НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Навчально-науковий інститут прикладного системного аналізу
Кафедра системного проектування

До захисту допущено:

Завідувач кафедри

_____ Вадим МУХІН

«___»_____2022 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою

“Інтелектуальні сервіс-орієнтовані розподілені обчислювання”

зі спеціальності 122 "Комп'ютерні науки"

на тему: « Аналіз алгоритмів контролю втоми та зосередженості користувача, з використанням засобів комп'ютерного зору **»**

Виконав:
студент IV курсу, групи ДА-82
Скицюк Станіслав Андрійович _____

Керівник:
доцент, к.т.н., с.н.с.
Кирюша Богдан Анатолійович _____

Консультант з економічного розділу:
доцент, к.е.н.
Рощина Надія Василівна _____

Рецензент:
Посада, науковий ступінь, вчене звання,
Прізвище, ім’я, по батькові _____

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

Київ – 2022

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Навчально-науковий інститут прикладного системного аналізу

Кафедра системного проектування

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 122 "Комп'ютерні науки"

Освітньо-професійна програма – "Інтелектуальні сервіс-орієнтовані розподілені обчислювання"

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Вадим МУХІН

«___»_____2022 р.

ЗАВДАННЯ

на дипломну роботу студенту

Скицюк Станіслав Андрійович

1. Тема роботи «», керівник роботи Кирюша Богдан Анатолійович , доцент, затверджений наказом по університету від «___»_____ 20__ р. №_____

2. Термін подання студентом роботи – .
3. Вихідні дані до роботи

4. Зміст роботи

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо)

6. Консультанти розділів роботи[1]

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Рощина Н.В.		

7. Дата видачі завдання

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання		
2	Ознайомлення з використанням технологій		
3	Аналіз наявних методі та рішень у сусідніх сферах		
4	Розробка рішень там методів		
5	Створення тестового додатку		
6	Аналіз роботи додатку		
7	Підведення висновків		

Студент Скицюк С.А.

Керівник Кирюша Б.А.

АНОТАЦІЯ

В даній роботі досліджено 3 різні алгоритми визначення втоми за допомогою засобів комп'ютерного зору. Для захоплення відео та обличчя на ньому використовувалися бібліотеки OpenCV і Dlib які є у вільному доступі. Було ро-зроблено додаток для визначення втоми на мові Python. Цей додаток було протестовано на власнозібраному датасеті. В результаті тестування було встановлено приблизну точність алгоритму і фактори які можуть мати значний вплив на неї.

ABSTRACT

This paper investigates 3 different algorithms for determining fatigue using computer vision. OpenCV and Dlib libraries, which are freely available, were used to capture video and faces. An application was made to determine fatigue in Python. This application was tested on a self-assembled dataset. As a result of testing, the approximate accuracy of the algorithm and factors that may have a significant impact on it were established.

ЗМІСТ

ВСТУП

Втома - це не просто відчуття втоми чи сонливості. Термін також охоплює відсутність енергії, а також психічне та фізичне виснаження. І хоча втома відчувається як звичайне захворювання, її не слід розглядати як неминучий факт. Втома значно збільшує кількість щоденних помилок та випадків безпеки, які трапляються на робочих місцях через порушення психічних та фізичних здібностей та зменшують координацію.

Втома на робочому місці може призвести до уповільнення реакції, зменшення пильності, зменшення здатності приймати рішення, відволікати увагу під час складних завдань та втрати обізнаності в критичних ситуаціях.

Незважаючи на те, що втому можна знайти частіше за допомогою змінної роботи, вона також має вплив на працівників з постійним графіком роботи, що може завдати шкоди їхньому здоров’ю, впливати на продуктивність праці та збільшити частоту помилок та інцидентів безпеки.

Існує три принципово різні методи визначення втоми:

1) Підхід, заснований на психології, який, як правило, покладається на психометричні анкети для оцінки рівня втоми людини.

2) Фізіологічний підхід, який відстежує фізіологічні сигнали(ЕЕГ, ЕКГ тощо), пов'язані з втомою

3) Підхід на основі відео, який зазвичай контролює поведінковий та фізичний статус водія, проводить оцінку таких параметрів як риси обличчя, положення голови, час реакції, помилки рульового управління, відхилення руху тощо.

Підхід на основі відео є найбільш зручним з точки зору кількості затрачуваних матеріальних і людських ресурсів, тому став найбільш вживаним. Безпосередньо для цієї задачі найкраще підходить використання засобів комп'ютерного зору, так я дана технологія задовольняє усі потреби розробників ПЗ.

Загалом сфера контролю втоми користувачів ПК не є достатньо розвинутою але має сусідню сферу контролю втоми водіїв, яка має велику кількість перевірених і точних методів та комерційних рішень, що можуть бути використання і для користувачів ПК через концептуально однакове визначення явища втоми.

1 Метод з використанням каскадів Хаара [1]

1.1 Каскади Хаара

Виявлення об'єктів за допомогою каскадних класифікаторів на основі примітивів Хаара - це ефективний метод виявлення об'єктів, запропонований Полом Віолою та Майклом Джонсом у своїй роботі "Швидке виявлення об'єктів, використовуючи підсилений каскад простих функцій" у 2001 році. Цей підхід, заснований на машинному навчанні, де а функція каскаду проходить навчання з багатьох позитивних та негативних зображень. Це використовується для виявлення об'єктів у інших зображеннях .

Каскади знаків зазвичай згадуються як основа для побудови систем виділення складних предметів, таких як особи, руки чи інші предмети. У більшості статей цей підхід нерозривно пов'язаний з алгоритмом навчання Adaboost. Сам каскад Хаара - це набір примітивів, для яких вони вважаються колекцією із зображенням. Використовуються найпростіші примітиви, що складаються з прямокутників і мають лише два рівні, +1 та -1. У той же час кожен прямокутник використовується в кілька разів різних розмірів.

