що засвідчують апробацію матеріалів дисертації:

4. Романець В., Бісікало О. Виявлення об'єктів дорожнього руху з камер відеоспостереження // Herald of Khmelnytskyi National University. Technical Sciences. 2025. Т. 355, № 4. С. 491–497. https://doi.org/10.31891/2307-5732-2025-355-70. ***Не фахове видання категорії Б***

5. Романець В., Бісікало О. Information technology in customer service: use of computer vision and decision support systems // SWorldJournal. 2023. Т. 1, № 21-01. С. 65–72. https://doi.org/10.30888/2663-5712.2023-21-01-046. ***Періодичне закордонне видання***

6. Романець В., Бісікало О., Маслій Р. Адаптація сучасних моделей комп'ютерного зору (YOLOv8-nano) для роботи в реальному часі в системах

відеоаналітики дорожнього руху. ВНТКП ВНТУ. Факультет інтелектуальних інформаційних технологій та автоматизації. 2026. URL: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2026/paper/view/28227 (дата звернення: 25.04.2026).

7. Романець В., Бісікало О. Навчання модифікованої моделі YOLOv8 для детекції транспортних засобів на комбінованому датасеті. Молодь в науці: дослідження, проблеми, перспективи (МН-2026). 2026. URL: https://conferences.vntu.edu.ua/index.php/mn/mn2026/paper/view/28837 (дата звернення: 30.04.2026).

8. Романець В., Бісікало О. Edge-Cloud архітектура інформаційної технології аналізу відеопотоку дорожнього руху. // Scientific development in a changing world. Proceedings of the 5th International scientific and practical conference. SPC “Sci-conf.com.ua”. Lviv, Ukraine. 2026. Pp. 363–367. URL: https://sci-conf.com.ua/wp-content/uploads/2026/05/SCIENTIFIC-DEVELOPMENT-IN-A-CHANGING-WORLD-11-13.05.26.pdf#page=363 (дата звернення: 13.05.2026).

Додаток В. Акт про впровадження результатів дисертаційної роботи

АКТ ПРО ВПРОВАДЖЕННЯ РЕЗУЛЬТАТІВ ДИСЕРТАЦІЙНОЇ РОБОТИ

Вінницька міська рада
Департамент транспорту та міської мобільності
(повна назва організації)
Україна, 2150, м. Вінниця, вул. Соборна, 59
(адреса)

ЗАТВЕРДЖУЮ
Д
(підпис) (ПІБ)
2026 р.

АКТ
про впровадження результатів дисертаційної роботи
Романця Владислава Олеговича
«Інтелектуальна інформаційна технологія аналізу відеопотоку дорожнього руху»,
представленої до захисту на здобуття наукового ступеня доктора філософії
зі спеціальності 126 Інформаційні системи та технології

Комісія у складі директора департаменту транспорту та міської мобільності ВМР Сорокіна А. О. (голова комісії), заступника директора департаменту Бузниковатого С. В. та секретаря постійної комісії з питань промисловості, підприємництва, транспорту, зв'язку та сфери послуг ВМР Перлова В. Є. (члени комісії) склала цей акт про те, що дійсно до «Департаменту транспорту та міської мобільності Вінницької міської ради» передано такі результати дисертаційної роботи на здобуття доктора філософії аспіранта кафедри АІІТ ВНТУ Романця Владислава Олеговича:

Під час апробації у 2026 році науково-технічної продукції було проведено ряд експериментальних досліджень, які дозволяють стверджувати наступне:

1. Запропоновано інтелектуальну інформаційну технологію аналізу відеопотоку дорожнього руху, яка використовує детектор об'єктів YOLOv8 із інтеграцією архітектурних модифікацій C3Ghost-CBAM і реалізована у вигляді програмного комплексу мовою програмування Python із застосуванням бібліотек PyTorch та Ultralytics, що забезпечує динамічну адаптацію архітектури та параметрів нейронних мереж до варіації вхідних даних.
2. Розроблено програмне забезпечення, що включає модулі детекції об'єктів (на основі YOLOv8 з модифікацією C3Ghost-CBAM), трекінгу (ByteTrackPlus із використанням IoU-suppression, механізмів відновлення втрачених треків та DIoU-matching), а також модуль виявлення аномальних ситуацій, який забезпечує ідентифікацію чотирьох типів порушень дорожнього руху – проїзду на червоний сигнал, перевищення швидкості, аварійного гальмування та тривалого перебування в зоні перехрестя.
3. Розроблене програмне забезпечення запропонованої інформаційної технології може бути застосоване на edge-пристроях з обмеженими ресурсами після експорту моделі у формати ONNX та TensorRT з використанням INT8-квантування. Це дозволило

зменшити розмір моделі приблизно вдвічі (порівняно з FP16) та підтверджує практичну цінність розробки.

