

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»**

**Навчально-науковий інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу**

До захисту допущено:
Завідувач кафедри
_____ Оксана ТИМОЩУК
« ____ » _____ 2026 р.

Дипломна робота

на здобуття ступеня бакалавра

**за освітньо-професійною програмою «Системний аналіз і управління»
спеціальності 124 «Системний аналіз»**

**на тему: «Інтелектуальна система відеомоніторингу стану водія
автотранспорту для підвищення безпеки дорожнього руху»**

Виконав:

студент IV курсу, групи КА-23
Кудрявцев Іван Михайлович _____

Керівник:

асистент кафедри ММСА, доктор філософії
Титаренко Андрій Миколайович _____

Консультант з економічного розділу:

доцент кафедри ЕК, к.е.н., доц.
Рощина Надія Василівна _____

Консультант з нормоконтролю:

доцент кафедри ММСА, к.ф.-м.н.
Статкевич Віталій Михайлович _____

Рецензент:

ст. викладач кафедри мережних технологій НаУКМА,
к.ф.-м.н. Волинець Євгеній Анатолійович _____

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.

Студент _____

Київ — 2026 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Навчально-науковий інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

Рівень вищої освіти — перший (бакалаврський)
Спеціальність — 124 «Системний аналіз»
Освітня програма «Системний аналіз і управління»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Оксана ТИМОЩУК
«___» _____ 2026 р.

ЗАВДАННЯ
на дипломну роботу студенту
Кудрявцеву Івану Михайловичу

1. Тема роботи: «Інтелектуальна система відеомоніторингу стану водія автотранспорту для підвищення безпеки дорожнього руху», керівник роботи Титаренко Андрій Миколайович, асистент кафедри ММСА, доктор філософії, затверджені наказом по університету від 27.05.2026 р. № 1944-с.
2. Термін подання студентом роботи: «10» червня 2026 р.
3. Вихідні дані до роботи: науково-методична література за темою роботи; відкритий датасет зображень очей водіїв з платформи Roboflow; готові класифікатори Haar Cascade з бібліотеки OpenCV; програмні засоби Python, OpenCV, Ultralytics.
4. Зміст роботи: дослідження предметної області та огляд існуючих систем моніторингу стану водія; теоретичний апарат класичних та нейромережових методів комп'ютерного зору; програмна реалізація системи на основі гібридної архітектури Haar Cascade + YOLOv8 з навчанням моделі та розробкою алгоритму накопичення індексу сонливості; функціонально-вартісний аналіз програмного продукту.
5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): блок-схема алгоритму роботи програмного продукту; графіки функцій втрат і метрик якості протягом навчання моделі; нормалізована матриця плутанини; скріншоти інтерфейсу системи у різних режимах роботи; приклади зображень з кастомного датасету; презентація.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Функціонально-вартісний аналіз	Рощина Н.В., доц., к.е.н.		

7. Дата видачі завдання: «15» квітня 2026 р.

Календарний план

№ з/п	Назва етапів виконання бакалаврської дипломної роботи (БДР)	Термін виконання етапів роботи	Примітка
1	Затвердження теми БДР, опанування вимог ДСТУ 3008:2015	20.04.2026–29.04.2026	виконано
2	Огляд літератури з методів комп'ютерного зору, аналізу стану водія та виявлення сонливості	30.04.2026–11.05.2026	виконано
3	Аналіз існуючих комерційних DMS-систем, обґрунтування вибору гібридної архітектури Haar Cascade + YOLOv8	12.05.2026–19.05.2026	виконано
4	Формування гібридного навчального датасету: напівавтоматичний збір власних зображень та їх автоматична розмітка	20.05.2026–27.05.2026	виконано
5	Навчання моделі YOLOv8n із застосуванням domain adaptation на Google Colab, аналіз метрик якості	28.05.2026–01.06.2026	виконано
6	Реалізація двошарового алгоритму накопичення індексу сонливості та інтеграція з модулем детекції Haar Cascade мовою Python	02.06.2026–04.06.2026	виконано
7	Оформлення пояснювальної записки, перевірка структури, висновків, джерел і додатків, підготовка презентації	05.06.2026–16.06.2026	виконано
8	Передзахист БДР	08.06.2026	виконано

Студент _____

Іван КУДРЯВЦЕВ

Керівник _____

Андрій ТИТАРЕНКО

РЕФЕРАТ

Дипломна робота: 101 с., 15 рис., 16 табл., 2 додатки, 24 джерела.

ВІДЕОМОНІТОРИНГ, ДЕТЕКЦІЯ СОНЛИВОСТІ, HAAR CASCADE, YOLOV8, КОМП'ЮТЕРНИЙ ЗІР, НЕЙРОННА МЕРЕЖА, DOMAIN ADAPTATION, PYTHON

Об'єкт дослідження — процес виявлення ознак сонливості водія транспортного засобу за відеопотоком у реальному часі.

Предмет дослідження — методи комп'ютерного зору, архітектури нейронних мереж та алгоритми обробки відеопотоку для моніторингу стану водія.

Мета роботи — розроблення інтелектуальної системи відеомоніторингу стану водія на основі гібридної архітектури OpenCV та YOLOv8, що функціонує в режимі реального часу на стандартній веб-камері без спеціалізованого апаратного забезпечення.

Як результат роботи, проаналізовано дорожньо-транспортну статистику, пов'язану з втомою водіїв, та визначено ключові візуальні маркери сонливості. Розроблено програмний продукт мовою програмування Python, що реалізує двоетапний пайплайн детекції: каскади Хаара локалізують обличчя і зону очей, а нейронна мережа YOLOv8n класифікує стан кожного ока. Сформовано гібридний навчальний датасет та проведено навчання моделі із застосуванням domain adaptation. Реалізовано двохаровий алгоритм накопичення індексу сонливості, який забезпечує стійкість системи до поодиноких хибних класифікацій.

Практична цінність роботи полягає в тому, що система розгортається на стандартній веб-камері та персональному комп'ютері без спеціалізованих сенсорів, що суттєво знижує поріг впровадження систем моніторингу стану водія у малому та середньому транспортному бізнесі.

ABSTRACT

Bachelor's thesis: 101 p., 15 fig., 16 tables, 2 appendices, 24 references.

VIDEO MONITORING, DROWSINESS DETECTION, HAAR CASCADE, YOLOV8, COMPUTER VISION, NEURAL NETWORK, DOMAIN ADAPTATION, PYTHON

The object of research is the process of detecting signs of driver drowsiness from a video stream in real time.

The subject of research comprises computer vision methods, neural network architectures and video-stream processing algorithms for monitoring the driver state.

The aim of the work is the development of an intelligent driver-state video-monitoring system based on a hybrid OpenCV and YOLOv8 architecture that operates in real time on a standard web camera without specialised hardware.

As a result, the road-accident statistics related to driver fatigue were analysed and the key visual markers of drowsiness were identified. A software product was developed in the Python programming language that implements a two-stage detection pipeline: Haar cascades localise the face and the eye region, while a YOLOv8n neural network classifies the state of each eye. A hybrid training dataset was assembled, and the model was trained using domain adaptation. A two-layer drowsiness-index accumulation algorithm was implemented, providing robustness against isolated misclassifications.

The practical value of the thesis is that the system can be deployed on a standard web camera and a personal computer without dedicated sensors, which significantly lowers the entry barrier for adopting driver-monitoring systems in small and medium transport businesses.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	9
ВСТУП	11
1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ	14
1.1 Втома водіїв як чинник дорожньо-транспортної небезпеки	14
1.2 Огляд існуючих систем моніторингу стану водія	17
1.2.1 Комерційні DMS-системи	17
1.2.2 Порівняльний аналіз та спільні обмеження	19
1.3 Підходи до автоматичного виявлення ознак сонливості	20
1.4 Формалізація задачі	22
Висновки до розділу 1	24
2 ТЕОРЕТИЧНИЙ АПАРАТ.....	25
2.1 Методи комп'ютерного зору для аналізу стану очей.....	25
2.1.1 Класичні методи: каскади Хаара та метод EAR	25
2.1.2 Методи на основі глибинного навчання: CNN та YOLO.....	27
2.1.3 Порівняльний аналіз методів	28
2.2 Математичний апарат детектора Viola–Jones	29
2.2.1 Інтегральне зображення та обчислення ознак Хаара	29
2.2.2 Алгоритм AdaBoost і каскадна структура класифікатора.....	31
2.3 Метод Eye Aspect Ratio та його обмеження	32
2.4 Архітектура нейромережі YOLOv8.....	33
2.4.1 Структура backbone–neck–head та anchor-free detection.....	33
2.4.2 Обґрунтування вибору варіанту YOLOv8n	35

	7
2.5 Алгоритм накопичення індексу сонливості у часі.....	36
2.6 Метрики оцінки якості системи.....	38
Висновки до розділу 2	39
3 ПРОГРАМНА РЕАЛІЗАЦІЯ.....	41
3.1 Технічні вимоги до системи.....	41
3.2 Вибір мови та середовища програмування. Опис бібліотек.....	42
3.3 Формування навчального датасету	44
3.3.1 Базовий датасет	44
3.3.2 Конвеєр напівавтоматичного збору кастомних даних	45
3.3.3 Аугментація та об'єднання датасетів.....	48
3.4 Навчання моделі та domain adaptation.....	49
3.5 Опис алгоритму роботи програмного продукту	51
3.6 Результати та аналіз	55
Висновки до розділу 3	60
4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ	
.....	61
4.1 Постановка задачі проєктування	61
4.2 Обґрунтування функцій програмного продукту.....	62
4.3 Обґрунтування системи параметрів програмного продукту	66
4.4 Аналіз експертного оцінювання параметрів	69
4.5 Аналіз рівня якості варіантів реалізації функцій.....	73
4.6 Економічний аналіз варіантів розробки ПП.....	75
4.6.1 Розрахунок трудомісткості.....	75
4.6.2 Розрахунок заробітної плати та відрахувань.....	77
4.6.3 Витрати на машинний час	78

	8
4.6.4 Повна вартість розробки	80
4.7 Вибір кращого варіанту ПП за техніко-економічним рівнем.....	80
Висновки до розділу 4	81
ВИСНОВКИ.....	83
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ	85
ДОДАТОК А ЛІСТИНГ ПРОГРАМНОГО ПРОДУКТУ	88
ДОДАТОК Б ПРЕЗЕНТАЦІЯ	93

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

CNN — Convolutional Neural Network, згорткова нейронна мережа.

DMS — Driver Monitoring System, система моніторингу стану водія.

EAR — Eye Aspect Ratio, відношення аспектів ока.

FPS — Frames Per Second, кадрів на секунду.

mAP — mean Average Precision, середня точність за всіма класами.

PERCLOS — Percentage of Eye Closure, відсоток часу закриття повік.

СППР — система підтримки прийняття рішень.

ІІ — штучний інтелект.

YOLOv8 — You Only Look Once version 8, архітектура нейромережі для детекції об'єктів.

ЄСВ — єдиний соціальний внесок.

ЗП — заробітна плата.

ІЧ — інфрачервоний (діапазон випромінювання).

НДІ — науково-дослідна інформація.

НКРЕКП — Національна комісія, що здійснює державне регулювання у сферах енергетики та комунальних послуг.

ПП — програмний продукт.

ФВА — функціонально-вартісний аналіз.

ЕОМ — електронно-обчислювальна машина.

ADAS — Advanced Driver Assistance Systems, системи допомоги водієві.

COCO — Common Objects in Context, відкритий датасет розмічених зображень.

CPU — Central Processing Unit, центральний процесор.

CSP — Cross Stage Partial, тип згорткового блоку в YOLOv8.

DA — domain adaptation, доменна адаптація моделі.

GPU — Graphics Processing Unit, графічний процесор.

HOG — Histogram of Oriented Gradients, гістограма орієнтованих градієнтів.

IoU — Intersection over Union, перетин над об'єднанням.

LDWS — Lane Departure Warning System, система попередження про вихід зі смуги.

MTCNN — Multi-task Cascaded Convolutional Networks, багатозадачна каскадна згорткова мережа.

RGB — Red, Green, Blue, тристимульна модель кольору.

TP, FP, FN — True Positive, False Positive, False Negative — правдиво позитивні, хибно позитивні та хибно негативні класифікації.

ВСТУП

За даними Всесвітньої організації охорони здоров'я, щороку у світі внаслідок дорожньо-транспортних пригод гине близько 1,35 млн людей, а від 20 до 50 млн отримують травми різного ступеня тяжкості [1]. Значна частка цих пригод спричинена людським фактором, серед якого окреме місце посідають втома і сонливість водія. За різними оцінками, від 10 до 30% серйозних аварій пов'язані саме зі станом втоми. Особливу небезпеку становить мікросон — короткі, тривалістю від 2 до 30 секунд, неусвідомлені засинання, під час яких водій повністю втрачає контроль над транспортним засобом. За одну секунду такого стану автомобіль, що рухається зі швидкістю 90 км/год, долає близько 25 метрів некерованого шляху.

Наявні комерційні системи моніторингу стану водія (Driver Monitoring System, DMS) — такі як рішення компаній Seeing Machines, Mobileye та Bosch — здатні надійно виявляти ознаки сонливості, проте потребують дорогого спеціалізованого обладнання: інфрачервоних камер, цифрових сигнальних процесорів та апаратних акселераторів [4-6]. Це обмежує їхнє масове впровадження, насамперед у малому та середньому транспортному бізнесі, де вартість обладнання є критичним чинником.

Мета роботи — розроблення інтелектуальної системи відеомоніторингу стану водія, що реалізує функції системи підтримки прийняття рішень (СППР) та забезпечує своєчасне виявлення ознак втоми на основі аналізу відеопотоку з веб-камери в режимі реального часу. Система має функціонувати на стандартному персональному комп'ютері без застосування спеціалізованого апаратного забезпечення.

Для досягнення поставленої мети у роботі вирішено такі задачі:

- 1) аналіз статистики дорожньо-транспортних пригод, пов'язаних із втомою водіїв, та визначення ключових візуальних ознак сонливості;

- 2) огляд існуючих комерційних DMS-систем та виявлення їхніх спільних обмежень;
- 3) огляд методів комп'ютерного зору та обґрунтування вибору технологічного стеку для розроблюваної системи;
- 4) формування навчального датасету та навчання моделі YOLOv8n із застосуванням domain adaptation;
- 5) розроблення алгоритму накопичення індексу сонливості та реалізація програмного прототипу системи.

Об'єктом дослідження є процес виявлення ознак сонливості та втоми водія за відеопотоком у реальному часі. Предметом дослідження є методи комп'ютерного зору, архітектури нейронних мереж та алгоритми обробки відеопотоку, що застосовуються для розв'язання поставленої задачі.

Практична цінність роботи полягає в тому, що розроблена система розгортається на стандартній веб-камері та персональному комп'ютері без спеціалізованих сенсорів. Це суттєво знижує поріг впровадження технологій моніторингу стану водія та робить їх доступними для широкого кола користувачів.

Методологічною основою роботи є поєднання класичних методів комп'ютерного зору та сучасних методів глибинного навчання. Для локалізації обличчя і зони очей застосовано каскадні класифікатори на ознаках Хаара, що забезпечують високу швидкодію без потреби у навчанні, а для класифікації стану ока — згорткову нейронну мережу архітектури YOLOv8, навчену з використанням перенесення навчання та domain adaptation. Така гібридна архітектура дозволяє поєднати переваги обох підходів: швидкість класичних методів і точність нейромереж. Дослідження виконувалося з опорою на відкриті набори даних, відкриті програмні бібліотеки та власноруч зібраний навчальний датасет, що забезпечує відтворюваність отриманих результатів.

Наукова та практична новизна роботи полягає у застосуванні напіваавтоматичного конвеєра збору власних навчальних даних безпосередньо

через робочий пайплайн системи, завдяки чому навчальні зображення проходять ту саму послідовність обробки, що й кадри під час реальної експлуатації. Це усуває зсув розподілу між навчальними та робочими даними і дозволяє досягти високої точності розпізнавання на доступному обладнанні. Додатковим елементом новизни є двошаровий алгоритм накопичення індексу сонливості з асиметричним оновленням, що забезпечує одночасно стійкість до поодиноких хибних класифікацій та швидке повернення системи у безпечний стан.

Робота складається із переліку умовних позначень, вступу, чотирьох розділів, висновків, переліку джерел посилання та двох додатків. У першому розділі досліджується предметна область, аналізується статистика дорожньо-транспортних пригод та огляд існуючих систем моніторингу. Другий розділ присвячено теоретичному апарату: розглядаються методи комп'ютерного зору, математичний апарат детектора Viola–Jones, архітектура нейромережі YOLOv8 та алгоритм накопичення індексу сонливості. У третьому розділі описано програмну реалізацію системи, формування датасету, навчання моделі та результати її роботи. Четвертий розділ присвячено функціонально-вартісному аналізу програмного продукту.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Втома водіїв як чинник дорожньо-транспортної небезпеки

Серед усіх чинників, що впливають на безпеку дорожнього руху, втома водія посідає особливе місце: вона є вкрай небезпечною та водночас погано піддається контролю. Якщо алкогольне або наркотичне сп'яніння виявляється швидкими інструментальними тестами, то для втоми такого простого індикатора не існує — вона не залишає за собою об'єктивних слідів, які можна було б зафіксувати на місці події. Через це в офіційних протоколах втома фігурує як причина пригоди вкрай рідко: на момент приїзду інспекторів водій, як правило, вже повертається у бадьорий стан, а постфактум встановити сонливість майже неможливо. Як наслідок, статистичні дані описують лише видимий шар проблеми, тоді як реальна частка аварій, спричинених втомою, лишається значно вищою за зафіксовані показники.

За оцінками Національної адміністрації безпеки дорожнього руху США (NHTSA), у Сполучених Штатах щорічно фіксується понад 100 000 дорожньо-транспортних пригод, безпосередньо пов'язаних із сонливістю за кермом; їх наслідком стають близько 71 000 травмованих та приблизно 1 500 загиблих [2]. Самі дослідники застерігають, що ці значення слід сприймати як нижню межу: альтернативні методики оцінювання дають у два-три рази вищі показники. Висновки глобальної доповіді Всесвітньої організації охорони здоров'я узгоджуються з цією картиною — людський фактор, складовою якого виступає й втома, лишається провідною причиною травматизму на дорогах у світовому масштабі [1]. Найгостріше проблема постає перед двома категоріями водіїв: професіоналами вантажних та пасажирських перевезень,

чия зміна триває багато годин поспіль, і тими, хто кермує у нічний час, коли біологічні ритми організму налаштовані на відпочинок.

Фізіологічним підґрунтям небезпеки є явище мікросну — короткочасних, тривалістю від двох до тридцяти секунд, неконтрольованих провалів у сон. Перебуваючи у такому стані, людина повністю випадає з керування ситуацією, не відгукується на зовнішні стимули, а нерідко й сама не помічає епізоду засинання, повертаючись у свідомість із враженням, ніби лише на мить розсіялась. Підступність цього явища полягає у його раптовості: воно настає поза волею людини, незалежно від її намірів зберігати увагу. Те, чим обертається мікросон на дорозі, безпосередньо визначається швидкістю руху. Транспортний засіб, що рухається зі швидкістю 90 км/год, за секунду проходить близько 25 метрів; отже, навіть короткий, п'ятисекундний епізод відповідає приблизно 125 метрам неконтрольованого шляху — такої відстані достатньо, щоб з'їхати у кювет, перетнути сусідню смугу або в'їхати у автомобіль попереду. Саме з цієї причини рання діагностика сонливості, ще до настання повного мікросну, становить пріоритетне завдання сучасних систем активної безпеки.

Для об'єктивної кількісної оцінки рівня сонливості у практиці моніторингу стану водія застосовують параметр PERCLOS (Percentage of Eye Closure) — частку часу протягом заданого інтервалу, упродовж якої очі закриті більш ніж на 80% [3, 21]. Цей показник був запропонований як результат серії досліджень, у яких порівнювали різні поведінкові ознаки втоми з еталонними фізіологічними вимірюваннями; саме PERCLOS продемонстрував найвищу кореляцію з реальним рівнем сонливості. Формально показник визначається як відношення кількості кадрів, у яких очі є закритими, до загальної кількості кадрів за інтервал спостереження, що подано формулою (1.1)

$$PERCLOS = \frac{N_{closed}}{N_{total}} \cdot 100\% \quad (1.1)$$

де N_{closed} — кількість кадрів інтервалу, у яких повіки закриті більш ніж на 80%; N_{total} — загальна кількість кадрів за інтервал спостереження. Наприклад, якщо за інтервал спостереження тривалістю 60 секунд при частоті 30 кадрів на секунду зафіксовано 1 800 кадрів, з яких очі були закритими у 320 кадрах, то значення показника складе $(320 / 1\,800) \cdot 100\% \approx 17,8\%$, що вже наближається до критичної межі. Експериментально встановлено, що значення PERCLOS, яке перевищує 15–20%, є надійною ознакою вираженої сонливості водія. Саме PERCLOS покладено в основу більшості сучасних відеосистем моніторингу як найбільш об'єктивний і відтворюваний маркер втоми, оскільки він не залежить від суб'єктивної самооцінки водія і обчислюється безпосередньо з відеопотоку.