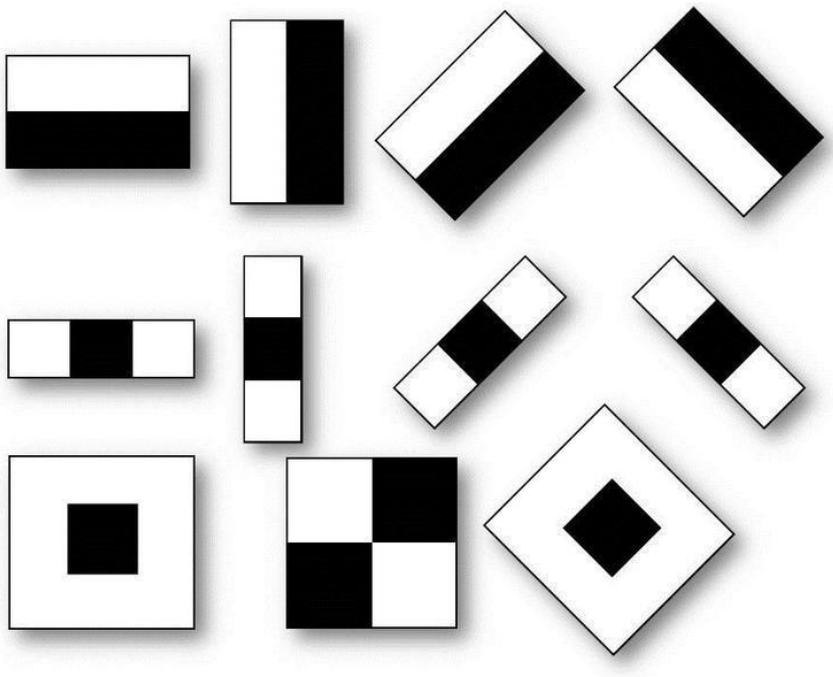

Рисунок 1.1 – Типові примітиви

Під згорткою тут мається на увазі s = x-y, де y - сума елементів зображення в темній області, а x - сума елементів зображення у яскравій області (ви також можете взяти x/y, тоді буде стабільність при зміні масштабу).

Такі згортки підкреслюють структурну інформацію об'єкта: наприклад, наступна згортка завжди буде негативною для центру обличчя людини:

Рисунок 1.2 – Приклад негативної згортки

Очі будуть темнішими, ніж область між ними, так само, як рот буде темнішим за лоб. Чим більше використовуються різні примітиви, тим точніше ви можливо класифікувати об’єкт. Більше того, якщо точна класифікація не потрібна, ви можете використовувати менше примітивів.

Перевага даного методу полягає в тому, що каскади Хаара дуже швидко вираховуються через інтегральне представлення зображень невід'ємним зображенням зображень.

Отже можна чітка виділити такі переваги даного методу:

1) Стійкість до зміни освітлення, навіть якщо це локальна зміна освітлення, стійкість до шуму (примітиви - це простий смужковий фільтр).
2) Якщо примітиви були не дуже малими, то це набагато стабільніше кореляція при зміні масштабу(розмір примітивів не вплине на точність, якщо обходити невеликим кроком).
3) Якщо ознаки на великому зображенні обчислюються заздалегідь і при переміщенні вікна пошуку, щоб взяти вже підраховане та актуальне для нього, пошук буде набагато швидшим, ніж кореляція (вам потрібно порівняти меншу кількість елементів).

1.2 Модель алгоритму

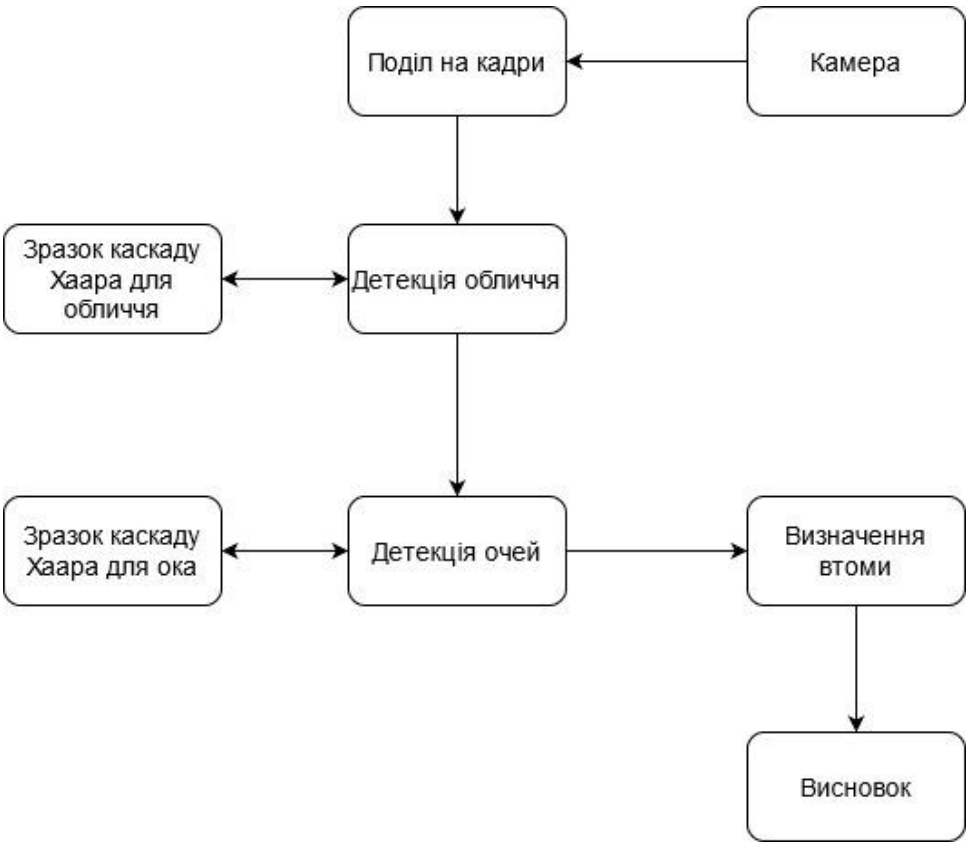

Рисунок 1.3 – Модель алгоритму

Функціонал кожного модулю системи може бути представлений наступним чином:

1.2.1 Захоплення відео: Захоплення відео в основному передбачає отримання живого відео -каналу. Отримання відео досягається, використовуванням камери. Поділення на кадри: цей модуль використовується для того, щоб розділяти потокове відео на серію кадрів-зображень.

1.2.2 Виявлення обличчя: Функція виявлення обличчя забирає один кадр за один раз від наданих кадрів, і в кожному кадрі вонф намагається виявити обличчя користувача. Це досягається шляхом використання набору заздалегідь визначених зразків каскадів Хаара.

1.2.3 Виявлення очей: Після того, як функція виявлення обличчя виявила обличчя автомобільного драйвера, Функція виявлення очей намагається виявити очі користувача . Це досягається використанням набору заздалегідь визначених зразків каскадів Хаара.

1.2.4 Виявлення сонливості: Після виявлення очей функція виявлення сонливості виявляє, чи є користувач ПК сонним, беручи до уваги стан очей, тобто відкритий або закритий і частоту кліпання.

Оскільки запропонована система використовує бібліотеки OPENCV, немає вимоги до необхідного мінімуму роздільної здатності камери.

1.3 Реалізація

1.3.1 Захоплення відео

Бібліотека OPENCV надає велику підтримку для захоплення та обробки відео в прямому ефірі. Також можливо вибрати, чи потрібно відео з вбудованої вебкамери чи зовнішньої камери, встановивши відповідні параметри. Як було сказано раніше, OPENCV не вказує ніякого мінімуму вимога до камери, однак OPENCV за замовчуванням очікує певної роздільної здатності відео, яке записується, якщо роздільні здатності не відповідають, то буде повідомлено про помилку. Цю помилку можна усунути, перевищуючи значення за замовчуванням, чого можна досягти вручну визначенням роздільної здатності запису відео.