4. Проведено експеримент з використанням розробленого програмного забезпечення для аналітики на відеозаписах з кількох стаціонарних IP-камер міста Вінниці. Відеоматеріал загальним обсягом 6 годин на камеру (по 720 фрагментів тривалістю 30 секунд, записаних у ранішній та обідній час доби в різні дні) охоплював 3 попередньо визначених зони спостереження:

   1) Перехрестя вул. Келецької та вул. 600-річчя – досліджувався проїзд на червоний сигнал світлофора; зафіксовано 25 аномальних подій, більшість з яких відбулася при одночасному горінні жовтого та червоного сигналів (кінець заборонної фази циклу); значення MCC = 0.796.

   2) Перехрестя вул. Келецької та вул. Шевченка – досліджувалися перевищення швидкості (MCC = 0.646, 16 аномальних подій) та тривале перебування у зоні перехрестя (MCC = 0.828, 49 подій); для спостереження використано fisheye-камеру з попередньою корекцією радіального спотворення.

   3) Пішохідний перехід до Лісопарку / Хмельницьке шосе (продовження проспекту Космонавтів) – досліджувався розподіл швидкостей транспортного потоку; встановлено, що 22 % транспортних засобів рухалися зі швидкістю менше 30 км/год (ефект нерегульованого пішохідного переходу), тоді як 33 % перевищували дозволену швидкість 50 км/год.

   Відеофрагменти надано з IP-камер різних виробників у різний час доби та при різних погодних умовах і рівнях освітлення, що забезпечило варіативність тестових даних і демонструє стабільну роботу запропонованої інформаційної технології в умовах різних сценаріїв.

5. Проведено порівняльний аналіз запропонованої інформаційної технології з впровадженими комерційними рішеннями (Hikvision, Genetec, Viisights), за результатами якого встановлено, що розроблена технологія характеризується відкритою архітектурою, можливістю виявлення динамічних аномалій (аварійне гальмування, тривале перебування в ROI), підтримкою edge-інференсу та суттєво нижчою вартістю впровадження, що підтверджує її практичну цінність.

Проаналізовано ефективність наведених результатів та рекомендовано їх впровадити у роботу «Департаменту транспорту та міської мобільності Вінницької міської ради» під час розв'язання задач автоматизованого моніторингу дорожнього руху та збору статистики транспортних потоків.