Окрім PERCLOS, до візуальних маркерів сонливості належать частота моргань, тривалість окремого моргання, опускання верхньої повіки та позіхання. У стані нормального неспання людина моргає 15–20 разів на хвилину, причому кожне моргання триває близько 100–150 мілісекунд. У стані втоми частота моргань знижується до 8–10 разів на хвилину, а тривалість кожного моргання зростає у кілька разів, оскільки м'язи, що утримують повіки, втрачають тонус. Додатковою ознакою є так зване «плавання погляду» — уповільнені, нечіткі рухи очей. Усі ці характеристики придатні для автоматизованого відеоаналізу, оскільки спостерігаються безпосередньо на зображенні обличчя водія і не потребують жодних контактних датчиків.

З аналізу перелічених маркерів впливає чітка послідовність технічних підзадач, які має розв'язати система відеомоніторингу:

- 1) локалізація обличчя водія у кадрі як першочерговий етап, що обмежує область подальшої обробки;
- 2) виділення зони очей у межах знайденого обличчя для зменшення обчислювального навантаження;
- 3) класифікація стану кожного ока як відкритого або закритого на основі вирізаного зображення;

- 4) накопичення статистики стану очей у часі та генерація сигналу тривоги при перевищенні критичного порогу.

Саме ця послідовність підзадач визначає архітектуру розроблюваної системи й буде деталізована у наступних розділах. Важливо підкреслити, що кожна з підзадач має власні вимоги до точності та швидкодії: помилка на етапі локалізації обличчя унеможлиблює всі подальші кроки, тоді як помилка класифікації окремого кадру може бути скомпенсована статистичним накопиченням у часі.

1.2 Огляд існуючих систем моніторингу стану водія

1.2.1 Комерційні DMS-системи

На сучасному ринку представлено кілька промислових систем моніторингу стану водія (Driver Monitoring System, DMS), що застосовуються переважно у складі систем активної безпеки автомобілів та у комерційному автотранспорті. Ці системи розроблялися великими технологічними компаніями впродовж останнього десятиліття і втілюють передові досягнення у галузі комп'ютерного зору та обробки сигналів. Розглянемо три найбільш відомі рішення із детальним аналізом принципів їхньої роботи, переваг та недоліків.

1. В основі системи Seeing Machines FOVIO лежить інфрачервона камера, поєднана з цифровим сигнальним процесором та програмованою логічною матрицею (DSP/FPGA). Робочий принцип ґрунтується на активному ІЧ-підсвічуванні водія, завдяки чому система отримує чітку картинку обличчя за будь-яких умов освітленості, у тому

числі вночі, коли звичайна RGB-камера фактично втрачає працездатність. Алгоритмічна частина одночасно стежить за напрямом погляду, мірою розкриття повік і просторовою орієнтацією голови, об'єднуючи ці три сигнали в інтегральну оцінку рівня уваги. До сильних сторін рішення слід віднести майже повну незалежність від зовнішнього освітлення та стабільну роботу впродовж доби. Серед недоліків — значна вартість, що впливає з використання вузькоспеціалізованих апаратних компонентів, та закритий програмний інтерфейс, який не дає стороннім інтеграторам адаптувати систему під власні задачі [4].

2. У рішенні Mobileye Guardian поєднуються RGB- та інфрачервоний канали, а сама система щільно вбудовується в архітектуру допоміжних засобів водія (Advanced Driver Assistance Systems, ADAS). Її специфіка полягає у тому, що стеження за станом водія — це лише одна з функцій ширшого комплексу безпеки, який паралельно сканує дорожнє оточення: розмітку, сусідні автомобілі, пішоходів. Завдяки такому об'єднанню рішення приймаються з огляду одночасно і на стан водія, і на дорожню обстановку: скажімо, інтенсивність попередження зростає тоді, коли ознаки втоми накладаються на критичне зближення з автомобілем попереду. У сильні сторони системи входить її роль повноцінної комплексної платформи безпеки. Серед слабких місць слід відзначити складність монтажу, необхідність кількох апаратних модулів, а отже й зростання як ціни, так і вимог до встановлення [5].

3. Рішення Bosch DMS базується на інфрачервоній камері й розраховане на заводську інтеграцію у нові автомобілі (ОЕМ-формат постачання). Його алгоритми оцінюють просторову орієнтацію голови та стан очей, причому первинною сферою застосування є транспортні засоби з рівнями автоматизації L2–L3, де водій зобов'язаний у будь-який момент бути готовим перехопити керування. Bosch DMS показовий саме як приклад глибокого вмонтовування модуля моніторингу водія в архітектуру автомобіля. Серед негативних сторін такого підходу —

закрите алгоритмічне ядро, недоступне для зовнішнього доопрацювання, та прив'язка винятково до нових моделей зі спеціально передбаченою інфраструктурою; дообладнати таким рішенням вже наявний у експлуатації транспортний засіб практично неможливо [6].

1.2.2 Порівняльний аналіз та спільні обмеження

Для систематизації наведених рішень доцільно зіставити їх за ключовими технічними характеристиками: типом сенсора, рівнем апаратних вимог, стійкістю до умов освітлення та вартістю. Таке зіставлення дозволяє виявити не лише індивідуальні відмінності, а й спільні структурні обмеження, притаманні всьому класу комерційних систем. Порівняльний аналіз DMS-систем подано у таблиці 1.1, де поряд із трьома комерційними рішеннями для контрасту наведено характеристики системи, розробленої у цій роботі.

Таблиця 1.1 — Порівняльний аналіз DMS-систем

Система	Тип сенсора	Апаратні вимоги	Стійкість до освітлення	Вартість
Seeing Machines FOVIO	ІЧ-камера + DSP/FPGA	Дуже високі	Висока (активне ІЧ)	Дуже висока
Mobileye Guardian	RGB + ІЧ	Високі	Обмежена (RGB)	Висока
Bosch DMS	ІЧ-камера	Високі (OEM)	Висока (активне ІЧ)	Висока
Розроблена система	RGB (веб-камера)	Низькі (CPU/GPU)	Адаптивна (equalizeHist)	Низька

Зіставлення, наведене у таблиці 1.1, показує, що всі розглянуті комерційні системи мають спільний принциповий недолік — залежність від дорогого спеціалізованого апаратного забезпечення, будь то інфрачервоні камери, сигнальні процесори чи програмовані матриці. Саме цей чинник є

основним бар'єром, що обмежує масове впровадження систем моніторингу стану водія. Для великих автовиробників вартість таких компонентів розподіляється на масовий випуск і є прийнятною, проте для власників окремих транспортних засобів, малих і середніх перевізників вона залишається непомірною. Реалізована система адресує саме цю проблему: вона функціонує на базі стандартної RGB веб-камери та звичайного центрального процесора, що принципово знижує її вартість і поріг впровадження, зберігаючи при цьому достатню для практичного застосування точність розпізнавання.

1.3 Підходи до автоматичного виявлення ознак сонливості

Усі підходи до автоматичного виявлення ознак сонливості водія можна поділити на три великі групи: фізіологічні методи, поведінкові методи на основі параметрів транспортного засобу та методи комп'ютерного зору [24]. Кожна група ґрунтується на власному джерелі інформації про стан водія і має характерний набір переваг та обмежень. Розгляд цих груп є необхідним для обґрунтованого вибору підходу, покладеного в основу розроблюваної системи.

1. Фізіологічна група методів спирається на знімання біоелектричних сигналів організму — електроенцефалограми (ЕЕГ), що фіксує електричну активність кори, електроокулограми (ЕОГ) як засобу реєстрації рухів очей, електрокардіограми (ЕКГ) та частоти серцевих скорочень. Такі підходи дають максимальну точність визначення стану водія, бо знімають інформацію безпосередньо із джерела — нервової та серцево-судинної систем, де ознаки сонливості з'являються раніше за будь-які поведінкові вияви. Так, посилення альфа- та тета-діапазонів у

ЕЕГ-спектрі традиційно розглядається як «золотий стандарт» виявлення сонливості. У літературі також пропонується компромісне рішення — застосування наручної фотоплетизмографії, яка балансує між точністю та зручністю носіння [13]. Слабке місце усіх цих методик одне й те саме і має принциповий характер — необхідність розміщення на тілі водія додаткових сенсорів: електродів на скальпі, нагрудних поясів, браслетів. У повсякденному кермуванні це є практично неприйнятним: жодний водій не погодиться постійно носити вимірювальну апаратуру.

2. Друга група — поведінкові методи, орієнтовані на параметри самого транспортного засобу. Вони відстежують особливості керування: дрібні відхилення кермового колеса, частоту та амплітуду коригувальних дій, манеру роботи з педалями, а також положення авто у межах смуги (для цього вже існують серійні системи Lane Departure Warning System, LDWS). Базова логіка проста: втомлений водій кермує менш плавно, у нього довшають інтервали пасивності, які переривають різкі підправлення траєкторії. До переваг таких методів належать відсутність камери, спрямованої на обличчя людини, і використання штатних сенсорів, що вже стоять у сучасних автомобілях. Однак суттєвим обмеженням залишається непрямий характер діагностики: про втому судять за наслідками у керуванні, а не за безпосереднім фізичним станом водія. Окрім того, манера керування істотно змінюється від типу дороги, швидкісного режиму та особистого стилю водіння, через що задати єдині універсальні пороги спрацювання виявляється непростю задачею.

3. Підходи на базі комп'ютерного зору працюють із відеозображенням обличчя: вони оцінюють стан очей (відкриті чи закриті, ступінь розкриття повік), стежать за положенням рота (зокрема за позіханням) і за нахилом та поворотом голови. Перевагами цього напряму є відсутність контакту з водієм, відносно дешеве апаратне забезпечення — звичайна камера — і пряма реєстрація тих візуальних

маркерів сонливості (див. підрозділ 1.1). Завдяки цьому саме комп'ютерний зір став магістральним напрямом для побудови масових систем моніторингу. Усередині самої групи розрізняють два основні підходи: класичний, де ознаки прописуються вручну за заздалегідь визначеними правилами, і нейромережевий, де ознаки виокремлюються самою моделлю у процесі навчання. Детальне зіставлення цих підходів подано у розділі 2.

Якщо звести три розглянуті групи методів воедино, висновок щодо технологічного вибору для розроблюваної системи стає однозначним. Фізіологічні підходи, хоч і дають найбільшу точність, відсіюються через прив'язку до контактних датчиків; поведінкові методи у кращому випадку забезпечують опосередковану й нестабільну оцінку. На відміну від них, методи комп'ютерного зору вдало поєднують безпосередність вимірювання стану водія з низькими апаратними вимогами — для них достатньо звичайної веб-камери. Саме тому в основу пропонованої у цій роботі системи покладено комп'ютерний зір: такий вибір повністю відповідає поставленій меті — створити доступне рішення, що не потребує спеціалізованого обладнання.

1.4 Формалізація задачі

На основі проведеного аналізу предметної області сформулюємо постановку задачі у формалізованому вигляді. Чітка формалізація необхідна для того, щоб однозначно визначити межі системи, її вхідні та вихідні дані, обмеження та критерії, за якими оцінюватиметься успішність розв'язання. Об'єктом дослідження є процес виявлення ознак сонливості водія за відеопотоком, а предметом — методи та засоби комп'ютерного зору і глибинного навчання, що застосовуються для розв'язання цієї задачі. Метою є

розроблення системи, яка у реальному часі аналізує стан очей водія і генерує попередження при виявленні ознак сонливості.

Вхідними даними системи є відеопотік з RGB-камери (веб-камери) з типовою роздільною здатністю 640×480 або 1280×720 пікселів та частотою 15–30 кадрів на секунду. Камера встановлюється у салоні автомобіля так, щоб обличчя водія перебувало у її полі зору. Жодних додаткових сенсорів — інфрачервоних випромінювачів, датчиків положення керма чи фізіологічних електродів — система не використовує. Вихідними даними є бінарна класифікація стану кожного виявленого ока (`drowsy` — сонне, або `not_drowsy` — активне), накопичений у часі індекс сонливості та візуальне попередження водієві у вигляді зміни кольору елементів інтерфейсу і текстового сигналу тривоги.

Ключовим обмеженням, що впливає з мети створення доступної системи, є вимога роботи у режимі реального часу — не менше 6 кадрів на секунду — на стандартному центральному процесорі без спеціалізованого апаратного забезпечення. Це обмеження безпосередньо впливає на вибір алгоритмів: вони мають бути достатньо легкими, щоб обробляти кадр у межах сотні мілісекунд, але при цьому достатньо точними, щоб не пропускати реальних випадків сонливості. Як критерій якості розпізнавання обрано перевищення порогових значень основних метрик детекції: точності (`Precision`) не менше 0.90, повноти (`Recall`) не менше 0.90 та усередненої точності (`mAP50`) не менше 0.90. Ці значення є цільовими орієнтирами, дотримання яких підтверджує придатність системи для практичного застосування. Особливий пріоритет надається повноті, оскільки для системи безпеки пропуск реального випадку сонливості є значно небезпечнішим за хибну тривогу.

Висновки до розділу 1

У розділі проаналізовано предметну область та встановлено масштаб проблеми втоми водіїв як чинника дорожньо-транспортної небезпеки. Показано, що втома і мікросон спричиняють значну частку серйозних дорожньо-транспортних пригод, а реальний внесок цього чинника систематично недооцінюється через складність його фіксації постфактум. Параметр PERCLOS та характеристики моргань визначено як ключові об'єктивні маркери сонливості, придатні для автоматизованого відеоаналізу, та наведено числовий приклад обчислення PERCLOS.

Огляд комерційних DMS-систем — Seeing Machines FOVIO, Mobileye Guardian та Bosch DMS — виявив їх спільне принципове обмеження: залежність від дорогого спеціалізованого апаратного забезпечення, що є основним бар'єром для масового впровадження. Порівняння трьох груп методів автоматичного виявлення сонливості — фізіологічних, поведінкових та методів комп'ютерного зору — показало, що для доступної системи без спеціалізованого обладнання найперспективнішими є методи комп'ютерного зору. На основі цього аналізу сформульовано постановку задачі, визначено вхідні та вихідні дані, обмеження роботи у реальному часі на стандартному CPU та цільові метрики якості системи.

2 ТЕОРЕТИЧНИЙ АПАРАТ

2.1 Методи комп'ютерного зору для аналізу стану очей

2.1.1 Класичні методи: каскади Хаара та метод EAR

Класичні методи комп'ютерного зору, призначені для аналізу стану очей, спираються не на навчання на великих масивах даних, а на ознаки зображення, описані заздалегідь у явному вигляді. До цієї групи належать каскадні класифікатори, побудовані на ознаках Хаара, а також метод відношення аспектів ока (Eye Aspect Ratio, EAR).

Локалізація обличчя у роботі виконується за допомогою каскадного детектора, який запропонували Viola та Jones [7, 23]. Його провідна ідея полягає у відмові від обробки сирих значень пікселів на користь обчислення прямокутних перепадів яскравості — так званих ознак Хаара, що чутливі до характерного для людського обличчя чергування світлих і темних зон. Класифікатор, попередньо натренований за каскадною схемою, по черзі переглядає підвікна зображення і одразу відсіює ті, у яких шуканого об'єкта точно немає, чим і досягається висока швидкість роботи. На сучасному центральному процесорі один кадр обробляється приблизно за 5–30 мс. У OpenCV вже міститься набір готових натренованих каскадів — зокрема для облич (файл `haarcascade_frontalface_default.xml`) та для очей (файл `haarcascade_eye.xml`) [11].

Геометрія ознак не випадкова: вона свідомо узгоджена з будовою людського обличчя. Пара горизонтально розташованих прямокутників добре

фіксує перепад яскравості між зоною лоба й очними западинами, а вертикальна трійка чутлива до характерного просвіту перенісся між двома очима. Така узгодженість форми ознаки з анатомією і пояснює ефективність детектора. Кожна ознака почергово «накладається» на підвікно у всіх можливих координатах і масштабах, тож загальна кількість потенційних ознак стає величезною. Очні западини здебільшого виглядають темнішими за прилеглі ділянки щік і чола — саме на цьому стійкому контрасті, який спостерігається на більшості облич, і ґрунтується сила ознак Хаара: область очей, як правило, темніша за смугу, розташовану безпосередньо нижче за неї.

Метод EAR, запропонований Soukupová та Čech у 2016 році [8], ґрунтується на шести ключових точках, розташованих навколо ока. Відношення аспектів ока обчислюється за формулою (2.1)

$$EAR = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2 \cdot \|p_1 - p_4\|} \quad (2.1)$$

де $p_1 \dots p_6$ — координати шести ключових точок ока, а подвійні вертикальні риски позначають евклідову відстань між точками. На відкритому оці EAR коливається приблизно від 0.25 до 0.35 залежно від людини; при повному закриванні повіки значення падає до нуля. За поріг спрацювання приймають $EAR < 0.25$ протягом щонайменше трьох послідовних кадрів, що інтерпретується як моргання або ознака сонливості. Основним обмеженням методу є залежність від якості детекції 68 ключових точок обличчя: точність EAR деградує при поворотах голови, поганому освітленні та вібрації камери.

2.1.2 Методи на основі глибинного навчання: CNN та YOLO

Принциповою рисою методів глибинного навчання є те, що ознаки виокремлюються з даних автоматично, без явного формулювання правил людиною [15]. Глибокі нейронні мережі вже встигли довести свою спроможність у широкому переліку задач розпізнавання — від обробки мовленнєвих сигналів [19] до підходів, що передбачають неперервне навчання у задачах розпізнавання активності [16]; теоретичні засади їх побудови узагальнено в класичних монографіях з глибинного навчання [20]. Для аналізу стану очей у літературі найчастіше залучають згорткові нейронні мережі (Convolutional Neural Network, CNN), серед яких популярні архітектури VGG16, ResNet-50 та MobileNetV2. Сильні сторони CNN — самостійне виокремлення інформативних ознак і толерантність до різних варіацій вхідного зображення; зворотним боком виступає необхідність виконувати локалізацію об'єкта окремим етапом та помітно вищі витрати часу на інференс.

Сімейство архітектур YOLO (You Only Look Once) втілює зовсім іншу логіку [9]. Для нашої задачі його головна перевага полягає у тому, що локалізація і класифікація виконуються одним наскрізним проходженням мережі. На відміну від нього, двостадійні детектори (з окремими стадіями пропозиції регіонів та подальшої їх класифікації) дають вищу точність на ускладнених сценах, але для відеопотоку реального часу виявляються занадто повільними. YOLO свідомо поступається часткою точності, отримуючи натомість змогу обробляти десятки кадрів щосекунди навіть на типовому CPU. У низці послідовних версій — зокрема YOLOv4 [18] — співвідношення швидкості та точності поступово вдосконалювалося. У 2023 році Ultralytics представила архітектуру YOLOv8 [10], яка спирається на модернізований CSP-backbone, шию у вигляді мережі піраміди ознак з агрегацією PANet та

anchor-free голову детекції. Серед підтримуваних режимів є режим детекції об’єктів — саме цей режим було використано у розробленій системі.

2.1.3 Порівняльний аналіз методів

Для обґрунтованого вибору технологічного стеку зіставимо розглянуті методи за основними критеріями. Результати порівняння наведено у таблиці 2.1.

Таблиця 2.1 — Порівняльний аналіз методів детекції сонливості

Метод	Точність	Швидкість	Стійкість до освітлення	Потреба в даних	Складність інтеграції
EAR (landmarks)	Середня	Висока	Низька	Мала	Середня
Haar Cascade	Середня	Дуже висока	Середня	Не потрібна	Низька
CNN (класифікатор)	Висока	Середня	Висока	Велика	Середня
YOLOv8 (детектор)	Висока	Висока	Висока	Середня	Низька

Зіставлення показує, що жоден окремий метод не є оптимальним за всіма критеріями одночасно. Тому у роботі обрано гібридну архітектуру, що поєднує сильні сторони двох підходів: каскад Хаара застосовується для швидкої локалізації обличчя та зони очей (висока швидкодія, відсутність потреби у навчанні), а нейромережа YOLOv8 — для точної та стійкої класифікації стану виділеного ока.

2.2 Математичний апарат детектора Viola–Jones

2.2.1 Інтегральне зображення та обчислення ознак Хаара

Висока швидкодія детектора Viola–Jones значною мірою тримається на одній структурі даних — інтегральному зображенні. Завдяки йому суму яскравостей пікселів у довільному прямокутнику вдається отримати за константний час, який не залежить від розмірів самого прямокутника. У кожній точці (x, y) значення інтегрального зображення дорівнює сумарній яскравості всіх пікселів, що містяться вище та лівіше від неї; формальне визначення подано у формулі (2.2)

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y') \quad (2.2)$$

де $i(x', y')$ — яскравість пікселя вхідного зображення. На практиці інтегральне зображення обчислюється за один прохід завдяки рекурентному співвідношенню (2.3)

$$ii(x, y) = i(x, y) + ii(x - 1, y) + ii(x, y - 1) - ii(x - 1, y - 1) \quad (2.3)$$

Маючи побудоване інтегральне зображення, суму яскравостей пікселів у довільному прямокутнику з вершинами A, B, C, D обчислюють лише за чотирма значеннями ii за формулою (2.4)

$$S = ii(D) + ii(A) - ii(B) - ii(C) \quad (2.4)$$

Власне ознака Хаара визначається як різниця між сумою яскравостей пікселів у світлих та темних прямокутних областях, що подано формулою (2.5)

$$f_j(x) = \sum_{r \in R^+} S(r) - \sum_{r \in R^-} S(r) \quad (2.5)$$

де R^+ та R^- — множини світлих та темних прямокутників ознаки відповідно. Головний виграш від інтегрального зображення — константний час обчислення суми у будь-якому прямокутнику незалежно від його розміру. Без нього складність зростала б пропорційно площі підвікна; тепер вона не залежить від неї взагалі — завжди чотири операції над чотирма точками кутів. Це і є тим прийомом, який робить детектор Viola–Jones придатним для роботи у реальному часі.