1.3.2 Поділ на кадри

Після захоплення відео наступним кроком є розділення його на серію кадрів/зображень. Цей процес проходить в 1 етапи за допомогою однокрокового процесу, коли одна функція захоплює кадр і повертає його шляхом декомпресії.

1.3.3 Захоплення обличчя

Після успішного вилучення кадрів наступним кроком є виявлення обличчя у кожному з цих кадрів. Це досягається за допомогою використання файлу Haarcascade для виявлення обличчя. Файл Haarcascade містить ряд функцій обличчя, таких як висота, ширина та пороги кольорів обличчя. Він побудований за допомогою ряду позитивних та негативних зразків. Для виявлення обличчя, ми спочатку завантажуємо каскадний файл. Потім передаємо набутий кадр до функції виявлення граней, яка виявляє всі можливі об'єкти різних розмірів у кадрі. Щоб пришвидшити обробку, замість виявлення об'єктів усіх можливих розмірів, ми можемо вказати детектору граней значення для виявлення тільки об'єктів певного розміру. Цей розмір визначається на основі файлу Haarcascade. Тепер детектор граней може знайти лице на кадрі порівнюючи власні значення зі значеннями у файлі Haarcascade.

1.3.4 Захоплення ока

Відбувається аналогічно до захоплення обличчя, тільки використовується файл Haarcascade для ока, в якому вказані додаткові параметри такі як: положення очей в верхній частині обличчя, тощо.

1.3.5 Визначення втоми

Після виявлення очей наступним кроком є визначення, чи очі в закритому чи відкритому стані. Це досягається шляхом вилучення значень пікселів з області очей. Після вилучення ми перевіряємо, чи ці значення пікселів білі, якщо вони білі, то це спричиняє, що очі знаходяться у відкритому стані, якщо значення пікселів не білі, тоді це викликає, що очі в закритому стані.

Це робиться для кожного вилученого кадру. Якщо очі являються закритими дві секунди або певну кількість послідовних кадрів, залежну від швидкості кадрів, то це свідчить про сонливість користувача ПК. Якщо очі фіксуються закритими у непослідовні кадри, тоді ми вважаємо це кліпанням очима.

Якщо виявлено сонливість, відображається текстове повідомлення разом із запуском звукової тривоги.

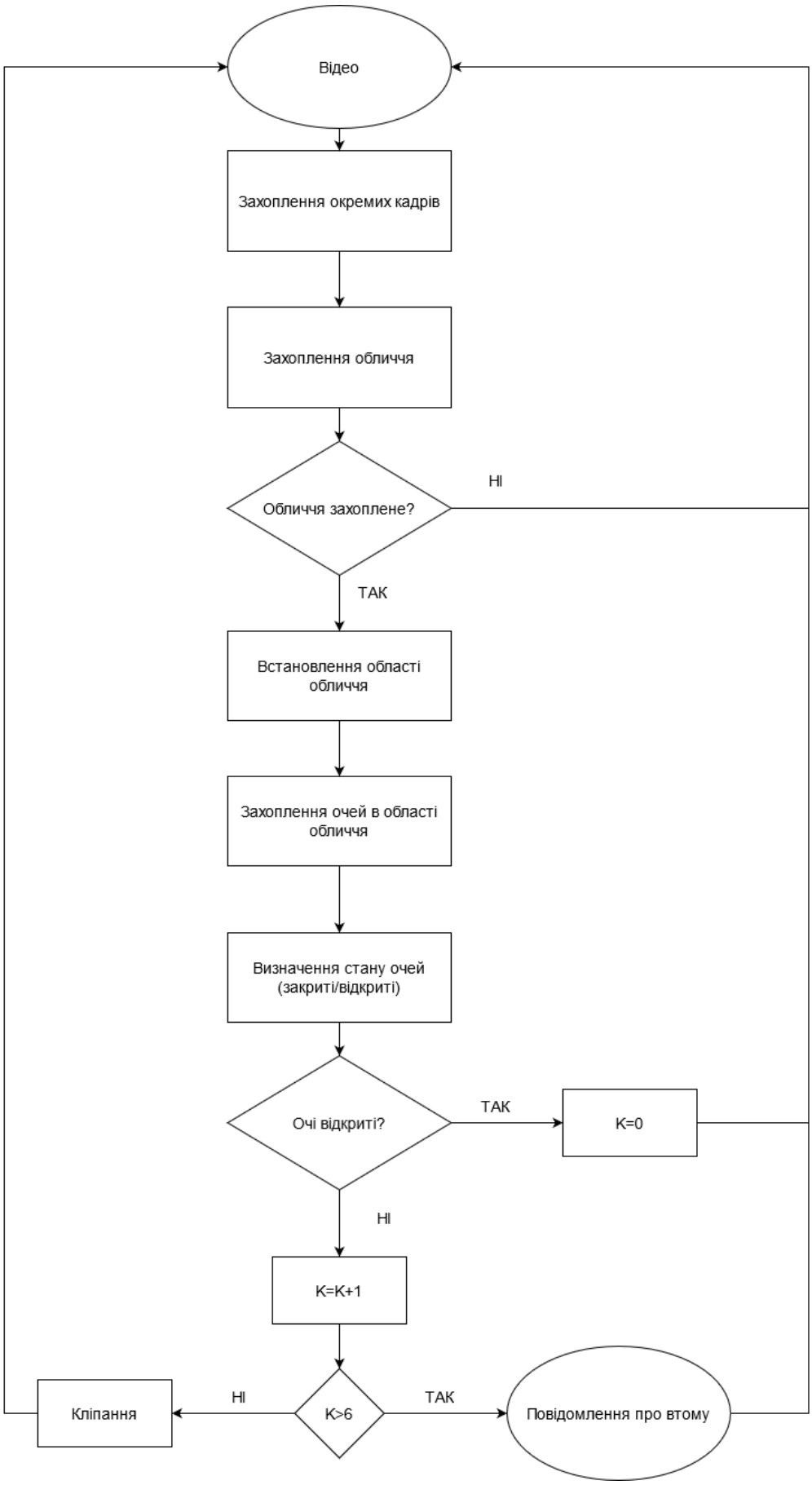
Відео
Захоплення окремих кадрів
Захоплення обличчя
Обличчя захоплене?
НІ
ТАК
Встановлення області обличчя
Захоплення очей в області обличчя
Визначення стану очей (закриті/відкриті)
Очі відкриті?
ТАК
K=0
НІ
K=K+1
K>6
НІ
ТАК
Кліпання
Повідомлення про втому

Рисунок 1.4 – Блок-схема алгоритму

1.4 Висновки до розділу 1

Каскади Хаара э зручним методом виявлення будь-якого предмету на зображенні. Через те що в основі них лежить чорно-білі примітиви даний метод не є чутливим до якості освітлення шуканого предмету на зображенні. Використані каскади мають документовану точність виявлення, але за необхідності поліпшення існує можливість розробки власних каскадів.