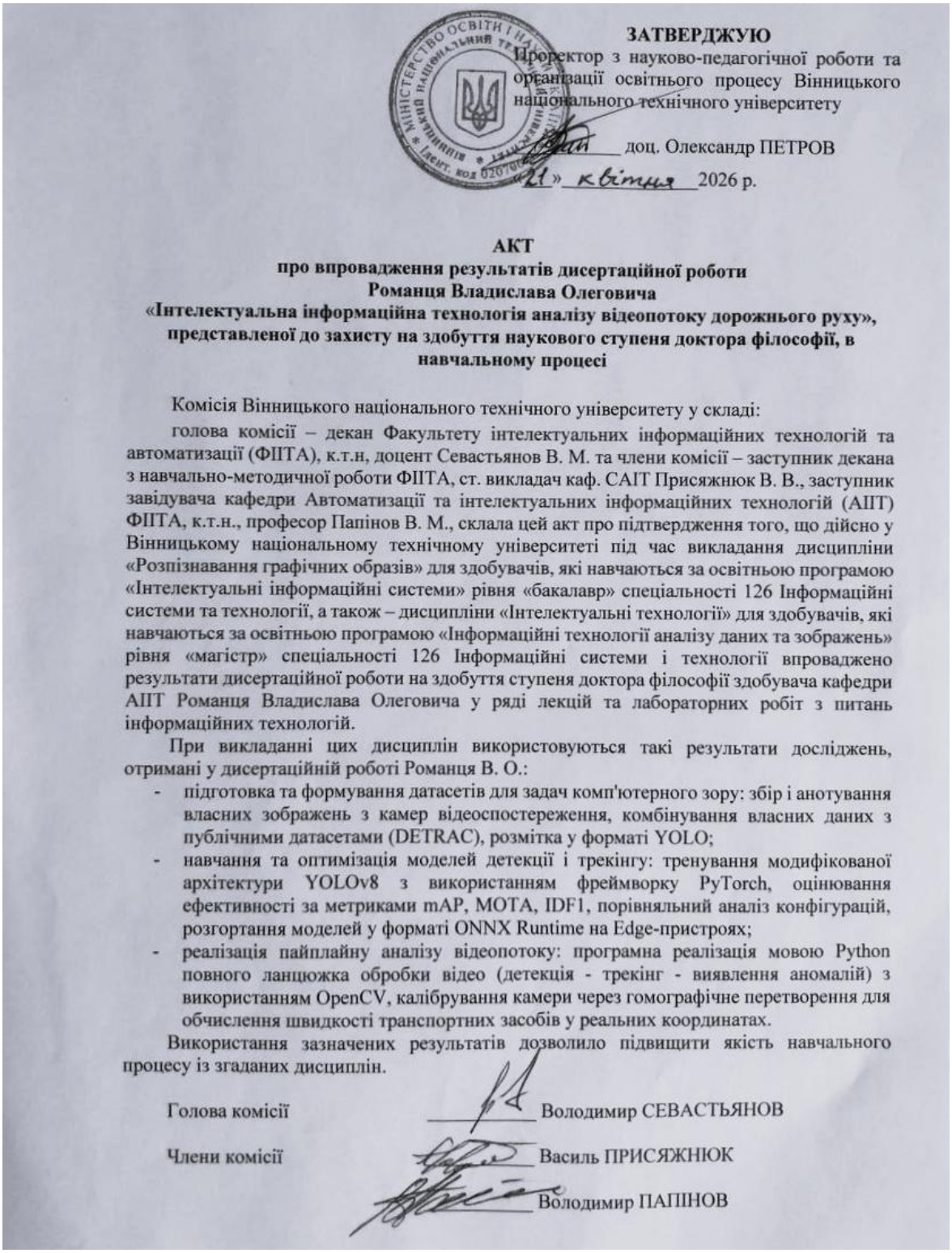
Даний акт не є основою для проведення фінансових взаєморозрахунків.

Голова комісії _____ Сорокін А. О.

Члени комісії _____ Бузниковатий С. В.

_____ Перлов В. Є.

Додаток Г. Акт про впровадження результатів у навчальний процес ВНТУ



ЗАТВЕРДЖУЮ
Проректор з науково-педагогічної роботи та організації освітнього процесу Вінницького національного технічного університету

_____ доц. Олександр ПЕТРОВ
«21» квітня 2026 р.

АКТ
про впровадження результатів дисертаційної роботи
Романця Владислава Олеговича
«Інтелектуальна інформаційна технологія аналізу відеопотоку дорожнього руху», представленої до захисту на здобуття наукового ступеня доктора філософії, в навчальному процесі

Комісія Вінницького національного технічного університету у складі:

голова комісії – декан Факультету інтелектуальних інформаційних технологій та автоматизації (ФІТА), к.т.н, доцент Севастьянов В. М. та члени комісії – заступник декана з навчально-методичної роботи ФІТА, ст. викладач каф. САІТ Присяжнюк В. В., заступник завідувача кафедри Автоматизації та інтелектуальних інформаційних технологій (АІТ) ФІТА, к.т.н., професор Папінов В. М., склала цей акт про підтвердження того, що дійсно у Вінницькому національному технічному університеті під час викладання дисципліни «Розпізнавання графічних образів» для здобувачів, які навчаються за освітньою програмою «Інтелектуальні інформаційні системи» рівня «бакалавр» спеціальності 126 Інформаційні системи та технології, а також – дисципліни «Інтелектуальні технології» для здобувачів, які навчаються за освітньою програмою «Інформаційні технології аналізу даних та зображень» рівня «магістр» спеціальності 126 Інформаційні системи і технології впроваджено результати дисертаційної роботи на здобуття ступеня доктора філософії здобувача кафедри АІТ Романця Владислава Олеговича у ряді лекцій та лабораторних робіт з питань інформаційних технологій.