Проілюструємо обчислення на простому прикладі. Нехай для прямокутної області значення інтегрального зображення у її чотирьох вершинах становлять: $ii(A) = 50$ (верхній лівий кут), $ii(B) = 120$ (верхній правий), $ii(C) = 90$ (нижній лівий), $ii(D) = 230$ (нижній правий). Тоді сума яскравостей усіх пікселів усередині цього прямокутника за формулою (2.4) дорівнює $S = 230 + 50 - 120 - 90 = 70$, і це значення отримано лише чотирма операціями додавання та віднімання незалежно від того, чи містить прямокутник десятки чи десятки тисяч пікселів. Саме сталий час обчислення суми, помножений на сотні тисяч ознак, які потрібно перевірити, і дає сумарний виграш у швидкодії, що відрізняє метод Viola–Jones від прямого підсумовування пікселів.

У розробленій системі саме ця властивість дозволяє детектору обличчя (haarcascade_frontalface_default.xml) обробляти кожен кадр 640×480 пікселів за 5–30 мс на CPU — що є необхідною умовою роботи всього пайплайну в реальному часі.

2.2.2 Алгоритм AdaBoost і каскадна структура класифікатора

Навіть для відносно маленького підвікна повний набір потенційних ознак Хаара обчислюється сотнями тисяч, тож одним із центральних завдань стає виокремлення вузької вибірки найінформативніших із них. Розв'язок дає бустинговий алгоритм AdaBoost: сильний класифікатор будується у ньому як зважена сума слабких. Кожен такий слабкий класифікатор використовує рівно одну ознаку і задається порогом θ_j та полярністю p_j згідно з формулою (2.6)

$$h_j(x) = \begin{cases} 1, & \text{якщо } p_j f_j(x) < p_j \theta_j; \\ 0 & - \text{інакше} \end{cases} \quad (2.6)$$

На кожному кроці бустингу (t) система «фокусується» на тих зображеннях, де попередні класифікатори помилилися — їхня вага підвищується. Це змушує кожен наступний слабкий класифікатор вчитися саме на складних прикладах, а не повторювати те, що вже вміють попередні. Вага нового класифікатора α_t у підсумковій суміші тим вища, чим нижча його зважена похибка ε_t , як бачимо з формули (2.7)

$$\alpha_t = \frac{1}{2} \ln \frac{1 - \varepsilon_t}{\varepsilon_t} \quad (2.7)$$

Після обчислення ваги оновлюються ваги навчальних зразків: вага правильно класифікованих зменшується, а неправильно класифікованих — відносно зростає, що змушує наступні слабкі класифікатори зосереджуватися на складних прикладах. Підсумковий сильний класифікатор формується як зважена сума слабких згідно з формулою (2.8).

Формула (2.7) має наочну інтерпретацію: чим точніший слабкий класифікатор (тобто чим менша його похибка ε_t), тим більшою є його вага α_t у підсумковій сумі. Наприклад, при похибці $\varepsilon_t = 0,2$ вага становить $\alpha_t = 0,5 \cdot \ln((1 - 0,2)/0,2) = 0,5 \cdot \ln 4 \approx 0,69$, тоді як при похибці $\varepsilon_t = 0,4$ вага

падає до $\alpha_t = 0,5 \cdot \ln(0,6/0,4) = 0,5 \cdot \ln 1,5 \approx 0,20$. Класифікатор, що працює на рівні випадкового вгадування ($\varepsilon_t = 0,5$), отримує нульову вагу і не впливає на результат. Таким чином AdaBoost автоматично надає більшої довіри інформативнішим ознакам, формуючи з безлічі слабких правил одне надійне:

$$H(x) = \text{sign}\left(\sum_{t=1}^T \alpha_t h_t(x)\right) \quad (2.8)$$

Для подальшого прискорення детектор організовано у вигляді каскаду: класифікатори об'єднано у послідовні стадії, кожна з яких відкидає очевидні негативні підвікна. Більшість підвікон зображення не містять обличчя і відсіюються вже на перших, найпростіших стадіях. Завдяки цьому обчислювальні ресурси витрачаються лише на нечисленні перспективні ділянки, що скорочує загальний час обробки на 90–95% порівняно з рівномірним застосуванням повного класифікатора до кожного підвікна.

2.3 Метод Eye Aspect Ratio та його обмеження

Метод Eye Aspect Ratio є простим і обчислювально дешевим способом оцінки ступеня розкриття ока на основі геометрії шести ключових точок $p_1 \dots p_6$, які локалізуються навколо ока детектором ключових точок обличчя. Дві точки задають горизонтальні кути ока, а чотири — верхню і нижню повіки. У чисельнику формули (2.1) обчислюється сума двох вертикальних евклідових відстаней між повіками, у знаменнику — подвоєна горизонтальна відстань між кутами ока. При закриванні повік вертикальні відстані зменшуються, тому значення EAR різко спадає, що й дозволяє виявити моргання чи закривання очей.

У реальних умовах експлуатації простота методу обертається помітними обмеженнями:

- 1) варто водієві повернути голову більш ніж приблизно на 15° , як перспективне спотворення геометрії ока починає провокувати хибні тривоги;
- 2) типова для автомобіля вібрація камери з амплітудою 2–6 пікселів у діапазоні частот 5–20 Гц вносить у обчислення EAR похибку приблизно на рівні 20–40%;
- 3) метод цілком залежить від детектора 68 ключових точок обличчя, а той у свою чергу втрачає стабільність при нерівномірному освітленні — і вся ланка деградує разом із ним.

Порівняно з методом EAR нейромережевий підхід YOLOv8 є суттєво стійкішим. Замість того щоб спиратися на координати шести окремих точок, мережа оперує глобальною текстурою всього зображення ока, тому є менш чутливою до локальних зміщень окремих точок. Крім того, навчання на аугментованому датасеті із варіаціями яскравості та геометрії забезпечує явну адаптацію моделі до різних умов освітлення та ракурсів. Саме ці міркування зумовили вибір YOLOv8 для класифікації стану ока у розробленій системі.

2.4 Архітектура нейромережі YOLOv8

2.4.1 Структура backbone–neck–head та anchor-free detection

Структурно YOLOv8 розпадається на три функціональні блоки: основу (backbone), шию (neck) та голову (head). Основою виступає оновлена мережа CSPDarknet, побудована на модулях C2f (Cross Stage Partial); саме ця частина

мережі відповідає за виокремлення низько- та середньорівневих ознак з вхідного зображення. Завдяки тому, що між стадіями використовується часткове перенесення карт ознак, обчислювальні витрати знижуються, але інформативність представлення при цьому не страждає.

Шия мережі реалізована у вигляді мережі піраміди ознак (Feature Pyramid Network) з агрегацією за схемою PANet. Її роль — зведення карт ознак різних масштабів, причому потоки інформації йдуть одночасно у двох напрямках: від нижчих рівнів до вищих і навпаки; саме це і дає змогу впевнено детектувати об'єкти різного розміру. Голова працює за принципом anchor-free, тобто обходиться без заздалегідь визначених опорних рамок: вона безпосередньо передбачає координати центру об'єкта, його ширину та висоту, а також імовірності належності до кожного з класів. Це водночас спрощує процес навчання і знімає потребу попередньо аналізувати розподіл розмірів об'єктів у датасеті.

Сімейство YOLOv8 включає кілька конфігурацій різної складності — від YOLOv8n (nano) до YOLOv8x. За один прохід основа і шия формують багатомасштабне представлення зображення, після чого голова повертає набір рамок та відповідних класів. Така одноетапна організація і забезпечує характерну для YOLO високу швидкодію.

Блок C2f, що лежить в основі backbone, є розвитком ідеї часткового перенесення ознак між стадіями (Cross Stage Partial): вхідна карта ознак розділяється на дві гілки, одна з яких проходить через послідовність згорткових перетворень, а друга передається далі без змін, після чого обидві гілки об'єднуються. Це зменшує дублювання обчислень градієнтів і кількість параметрів, зберігаючи багатство представлення ознак. Мережа піраміди ознак у шії будує ієрархію карт різного просторового розрізнення: карти з високою роздільною здатністю несуть детальну інформацію, корисну для дрібних об'єктів, а карти низької роздільної здатності — узагальнену семантичну інформацію. Двонапрямкове об'єднання за схемою PANet

поширює як низькорівневі, так і високорівневі ознаки в обох напрямках, що підвищує якість детекції об'єктів усіх розмірів.

2.4.2 Обґрунтування вибору варіанту YOLOv8n

Для розроблюваної системи обрано найлегшу конфігурацію сімейства — YOLOv8n. Ця модель містить приблизно 3,2 млн параметрів, потребує близько 8,7 GFLOPs обчислень і має розмір ваг близько 6 МБ. Час інференсу нано-моделі на центральному процесорі становить порядку десятків мілісекунд на кадр, що робить її придатною для роботи на CPU без графічного прискорювача.

Вибір саме нано-конфігурації обґрунтований характером задачі. Двокласова класифікація стану ока (drowsy або not_drowsy) є відносно простою, тому застосування надлишкових моделей (YOLOv8m, YOLOv8l) не дало б суттєвого приросту точності, але помітно уповільнило б роботу системи й унеможливило б її функціонування на CPU у реальному часі. Для порівняння: модель YOLOv8s містить близько 11,2 млн параметрів, YOLOv8m — близько 25,9 млн, тобто у вісім разів більше за нано-версію, що пропорційно збільшує час інференсу. Оскільки цільовим середовищем виконання є звичайний центральний процесор без графічного прискорювача, кожен зайвий мільйон параметрів безпосередньо знижує досяжну частоту кадрів, тому компроміс на користь найлегшої моделі є обґрунтованим.

Важливим архітектурним нюансом є те, що модель формально використовується як детектор (результат зчитується через інтерфейс `results[0].boxes`), однак функціонально працює як класифікатор. На вхід мережі подається вже ізольоване зображення ока, вирізане каскадом Хаара, а в навчальному датасеті обмежувальна рамка покриває весь кадр із

нормалізованими координатами 0.5 0.5 1.0 1.0. Через це координати рамки, які повертає YOLO, у роботі не використовуються — з результату беруться лише номер класу та значення впевненості моделі.

2.5 Алгоритм накопичення індексу сонливості у часі

Окремі хибні рішення нейромережі — наприклад, при звичайному моргані або частковому затуленні ока — не повинні призводити до спрацювання тривоги. Щоб надати системі стійкості до подібних випадкових помилок, у роботі побудовано двошаровий алгоритм накопичення індексу сонливості. Він зводить разом два механізми: мажоритарне голосування в межах короткого вікна кадрів та накопичувальний лічильник, у якому збільшення та зменшення відбуваються несиметрично.

Перший шар реалізує мажоритарне голосування за ковзним вікном. Структура даних deque з обмеженою довжиною $\text{maxlen} = 5$ зберігає результати класифікації останніх п'яти кадрів, де 1 відповідає сонному стану, а 0 — активному. Поточний кадр вважається сонним, якщо сума значень у вікні перевищує половину його розміру, що формалізовано умовою (2.9)

$$D_{frame} = 1, \text{ якщо } \sum_{i=1}^N e_i > \frac{N}{2} \quad (2.9)$$

де $N = 5$ — розмір ковзного вікна, $e_i \in \{0, 1\}$ — результат класифікації i -го кадру вікна. Поріг голосування становить $N/2 = 2.5$, тобто для визнання кадру сонним потрібно щонайменше три сонні кадри з останніх п'яти. Цей шар захищає систему від поодиноких помилок класифікатора та від реакції на звичайне моргання.

Другий шар реалізує накопичувальний лічильник *drowsiness_score*, що інтегрує стан у часі. Лічильник є цілим числом у діапазоні від 0 до $MAX_SCORE = 30$. При сонному кадрі він збільшується на одиницю, що описується співвідношенням (2.10), а при активному — зменшується на величину $RECOVERY_RATE = 2$ згідно зі співвідношенням (2.11)

$$score = \min(score + 1, MAX_SCORE) \quad (2.10)$$

$$score = \max(score - RECOVERY_RATE, 0) \quad (2.11)$$

Тривога активується при досягненні порогу $ALARM_THRESHOLD = 15$. Важлива особливість алгоритму — асиметрія оновлення: спад лічильника відбувається вдвічі швидше за накопичення (-2 проти $+1$). Це навмисне проєктне рішення на користь безпеки — система «прокидається» швидше, ніж «засинає», тому повертається у безпечний стан одразу після поновлення активності водія. Розглянемо чисельний приклад: якщо водій перебуває у сонному стані безперервно, лічильник зростає на одиницю кожного кадру і досягає порогу 15 власне за 15 кадрів, тобто приблизно за 1,5–2 секунди залежно від частоти кадрів. Якщо ж після цього водій розплющує очі, лічильник зменшується по 2 одиниці кожного кадру і повертається з 15 до нуля вже за 8 кадрів. Таким чином хибна тривога швидко знімається, а реальна сонливість фіксується гарантовано. Значення параметрів алгоритму зведено у таблиці 2.2.

Таблиця 2.2 — Параметри алгоритму накопичення індексу сонливості

Параметр	Значення	Пояснення
maxlen (deque)	5 кадрів	Розмір ковзного вікна
Поріг голосування	$> N/2 = 2.5$	Мажоритарна більшість
$ALARM_THRESHOLD$	15	Поріг тривоги
MAX_SCORE	30	Максимальне значення лічильника
$RECOVERY_RATE$	2	Швидкість відновлення (зменшення score)

2.6 Метрики оцінки якості системи

Для кількісної оцінки якості роботи моделі детекції застосовують стандартні метрики, що ґрунтуються на кількості правильно та хибно класифікованих об'єктів: TP (true positive) — правильно виявлені позитивні випадки, FP (false positive) — хибні спрацювання, FN (false negative) — пропущені позитивні випадки. Метрика Precision (точність) визначає частку правильних спрацювань серед усіх позитивних і обчислюється за формулою (2.12)

$$Precision = \frac{TP}{TP + FP} \quad (2.12)$$

Метрика Recall (повнота) визначає частку виявлених позитивних випадків серед усіх реально наявних і обчислюється за формулою (2.13)

$$Recall = \frac{TP}{TP + FN} \quad (2.13)$$

Збалансовану оцінку, що враховує обидві метрики одночасно, дає F1-міра — їх гармонічне середнє, визначене формулою (2.14)

$$F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \quad (2.14)$$

Для задач детекції об'єктів використовують узагальнену метрику mAP50 — середню точність за всіма класами за умови, що перекриття передбаченої та еталонної рамок за показником IoU (Intersection over Union) становить не менше 0.5. Показник IoU визначається як відношення площі перетину двох рамок до площі їх об'єднання. Метрика mAP50 обчислюється за формулою (2.15)

$$mAP50 = \frac{1}{|C|} \sum_{c \in C} AP_c \quad (2.15)$$

де C — множина класів, AP_c — середня точність для класу c . Наведемо числовий приклад інтерпретації метрик. Нехай на тестовій вибірці модель правильно виявила 95 сонних станів (TP), помилково спрацювала 3 рази (FP) і пропустила 2 реальні випадки (FN). Тоді $Precision = 95 / (95 + 3) \approx 0,969$, $Recall = 95 / (95 + 2) \approx 0,979$, а $F1 = 2 \cdot 0,969 \cdot 0,979 / (0,969 + 0,979) \approx 0,974$. Усі три значення є високими, причому повнота перевищує точність — це бажана для системи безпеки ситуація, оскільки кількість пропущених небезпечних станів ($FN = 2$) є мінімальною.

Цільовими значеннями для розроблюваної системи обрано $Precision \geq 0.90$, $Recall \geq 0.90$ та $mAP50 \geq 0.90$. Серед усіх метрик особливе значення для системи безпеки має Recall: пропуск реального випадку сонливості (false negative) є значно небезпечнішим, ніж хибна тривога (false positive), оскільки невиявлена втома може призвести до аварії, тоді як хибна тривога призводить лише до тимчасового дискомфорту водія. Тому при оцінці системи пріоритет надається максимізації повноти виявлення сонних станів, навіть якщо це супроводжується незначним зниженням точності.

Висновки до розділу 2

У розділі обґрунтовано вибір гібридної архітектури системи та описано її математичний апарат. Розглянуто класичні методи комп'ютерного зору — каскади Хаара на основі детектора Viola–Jones та метод EAR — і методи глибокого навчання, зокрема архітектуру YOLOv8. Детально викладено математичний апарат детектора Viola–Jones: інтегральне зображення,

обчислення ознак Хаара та алгоритм AdaBoost із каскадною структурою, з наведенням покрокових числових прикладів.

Порівняння методу EAR і нейромережевого підходу показало переваги YOLOv8 у стійкості до поворотів голови, вібрації та змін освітлення, що зумовило вибір саме нейромережевої класифікації стану ока. Описано архітектуру YOLOv8 (backbone–neck–head, anchor-free detection) та обґрунтовано вибір найлегшої конфігурації YOLOv8n як оптимальної для роботи на центральному процесорі у реальному часі. Описано двошаровий алгоритм накопичення індексу сонливості з конкретними параметрами та обґрунтовано асиметрію його оновлення на користь безпеки. Визначено й формалізовано метрики оцінки якості системи — Precision, Recall, F1 та mAP50 — та цільові значення цих метрик, а також пояснено пріоритет повноти для системи безпеки.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Технічні вимоги до системи

Перед описом програмної реалізації визначимо технічні вимоги до апаратного і програмного забезпечення, на якому функціонує система. Ключовою вимогою є можливість роботи на загальнодоступному обладнанні без спеціалізованих компонентів. Повний перелік вимог наведено у таблиці 3.1.

Таблиця 3.1 — Технічні вимоги до системи

Компонент	Вимога
Операційна система	Windows 10/11 або Linux (Ubuntu 20.04+)
Мова програмування	Python 3.9+
Оперативна пам'ять	Мінімум 4 ГБ (рекомендовано 8 ГБ)
Процесор	Будь-який сучасний CPU (Intel/AMD, 2+ ядра)
Відеокарта	Опціонально (NVIDIA з підтримкою CUDA для пришвидшення)
Камера	Будь-яка USB або вбудована веб-камера, мінімум 640×480
Простір на диску	Мінімум 2 ГБ (для залежностей та моделі)

Наведені вимоги є мінімальними і відповідають характеристикам типового офісного чи побутового персонального комп'ютера. Це підтверджує основну тезу роботи про доступність системи: для її розгортання не потрібно ані інфрачервоних камер, ані апаратних акселераторів.

Вимога до оперативної пам'яті у 4 ГБ зумовлена потребою одночасно тримати в пам'яті ваги моделі, буфери відеокадрів та робочі масиви бібліотек обробки зображень; рекомендовані 8 ГБ забезпечують комфортну роботу за

наявності інших застосунків. Відеокарта зазначена як опціональний компонент: за її відсутності інференс виконується на центральному процесорі, чого достатньо для роботи у реальному часі завдяки легкості моделі YOLOv8n, а за наявності сумісного з CUDA графічного прискорювача частота кадрів може бути додатково підвищена. Вимоги до камери є мінімальними — підійде будь-яка USB- або вбудована веб-камера з роздільною здатністю від 640×480, що є стандартом навіть для найдешевших пристроїв. Підтримка двох операційних систем (Windows та Linux) забезпечується кросплатформенністю мови Python та використаних бібліотек.

3.2 Вибір мови та середовища програмування. Опис бібліотек

Для реалізації системи обрано мову програмування Python, що є, де-факто, стандартом у галузі машинного навчання та комп'ютерного зору. Її перевагою є потужна екосистема спеціалізованих бібліотек, які покривають усі етапи розробки — від захоплення відео до навчання нейромереж. Нижче описано основні бібліотеки, використані у роботі.

Бібліотека OpenCV (модуль cv2) [11, 22] є основним інструментом комп'ютерного зору в системі. Вона застосовується для захоплення відеопотоку (cv2.VideoCapture), перетворення кольорових просторів (cv2.cvtColor), вирівнювання гістограми (cv2.equalizeHist), детекції обличчя та очей за допомогою каскадів Хаара (cv2.CascadeClassifier), відображення результатів на екрані (cv2.imshow, cv2.rectangle, cv2.putText) та збереження зображень кастомного датасету.

Фреймворк Ultralytics YOLOv8 [10] використовується для навчання та інференсу нейромережевої моделі. За його допомогою завантажується навчена модель (YOLO(model_path)), виконується класифікація стану ока (виклик

`model(processed_eye, conf=0.4))` та проводиться навчання моделі (`model.train(...)`).

Бібліотека NumPy забезпечує ефективні числові обчислення та роботу з багатовимірними масивами, які представляють пікселі зображень. Стандартний контейнер `collections.deque` — двостороння черга з обмеженою довжиною — використовується для реалізації ковзного вікна `eye_history = deque(maxlen=5)`. Він забезпечує сталий час $O(1)$ для додавання елементів та автоматично видаляє найстаріший запис при перевищенні заданої довжини, що ідеально відповідає логіці мажоритарного голосування.