Метод розроблений в даному розділі передбачає використання вбудованої веб-камери на пристрої, тому можливий для імплементації майже на будь-який пристрій. В основі лежить виявлення обличчя та очей з визначеним станом на ньому і встановлення рівню втоми користувача. Перевагою цього методу є висока швидкодія і стійкість до зміни масштабу зображення, при достатньо невеликих за розмірами примітивах.

2 Метод з використанням Perclos та EAR[2]

Даний метод базується на визначенні двох ключових параметрів на обличчі, а саме Perclos, що відповідає часу, який людина витрачає на закриття ока і EAR, який позначає степінь відкритості ока. Одночасне використання цих параметрів дозволяє сильно підвищити загальну точність роботи алгоритму.

2.1 Модель алгоритму

Алгоритм складається з двох наступних частин:

1) Оффлайн навчання:

Стан втоми користувача ПК, як правило, визнається ознаками обличчя, такими як відкриття та закриття очей та ступінь відкриття рота при позіханні. Спочатку застосовується алгоритм глибокої згорткової мережі (TCDCN) для розпізнавання обличчя користувача та очей на ньому для кожного зібраного зображення. Для цього використовується 68-точкова модель обличчя яку створює нейронна мережа при захопленні обличчя.

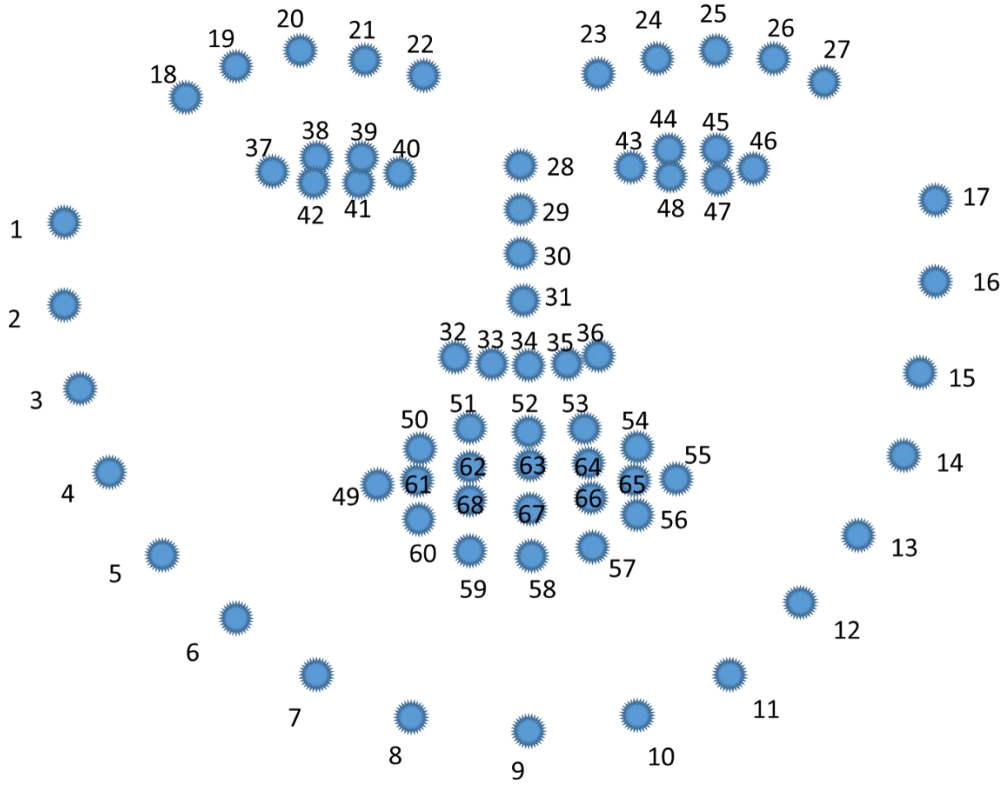

Рисунок 2.1 – 68-точкова модель обличчя

Перевага цього алгоритм полягає в тому, щоб провести навчання з декількома ознаками (включаючи стать, носіння окулярів або сонцезахисних окулярів) одночасно. Ці допоміжні атрибути можуть допомогти краще знаходити обличчя на кожному кадрі.

2) Онлайн тестування:

Ця частина алгоритму в режимі реального часу для виявлення втоми водія з прямого відео. Усі відеозображення імпортуються в TCDCN, щоб розпізнати обличчя користувача, потім орієнтовні орієнтири отримують в режимі реального часу [31]. Потім навчений офлайн класифікатор Adaboost використовується для визначення того, чи відкриті очі та рот користувача ПК. Нарешті, ми класифікуємо, чи користувач є сонним чи ні відповідно до умови, що середня тривалість часу закриття (або відкриття роту) перевищує вибраний поріг за певний проміжок часу. Крім того, для ідентифікації втоми користувача більш точно, TCDCN може усунути вплив носіння окулярів, постави голови та інших факторів. Після захоплення обличчя ми можемо визначити такі параметри як Perclos1 і Perclos2, що відповідають стану закритих і відкритих очей.

