ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ АВТОМАТИЗОВАНИХ СИСТЕМ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Інтелектуальна система виявлення та запобігання сонливості водія за кермом»

на здобуття освітнього ступеня бакалавра (магістра)

зі спеціальності 126 Інформаційні системи та технології
(код, найменування спеціальності)

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

_____ Владислав Посиняк
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: здобувач вищої освіти гр. __
Владислав Посиняк
Ім'я, ПРІЗВИЩЕ

Керівник: Ігор Сініцин, доктор технічних наук
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

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

Київ – 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення автоматизованих систем

Ступінь вищої освіти Магістр

Спеціальність 126 Інформаційні системи та технлогії

Освітньо-професійна програма 126 Інформаційні системи та технлогії

ЗАТВЕРДЖУЮ

Завідувач кафедрою _____
_____ Ім'я, ПРІЗВИЩЕ
«____» _____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
_____Посиняк Владислав Юрійович_____
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Інтелектуальна система виявлення та запобігання сонливості водія за кермом
керівник кваліфікаційної роботи Ігор Сініцин, доктор технічних наук
(Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «19» 10.2023 р. № 145
2. Строк подання кваліфікаційної роботи «29» грудня 2023р.
3. Вихідні дані до кваліфікаційної роботи: науково-технічна література, дані про поведінку водія, алгоритми обробки даних,
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
4.1. Аналіз предметної області
4.2. Аналіз алгоритмічних рішень виявлення та запобігання втомленості водія
4.3. Проєктування та розробка програмної системи виявлення та запобігання втомленості водія
5. Перелік ілюстративного матеріалу: презентація
6. Дата видачі завдання«19» жовтня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	19.10 - 05.11.23	
2	Аналіз проблематики виявлення сонливості водія за кермом	05.11 - 12.11.23	
3	Дослідження існуючих методів втомленості та сонливості водія	13.11 - 19.11.23	
4	Аналіз алгоритмічних рішень для виявлення та запобігання сонливості	20.11 - 25.11.23	
5	Проєктування, розробка та тестування алгоритму виявлення та запобігання втомленості водія за кермом	27.11 - 03.12.23	
6	Інтеграція розробленого алгоритму до програмної системи	04.12 - 10.12.23	
7	Оформлення роботи: вступ, висновки, реферат	11.12 - 20.12.23	
8	Розробка демонстраційних матеріалів	21.12 - 29.12.23	

Здобувач вищої освіти _____ Владислав Посиняк
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Ігор сініцин
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 77 стор., 50 рис., 30 джерел.

Мета роботи – дослідження методів та розробка програмної системи, яка зможе автоматично виявляти ознаки втомленості або сонливості водія під час керування транспортним засобом.

Об'єкт дослідження – процес розпізнавання втоми водія на даних, отриманих вбудованою камерою.

Предмет дослідження – методи та програмні засоби, які дозволяють виявляти, аналізувати та оцінювати ознаки втомленості або сонливості водія під час керування транспортним засобом.

Короткий зміст роботи: В першому розділі проведено аналіз предметної області. На основі розглянутих досліджень виявлено, що під час тривалих подорожей однією з найбільших небезпек, з якою стикаються водії, є ризик заснути за кермом. Досліджено існуючі системи виявлення та запобігання втомленості. Також, розглянуті існуючі методи виявлення та запобігання втомленості водія. В другому розділі проаналізовані існуючі алгоритмічні рішення виявлення сонливості водія, а саме: алгоритм Віоли-Джонса, алгоритм головних компонент, алгоритм локальних бінарних шаблонів та алгоритми глибого навчання. В третьому розділі кваліфікаційної роботи розглянуто проєктування та розробка програмної системи для виявлення та запобігання втомленості водія. Для виявлення втомленості водія використовується техніка співвідношення сторін очей (EAR).

КЛЮЧОВІ СЛОВА: ШТУЧНИЙ ІНТЕЛЕКТ, КОМП'ЮТЕРНИЙ ЗІР, OPENCV, РОЗПІЗНАВАННЯ ОБЛИЧЧЯ, БЕЗПЕКА ДОРОЖНЬОГО РУХУ.

ABSTRACT

The text part of the qualification work for obtaining the master's degree: 77 pages, 50 figures, 30 sources.

The purpose of the work is to research methods and develop a software system that can automatically detect signs of driver fatigue or drowsiness while driving a vehicle.

The object of the study is the process of driver fatigue recognition based on the data obtained by the built-in camera.

The subject of research is methods and software tools that allow detecting, analyzing and evaluating signs of driver fatigue or drowsiness while driving a vehicle.

Summary of the work: In the first section, an analysis of the subject area was carried out. Based on the studies reviewed, it was found that during long journeys, one of the biggest dangers faced by drivers is the risk of falling asleep at the wheel. The existing systems of detection and prevention of fatigue were studied. Also, existing methods of detecting and preventing driver fatigue are considered. In the second section, existing algorithmic solutions for detecting driver drowsiness are analyzed, namely: the Viola-Jones algorithm, the principal components algorithm, the local binary patterns algorithm, and deep learning algorithms. In the third section of the qualification work, the design and development of a software system for detecting and preventing driver fatigue is considered. The Eye Aspect Ratio (EAR) technique is used to detect driver fatigue.

KEY WORDS: ARTIFICIAL INTELLIGENCE, COMPUTER VISION, OPENCV, FACE RECOGNITION, ROAD SAFETY.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ADAS (англ. Advanced driver-assistance systems) – Удосконалена система допомоги водію.

EAR (англ. Eye Aspect Ratio) – Числове значення, яке кількісно визначає співвідношення сторін ока по вертикалі та горизонталі.

ВСТУП

Актуальність роботи. Сонливість при водінні може мати катастрофічні наслідки, навіть якщо вона триває лише кілька хвилин. Хоча виснаження зазвичай є основною причиною відчуття сонливості, що призводить до зниження пильності та уваги, існують інші причини, такі як відсутність концентрації, побічні ефекти ліків, розлади сну, вживання алкоголю або змінна робота. Через непередбачуваний характер сну людям важко передбачити, коли може виникнути сонливість, що становить серйозну небезпеку, особливо під час водіння. Водіння в сонному стані становить значні ризики, і небезпечно не лише заснути за кермом. Втома погіршує здатність керувати автомобілем, навіть якщо водієві вдається не спати. На жаль, статистика показує, що кожен двадцятий водій зізнався, що спав за кермом. Найбільш вразливою групою до нещасних випадків за кермом у стані сонливості є водії вантажівок і автобусів, чиї довгі поїздки тривалістю 10-12 годин наражають на небезпеку не тільки себе, але й інших. Водіння на довгі відстані з недостатнім сном може викликати сонливість, як і водіння, коли потрібно відпочити. За таких обставин сонливість водія є основним каталізатором будь-яких нещасних випадків, які можуть статися на дорозі. Згідно з даними Національного управління безпеки дорожнього руху (NHTSA) [1], поліцейські та лікарняні звіти показують, що приблизно 100 000 автомобільних аварій і понад 1500 смертельних випадків щорічно пояснюються сонним водінням. Згідно зі звітами NHTSA, залишається складним визначити точну кількість аварій і смертельних випадків, спричинених сонливістю. Насправді, за оцінками, водіння в сонному стані призводить до приблизно 1550 смертей, 71 000 травм і фінансових втрат у розмірі 12,5 мільярдів доларів США щороку. На жаль, лише у 2021 році сонливість зіграла роль у 697 летальних наслідках. NHTSA визнає, що наведені цифри, ймовірно, консервативні та не враховують повний обсяг аварій і смертельних випадків, спричинених сонним керуванням [2]. Один з переважаючих методів передбачає використання моделі керування рульовим керуванням, яка працює за допомогою функцій електричної системи рульового керування. Примітно, що ця техніка вимагає від водія активної

участі в керуванні транспортним засобом за допомогою керма, а не покладатися виключно на автоматизовану систему утримання в смузі руху. Завдяки такому підходу можна в режимі реального часу відстежувати поведінку водія під керуванням, що дозволяє своєчасно виявляти ознаки сонливості. Крім того, інший підхід передбачає використання камери моніторингу смуги для оцінки положення автомобіля під час контролю смуги. Так само цей метод передбачає активну участь водія в керуванні транспортним засобом. Вивчаючи положення автомобіля в межах смуги, можна виявити потенційні ознаки сонливості, що ще більше підвищує безпеку на дорозі. Останнім часом алгоритми машинного навчання все частіше використовуються в системах виявлення сонливості водія, пропонуючи точний підхід для точного прогнозування рівня сонливості. Ці інтелектуальні алгоритми мають здатність аналізувати та ідентифікувати закономірності в фізіологічних і поведінкових даних, зібраних від водіїв. Використовуючи різні функції, такі як рух очей, вираз обличчя та рухи голови, ці алгоритми ефективно виявляють випадки сонливості серед водіїв. Розробка та впровадження систем виявлення сонливості водія відіграє вирішальну роль у підвищенні безпеки дорожнього руху, спричинених втомою водія. Ці системи мають подвійну мету: попереджають водіїв про необхідність робити необхідні перерви, коли вони починають відчувати надмірну втому, таким чином значно знижуючи ризик аварій. Крім того, вони також забезпечують цінний зворотний зв’язок для менеджерів автопарків, пропонуючи корисну інформацію, яка може допомогти у формулюванні політики та вказівок, які заохочують водіїв робити достатні перерви, особливо коли вони виявляють, що стають надмірно втомленими [3]. Отже, системи виявлення сонливості представляють важливу область досліджень, яка може істотно підвищити безпеку дорожнього руху. Використовуючи ефективні та надійні технології виявлення, частота аварій може бути суттєво зменшена.

Метою кваліфікаційної роботи магістра є дослідження методів та розробка програмної системи, яка зможе автоматично виявляти ознаки втомленості або сонливості водія під час керування транспортним засобом.

Для досягнення мети кваліфікаційної роботи поставлені наступні **завдання**:

– Проаналізувати та систематизувати існуючі методи виявлення втомленості та сонливості водія, які використовуються в сучасних системах.

– Проектування та програмна реалізація алгоритму виявлення втомленості водія на основні фізіологічних показників руху очей.

– Інтеграція розробленого алгоритму і тестування програмної системи для визначення її ефективності та надійності визначення втомленості водія.

Об’єктом дослідження кваліфікаційної роботи магістра є процес розпізнавання втоми водія на даних, отриманих вбудованою камерою.

Предметом дослідження кваліфікаційної роботи магістра є методи та програмні засоби, які дозволяють виявляти, аналізувати та оцінювати ознаки втомленості або сонливості водія під час керування транспортним засобом.

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Проблематика виявлення сонливості водія за кермом

Перевтома та засипання за кермом реєструються як фактори невеликого відсотка ДТП в Україні, а саме приблизно від 80 до 160 випадків на рік [2].

Однак поширена думка, що ці цифри не точно відображають реальність на дорогах. Дослідження [3], присвячене водіям вантажівок, які займаються нічними перевезеннями, виявило наступні результати: понад 37% водіїв мали розлади сну, при цьому 12,5% мали важкі форми синдрому апное уві сні (SSA). Серед тих, хто мав розлади сну, дві третини повідомили про епізоди засинання під час водіння, а 42% потрапили в ДТП, які вони пояснювали підвищеною втомою. Викликає тривогу те, що 95,5% опитаних водіїв не усвідомлювали зв’язку між їхніми розладами сну та ймовірністю потрапити в дорожньо-транспортну пригоду під час водіння вночі. Наслідки перевтоми мають далекосяжний характер: статичне перенапруження опорно-рухового апарату, зниження сенсомоторних реакцій, зниження пам’яті та стійкості до одноманітності, а також підвищена дратівливість і агресивність.

Примітно, що більше чверті водіїв зазначають, що регулярно стикаються з епізодами засинання за кермом. За допомогою експерименту з моніторингу руху очей, проведеного на професійних водіях вантажівок, було виявлено, що в 14% випадків водіння водії відчували принаймні два епізоди закритих очей тривалістю більше 3 секунд. Варто відзначити, що кілька авторів стверджують, що втома і сонливість спричиняють 15-25% усіх ДТП [3, 4, 5].

Під час тривалих подорожей однією з найбільших небезпек, з якою стикаються люди, є ризик заснути за кермом. Цей ризик особливо помітний під час подорожей у нічний час та під час їзди одноманітними дорогами.

Дослідження [6] демонструє, що час реакції водія значно скорочується після чотирьох годин безперервної їзди, а до восьми годин він зменшується лише до однієї шостої від початкової здатності. Хоча автоматизація продовжує

контролювати рух у поїздах і літаках, навіть кілька секунд сну за кермом на дорозі потенційно можуть призвести до втрати життя.

Основна причина того, що водії несподівано засинають, широко відома – це насамперед пов’язано з надмірною втомою, яка є результатом сукупної нестачі сну або недостатньої якості сну. Найпоширенішим станом, що призводить до денної сонливості, є синдром обструктивного апное сну (СОАС). За статистичними даними з дослідження [8], СОАС страждає приблизно 5-8% осіб у віці 35 років і старше. Крім того, близько 3% всього дорослого населення має важку форму цього захворювання. Сонливість – складне явище, на яке впливають різні фактори. Розглядаючи це, важливо брати до уваги психофізіологію окремих осіб. Певні характеристики, такі як астенічність, а також залежність від міцної кави для початку дня, можуть зробити деяких людей більш схильними до раптових нападів сонливості порівняно з іншими. Іншим фактором ризику нападів сну є невідповідність або конфлікт між біоритмами людини та її робочим графіком, що може особливо вплинути на людей, які класифікують себе як «жайворонків», так і «сов». Крім того, ергономіка оточення, включаючи такі фактори, як мерехтливі вогні або недостатнє освітлення, також можуть впливати на рівень сонливості. Варто зазначити, що сонливість не пов’язана виключно з втомою; скоріше, на це також можуть впливати стани здоров’я, емоційні страждання та стани тривоги.

Система попередження про втому водія – це практичний інструмент, реалізований для моніторингу стану та пильності водія під час керування транспортним засобом, що рухається. Він охоплює три різні типи, кожен з яких розрізняється на основі аналізованих показників. Ці індикатори включають рухи водія під час поїздки, моніторинг фокусу або погляду водія та відстеження траєкторії автомобіля. Додатково враховуються тривалість поїздки, час доби, завантаженість систем керування автомобілем, спостереження за поглядом водія. Щоразу, коли в поведінці водія виявляються будь-які відхилення або порушення, вмикається комбінація звукових і світлових сигналів. Ці сигнали призначені для негайного привернення уваги водія та безперервно повторюються кожні 15 хвилин, доки поведінка водія не повернеться до нормального або попередження не буде

підтверджено. Впровадження таких передових систем безпеки довело високу ефективність у зниженні кількості аварій на дорогах, спричинених втомою водія. Завдяки ефективному використанню системи попередження про втому водія значно зменшується ризик, пов’язаний із сонливістю водія та наступними аваріями.

На рисунку 1.1 представлена концепція попередження втоми водія.

Рисунок 1.1 – Концепція системи попередження втоми водія

Сонливість – це психічний стан, який виникає внаслідок недостатнього сну і вважається формою розладу. Його не можна легко подолати, просто прокинувшись; єдиний вихід для боротьби з нею – забезпечити тілу належний відпочинок. На відміну від споживання алкоголю, яке можна виміряти за допомогою дихальних тестів, точно визначити рівень сонливості водія виявилося складним завданням. Однак завдяки більш ніж десятиріччю інтенсивних досліджень, використовуючи найсучасніші базові алгоритми відстеження голови та очей і використовуючи останні досягнення в машинному навчанні, Seeing Machines досягла можливості пропонувати систему моніторингу водія (DMS). Їх DMS розроблена для підвищення безпеки шляхом моніторингу рівня уваги та

бадьорості водія, головним чином спостерігаючи за рухами очей та обличчя. Основа системи ґрунтується на алгоритмах комп'ютерного зору, які постійно сканують та інтерпретують обличчя водія. Це включає в себе відстеження погляду очей, рухів повік та положення голови. Компанія Seeing Machines розширила алгоритми FOVIO eDME (вбудований механізм моніторингу драйверів) і оптимізоване та прискорене програмне забезпечення для процесора, яке забезпечує оптимізовану площу обробки, низьке розсіювання тепла та невеликі механічні розміри та вагу, необхідні для рішення на основі моніторингової системи.

1.2 Існуючі системи виявлення сонливості водія

Автоматизовані системи допомоги водієві були розроблені з метою допомогти водіям запобігти дорожньо-транспортним пригодам і зменшити їхній вплив. Ці системи віддають перевагу подачі високопріоритетних попереджувальних сигналів, які спонукають водіїв бути пильними та вживати своєчасних відповідних заходів у ситуаціях, коли існує високий ризик серйозних травм або неминучої небезпеки. Варто зазначити, що кожен виробник автомобілів налаштовує та адаптує ці системи відповідно до власних конкретних вимог і дизайну. Для ефективного відстеження сонливості водія використовуються численні технології, включаючи моніторинг кермового керування, моніторинг положення автомобіля щодо смуги руху, аналіз очей і рухів обличчя водія, а також фізіологічні вимірювання. Кожна автомобільна компанія може використовувати ці технології окремо або в поєднанні.

Система Audi «Rest Recommendation» – це функція безпеки, призначена для виявлення ознак втоми водія та пропозиції зробити перерву. Система в основному використовує поведінку керма, щоб визначити, чи може водій втомитися. Під час руху зазвичай вносяться невеликі, але постійні корекції керма. Втомлений водій може вносити менше незначних виправлень, а потім час від часу вносити більш суттєві виправлення – система розпізнає цю модель як потенційну втому. Коли система визначає такі закономірності, що вказують на втому або зниження уваги,

вона попереджає водія візуальним індикатором, як правило, значком чашки кави на панелі приладів, що супроводжується звуковим сигналом. Таким чином Audi рекомендує водієві відпочити або зробити перерву. Окрім поведінки керма, система також може враховувати інші змінні, такі як час їзди без перерв, час доби (наприклад, нічні поїздки) і дані з інших систем допомоги, щоб формувати обґрунтовані рекомендації. На рисунку 1.2 представлений приклад роботи системи «Rest Recommendation».

Рисунок 1.2 – Концепція роботи системи Audi «Rest Recommendation»

«Driver Attention Assist» від Mercedes – це система безпеки, призначена для виявлення ранніх ознак втоми або неуважності водія та своєчасного попередження. Driver Attention Assist контролює кілька параметрів, щоб проаналізувати поведінку водія, особливо протягом перших хвилин поїздки, створюючи індивідуальний профіль водія. Під час подорожі система постійно порівнює поточну поведінку керма з цим профілем, шукаючи закономірності, які можуть вказувати на сонливість або зниження уваги. Коли система виявляє, що водій ризикує заснути,

вона негайно попереджає його за допомогою комбінації візуальних і звукових сигналів.

Система Driver Attention Assist запрограмована на визначення певних моделей водіння, які часто свідчать про втому водія, і миттєво попереджає водія за допомогою візуальних попереджень, що відображаються на панелі приладів. Ці візуальні попередження представлені символом, що нагадує чашку кави, сигналізуючи водієві про необхідність зробити перерву.

Для додаткового привернення уваги водія також використовуються звукові сигнали, які забезпечують додатковий рівень попередження. На рисунку 1.3 представлений приклад роботи системи «Driver Attention Assist».

Рисунок 1.3 – Візуалізація роботи системи Mercedes «Driver Attention Assist»

Окрім швидкості, поперечного та поздовжнього прискорення, система Mercedes також визначає рухи керма, використання покажчиків повороту чи педалей і певні вхідні дані керування, не кажучи вже про зовнішні впливи, такі як бічний вітер або нерівності дороги, наприклад. Спостереження за поведінкою керма виявилося надзвичайно важливим, оскільки сонним водіям важко керувати точним курсом у своїй смузі. Вони допускають незначні помилки в кермуванні, які часто виправляються швидко і раптово. Інтенсивні випробування, проведені

інженерами Mercedes за участю понад 550 водіїв, показали, що цей ефект виникає на дуже ранній стадії, коли починається сонливість – часто перед небезпечною ситуацією, в якій водій миттєво засинає.

Система Volvo Driver Alert Control відстежує рухи автомобіля та оцінює, чи керується транспортним засобом контрольовано чи неконтрольовано. Якщо система виявляє високий ризик сонливості водія, водій попереджається за допомогою звукового сигналу. Крім того, текстове повідомлення з'являється на інформаційному дисплеї автомобіля, сповіщаючи його або її символом чашки кави, щоб зробити перерву. Крім того, водій може постійно отримувати інформацію про водіння з бортового комп'ютера автомобіля. Система попередження водія стежить за рухами автомобіля та оцінює, чи керується транспортним засобом контрольовано чи неконтрольовано. Оцінюючи дії керма та траєкторію автомобіля, система може визначити, чи може така поведінка вказувати на втому або відволікання водія. Якщо система визначить, що режим водіння нестабільний або вказує на сонливість водія, вона активує візуальне та звукове сповіщення. Як правило, візуальне сповіщення відображає значок чашки кави на панелі приладів автомобіля, натякаючи на те, що водієві необхідно зробити перерву. Чутливість системи можна регулювати, дозволяючи водіям налаштовувати її відповідно до своїх уподобань. Крім того, для деяких моделей Volvo систему можна вмикати або вимикати, що дає водіям більше контролю над тим, коли вони хочуть, щоб Driver Alert Control був активним. DAC в основному призначений для високих швидкостей і зазвичай працює, коли транспортний засіб рухається зі швидкістю вище певного порогу, наприклад 40-140 км/год (ці значення можуть відрізнятися залежно від моделі та ринку). Це робить його особливо корисним під час тривалих поїздок по шосе, де ймовірність виникнення втоми водія. У деяких моделях Volvo Driver Alert Control є частиною ширшого набору технологій безпеки, які часто працюють у парі з іншими системами. як Lane Keeping Aid. Цей комплексний підхід гарантує, що автомобіль надає кілька рівнів попереджень безпеки та втручання. На рисунку 1.4 представлений приклад роботи системи Volvo Driver Alert Control.

Рисунок 1.4 – Візуалізація роботи системи Volvo «Driver Alert Control»

DS Driver Attention Monitoring визначає будь-яке зниження уважності водія. Камери для водія, встановлені над кермом і у верхній частині вітрового скла, відстежують три основні фізичні ознаки відволікання або сонливості; рух очей, повік або шиї. У разі виявлення будь-якого з них вмикається звуковий сигнал, а на дисплеї цифрових приладів з'являється попереджувальне повідомлення. У той же час система моніторингу положення автомобіля постійно стежить за автомобілем щодо дорожньої розмітки та попереджає водія звуковим сигналом про будь-які раптові або несподівані рухи керма.

Система моніторингу має унікальну особливість, яка вирізняє її серед інших - її можна використовувати як вдень, так і вночі. Крім того, його інфрачервоні датчики мають чудову здатність проникати крізь сонцезахисні окуляри, забезпечуючи їм ефективну роботу. Однією з важливих функцій цієї системи є її здатність виявляти, коли водій відволікається на свій телефон під час дороги. Розцінюючи цю дію як несправність, система негайно видає попередження на центральний екран автомобіля, що супроводжується звуковим сигналом. Це гарантує, що водій буде негайно попереджений про потенційні ризики, пов'язані з

відволіканням від керування. На рисунку 1.5 представлений приклад роботи системи DS Driver Attention Monitoring.

Рисунок 1.5 – Візуалізація роботи системи DS «Driver Attention Monitoring»

1.3 Існуючі методи виявлення втомленості та сонливості водія

Система розпізнавання обличчя – це технологія, яка має здатність виявляти та ідентифікувати особу на цифровому зображенні чи відеокадрі. Ці системи використовують різні методи для досягнення точних результатів, але основний принцип передбачає порівняння чітких атрибутів обличчя даного зображення з уже існуючими даними обличчя, які зберігаються в базі даних. Цей процес, який можна назвати біометричним додатком на основі штучного інтелекту, дозволяє системі унікально ідентифікувати особу шляхом ретельного аналізу складних візерунків, отриманих із текстури людського обличчя. Завдяки індивідуальності структур обличчя спеціальне програмне забезпечення має можливість досліджувати ці чіткі шаблони та порівнювати їх з інформацією, що зберігається в базах даних, що дозволяє подальшу ідентифікацію відповідної особи. На рисунку 1.6

представлений приклад біометричної ідентифікації людини на зображенні з використанням ШІ.