Вибір саме цього стеку технологій є обґрунтованим з кількох міркувань:

- 1) усі перелічені бібліотеки є відкритими, безкоштовними та широко підтримуваними спільнотою, що знижує вартість розробки і ризики залежності від конкретного постачальника;
- 2) OpenCV та Ultralytics YOLOv8 написані з використанням оптимізованих низькорівневих реалізацій (C++ та CUDA відповідно), тому, попри високорівневий інтерфейс мовою Python, забезпечують швидкодію, достатню для роботи у реальному часі;
- 3) такий стек є, де-факто, стандартом у галузі прикладного комп'ютерного зору, що полегшує подальшу підтримку і розвиток системи іншими розробниками.

3.3 Формування навчального датасету

3.3.1 Базовий датасет

Основою навчальної вибірки слугував відкритий датасет `driver drowsiness1` з платформи Roboflow [17], розповсюджуваний за ліцензією CC BY 4.0. Він містить 5 569 зображень, розподілених на навчальну (4 704), валідаційну (1 083) та тестову (782) підмножини. Розмітка виконана у форматі YOLO v5 PyTorch — текстові файли з нормалізованими координатами. Датасет містить два класи: 0 — `drowsy` (сонний стан) та 1 — `not_drowsy` (активний стан).

При публікації до зображень датасету застосовано препроцесинг: автоматичну орієнтацію з видаленням EXIF-повороту, масштабування до розміру 640×640 пікселів та адаптивне вирівнювання контрасту (CLAHE). Крім того, виконано аугментацію: горизонтальне відображення з імовірністю 50%, повороти на 90°, випадкове кадрування в межах 0–20%, поворот на $\pm 15^\circ$, зміну яскравості на $\pm 15\%$, зміну експозиції на $\pm 10\%$ та розмиття Гауса в межах 0–2,5 пікселя. Така аугментація підвищує різноманітність даних і стійкість моделі до варіацій умов зйомки.

Розподіл базового датасету на три підмножини виконано у пропорції, близькій до 85/15 для пари «навчання + валідація» та тестування, що є типовою практикою для задач навчання нейромереж. Навчальна підмножина (4 704 зображення) безпосередньо використовується для оновлення ваг моделі під час навчання; валідаційна (1 083 зображення) слугує для контролю якості наприкінці кожної епохи і дозволяє виявити перенавчання; тестова (782 зображення) застосовується для остаточної неупередженої оцінки навченої моделі на даних, які вона жодного разу не «бачила». Наявність двох класів —

drowsy і not_drowsy — робить задачу бінарною, що відповідає кінцевій меті системи: розрізнити сонний і активний стан ока.

3.3.2 Конвеєр напіваавтоматичного збору кастомних даних

Для адаптації моделі до реальних умов роботи розроблено конвеєр напіваавтоматичного збору власних навчальних даних, інтегрований безпосередньо у програму. Опишемо цей конвеєр покроково — від збереження зображення до появи розміченого файлу у гібридному датасеті.

Перший крок — збереження зображення з програми. Під час роботи системи каскад Хаара знаходить обличчя, вирізає зону очей і для кожного виявленого ока викликає функцію `preprocess_for_yolo`, яка виконує переведення у відтінки сірого, вирівнювання гістограми та повернення у формат BGR. Натискання клавіші 1 зберігає поточний кадр ока до папки `custom_dataset/not_drowsy/`, клавіші 2 — до `custom_dataset/drowsy/`. Ім'я файлу формується як `eye_<мітка_часу_мс>.jpg`, що гарантує унікальність. У терміналі виводиться підтвердження збереження. Як показано на рисунку 3.1, збережені зображення є препроцесованими вирізками очей з рівномірним контрастом.

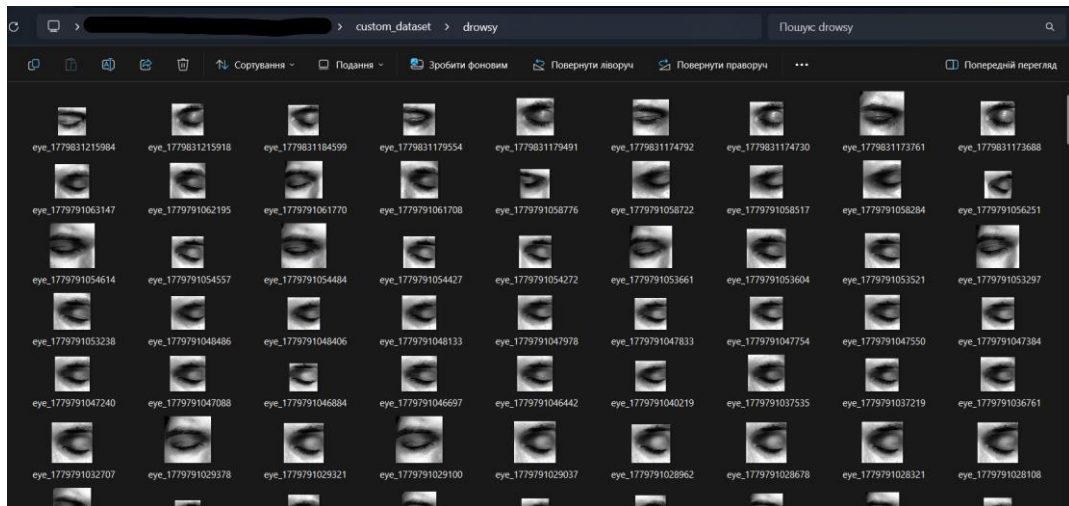


Рисунок 3.1 — Приклади зображень кастомного датасету класу drowsy після препроцесингу

Ключовою перевагою конвеєру є те, що зображення, які потрапляють у датасет, проходять точно ту саму послідовність обробки, що й кадри під час реального інференсу. Це усуває зсув розподілу (domain shift) між навчальними та робочими даними — модель навчається саме на тих ознаках, які бачитиме у роботі.

Другий крок — автоматична розмітка і об'єднання. Після накопичення достатньої кількості зображень запускається скрипт `merge_custom_dataset.py`. Він перемішує зображення кожного класу випадково (`random.shuffle`), ділить їх у пропорції 80/20 на навчальну та валідаційну підмножини, копіює до відповідних директорій `dataset/raw/train(valid)/images/` і для кожного файлу автоматично генерує текстову мітку у форматі YOLO. Оскільки зображення ока вже є тісним вирізом після роботи каскаду Хаара, обмежувальна рамка задається на весь кадр: клас, центр (0.5, 0.5), ширина і висота 1.0. Окрема локалізація не потрібна, бо вирізку ока вже зроблено на попередньому етапі.

Структуру гібридного датасету після виконання скрипту, де зображення кастомного датасету за іменами виду `eye_*.jpg` розміщено поряд зі зображеннями базового датасету, а в директорії `labels` містяться відповідні текстові файли розмітки, наведено на рисунках 3.2 та 3.3.

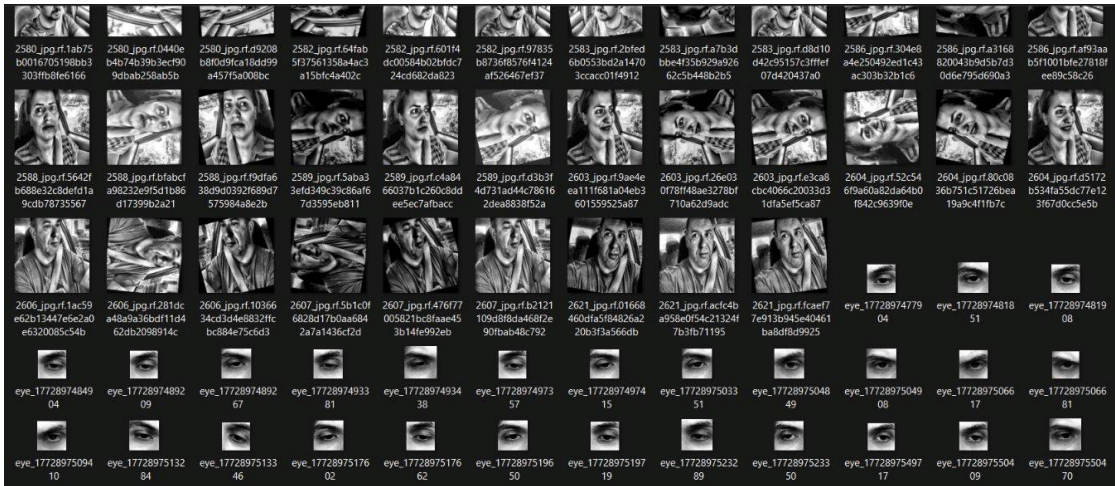


Рисунок 3.2 — Структура гібридного датасету після виконання `merge_custom_dataset.py`

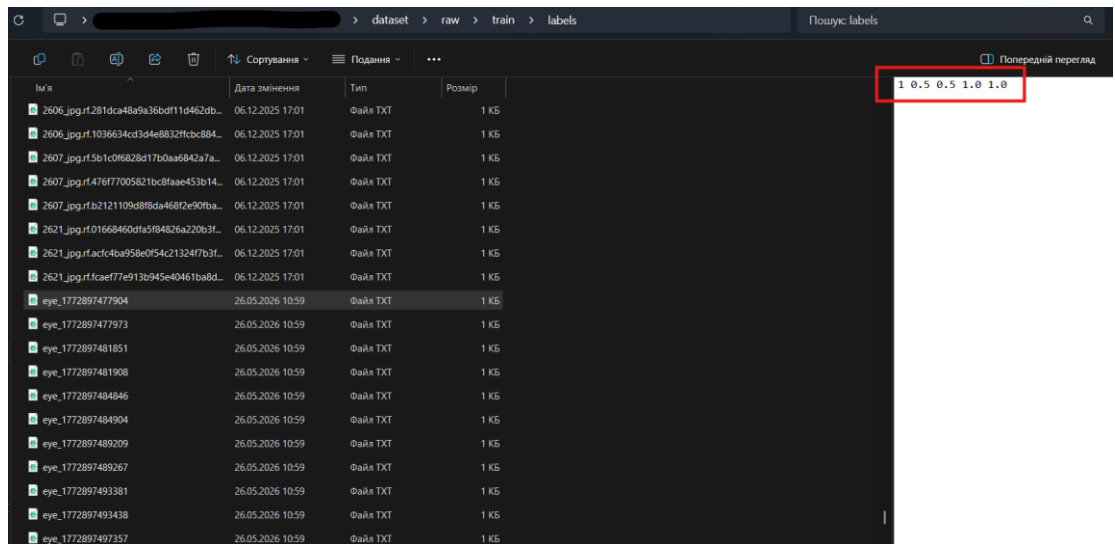


Рисунок 3.3 — Згенерований автоматично файл розмітки одного з зображень після виконання `merge_custom_dataset.py`

Результатом роботи конвеєра стало 426 зображень класу `drowsy` та 389 зображень класу `not_drowsy`, разом 815 кастомних зображень, з яких 652 потрапили до навчальної підмножини і 163 — до валідаційної.

Розглянемо принцип автоматичної розмітки докладніше. У задачах детекції об'єктів розмітка зазвичай є трудомісткою, оскільки потребує ручного окреслення обмежувальної рамки навколо кожного об'єкта на кожному зображенні. Однак у розробленому конвеєрі цей етап повністю автоматизовано завдяки особливості вхідних даних: зображення, що

зберігаються, вже є тісною вирізкою ока, отриманою каскадом Хаара, тому шуканий об'єкт займає практично весь кадр. Це дозволяє задати обмежувальну рамку формально — з центром у точці (0.5, 0.5) та шириною і висотою 1.0 у нормалізованих координатах, тобто на весь кадр. Скрипт `merge_custom_dataset.py` генерує відповідний текстовий файл розмітки для кожного зображення автоматично, що повністю усуває ручну працю і виключає пов'язані з нею помилки. Саме тому YOLOv8, формально будучи детектором, у цій системі функціонує як класифікатор стану заздалегідь локалізованого ока.

3.3.3 Аугментація та об'єднання датасетів

На етапі навчання до об'єднаного датасету застосовано вбудовану аугментацію фреймворку Ultralytics YOLOv8, параметри якої наведено у таблиці 3.2. Аугментація розширює різноманітність навчальних даних і знижує ризик перенавчання.

Таблиця 3.2 — Параметри аугментації навчального датасету
(налаштування Ultralytics YOLOv8)

Тип аугментації	Параметр	Мета
HSV-Hue	hsv_h = 0.015	Варіація відтінку для різних освітлювачів
HSV-Saturation	hsv_s = 0.7	Адаптація до різного насичення кольору
HSV-Value	hsv_v = 0.4	Імітація денних та нічних умов
Горизонтальне відображення	flip_lr = 0.5	Симетрія між лівим та правим оком
Mosaic	mosaic = 1.0	Різноманітність фону
MixUp	mixup = 0.1	Регуляризація
Афінні трансформації	degrees=10°, translate=0.1, scale=0.5	Імітація нахилу голови і відстані до камери

Загальний обсяг навчальних даних після об'єднання склав 5 569 базових зображень з Roboflow та 815 кастомних, разом 6 384 зображення. Поєднання великого базового датасету з власними даними, зібраними у форматі реального інференсу, є ключем до досягнення високої точності моделі в умовах експлуатації.

3.4 Навчання моделі та domain adaptation

Навчання моделі виконано за принципом перенесення навчання (transfer learning) [12]: за стартові обрано ваги YOLOv8n, попередньо навчені на датасеті COCO [10, 14]. Навчання проводилося у середовищі Google Colab на графічному прискорювачі NVIDIA Tesla T4 (16 ГБ) протягом 50 епох. Перенесення навчання дозволяє використати загальні візуальні ознаки, вже

вивчені моделлю на великому датасеті, і донавчити її під конкретну задачу класифікації стану ока. Гіперпараметри навчання наведено у таблиці 3.3.

Таблиця 3.3 — Гіперпараметри навчання

Гіперпараметр	Значення	Обґрунтування
Кількість епох	50	Достатньо для виходу на плато якості
Розмір батчу	16	Компроміс між стабільністю та пам'яттю T4
Розмір вхідного зображення	640×640	Стандартний для YOLOv8, баланс точності та швидкості
Оптимізатор	SGD (momentum=0.937)	Базова конфігурація Ultralytics
Початкова швидкість навчання	0.01	Стандартне для fine-tuning
Розклад швидкості навчання	cosine, кінцева 0.0001	Стабілізація наприкінці
Weight decay	0.0005	L2-регуляризація проти перенавчання

Застосування domain adaptation — донавчання моделі на гібридному датасеті, що включає власні зображення у форматі реального інференсу, — дало суттєве покращення всіх метрик якості. Порівняння показників базової моделі та моделі після domain adaptation наведено у таблиці 3.4.

Таблиця 3.4 — Порівняння метрик до і після domain adaptation

Метрика	Базова модель (без DA)	Після domain adaptation	Зміна
Precision	0.85	0.98	+0.13
Recall	0.95	0.99	+0.04
F1-score	0.90	0.98	+0.08
mAP50	0.96	0.99	+0.03

Результати демонструють покращення всіх метрик на цільових (власних) даних. Найбільший ефект — зростання точності (Precision) з 0.85 до

0.98: базова модель, навчена лише на відкритому датасеті, давала помітну частку хибних спрацювань на зображеннях з реальної камери, і саме domain adaptation практично усуває їх. Повнота (Recall) зросла з 0.95 до 0.99, а усереднена точність mAP50 — з 0.96 до 0.99. Висока повнота є критичною для системи безпеки, оскільки означає, що система практично не пропускає реальних випадків сонливості. Досягнуті значення (Precision 0.98, Recall 0.99, mAP50 0.99) перевищують цільові орієнтири (≥ 0.90 , див. підрозділ 2.6).

Покращення обумовлене усуненням зсуву розподілу (domain shift). Базова модель навчалася на зображеннях загального датасету, умови зйомки яких — освітлення, ракурс, роздільна здатність, характер препроцесингу — відрізняються від тих, що виникають під час реальної роботи системи на конкретній веб-камері. Донавчання на власних зображеннях, зібраних безпосередньо через робочий пайплайн (з тим самим переведенням у відтінки сірого та вирівнюванням гістограми), «переналаштовує» модель на фактичний розподіл вхідних даних. Унаслідок цього модель перестає робити систематичні помилки, характерні для невідповідності між навчальними та робочими умовами, і досягає майже ідеальної точності на цільовому обладнанні.

3.5 Опис алгоритму роботи програмного продукту

При запуску програма виконує ініціалізацію компонентів: завантажує два каскади Хаара з файлів `haarcascade_frontalface_default.xml` та `haarcascade_eye.xml`, завантажує навчену модель `model_v2_custom.pt` через інтерфейс Ultralytics YOLO, відкриває веб-камеру викликом `cv2.VideoCapture(0)` та ініціалізує структури `eye_history = deque(maxlen=5)` і `drowsiness_score = 0`.

Після ініціалізації запускається головний цикл покадрової обробки відео, який реалізує таку послідовність дій. Узагальнену блок-схему алгоритму наведено на рисунку 3.4.

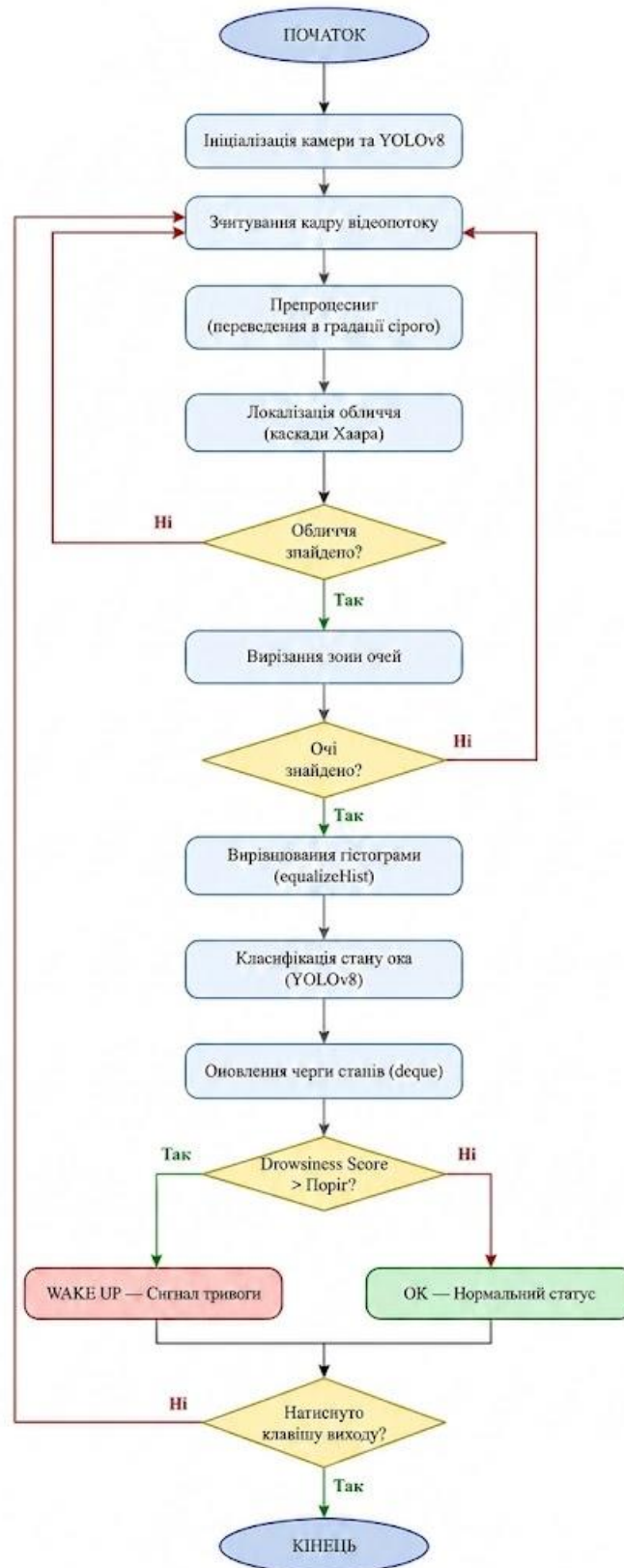


Рисунок 3.4 — Блок-схема алгоритму роботи програмного продукту

Послідовність кроків головного циклу така:

- 1) зчитування чергового кадру з камери у форматі BGR викликом `cap.read()`;
- 2) конвертація кадру у відтінки сірого та детекція обличчя викликом `detectMultiScale` з параметрами `scaleFactor=1.3`, `minNeighbors=5`, `minSize=(100,100)`, а за відсутності обличчя — перехід до наступного кадру;
- 3) виділення верхніх 60% області обличчя, де фізично розташовані очі, що прискорює подальшу детекцію;
- 4) детекція очей у виділеній зоні з параметрами `scaleFactor=1.05`, `minNeighbors=3`, `minSize=(30,30)`; дрібніший крок масштабування зумовлений меншим розміром очей;
- 5) для кожного знайденого ока — вирізка зображення, препроцесинг функцією `preprocess_for_yolo`, інференс YOLO з порогом впевненості `conf=0.4` та зчитування класу і впевненості;
- 6) визначення статусу кадру: якщо хоча б одне око класифіковане як `not_drowsy`, кадр вважається активним; якщо всі виявлені очі класифіковані як `drowsy`, кадр вважається сонним;
- 7) оновлення ковзного вікна `eye_history`, мажоритарне голосування та оновлення лічильника `drowsiness_score`;
- 8) перевірка порогу `ALARM_THRESHOLD` та відображення попередження за його досягнення;
- 9) відображення смуги втоми, поточного статусу та рамок навколо обличчя й очей.