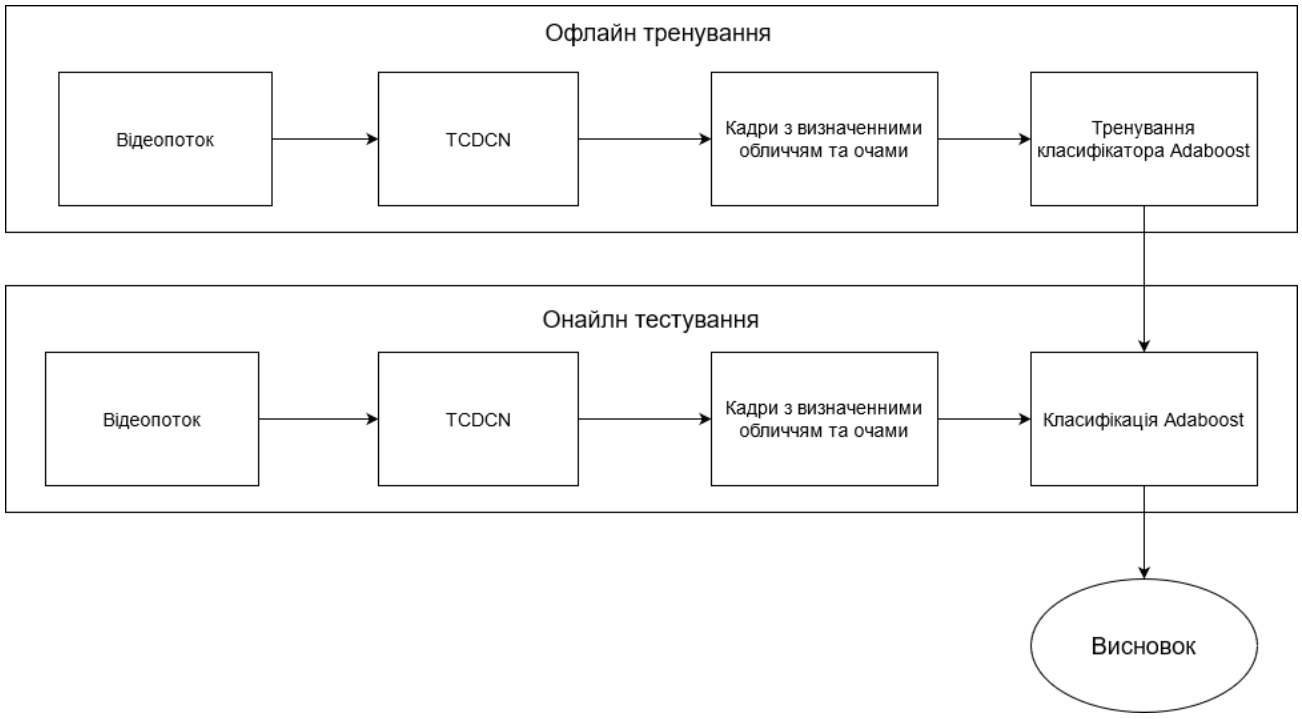

Рисунок 2.2 – Модель алгоритму

2.2 Реалізація

2.2.1 TDCN

Діаграма алгоритму TCDCN показана на малюнку . На вхід постачається сіре зображення розміром 40 40. Особливість функцій включає чотири шари згортки , три шари асоціації та один шар повного з'єднання. Вага фільтра не ділиться в просторі, а це означає, що на вхідній карті використовуються різні набори фільтрів. Гіперболічна дотична випрямлення абсолютного значення була обрана як функція активації; Максимальна асоціація проводиться в області без перекриття, а власний вектор генерується за допомогою повного шару з'єднання після чотирьох пучків. Цей власний вектор розділений між кількома завданнями на етапі оцінки, серед яких лінійна регресія використовується для отримання розташування орієнтирів, і багаторазова регресія логістики використовується для обробки багатьох інших особистих завдань, наприклад, статі, положення обличчя, позиції, положення обличчя, тощо.

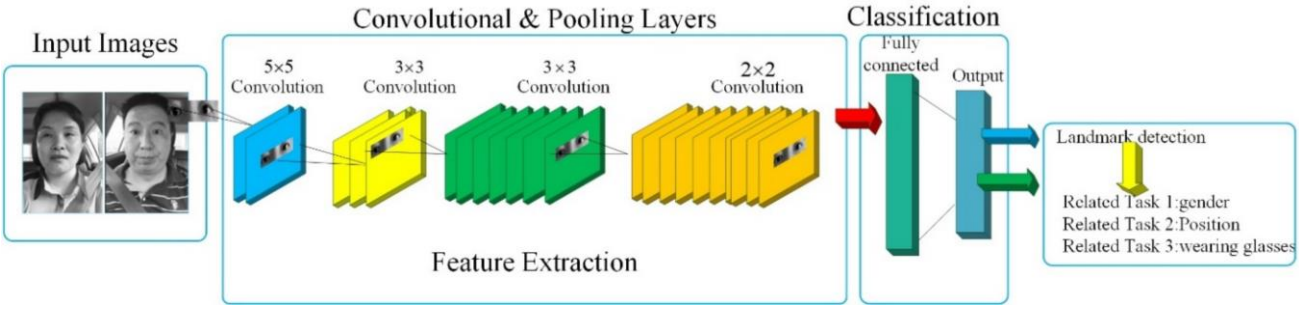

Рисунок 2.3 – Схема алгоритму

2.2.2 Архітектура TDCN

Взагалі кажучи, коли люди дивляться на обличчя на фотографії, вони визначають, де знаходиться обличчя, а потім визначають стать, положення обличчя, чи має водій окуляри тощо. Однак TCDCN може розробити складну мережу для виконання всіх цих завдань одночасно та використовувати взаємозв'язок між завданнями. Складність багатозадачності навчання полягає в тому, що різні завдання мають різні характеристики та швидкість збіжності. Метод, запропонований у цьому алгоритми , перевищує існуючі методи, особливо при роботі з ситуацією серйозної оклюзії та змін у положенні, а також зменшує складність моделі.

Для того, щоб використовувати всю інформацію з багатозадачних наборів даних, таких як людина, точки підтримки обличчя, положення обличчя, стать та інформацію про носіння окулярів, кілька підмереж можуть навчити набори даних, пов'язані з завданням та потім спільно використовувати параметри, оскільки жоден набір даних не містить усієї інформації про анотацію, необхідну для завдання виявлення обличчя. Завдяки цьому методу, TCDCN може використовувати та адаптувати спільне використання параметрів для всієї області, а не для певної області завдання.

На етапі тренувань TCDCN як дані про тренування використовуються багатозадачні дані на орієнтирах обличчя (MTFL) . Цей набір даних містить 12 995 зображень осіб, зібраних з Інтернету. Зображення анотуються (1) п'ятьма орієнтиром на обличчі та (2) атрибутами статі, наявності окуляр та пози голови. Процедура навчання TCDCN спрямована на отримання оптимальної моделі за допомогою динамічних налаштувань параметрів. Щоб виразити вплив різних параметрів на продуктивність TCDCN, в процесі навчання вводиться функція втрат. У цій роботі метод втрат для основного завдання використовує метод найменших

квадратів, а також для допоміжного завдання - функції втрат перехресної ентропії. Функція втрат для основного завдання полягає в наступному:

$$\mathrm{L}(\mathrm{y}, \mathrm{f}(\mathrm{x})) = \sum_{\mathrm{i}=1}^{\mathrm{N}} \left(y_i - \mathrm{f}(x_i)\right)^2$$

Де, L - функція втрат, i - перший зразок, n - загальна кількість зразків, y - це очікуєме значення, а f (x) - істинне значення.

Функція втрат перехресної ентропії для допоміжного завдання (стать, положення обличчя та окуляри) така:

$$\mathrm{L}\big(\mathrm{Y}, \mathrm{p}(\mathrm{Y}|\mathrm{X})\big) = -\log \mathrm{p}(\mathrm{Y}|\mathrm{X})$$

Де функція втрат L означає, що ймовірність P (Y|X) зразка з вибірки X досягає максимального значення, за умови приналежності до вибірки Y.

Завдяки тренуванню TCDCN, обличчя користувача ПК можна отримати точно, що забезпечує стабільне зображення обличчя для наступних алгоритмів.