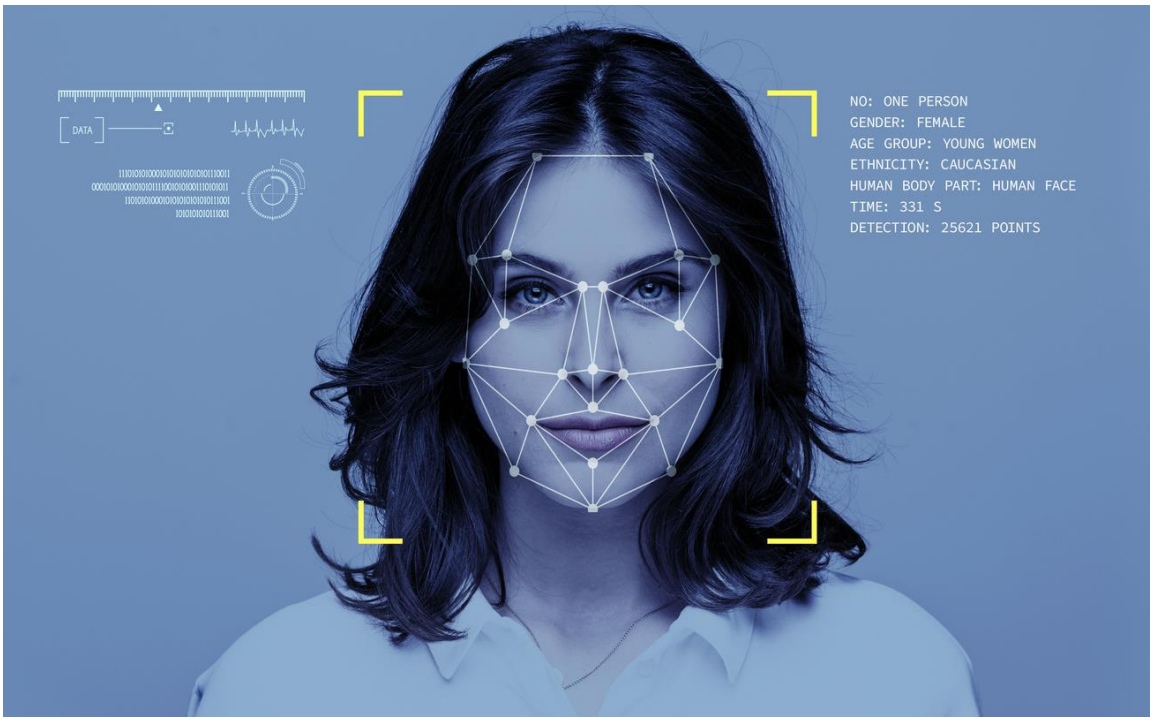


Рисунок 1.6 – Концепція біометричної ідентифікації людини на зображенні

Алгоритми розпізнавання обличчя використовують різні методи для аналізу рис обличчя з метою ідентифікації. У деяких випадках ці алгоритми використовують певні ключові моменти або характеристики, присутні на зображенні обличчя суб’єкта. Ці ключові моменти можуть включати взаємне розташування, розмір і форму очей, носа та інших помітних рис. З іншого боку, певні алгоритми попередньо нормалізують колекцію зображень обличчя, згодом стискаючи відповідні дані обличчя, зберігаючи лише необхідну інформацію для точного розпізнавання обличчя. Більш успішний підхід передбачає використання методів зіставлення шаблонів, застосованих до вибору рис обличчя. Ця техніка ефективно стискає опис обличчя, що призводить до короткого, але всебічного представлення обличчя суб’єкта. Коли тестове зображення представлено, воно порівнюється із збереженими даними, які відповідають рисам обличчя, що полегшує процес ідентифікації.

Алгоритми розпізнавання можна розділити на два основні підходи: геометричний підхід і фотометричний підхід. Геометричний підхід пов’язаний з визначенням характерних особливостей, тоді як фотометричний підхід використовує статистичні методи для перетворення зображення в ряд значень. Потім ці значення порівнюються з попередньо визначеними шаблонами, щоб усунути будь-які розбіжності. Крім того, ці алгоритми можна додатково класифікувати на дві великі категорії: цілісні та на основі функцій.

Цілісні алгоритми спрямовані на розпізнавання обличчя в повному обсязі, тоді як алгоритми на основі ознак розбивають обличчя на окремі компоненти, наприклад відповідні ознаки, і аналізують кожен компонент разом із його просторовим співвідношенням з іншими об’єктами.

Враховуючи як геометричний, так і фотометричний підходи, а також цілісні категорії та категорії на основі ознак, алгоритми розпізнавання можуть ефективно ідентифікувати та аналізувати обличчя з високою точністю.

Технологія розпізнавання обличчя не позбавлена обмежень. Вона може зіткнутися з проблемами та стати менш надійною за певних обставин. Наприклад, такі фактори, як погані умови освітлення або зміни положення голови та кута, можуть значно погіршити точність розпізнавання. Для розробки алгоритмів розпізнавання облич використовуються різні підходи, одним із яких є емпіричний метод. Ця техніка передбачає застосування правил, які імітують процеси ідентифікації людини, такі як аналіз яскравості чола, рівномірності кольору та яскравості в центральній частині обличчя та наявності ключових ознак, таких як ніс, рот і очі. Щоб точно визначити людей на зображеннях, область пошуку часто звужується або генеруються перпендикулярні гістограми. Однак слід зазначити, що ці методи, незважаючи на те, що вони відносно прості, не завжди дають оптимальні результати, особливо коли на задньому плані є кілька об’єктів, у кадрі зображено кілька людей або відбуваються значні зміни кутів зображення. Таким чином, важливо розуміти, що незважаючи на прогрес, досягнутий у технології розпізнавання обличчя, контекстуальні фактори все ще можуть впливати на її ефективність.

Наступна техніка, яка використовується для виявлення ідентичності, передбачає використання попередньо визначених шаблонів, встановлених розробником. Ці шаблони діють як шаблони для зовнішнього вигляду людських облич, дозволяючи алгоритму ідентифікувати та зіставляти їх із різними сегментами зображення, незалежно від їхніх кутів чи масштабу. Хоча це вимагає значних обчислювальних ресурсів, сучасні технології розпізнавання облич просунулися настільки, що тестові зображення можна використовувати як навчальний матеріал. Це передбачає використання баз даних зображень, які складаються як із зображень людей, так і не людей, для всебічного навчання системи. На рисунку 1.7 представлений приклад спеціально розробленої бази даних зображень для тренування системи.

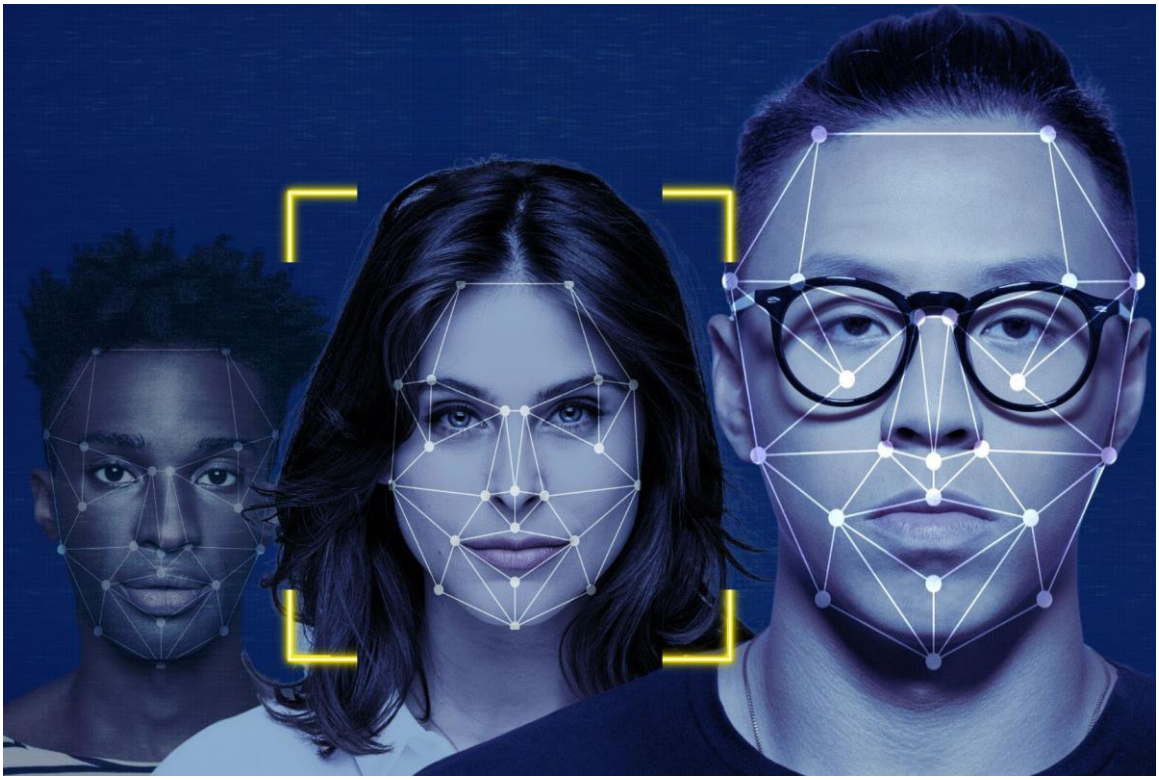


Рисунок 1.7 – База даних зображенн для технології розпізнавання обличчя

1.3.1 Метод прихованих марковських моделей

Приховані моделі Маркова (HMM) набули значної популярності як надійний інструмент для моделювання широкого діапазону процесів і полегшення

розпізнавання образів. HMM вирізняються своєю здатністю ефективно фіксувати просторово-часові характеристики сигналів, завдяки чому вони широко використовуються в різних сферах, таких як розпізнавання мови, а останнім часом і в аналізі зображень обличчя.

Кожна HMM складається з набору N станів, які можуть переходити один від одного. У будь-якому конкретному випадку система знаходиться в строго визначеному стані, і в традиційних марковських моделях першого порядку цей стан залежить виключно від поточного стану. Крім того, при переході з одного стану в інший генерується спостережуваний символ, що представляє фізичний сигнал, отриманий з виходу змодельованої системи. Варто зазначити, що вихід моделі може бути дискретним або безперервним, залежно від характеру програми. Крім того, існують HMM, які використовують однаковий набір символів для всіх станів, таким чином зберігаючи узгодженість у всій моделі.

На рисунку 1.8 представлений приклад використанням прихованих марковських моделей для технології розпізнавання обличчя.

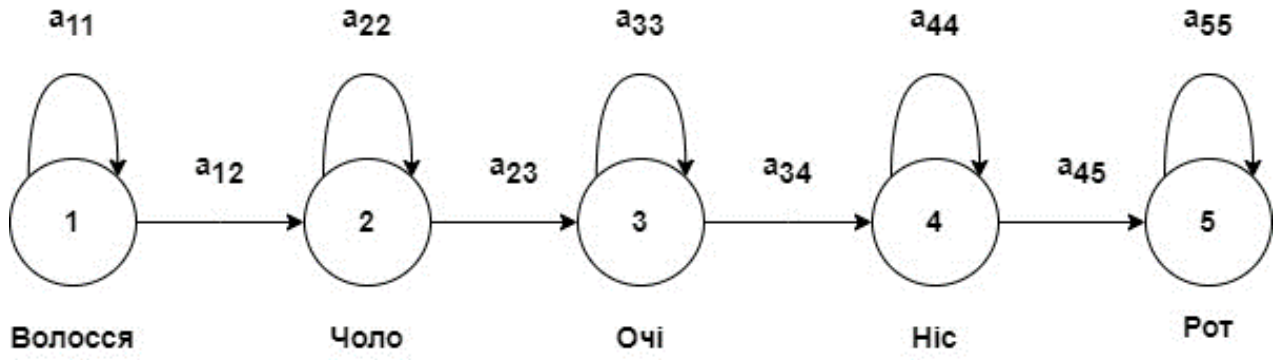


Рисунок 1.8 – Використання HMM для технології розпізнавання обличчя

Розпізнавання мовлення використовує лінійні моделі як засіб мінімізації обчислювальних ресурсів. Ці моделі розроблені таким чином, що кожен стан має один наступний стан, що дозволяє потенційно повернутися до того самого стану. Крім того, ці моделі враховують часові атрибути мовного сигналу, включаючи послідовний порядок сегментів сигналу, їх відносне положення та потенціал для

локалізованого розширення або стиснення. Беручи до уваги ці фактори, лінійні моделі ефективно використовуються у сфері розпізнавання зображень, таким чином підвищуючи їхню застосовність у різних контекстах.

Основна концепція двовимірних марковських моделей полягає в тому, що вони пропонують унікальний підхід до моделювання спотворення зображення та просторового співвідношення між різними областями. На відміну від одновимірних лінійних ПММ, які розглядають лише горизонтальні або вертикальні аспекти окремо, двовимірні моделі Маркова враховують обидва напрямки одночасно. Це дозволяє більш повно зрозуміти спотворення зображення та взаємне розташування різних ділянок. Щоб спростити обчислювальну складність, використовуються псевдодвовимірні ПММ. Ця модель складається з кількох лінійних вертикальних моделей на нижньому рівні та однієї лінійної горизонтальної моделі на верхньому рівні. Виходи моделей нижнього рівня подаються як вхідні дані моделі верхнього рівня. Кожен стан моделі верхнього рівня містить послідовність станів з відповідної моделі нижнього рівня. Варто зазначити, що моделі нижчого рівня незалежні одна від одної. Спочатку моделі верхнього рівня були вертикальними. Проте подальші дослідницькі зусилля розширили моделі, включивши в них горизонтальні компоненти, що дозволило вертикальним моделям нижчого рівня враховувати варіації у висоті очей. Таким чином, псевдо-двовимірна модель ефективно обробляє локальні деформації та взаємне розташування ділянок зображення. На відміну від оптичних потоків та інших методів відображення деформацій, псевдо-двовимірна модель враховує природу деформацій і дізнається про можливі варіації в процесі навчання. В результаті, це гарантує, що область, яка відповідає, наприклад, оку, ніколи не буде співпадати з областю, де розташований рот. Загалом, двовимірна модель Маркова, зокрема псевдодвовимірний варіант, пропонує потужну основу для розуміння спотворення зображення та зв’язків між різними розділами.

Це забезпечує глибше розуміння локальних деформацій і забезпечує точне зіставлення відповідних областей аналізованого зображення.

У сфері моделей зіставлення шаблонів початкова ініціалізація моделі відіграє вирішальну роль. Для ініціалізації моделей використовуються всі зображення з навчального набору. Зокрема, кожен клас пов’язаний з моделлю, а для його представлення призначається зображення. Однак ПММ має суттєвий недолік у тому, що йому не вистачає дискримінаційної здатності. Це означає, що алгоритм навчання зосереджується виключно на максимізації реакції кожної моделі на відповідний клас, без мінімізації реакції на інші класи або ефективного визначення ключових відмінних ознак між різними класами.

1.3.2 Метод гнучкого порівняння на графах

Основа методу гнучкого порівняння на графах, також відомого як Elastic Graph Matching, зводиться до еластичного порівняння графів, які представляють зображення обличчя. У цьому конкретному методі, який називається зіставленням еластичного пучка графів, обличчя зображується як графік, де вершини стратегічно розташовані на значущих орієнтирах обличчя, таких як контури голови, губ, носа та їхні крайні точки. Ці вершини характеризуються відстанями між ними. Для аналізу обличчя в кожній ключовій точці розраховуються коефіцієнти розширення функцій Габора для п’яти різних частот і восьми орієнтацій. Отриманий набір коефіцієнтів, позначений як $J = \{J_j\}$, відомий як джет. Джети виконують подвійну роль: по-перше, вони допомагають ідентифікувати відповідні точки в заданій області на двох різних зображеннях, а по-друге, вони дозволяють порівнювати відповідні області на різних зображеннях. Кожен коефіцієнт $J_j = aj_{exp}(i\varphi_j)$ в одній області різних зображень характеризується двома основними атрибутами. Перший – це повільно змінювана амплітуда aj, яка змінюється зі зміною положення точки. Другим атрибутом є фаза φ_j, яка обертається зі швидкістю, пропорційною частоті базового хвильового вектора базової функції. Враховуючи ці коефіцієнти, можна отримати повне розуміння рис обличчя та їх варіацій. У найпростішому сценарії, шукаючи точки зі схожими атрибутами на новому зображенні, функція подібності не враховує фазу.

На рисунку 1.9 представлена візуалізація концепції використання графів на основі антропометричних точок обличчя.

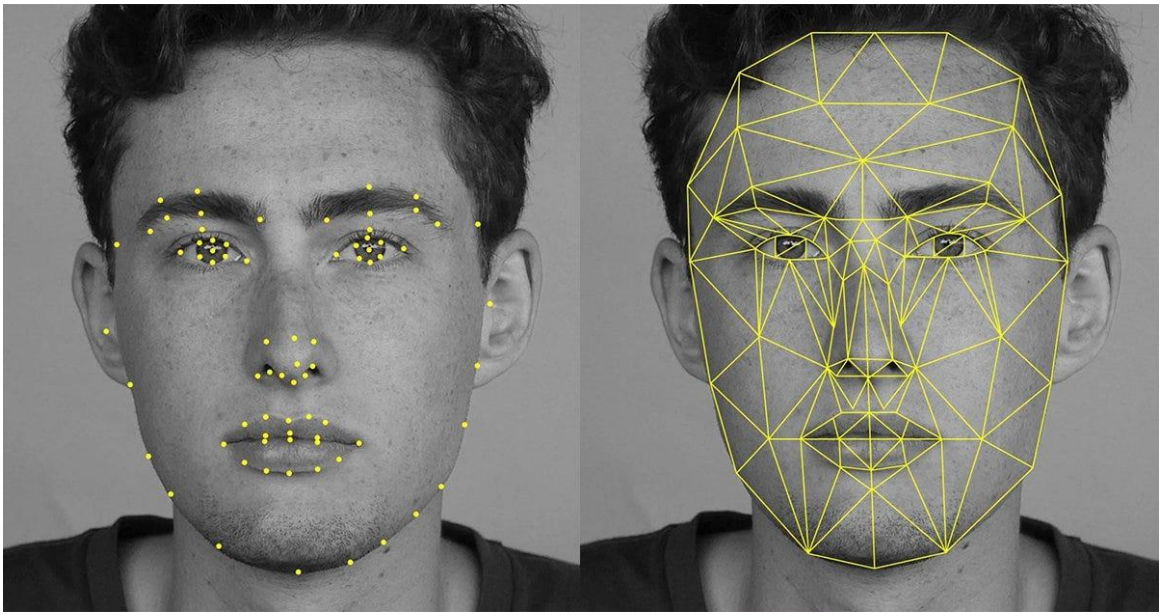


Рисунок 1.9 – Концепція використання графів на основі антропометричних точок обличчя для розпізнавання обличчя [12]

Цей окремий випадок описується наступною формулою (1.1):

$$S_a = \frac{\sum_j a_j a'_j}{\sqrt{\sum_j a_j^2 \sum_j a_j^{r^2}}} \tag{1.1}$$

Функція, яка вимірює подібність між двома точками, де одна є фіксованою, а інша може змінюватися, демонструє плавність, що сприяє швидкому та надійному збіжності під час пошуку з використанням простих методів, таких як дифузія або градієнтний спад. Більш просунуті функції подібності містять інформацію про фазу [16].

Щоб врахувати різні кути, відповідні ключові точки визначаються вручну в навчальному наборі. Крім того, для представлення кількох варіацій одного обличчя на одному графіку використовується кілька точок, кожна з яких відповідає окремим локальним характеристикам цієї точки, наприклад, відкрите або закрите око. Процес розпізнавання незнайомої особи включає порівняння графіка зображення

обличчя G' з усіма іншими графіками в наборі B, використовуючи функцію подібності, описану у формулі (1.2):

$$S_b(G', B) = \frac{1}{N}\sum_n \max_m S_\varphi\left(J_n^I, J_n^{Bm}\right) - \frac{\lambda}{E}\sum_e \frac{\left(\Delta x_e^I - \Delta x_e^B\right)}{\left(\Delta x_e^B\right)^2} \quad (1.2)$$

де λ – коефіцієнт важливості топографічної інформації, E – кількість граней, N – кількість вершин.

Ліва сума в цьому методі представляє спосіб вимірювання подібності між джетами. Визначається за допомогою фазочутливої функції. Права сума вказує на топографічну відповідність між джетами. Ця відповідність пропорційна квадрату різниці відстаней між відповідними вершинами порівнюваних зображень. У своїй поточній формі цей метод здатний точно виявляти зміни кутів до 20°. Однак при роботі з більшими кутами точність розпізнавання значно знижується. Варто зазначити, що функція подібності більш чутлива до змін кута, ніж до відмінностей між класами.

Для подальшого розвитку вищеописаного методу коефіцієнти важливості отримуються шляхом аналізу навчальної вибірки. Ці коефіцієнти розраховуються для кожного джета за допомогою симплексного методу та пізніше включаються у функцію подібності. Коефіцієнти важливості виводяться на основі принципу максимізації функції подібності для однакових граней і мінімізації її для різних граней. Важливо зазначити, що існують більш ранні варіації цього методу, які спочатку не включають певні ключові точки та структури графіків. Ці варіації використовуються для порівняння масивів джетів, накладених на зображення.У процесі пошуку точок відповідності в невідомому зображенні за допомогою цих ідентифікованих точок будується спотворена сітка. Потім вимірюється ступінь спотворення цієї сітки, щоб визначити зображення, яке має найбільшу схожість. В альтернативних підходах точки, виділені із зображення, спочатку організовуються в сітку, а на етапі навчання відфільтровуються найменш помітні точки.

Метод гнучкого порівняння на графах має деякі недоліки, а саме високу обчислювальну складність і обмежену здатність ефективного запам’ятовування нових стандартів, що може вплинути на її результативність.

1.3.3 Метод штучних нейронних мереж

Архітектура багатошарової нейронної мережі (БНМ) складається з взаємопов’язаних рівнів, кожен з яких складається з нейронів, які пов’язані з нейронами попереднього рівня як входи та з нейронами наступного рівня як виходи. Коли НМ має два рівні прийняття рішень, вона має здатність апроксимувати будь-яку багатовимірну функцію з високою точністю. Однак, якщо НМ має лише один рівень прийняття рішень, вона може створювати лише лінійні розділові поверхні, що значно обмежує типи проблем, які НМ може вирішити. Наприклад, мережа лише з одним рівнем прийняття рішень не може ефективно вирішувати проблеми типу «що виключає або». З іншого боку, НМ з нелінійною функцією активації та двома рівнями рішень може формувати опуклі області в просторі рішень, тоді як НМ з трьома рівнями рішень може створювати області будь-якої складності, включаючи невипуклі. Важливо, що узагальнююча здатність БНМ не порушується в цьому процесі.

Для навчання MNN використовується алгоритм зворотного поширення помилки, який передбачає коригування вагових коефіцієнтів у спосіб градієнтного спаду, щоб мінімізувати загальну помилку мережі. Цей алгоритм поширює помилки, представлені як значення корекції ваги, у зворотному напрямку від входів до виходів уздовж з’єднань між нейронами.

Загалом, архітектура та можливості багатошарової нейронної мережі залежать від кількості рівнів прийняття рішень, типу використовуваної функції активації та конкретної проблеми, яку вона має вирішити.

Фундаментальне використання одношарової нейронної мережі, відомої як автоасоціативна пам’ять, полягає в тому, щоб навчити мережу реконструювати представлені зображення. Надавши тестове зображення та оцінивши якість реконструйованого зображення, можна оцінити здатність мережі точно розпізнавати вхідне зображення. Перевага цього методу полягає в тому, що він

здатний відновлювати спотворені та шумні зображення. Однак він може не підходити для більш критичних застосувань. Ще один спосіб використання нейронних мереж для обробки зображень – це пряма класифікація зображень. У цьому підході вхідними даними може бути або саме зображення, або набір ключових характеристик, отриманих із зображення. Потім вихідний нейрон із найвищою активністю вказує на класифікацію зображення до розпізнаного класу. У розпізнаванні зображень нейрон з найвищим рівнем активності, як правило, перший нейрон у мережі, вказує, що вхідне зображення належить до розпізнаного класу. Однак, якщо рівень активності падає нижче певного порогу, вважається, що зображення не належить до жодного відомого класу. Цей процес, відомий як «навчання з учителем», передбачає зіставлення вхідних зображень із відповідним класом.

Цей підхід зазвичай використовується в задачах контролю доступу, пов’язаних із невеликою групою облич, зокрема під час розпізнавання зображень людських облич. Шляхом прямого порівняння самих зображень цей метод дозволяє отримати пряму оцінку. Однак із збільшенням кількості класів як час навчання, так і продуктивність мережі зростають експоненціально. Отже, коли справа доходить до таких завдань, як пошук схожої особи у великій базі даних, виникає необхідність вивести короткий набір ключових характеристик. Ці характеристики служать основою для проведення пошуку, забезпечуючи ефективні та ефективні результати.

У сфері класифікації зображень одним із конкретних підходів є використання БНМ, яка враховує різні характеристики обличчя, як відстані між окремими частинами обличчя, такими як ніс, рот і очі. Крім того, існують гібридні системи, які поєднують БНМ з іншими моделями, такими як модель Маркова. У класичній БНМ нейронні зв’язки встановлюються між шарами, щоб представити зображення як одновимірний вектор, незважаючи на притаманну йому двовимірність. Згорткова архітектура НМ усуває ці обмеження шляхом включення локальних рецепторних полів, що забезпечує локальне двовимірне з’єднання нейронів. ЗНМ також використовує загальні ваги для виявлення особливостей у всьому

зображенні, незалежно від їх розташування. Ця ієрархічна організація разом із просторовою субдискретизацією дозволяє складним НМ забезпечувати частковий опір різним змінам, таким як масштаб, зрушення, повороти та спотворення [12].