Окремої уваги заслуговує функція препроцесингу `preprocess_for_yolo`. Вона перетворює вхідне зображення з простору BGR у відтінки сірого, застосовує вирівнювання гістограми `cv2.equalizeHist` і повертає результат назад у простір BGR для подачі в модель. Вирівнювання гістограми перерозподіляє яскравість так, щоб її розподіл наближався до рівномірного: темні зображення стають яскравішими, а пересвічені — темнішими. Це

забезпечує програмну стійкість системи до змін освітлення без застосування спеціальної інфрачервоної камери.

Технічно вирівнювання гістограми буде кумулятивну функцію розподілу інтенсивностей пікселів і використовує її як перетворювальну криву, що «розтягує» діапазон яскравостей на всю доступну шкалу від 0 до 255. Повернення зображення у трьохканальний формат BGR після обробки є необхідним, оскільки модель YOLOv8 очікує на вході трьохканальне зображення; при цьому всі три канали стають ідентичними, несучи однакову яскравісну інформацію. Важливо, що та сама функція застосовується як під час збору навчального датасету, так і під час реального інференсу, завдяки чому модель завжди працює з однаково обробленими даними.

Важливою особливістю логіки є правило «хоча б одне відкрите око». У реальних умовах одне око може бути відсутнім у кадрі через поворот голови, затінення чи перекриття оправою окулярів. Система реалізує консервативну логіку: якщо хоча б одне виявлене око класифіковане як `not_drowsy`, кадр вважається активним і лічильник сонливості не збільшується. Тривога спрацьовує лише тоді, коли всі виявлені очі одночасно класифіковані як `drowsy`. Це запобігає хибним спрацюванням при частковій видимості обличчя.

Таке проєктне рішення є компромісом на користь зниження частоти хибних тривог. Альтернативна, більш сувора логіка — коли тривога активується, якщо хоча б одне око виглядає сонним — призводила б до численних хибних спрацювань у ситуаціях, коли одне око тимчасово перекрите або погано видиме. Оскільки сонливість фізіологічно проявляється симетрично (заплющуються обидва ока одночасно), вимога одночасної класифікації обох очей як сонних є достатньою для надійного виявлення реальної втоми і водночас стійкою до часткової видимості.

Програма містить вбудований інструмент збору даних, керований клавішами. Натискання клавіші 1 зберігає поточне зображення ока як приклад класу `not_drowsy`, клавіші 2 — як приклад класу `drowsy`, а клавіші Q — завершує роботу програми. Цей механізм дозволяє безпосередньо під час

експлуатації поповнювати кастомний датасет новими прикладами для подальшого донавчання моделі.

3.6 Результати та аналіз

Розглянемо результати навчання моделі та роботи системи. У процесі навчання відстежувалися три складові тренувальних втрат: `box_loss`, що відповідає за точність локалізації рамок і знизилася з 1.27 до 0.78; `cls_loss`, що відповідає за класифікацію і знизилася з 2.30 до 0.40; та `dfl_loss` (distribution focal loss), що знизилася з 1.68 до 1.32. Усі складові стабільно спадають протягом навчання; незначне підвищення на епохах 40–41 пояснюється зміною режиму аугментації та загасанням швидкості навчання. Метрики якості виходять на плато вже після приблизно 15-ї епохи, досягаючи значень $\text{Precision} \approx 0.99$, $\text{Recall} \approx 0.99$ та $\text{mAP50} \approx 0.99$ за $\text{mAP50-95} \approx 0.66$. Графіки функцій втрат і метрик протягом навчання наведено на рисунку 3.5.

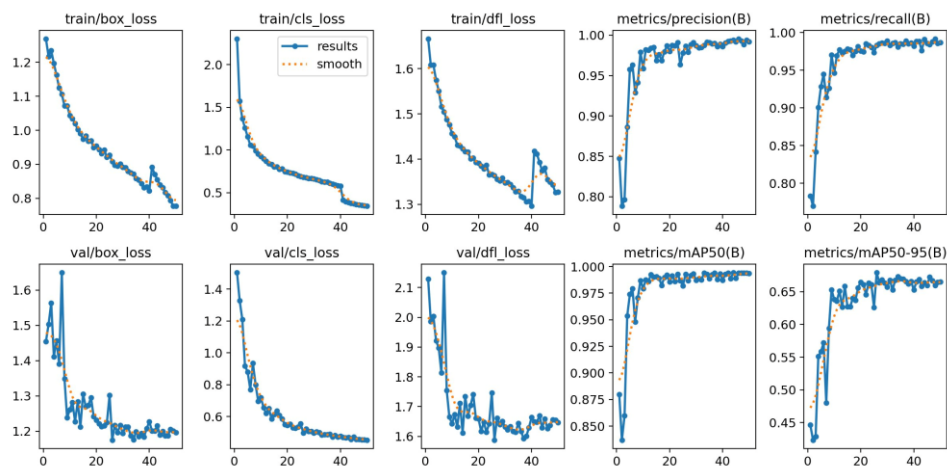


Рисунок 3.5 — Графіки функцій втрат і метрик якості протягом навчання моделі

Якість класифікації за класами зручно аналізувати за нормалізованою матрицею плутанини, наведеною на рисунку 3.6. Клас `drowsy` розпізнається з

точністю 0.99 — 99% реально сонних станів класифікуються правильно. Клас `not_drowsy` розпізнається з точністю 0.98 — лише 2% зображень відкритих очей помилково класифікуються як сонні.

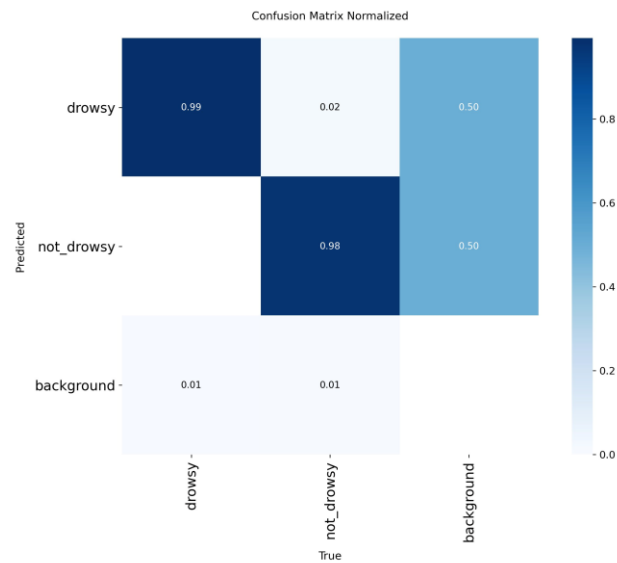


Рисунок 3.6 — Нормалізована матриця плутанини моделі на валідаційній вибірці

Робота системи в різних режимах ілюструється екранними знімками інтерфейсу. У нормальному режимі роботи у верхньому лівому куті відображається зелений напис `STATUS: ACTIVE`, навколо виявлених очей малюються зелені прямокутники з підписом `OK` та значенням впевненості моделі, а смуга втоми (`Fatigue`) внизу кадру майже порожня. Знайдене обличчя позначається синім прямокутником. Цей режим ілюструє рисунок 3.7.

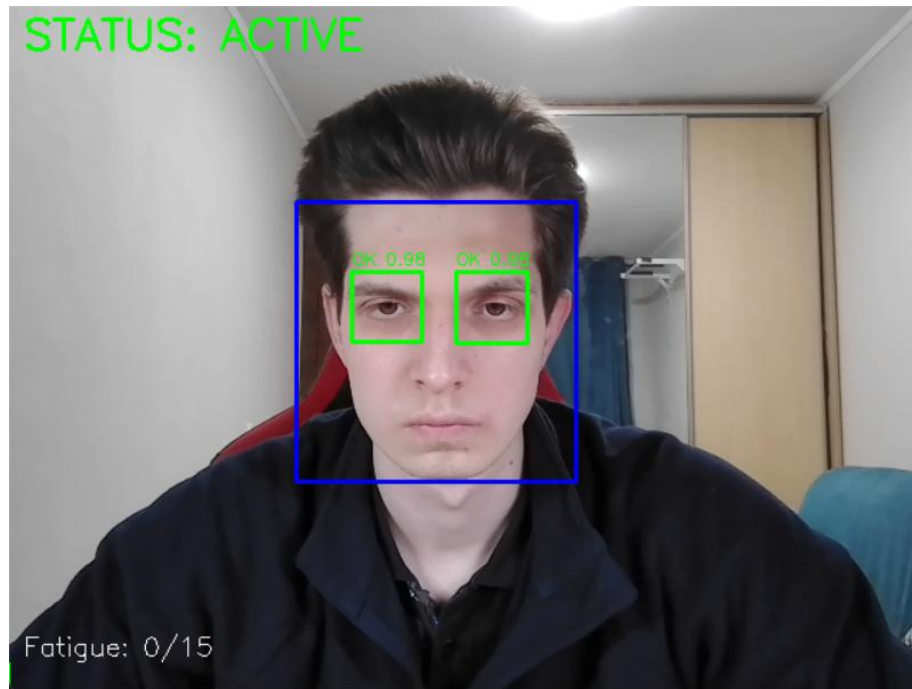


Рисунок 3.7 — Інтерфейс системи в режимі активного стану водія

При виявленні сонливості статус змінюється на червоний напис STATUS: DROWSY, рамки навколо очей стають червоними з підписом SLEEP, а смуга Fatigue заповнюється пропорційно до накопиченого значення drowsiness_score. Поточне значення лічильника відображається у форматі Fatigue: X/15. Цей режим показано на рисунку 3.8.

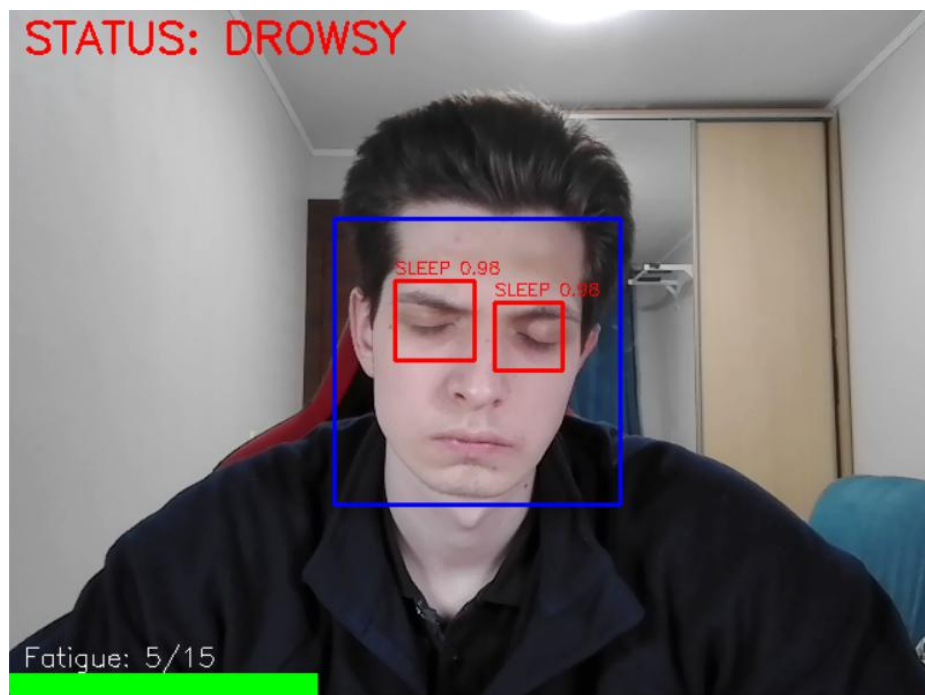


Рисунок 3.8 — Інтерфейс системи при виявленні ознак сонливості

При досягненні лічильником `drowsiness_score` значення 15 (`ALARM_THRESHOLD`) з'являється великий червоний напис **WAKE UP!!!**, а смуга `Fatigue` повністю заповнюється червоним кольором. Система залишається у стані тривоги доти, доки значення лічильника не знизиться нижче порогу завдяки відновленню з темпом `RECOVERY_RATE` = 2. Спрацювання сигналу тривоги ілюструє рисунок 3.9.



Рисунок 3.9 — Інтерфейс системи при спрацюванні сигналу тривоги

Окремо проілюструємо роботу правила «хоча б одне відкрите око». На рисунку 3.10 показано ситуацію, коли одне око позначене зеленою рамкою ОК, а друге або відсутнє у кадрі, або позначене червоною рамкою SLEEP. Попри це статус залишається `ACTIVE`, а смуга `Fatigue` не зростає, що демонструє стійкість системи до часткової видимості очей у реальних умовах водіння.

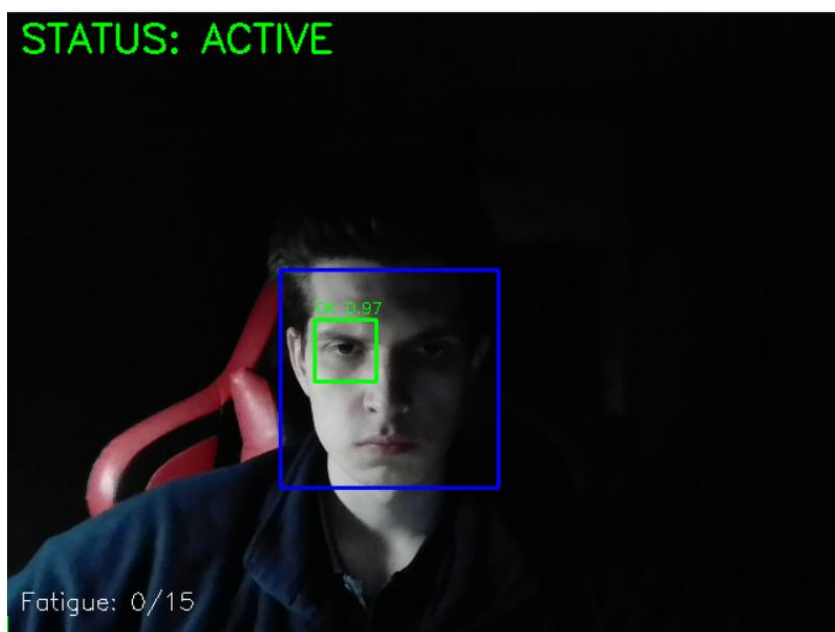


Рисунок 3.10 — Робота системи при частковій видимості очей: статус активний за наявності хоча б одного відкритого ока

Аналіз продуктивності показав, що на ноутбучі з процесором AMD Ryzen 5 5600H без графічного прискорювача система обробляє відеопотік зі швидкістю близько 8–9 кадрів на секунду. Загальний час обробки одного кадру становить близько 110–125 мс, основну частку якого становить інференс нейромережі YOLOv8n, що виконується для кожного з виявлених очей. Ці показники підтверджують, що система задовольняє вимогу роботи у режимі реального часу (не менше 6 FPS), визначену у постановці задачі: оскільки сонливість є повільним явищем, а сигнал тривоги накопичується приблизно за 1,5–2 секунди, система виявляє ознаки засинання значно швидше за тривалість мікросну (2–30 секунд).

Сумарний час обробки кадру складається переважно з двох внесків — локалізації обличчя та зони очей каскадом Хаара і класифікації стану ока нейромережею. Двоетапна гібридна архітектура дозволяє оптимізувати кожен етап окремо: каскад Хаара швидко відсікає кадри без обличчя, не запускаючи на них витратний нейромережевий інференс, а класифікація YOLOv8n виконується лише на невеликих вирізках очей, а не на повному кадрі. Завдяки цьому досягнутої швидкодії достатньо для практичного застосування навіть

на бюджетному обладнанні без графічного прискорювача, а за наявності GPU з підтримкою CUDA частота кадрів може бути додатково підвищена.

Висновки до розділу 3

У розділі описано програмну реалізацію системи. Визначено технічні вимоги, що відповідають характеристикам типового персонального комп'ютера. Обґрунтовано вибір мови Python та описано стек використаних бібліотек — OpenCV, Ultralytics YOLOv8, NumPy та collections.deque. Сформовано гібридний навчальний датасет обсягом 6 384 зображення, що поєднує базовий датасет Roboflow з власними даними, зібраними у форматі реального інференсу. Проведено навчання моделі YOLOv8n із застосуванням domain adaptation, що дало значення $mAP50 = 0.99$, $Precision = 0.98$ та $Recall = 0.99$, які суттєво перевищують цільові орієнтири. Детально описано алгоритм роботи програмного продукту, зокрема функцію препроцесингу та консервативну логіку визначення статусу кадру. Проаналізовано результати навчання й роботи системи та підтверджено її функціонування у режимі реального часу.

4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

У даному розділі виконується техніко-економічне обґрунтування розробки інтелектуальної системи виявлення сонливості водія. Метод функціонально-вартісного аналізу (ФВА) застосовується для виявлення оптимального співвідношення «якість/вартість» серед альтернативних архітектурних рішень на ранніх етапах проєктування. Аналіз проводиться у такій послідовності: постановка задачі, обґрунтування функцій програмного продукту, визначення параметрів якості, експертне оцінювання вагомості параметрів, розрахунок технічного рівня варіантів, економічний аналіз витрат та вибір кращого варіанту за техніко-економічним рівнем.

4.1 Постановка задачі проєктування

Метою розробки є створення інтелектуальної системи виявлення ознак сонливості водія у режимі реального часу на базі гібридної архітектури, що поєднує каскади Хаара для локалізації обличчя та зони очей із неймережевою моделлю YOLOv8n для класифікації стану ока. Система повинна функціонувати на стандартному персональному комп'ютері без спеціалізованого обладнання, а саме без графічного прискорювача та інфрачервоних сенсорів.

До програмного продукту висуваються такі вимоги:

- 1) висока точність класифікації стану ока та стійкість до зміни умов освітлення;

- 2) функціонування у режимі реального часу з частотою не менше 6 кадрів на секунду на CPU без використання GPU;
- 3) робота на стандартній веб-камері без спеціалізованих інфрачервоних сенсорів;
- 4) стійкість до поодиноких хибних класифікацій, у тому числі при морганні;
- 5) кросплатформенність — підтримка операційних систем Windows та Linux.

4.2 Обґрунтування функцій програмного продукту

Головною функцією F0 програмного продукту визначено розробку інтелектуальної системи виявлення ознак сонливості водія за відеопотоком з веб-камери у реальному часі. Для реалізації головної функції виділено чотири підфункції, кожна з яких допускає кілька варіантів реалізації.

F1 — вибір мови програмування:

- 1) Python — стандарт у галузі машинного навчання та комп'ютерного зору;
- 2) C++ — висока швидкодія, проте складна інтеграція моделей машинного навчання.

F2 — метод локалізації обличчя та зони очей:

- 1) каскади Хаара з бібліотеки OpenCV — швидка робота на CPU, готові класифікатори;
- 2) нейромережа MTCNN — висока точність детекції, потребує GPU для роботи у реальному часі;
- 3) HOG-детектор бібліотеки Dlib — стабільність, проте нижча швидкодія порівняно з каскадами Хаара.

F3 — стратегія навчання моделі класифікації стану ока:

- 1) YOLOv8n із застосуванням domain adaptation — донавчання на кастомному датасеті, висока надійність на цільових даних (precision 0.98, mAP50 0.99);
- 2) YOLOv8n без додаткового донавчання — використання предтренованої моделі «з коробки», нижча надійність на цільових даних: більше хибних спрацювань (precision ≈ 0.85);
- 3) згорткова неймережа власної архітектури — повний контроль, проте велика трудомісткість розробки;
- 4) MobileNetV2 — легкий класифікатор, без вбудованої локалізації об'єкта.

F4 — алгоритм формування рішення про сонливість:

- 1) двошаровий алгоритм накопичення з мажоритарним голосуванням на основі deque та асиметричним лічильником (запропонована реалізація);
- 2) покадрове рішення без накопичення (базовий підхід).

Морфологічну карту системи з усіма варіантами реалізації функцій F1—F4 наведено на рисунку 4.1.