2.2.3 68-точкова модель захоплення обличчя

У цій роботі втома визначається за станом очей водія. Для цього важливо отримати орієнтири та форму очей і рота. DLIB - це бібліотека з набором інструментів з відкритим кодом для алгоритмів машинного навчання та визначення орієнтирів на обличчі. У цьому алгоритмі, використовується 68-точкову модель DLIB, які калібруються та зображуються за допомогою OPENCV. Як було сказано вище, TCDCN використовує, пов’язані з обличчям атрибути для вивчення розташування особливих точок обличчя. Завдяки такій багатозадачності, алгоритм може підвищити надійність виявлення характерних точок обличчя. Зокрема, це необхі-

дно для навчання визначення декількох ознак (включаючи стать, одягання окуляpів та позу обличчя) одночасно. Ці допоміжні атрибути можуть допомогти краще визначити розташування характерних ознак втоми на обличчі.

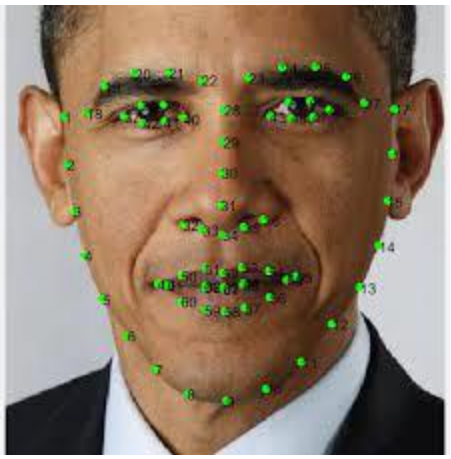

Рисунок 2.4 – Приклад захоплення обличчя 68-точковою схемою

2.2.4 Ознаки втоми користувача

Після огляду відповідної літератури встановлено дві загальні ознаки втоми людини: позіхання (рот є широко відкритим і залишається в цьому стані протягом відносно тривалого часу) і сповільнене кліпання . На основі цих даних концентрація уваги і втомленість користувачів може бути обчислена в режимі реального часу.

2.2.5 EAR

Концептуальна модель використовує відстежування кліпання очима в режимі поточного часу. Однак результати виявлення можуть бути обмежені наступними фактами:

1) Очевидно, що EAR(співвідношення сторін очей) сильно коливається під час носіння окулярів

2) Роздільна здатність сильно впливає на точність і швидкодію алгоритму.

Чим нижча кількість пікселів, тим швидше проходить процес обробки, але в той же час також необхідна певна роздільна здатність для поліпшення якості розпізнавання. Як було сказано вище, запропонований метод у цій роботі відрізняється від традиційного методу обчислення кліпання на зображенні . Використання EAR з TCDCN є більш надійним рішенням незалежно від того, носить користувач окуляри чи ні. Незалежно від того, чи носить користувач окуляри чи ні, схема може точно визначити функцію точок очей і застосувати дуже простий метод обчислення, заснований на співвідношенні відстані між точками очей для виявлення втоми користувача .

$$\mathrm{EAR} = \frac{\|\mathrm{p2} - \mathrm{p6}\| + \|\mathrm{p3} - \mathrm{p5}\|}{2\|\mathrm{p1} - \mathrm{p4}\|}$$

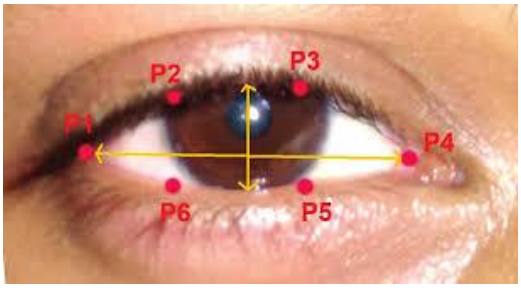

Рисунок 2.5 – Умовні точки на оці для визначення EAR

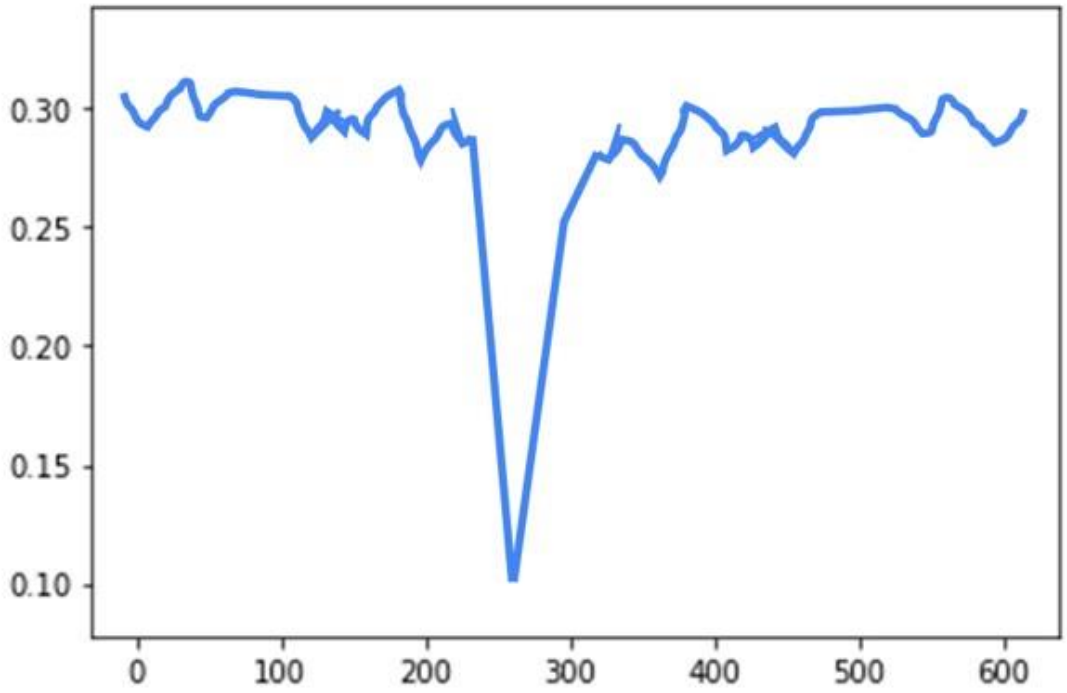

Рисунок 2.5 – Графік EAR відносно часу

Значення ординат на графіку - це значення EAR, що відповідають різним моментам часу. Значення EAR швидко падає до нуля, а потім швидко піднімається, що означає, що користувач блимає один раз. Коли очі користувача відкриті, EAR перевищує 0,25 , але коли EAR менше 0,25 , це означає, що очі користувача закрилися через втому . Тому значення EAR можуть бути використані , щоб виявити, чи є користувач втомленим.

2.2.6 Perclos

Відсоток закриття повік над зіницею з часом (Perclos) визначається як ступінь закриття очей протягом певного часу. Це ефективний метод вимірювання стану втоми . Perclos судить втому, що керує часом, часткою закриття очей. Тому що

Розмір очей варіюється від людини до людини, область очей динамічно змінюється через вплив оточуючого середовища та руху голови. В даному алгоритмі граничним значенням Perclos було встановлено – 0.8. Якщо значення Perclos перевищує 0.8, то можна стверджувати що користувач втомився.