При викладанні цих дисциплін використовуються такі результати досліджень, отримані у дисертаційній роботі Романця В. О.:

- підготовка та формування датасетів для задач комп'ютерного зору: збір і анотування власних зображень з камер відеоспостереження, комбінування власних даних з публічними датасетами (DETRAC), розмітка у форматі YOLO;
- навчання та оптимізація моделей детекції і трекінгу: тренування модифікованої архітектури YOLOv8 з використанням фреймворку PyTorch, оцінювання ефективності за метриками mAP, MOTA, IDF1, порівняльний аналіз конфігурацій, розгортання моделей у форматі ONNX Runtime на Edge-пристроях;
- реалізація пайплайну аналізу відеопотоку: програмна реалізація мовою Python повного ланцюжка обробки відео (детекція - трекінг - виявлення аномалій) з використанням OpenCV, калібрування камери через гомографічне перетворення для обчислення швидкості транспортних засобів у реальних координатах.

Використання зазначених результатів дозволило підвищити якість навчального процесу із згаданих дисциплін.

Голова комісії _____ Володимир СЕВАСТЬЯНОВ

Члени комісії _____ Василь ПРИСЯЖНЮК

_____ Володимир ПАПІНОВ

Додаток Д. Лістинг програми трекінгу об'єктів методу ByteTrackPlus

```
"""
ByteTrackPlus v2
"""

import numpy as np
from collections import deque
import copy
import time

from .kalman_filter import KalmanFilter
from . import matching
from .basetrack import BaseTrack, TrackState

class STrack(BaseTrack):
    shared_kalman = KalmanFilter()

    def __init__(self, tlwh, score):
        self._tlwh = np.asarray(tlwh, dtype=np.float64)
        self.kalman_filter = None
        self.mean, self.covariance = None, None
        self.is_activated = False
        self.score = score
        self.tracklet_len = 0
        self._occlusion_attrs_initialized = False

    def predict(self):
        mean_state = self.mean.copy()
        if self.state != TrackState.Tracked:
            mean_state[7] = 0
        self.mean, self.covariance = self.kalman_filter.predict(mean_state, self.covariance)

    def activate(self, kalman_filter, frame_id):
```

```
        self.kalman_filter = kalman_filter
        self.track_id = self.next_id()
        self.mean, self.covariance = self.kalman_filter.initiate(self.tlwh_to_xyah(self._tlwh))
        self.tracklet_len = 0
        self.state = TrackState.Tracked
        if frame_id == 1:
            self.is_activated = True
        self.frame_id = frame_id
        self.start_frame = frame_id

    @staticmethod
    def multi_predict(stracks):
        if len(stracks) > 0:
            multi_mean = np.asarray([st.mean.copy() for st in stracks])
            multi_covariance = np.asarray([st.covariance for st in stracks])
            for i, st in enumerate(stracks):
                if st.state != TrackState.Tracked:
                    multi_mean[i][7] = 0
            multi_mean, multi_covariance = STrack.shared_kalman.multi_predict(multi_mean,
multi_covariance)
            for i, (mean, cov) in enumerate(zip(multi_mean, multi_covariance)):
                stracks[i].mean = mean
                stracks[i].covariance = cov

    def activate(self, kalman_filter, frame_id):
        self.kalman_filter = kalman_filter
        self.track_id = self.next_id()
        self.mean, self.covariance = self.kalman_filter.initiate(self.tlwh_to_xyah(self._tlwh))
        self.tracklet_len = 0
        self.state = TrackState.Tracked
        if frame_id == 1:
            self.is_activated = True
        self.frame_id = frame_id
        self.start_frame = frame_id
```

```
def re_activate(self, new_track, frame_id, new_id=False):
    self.mean, self.covariance = self.kalman_filter.update(
        self.mean, self.covariance, self.tlwh_to_xyah(new_track.tlwh)
    )
    self.tracklet_len = 0
    self.state = TrackState.Tracked
    self.is_activated = True
    self.frame_id = frame_id
    if new_id:
        self.track_id = self.next_id()
    self.score = new_track.score

def update(self, new_track, frame_id):
    self.frame_id = frame_id
    self.tracklet_len += 1
    new_tlwh = new_track.tlwh
    self.mean, self.covariance = self.kalman_filter.update(
        self.mean, self.covariance, self.tlwh_to_xyah(new_tlwh))
    self.state = TrackState.Tracked
    self.is_activated = True