На рисунку 1.10 представлено архітектуру типової згорткової нейронної мережі для використання в сфері класифікації зображень.

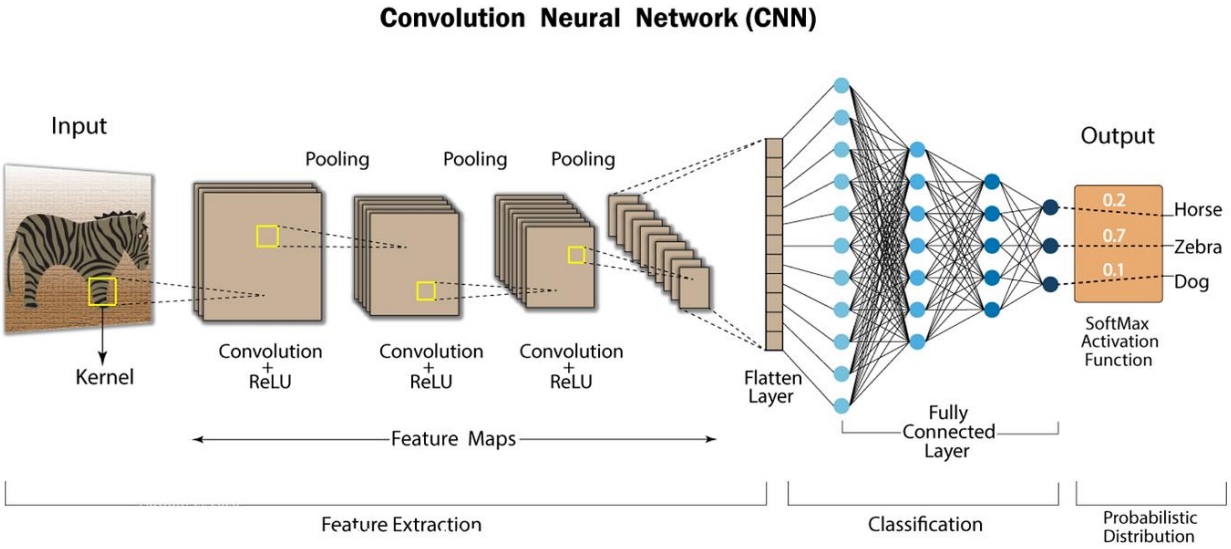


Рисунок 1.10 – Типова архітектура згорткової нейронної мережі для використання в сфері класифікації зображень [15]

Переходячи до архітектури ЗНМ (згорткова нейронна мережа), вона складається з кількох шарів, кожен із яких має кілька площин. Нейрони в кожному наступному шарі пов’язані лише з невеликою кількістю нейронів попереднього шару, розташованих на периферії локальної області, імітуючи організацію, знайдену в зоровій корі головного мозку людини. Крім того, ваги в кожній площині залишаються незмінними (згорткові шари).

Після згорткового шару є рівень зменшення розмірності, який досягає цього за допомогою локального усереднення. Потім цей процес повторюється з іншим згортковим шаром і так далі, що призводить до ієрархічної організації.

Пізніші рівні в архітектурі відповідають за виділення більш загальних функцій, на які не сильно впливають спотворення зображення. Згорткова нейронна мережа (ЗНМ) навчається за допомогою стандартного методу зворотного

поширення для виправлення помилок, як описано в дослідженні [20]. У порівнянні з багатошаровою нейронною мережею (БНМ), ЗНМ демонструє значні переваги як у швидкості, так і в надійності для цілей класифікації. Однією з примітних особливостей згорткової нейронної мережі (ЗНМ) є те, що характеристики, сформовані на вищих рівнях мережі, можна використовувати для класифікації за допомогою таких методів, як підхід найближчого сусіда, де обчислюється евклідова відстань. ЗНМ успішно вивчає ці характеристики навіть для зображень, яких немає в навчальному наборі. ЗНМ демонструє вражаючу швидкість як у навчанні, так і у виконанні, що робить її перспективною архітектурою для майбутніх розробок у сфері розпізнавання просторових об’єктів. Крім того, БНМ також використовується для виявлення конкретних типів об'єктів. Крім того, будь-яка навчена БНМ має певну здатність визначати ймовірність того, що зображення належать до призначених йому класів, і його можна спеціально навчити надійно виявляти певні класи. У цьому сценарії вихідні класи представлятимуть класи, до яких належить і не належить зображення. Загалом, ЗНМ з її надійними можливостями класифікації та БНМ з її функціями виявлення об’єктів пропонують цінний внесок у сферу розпізнавання та класифікації зображень.

РОЗДІЛ 2 АНАЛІЗ АЛГОРИТМІЧНИХ РІШЕНЬ ВИЯВЛЕННЯ ТА ЗАПОБІГАННЯ ВТОМЛЕНОСТІ ВОДІЯ

2.1 Алгоритм Віоли-Джонса

Алгоритм Віоли-Джонса, який був представлений Полом Віолою та Майклом Джонсом у 2001 році [18], є алгоритмом, розробленим для виявлення об’єктів на зображеннях у реальному часі. Незважаючи на те, що цей метод був розроблений два десятиліття тому, він все ще має величезне значення для виявлення облич на зображеннях. До широкого впровадження глибокого навчання для виявлення та розпізнавання облич, метод Віоли-Джонса виділявся своєю винятковою продуктивністю та швидкістю, що робило його одним із найефективніших доступних методів. Важливо зазначити, що основна увага методу Віоли-Джонса полягає в ідентифікації розташування обличчя на зображенні. Однак, якщо метою є розпізнавання обличчя та ідентифікація конкретної людини, цей метод можна поєднати з існуючими алгоритмами розпізнавання обличчя. Використовуючи ефективність методу Віоли-Джонса в поєднанні з іншими методами, стає можливим досягти значного підвищення продуктивності та точності.

Цей метод складається з трьох основних етапів:

– Формування інтегрального представлення зображення.
– Пошук облич на зображенні за допомогою ознак Хаара.
– Навчання класифікатора з використанням процедури Boosting.

Метод Віоли-Джонса використовує зображення в інтегральній формі. Це представлення зображень дозволяє швидко обчислити загальну інтенсивність у будь-якій прямокутній області зображення, незалежно від його розміру. Інтегральне зображення – це матриця, яка має ті самі розміри, що й оригінальне вхідне зображення. Кожен елемент цієї матриці представляє суму інтенсивностей усіх пікселів, розташованих ліворуч і вище цього конкретного елемента, включаючи значення самого елемента. Обчислюючи цю інтегральну матрицю,

метод гарантує, що час, необхідний для обчислення, залишається постійним, незалежно від розміру зображення.

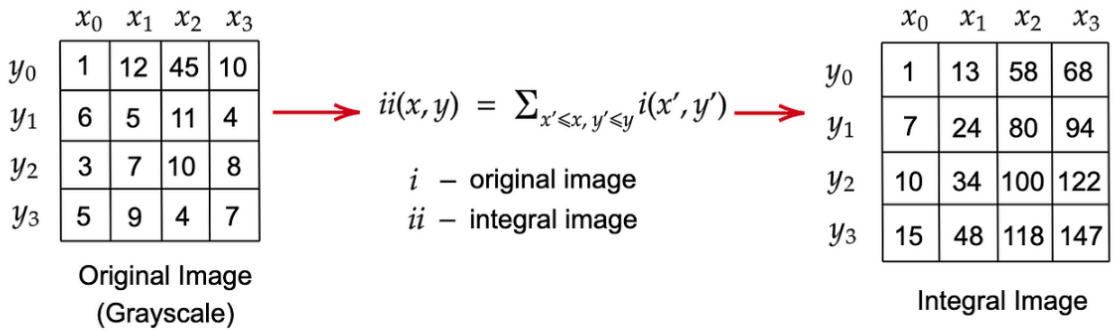
Варто зазначити, що час, необхідний для обчислення інтегральної матриці, прямо пропорційний кількості пікселів, присутніх на зображенні.

Нижче представлена формула для розрахунку інтегральної матриці (2.1):

$$L(x,y) = \sum_{i=0,j=0}^{i\leq x, j\leq y} A(i,j) \tag{2.1}$$

де $A(i,j)$ – насиченість пікселя вихідного зображення.

На рисунку 2.1 представлений приклад матриці початкового зображення, а також його інтегральної матриці.



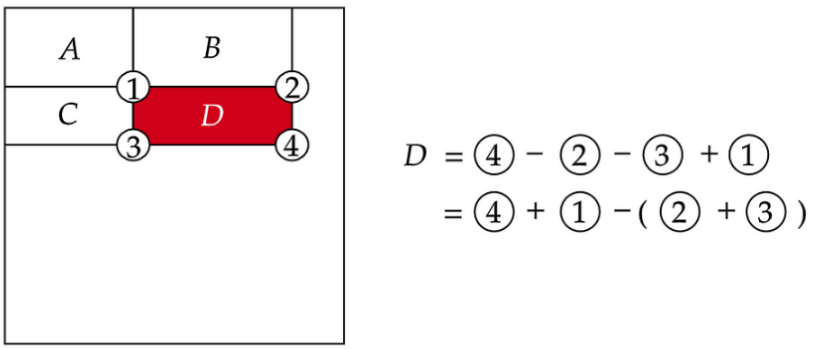


Рисунок 2.1 – Приклад отримання інтегральної матриці зображення

Припустимо, що існує набір об’єктів, який називається набором A, і відповідний набір відповідей, позначений набором B. Коли функція g: $A \rightarrow B$ встановлена для прийняття рішень, вона відома як вирішальна функція. Кожен об’єкт у наборі A має специфічну ознаку, позначену як f, яка представляє відображення з множини A на набір допустимих значень ознак, позначених як D_f. Якщо надано набір ознак, вектор $x = f_1(a), \dots, f_n(a)$ представляє опис ознак об’єкта a, який належить до множини A. Одночасно множина $A = D_{f1} \times \dots \times D_{fn}$ називається простором ознак, де знаходяться всі ці об’єкти.

Під час другої фази алгоритму Віоли-Джонса використовується важливий компонент, який називається маскою Хаара. Ця маска, яка набуває чотирикутної форми, складається з кількох прямокутних секцій, які розташовані поруч. Ці прямокутники згруповані з різними розмірами за висотою та шириною, і їх можна розмістити в різних місцях зображення. На рисунку 2.2 представлений приклад представлення примітивів Хаара.

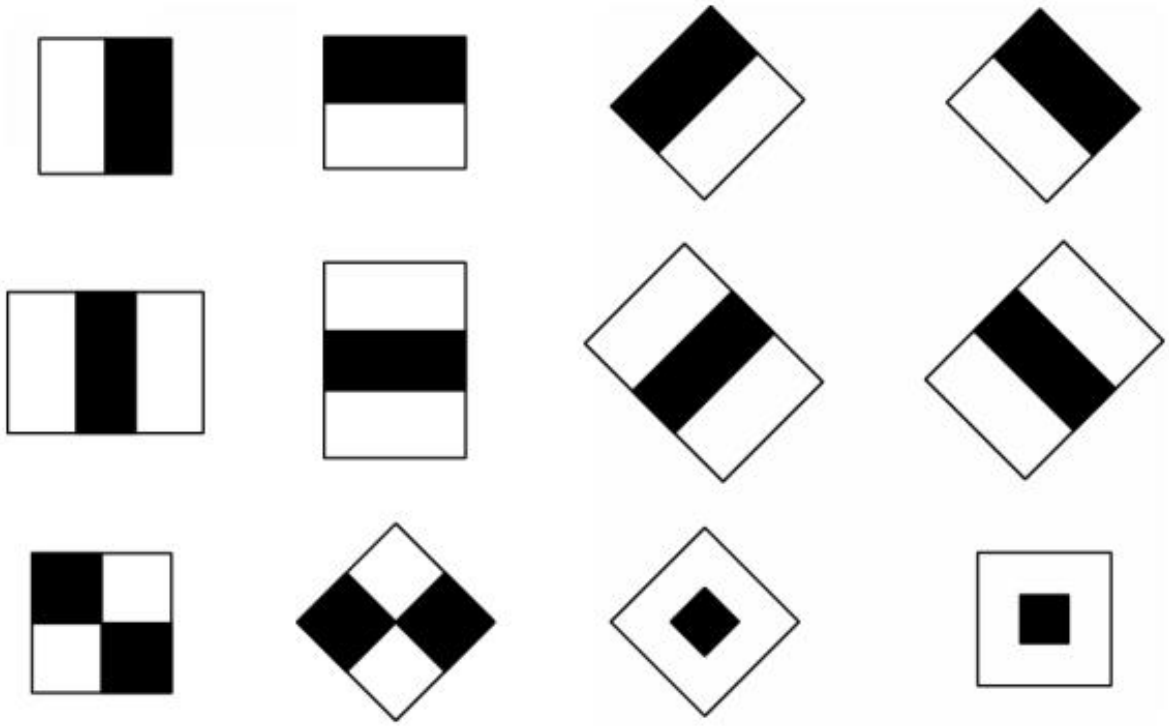


Рисунок 2.2 – Візуалізація примітивів Хаара

Їх призначення – виявити на зображенні ділянки, які демонструють контрастні характеристики, оскільки ці контрастні області часто вказують на наявність особливих ознак. Маска Хаара систематично перетинає зображення послідовним чином, динамічно змінюючи його положення та розмір.

На рисунку 2.3 представлений приклад використання ознак (або примітивів) Хаара для для вилучення ознак обличчя на зображеннях.

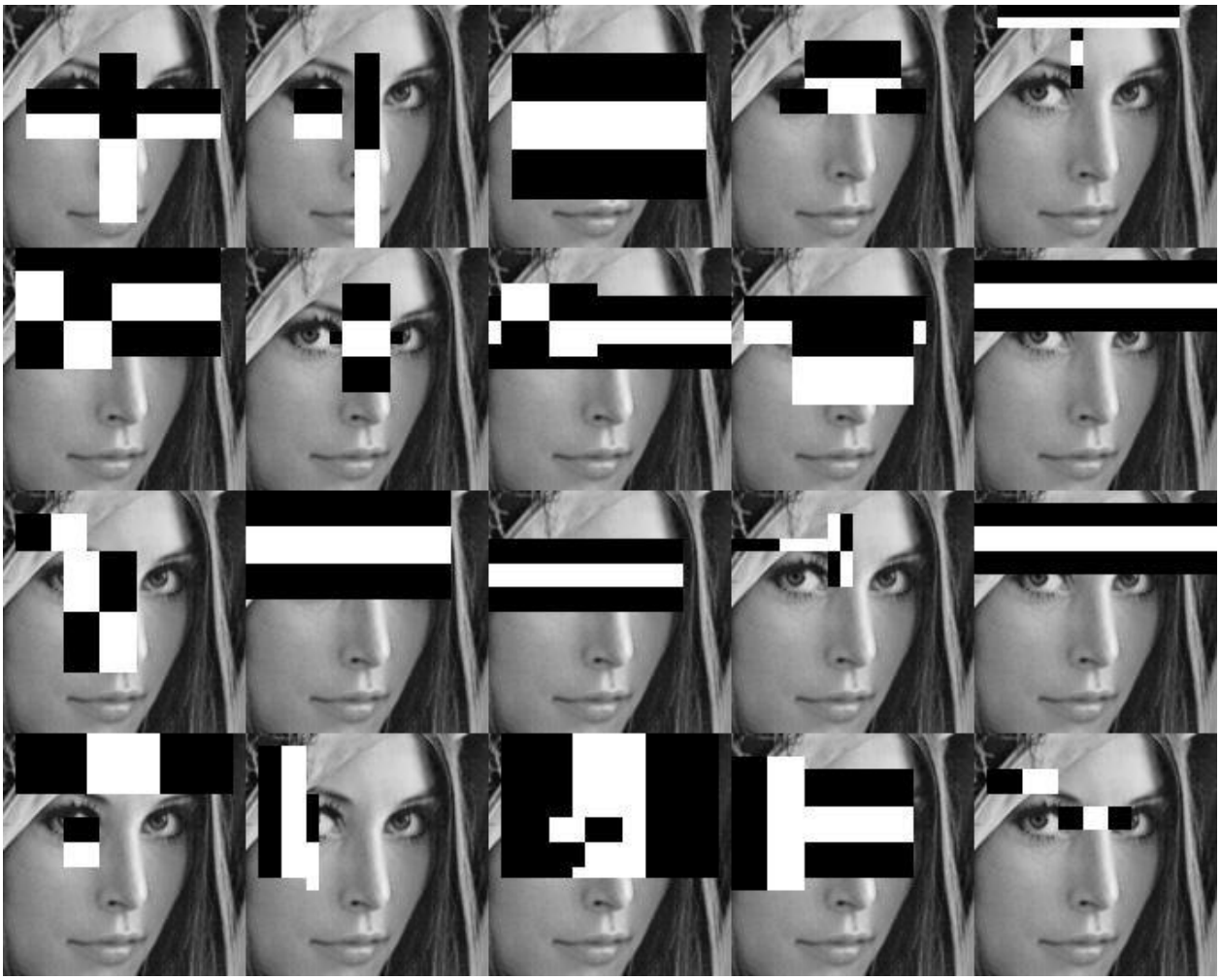


Рисунок 2.3 – Використання ознак (примітивів) Хаара для вилучення ознак обличчя на зображеннях

Використовуючи інтегральне зображення, алгоритм швидко обчислює ознаки Хаара, які є простими формами (крапки, лінії, прямокутники), для яких обчислюється різниця суми пікселів у чорних та білих областях. Оскільки

особливості можуть бути інформативними для виявлення облич (наприклад, тінь під носом може створювати характерний прямокутник на обличчі), вони використовуються для класифікації частин зображення. Нижче представлена формула, яка використовується для розрахунків значень ознак Хаара (2.2):

$$F = U - V \tag{2.2}$$

У цьому контексті U представляє загальну кількість значень яскравості точок, охоплених освітленою ділянкою об’єкта, тоді як V відповідає сукупності значень яскравості точок, охоплених затіненою ділянкою об’єкта.

Процедуру сканування зображення за допомогою функцій можна розділити на декілька етапів. По-перше, вхідні дані містять досліджуване зображення, вибране вікно сканування та вибрані функції. По-друге, вибране вікно сканування починає свій послідовний рух по зображенню, при цьому кожен крок охоплює один елемент вікна. Якщо припустити, що розмір вікна становить 24 * 24 елементи, цей рух відбувається по всьому зображенню. У процесі сканування в кожному вікні обчислюється понад 200 000 різних варіантів розташування об’єктів. Ця мінливість виникає внаслідок розташування об’єктів у вікні сканування та змін у їхньому масштабі. Сканування повторюється кілька разів, щоразу змінюючи масштаб, щоб забезпечити повне охоплення всіх можливих сегментів зображення. Примітно, що масштабування застосовується до самого вікна сканування, а не до зображення. Нарешті, усі ідентифіковані ознаки заносяться до класифікатора, де приймається остаточне рішення щодо відповідності зображення. Класифікатор визначає, чи відповідає зображення бажаним критеріям чи ні.

Процес застосування функцій Хаара в різних масштабах до зображення потребує значних обчислювальних ресурсів. Крім того, вкрай важливо розглянути модель машинного навчання, яка може ефективно вирішувати проблеми класифікації. У цьому сценарії алгоритм повинен ідентифікувати конкретні шаблони, які визначають, чи є обличчя на зображенні чи ні. Класифікатори можна класифікувати як слабкі та сильні. Слабкі класифікатори, як правило, допускають

численні помилки, розрізняючи різні класи, тоді як сильні класифікатори виявляють вищий ступінь точності при віднесенні об’єктів до відповідних класів. Підвищення, техніка, яка використовується для побудови композиції класифікаторів, має на меті компенсувати недоліки всього ансамблю класифікаторів шляхом введення наступних класифікаторів. Ці наступні класифікатори, за винятком першого, вчаться на помилках своїх попередників. Основна мета підвищення – об’єднати слабкі класифікатори в один сильний класифікатор.

Під час тренування метод Віоли-Джонса використовує розширену версію відомого своєю ефективністю алгоритму AdaBoost (Adaptive Boosting). Для того щоб класифікатор міг визначити, чи містить вікно, що розглядається, шуканий шаблон, необхідно виконати умову сильного класифікатора (2.3):

$$H(x) = \begin{cases} 1, \text{якщо } \sum_{t=1}^{T} a^{(t)} h_j^{(t)} \geq \frac{1}{2} \sum_{t=1}^{T} a^{(t)} \\ 0, \text{в інших випадках} \end{cases} \quad (2.3)$$

В даний час існує кілька методик оптимізації методу Віоли-Джонса. Одна з таких оптимізацій передбачає використання модифікованого вікна сканування, де крок зсуву можна регулювати незалежно для осей X і Y.

AdaBoost пропонує кілька переваг. По-перше, він може похвалитися простотою впровадження та працює з високою швидкістю, що робить його зручним вибором для практичного застосування. AdaBoost демонструє видатну здатність до узагальнення, що дозволяє йому ефективно адаптуватися до складних елементів у навчальній вибірці. Ця адаптивність забезпечує надійну продуктивність навіть за наявності проблемних точок даних. Крім того, AdaBoost має здатність ідентифікувати зашумлені об’єкти.

Варто зазначити, що навчання AdaBoost є складним завданням через потребу великої кількості вихідних даних і значні витрати часу на сам процес.

2.2 Алгоритм головних компонент

Алгоритм головних компонентів (PCA) – техніка, яка використовується для розпізнавання обличчя. Ключовим поняттям, введеним PCA в цьому контексті, є поняття «власних граней», які представляють собою зображення обличчя, що складаються з набору основних компонентів зображення. Розпізнавання обличчя передбачає порівняння основних компонентів невідомого зображення з компонентами відомих зображень.

Застосування PCA для розпізнавання облич пропонує кілька переваг, зокрема щодо зберігання та отримання зображень у великих базах даних, а також реконструкції зображень. Метод головних компонентів використовується для стиснення інформації, зберігаючи її основний зміст. Алгоритм передбачає лінійне ортогональне перетворення, де вхідний вектор x із розмірністю N перетворюється на вихідний вектор y з розмірністю M. В цьому випадку N більше за M. Однією з ключових переваг цього методу є те, що компоненти вектора y є некорельованими, що призводить до збереження загальної дисперсії після перетворення. Крім того, матриця X представляє компіляцію всіх прикладів зображень, знайдених у навчальному наборі.

Для отримання матриці власних векторів Ф використовується рівняння (2.4):

$$\Lambda = \Phi^{\mathrm{T}} \sum \Phi \tag{2.4}$$

де Λ – діагональна матриця власних чисел, $\sum \Phi$ – коваріаційна матриця.

Вибираючи матрицю $\Phi_{\text{м}}$ з множини Ф, яка відповідає найбільшим власним значенням M, можна помітити, що перетворення (2.5):

$$y = \psi_M^T X' \tag{2.5}$$

Вектор $X' = X - X'$, який є нормалізованим вектором із математичним очікуванням, що дорівнює нулю, відіграє вирішальну роль у характеристиці

більшості загальної дисперсії та охопленні найбільш значущих варіацій у змінній x. Коли вибираються перші M головні компоненти, це ділить векторний простір на дві окремі частини. Перша частина називається головним (власним) простором, позначеним як F, який складається з ϕi векторів (де i коливається від 1 до M), що представляють головні компоненти. Друга частина є ортогональним доповненням $F = \{\phi_i\}_{i=M+1}^{M}$.

У алгоритмі головних компонент існують конкретні показники, які використовуються для визначення приналежності. Ці показники включають DIFS (відстань у просторі ознак) і DFFS (відстань від простору ознак). DIFS вимірює відстань між аналізованим зображенням і еталонним зображенням у відповідних просторах ознак. З іншого боку, DFFS обчислює відстань між представленням аналізованого зображення в просторі спостереження та проекцією стандарту у власному просторі. Коли мова йде про розпізнавання людини за зображенням обличчя, процес передбачає роботу з вхідними векторами, які є зображеннями облич. Ці грані відцентровано та змінено в єдиному масштабі. Власні вектори, відомі як власні грані, обчислюються для даного набору зображень облич. Метод головних компонентів, застосований до цих зображень обличчя, зазвичай називають методом власних граней. Використовуючи попередньо розраховані матриці, вхідне зображення можна розкласти на набір лінійних коефіцієнтів, відомих як головні компоненти. Ці компоненти відіграють вирішальну роль у наближенні оригінального зображення. Підсумовування перших N головних компонентів, помножених на їхні відповідні власні вектори, дає наближення зображення порядку N.

У контексті розпізнавання обличчя розраховуються основні компоненти кожного зображення обличчя. Зазвичай вибирається діапазон від 5 до 200 основних компонентів, а решта компонентів кодують тонкі варіації між обличчями та шумом. Процес розпізнавання включає порівняння основних компонентів невідомого зображення з компонентами інших зображень. Для полегшення цього порівняння зазвичай використовується метрика, найпростішою з яких є евклідова відстань.

Крім того, для підвищення надійності аналіз головних компонентів додатково застосовується до певних областей обличчя, таких як очі, ніс і рот.