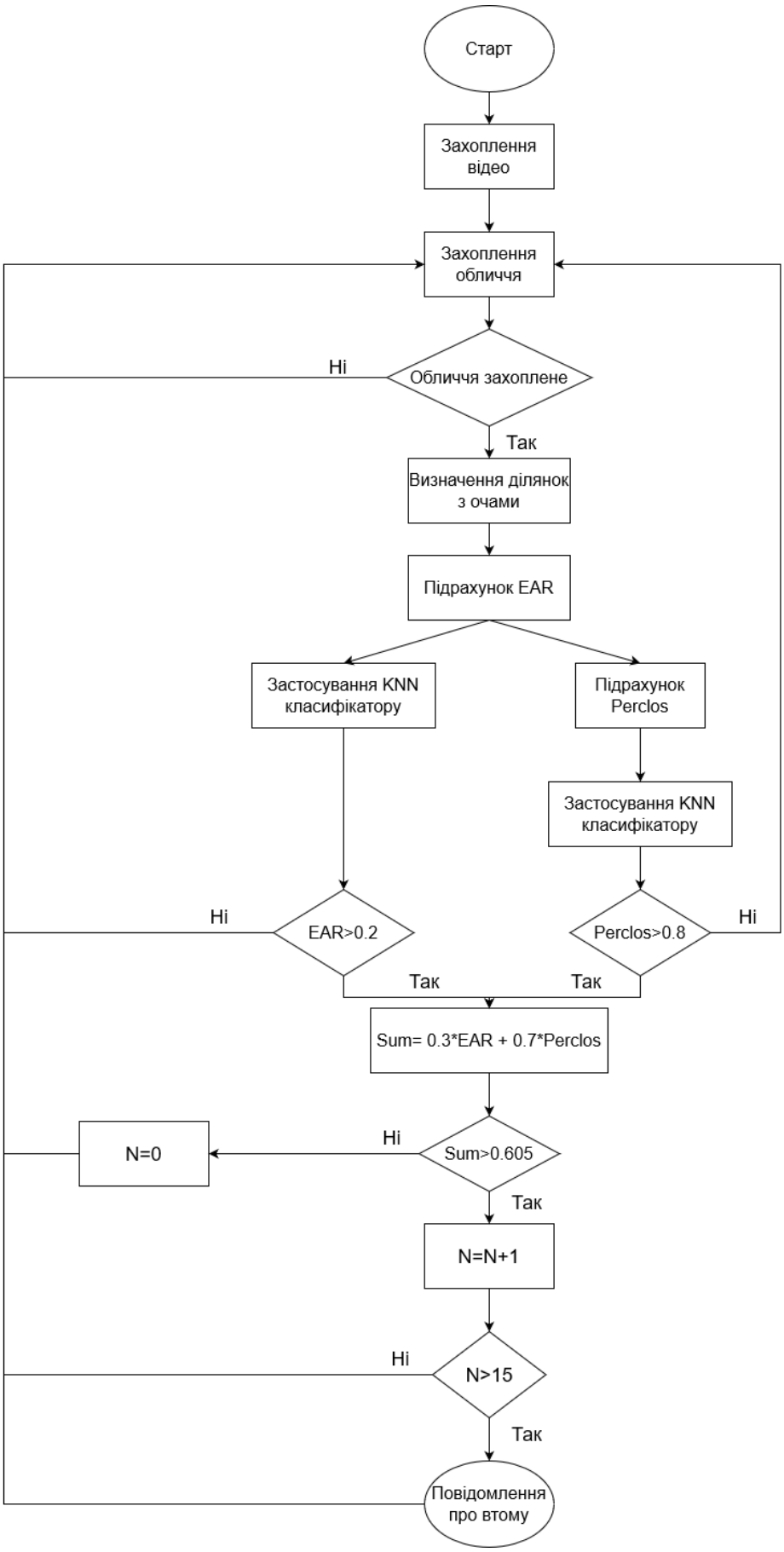
Старт
Захоплення відео
Захоплення обличчя
Обличчя захоплене
Ні
Так
Визначення ділянок з очами
Підрахунок EAR
Застосування KNN класифікатору
Підрахунок Perclos
Застосування KNN класифікатору
EAR>0.2
Perclos>0.8
Ні
Ні
Так
Так
Sum= 0.3*EAR + 0.7*Perclos
Sum>0.605
Ні
N=0
Так
N=N+1
N>15
Ні
Так
Повідомлення про втому

Рисунок 2.6 – Блок-схема алгоритму

2.3 Висновки до розділу 2

В цьому розділі було розроблено алгоритм контролю стану втоми користувача ПК за допомогою 68-точкової моделі обличчя. Для захоплення обличчя було використано нейронну мережу TDCN. Основною перевагою її використання я можливість обрахунків в режимі багатозадачності, що значно прискорю процес виявлення обличчя.

Для визначення втоми використовуються такі параметри як Ear та Perclos. Ці параметри достатньо точно описують стан втоми користувача через контроль степені розпахнутості ока та довжини блимання. Для покращення точності алгоритму було одночасно використано ці два параметри. Перевагою цього методу є точне визначення стану користувача за рахунок контролю декількох параметрів. З недоліків можна зазначити що алгоритм погану працює з відео в якому було погане освітлення. Дану проблему можливо вирішити встановленням більш якісної камери з інфрачервоним підсвітленням.

3 Метод з використанням глибинного навчання[3]

3.1 Модель алгоритму

Алгоритм проходить в 2 кроки, а саме:

1) Захоплення обличчя з відеозапису першою нейронною мережею

2) Визначення стану втоми другою нейронною мережею

3.2 Реалізація

В даному методі втома визначається виявленням певних ознак на зображенні.

Зображення отримуються шляхом поділення вхідного відеопотоку на кадри.

У алгоритмі використовується декілька різних нейронних мереж, а саме:

1) SSD для захоплення обличчя на зображенні
2) VGG16 для визначення статусу втоми користувача ПК

3.2.1 Захоплення обличчя

В частині алгоритму, де відбувається виявлення обличчя було використано нейронну мережу Single Shot Multi-Box Detextor (SSD). Традиційний засіб виявлення обличчя заснований на машинному навчанні, який використовує каскадні, Adaboost та інші стратегії. Це метод має невелику кількість розрахунків, хороші показники в режимі реального часу, але низьку точність.

$$f(x) = \text{sign}(w * x) + b \qquad (4)$$

Алгоритм виявлення обличчя на основі глибокого навчання через значну обчислювальну складність структури згортокової нейронної мережі має достатньо високий рівень точності , але ефективність роботи в режимі реального часу потребує покращення. SSD є абсолютно новим засобом для виявленням цілей на зображенні. Його характерними ознаками є :

1) Використання карт різних масштабів: карти великих ознак використовуються для виявлення малих об'єктів, а карти малих ознак користовуються для виявлення великих об'єктів;
2) Використання апріорних кадрів з різними масштабами та співвідношеннями сторін.. Цей алгоритм використовує мережу VGG16 як основу , повністю замінюючи підключений шар vgg16 згортковим шаром і додає низку

згорткових шарів для отримання більшої кількості ознак для виявлення шуканих ознак.