    self.score = new_track.score



@property
def tlwh(self):
    if self.mean is None:
        return self._tlwh.copy()
    ret = self.mean[:4].copy()
    ret[2] *= ret[3]
    ret[:2] -= ret[2:] / 2
    return ret
```

```
@property
def tlbr(self):
    ret = self.tlwh.copy()
    ret[2:] += ret[:2]
    return ret

@staticmethod
def tlwh_to_xyah(tlwh):
    ret = np.asarray(tlwh).copy()
    ret[:2] += ret[2:] / 2
    ret[2] /= ret[3]
    return ret

def to_xyah(self):
    return self.tlwh_to_xyah(self.tlwh)

@staticmethod
def tlbr_to_tlwh(tlbr):
    ret = np.asarray(tlbr).copy()
    ret[2:] -= ret[:2]
    return ret

@staticmethod
def tlwh_to_tlbr(tlwh):
    ret = np.asarray(tlwh).copy()
    ret[2:] += ret[:2]
    return ret

def init_occlusion_attrs(self):
    if not self._occlusion_attrs_initialized:
        self.is_occluded = False
        self.occluded_len = 0
        self.last_occluded_frame = 0
        self.was_recently_occluded = False
```

```
            self._occlusion_attrs_initialized = True

    def __repr__(self):
        return 'OT_{}_({}-{})'.format(self.track_id, self.start_frame, self.end_frame)


class ByteTrackPlus(object):
    def __init__(self, args, config, frame_rate=30):
        self.tracked_stracks = []
        self.lost_stracks = []
        self.removed_stracks = []
        self.frame_id = 0
        self.args = args

        # ByteTrack базові параметри
        self.det_thresh = args.track_thresh + 0.1
        self.buffer_size = int(frame_rate / 30.0 * args.track_buffer)
        self.max_time_lost = self.buffer_size
        self.kalman_filter = KalmanFilter()

        # Розширення - тільки 3 компоненти
        self.config = config
        self.use_iou_suppression = config.get("use_iou_suppression", False)
        self.init_iou_suppress = config.get("init_iou_suppress", 0.65)
        self.use_occlusion_grace = config.get("use_occlusion_grace", False)
        self.grace_period = config.get("grace_period", 25)
        self.use_diou = config.get("use_diou", False)

        self.needs_occlusion_attrs = self.use_occlusion_grace

        self._print_config()

    def update(self, output_results, img_info, img_size):
        self.frame_id += 1
```

```
        activated_starcks = []
        refind_stracks = []
        lost_stracks = []
        removed_stracks = []

        if output_results.shape[1] == 5:
            scores = output_results[:, 4]
            bboxes = output_results[:, :4]
        else:
            output_results = output_results.cpu().numpy()
            scores = output_results[:, 4] * output_results[:, 5]
            bboxes = output_results[:, :4]

        img_h, img_w = img_info[0], img_info[1]
        scale = min(img_size[0] / float(img_h), img_size[1] / float(img_w))
        bboxes /= scale

        remain_inds = scores > self.args.track_thresh
        inds_low = scores > 0.1
        inds_high = scores < self.args.track_thresh
        inds_second = np.logical_and(inds_low, inds_high)

        dets = bboxes[remain_inds]
        dets_second = bboxes[inds_second]
        scores_keep = scores[remain_inds]
        scores_second = scores[inds_second]

        detections = [STrack(STrack.tlbr_to_tlwh(tlbr), s) for (tlbr, s) in zip(dets, scores_keep)] if len(dets) > 0 else []

        unconfirmed = []
        tracked_stracks = []
        for track in self.tracked_stracks:
            if not track.is_activated:
```

```
                unconfirmed.append(track)
            else:
                tracked_stracks.append(track)

        strack_pool = joint_stracks(tracked_stracks, self.lost_stracks)
        STrack.multi_predict(strack_pool)

        if self.use_diou:
            dists = self._diou_distance(strack_pool, detections)
        else:
            dists = matching.iou_distance(strack_pool, detections)

        if not self.args.mot20:
            dists = matching.fuse_score(dists, detections)

        matches, u_track, u_detection = matching.linear_assignment(dists,
thresh=self.args.match_thresh)

        for itracked, idet in matches:
            track = strack_pool[itracked]
            det = detections[idet]
            if track.state == TrackState.Tracked:
                track.update(detections[idet], self.frame_id)
                activated_starcks.append(track)
            else:
                track.re_activate(det, self.frame_id, new_id=False)
                refind_stracks.append(track)

        if len(dets_second) > 0:
            detections_second = [STrack(STrack.tlbr_to_tlwh(tlbr), s) for (tlbr, s) in
zip(dets_second, scores_second)]
        else:
            detections_second = []
```

```
        r_tracked_stracks = [strack_pool[i] for i in u_track if strack_pool[i].state == TrackState.Tracked]

        if self.use_diou:
            dists = self._diou_distance(r_tracked_stracks, detections_second)
        else:
            dists = matching.iou_distance(r_tracked_stracks, detections_second)

        matches, u_track, u_detection_second = matching.linear_assignment(dists, thresh=0.5)

        for itracked, idet in matches:
            track = r_tracked_stracks[itracked]
            det = detections_second[idet]
            if track.state == TrackState.Tracked:
                track.update(det, self.frame_id)
                activated_starcks.append(track)
            else:
                track.re_activate(det, self.frame_id, new_id=False)
                refind_stracks.append(track)

        for it in u_track:
            track = r_tracked_stracks[it]

            if self.needs_occlusion_attrs:
                track.init_occlusion_attrs()

            if not track.state == TrackState.Lost:
                track.mark_lost()
                lost_stracks.append(track)

        detections = [detections[i] for i in u_detection]
        dists = matching.iou_distance(unconfirmed, detections)
        if not self.args.mot20:
            dists = matching.fuse_score(dists, detections)
```

```
            matches, u_unconfirmed, u_detection = matching.linear_assignment(dists, thresh=0.7)

            for itracked, idet in matches:
                unconfirmed[itracked].update(detections[idet], self.frame_id)
                activated_starcks.append(unconfirmed[itracked])

            for it in u_unconfirmed:
                track = unconfirmed[it]
                track.mark_removed()
                removed_stracks.append(track)

            if self.use_iou_suppression:
                new_tracks = self._init_with_suppression(detections, u_detection, activated_starcks)
            else:
                new_tracks = []
                for inew in u_detection:
                    track = detections[inew]
                    if track.score < self.det_thresh:
                        continue
                    track.activate(self.kalman_filter, self.frame_id)
                    new_tracks.append(track)

            activated_starcks.extend(new_tracks)

            for track in self.lost_stracks:
                if self.use_occlusion_grace:
                    if (track._occlusion_attrs_initialized and
                        track.was_recently_occluded and
                        (self.frame_id - track.last_occluded_frame <= self.grace_period)):
                        continue

                if self.frame_id - track.end_frame > self.max_time_lost:
                    track.mark_removed()
```

```
                removed_stracks.append(track)

            self.tracked_stracks = [t for t in self.tracked_stracks if t.state == TrackState.Tracked]
            self.tracked_stracks = joint_stracks(self.tracked_stracks, activated_starcks)
            self.tracked_stracks = joint_stracks(self.tracked_stracks, refind_stracks)
            self.lost_stracks = sub_stracks(self.lost_stracks, self.tracked_stracks)
            self.lost_stracks.extend(lost_stracks)
            self.lost_stracks = sub_stracks(self.lost_stracks, self.removed_stracks)
            self.removed_stracks.extend(removed_stracks)
            self.tracked_stracks, self.lost_stracks = remove_duplicate_stracks(self.tracked_stracks, self.lost_stracks)

            output_stracks = [track for track in self.tracked_stracks if track.is_activated]
            return output_stracks

        def _print_config(self):
            """Компактний вивід конфігурації"""
            components = [
                ("IoU Supp", self.use_iou_suppression),
                ("Grace", self.use_occlusion_grace),
                ("DIoU", self.use_diou),
            ]

            active = [name for name, enabled in components if enabled]