Алгоритм головних компонентів зазвичай використовується для виявлення облич на зображеннях. У випадку облич компоненти в їх конкретному просторі, як правило, мають значні значення, а в інших просторах вони ближчі до нуля. Цю властивість можна використати, щоб визначити, чи містить вхідне зображення обличчя. У результаті перевіряється значення помилки реконструкції, причому більша помилка свідчить про меншу ймовірність присутності обличчя. Крім того, варіації зображень обличчя, як-от раса, стать, емоції та освітлення, призведуть до помітних варіацій у компонентах, причому на ці варіації в першу чергу впливають ці фактори. Отже, значення відповідних головних компонентів можна використовувати для визначення певних характеристик, таких як раса або стать людини. Однак важливо відзначити недоліки методу аналізу головних компонентів PCA вимагає певних умов, включаючи однакові параметри освітлення, нейтральний вираз обличчя та відсутність перешкод, таких як окуляри чи борода, щоб отримати точні результати.

Невиконання цих умов може призвести до того, що основні компоненти не зможуть вловити міжкласові варіації [33]. Наприклад, за різних умов освітлення метод власних граней практично незастосовний через те, що перші головні компоненти значною мірою відображають зміни освітленості, що призводить до порівняння зображення з подібними рівнями освітлення.

Розглянемо модифікацію алгоритму PCA під назвою Eigenfaces. Зображення у градаціях сірого з розмірами p x q може бути представлено у векторному просторі з розмірами $p*q$. Наприклад, якщо взяти зображення розміром 100 x 100 пікселів, воно належатиме до 10000-вимірного простору. Виконання обчислень на зображеннях із такими просторами великої розмірності потребує значних обчислювальних ресурсів. Однак не всі вимірювання в цих просторах доречні або корисні для подальшого аналізу. Для вирішення цієї проблеми можна застосувати метод головних компонент. Цей метод спрямований на перетворення набору змінних, які пов’язані одна з одною, у менший набір непов’язаних змінних.

Основна ідея полягає в тому, що багатовимірні набори даних часто демонструють асоціації між змінними, тобто лише кілька вимірів містять найбільш цінні та інформативні дані.

Нижче представлено деталізований опис роботи Eigenfaces [18]:

1. Формування початкового набору даних. Для прикладу, на рисунку 2.4 представлені зображення з обличчями у відтінках сірого (один канал).

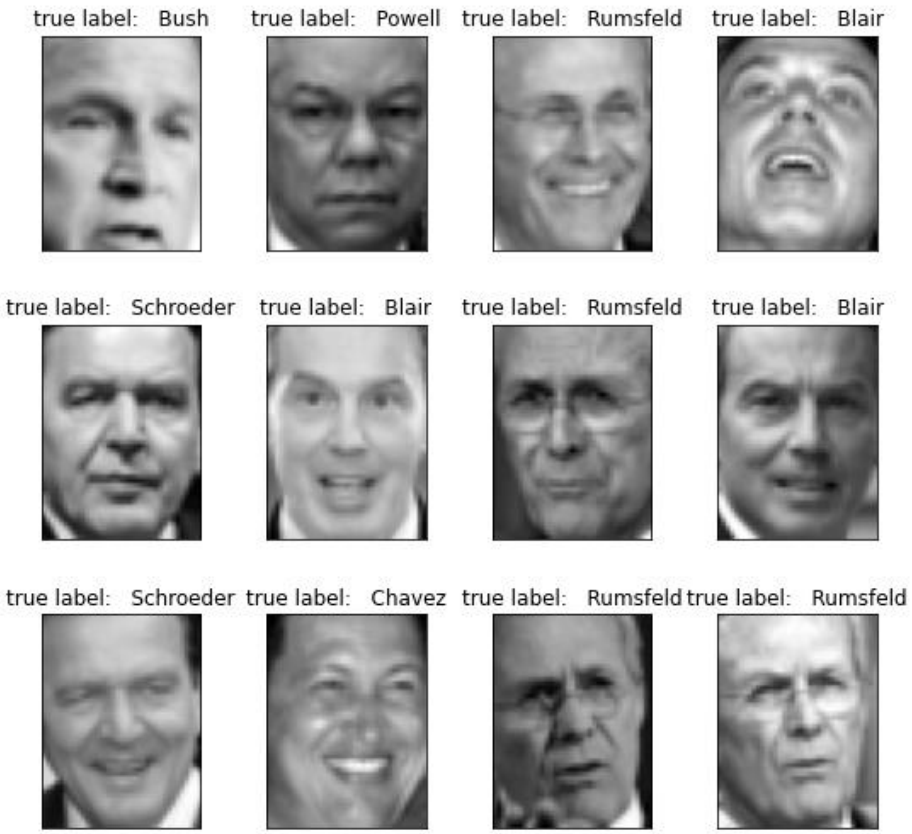


Рисунок 2.4 – Приклад початкових зображень з набору даних

2. Процес перетворення зображень обличчя на вектор обличчя передбачає перетворення зображень у відтінках сірого, які представлені у вигляді матриць пікселів, у векторну форму. У цих матрицях значення пікселів коливаються від 0 до 255, де 0 означає чорний колір, а 255 – білий. Перетворюючи матрицю у вектор, отримуємо вектори $x_1, x_2 \dots, x_m$ з розмірністю $N^2 * 1$ кожен. Результуючий вектор після перетворення буде мати вигляд, представлений на рисунку 2.5. Цей метод

перетворення забезпечує більш стисле та структуроване представлення зображень обличчя.

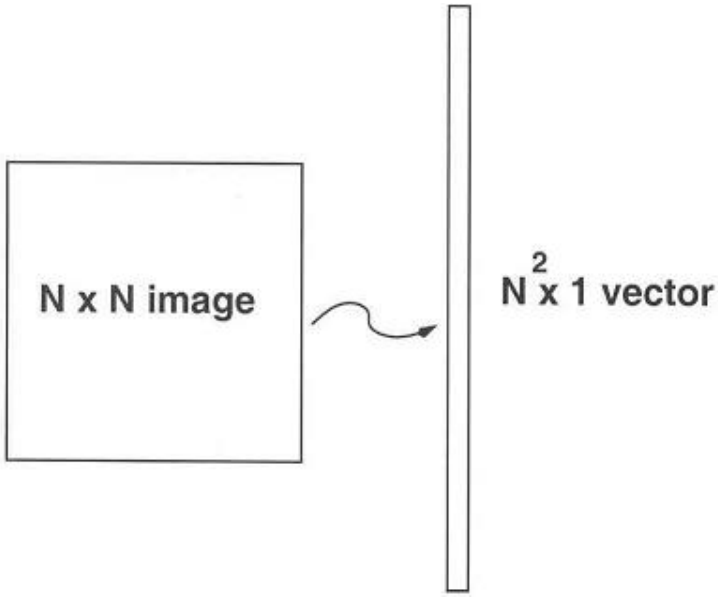


Рисунок 2.5 – Процес перетворення зображення у вектор

3. Нормалізація вектора обличчя відбувається в два етапи. На початковому етапі (2.6) знаходиться «усереднене обличчя» (рисунок 2.6).

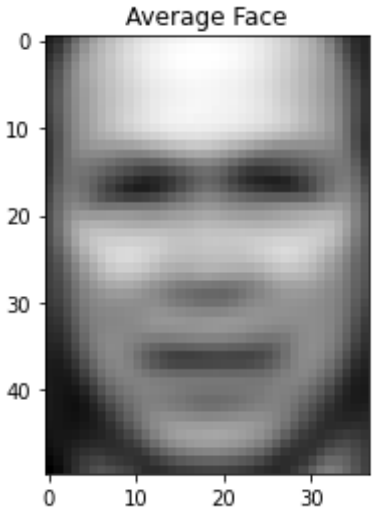


Рисунок 2.6 – Приклад знаходження «усередненого обличчя»

$$\psi = \frac{1}{m}\sum_{i=1}^{m} x_i \quad (2.6)$$

де m – кількість зображень в наборі даних, x_i – вектори обличчя.

Обличчя, отримане шляхом усереднення всіх облич у наборі даних, представляє спільні риси, спільні для всіх облич. Далі, усереднене обличчя віднімається від кожного окремого вектора обличчя (2.7):

$$a_i = x_i - \psi \quad (2.7)$$

В результаті , формується новий вектор A (розмірність $N^2 * M$) (2.8):

$$A = [a_1, a_2, .., a_m], \quad (2.8)$$

4. Для отримання коваріаційної матриці, необхідно помножити матрицю A на її транспоновану матрицю, позначену як A^T. Розмірність матриці A задається як $N^2 * M$, тоді як розмірність $A^T = M * N^2$. При множенні A помножити на A^T, результуюча матриця матиме розмірність $N^2 * N^2$, що дасть N^2 власних векторів розмірності N^2. Однак важливо зазначити, що зі збільшенням значення N (кількості елементів) виконання обчислень на матрицях таких великих розмірів вимагатиме значних обчислювальних ресурсів. Отже, коваріаційна матриця зазвичай обчислюється як добуток A^Tі A, що формує матрицю з розмірністю $M * M$ (прийнято, що $M \ll N^2$).

5. Для знаходження власних значень (eigenvalues) та власних векторів (eigenvectors) для коваріаційної матриці використовується наступна формула (2.9):

$$C'u_i = \lambda_i u_i, i = 1, 2, \dots, m, C' = AA^T \quad (2.9)$$

6. У процесі сортування власних векторів, вони впорядковуються у порядку спадання на основі їх пов’язаних власних значень. Це дозволяє ідентифікувати K

найважливіших власних векторів, також відомих як власні грані (рис. 2.7), які відповідають K найбільшим власним значенням.

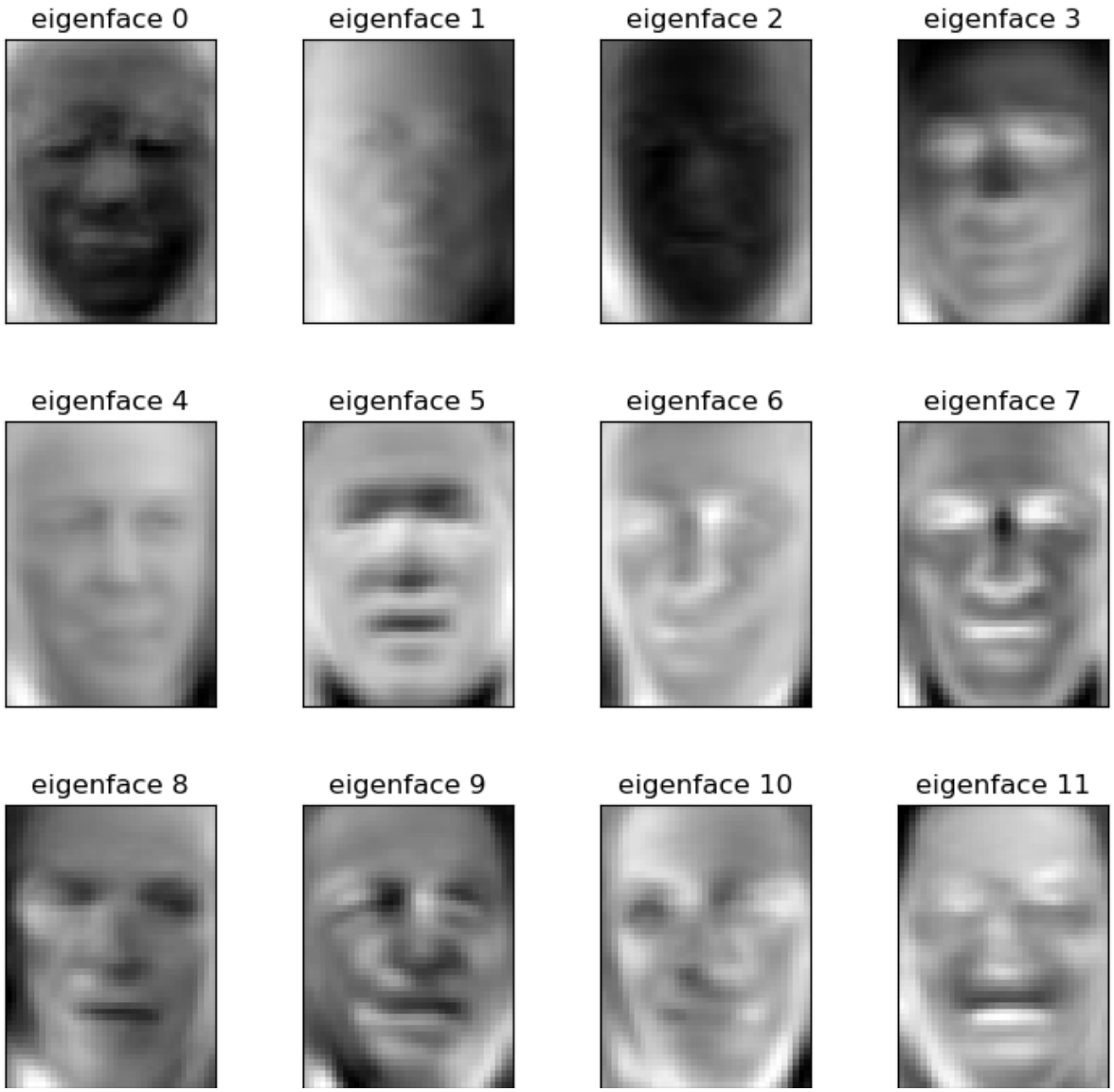


Рисунок 2.7 – Набір зображень власних осіб (eigenfaces), що використовуються в алгоритмі головних компонентів

7. Далі, відбувається перетворення K власних векторів меншої розмірності в початкову розмірність з використанням наступної формули (2.10):

$$u_i = Av_i \tag{2.10}$$

8. Щоб представити кожну грань у наборі даних, можна виразити її як комбінацію K власних векторів. Це означає, що, взявши зважені суми K власних граней і додавши усереднену грань ψ, можна представити будь-яку грань (лице) із набору даних. Нижче представлена формула для розрахунку (2.11):

$$x_i = \sum_{j=1}^{K} w_j u_j + \psi, \tag{2.11}$$

де w_j – ваги, u_j – власні обличчя (представлені на рисунку 2.7).

Результат комбінування власних векторів представлено на рисунку 2.8.

Рисунок 2.8 – Обличчя для тренування, яке представлено як комбінація власних векторів

Після завершення етапу навчання, розглянемо етап ідентифікації незнайомих облич. Алгоритм, який використовується для розпізнавання обличчя за допомогою Eigenfaces, функціонує таким чином:

1. Після отримання зображення обличчя, необхідно виконати операції попередньої обробки (вирівнювання обличчя та коригування розмірів).

2. Зображення обличчя перетворюється на вектор обличчя, який складається з числових елементів, що представляють різні риси обличчя.

3. Потім вектор обличчя нормалізується, подібно до процедури, описаної на кроці 3 алгоритму. Ця нормалізація забезпечує послідовність і однаковість у представленні облич на різних зображеннях.

4. Нормалізований вектор граней проектується на власний простір, що призводить до лінійної комбінації власних граней і зваженого вектора:

$$\Omega = \begin{bmatrix} w_1 \\ w_2 \\ \dots \\ w_k \end{bmatrix}$$

5. У контексті розпізнавання обличчя процес передбачає порівняння векторів відомих облич із набору даних із вектором невпізнаного обличчя.

Коли відмінність між цими векторами потрапляє в межі визначеного порогу помилки, це означає, що обличчя було успішно розпізнано. Однак, якщо несхожість перевищує наявну похибку, обличчя класифікується як невідоме.

2.3 Алгоритм локальних бінарних шаблонів

Локальний бінарний шаблон (LBP) – це оператор, який використовується для цілей класифікації в області комп’ютерного зору. Спочатку він був представлений у 1996 році для аналізу текстури напівтонових зображень. Подальші дослідження [21, 22] показали, що LBP здатний зберігати свою ефективність навіть при незначних варіаціях умов освітлення та незначних поворотах зображення. ЛБШ служить двійковим представленням для опису оточення пікселя в зображенні. Якщо значення інтенсивності пікселя дорівнює або перевищує значення інтенсивності центрального пікселя, йому присвоюється значення «1»; інакше йому присвоюється значення 0. У результаті базовий оператор ЛБШ генерує 8-бітний двійковий код, що пропонує детальний опис оточення пікселя.

Фундаментальний оператор ЛБШ враховує значення інтенсивності центрального пікселя, використовуючи вісім сусідніх пікселів (рисунок 2.9).

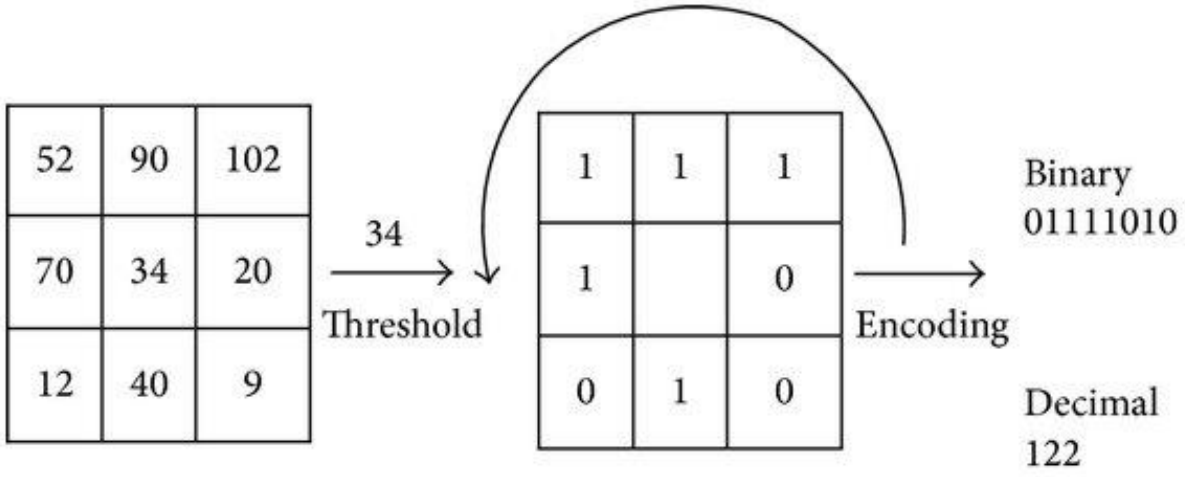


Рисунок 2.9 – Схема базового оператора ЛБШ

Використання кругової околиці та використання білінійної інтерполяції інтенсивності пікселів дає можливість побудувати локальний бінарний шаблон, що охоплює гнучку кількість точок, позначених як P, і заданий радіус, який позначається як R. У цих двійкових кодах певні шаблони містять більше інформаційний контент, ніж інші. У цьому відношенні локальний двійковий шаблон вважається рівномірним, якщо він містить не більше трьох послідовних рядів, що складаються з нулів і одиниць, що чергуються. Наприклад, приклади уніфікованих локальних двійкових шаблонів включають 00000000, 001110000 і 11100001. Уніфіковані локальні двійкові шаблони служать двом основним цілям. По-перше, вони зосереджені виключно на захопленні значущих локальних особливостей зображення, таких як кінці ліній, грані, кути та плями, як представлена на рисунку 2.10. Цей підхід гарантує, що враховуються лише найважливіші деталі.

По-друге, застосування єдиних локальних двійкових шаблонів призводить до значної економії пам’яті – ($P(P-1)+2$ шаблонів замість $2P$).

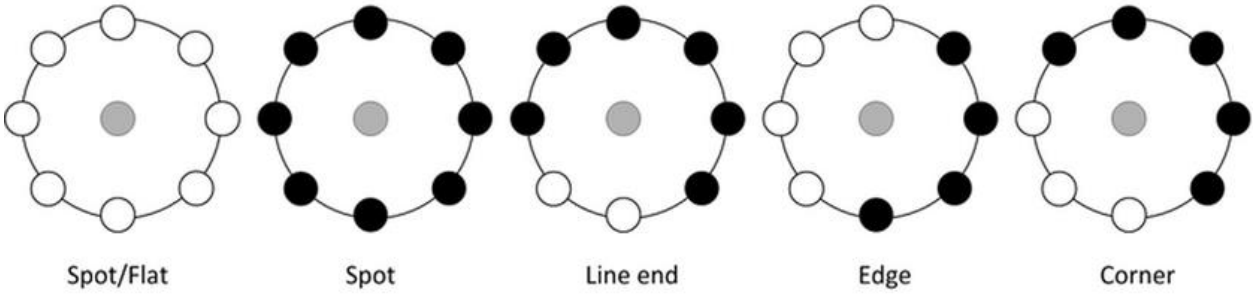


Рисунок 2.10 – Оператори, що відповідають графічним примітивам (зліва направо – п’ятно/фон, п’ятно, кінець лінії, грань, кут) [20]

Використовуючи оператор ЛБШ для кожного окремого пікселя зображення, в результаті буде сформована гістограма.

Ця гістограма складається з окремих стовпців, у яких кожен стовпець представляє унікальний уніфікований код ЛБШ. Крім того, є додатковий стовпець, спеціально призначений для отримання інформації про будь-які неправильні шаблони, які можуть бути присутніми на зображенні. Нижче представлено опис роботи алгоритму локальних бінарних шаблонів [22]:

Формально виразити опис оператора ЛБШ можна наступним чином (2.12):

$$LBP(x_c y_c) = \sum_{p=0}^{p-1} 2^P S(i_p - i_c) \tag{2.12}$$

де $i_p - i_c$ – інтенсивність пікселя, S – функція, виражена рівністю (2.13):

$$s(x) = \begin{cases} 1, \text{якщо } x \geq 0 \\ 0 \text{ інакше} \end{cases} \tag{2.13}$$

Розглянемо оригінальне зображення в градаціях сірого, яке можна представити у вигляді матриці $n * m$, що містить значення інтенсивності кожного пікселя. Далі, представляємо «рухоме вікно» розміром 3 х 3. У цьому процесі порогове значення встановлюється як значення інтенсивності центрального пікселя. Рухаючись далі, порівнюємо значення інтенсивності кожного пікселя,

прилеглого до центрального, з пороговим значенням. Якщо значення пікселя перевищує порогове значення, присвоюємо 1 відповідній комірці; інакше присвоюємо 0. На рисунку 2.11 порогове значення вказано, як 82. Наприклад, якщо розглядати на першій клітинці в першому рядку зі значенням 137, видно, що воно перевищує порогове значення 82. Отже, призначаємо 1 цій комірці. Просуваючись вперед, повторюємо той самий процес для решти сусідніх пікселів. Далі, настає етап, на якому об'єднуємо всі отримані двійкові значення в нове двійкове число.

На рисунку 2.11 отримане двійкове число представлено як 11100001. Отримавши це двійкове число, переводимо його в десяткову систему числення. Нарешті, замінюємо початкове значення інтенсивності центрального пікселя в матриці на щойно обчислене десяткове значення. Переміщуємо вікно в наступне положення і повторюємо усі вищеописані дії.

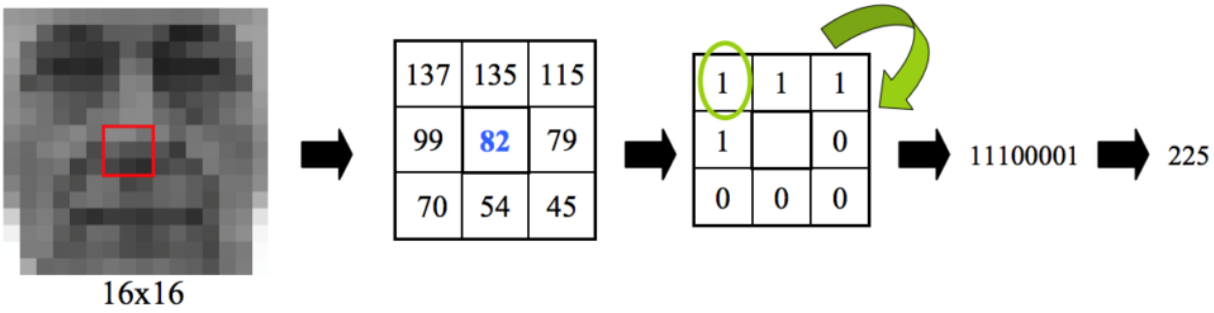


Рисунок 2.11 – Візуалізація знаходження локальних бінарнів шаблонів

Після проходження серії перетворень отримане зображення служить основою для побудови гістограми. Після завершення процесу навчання алгоритму генерується гістограма для кожного зображення в наборі даних.

На етапі тестування вхідне зображення проходить ту саму процедуру, що й навчальні дані. Згодом гістограма невідомого зображення порівнюється з гістограмами навчальних зображень. Ті гістограми, які виявляють найбільшу схожість, визначаються як належні одній людині. Для знаходження Евклідової відстані між гістограма використовується наступна формула (2.14):

$$D = \sqrt{\sum_{i=1}^{n}(hist1_i - hist2_i)^2} \quad (2.14)$$

Вимірювання відстані між гістограмами також служить індикатором рівня «довіри» до результатів. Коли відстань між гістограмами менша, це означає нижче значення «впевненості». Крім того, це значення можна використовувати для оцінки точності отриманого результату. Для успішного розпізнавання, значення «впевненості» має бути нижчим за вказаний поріг.