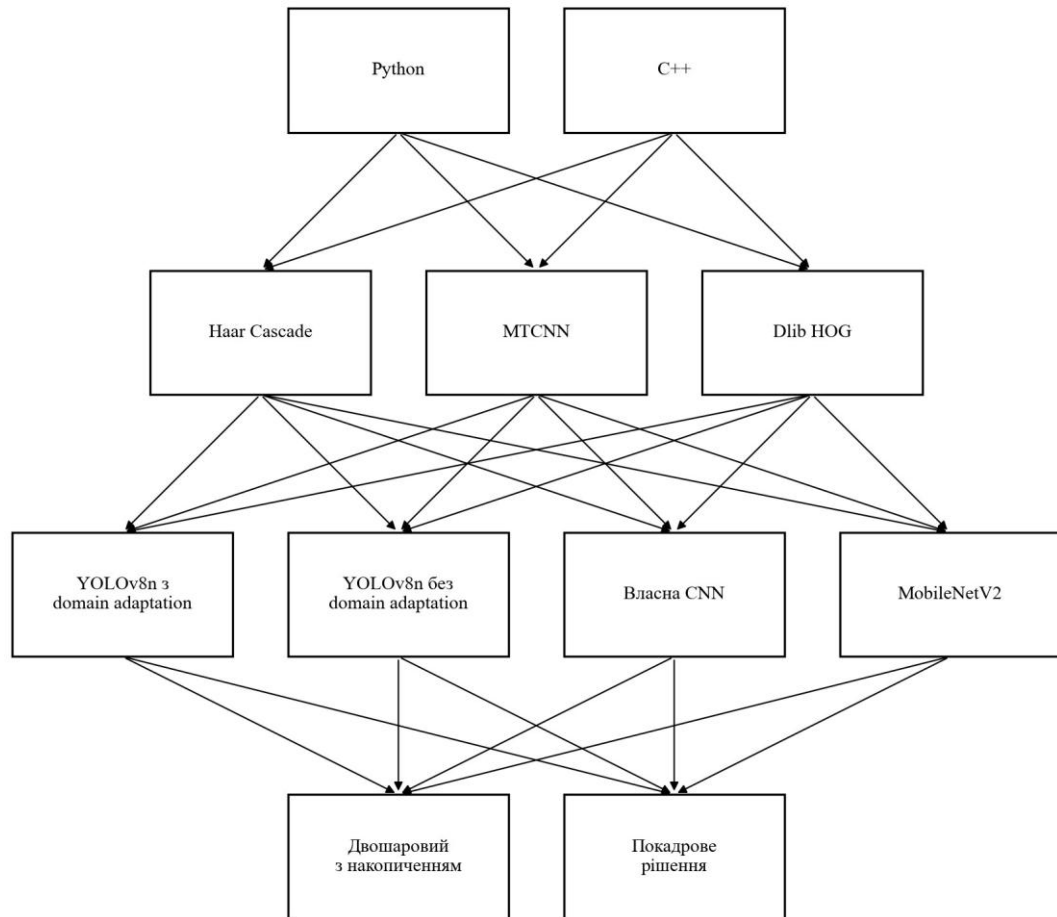


Рисунок 4.1 — Морфологічна карта системи виявлення сонливості водія з варіантами реалізації функцій F1–F4

На основі морфологічної карти проведено аналіз варіантів реалізації функцій за допомогою позитивно-негативної матриці. Результати наведено у таблиці 4.1.

Таблиця 4.1 — Позитивно-негативна матриця варіантів реалізації функцій

Функція	Варіант	Переваги	Недоліки
F1	1) Python	Велика екосистема ML/CV-бібліотек, простота розробки	Нижча швидкодія порівняно з C++
F1	2) C++	Висока швидкодія	Складна інтеграція моделей ML, довший цикл розробки
F2	1) Haar Cascade	Швидка робота на CPU (5–15 мс), готові класифікатори OpenCV	Чутливість до поворотів голови та різких змін освітлення
F2	2) MTCNN	Висока точність детекції	Потребує GPU для роботи у реальному часі
F2	3) Dlib HOG	Стабільність детекції	Нижча швидкодія порівняно з Haar Cascade
F3	1) YOLOv8n з DA	Висока точність ($mAP50 \approx 0.99$) на цільових даних після донавчання	Трудомістке навчання, потреба у кастомному датасеті
F3	2) YOLOv8n без DA	Готова модель, швидке розгортання, мінімальні витрати на підготовку	Нижча точність ($precision \approx 0.85$) на специфічних даних
F3	3) CNN власної архітектури	Повний контроль над архітектурою	Велика трудомісткість розробки та навчання
F3	4) MobileNetV2	Легка модель	Потребує окремої локалізації об'єкта, нижча точність
F4	1) Двошаровий з накопиченням	Стійкість до хибних спрацювань при морганні	Дещо складніша логіка реалізації
F4	2) Покадрове рішення	Проста реалізація	Хибні спрацювання при морганні

На основі аналізу позитивно-негативної матриці виділено два варіанти реалізації для подальшого порівняння за технічним та економічним критеріями.

Варіант 1 (B1) — запропонована реалізація: F1(1) Python + F2(1) Haar Cascade + F3(1) YOLOv8n з domain adaptation + F4(1) двошаровий алгоритм накопичення.

Варіант 2 (B2) — базовий підхід: F1(1) Python + F2(1) Haar Cascade + F3(2) YOLOv8n без domain adaptation + F4(2) покадрове рішення без накопичення.

Обидва варіанти використовують Python (F1) та каскади Хаара (F2), а відрізняються за функціями F3 (наявність чи відсутність domain adaptation) та F4 (двошаровий алгоритм накопичення або покадрове рішення). Така постановка дозволяє коректно зіставити внесок цих ключових архітектурних рішень у якість та вартість розробки.

4.3 Обґрунтування системи параметрів програмного продукту

Для оцінювання якості варіантів реалізації виділено чотири параметри.

X1 — точність класифікації стану ока, бал (шкала 1–10). Параметр відображає здатність системи коректно класифікувати стан кожного ока як drowsy або not_drowsy і базується на метриці mAP50, переведеній у бальну шкалу.

X2 — час обробки одного кадру, мс. Параметр визначає швидкодію системи у режимі реального часу.

X3 — обсяг оперативної пам'яті, що споживається моделлю, МБ. Параметр характеризує ресурсомісткість програмного продукту.

X4 — стійкість до зміни умов експлуатації, бал (шкала 1–10). Враховує адаптацію до зміни освітлення та стійкість до поодиноких хибних класифікацій.

Числові межі параметрів наведено у таблиці 4.2.

Таблиця 4.2 — Основні параметри програмного продукту

Параметр	Одиниця	Гірше значення	Середнє значення	Краще значення
X1 – точність	бал	3	6	10
X2 – час обробки кадру	мс	80	45	20
X3 – обсяг ОЗП	МБ	200	100	30
X4 – стійкість	бал	3	6	10

За даними таблиці 4.2 будуються графічні характеристики параметрів (як показано далі), за якими визначаються бальні оцінки кожного варіанту реалізації на основі фактичних значень параметрів. Графічна характеристика параметра X1 (точність класифікації) наведена на рисунку 4.2.

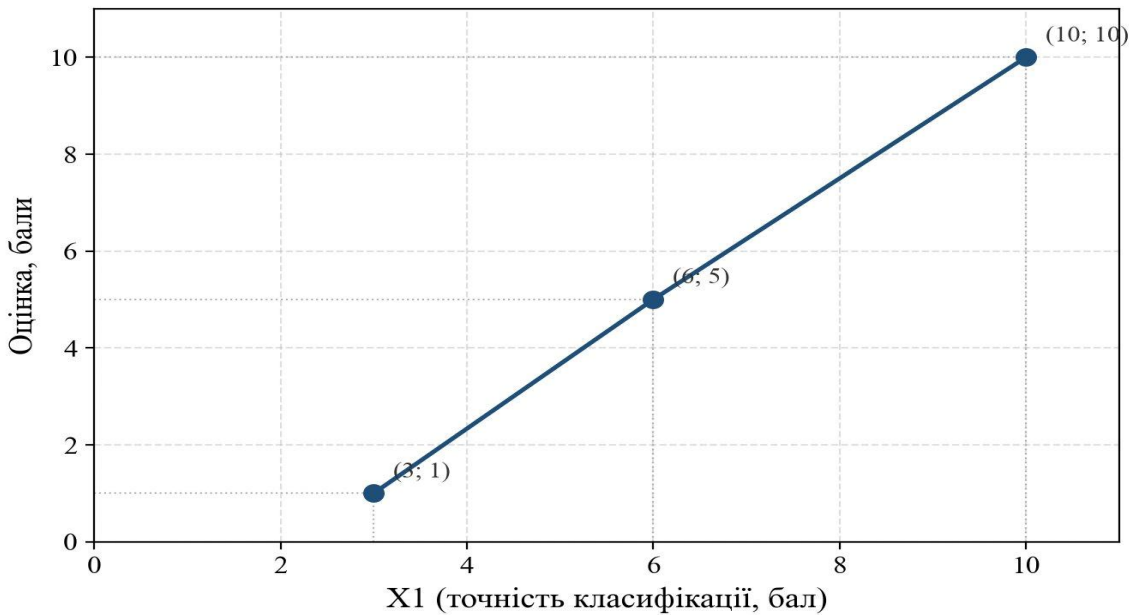


Рисунок 4.2 — Графічна характеристика параметра X1 (точність класифікації, бали)

Графічна характеристика параметра X2 (час обробки одного кадру) показана на рисунку 4.3.

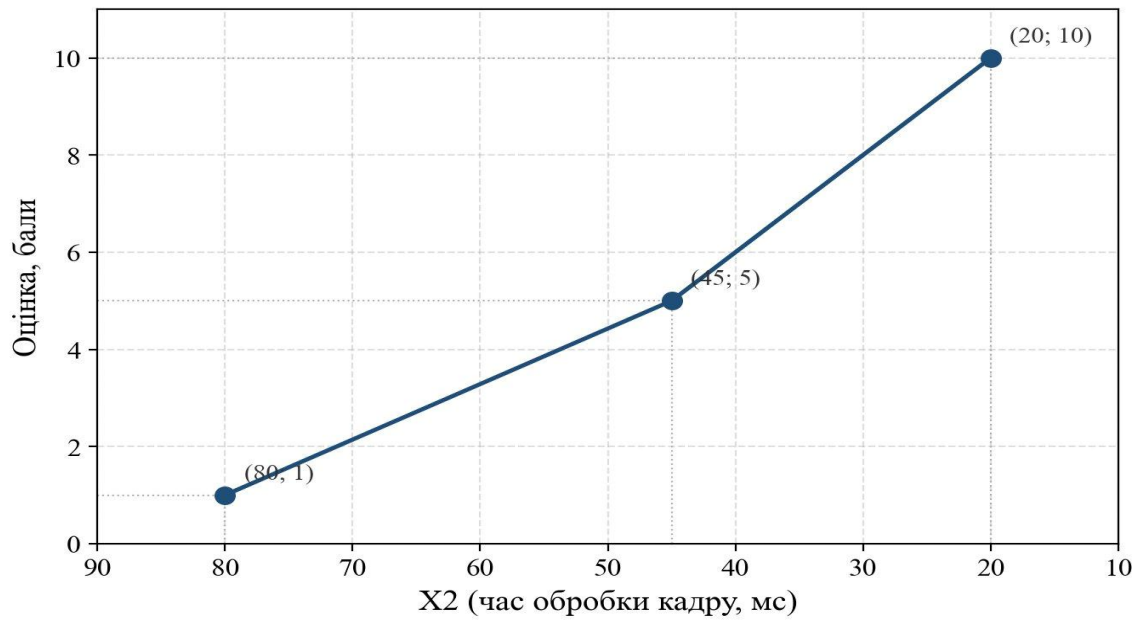


Рисунок 4.3 — Графічна характеристика параметра X2 (час обробки кадру, мс)

Графічна характеристика параметра X3 (обсяг ОЗП, що споживається моделлю) наведена на рисунку 4.4.

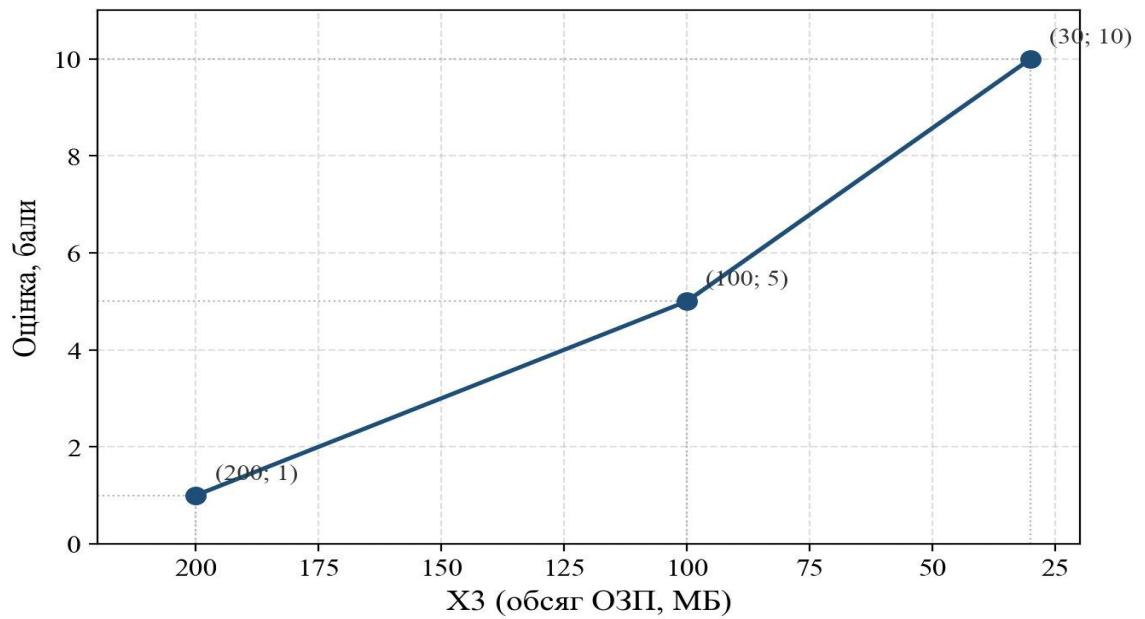


Рисунок 4.4 — Графічна характеристика параметра X3 (обсяг ОЗП, МБ)

Графічна характеристика параметра X4 (стійкість до зміни умов експлуатації) показана на рисунку 4.5.

Перевіримо достовірність експертних оцінок.

Загальну суму рангів усіх параметрів обчислимо за формулою (4.1)

$$\Sigma = \frac{m \cdot n(n + 1)}{2} = \frac{7 \cdot 4 \cdot 5}{2} = 70. \quad (4.1)$$

Середню суму рангів визначимо за формулою (4.2)

$$R = \frac{\Sigma}{n} = \frac{70}{4} = 17.5. \quad (4.2)$$

Відхилення суми рангів кожного параметра від середньої суми рангів визначимо за формулою (4.3)

$$d_i = \Sigma_i - R. \quad (4.3)$$

Числові значення відхилень та їхні квадрати наведено у таблиці 4.3. Суму квадратів відхилень обчислимо за формулою (4.4)

$$S = \sum_{i=1} d_i^2 = 195. \quad (4.4)$$

Коефіцієнт узгодженості думок експертів (коефіцієнт конкордації Кендалла) обчислимо за формулою (4.5)

$$W = \frac{12 \cdot S}{m^2 \cdot (n^3 - n)} = \frac{12 \cdot 195}{49 \cdot (64 - 4)} = \frac{2340}{2940} \approx 0,796. \quad (4.5)$$

Отримане значення $W = 0,796$ перевищує порогове 0,7, тому результати ранжування визнаються достовірними та придатними для подальших розрахунків.

Перевіримо статистичну значущість узгодженості за критерієм χ^2 Пірсона — формула (4.6)

$$\chi_{\text{розр}}^2 = m \cdot (n - 1) \cdot W = 7 \cdot 3 \cdot 0,796 = 16,71. \quad (4.6)$$

Табличне значення критерію χ^2 для рівня значущості $\alpha = 0,05$ та числа ступенів свободи $k = n - 1 = 3$ становить 7,815. Оскільки $\chi_{\text{розр}}^2 = 16,71 > \chi_{\text{табл}}^2 = 7,815$, узгодженість думок експертів є статистично значущою на рівні 5%.

Для подальшого визначення вагомості параметрів виконаємо їхнє попарне порівняння. Числове значення a_{ij} визначається за правилом: $a_{ij} = 1,5$ якщо X_i важливіше за X_j ; $a_{ij} = 1,0$ якщо параметри рівноцінні; $a_{ij} = 0,5$ якщо X_i менш важливе за X_j . Результати попарного порівняння наведено у таблиці 4.4.

Таблиця 4.4 — Попарне порівняння параметрів

	X1	X2	X3	X4	Сума
X1	1,0	1,5	1,5	1,5	5,5
X2	0,5	1,0	1,5	1,5	4,5
X3	0,5	0,5	1,0	0,5	2,5
X4	0,5	0,5	1,5	1,0	3,5
Σ	—	—	—	—	16,0

З отриманих числових оцінок формуємо матрицю A та розраховуємо вагомість K_w кожного параметра ітераційним методом до досягнення різниці між послідовними ітераціями менше 2%.

Сума елементів i -го рядка матриці A визначається за формулою (4.7)

$$b_i = \sum_{j=1} a_{ij}. \quad (4.7)$$

Вагомість параметра на першій ітерації обчислюється за формулою (4.8)

$$K_w^1(X_i) = \frac{b_i}{\sum_{i=1} b_i}. \quad (4.8)$$

Підставивши значення з таблиці 4.4, отримаємо результати першої ітерації:

$$K_w^1(X1) = 5,5 / 16,0 = 0,3438;$$

$$K_w^1(X2) = 4,5 / 16,0 = 0,2813;$$

$$K_w^1(X3) = 2,5 / 16,0 = 0,1563;$$

$$K_w^1(X4) = 3,5 / 16,0 = 0,2188.$$

На наступних ітераціях значення b_i перераховуються за формулою (4.9)

$$b_i' = \sum_{j=1} a_{ij} \cdot b_j. \quad (4.9)$$

Вагомість параметра на k -й ітерації визначається за формулою (4.10)

$$K_w^k(X_i) = \frac{b_i^{(k)}}{\sum_{i=1} b_i^{(k)}}. \quad (4.10)$$

Друга ітерація — множення матриці A на вектор сум попередньої ітерації:

$$b_1 = 1 \cdot 5,5 + 1,5 \cdot 4,5 + 1,5 \cdot 2,5 + 1,5 \cdot 3,5 = 21,25;$$

$$b_2 = 0,5 \cdot 5,5 + 1 \cdot 4,5 + 1,5 \cdot 2,5 + 1,5 \cdot 3,5 = 16,25;$$

$$b_3 = 0,5 \cdot 5,5 + 0,5 \cdot 4,5 + 1 \cdot 2,5 + 0,5 \cdot 3,5 = 9,25;$$

$$b_4 = 0,5 \cdot 5,5 + 0,5 \cdot 4,5 + 1,5 \cdot 2,5 + 1 \cdot 3,5 = 12,25;$$

$$\Sigma b = 59,0.$$

Звідки:

$$K_w^2(X1) = 21,25 / 59,0 = 0,3602;$$

$$K_w^2(X2) = 16,25 / 59,0 = 0,2754;$$

$$K_w^2(X3) = 9,25 / 59,0 = 0,1568;$$

$$K_w^2(X4) = 12,25 / 59,0 = 0,2076.$$

Третя ітерація:

$$c_1 = 77,875; c_2 = 59,125; c_3 = 34,125; c_4 = 44,875; \Sigma c = 216,0.$$

Відповідні вагомості:

$$K_w^3(X1) = 0,3605;$$

$$K_w^3(X2) = 0,2737;$$

$$K_w^3(X3) = 0,1580;$$

$$K_w^3(X4) = 0,2078.$$

Різниця між значеннями другої та третьої ітерацій не перевищує 2%, тому ітераційний процес завершується, а результати третьої ітерації приймаються як остаточні. Узагальнені дані ітераційного розрахунку вагомості параметрів наведено у таблиці 4.5.

Таблиця 4.5 — Розрахунок вагомості параметрів

Параметр	K_w^1	K_w^2	K_w^3	Різниця 2-ї та 3-ї ітерацій, %
X1	0,3438	0,3602	0,3605	0,08
X2	0,2813	0,2754	0,2737	0,62
X3	0,1563	0,1568	0,1580	0,77
X4	0,2188	0,2076	0,2078	0,10

Остаточні вагомості параметрів: $K_w(X1) = 0,3605$; $K_w(X2) = 0,2737$; $K_w(X3) = 0,1580$; $K_w(X4) = 0,2078$. Найвагомішим параметром є точність класифікації стану ока.

4.5 Аналіз рівня якості варіантів реалізації функцій

Коефіцієнт технічного рівня j -го варіанта реалізації обчислюється як зважена сума бальних оцінок параметрів — формула (4.11)

$$KK_j = \sum_{i=1} K_{w,i} \cdot b_{ij}, \quad (4.11)$$

де $K_{w,i}$ — коефіцієнт вагомості i -го параметра; b_{ij} — бальна оцінка i -го параметра j -го варіанта, визначена з графічних характеристик (рисунки 4.2–4.5) на основі фактичних значень параметрів.

Бальні оцінки параметрів варіантів отримані з графічних характеристик (рисунки 4.2–4.5) шляхом підстановки фактичних значень параметрів кожного варіанту.

Варіант 1 (запропонована реалізація — YOLOv8n з domain adaptation + двошаровий алгоритм):

X1 — висока надійність на цільових даних (precision 0.98) — 10 балів;

X2 — час обробки одного кадру 30 мс — 8 балів;

X3 — обсяг ОЗП моделі ~30 МБ — 9 балів;

X4 — двошарова стійкість через equalizeHist, deque-голосування та асиметричний лічильник — 10 балів.

Варіант 2 (базовий підхід — YOLOv8n без domain adaptation, покадрове рішення):

X1 — нижча надійність на цільових даних ($\text{precision} \approx 0.85$, помітна частка хибних спрацювань) — 4 бали;

X2 — час обробки одного кадру 25 мс — 9 балів;

X3 — обсяг ОЗП моделі 30 МБ — 9 балів;

X4 — відсутня стійкість до поодиноких хибних класифікацій — 3 бали.