            print(f"[Tracker v2] thresh={self.args.track_thresh:.2f}, match={self.args.match_thresh:.2f}, "
                  f"buffer={self.args.track_buffer} | Components: {', '.join(active) if active else 'None (Baseline)'}")

        # Extension methods
        def _diou_distance(self, tracks, detections):
            if len(tracks) == 0 or len(detections) == 0:
                return np.zeros((len(tracks), len(detections)))
```

```
    track_boxes = np.array([track.tlbr for track in tracks])
    det_boxes = np.array([STrack.tlwh_to_tlbr(det.tlwh) for det in detections])

    ious = self._batch_iou(track_boxes, det_boxes)

    track_cx = (track_boxes[:, 0] + track_boxes[:, 2]) * 0.5
    track_cy = (track_boxes[:, 1] + track_boxes[:, 3]) * 0.5
    det_cx = (det_boxes[:, 0] + det_boxes[:, 2]) * 0.5
    det_cy = (det_boxes[:, 1] + det_boxes[:, 3]) * 0.5

    dx = track_cx[:, None] - det_cx[None, :]
    dy = track_cy[:, None] - det_cy[None, :]
    center_dist_sq = dx * dx + dy * dy

    x1_enc = np.minimum(track_boxes[:, None, 0], det_boxes[None, :, 0])
    y1_enc = np.minimum(track_boxes[:, None, 1], det_boxes[None, :, 1])
    x2_enc = np.maximum(track_boxes[:, None, 2], det_boxes[None, :, 2])
    y2_enc = np.maximum(track_boxes[:, None, 3], det_boxes[None, :, 3])

    diag_sq = (x2_enc - x1_enc) ** 2 + (y2_enc - y1_enc) ** 2
    dious = ious - (center_dist_sq / (diag_sq + 1e-7))

    return 1 - dious

def _batch_iou(self, boxes1, boxes2):
    area1 = (boxes1[:, 2] - boxes1[:, 0]) * (boxes1[:, 3] - boxes1[:, 1])
    area2 = (boxes2[:, 2] - boxes2[:, 0]) * (boxes2[:, 3] - boxes2[:, 1])

    inter_w = np.maximum(0, np.minimum(boxes1[:, None, 2], boxes2[None, :, 2]) -
                         np.maximum(boxes1[:, None, 0], boxes2[None, :, 0]))
    inter_h = np.maximum(0, np.minimum(boxes1[:, None, 3], boxes2[None, :, 3]) -
                         np.maximum(boxes1[:, None, 1], boxes2[None, :, 1]))
    inter_area = inter_w * inter_h
```

```
        union_area = area1[:, None] + area2[None, :] - inter_area
        return inter_area / (union_area + 1e-7)

    def _init_with_suppression(self, detections, u_detection, active_tracks):
        new_tracks = []
        if len(u_detection) == 0 or len(active_tracks) == 0:
            for inew in u_detection:
                track = detections[inew]
                if track.score < self.det_thresh:
                    continue
                track.activate(self.kalman_filter, self.frame_id)
                new_tracks.append(track)
            return new_tracks
        det_boxes = np.array([STrack.tlwh_to_tlbr(detections[i].tlwh) for i in u_detection])
        active_boxes = np.array([t.tlbr for t in active_tracks])

        ious = self._batch_iou(det_boxes, active_boxes)
        max_ious = ious.max(axis=1)

        for idx, inew in enumerate(u_detection):
            track = detections[inew]
            if track.score < self.det_thresh:
                continue

            if max_ious[idx] < self.init_iou_suppress:
                track.activate(self.kalman_filter, self.frame_id)
                new_tracks.append(track)
        return new_tracks

def joint_stracks(tlista, tlistb):
    exists = {}
    res = []
    for t in tlista:
        exists[t.track_id] = 1
```

```
            res.append(t)
    for t in tlistb:
        tid = t.track_id
        if not exists.get(tid, 0):
            exists[tid] = 1
            res.append(t)
    return res


def sub_stracks(tlista, tlistb):
    stracks = {}
    for t in tlista:
        stracks[t.track_id] = t
    for t in tlistb:
        tid = t.track_id
        if stracks.get(tid, 0):
            del stracks[tid]
    return list(stracks.values())


def remove_duplicate_stracks(stracksa, stracksb):
    pdist = matching.iou_distance(stracksa, stracksb)
    pairs = np.where(pdist < 0.15)
    dupa, dupb = list(), list()
    for p, q in zip(*pairs):
        timep = stracksa[p].frame_id - stracksa[p].start_frame
        timeq = stracksb[q].frame_id - stracksb[q].start_frame
        if timep > timeq:
            dupb.append(q)
        else:
            dupa.append(p)
    resa = [t for i, t in enumerate(stracksa) if not i in dupa]
    resb = [t for i, t in enumerate(stracksb) if not i in dupb]
    return resa, resb
```