2.4 Алгоритми глибокого навчання

Глибоке навчання стало потужною технікою класифікації зображень, особливо в області виявлення та розпізнавання облич. У порівнянні з традиційними підходами до комп'ютерного бачення, глибоке навчання, зокрема за допомогою застосування згорткових нейронних мереж (CNN) або ConvNet, незмінно демонструє вищу продуктивність. Примітно, що будь-яка нейронна мережа, яка включає принаймні один згортковий шар, може бути класифікована як CNN, що робить її невід'ємним компонентом цієї передової методології. На додаток до своїх можливостей у задачах розпізнавання обличчя, CNN знаходять широке застосування в різних системах комп'ютерного зору, включно з тими, які використовуються в робототехніці та автономних транспортних засобах, таких як безпілотні автомобілі.

Локальність є фундаментальним поняттям, яке має велике значення з людської точки зору. Коли ми дивимося на навколишній світ, наші очі сприймають об'єкти в невеликих, локалізованих областях нашого зору та ретельно аналізують їхні чіткі особливості, такі як кути, краї та текстури. Цю концепцію також можна спостерігати в області згорткових нейронних мереж (ЗНМ).

Ці шари використовують фільтри, які проходять через вхідне зображення, вилучаючи локальні особливості, такі як межі та форми.

Мережа LeNet-5, запропонована у 1989 році [24], була однією з перших згорткових нейронних мереж (CNN). Спочатку вона отримала визнання за свою здатність точно розпізнавати поштові індекси та номери. Відтоді з'явилося кілька новіших архітектур CNN, які перевершують можливості LeNet. Однак фундаментальні принципи, що лежать в основі цих мереж, залишилися в основному незмінними з моменту заснування першої CNN. На рисунку 2.12 представлена архітектури згорткової нейронної мережі, схожої на LeNet. Ця мережа призначена для визначення категорії, до якої належить вхідне зображення. Архітектура складається з чотирьох ключових операцій: згортки (Convolution), застосування функції ReLU, об'єднання та класифікації.

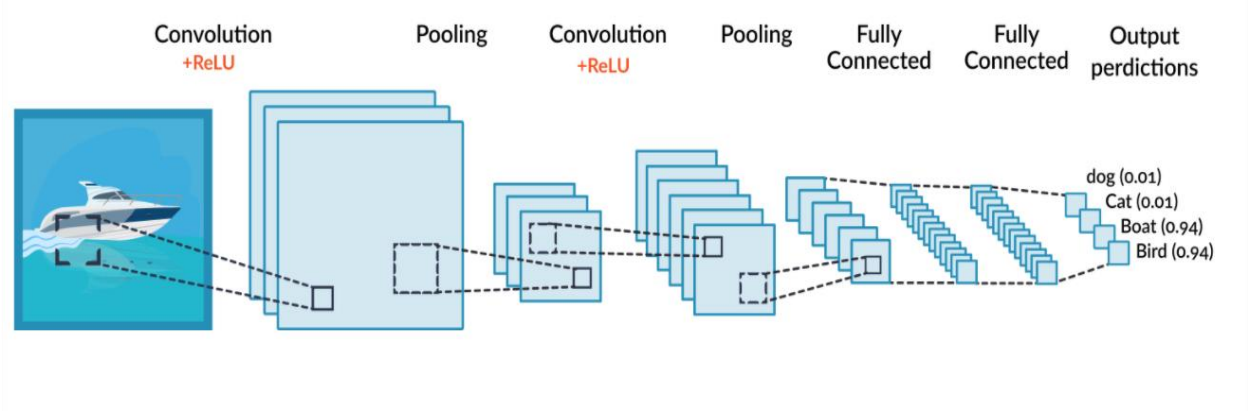


Рисунок 2.12 – Архітектура згорткової мережі LeNet-5

Згортка застосовується до вхідних даних та відіграє вирішальну роль у виділенні ознак із зображення. Використовуючи фільтри, основна мета згортки полягає в тому, щоб вивчити та ідентифікувати відмінні атрибути у вхідному зображенні. Слід зазначити, що згортка ефективно зберігає просторову кореляцію між пікселями під час цього процесу. Для кращого розуміння розглянемо зображення в градаціях сірого, яке може бути адекватно представлене однією матрицею. У більшості випадків цього достатньо для таких завдань, як розпізнавання обличчя. Тепер розглянемо матрицю зображення розміром 5 x 5 і фільтр (або детектор функцій) розміром 3 x 3. Коли цей фільтр застосовано, у

матриці вибирається певна квадратна область 3 x 3. Шляхом множення відповідних елементів матриці та детектора ознак добутки підсумовуються та записуються в результуючу матрицю 3 x 3, відому як карта ознак. Приклад карти ознак та фільтра представлений на рисунку 2.13.

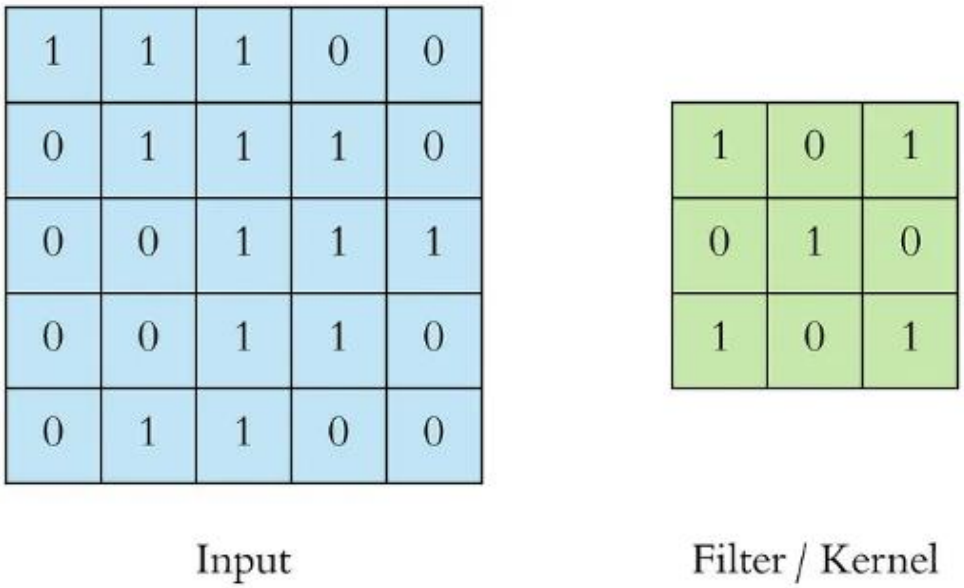


Рисунок 2.13 – Матриця зображення (5 x 5) та згортковий фільтр (3 x 3)

У процесі аналізу матриці квадрати всередині матриці відбираються, починаючи з самого першого елемента зліва. Цей процес відбору можна представити як застосування фільтра до матриці. Після вибору кожного квадрата обчислюється добуток його елементів і додається до загальної суми. Згодом детектор ознак пересувається праворуч на одну клітинку, дозволяючи розглянути наступний квадрат. Якщо оброблено весь рядок, детектор ознак повертається до початкової позиції та переміщується на одну клітинку вниз, забезпечуючи безпроблемне продовження операцій множення та додавання.

На рисунку 2.14 представлений результат застосування фільтра.

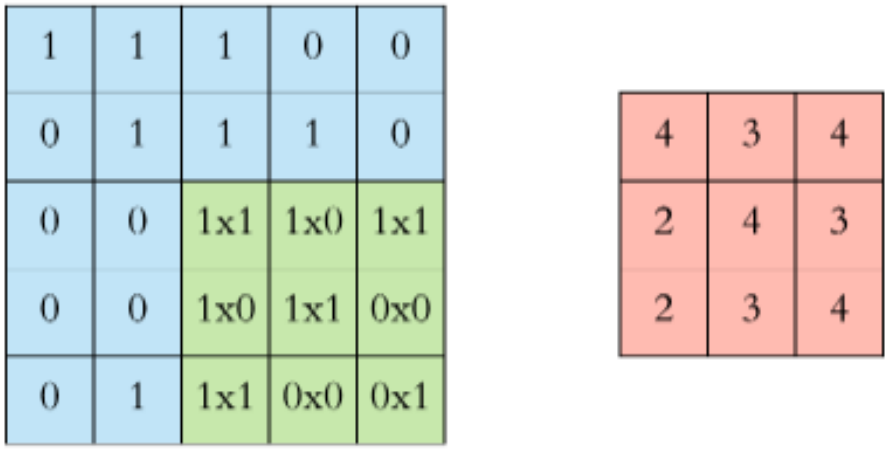


Рисунок 2.14 – Фінальна ітерація застосування конволюційного фільтра

У цьому прикладі розглянуто операцію двовимірної згортки, яка виконується за допомогою фільтра 3x3. Однак у практичних застосуваннях згортки виконуються в 3D. Важливо взяти до уваги, що насправді зображення представляється як тривимірна матриця, яка включає розміри висоти, ширини та глибини. У контексті представлення зображення компонент глибини відповідає різним кольоровим каналам, таким як червоний, зелений і синій (RGB). Отже, згортковий фільтр має певні параметри висоти та ширини, наприклад 3x3 або 5x5, і має на меті охопити всю глибину свого входу.

Застосування фільтрів із різними значеннями до зображення призводе до різноманітних карт функцій. Ці фільтри служать для виконання таких операцій із зображенням, як виявлення країв, підвищення різкості, розмиття тощо. На рисунку 2.15 представлено застосування різноманітних детекторів ознак до одного зображення. У сфері згорткових нейронних мереж значення фільтрів вивчаються самостійно під час процесу навчання, незважаючи на вимогу ручного налаштування певних параметрів. Ці параметри можуть включати кількість фільтрів, розміри фільтрів, архітектуру мережі та інші. Збільшуючи кількість фільтрів, згорточна мережа здатна виявляти більший набір ознак, що дозволяє досягати кращих результатів у задачі розпізнавання образів.

Operation	Kernel	Image result
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	

Рисунок 2.15 – Застосування різноманітних фільтрів до зображення [14]

На розміри карти ознак впливають три ключові параметри, які необхідно визначити перед виконанням процесу згортки. По-перше, параметр глибини стосується кількості фільтрів, які використовуються під час згортки. Важливо зауважити, що кількість карт функцій відповідає кількості використовуваних фільтрів. По-друге, параметр кроку визначає кількість пікселів, на яку переміщується фільтр по вихідному зображенню. Чим більший крок, тим меншою буде результуюча карта функцій. Нарешті, концепція заповнення нулями виникає в певних сценаріях, де виявляється вигідним заповнити краю матриці вхідного зображення нулями. Це полегшує застосування фільтрів до пікселів, розташованих на межах зображення. Після виконання операції згортання застосовується функція

ReLU, також відома як функція випрямленої лінійної одиниці або функція випрямлення (як продемонстровано на рис. 2.16).

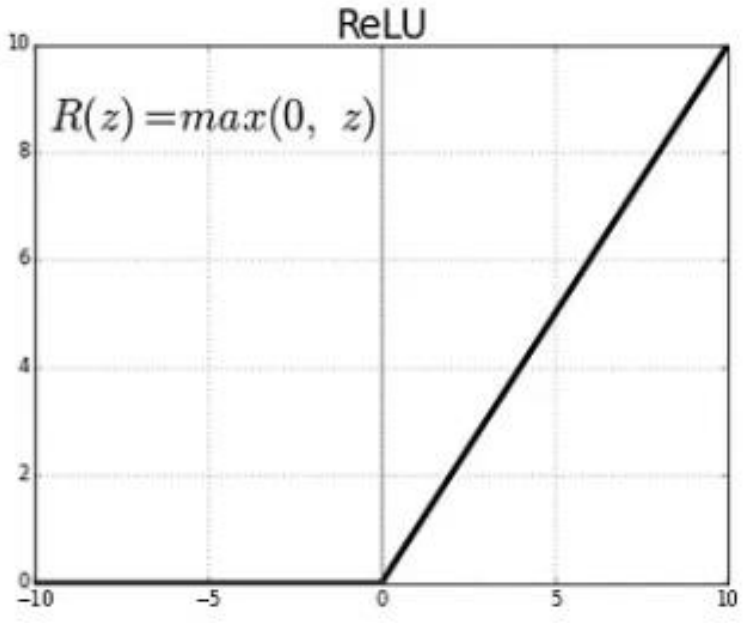


Рисунок 2.16 – Представлення передавальної функції ReLU

Ця функція є функцією активації та визначається наступним чином (2.15):

$$f(x) = \max(0, x) \tag{2.15}$$

Функція ReLU використовується на поелементній основі, особливо коли йдеться про зображення, де вона працює на рівні пікселів. Отже, будь-які негативні значення пікселів на карті функцій замінюються нулями. Метою використання ReLU у згортковій нейронній мережі є впровадження нелінійності.

Це важливо, оскільки більшість реальних даних, які використовуються для навчання нейронної мережі, мають нелінійні характеристики. На етапі згортки, який включає поелементне множення та додавання матриць, виконуються лише лінійні операції. Однак для врахування нелінійності включена функція ReLU.

Варто зазначити, що ReLU можна замінити альтернативними нелінійними функціями, як-от сигмоїда або гіперболічний тангенс, але в більшості випадків було доведено, що ReLU дає кращі результати [25].

Після виконання операції згортки зазвичай застосовують об’єднання (pooling), щоб зменшити розмірність вхідних даних. Це служить для зменшення загальної кількості параметрів, що призводить до скорочення часу навчання та пом’якшення проблеми переобладнання. Об’єднані шари виконують зменшення дискретизації незалежно на кожній карті об’єктів, зменшуючи висоту та ширину, зберігаючи глибину незмінною. Серед різних типів методів об’єднання широко використовується максимальне об’єднання, яке передбачає вибір максимального значення в межах визначеного вікна об’єднання. На відміну від операції згортки, об'єднання не має параметрів; швидше, він проходить вікном по вхідних даних і витягує максимальне значення в цьому вікні. Подібно до згортки, розмір вікна та крок вказуються.

Наприклад, на рисунку 2.17 використовується вікно розміром 2 x 2 і вибирається максимальне значення серед тих, що знаходяться у вікні.

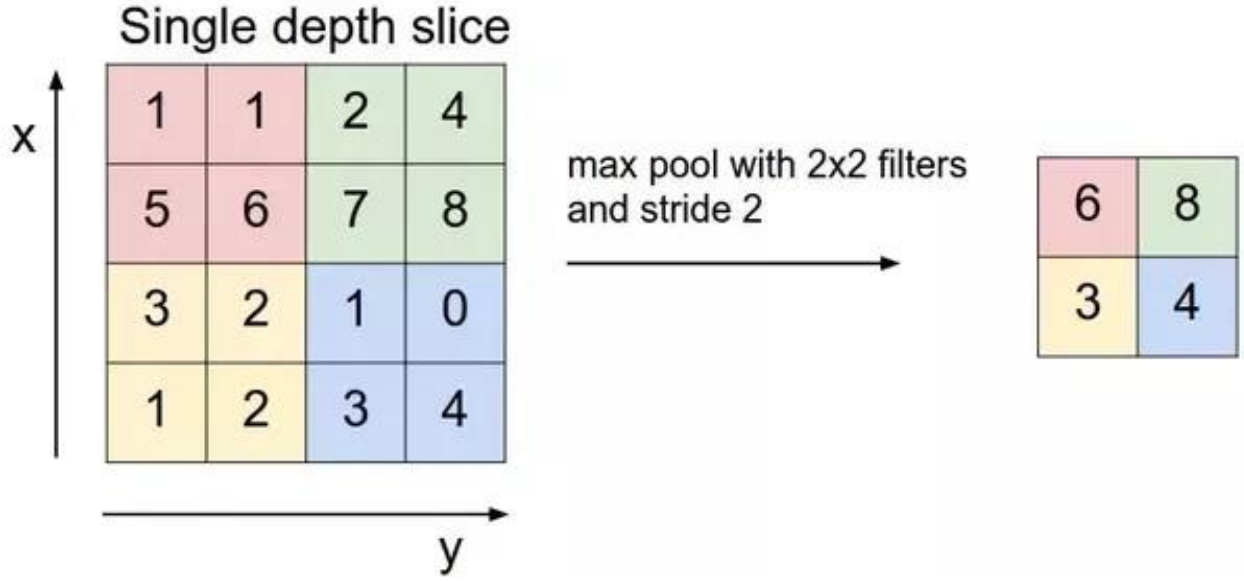


Рисунок 2.17 – Застосування операції пулингу максимального значення

У контексті застосування операції об’єднання, вікно плавно зміщується як горизонтально (вздовж осі x), так і вертикально (вздовж осі y) з невеликими

кроками. Цей рух можна порівняти з поступовим рухом під час прокручування. На рисунку 2.17 кожен крок уздовж осей x і ординат відповідає зміщенню двох комірок. Ще одна схожість з операцією згортки полягає в зменшенні розмірності карти ознак. На рисунку 2.17 матриця до об’єднання мала розміри 4 x 4. Однак після застосування операції об’єднання розміри було зменшено до 2 x 2. Важливо зазначити, що об’єднання виконується окремо для кожної окремої карти об’єктів. Ця операція має важливе значення в згорткових мережах, оскільки служить кільком цілям. Однією з основних цілей є ефективне зменшення кількості параметрів і обчислень, необхідних у мережі. Таким чином, це дає змогу краще контролювати можливість перетренування, сценарій, коли модель демонструє високу точність під час класифікації даних навчання, але є нижчою, коли вводить нові, невидимі дані. Крім того, об'єднання також сприяє здатності згорткової мережі залишатися відносно вільним від незначних спотворень і рухів, присутніх у вхідному зображенні. Завдяки об’єднанню (pooling) мережа досягає рівня нечутливості до цих незначних змін, що підвищує її надійність.

Мережа LeNet структурована наступним чином. Спочатку виконується ряд операцій згортки, ReLU та об’єднання. Потім ці операції повторюються з використанням результатів попереднього раунду. Кількість повторень залежить від архітектури згорткової нейронної мережі. У сучасних згорткових нейронних мережах може бути кілька рівнів згортки, ReLU та об’єднання, іноді сягаючи десятків шарів. Також можна послідовно виконувати операції згортки та ReLU перед застосуванням об’єднання. Це дозволяє мережі розпізнавати дедалі складніші функції. Наприклад, коли згортка вперше застосована до вхідного зображення, вона виявляє краї. Подальші операції згортання визначають прості форми, які потім використовуються для виявлення складніших функцій, таких як форма обличчя. Глибина та складність можливостей розпізнавання мережі зростає з кількістю виконуваних операцій згортки.

У сфері нейронних мереж операції згортання та об’єднання дають значні функції, які потім передаються на повністю зв’язаний рівень (fully-connected), також відомий як багатошаровий перцептрон. Цей тип нейронної мережі має три

основні рівні: вхідний, прихований і вихідний рівні, позбавлені будь-яких петель зворотного зв’язку. У повністю зв’язаному шарі кожен нейрон попереднього шару складно з’єднаний з кожним нейроном наступного шару. Використовуючи функцію активації Softmax, хоча й не обмежується лише нею, перцептрон визначає клас, до якого належить вхідне зображення. Це визначення залежить від застосування «знань», набутих під час фази навчання.

Для наочності розглянемо сучасну архітектуру CNN під назвою ResNet. ResNet, глибока нейронна мережа, представлена в 2015 році, заснована на пірамідальних нейронах, знайдених у корі головного мозку. Його революційним досягненням стала здатність досягати точності класифікації зображень, порівнянної з людською. До появи ResNet дослідники намагалися покращити можливості виявлення функцій згорткових нейронних мереж шляхом збільшення їх глибини. Однак вони зіткнулися з явищем, коли продуктивність мережі погіршувалася замість покращення зі збільшенням глибини, як спостерігалося в дослідженні. Два основні фактори сприяють цьому результату: по-перше, реалізація ідентичних відображень стає складною на наступних рівнях, а додавання зайвих рівнів може негативно вплинути на функціональність мережі та процес навчання. По-друге, із збільшенням глибини нейронної мережі виникають труднощі в обчисленні градієнтів для зворотного поширення помилок. Існування функцій активації призводить або до експоненціального збільшення, або до майже нульового зменшення значень градієнта всередині мережі, що ще більше ускладнює процес навчання. Для вирішення проблем, згаданих вище, ResNet використовує модулі Residual замість звичайних шарів.

Ці модулі являють собою складені шари, які вводять контури, полегшуючи встановлення ідентичного відображення та сприяючи поширенню градієнтів помилок у протилежному напрямку під час процесу навчання. Рисунок 2.18 ілюструє архітектуру модуля Residual і зображує приклад мережі з 50 шарами.

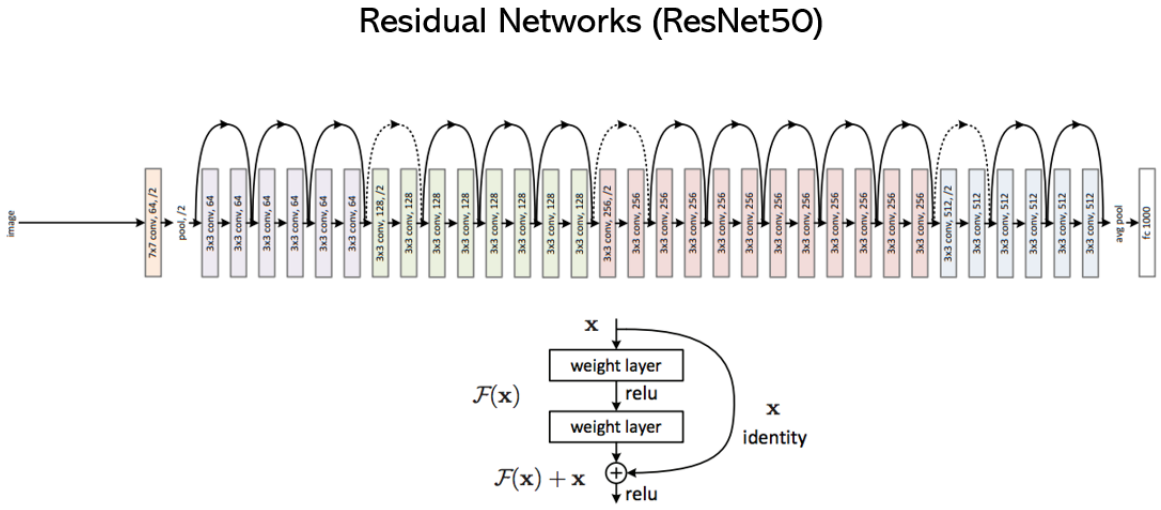


Рисунок 2.18 – Архітектура нейронної мережі ResNet50 (зверху – структура мережі з 50 шарами, знизу – послідовний Residual-модуль).

На відміну від звичайних мереж, нейронна мережа ResNet може досягати значної глибини, зазвичай перевищуючи 18 рівнів і потенційно досягаючи до 1024 рівнів. Із глибиною 152 шарів мережа досягає рівня помилок лише 4,49%.

РОЗДІЛ 3 ПРОЄКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ ВИЯВЛЕННЯ ТА ЗАПОБІГАННЯ ВТОМЛЕНОСТІ ВОДІЯ

3.1 Опис алгоритму виявлення втомленості

Для виявлення втомленості водія буде використана техніка визначення співвідношення сторін очей (англ. Eye Aspect Ratio). Використання функції EAR для виявлення моргання має кілька переваг перед традиційними методами обробки зображень. У звичайних підходах першим кроком є локалізація очей на зображенні, а потім порогове визначення для ідентифікації зіниць очей. Згодом миготіння виявляється шляхом моніторингу зникнення білої області. З іншого боку, метрика EAR усуває потребу в інтенсивній обробці зображень. Це призводить до зменшення використання пам’яті та часу обробки. Натомість функція EAR базується на обчисленні співвідношення відстаней між конкретними орієнтирами обличчя, пов’язаними з очима. Як показано на рисунку 3.1, ці орієнтири складаються з шести координат, позначених від p_1 до p_6, починаючи з кута лівого ока та обертаючи за годинниковою стрілкою. Всі шість координат є двовимірними.

Значення EAR залишається відносно постійним, коли очі відкриті. Отже, техніка EAR забезпечує просте та ефективне рішення для виявлення моргання.