Розрахунок коефіцієнтів технічного рівня варіантів за формулою (4.11) наведено у таблиці 4.6.

Таблиця 4.6 — Розрахунок показників рівня якості варіантів реалізації

Параметр	K_w	b_1 (B1)	$K_w \cdot b_1$	b_2 (B2)	$K_w \cdot b_2$
X1	0,3605	10	3,605	4	1,442
X2	0,2737	8	2,190	9	2,463
X3	0,1580	9	1,422	9	1,422
X4	0,2078	10	2,078	3	0,623
KK	—	—	9,30	—	5,95

За результатами розрахунку: $KK_1 = 9,30$; $KK_2 = 5,95$. Варіант 1 суттєво перевершує Варіант 2 за технічним рівнем. Перевага зумовлена насамперед вищою точністю класифікації після domain adaptation (10 балів проти 4 за параметром X1) та наявністю двошарового алгоритму накопичення, який забезпечує стійкість до поодиноких хибних класифікацій (10 балів проти 3 за параметром X4).

4.6 Економічний аналіз варіантів розробки ПП

Економічний аналіз варіантів передбачає розрахунок трудомісткості розробки, заробітної плати з відрахуваннями, витрат на машинний час та накладних витрат для визначення повної вартості розробки кожного варіанту.

4.6.1 Розрахунок трудомісткості

Розробка системи передбачає виконання чотирьох основних завдань:

- 1) розробка модуля захоплення відеопотоку та функції препроцесингу `preprocess_for_yolo`;
- 2) інтеграція каскадів Хаара з бібліотеки OpenCV для локалізації обличчя та зони очей;
- 3) навчання моделі YOLOv8n із застосуванням domain adaptation на гібридному датасеті (для Варіанту 1) або використання предтренованої моделі без донавчання (для Варіанту 2);
- 4) реалізація двошарового алгоритму накопичення індексу сонливості з візуальним інтерфейсом (для Варіанту 1) або покадрового рішення без накопичення (для Варіанту 2).

Класифікацію завдань за ступенем новизни (А/Б/В) і групою складності алгоритму (1/2/3) виконано за нормами методики функціонально-вартісного аналізу КПП; поправочні коефіцієнти K_{Π} визначено для кожного завдання. Результати наведено у таблиці 4.7.

Таблиця 4.7 — Класифікація завдань і поправочні коефіцієнти

№	Завдання	Ступінь новизни	Група складності	Вид інформації	T_R, ld	K_P
1	Захоплення відеопотоку та препроцесинг	В	3	НДІ	21	1,21
2	Інтеграція Haar Cascade	Б	2	НДІ	28	1,51
3а	YOLO + domain adaptation (тільки В1)	А	3	НДІ	35	1,21
3б	Готова YOLO без domain adaptation (тільки В2)	В	2	НДІ	15	1,21
4а	Двошаровий алгоритм накопичення (тільки В1)	Б	2	НДІ	18	1,51
4б	Покадрове рішення (тільки В2)	В	1	НДІ	10	1,21

Загальна трудомісткість виконання завдання визначається за формулою (4.12)

$$T_O = T_R \cdot K_P \cdot K_{СК} \cdot K_M \cdot K_{СТ} \cdot K_{СТ.М}, \quad (4.12)$$

де T_R — базова трудомісткість завдання, ld; K_P — поправочний коефіцієнт за складністю алгоритму; $K_{СК} = 1$ — коефіцієнт стандартної складності вхідної інформації; $K_M = 0,85$ — коефіцієнт мови програмування (Python); $K_{СТ} = 0,8$ — коефіцієнт використання стандартних модулів (OpenCV, Ultralytics); $K_{СТ.М} = 1$.
Спільний множник: $K_{СК} \cdot K_M \cdot K_{СТ} \cdot K_{СТ.М} = 1 \cdot 0,85 \cdot 0,8 \cdot 1 = 0,68$.

Розрахунок для Варіанту 1.

$$T_1 = 21 \cdot 1,21 \cdot 0,68 \approx 17,28 \text{ ld};$$

$$T_2 = 28 \cdot 1,51 \cdot 0,68 \approx 28,75 \text{ ld};$$

$$T_3 = 35 \cdot 1,21 \cdot 0,68 \approx 28,80 \text{ ld};$$

$$T_4 = 18 \cdot 1,51 \cdot 0,68 \approx 18,48 \text{ ld.}$$

Сумарна трудомісткість Варіанту 1: $T_I \approx 93,31 \text{ ld} \approx 746,49$ людино-годин (з урахуванням 8-годинного робочого дня).

Розрахунок для Варіанту 2.

$$T_1 = 17,28 \text{ ld;}$$

$$T_2 = 28,75 \text{ ld;}$$

$$T_3 = 15 \cdot 1,21 \cdot 0,68 \approx 12,34 \text{ ld;}$$

$$T_4 = 10 \cdot 1,21 \cdot 0,68 \approx 8,23 \text{ ld.}$$

Сумарна трудомісткість Варіанту 2: $T_{II} \approx 66,60 \text{ ld} \approx 532,79$ людино-годин.

4.6.2 Розрахунок заробітної плати та відрахувань

У розробці бере участь один виконавець — Junior Computer Vision Engineer — з місячним окладом $M = 35\,000$ грн. Середня погодинна тарифна ставка обчислюється за формулою (4.13)

$$C_{\text{ч}} = \frac{M}{T_m \cdot t} = \frac{35\,000}{21,1 \cdot 8} \approx 207,35 \text{ грн/год.} \quad (4.13)$$

де $M = 35\,000$ грн — місячний оклад; $T_m = 21,1$ — середня кількість робочих днів у місяці; $t = 8$ год — тривалість робочого дня.

Заробітна плата виконавця за весь період розробки з урахуванням коефіцієнта додаткової заробітної плати $K_d = 1,2$ обчислюється за формулою (4.14)

$$C_{\text{зп}} = C_{\text{ч}} \cdot T_i \cdot K_d. \quad (4.14)$$

Підставивши значення трудомісткості, отримаємо:

$$C_{\text{зп(I)}} = 207,35 \cdot 746,49 \cdot 1,2 \approx 185\,742 \text{ грн;}$$

$$C_{\text{зп(II)}} = 207,35 \cdot 532,79 \cdot 1,2 \approx 132\,569 \text{ грн.}$$

Відрахування на єдиний соціальний внесок (22% від заробітної плати):

$$C_{\text{ВД}}(\text{I}) = 185\,742 \cdot 0,22 \approx 40\,863 \text{ грн};$$

$$C_{\text{ВД}}(\text{II}) = 132\,569 \cdot 0,22 \approx 29\,165 \text{ грн}.$$

4.6.3 Витрати на машинний час

Для визначення витрат на машинний час обчислимо собівартість однієї машино-години роботи ЕОМ. ПК обслуговує одного розробника з окладом 35 000 грн; коефіцієнт зайнятості ПК завданнями розробки $K_3 = 0,2$:

$$C_{\text{Г}} = 12 \cdot M \cdot K_3 = 12 \cdot 35\,000 \cdot 0,2 = 84\,000 \text{ грн}.$$

З урахуванням коефіцієнта додаткової заробітної плати:

$$C_{\text{ЗП}}(\text{ЕОМ}) = 84\,000 \cdot (1 + 0,2) = 100\,800 \text{ грн}.$$

Відрахування на єдиний соціальний внесок:

$$C_{\text{ВД}}(\text{ЕОМ}) = 100\,800 \cdot 0,22 \approx 22\,176 \text{ грн}.$$

Амортизаційні відрахування при нормі амортизації $K_A = 0,25$ (термін експлуатації 4 роки), вартості ноутбука $\text{Ц}_{\text{ПР}} = 35\,000$ грн та коефіцієнті транспортно-монтажних витрат $K_{\text{ТМ}} = 1,15$:

$$C_A = K_{\text{ТМ}} \cdot K_A \cdot \text{Ц}_{\text{ПР}} = 1,15 \cdot 0,25 \cdot 35\,000 \approx 10\,063 \text{ грн}.$$

Витрати на ремонт та профілактику при коефіцієнті $K_P = 0,05$:

$$C_P = K_{\text{ТМ}} \cdot \text{Ц}_{\text{ПР}} \cdot K_P = 1,15 \cdot 35\,000 \cdot 0,05 \approx 2\,013 \text{ грн}.$$

Ефективний річний фонд робочого часу ПК (з урахуванням 104 днів вихідних, 11 днів святкових, 8 днів відпустки та коефіцієнта використання обладнання 0,85):

$$T_{\text{ЕФ}} = (365 - 104 - 11 - 8) \cdot 8 \cdot 0,85 = 1\,645,6 \text{ год}.$$

Витрати на електроенергію розраховано за тарифом $\text{Ц}_{\text{ЕН}} = 13,64$ грн/кВт·год (стандартний тариф 2-го класу напруги для бізнесу з ПДВ), потужності споживання ПК $N_C = 0,2$ кВт та коефіцієнті зайнятості $K_3 = 0,2$:

$$C_{\text{ЕЛ}} = T_{\text{ЕФ}} \cdot N_{\text{С}} \cdot K_3 \cdot C_{\text{ЕН}} = 1\,645,6 \cdot 0,2 \cdot 0,2 \cdot 13,64 \approx 898 \text{ грн.}$$

Накладні витрати на обладнання (67% від вартості ПК):

$$C_{\text{Н(ЕОМ)}} = C_{\text{ПР}} \cdot 0,67 = 35\,000 \cdot 0,67 = 23\,450 \text{ грн.}$$

Річні експлуатаційні витрати на ПК визначаються за формулою (4.15)

$$C_{\text{ЕКС}} = C_{\text{ЗП(ЕОМ)}} + C_{\text{ВІД(ЕОМ)}} + C_{\text{А}} + C_{\text{Р}} + C_{\text{ЕЛ}} + C_{\text{Н(ЕОМ)}}. \quad (4.15)$$

Підставивши значення:

$$C_{\text{ЕКС}} = 100\,800 + 22\,176 + 10\,063 + 2\,013 + 898 + 23\,450 \approx 159\,400 \text{ грн.}$$

Собівартість однієї машино-години:

$$C_{\text{М-Г}} = C_{\text{ЕКС}} / T_{\text{ЕФ}} = 159\,400 / 1\,645,6 \approx 96,86 \text{ грн/год.}$$

Витрати на машинний час за період розробки обчислюються за формулою (4.16)

$$C_{\text{М}} = C_{\text{М-Г}} \cdot T_i. \quad (4.16)$$

Звідки:

$$C_{\text{М(I)}} = 96,86 \cdot 746,49 \approx 72\,305 \text{ грн;}$$

$$C_{\text{М(II)}} = 96,86 \cdot 532,79 \approx 51\,605 \text{ грн.}$$

Накладні витрати на розробку (67% від заробітної плати) визначаються за формулою (4.17)

$$C_{\text{Н}} = C_{\text{ЗП}} \cdot 0,6. \quad (4.17)$$

$$C_{\text{Н(I)}} = 185\,742 \cdot 0,67 \approx 124\,447 \text{ грн;}$$

$$C_{\text{Н(II)}} = 132\,569 \cdot 0,67 \approx 88\,821 \text{ грн.}$$

4.6.4 Повна вартість розробки

Повна вартість розробки програмного продукту визначається сумою основної заробітної плати, відрахувань, витрат на машинний час та накладних витрат — формула (4.18)

$$C_{\text{ПП}} = C_{\text{ЗП}} + C_{\text{ВІД}} + C_{\text{М}} + C_{\text{Н}}. \quad (4.18)$$

Підставивши значення для кожного варіанта:

$$C_{\text{ПП}}(\text{I}) = 185\,742 + 40\,863 + 72\,305 + 124\,447 \approx 423\,357 \text{ грн};$$

$$C_{\text{ПП}}(\text{II}) = 132\,569 + 29\,165 + 51\,605 + 88\,821 \approx 302\,160 \text{ грн}.$$

Порівняльну характеристику економічних показників варіантів розробки наведено у таблиці 4.8.

Таблиця 4.8 — Порівняльна таблиця економічних показників варіантів розробки

Стаття витрат	Варіант 1, грн	Варіант 2, грн
Заробітна плата ($C_{\text{ЗП}}$)	185 742	132 569
Відрахування на ЄСВ ($C_{\text{ВІД}}$)	40 863	29 165
Витрати на машинний час ($C_{\text{М}}$)	72 305	51 605
Накладні витрати ($C_{\text{Н}}$)	124 447	88 821
Повна вартість розробки ($C_{\text{ПП}}$)	423 357	302 160

4.7 Вибір кращого варіанту ПП за техніко-економічним рівнем

Для остаточного вибору варіанту реалізації використовується коефіцієнт техніко-економічного рівня, який поєднує технічний рівень варіанта та його повну вартість — формула (4.19)

$$K_{\text{ТЕР}}(j) = \frac{KK_j}{C_{\text{ПП}}(j)}. \quad (4.19)$$

Для зручності зіставлення значення наводяться помножені на 10^5 :

$$K_{\text{ТЕР}}(\text{I}) = (9,30 / 423\,357) \cdot 10^5 \approx 2,20 \cdot 10^{-5};$$

$$K_{\text{ТЕР}}(\text{II}) = (5,95 / 302\,160) \cdot 10^5 \approx 1,97 \cdot 10^{-5}.$$

Оскільки $K_{\text{ТЕР}}(\text{I}) = 2,20 \cdot 10^{-5} > K_{\text{ТЕР}}(\text{II}) = 1,97 \cdot 10^{-5}$, Варіант 1 (Python + Haar Cascade + YOLOv8n з domain adaptation + двошаровий алгоритм накопичення) є економічно та технічно більш доцільним.

Підсумкові показники порівняння варіантів зведено у таблиці 4.9.

Таблиця 4.9 — Підсумкові показники порівняння варіантів

Показник	Варіант 1	Варіант 2
Коефіцієнт технічного рівня КК	9,30	5,95
Повна вартість розробки $C_{\text{ПП}}$, грн	423 357	302 160
Коефіцієнт техніко-економічного рівня $K_{\text{ТЕР}} \cdot 10^5$	2,20	1,97

Висновки до розділу 4

1. Визначено функціональну структуру програмного продукту: виділено чотири ключові підфункції — вибір мови програмування (F1), метод локалізації обличчя і зони очей (F2), модель класифікації стану ока (F3) та алгоритм формування рішення про сонливість (F4). Сформовано два варіанти реалізації: Варіант 1 — запропонована реалізація (Python + Haar Cascade + YOLOv8n з domain adaptation + двошаровий алгоритм накопичення); Варіант 2 — базовий підхід (Python + Haar Cascade + YOLOv8n без domain adaptation + покадрове рішення).

2. Проведено експертне оцінювання значущості параметрів: залучено 7 фахівців, коефіцієнт узгодженості Кендалла $W = 0,796 (> 0,7)$, що підтверджує достовірність експертних оцінок. Узгодженість є статистично значущою на рівні 5% ($\chi^2_{\text{розр}} = 16,71 > \chi^2_{\text{табл}} = 7,815$). Найвагомим параметром визнано точність класифікації стану ока: $K_w(X1) = 0,3605$.
3. Розраховано коефіцієнт технічного рівня варіантів: $KK_1 = 9,30$ проти $KK_2 = 5,95$, що підтверджує суттєву перевагу Варіанту 1. Ця перевага зумовлена насамперед застосуванням domain adaptation, яке підвищило точність на цільових даних (precision з 0.85 до 0.98) і усунило хибні спрацювання, та двошаровим алгоритмом накопичення, який забезпечує стійкість системи до поодиноких хибних класифікацій.
4. Виконано економічний розрахунок: трудомісткість Варіанту 1 становить 746,49 людино-годин, Варіанту 2 — 532,79 людино-годин. Повна вартість розробки Варіанту 1 — 423 357 грн, Варіанту 2 — 302 160 грн.
5. Обрано оптимальний варіант: за коефіцієнтом техніко-економічного рівня $K_{\text{ТЕР}}(I) = 2,20 \cdot 10^{-5} > K_{\text{ТЕР}}(II) = 1,97 \cdot 10^{-5}$. Варіант 1 (Python + Haar Cascade + YOLOv8n з domain adaptation + двошаровий алгоритм накопичення) є економічно обґрунтованим і технічно найдоцільнішим рішенням, що й реалізовано у даній дипломній роботі.

ВИСНОВКИ

У дипломній роботі розроблено інтелектуальну систему відеомоніторингу стану водія автотранспорту, що функціонує в режимі реального часу на стандартній веб-камері. За результатами виконаної роботи можна сформулювати такі висновки.

1. Аналіз дорожньо-транспортної статистики підтвердив, що втома і мікросон спричиняють до 30% дорожньо-транспортних пригод. Параметр PERCLOS та характеристики моргань визначено як ключові об'єктивні маркери сонливості, придатні для автоматизованого відеоаналізу.
2. Огляд комерційних DMS-систем (Seeing Machines FOVIO, Mobileye Guardian, Bosch DMS) виявив їх спільне принципове обмеження — залежність від дорогого спеціалізованого апаратного забезпечення. Розроблена система адресує цю проблему, функціонуючи на базі стандартної USB-камери та центрального процесора.
3. Обрано гібридну архітектуру: каскади Хаара — для швидкої локалізації обличчя і зони ока, YOLOv8n у режимі детекції — для класифікації стану ока на вирізаному зображенні. Порівняльний аналіз методів показав, що такий підхід поєднує швидкодію класичних методів з точністю сучасного глибокого навчання.
4. Сформовано датасет з 6 384 зображень (5 569 базових та 815 кастомних). Навчання моделі YOLOv8n протягом 50 епох із застосуванням domain adaptation підвищило метрики: mAP50 з 0.96 до 0.99, Recall з 0.95 до 0.99, Precision з 0.85 до 0.98. Клас drowsy розпізнається з точністю 0.99.
5. Розроблено програмний продукт мовою Python, що реалізує двошаровий алгоритм накопичення індексу сонливості — мажоритарне вікно з 5 кадрів та накопичувальний лічильник з асиметричним

оновленням (+1 / -2). Система стабільно працює у режимі реального часу (близько 8–9 FPS на CPU) і є готовою основою для подальшого розгортання.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. World Health Organization. Global status report on road safety 2023. Geneva : WHO, 2023. URL: <https://www.who.int/teams/social-determinants-of-health/safety-and-mobility/global-status-report-on-road-safety-2023> (дата звернення: 19.05.2026).
2. National Highway Traffic Safety Administration. Drowsy Driving Research and Program. Washington : NHTSA, 2021. URL: <https://www.nhtsa.gov/risky-driving/drowsy-driving> (дата звернення: 19.05.2026).
3. Wierwille W. W., Ellsworth L. A. Evaluation of driver drowsiness by trained raters. Accident Analysis & Prevention. 1994. Vol. 26, No. 5. P. 571–581.
4. Seeing Machines. Automotive Driver Monitoring — FOVIO Platform. 2024. URL: <https://www.seeingmachines.com/automotive> (дата звернення: 20.05.2026).
5. Mobileye. Guardian by Mobileye — Fleet Safety System. 2024. URL: <https://www.mobileye.com> (дата звернення: 20.05.2026).
6. Bosch Mobility. Driver Monitoring System. 2024. URL: <https://www.bosch-mobility.com> (дата звернення: 20.05.2026).
7. Viola P., Jones M. Rapid Object Detection using a Boosted Cascade of Simple Features. Proceedings of IEEE CVPR 2001, Kauai, HI, USA, 8–14 Dec. 2001. Vol. 1. P. I-511–I-518.
8. Soukupová T., Čech J. Real-Time Eye Blink Detection using Facial Landmarks. Proceedings of the 21st Computer Vision Winter Workshop, Rimske Toplice, Slovenia, 3–5 Feb. 2016. Ljubljana : Slovenian Pattern Recognition Society, 2016. P. 1–8.
9. You Only Look Once: Unified, Real-Time Object Detection / J. Redmon et al. Proceedings of IEEE CVPR 2016, Las Vegas, NV, USA, 27–30 June 2016. P. 779–788.