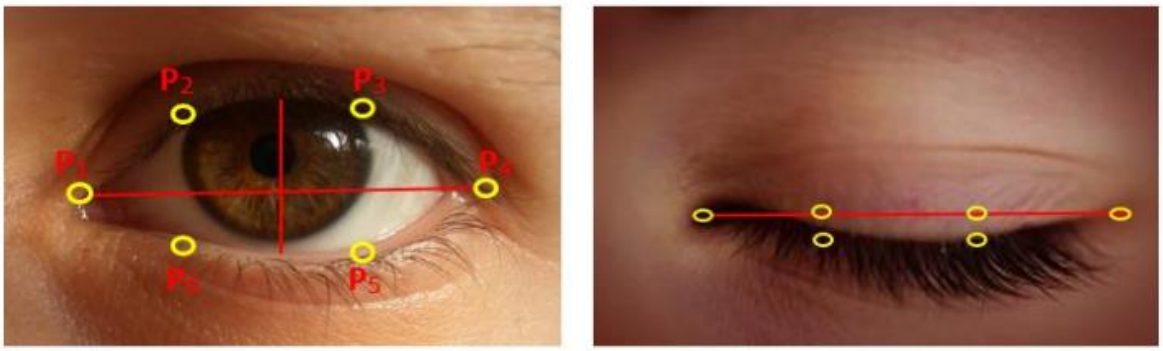


Рисунок 3.1 – Розташування шести орієнтирів ока (від P1 до P6) під час відкритого (ліворуч) або закритого (праворуч) станів

Значення EAR розраховується за наступною формулою (3.1), а середнє EAR розраховується за формулою (3.2):

$$EAR_R = \frac{\left||p_2^R - p_6^R|\right| + ||p_3^R - p_5^R||}{2*||p_1^R - p_4^R||},\ EAR_L = \frac{\left||p_2^L - p_6^L|\right| + ||p_3^L - p_5^L||}{2*||p_1^L - p_4^L||} \quad (3.1)$$

$$\text{Середнє EAR} = \frac{EAR_L + EAR_R}{2} \quad (3.2)$$

Посилаючись на рівняння вище, чисельник рівняння обчислює вимірювання відстані між верхньою повікою та нижньою повікою. Знаменник представляє горизонтальну відстань ока. Варто зазначити, що коли очі людини відкриті, значення чисельника збільшується, що, як наслідок, призводить до збільшення значення EAR і навпаки, коли очі закриті, значениня чисельника зменшується, що призводить до зменшення значення EAR. Щоб отримати більш точне уявлення, значення EAR для лівого та правого ока розраховується окремо, а потім усереднюється. У розробленій системі постійно буде контролюватися значення Eye Aspect Ratio, щоб визначити, чи опускається його значення нижче попередньо встановленого порогу. Крім того, дуже важливо переконатися, що значення знову не перевищить порогове значення в наступному кадрі. Якщо ці умови виконуються, це говорить про те, що людина заплющила очі і перебуває в сонному стані. Навпаки, якщо значення EAR знову підвищується, це означає, що немає причин щодо занепокоєння.

На рисунку 3.2 представлена блок-схема алгоритму виявлення сонливості, який буде використаний для розробки програмної системи.

На початковому етапі система ініціює процес, записуючи відео, яке починає фіксацію рухів голови водія, а потім витягує з нього кадри. Встановлюється змінна, що функціонує як лічильник для слідкування за кількістю кадрів, у яких виявлено ознаки втомленості. Далі, увага приділяється області очей, де використовуються методи виявлення обличчя за допомогою детектора обличчя Histogram of Oriented Gradients (HOG). Це допомагає визначити риси обличчя. Крім того, детектор орієнтирів обличчя використовується для точного виділення областей, що

відповідають очам. На цьому етапі попередньої обробки для обличчя використовується сітка MediaPipe для створення 3D-моделі обличчя. Далі, обчислюється показник, який визначає відкритість очей. Якщо очі закриті або майже закриті, EAR зменшується, що може вказувати на втому або сонливість.

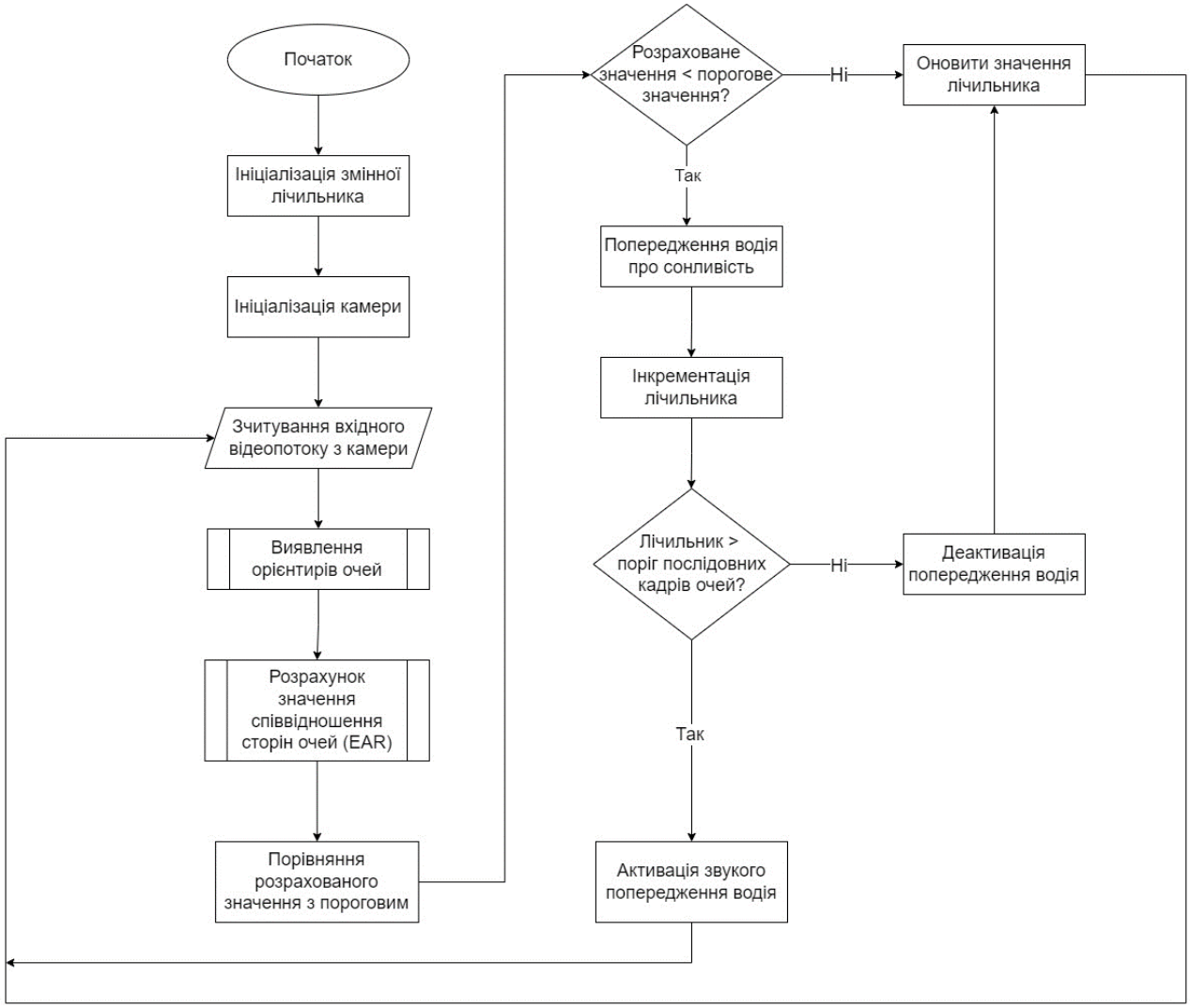


Рисунок 3.2 – Алгоритм виявлення та запобігання втомленості водія

Розраховане EAR порівнюється з попередньо встановленим пороговим значенням. Якщо EAR менше порогу, це може означати, що очі водія були закриті декілька кадрів. Алгоритм визначає, чи є необхідність оновлення значення лічильника, що може вказувати на тривале закриття очей. Якщо є відхилення, лічильник оновлюється. Якщо відхилення немає - виконання продовжується. Якщо

EAR менше порогового значення, лічильник збільшується на одиницю, що індикатор того, що очі водія залишалися закритими на додатковий кадр. Перевіряється, чи кількість кадрів, в яких виявлено закриті очі, перевищує встановлений поріг. Якщо лічильник перевищує поріг, це може вказувати на високу ймовірність втоми, тому активується звукове попередження для водія. Якщо лічильник не перевищує поріг, продовжується моніторинг наступного кадру без активації попередження. Якщо водій відкрив очі, звукове попередження вимикається.

3.2 Опис використаних технологій для розробки

Розглянемо детальніше використані в процесі розробки технології, а саме: мова програмування Python, бібліотеки OpenCV, Numpy, PyDub, MediaPipe, а також середовища розробки Jupyter Notebook та PyCharm.

3.2.1 Мова програмування Python

Python – це інтерпретована мова програмування високого рівня, яка відома своїм акцентом на читабельності та простоті. Його синтаксис розроблений таким чином, щоб бути зрозумілим і виразним, що робить його особливо привабливим для початківців і досвідчених програмістів. Як інтерпретована мова, Python виконує код рядок за рядком, що спрощує процес налагодження. Однією з ключових особливостей Python є його динамічна система типізації, яка означає, що типи змінних визначаються під час виконання. Це додає мові певний рівень гнучкості, але також вимагає ретельного тестування для керування потенційними помилками, пов’язаними з типом. Python неймовірно універсальний, широко використовується в різних сферах, таких як веб-розробка, аналіз даних, штучний інтелект, наукові обчислення та автоматизація. Ця універсальність підтримується повною стандартною бібліотекою та широкою екосистемою пакетів сторонніх розробників, які розширюють її функціональність у різноманітних областях.

Python є потужним інструментом для створення систем виявлення та запобігання сонливості водіїв завдяки своїй здатності інтегрувати різноманітні

технології обробки зображень та машинного навчання. Використання бібліотеки OpenCV дозволяє реалізувати відстеження рухів очей та обличчя в реальному часі, що є критично важливим для моніторингу ознак втоми. Аналізуючи відеопотік, програма може обчислювати Eye Aspect Ratio, що відображає частоту моргання та тривалість закриття очей. За допомогою алгоритмів машинного навчання, таких як ті, що пропонуються бібліотеками TensorFlow або PyTorch, система може навчитися розпізнавати складніші патерни поведінки, що вказують на втому або сонливість. Це може включати аналіз змін у розміщенні голови, частоти моргання, а також довгострокових змін в поведінці водія. Коли система виявляє ознаки сонливості, вона може використовувати звукові попередження для привертання уваги водія. Python має здатність інтегруватися з аудіосистемою, щоб відтворювати звукові сигнали, які можуть бути простими/складними голосовими повідомленнями.

3.2.2 Бібліотека OpenCV

OpenCV, розроблена Intel, є дуже популярною та широко використовуваною міжплатформною бібліотекою комп'ютерного зору, призначеною для обробки зображень у реальному часі. Будучи незамінним інструментом для всього, що пов'язано з комп'ютерним зором, OpenCV став стандартом де-факто в цій галузі. Навіть у 2023 році його популярність залишається непохитною, із вражаючим показником завантажень понад 29 000 завантажень щотижня. OpenCV в основному реалізовано на C і C++, що робить його сумісним з різними операційними системами, включаючи GNU/Linux, OS X, Windows, Android, iOS та інші. Одним із чудових аспектів OpenCV є його доступність як безкоштовного програмного забезпечення під ліцензією Apache 2. Крім того, він продовжує розвиватися завдяки активній розробці інтерфейсів для таких мов, як Python, Ruby, Matlab та багатьох інших. З великою колекцією понад 2500 алгоритмів бібліотека OpenCV пропонує повний ресурс для завдань комп'ютерного зору в реальному часі. Його документація та зразок коду полегшують застосування цих алгоритмів у багатьох областях. OpenCV використовується для з'єднання зображень із камери для створення супутникових або веб-карт, вирівнювання сканованих зображень,

зменшення шуму на медичних зображеннях, проведення аналізу об’єктів, забезпечення живлення систем безпеки та спостереження, включаючи виявлення вторгнень, автоматичний моніторинг і системи безпеки. На рисунку 3.3 представлений приклад розпізнавання обличчя з використанням можливостей мови програмування Python та функцій OpenCV.

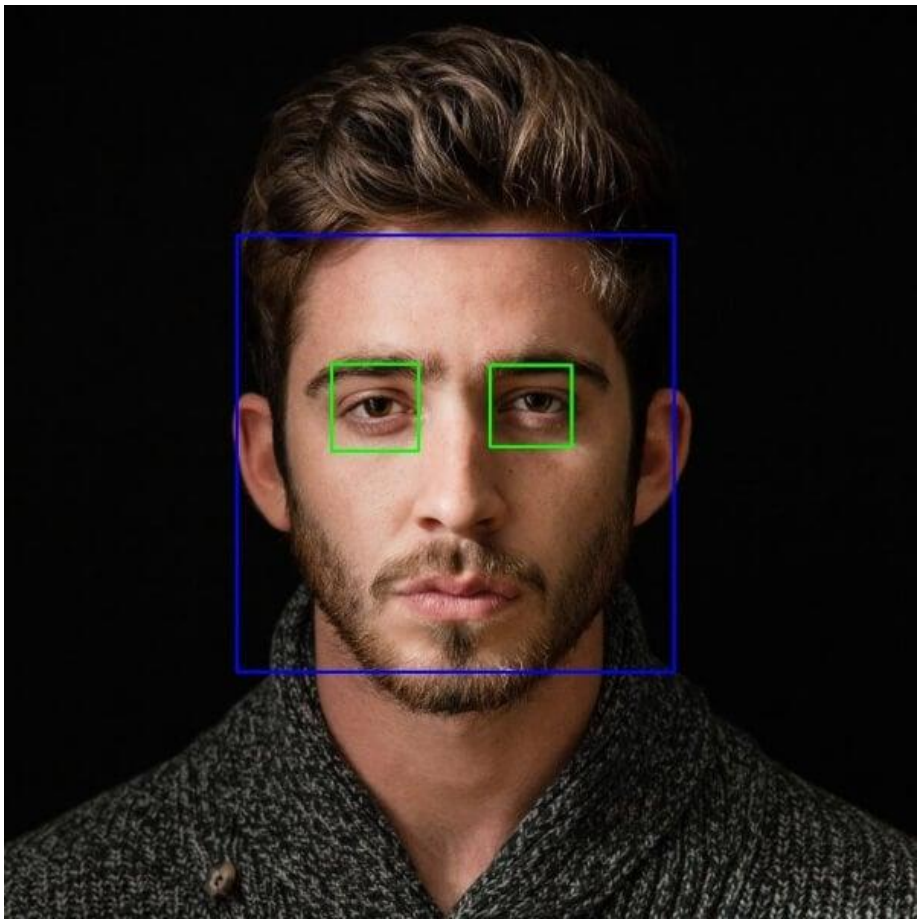


Рисунок 3.3 – Приклад розпізнавання обличчя з використанням OpenCV

3.2.3 Бібліотека NumPy

NumPy служить основним пакетом для наукових обчислень на мові програмування Python. NumPy пропонує різноманітні функції, включаючи об’єкт багатовимірного масиву, похідні об’єкти, такі як масковані масиви та матриці, а також широкий набір функцій, оптимізованих для швидких операцій з масивами. Ці операції охоплюють математичні обчислення, логічні маніпуляції, перетворення форми, сортування, вибір, операції введення/виведення, дискретні перетворення

Фур’є, основні обчислення лінійної алгебри, фундаментальні статистичні операції, випадкове моделювання та багато іншого. Основним елементом пакета NumPy є об’єкт ndarray, який інкапсулює n-вимірні масиви, що складаються з даних однорідних типів. Для оптимізації продуктивності численні операції над цими масивами виконуються в скомпільованому коді. На рисунку 3.4 представлений приклад масивів з різною розмірністю, які використовуються в NumPy.

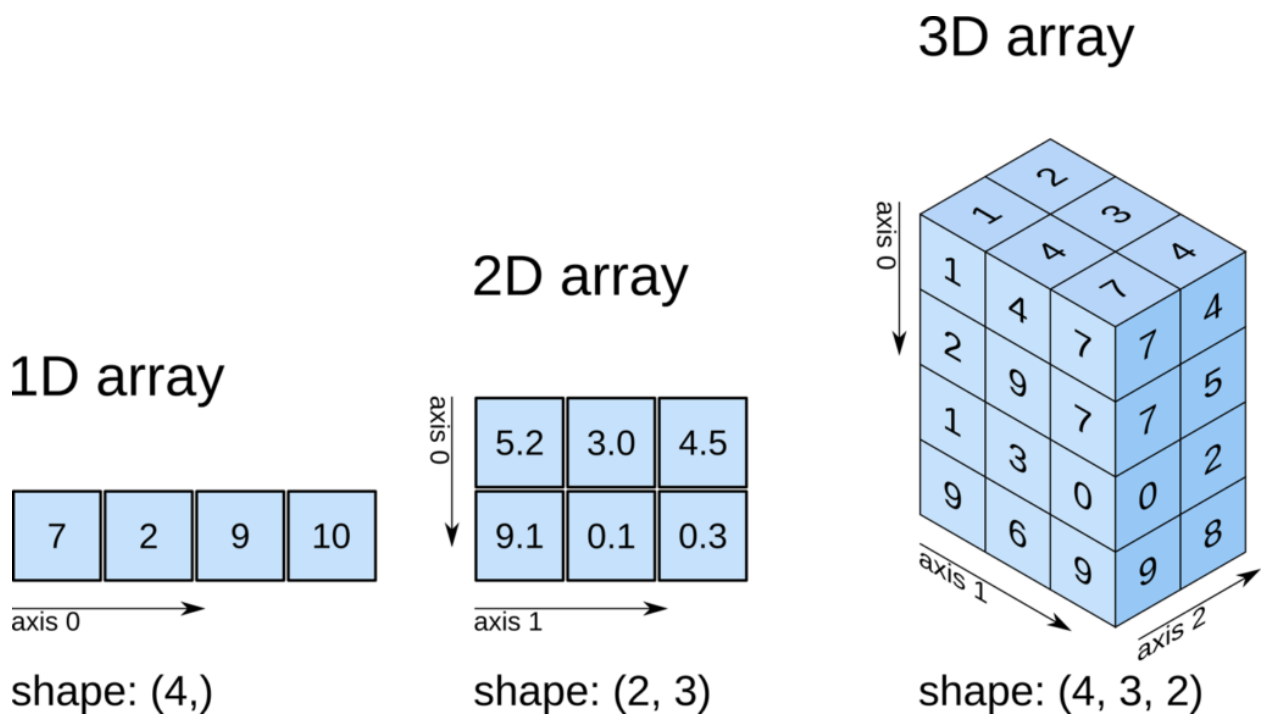


Рисунок 3.4 – Приклад масивів NumPy (зліва – представлення вектора, по центру – представлення матриці, зправа – представлення тензора)

Примітно, що масиви NumPy відрізняються від стандартних послідовностей Python кількома суттєвими ознаками. По-перше, масиви NumPy мають фіксований розмір після створення, на відміну від списків Python, які можуть динамічно зростати. Таким чином, зміна розміру масиву ndarray призводить до створення нового масиву та видалення вихідного. По-друге, усі елементи в масиві NumPy повинні використовувати один і той же тип даних, що призводить до узгодженого розподілу пам’яті для кожного елемента. Однак із цього правила є виняток,

оскільки NumPy підтримує масиви, що містять об’єкти Python, дозволяючи різні розміри елементів.

3.2.4 Бібліотека MediaPipe

MediaPipe – це платформа, яка може використовуватися для створення конвеєрів для аналізу різних типів сенсорних даних, таких як відео чи аудіо, за допомогою методів комп’ютерного зору. Цей потужний інструмент дозволяє створювати конвеєри сприйняття шляхом збирання модульних компонентів у формі графа. Його головна мета полягає в тому, щоб сприяти швидкому розвитку цих конвеєрів шляхом включення моделей штучного інтелекту для висновків та інших багаторазових компонентів. Крім того, MediaPipe спрощує розгортання додатків комп’ютерного зору на різних апаратних платформах, що робить його надзвичайно універсальним. Коли справа доходить до обробки даних, MediaPipe виходить за рамки типового підходу, який використовують нейронні мережі, такі як TensorFlow, PyTorch, CNTK або MXNet. Ці мережі обробляють дані прямим і детермінованим способом, де один вхід генерує один вихід. MediaPipe, однак, працює на вищому рівні абстракції, дозволяючи більш складну та динамічну поведінку. Він може обробляти сценарії, коли один вхід може генерувати нуль, один або навіть кілька виходів, що неможливо з традиційними нейронними мережами. Крім того, MediaPipe чудово підходить для потокової обробки, що важливо для таких завдань, як обробка відео та сприйняття ШІ. На відміну від методів пакетної обробки, які обробляють дані порціями, потокова обробка передбачає безперервний аналіз даних у міру їх надходження. OpenCV 4.0 представив Graph API, який дозволив будувати послідовності операцій обробки зображень у вигляді графа. Однак MediaPipe перевершує цю можливість, дозволяючи виконувати операції з різними типами даних і надаючи вбудовану підтримку для потокової передачі даних часових рядів. MediaPipe пропонує розробникам можливість поступово будувати конвеєр для своїх проектів. У цьому контексті конвеєр бачення – це набір компонентів, упорядкованих у орієнтований граф. Ці компоненти представляють вузли графіка. Зв'язок між калькуляторами встановлюється через дані. Ці потоки фіксують дані часових рядів у формі

«пакетів». Поєднання калькуляторів і потоків визначає графік потоку даних. Щоб забезпечити правильну послідовність пакетів у часовому ряді, вони організовані на основі своїх часових позначок. Кожен вхідний потік має власну чергу, що дозволяє вузлу-одержувачу споживати пакети з бажаною швидкістю. Ця функція сприяє гнучкості в управлінні потоком даних у всьому графіку. Розробники мають свободу поступово вдосконалювати конвеєр, вставляючи або замінюючи калькулятори в будь-якому місці на графіку. Крім того, можна створювати спеціальні калькулятори для задоволення конкретних вимог. Хоча калькулятори працюють паралельно, важливо зазначити, що кожен калькулятор виконується в одному потоці за раз. Це обмеження спрощує процес розробки, усуваючи потребу в складних знаннях багатопотокового програмування. Модуль трасування MediaPipe служить для запису подій синхронізації по всьому графіку. Він фіксує різні поля даних, пов’язані з цими подіями, включаючи час, мітку часу пакета, ідентифікатор даних, ідентифікатор вузла та ідентифікатор потоку. Крім того, модуль трасування генерує гістограми, які надають інформацію про використання ресурсів, наприклад, час процесора, що минув, для кожного калькулятора та потоку. Щоб активувати модуль трасування, користувачі можуть увімкнути його за допомогою налаштування конфігурації в GraphConfig. Крім того, користувачі мають можливість повністю виключити код модуля трасування за допомогою позначки компілятора. Ця гнучкість дозволяє налаштувати на основі конкретних вимог.

3.2.5 Середовище розробки Jupyter Notebook

Для тестування розробленого алгоритму на тестових зображення, було використано середовище розробки Jupyter Notebook. Jupyter Notebook – це веб-програма з відкритим кодом, яка революціонізувала спосіб взаємодії з кодом і даними. Він відомий тим, що дозволяє інтегрувати код, візуалізації та описовий текст в одному документі. Спочатку розроблений для Python, тепер він підтримує різні мови програмування завдяки використанню різних ядер, таких як Julia, R і Scala. Інтерактивний обчислювальний аспект Jupyter Notebook є однією з його основних функцій. Jupyter Notebook надає можливість писати та виконувати код у реальному часі, що підходить для дослідницької роботи, аналізу даних і навчання

програмуванню. Ця інтерактивність також робить його чудовим інструментом для експериментування зі змінами коду та спостереження за їх результатами. Однією з видатних особливостей Jupyter Notebook є підтримка мультимедіа. Він підтримує можливість додавання зображення, відео, LaTeX і навіть JavaScript безпосередньо у документи. Ця можливість робить його потужним інструментом для створення комплексних навчальних матеріалів або докладних звітів. Jupyter Notebook легко інтегрується з популярними бібліотеками візуалізації даних, дозволяючи користувачам створювати та відображати графіки та діаграми. Це особливо корисно в таких сферах, як наука про дані та машинне навчання, де візуалізація даних є ключовою частиною процесу аналізу. Окрім обчислювальних можливостей, Jupyter Notebook підтримує Markdown для описового тексту. Це дозволяє користувачам додавати контекст, пояснення або інструкції поряд зі своїм кодом, роблячи блокноти засобом для документації розроблених частин коду.

На рисунку 3.5 представлений інтерфейс Jupyter Notebook.

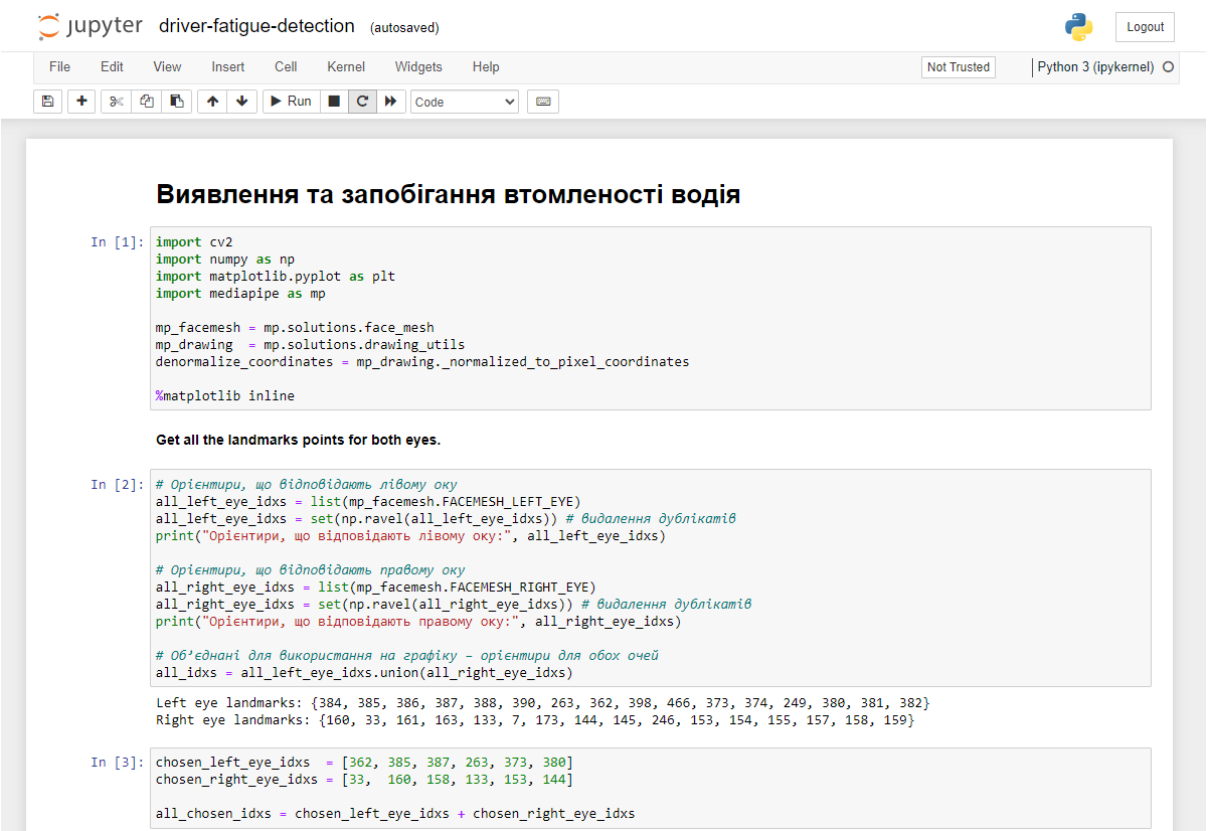


Рисунок 3.5 – Зовнішній вигляд Jupyter Notebook

3.2.6 Середовище розробки PyCharm

PyCharm – це інтегроване середовище розробки (IDE), призначене для програмування на Python, розроблене JetBrains. Він широко відомий завдяки своїй інтелектуальній допомозі в кодуванні, надаючи розширені можливості, як-от автоматичний рефакторинг коду та глибокий аналіз коду, що значно підвищує продуктивність і якість коду. Редактор коду PyCharm призначений не лише для ефективного написання коду; він також підтримує Python із допомогою для стилів кодування та стандартів, що полегшує підтримку узгодженої кодової бази. Однією з видатних особливостей PyCharm є його потужні можливості налагодження. Він пропонує графічні інструменти налагодження, які допомагають розробникам візуалізувати виконання коду, полегшуючи розуміння складних кодових баз і виявлення помилок. Ця IDE також чудово підтримує веб-розробку, інтегруючись із популярними веб-фреймворками Python, такими як Django, Flask і Pyramid. Крім того, він пропонує комплексні інструменти для зовнішніх технологій, включаючи HTML, CSS і JavaScript. Для тих, хто працює з базами даних, PyCharm містить інструменти для керування базами даних і підтримки SQL, що спрощує процес розробки програм, керованих даними. IDE також відмінно справляється з керуванням середовищами та залежностями Python, спрощуючи завдання оперування різними середовищами проекту, що є проблемою в екосистемі Python. Сумісність PyCharm із Jupyter Notebook додає ще один рівень його універсальності. Ця функція дозволяє розробникам і дослідникам даних безпосередньо створювати, редагувати та запускати файли Jupyter Notebook у PyCharm, поєднуючи потужні функції PyCharm з інтерактивними можливостями Jupyter Notebooks для обробки даних.

Ця інтеграція означає, що можна писати код Python у форматі Jupyter Notebook, виконувати його, переглядати вихідні дані та перетворювати блокноти на сценарії Python для більш масштабної розробки застосунків.

3.3 Опис реалізації алгоритму та програмної системи

На рисунках 3.6 – 3.7 представлений код, який призначений для завантаження зображення, ініціаліазації компонентів MediaPipe та обробки. Код визначає індекси точок для лівого та правого очей, використовуючи предвизначені індекси від MediaPipe, та виводить їх. Зображення завантажується через OpenCV, конвертується в RGB формат, а потім відображається за допомогою Matplotlib.

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import mediapipe as mp

mp_facemesh = mp.solutions.face_mesh
mp_drawing  = mp.solutions.drawing_utils
denormalize_coordinates = mp_drawing._normalized_to_pixel_coordinates

%matplotlib inline
```

```
# Орієнтири, що відповідають лівому оку
all_left_eye_idxs = list(mp_facemesh.FACEMESH_LEFT_EYE)
all_left_eye_idxs = set(np.ravel(all_left_eye_idxs)) # видалення дублікатів
print("Орієнтири, що відповідають лівому оку:", all_left_eye_idxs)

# Орієнтири, що відповідають правому оку
all_right_eye_idxs = list(mp_facemesh.FACEMESH_RIGHT_EYE)
all_right_eye_idxs = set(np.ravel(all_right_eye_idxs)) # видалення дублікатів
print("Орієнтири, що відповідають правому оку:", all_right_eye_idxs)

# Об'єднані для використання на графіку – орієнтири для обох очей
all_idxs = all_left_eye_idxs.union(all_right_eye_idxs)
```

```
Орієнтири, що відповідають лівому оку: {384, 385, 386, 387, 388, 390, 263, 362, 398, 466, 373, 374, 249, 380, 381, 382}
Орієнтири, що відповідають правому оку: {160, 33, 161, 163, 133, 7, 173, 144, 145, 246, 153, 154, 155, 157, 158, 159}
```

Рисунок 3.6 – Лістинг коду для обробки зображення та визначення орієнтирів лівого та правого ока

```
chosen_left_eye_idxs  = [362, 385, 387, 263, 373, 380]
chosen_right_eye_idxs = [33,  160, 158, 133, 153, 144]

all_chosen_idxs = chosen_left_eye_idxs + chosen_right_eye_idxs
```

```
# Завантаження зображення.

image = cv2.imread("input/Australian-actor-Chris-Hemsworth-2019.jpg")
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB) # convert to RGB

image = np.ascontiguousarray(image)

imgH, imgW, _ = image.shape

plt.imshow(image);
```

Рисунок 3.7 – Лістинг коду для обробки зображення та визначення орієнтирів лівого та правого ока (продовження)

Далі, на рисунку 3.8 представлений вигляд тестового зображення, яке в подальшому буде використано для проведення тестування розроахунку EAR.

Рисунок 3.8 – Завантажене тестове зображення

На рисунку 3.9 представлена ініціалізація інференції FaceMesh, а також вивід результату змінної, яка індикує доступність розпізнавання зображення.

```
with mp_facemesh.FaceMesh(
    static_image_mode=True,
    max_num_faces=1,
    refine_landmarks=False,
    min_detection_confidence=0.5,
    min_tracking_confidence= 0.5,
) as face_mesh:

    results = face_mesh.process(image)

print(bool(results.multi_face_landmarks)) # Індикатор доступності розпізнавання
```

```
True
```

Рисунок 3.9 – Ініціалізація інференції FaceMesh з MediaPipe

Наступна мета передбачає доступ до визначених орієнтирів обличчя. Слід зазначити, що конвеєр має можливість виявляти кілька граней і визначати орієнтири для кожної з цих граней. Об’єкт results.multi_face_landmarks представляє список, який містить визначення орієнтирів для кожного обличчя. Довжина цього списку залежить від значення, встановленого для параметра "max_num_faces". На рисунку 3.10 представлений лістинг коду для отримання орієнтирів обличчя.

```
# Отримати перший орієнтир на першому виявленому обличчі

landmark_0 = results.multi_face_landmarks[0].landmark[0]
print(landmark_0)

landmark_0_x = landmark_0.x * imgW
landmark_0_y = landmark_0.y * imgH
landmark_0_z = landmark_0.z * imgW

print("X:", landmark_0_x)
print("Y:", landmark_0_y)
print("Z:", landmark_0_z)

print()
print("Загальна довжина '.landmark':", len(results.multi_face_landmarks[0].landmark))
```

```
x: 0.4618982970714569
y: 0.5582389831542969
z: -0.05166543275117874

X: 539.4972109794617
Y: 893.182373046875
Z: -60.34522545337677

Загальна довжина '.landmark': 468
```

Рисунок 3.10 – Отримання визначених орієнтирів для обличчя

Далі, побудуємо графік з визначеними орієнтирами обличчя. Для цього буде використана розроблена функція plot. Функція plot використовується для візуалізації обличчя та ознак очей на зображенні. Вона приймає кілька параметрів: основне зображення (img_dt), зображення для відображення всіх точок ока (img_eye_lmks), зображення для відображення вибраних точок ока (img_eye_lmks_chosen), дані про точки орієнтирів обличчя (face_landmarks), параметри для візуалізації як товщина ліній, радіус кіл, та назва для можливого збереження зображення (name). Функція створює копії початкового зображення для різних типів візуалізацій. Вона визначає специфікації для малювання за допомогою mediapipe. Фігура Matplotlib створюється для відображення зображень. Далі виконується формування характеристик на зображеннях: триангуляція обличчя та окремі точки для очей. Функція відображає три різних зображення з налаштованими параметрами візуалізації.

Це дозволяє детально візуалізувати та аналізувати різні аспекти обличчя на зображенні, особливо ознаки очей. На рисунку 3.11 представлена реалізація функції побудови відображення орієнтирів, а на рисунку 3.12 представлений результат побудови всіх орієнтирів обличчя і визначених орієнтирів обличчя.

```python
def plot(
    *,
    img_dt,
    img_eye_lmks=None,
    img_eye_lmks_chosen=None,
    face_landmarks=None,
    ts_thickness=1,
    ts_circle_radius=2,
    lmk_circle_radius=3,
    name="1",
):
    image_drawing_tool = img_dt
    image_eye_lmks = img_dt.copy() if img_eye_lmks is None else img_eye_lmks
    img_eye_lmks_chosen = img_dt.copy() if img_eye_lmks_chosen is None else img_eye_lmks_chosen
    connections_drawing_spec = mp_drawing.DrawingSpec(
        thickness=ts_thickness, circle_radius=ts_circle_radius, color=(255, 255, 255)
    )

    fig = plt.figure(figsize=(20, 15))
    fig.set_facecolor("white")

    mp_drawing.draw_landmarks(
        image=image_drawing_tool,
        landmark_list=face_landmarks,
        connections=mp_facemesh.FACEMESH_TESSELATION,
        landmark_drawing_spec=None,
        connection_drawing_spec=connections_drawing_spec,
    )

    landmarks = face_landmarks.landmark
    for landmark_idx, landmark in enumerate(landmarks):
        if landmark_idx in all_idxs:
            pred_cord = denormalize_coordinates(landmark.x, landmark.y, imgW, imgH)
            cv2.circle(image_eye_lmks, pred_cord, lmk_circle_radius, (255, 255, 255), -1)

        if landmark_idx in all_chosen_idxs:
            pred_cord = denormalize_coordinates(landmark.x, landmark.y, imgW, imgH)
            cv2.circle(img_eye_lmks_chosen, pred_cord, lmk_circle_radius, (255, 255, 255), -1)
```

Рисунок 3.11 – Лістинг реалізації функції побудови визначених орієнтирів

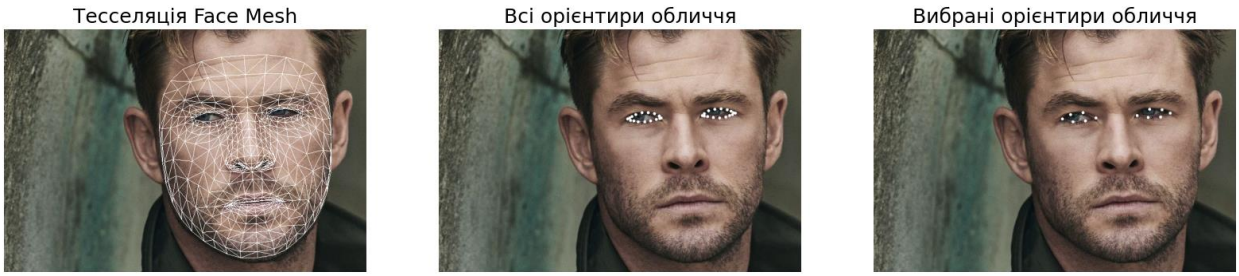


Рисунок 3.12 – Результати відображення орієнтирів обличчя (зліва – побудова 3D маски, по центру – всі орієнтирі, зправа – орієнтири EAR)

На рисунку 3.13 представлений лістинг функцій для розрахунку відстані та метрики співвідношення сторін очей EAR Перша функція, distance, визначає евклідову відстань між двома точками. Вона приймає два аргументи, point_1 та point_2, які представляють собою координати цих точок. Відстань розраховується

за допомогою визначення суми квадратів різниць між відповідними координатами точок, а потім взяття квадратного кореня з суми.

```python
def distance(point_1, point_2):
    dist = sum([(i - j) ** 2 for i, j in zip(point_1, point_2)]) ** 0.5
    return dist
```

```python
def get_ear(landmarks, refer_idxs, frame_width, frame_height):
    try:
        # Розрахунок евклідової відстані
        coords_points = []
        for i in refer_idxs:
            lm = landmarks[i]
            coord = denormalize_coordinates(lm.x, lm.y, frame_width, frame_height)
            coords_points.append(coord)

        P2_P6 = distance(coords_points[1], coords_points[5])
        P3_P5 = distance(coords_points[2], coords_points[4])
        P1_P4 = distance(coords_points[0], coords_points[3])

        # Розрахунок EAR
        ear = (P2_P6 + P3_P5) / (2.0 * P1_P4)

    except:
        ear = 0.0
        coords_points = None

    return ear, coords_points
```

Рисунок 3.13 – Функції для розрахунку відстані та метрики співвідношення сторін очей EAR

Функція get_ear призначена для розрахунку співвідношення сторін очей. Вона приймає кілька аргументів: список земних пам'яток landmarks, які визначають ключові точки на обличчі, список індексів refer_idxs, що вказують на позиції цих точок у порядку P1, P2, P3, P4, P5, P6, а також ширину frame_width і висоту frame_height кадру, на якому було зафіксовано обличчя. У функції спочатку відбувається перетворення координат земних пам'яток з нормалізованої форми у звичайні координати з врахуванням розмірів кадру. Потім обчислюються евклідові відстані між парами точок: між P2 і P6, P3 і P5, а також між P1 і P4. Ці відстані використовуються для розрахунку EAR. Якщо виникає помилка, EAR встановлюється як 0.0, а координати точок як None.

На рисунку 3.14 представлений лістинг для тестування розрахунку значення співвідношення сторін очей (EAR) для тестових зображень.

```
image_eyes_open  = cv2.imread("input/chris-slider-757x505.jpg")[:, :, ::-1]
image_eyes_close = cv2.imread("input/istockphoto-1163491597-612x612.jpg")[:, :, ::-1]

for idx, image in enumerate([image_eyes_open, image_eyes_close]):

    image = np.ascontiguousarray(image)
    imgH, imgW, _ = image.shape

    custom_chosen_lmk_image = image.copy()

    with mp_facemesh.FaceMesh(refine_landmarks=True) as face_mesh:
        results = face_mesh.process(image)

        if results.multi_face_landmarks:

            for face_id, face_landmarks in enumerate(results.multi_face_landmarks):

                landmarks = face_landmarks.landmark
                EAR = calculate_avg_ear(landmarks, chosen_left_eye_idxs, chosen_right_eye_idxs, imgW, imgH)

                cv2.putText(custom_chosen_lmk_image, f"EAR: {round(EAR, 2)}", (1, 24),
                            cv2.FONT_HERSHEY_COMPLEX, 0.9, (0, 255, 0), 2
                )

                plot(img_dt=image.copy(),img_eye_lmks_chosen=custom_chosen_lmk_image, face_landmarks=face_landmarks,
                     ts_thickness=1, ts_circle_radius=3, lmk_circle_radius=3
                )
```

Рисунок 3.14 – Лістинг для розрахунку значення співвідношення сторін очей (EAR) для тестових зображень

На рисунку 3.15 представлений результат розрахунку EAR для зображення, де представлена людина з відкритими очами.

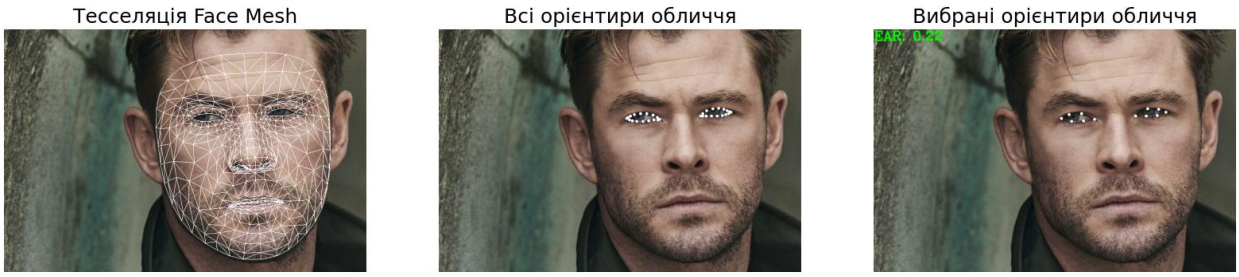


Рисунок 3.15 – Результат розрахунку EAR для відкритих очей (EAR = 0.22)

На рисунку 3.16 представлений результат розрахунку EAR для зображення, де представлена людина з закритими очами.

Рисунок 3.16 – Результат розрахунку EAR для відкритих очей (EAR = 0.03)

На основі вищенаведеного тестування можна зробити висновок, що алгоритм вірно розраховує значення метрики співвідношення сторін очей.

Також, для програмної системи розроблено клас AudioHandler, який відповідає за сповіщення водія при виявленні небезпеки. На рисунку 3.17 представлена реалізація конструктора розробленого класу AudioHandler.

```
class AudioHandler:
    def __init__(self, sound_file_path: str = ""):

        self.custom_audio = AudioSegment.from_file(file=sound_file_path, format="wav")
        self.custom_audio_len = len(self.custom_audio)

        self.ms_per_audio_segment: int = 20
        self.audio_segment_shape: tuple

        self.play_state_tracker: dict = {"curr_segment": -1}
        self.audio_segments_created: bool = False
        self.audio_segments: list = []
```

Рисунок 3.17 – Конструктор класу AudioHanlder

Конструктор класу AudioFrameHandler ініціалізує кілька атрибутів для обробки аудіофайлів. Він приймає один параметр sound_file_path, який є шляхом до аудіофайлу. Цей файл завантажується у форматі WAV. Також встановлюється кількість мілісекунд на один аудіосегмент. Додатково, існує механізм для

відстеження стану відтворення аудіосегментів. Далі, на рисунку 3.18 представлена реалізація методу класу prepare_audio для підготовки аудіо.

```
1 usage
def prepare_audio(self, frame: av.AudioFrame):
    raw_samples = frame.to_ndarray()
    sound = AudioSegment(
        data=raw_samples.tobytes(),
        sample_width=frame.format.bytes,
        frame_rate=frame.sample_rate,
        channels=len(frame.layout.channels),
    )

    self.ms_per_audio_segment = len(sound)
    self.audio_segment_shape = raw_samples.shape

    self.custom_audio = self.custom_audio.set_channels(sound.channels)
    self.custom_audio = self.custom_audio.set_frame_rate(sound.frame_rate)
    self.custom_audio = self.custom_audio.set_sample_width(sound.sample_width)

    self.audio_segments = [
        self.custom_audio[i_: i + self.ms_per_audio_segment]
        for i in range(0, self.custom_audio_len - self.custom_audio_len % self.ms_per_audio_segment, self.ms_per_audio_segment)
    ]
    self.total_segments = len(self.audio_segments) - 1

    self.audio_segments_created = True
```

Рисунок 3.18 – Метод класу для підготовки сигналу оповіщення

Метод prepare_audio класу AudioHandler обробляє аудіофрейм, отриманий від бібліотеки av. Він перетворює аудіофрейм у масив numpy, а потім створює об'єкт AudioSegment, використовуючи ці дані. Для цього метод бере байти з масиву, інформацію про ширину зразка, частоту дискретизації та кількість каналів з фрейму. Після цього метод оновлює деякі атрибути класу: встановлює кількість мілісекунд на аудіосегмент, оновлює форму сегмента аудіо, а також налаштовує основні параметри custom_audio (кількість каналів, частоту дискретизації та ширину зразка) відповідно до параметрів новоствореного AudioSegment. Далі метод розділяє custom_audio на менші сегменти і зберігає їх. В кінці метод встановлює audio_segments_created в True.

На рисунках 3.19 – 3.20 представлений код, який відповідає за можливість зміни параметрів системи і відстеження відео в реальному часі.

```python
import os
import av
import threading
import streamlit as st
from streamlit_webrtc import (
    VideoHTMLAttributes,
    webrtc_streamer
)
import streamlit_nested_layout

from audio_utils import AudioFrameHandler
from detection_utils import VideoFrameHandler


alarm_file_path = os.path.join("audio", "wake_up.wav")

# Компоненти бібліотеки Streamlit
st.set_page_config(
    page_title="Виявлення сонливості",
    page_icon="✊ ",
    layout="wide",  # centered, wide
    initial_sidebar_state="expanded"
)

st.sidebar.title('Налаштування')

user_choice = st.sidebar.selectbox("Виберіть тип розпізнавання", ['Розпізнавання сонливості з відео'])
```

Рисунок 3.19 – Код для ініціалізації параметрів програмної системи

```python
if user_choice == 'Розпізнавання сонливості з відео':
    col1, col2 = st.columns(spec=[4, 2], gap="medium")

    st.markdown(
        "<h1 style='text-align: center; color: black;'>Розпізнавання сонливості з відео 😳😪😴</h1>", unsafe_allow_html=True
    )
    with st.container():
        c1, c2 = st.columns(2)
        with c1:
            # Проміжок часу (у секундах), який необхідно зачекати, перш ніж спрацює оповіщення
            WAIT_TIME = st.slider(
                "Кількість секунд очікування перед запуском оповіщення:", 0.0, 5.0, 1.0, 0.25
            )
        with c2:
            # Найнижче дійсне значення Eye Aspect Ratio. Ідеальні значення = [0,15, 0,2].
            EAR_THRESH = st.slider(
                "Встановлення порогового значення співвідношення сторін очей:", 0.0, 0.4, 0.18, 0.01
            )
```

Рисунок 3.20 – Код для налаштування параметрів роботи системи

На рисунку 3.21 представлені додаткові функції, які відповідають за управління відео та оповіщенням водія, а також створення об’єкта WebRTC.