10. Jocher G., Chaurasia A., Qiu J. Ultralytics YOLOv8. Version 8.0.0. 2023.
URL: <https://docs.ultralytics.com> (дата звернення: 21.05.2026).
11. OpenCV Team. OpenCV — Open Source Computer Vision Library.
Documentation v. 4.x. 2024. URL: <https://docs.opencv.org> (дата звернення: 21.05.2026).
12. Pan S. J., Yang Q. A Survey on Transfer Learning. IEEE Transactions on Knowledge and Data Engineering. 2010. Vol. 22, No. 10. P. 1345–1359.
13. Kunding T., Sofra N., Riener A. Assessment of the Potential of Wrist-Worn Wearable Sensors for Driver Drowsiness Detection. Sensors. 2020. Vol. 20, No. 4. Art. 1029.
14. ImageNet: A Large-Scale Hierarchical Image Database / J. Deng et al. Proceedings of IEEE CVPR 2009, Miami, FL, USA, 20–25 June 2009. P. 248–255.
15. LeCun Y., Bengio Y., Hinton G. Deep Learning. Nature. 2015. Vol. 521. P. 436–444.
16. Hasan M., Roy-Chowdhury A. K. A Continuous Learning Framework for Activity Recognition Using Deep Hybrid Feature Models. IEEE Transactions on Multimedia. 2015. Vol. 17, No. 11. DOI: 10.1109/TMM.2015.2477242.
17. Roboflow Inc. Roboflow Platform for Computer Vision Datasets. 2023. URL: <https://roboflow.com> (дата звернення: 15.05.2026).
18. Bochkovskiy A., Wang C.-Y., Liao H.-Y. M. YOLOv4: Optimal Speed and Accuracy of Object Detection. 2020. 17 p. URL: <https://arxiv.org/abs/2004.10934> (дата звернення: 21.05.2026).
19. Graves A., Mohamed A., Hinton G. Speech Recognition with Deep Recurrent Neural Networks. Proceedings of IEEE ICASSP 2013, Vancouver, BC, Canada, 26–31 May 2013. P. 6645–6649.
20. Goodfellow I., Bengio Y., Courville A. Deep Learning. Cambridge : MIT Press, 2016. 800 p.
21. Dinges D. F., Grace R. PERCLOS: A Valid Psychophysiological Measure of Alertness as Assessed by Psychomotor Vigilance. Washington : Federal

- Highway Administration, 1998. 76 p. (Publication No. FHWA-MCRT-98-006).
22. Bradski G., Kaehler A. Learning OpenCV: Computer Vision with the OpenCV Library. Sebastopol : O'Reilly Media, 2008. 580 p.
 23. Li S. Z., Jain A. K. Handbook of Face Recognition. London : Springer, 2011. 699 p.
 24. A Survey on State-of-the-Art Drowsiness Detection Techniques / M. Ramzan et. al. IEEE Access. 2019. Vol. 7. P. 61904–61919. DOI: 10.1109/ACCESS.2019.2914373.

ДОДАТОК А ЛІСТИНГ ПРОГРАМНОГО ПРОДУКТУ

Лістинг програмного продукту — inference_final.py.

```
import cv2
from ultralytics import YOLO
import numpy as np
import os
import time
from collections import deque

# Налаштування
eye_history = deque(maxlen=5)
drowsiness_score = 0
ALARM_THRESHOLD = 15
MAX_SCORE = 30
RECOVERY_RATE = 2

# Створення папок для кастомного датасету
dataset_dir = "custom_dataset"
os.makedirs(os.path.join(dataset_dir, "drowsy"), exist_ok=True)
os.makedirs(os.path.join(dataset_dir, "not_drowsy"), exist_ok=True)

def preprocess_for_yolo(img_roi):
    # Перетворення кольорового зображення на чорно-біле
    gray = cv2.cvtColor(img_roi, cv2.COLOR_BGR2GRAY)
    gray_eq = cv2.equalizeHist(gray)
    bgr_eq = cv2.cvtColor(gray_eq, cv2.COLOR_GRAY2BGR)
    return bgr_eq

current_dir = os.path.dirname(os.path.abspath(__file__))
original_dir = os.getcwd()

try:
    os.chdir(current_dir)
    face_cascade =
cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
    eye_cascade = cv2.CascadeClassifier("haarcascade_eye.xml")
    if face_cascade.empty() or eye_cascade.empty():
        print("Помилка: XML файли не знайдено.")
        exit()
finally:
    os.chdir(original_dir)

base_dir = os.path.dirname(current_dir)

model_path = os.path.join(base_dir, 'model_v2_custom.pt')

model = YOLO(model_path)

cap = cv2.VideoCapture(0, cv2.CAP_DSHOW)
font = cv2.FONT_HERSHEY_SIMPLEX
```

```

RED = (0, 0, 255)
GREEN = (0, 255, 0)
BLUE = (255, 0, 0)

print("Система запущена. Клавіші: 1 - Зберегти 'not_drowsy', 2 -
Зберегти 'drowsy', Q - Вихід.")

while True:
    ret, frame = cap.read()
    if not ret:
        break

    height, width = frame.shape[:2]
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.3,
minNeighbors=5, minSize=(100, 100))

    frame_status_is_sleep = False

    # Зчитування натискання клавіш для розмітки даних
    key = cv2.waitKey(1) & 0xFF

    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x+w, y+h), BLUE, 2)

        roi_gray_eyes = gray[y:y+int(h*0.6), x:x+w]
        roi_color_eyes = frame[y:y+int(h*0.6), x:x+w]
        eyes = eye_cascade.detectMultiScale(roi_gray_eyes,
scaleFactor=1.05, minNeighbors=3, minSize=(30, 30))

        eyes_open_count = 0
        eyes_total = 0

        for (ex, ey, ew, eh) in eyes:
            eyes_total += 1
            eye_img_raw = roi_color_eyes[ey:ey+eh, ex:ex+ew]
            processed_eye = preprocess_for_yolo(eye_img_raw)

            # Логіка збереження кастомного датасету
            if key == ord('1'):
                filename = os.path.join(dataset_dir, "not_drowsy",
f"eye_{int(time.time()*1000)}.jpg")
                cv2.imwrite(filename, processed_eye)
                print(f"Збережено: {filename}")
            elif key == ord('2'):
                filename = os.path.join(dataset_dir, "drowsy",
f"eye_{int(time.time()*1000)}.jpg")
                cv2.imwrite(filename, processed_eye)
                print(f"Збережено: {filename}")

            results = model(processed_eye, conf=0.4, verbose=False)

            color = GREEN
            text = "OK"
            is_eye_open = True

            if len(results) > 0 and len(results[0].boxes) > 0:

```

```

        box = results[0].boxes[0]
        cls_id = int(box.cls[0])
        label_name = model.names[cls_id]
        conf_val = float(box.conf[0])

        if label_name == 'drowsy':
            color = RED
            text = "SLEEP"
            is_eye_open = False

            abs_x = x + ex
            abs_y = y + ey
            cv2.rectangle(frame, (abs_x, abs_y), (abs_x+ew,
abs_y+eh), color, 2)
            cv2.putText(frame, f"{text} {conf_val:.2f}", (abs_x,
abs_y - 5), font, 0.4, color, 1)

            if is_eye_open:
                eyes_open_count += 1

        if eyes_total > 0:
            if eyes_open_count >= 1:
                frame_status_is_sleep = False
            else:
                frame_status_is_sleep = True
        else:
            frame_status_is_sleep = False

        if frame_status_is_sleep:
            eye_history.append(1)
        else:
            eye_history.append(0)

        if sum(eye_history) > len(eye_history) / 2:
            drowsiness_score = min(drowsiness_score + 1, MAX_SCORE)
            cv2.putText(frame, "STATUS: DROWSY", (10, 30), font, 1, RED,
2)
        else:
            drowsiness_score = max(drowsiness_score - RECOVERY_RATE, 0)
            cv2.putText(frame, "STATUS: ACTIVE", (10, 30), font, 1, GREEN,
2)

        score_ratio = drowsiness_score / ALARM_THRESHOLD
        score_width = int(min(score_ratio, 1.0) * width)

        bar_color = GREEN
        if drowsiness_score >= ALARM_THRESHOLD: bar_color = RED

        cv2.rectangle(frame, (0, height-20), (score_width, height),
bar_color, -1)
        cv2.putText(frame, f"Fatigue:
{drowsiness_score}/{ALARM_THRESHOLD}", (10, height-25), font, 0.6,
(255,255,255), 1)

        if drowsiness_score >= ALARM_THRESHOLD:
            cv2.putText(frame, "WAKE UP!!!", (width//2 - 100, height//2),
font, 2, RED, 5)

```

```

cv2.imshow('Final Driver Monitor v6', frame)

# Вихід - Q
if key == ord('q'):
    break

cap.release()
cv2.destroyAllWindows()

```

Лістинг скрипту об'єднання датасетів — merge_custom_dataset.py.

```

import os
import shutil
import random

# Налаштування шляхів
custom_dir = "custom_dataset"
main_dataset_dir = os.path.join("dataset", "raw")

# Класи: 0 - drowsy, 1 - not_drowsy
classes = {"drowsy": 0, "not_drowsy": 1}
split_ratio = 0.8 # 80% тренування, 20% валідація

print("Починаємо інтеграцію та розмітку нових даних...")

for class_name, class_id in classes.items():
    source_folder = os.path.join(custom_dir, class_name)
    if not os.path.exists(source_folder):
        print(f"Папку {source_folder} не знайдено. Пропускаємо.")
        continue

    images = [f for f in os.listdir(source_folder) if
f.endswith('.jpg')]
    random.shuffle(images)

    split_index = int(len(images) * split_ratio)
    train_imgs = images[:split_index]
    valid_imgs = images[split_index:]

    def process_images(img_list, split_name):
        img_dest_dir = os.path.join(main_dataset_dir, split_name,
"images")
        lbl_dest_dir = os.path.join(main_dataset_dir, split_name,
"labels")

        os.makedirs(img_dest_dir, exist_ok=True)
        os.makedirs(lbl_dest_dir, exist_ok=True)

        for img_name in img_list:
            # Копіювання зображення
            src_img_path = os.path.join(source_folder, img_name)
            dst_img_path = os.path.join(img_dest_dir, img_name)
            shutil.copy(src_img_path, dst_img_path)

            # Створення текстового файлу розмітки YOLO
            lbl_name = img_name.replace('.jpg', '.txt')

```

```

dst_lbl_path = os.path.join(lbl_dest_dir, lbl_name)

# Формат: class_id x_center y_center width height
with open(dst_lbl_path, 'w') as f:
    f.write(f"{class_id} 0.5 0.5 1.0 1.0\n")

process_images(train_imgs, "train")
process_images(valid_imgs, "valid")
print(f"Клас '{class_name}': {len(train_imgs)} у train,
{len(valid_imgs)} у valid.")

print("Нові дані успішно розмічені та додані до основного датасету.")

```

ДОДАТОК Б ПРЕЗЕНТАЦІЯ



Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Навчально-науковий інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

ІНТЕЛЕКТУАЛЬНА СИСТЕМА ВІДЕОМОНІТОРИНГУ СТАНУ ВОДІЯ АВТОТРАНСПОРТУ ДЛЯ ПІДВИЩЕННЯ БЕЗПЕКИ ДОРОЖНЬОГО РУХУ

Виконав: Кудрявцев Іван Михайлович, студент IV курсу, група КА-23

Керівник: Титаренко Андрій Миколайович, асистент кафедри ММСА, доктор
філософії

2026

Актуальність дослідження

1. Масштаб проблеми

- За даними ВООЗ – у світі щороку гине ~1.35 млн людей у ДТП.
- 10–30 % серйозних аварій пов'язані з втомою водія.
- NHTSA: понад 100 000 аварій / 71 000 травм / 1 500 смертей лише у США.

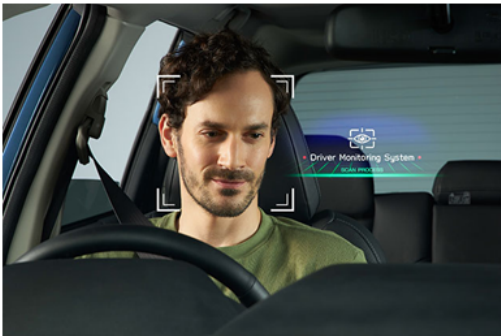
2. Мікросон – головна загроза

- Короткі (2–30 сек) неусвідомлені засинання.
- При швидкості 90 км/год за 1 секунду авто долає ~25 м некерованого шляху.

3. Існуючі рішення недоступні

- Комерційні DMS (Seeing Machines, Mobileye, Bosch) потребують спеціалізованого обладнання (ІЧ-камери, DSP/FPGA).
- Непридатні для масового впровадження у малому транспортному бізнесі.

Огляд комерційних DMS-систем



Джерело: RAC Horizons, In-car driver monitoring (rac.com.au), 2024. Ілюстративне зображення.

Система	Сенсор	Апаратні вимоги	Вартість
Seeing Machines FOVIO	IЧ + DSP/FPGA	Дуже високі	Дуже висока
Mobileye Guardian	RGB + IЧ	Високі	Висока
Bosch DMS	IЧ-камера, OEM	Високі	Висока
Запропонована система	RGB веб-камера	Низькі (CPU)	Низька

Спільний недолік комерційних систем – залежність від дорогого спеціалізованого обладнання. Запропонована система адресує цю проблему.

Об'єкт, предмет та мета дослідження

Об'єкт дослідження

процес виявлення ознак сонливості водія за відеопотоком у реальному часі.

Предмет дослідження

методи комп'ютерного зору, архітектури нейронних мереж та алгоритми обробки відеопотоку.

Мета роботи

розробити інтелектуальну систему відеомоніторингу стану водія на основі гібридної архітектури OpenCV + YOLOv8, що функціонує в режимі реального часу на стандартній веб-камері без спеціалізованого обладнання.

Постановка задачі

Вхідні дані

Відеопотік з RGB веб-камери в реальному часі, без спеціальних сенсорів.

Вихідні дані

Оцінка стану водія в реальному часі (активний / ознаки сонливості), накопичений індекс сонливості та сигнал тривоги при стійкому виявленні засинання.

Критерії та вимоги

Робота на стандартному CPU без GPU; точність Precision, Recall, mAP50 ≥ 0.90 .

5

Задачі дослідження

- 1 Аналіз статистики ДТП через втому та визначення візуальних маркерів сонливості.
- 2 Огляд комерційних DMS-систем та виявлення їх обмежень.
- 3 Огляд методів комп'ютерного зору та обґрунтування технологічного стеку.
- 4 Формування датасету та навчання моделі YOLOv8n із застосуванням domain adaptation.
- 5 Розроблення алгоритму накопичення індексу сонливості та реалізація програмного прототипу.

6

Огляд методів детекції сонливості

Метод	Точність	Швидкість	Стійкість до освітлення	Потреба у даних
EAR (landmarks)	Середня	Висока	Низька	Мала
Haar Cascade	Середня	Дуже висока	Середня	Не потрібна
CNN-класифікатор	Висока	Середня	Висока	Велика
YOLOv8	Висока	Висока	Висока	Середня

Обрано гібридну архітектуру: каскади Хаара – для швидкої локалізації обличчя та зони очей; YOLOv8n – для точної класифікації стану ока. Поеднання швидкості класичних методів і точності нейромереж.

7

Двоетапна гібридна архітектура

Логіка роботи

Кожен кадр відеопотоку проходить двоетапну обробку. Спочатку каскади Хаара локалізують обличчя, а в його межах – зону очей.

Кожне виявлене око вирізається, проходить препроцесинг (вирівнювання гістограми) і подається у нейромережу **YOLOv8n**, яка класифікує його стан як **drowsy** або **not_drowsy**.

Результати накопичуються у часі двошаровим алгоритмом, і при перевищенні порогу система видає сигнал тривоги **“WAKE UP!!!!”**.

Швидкодія **~8-9 FPS на CPU** · точність **mAP50 = 0.99**



Рис. 3.3 – алгоритм роботи

8

Двошаровий алгоритм прийняття рішення

Шар 1

Мажоритарне голосування

ковзне вікно 5 кадрів

- deque(maxlen=5) зберігає результати останніх 5 кадрів
- Захист від випадкових помилок класифікатора (морганія)
- Кадр «сонний», якщо >2.5 з 5 кадрів – сонні

Шар 2

Накопичувальний лічильник

drowsiness_score ∈ [0; 30]

- Сонний кадр + score +1
- Активний кадр + score -2 (асиметрія!)
- Тривога: score ≥ ALARM_THRESHOLD = 15

Асиметрія оновлення (-2 проти +1): водій «прокидається» вдвічі швидше, ніж «засинає». Дизайн на користь безпеки.

9

Гібридний навчальний датасет – 6 384 зображення

Базовий датасет

Roboflow «driver drowsiness1»

- 5 569 зображень
- Класи: drowsy / not_drowsy
- Формат: YOLO v5 PyTorch
- Препроцесинг: CLAHE, augmentation

Кастомний датасет

815 зображень (426 drowsy + 389 not_drowsy)

- Зібрано через робочий пайплайн системи (клавіші 1/2 під час інференсу)
- Автоматична розмітка merge_custom_dataset.py
- Поділ 80/20 на train / valid



Рис. 3.1 – приклади зображень датасету

Ключова перевага: зображення проходять ту саму обробку, що й кадри під час роботи + усуває domain shift.

10

Навчання YOLOv8n + Domain Adaptation

Ключові графіки для задачі

- cls_loss – помилка класифікації стану ока – стабільно спадає
- Precision і Recall – зростають і виходять на плато ~0.99
- mAP50 – узагальнена точність;
- box_loss / dfl_loss / mAP50-95 – другорядні, адже геометрія рамки в системі не використовується

Метрика	До DA	Після DA	Δ
Precision	0.85	0.98	+0.13
Recall	0.95	0.99	+0.04
F1-score	0.90	0.98	+0.08
mAP50	0.96	0.99	+0.03

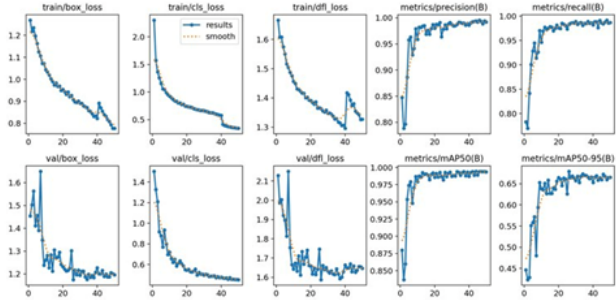


Рис. 3.4 – функції втрат і метрики якості за епохами

11

Якість моделі на валідаційній вибірці

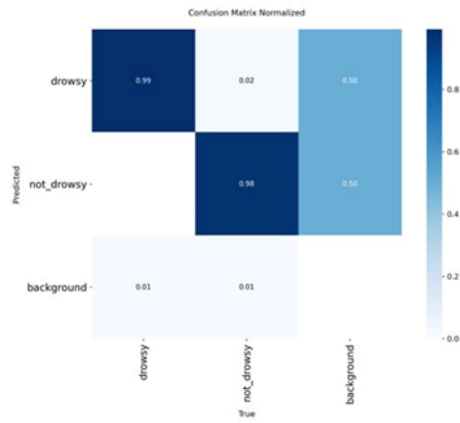


Рис. 3.5 – нормалізована матриця плутанини

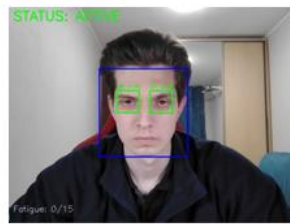
0.99
Клас drowsy – точність

0.98
Клас not_drowsy – точність

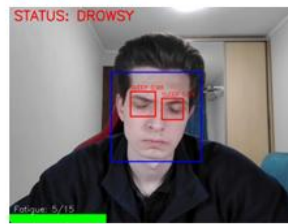
≈ 0
False negative практично відсутні – критично для системи безпеки

12

Інтерфейс системи у різних режимах



Активний стан



Виявлено сонливість



Спрацювала тривога



Стійкість до часткової видимості

Аналіз продуктивності:
~8-9 FPS на
стандартному CPU без
GPU.

13

Функціонально-вартісний аналіз



Рис. 4.1 – морфологічна карта системи

Обґрунтування оптимальної конфігурації системи за співвідношенням «якість / вартість».

Методи системного аналізу: морфологічний аналіз (варіанти реалізації) та експертне ранжування критеріїв якості.

Критерії якості: точність, швидкодія, обсяг ОЗП, стійкість.

Узгодженість 7 експертів підтверджено коефіцієнтом Кендалла $W = 0,796$; найвагоміший критерій – точність.

Обрана конфігурація (Haar + YOLOv8n з DA + двошаровий алгоритм) – технічно й економічно найдоцільніша.

14

Висновки

- 1 Розроблено інтелектуальну систему відеомоніторингу стану водія, що працює в реальному часі на стандартному CPU без спеціалізованого обладнання.
- 2 Основа системи – гібридна двоетапна архітектура: каскади Хаара локалізують зону очей, а неймережа YOLOv8n класифікує стан кожного ока.
- 3 Ключова особливість – двошаровий алгоритм накопичення індексу сонливості (вікно з 5 кадрів + асиметричний лічильник), стійкий до поодиноких помилок і моргань.
- 4 Навчання на власному кастомному датасеті підвищило точність на цільових даних: precision з 0.85 до 0.98 (усунено хибні спрацювання), Recall 0.99.
- 5 Головні переваги: робота на звичайному CPU без GPU, стійкість до хибних класифікацій і висока надійність виявлення сонливості.

Практична цінність: система розгортається на стандартній веб-камері та ПК – суттєво знижує поріг впровадження систем моніторингу водія.

15

Перспективи подальших досліджень

1. Мультиmodalний аналіз стану

- Окрім стану очей – позивання, нахил і поворот голови та напрям погляду. Дасть змогу виявляти не лише сонливість, а й відволікання уваги водія від дороги.

2. Інтеграція з носимими пристроями

- Підключення смарт-годинника чи браслета для зчитування пульсу й варіабельності серцевого ритму. Фізіологічні дані підтверджують відеооцінку та виявляють втому раніше.

3. Активна безпека в автомобілі

- Інтеграція в бортові системи авто: якщо водій не реагує на тривогу – увімкнення аварійки та безпечна зупинка. Перехід від виявлення сонливості до запобігання аварії.

16

ДЯКУЮ ЗА УВАГУ!

Кудрявцев Іван Михайлович · КА-23 · 2026
НТУУ КПІ ім. Сікорського · НН ІПСА · Кафедра ММСА