```
1 usage
def video_frame_callback(frame: av.VideoFrame):
    frame = frame.to_ndarray(format="bgr24")  # Декодування та перетворення кадру в RGB

    frame, play_alarm = video_handler.process(frame, thresholds)  # Обробка вхідного потоку
    with lock:
        shared_state["play_alarm"] = play_alarm  # Оновлення стану бібліотеки

    return av.VideoFrame.from_ndarray(frame, format="bgr24")  # Кодування та повернення кадру BGR


1 usage
def audio_frame_callback(frame: av.AudioFrame):
    with lock:
        play_alarm = shared_state["play_alarm"]

    new_frame: av.AudioFrame = audio_handler.process(frame, play_sound=play_alarm)
    return new_frame


ctx = webrtc_streamer(
    key="drowsiness-detection",
    video_frame_callback=video_frame_callback,
    audio_frame_callback=audio_frame_callback,
    media_stream_constraints={
        "video": {"height": {"ideal": 480}}, "audio": True
    },
    video_html_attrs=VideoHTMLAttributes(autoPlay=True, controls=False, muted=False),
)
```

Рисунок 3.21 – Функції, які відповідають за управління відео та оповіщенням водія та створення об’єкта WebRTC (відео в режимі реального часу)

Функція video_frame_callback приймає відеофрейм як вхідний параметр. Перший крок - це декодування та перетворення кадру в формат RGB. Потім цей кадр обробляється за допомогою video_handler.process, який може включати різні види обробки, такі як детекція руху чи зміна кольорових параметрів. Результатом цієї обробки є оновлений кадр та флаг play_alarm, який позначає, чи потрібно відтворювати сигнал тривоги. Стан play_alarm зберігається у спільному стані. Оновлений кадр повертається у форматі BGR. audio_frame_callback керує аудіофреймами. Він використовує загальний стан для визначення, чи потрібно відтворювати оповіщення (play_alarm). Це стан використовується в методі audio_handler.process, який обробляє вхідний аудіофрейм. В залежності від стану

play_alarm, цей метод може додавати до аудіо сигнали оповіщення чи інші ефекти. Результатом є новий аудіофрейм, який потім повертається. Виклик webrtc_streamer ініціалізує веб-стрімінг з використанням визначених раніше функцій для обробки відео та аудіо. Він використовує унікальний ключ для ідентифікації стріму, задає обмеження для медіа потоків (наприклад, відео 480 пікселів) та задає параметри HTML для відео, такі як автоматичне відтворення, наявність контролів та стан гучності.

На рисунку 3.22 представлений приклад виявлення стану без ознак втомленості з використанням розробленої програмної системи.

Рисунок 3.22 – Приклад виявлення стану людини без ознак втомленості

На рисунку 3.22 представлений приклад виявлення стану сонливості водія з автоматичним оповіщенням при перевищенні порогового значення.

Рисунок 3.22 – Приклад розпізнавання стану сонливості водія

На основі вищенаведених результатів тестування програмної системи можна зробити висновок, що виявлення втомленості водія відбувається миттєво, а за цим слідує звукове оповіщення водія про небезпечну ситуацію. В результаті проведених тестувань виявлено, що система виявлення може працювати під різними кутами, на більшій відстані та без гарного освітлення.

Для підвищення ефективності системи виявлення сонливості водія, пропонується використовувати додаткові технології та механізми. Наприклад, інтелектуальна система управління кліматом в салоні автомобіля, яка автоматично регулює температуру, може допомогти водієві залишатися пильним, знижуючи температуру для зменшення відчуття втоми. Використання системи адаптивнного освітлення також може бути корисним, змінюючи інтенсивність та колір освітлення для зниження сонливості водія.

Інтеграція системи зі смарт-гаджетами, які носить водій, дозволить забезпечити додаткові дані про його фізіологічний стан у поєднанні з EAR.

ВИСНОВКИ

В першому розділі кваліфікаційної роботи було проведено аналіз предметної області. На основі розглянутих досліджень виявлено, що під час тривалих подорожей однією з найбільших небезпек, з якою стикаються водії, є ризик заснути за кермом. Цей ризик особливо помітний під час подорожей у нічний час та під час їзди одноманітними дорогами. Зроблено висновок, що час реакції водія значно скорочується після чотирьох годин безперервної їзди, а до восьми годин він зменшується лише до однієї шостої від початкової здатності. Досліджено існуючі системи виявлення та запобігання втомленості. Також, розглянуті існуючі методи виявлення та запобігання втомленості водія.

В другому розділі кваліфікаційної роботи були проаналізовані існуючі алгоритмічні рішення виявлення сонливості водія, а саме: алгоритм Віоли-Джонса, алгоритм головних компонент, алгоритм локальних бінарних шаблонів та алгоритми глибого навчання. Використання методу головних компонентів впропонує перевагу зменшення розмірності матриць, залучених до обчислень. Як результат, цей алгоритм вимагає мінімальних обчислювальних ресурсів і легко реалізується програмно. Однією з особливостей цього алгоритму є те, що він працює на основі матриці інтенсивності пікселів як вхідних даних, усуваючи потребу в будь-якій попередній обробці зображення. Однак важливо визнати деякі обмеження алгоритму, зокрема його чутливість до таких факторів, як умови освітлення, масштаб, фон і поворот обличчя. В процесі дослідження визначено, що Eigenfaces забезпечує оптимальні результати при застосуванні до фотографій, знятих в «ідеальних» умовах, що характеризуються рівномірним освітленням і рівним фоном. Тим не менш, варто зазначити, що реальна фотографія часто не відповідає цим ідеальним умовам. Крім того, через цілісний характер обробки зображення будь-який неоднорідний фон може спотворити результати алгоритму. Алгоритм LBPH відрізняється від головних компонент тим, що досліджує конкретні ділянки зображення, а не все зображення. На відміну від Eigenfaces, який зосереджується на загальному обличчі, LBPH аналізує різницю в інтенсивності

пікселів. Однак цей підхід також робить LBPH чутливим до шуму, наявного на фотографії. У випадках, коли зображенню не вистачає чіткої та визначеної інтенсивності, наприклад, бляклим зображенням, LBPH може важко ідентифікувати обличчя в ньому через свою чутливість до шуму. Метод Віоли-Джонса відомий ефективними та швидкими можливостями обробки зображень, які досягаються за допомогою реалізації каскадів Хаара та методів посилення. Однією з важливих переваг цього методу є його здатність виявляти обличчя в будь-якій частині фотографії. Визначено, що точність визначення обличчя значно знижується, коли обличчя повертаються під кутом більше 30°, на відміну від облич, які дивляться прямо в камеру. Це обмеження спричинило зростання популярності згорткових нейронних мереж, оскільки вони забезпечують високоточне розпізнавання обличчя порівняно з вищезгаданими методами. Одним із ключових аспектів, який слід враховувати, є те, що для підвищення точності систем розпізнавання облич на основі CNN потрібне тривале навчання на великому наборі даних. Цей навчальний процес вимагає значних обчислювальних ресурсів і витрат часу. Тим не менш, використання CNN довело ефективність у досягненні високих результатів розпізнавання обличчя.

В третьому розділі кваліфікаційної роботи розглянуто проєктування та розробка програмної системи для виявлення та запобігання втомленості водія. Для виявлення втомленості водія використовується техніка визначення співвідношення сторін очей. Використання функції EAR для виявлення моргання має кілька переваг перед традиційними методами обробки зображень. У звичайних підходах першим кроком є локалізація очей на зображенні, а потім порогове визначення для ідентифікації зіниць очей. Згодом миготіння виявляється шляхом моніторингу зникнення білої області. Натомість використана функція EAR базується на обчисленні співвідношення відстаней між конкретними орієнтирами обличчя, пов’язаними з очима водія.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. The case for drowsiness detection systems. URL: https://www.here.com/driver-drowsiness-detection (дата звернення: 10.11.2023).

2. What are driver drowsiness detection systems and how do they work? URL: https://www.tom.com/driver-drowsiness-detection (дата звернення: 10.11.2023).

3. V. Kalisetti, V. S. C. Vasarla, S. B. Kolli, R. Varaparla, V. Enireddy and M. Mohammed. Analysis of Driver Drowsiness Detection Methods. International Conference on Electronics and Renewable Systems. 2023. pp. 1481-1485. URL: https://ieeexplore.ieee.org/document/10084986 (дата звернення: 12.11.2023).

4. A. M. Al-madani, A. T. Gaikwad, V. Mahale, Z. A. T. Ahmed and A. A. A. Shareef. Real-time Driver Drowsiness Detection based on Eye Movement and Yawning using Facial Landmark. International Conference on Computer Communication and Informatics. 2021, pp. 1-4. URL: https://ieeexplore.ieee.org/document/9457005 (дата звернення: 12.11.2023).

5. Driver assistance systems | Audi MediaCenter. URL: https://www.audi-mediacenter.com/en/driver-assistance-systems (дата звернення: 12.11.2023).

6. How Does Mercedes-Benz Attention Assist Work? URL: https://www.mercedesbenz/attention-assist (дата звернення: 12.11.2023).

7. How Driver Alert Control Works? URL: https://www.crestvolvocars.com/ /how-driver-alert-control-works.html (дата звернення: 12.11.2023).

8. E. Tadesse, W. Sheng and M. Liu. Driver drowsiness detection through HMM dynamic modeling. *IEEE International Conference.* 2014, pp. 4003-4008. URL: https://ieeexplore.ieee.org/document/6907440 (дата звернення: 12.11.2023).

9. C. C. Ukwuoma and C. Bo. Deep Learning Review on Drivers Drowsiness Detection. *Engineering Science International Conference.* 2019. pp. 1-5. URL: https://ieeexplore.ieee.org/document/9024642 (дата звернення: 12.11.2023).

10. L. D. S. Cueva and J. Cordero. Advanced Driver Assistance System for the drowsiness detection using facial landmarks. *Conference on Information Systems and*

Technologies. 2020. pp. 1-4. URL: https://ieeexplore.ieee.org/document/9140893 (дата звернення: 12.11.2023).

11. Suherwin, Z. Zainuddin and A. A. Ilham. The Performance of Face Recognition Using the Combination of Viola-Jones, Local Binary Pattern Histogram and Euclidean Distance. Conference on Computational Sciences. 2020. pp. 1-4. URL: https://ieeexplore.ieee.org/document/9299073 (дата звернення: 12.11.2023).

12. A. Rizqullah, N. F. A. Hakim, S. A. T. A. Azhima and I. Kustiawan. Face Recognition Based on Viola-Jones Algorithm as Dataset for Image Classification. International Electrical Engineering Conference. 2021. pp. 295-298. URL: https://ieeexplore.ieee.org/document/9774251 (дата звернення: 14.11.2023).

13. A. A. Sambhe and A. V. Deorankar. Face Detection And Recognization System. *International Conference on Advances in Computing, Communication Control and Networking.* 2022. pp. 1175-1179. URL: https://ieeexplore.ieee.org/document/10074142 (дата звернення: 16.11.2023).

14. R. A. Movahed, M. Rezaeian, S. Javadifar and M. Alimoradijazi. A Face Recognition Framework Based on the Integration of Eigenfaces Algorithm and Image Registration Technique. *Conference on Engineering*. 2020. pp. 26-30. URL: https://ieeexplore.ieee.org/document/9319457 (дата звернення: 16.11.2023).

15. M. Gupta, K. Bisht and D. Upadhyay. HaarCascade and LBPH Algorithms in Face Recognition. *World Conference on Computing.* 2023. pp. 1-4. URL: https://ieeexplore.ieee.org/document/10235019 (дата звернення: 16.11.2023).

16. Convolutional Neural Network (CNN) in Machine Learning. URL: https://towardsdatascience.com/cnn-explained (дата звернення: 16.11.2023).

17. LeNet-5 Neural Network Architecture Explained. URL: https://paperswithcode.com/method/lenet (дата звернення: 16.11.2023).

18. ResNet: The Basics and 3 ResNet Extensions. URL: https://datagen.tech/guides/computer-vision/resnet/ (дата звернення: 18.11.2023).

19. Different Types of CNN Architectures Explained: Examples. URL: https://towardsdatascience.com/5-cnn-architectures (дата звернення: 16.11.2023).

20. A. U. Rafid, A. I. Chowdhury, A. R. Niloy and N. Sharmin. A Deep Learning Based Approach for Real-time Driver Drowsiness Detection. *International Conference on Electrical Engineering*. 2021, pp. 1-5. URL: https://ieeexplore.ieee.org/document/9667944 (дата звернення: 18.11.2023).

21. A. Suresh, A. S. Naik, A. Pramod, N. Ashwin Kumar and N. Mayadevi. Analysis and Implementation of Deep Convolutional Neural Network Models for Intelligent Driver Drowsiness Detection System. *International Conference on Intelligent Computing and Control Systems.* 2023. pp. 553-559. URL: https://ieeexplore.ieee.org/document/10142299 (дата звернення: 18.11.2023).

22 B. Ganguly, D. Dey and S. Munshi. An Integrated System for Drowsiness Detection Using Deep Learning. IEEE VLSI Device Circuit. 2022. pp. 55-59. URL: https://ieeexplore.ieee.org/document/9811442 (дата звернення: 22.11.2023).

23. J. R and C. J. Deep CNN Based Approach for Driver Drowsiness Detection. *IEEE International Power Energy Conference.* 2022. pp. 1-6. URL: https://ieeexplore.ieee.org/document/10059547 (дата звернення: 22.11.2023).

24. How Is Python Used in Machine Learning? URL: https://builtin.com/ /python-machine-learning (дата звернення: 22.11.2023).

25. What is OpenCV? The Complete Guide (2023). URL: https://viso.ai/computer-vision/opencv/ (дата звернення: 24.11.2023).

26. Jupyter Notebook: Introduction. URL: https://realpython.com/jupyter-notebook-introduction/ (дата звернення: 24.11.2023).

27. PyCharm: all about the most popular Python IDE. URL: https://hackr.io/blog/what-is-pycharm (дата звернення: 26.11.2023).

28. MediaPipe Framework for ML Solutions Explained. URL: https://viso.ai/computer-vision/mediapipe (дата звернення: 24.11.2023).

29. Streamlit Components API Reference. URL: https://docs.streamlit.io/ /components/components-api (дата звернення: 28.11.2023).

30. Mediapipe Tasks and Solutions. URL: https://www.opencv.ai/blog/look-into-mediapipe-tasks-and-solutions-with-python (дата звернення: 28.11.2023).

ПРЕЗЕНТАЦІЯ

Державний університет інформаційно-комунікаційних технологій

Кафедра Інженерії програмного забезпечення автоматизованих систем

КВАЛІФІКАЦІЙНА РОБОТА
на тему:
“Інтелектуальна система виявлення та запобігання сонливості водія за кермом”

на здобуття освітнього ступеня магістра
зі спеціальності 126 Інформаційні системи та технології
освітньо-професійної програми Інформаційні системи та технології

Виконав: здобувач вищої освіти гр. ІСДМ-61
Владислав ПОСИНЯК
Керівник: старший викладач кафедри
Віра Миколайчук

Київ - 2023

- **Актуальність теми:** Втома, побічні ефекти ліків, розлади сну, вживання алкоголю та змінний робочий графік є основними причинами сонливості, що може несподівано вплинути на водіїв, зокрема на тих, хто керує вантажівками та автобусами на довгі дистанції. Впровадження технологій моніторингу поведінки водіїв та алгоритмів машинного навчання для виявлення сонливості є ключовими у запобіганні аварій, пов'язаних з втомою водіїв. Актуальність кваліфікаційної роботи магістра зумовлена недосконалістю сучасних технологій виявлення та запобігання аварійним ситуаціям під час керування транспортним засобом, які пов’язані з втомленістю водія за кермом.
- **Об`єкт дослідження:** процес розпізнавання втоми водія на даних, отриманих вбудованою камерою.
- **Предмет дослідження:** методи та програмні засоби, які дозволяють виявляти, аналізувати та оцінювати ознаки втомленості або сонливості водія під час керування транспортним засобом.
- **Мета дослідження:** дослідження методів та розробка програмної системи, яка зможе автоматично виявляти ознаки втомленості або сонливості водія під час керування транспортним засобом.

2

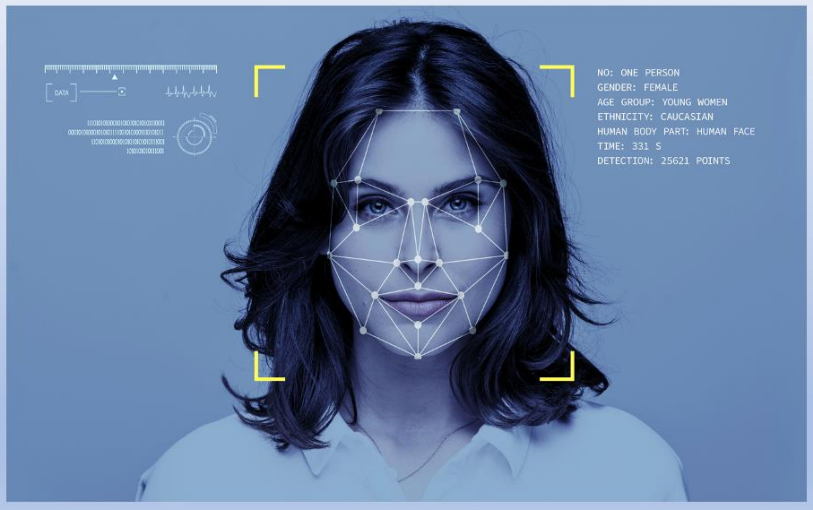
ПРИКЛАД ІСНУЮЧИХ СИСТЕМ

Audi Rest Recommendation

Volvo Driver Alert Control

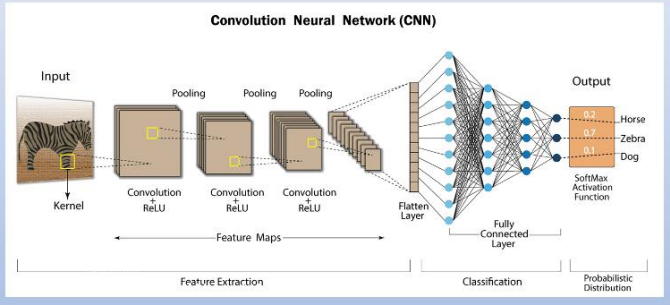
3

КОНЦЕПЦІЯ РОЗПІЗНАВАННЯ ОБЛИЧЧЯ

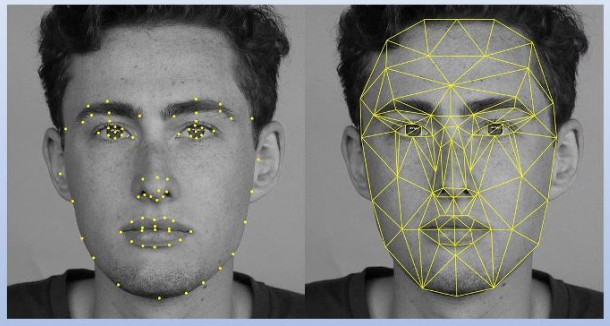


4

ПРИКЛАД ІСНУЮЧИХ МЕТОДІВ



Типова архітектура згорткової нейронної мережі для використання в сфері класифікації зображень

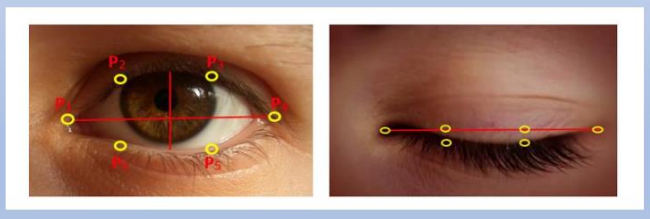


Концепція використання графів на основі антропометричних точок обличчя для розпізнавання обличчя

5

ВИКОРИСТАННЯ ПІДХОДУ ВИЯВЛЕННЯ СОНЛИВОСТІ НА ОСНОВІ EAR

Для виявлення втомленості водія використовується техніка визначення співвідношення сторін очей.
Формула EAR повертає скалярну величину, яка відображає рівень відкриття очей.



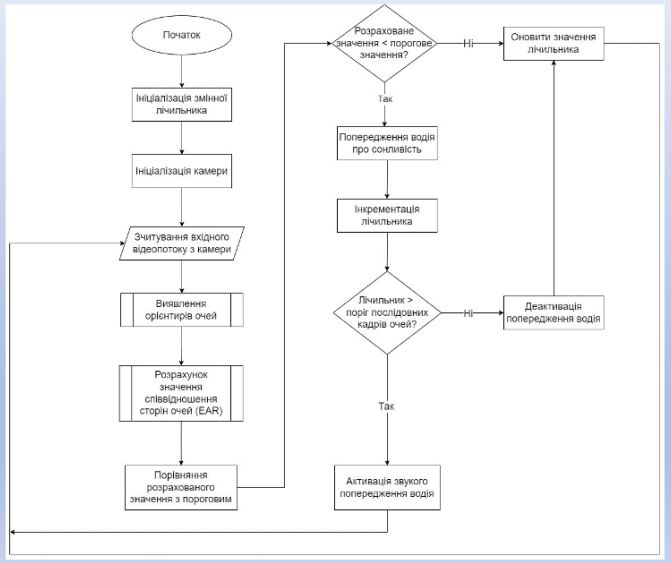
$$EAR_L = \frac{\left||p_2^L - p_6^L|\right| + ||p_3^L - p_5^L||}{2 * ||p_1^L - p_4^L||}$$

$$EAR_R = \frac{\left||p_2^R - p_6^R|\right| + ||p_3^R - p_5^R||}{2 * ||p_1^R - p_4^R||}$$

$$\text{Середнє EAR} = \frac{EAR_L + EAR_R}{2}$$

6

БЛОК-СХЕМА АЛГОРИТМУ ВИЯВЛЕННЯ ТА ЗАПОБІГАННЯ СОНЛИВОСТІ ВОДІЯ



7

ВИКОРИСТАНІ ТЕХНОЛОГІЇ ДЛЯ РОЗРОБКИ ПРОГРАМНОЇ СИСТЕМИ

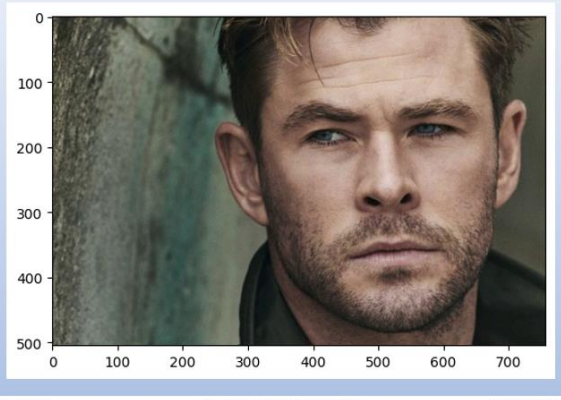
OpenCV

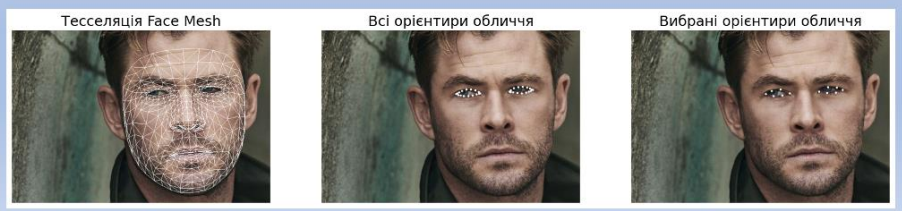
NumPy

MediaPipe

8

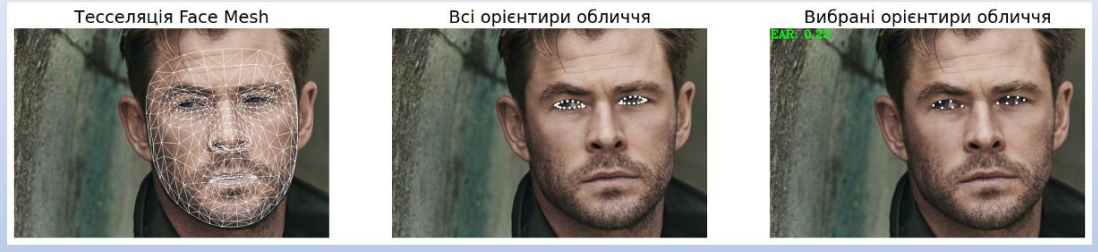
ПРИКЛАД ВИЯВЛЕННЯ ОРІЄНТИРІВ ПРАВОГО ТА ЛІВОГО ОКА





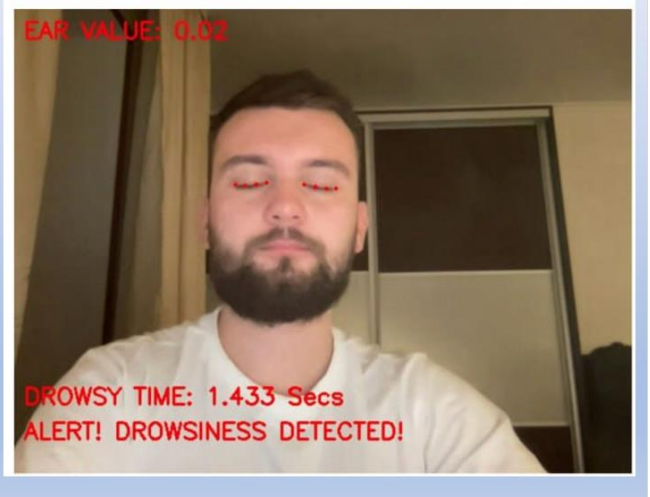
9

РОЗРАХУНОК EAR НА ОСНОВІ ВИЗНАЧЕНИХ ОРІЄНТИРІВ ЛІВОГО ТА ПРАВОГО ОКА



10

ПРОГРАМНА СИСТЕМА ВИЯВЛЕННЯ ТА ЗАПОБІГАННЯ ВТОМЛЕНОСТІ ВОДІЯ



11

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

- У кваліфікаційній магістерській роботі досліджено існуючі системи виявлення та запобігання втомленості. Також, розглянуті існуючі методи виявлення та запобігання втомленості водія.
- Спроєктовано та розроблено програмну систему для виявлення та запобігання втомленості водія.
- Використання функції EAR для виявлення моргання має кілька переваг перед традиційними методами обробки зображень. У звичайних підходах першим кроком є локалізація очей на зображенні, а потім порогове визначення для ідентифікації зіниць очей. Згодом миготіння виявляється шляхом моніторингу зникнення білої області. Натомість використана функція EAR базується на обчисленні співвідношення відстаней між конкретними орієнтирами обличчя, пов’язаними з очима водія.
- Для підвищення ефективності системи виявлення сонливості водія, пропонується використовувати додаткові технології та механізми. Наприклад, інтелектуальна система управління кліматом в салоні автомобіля, яка автоматично регулює температуру, може допомогти водієві залишатися пильним, знижуючи температуру для зменшення відчуття втоми. Системи адаптивнного освітлення може бути корисним доповненням, змінюючи інтенсивність та колір освітлення для зниження сонливості водія.

12

Апробація результатів дослідження:

1. Посиняк В.Ю. «Інтелектуальна система виявлення та запобігання сонливості водія за кермом». Тези доповіді на CXXXIII Міжнародній інтернет — конференції «РОЗВИТОК НАУКИ ТА ТЕХНІКИ УКРАЇНИ ПІД ЧАС ВОЄННОГО СТАНУ». – Київ, 8 грудня, 2023 р.

ДЯКУЮ ЗА УВАГУ!