Мережа SSD загалом включає 11 блоків. Четвертий шар п’ятого блоку VGG16 змінюється, а повністю з'єднані шари FC6 та FC7 VGG16 замінюються на 3 × 3 згорткові шари та 1 × 1 згорткові шари. І вилучений шар і шар FC8 видалено, і додано новий згортковий шар. Мережева структура SSD показана на малюнку .

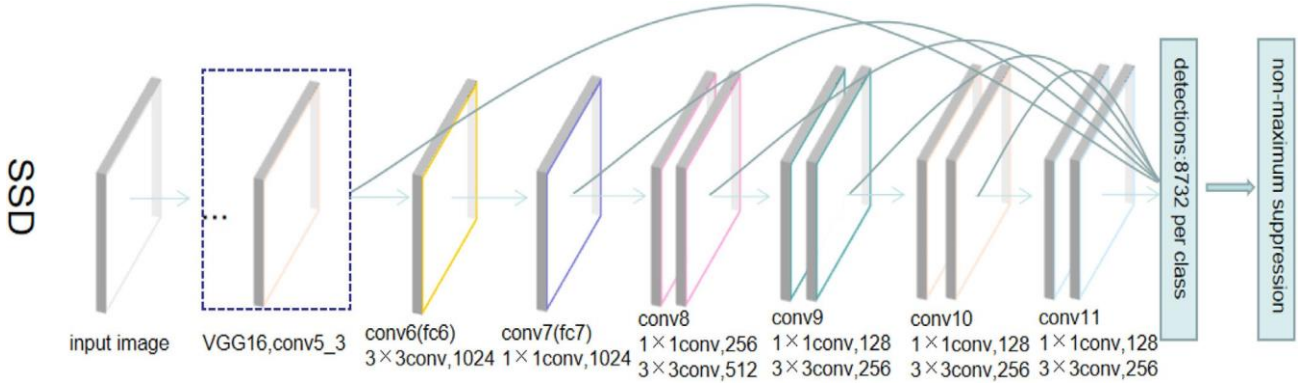

Рисунок 3.1 – Архітектура мережі SSD

Крім того, описаний згортковий шар також включає функції активації та нормалізацію. RELU- це типова функція нелінійної активації, а її формула- f (x) = max (0, x), яка відображає вхідний сигнал на простір функції. По-перше, порівняно з традиційною функцією активації сигмоїдів, яка потребує обчислення показника та зворотного, RELU має нижчу обчислювальну вартість та більшу швидкість. По-друге, функція RELU є розрідженою. Тобто вона не активується, коли на вхід подається значення менше 0. Порівняно з функціонуванням активації сигмоїдів зі швидкістю активації близько 50%, може бути отримана нижча швид-

кість активації, що ближче до імітації процесу транспортування біологічного сигналу, під час роботи мозку. Це суттєво впливає на полегшення проблеми надмірного пристосування мережі. Оскільки Conv4_3 у VGG16 використовується для виявлення першої карти ознак, розмір карти функцій становить 38 × 38. Цей шар знаходиться відносно спереду, тому нормалізація додається ззаду, і кожен піксель нормалізується в просторі каналу для того, щоб різниця від наступного згорткового шару не була невеликою. Для оптимізації використовується імпульсний алгоритм. Алгоритм імпульсу є вдосконаленням локальної проблеми вібрації мінімальної точки в просторі оптимізації стохастичного градієнта. Ітеративна формула оновлення алгоритму імпульсу показана в рівнянні (4a). β - показник накопичення градієнта, і значення було встановлено на 0,9. α - рівень навчання мережі. dw- це початковий градієнт, який ми наступили, а v- градієнт, обчислений експоненціально зваженим середнім значенням . Це еквівалентно згладжуванню оригінального градієнта, а потім використання його для градієнтного спуску. Коли ми використовуємо алгоритм оптимізації імпульсу, ми можемо вирішити проблему надмірного розгойдування в діапазоні оновлень алгоритму оптимізації стохастичного градієнта, і в той же час зробити конвергенцію мережі швидше.

$$v = \beta v + (1 - \beta)dw \qquad (4a)$$

$$w = w - \alpha v. \qquad (4b)$$

3.2.2 Визначення стану втоми

Спочатку отримане відео ділиться на кадрии, ці кадри подаються до мережі SSD для виявлення обличчя, а потім зображення з визначеними на них обличчям передаються на вхід до VGG16, яка визначає стан втоми користувача. Тренування

мережі відбувається на готовому розміченому датасеті. VGG16 - це проста згорткова нейрона мережа. Вона складається з декількох 3 × 3 згорткових шарів та 2 × 2 об'єднануючих шарів, що неодноразово складено. VGG16 містить загалом 13 шарів, 3 повністю з'єднані шари та 5 шарів об'єднання.

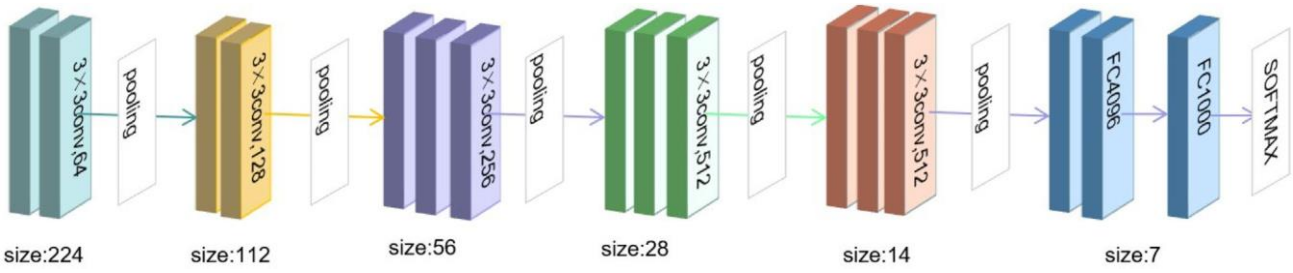

Рисунок 3.1 – Архітектура мережі VGG16

3.2 Висновок до розділу 3

В цьому розділі було розроблено метод з використанням глибинного навчання. Алгоритм проходить у 2 кроки, а саме захоплення обличчя першою нейронною мережею і визначення втоми другою мережою

Для визначення положення обличчя було використано мережу SSD. Визначення втоми відбувається за рахунок VGG16.

Цей метод є найбільш перспективним, оскільки у сучасному світі найбільш високої точності зазвичай досягається шляхом використанням машинного навчання. Крім того точність методу буде поліпшуватись з подальшим модифікаціями та вдосконаленнями нейронних мереж і збільшенням матеріалів для тренування.

4 Розробка програмного забезпечення з метою перевірки точності визначення стану втоми користувача.

4.1 Вибір методу реалізації

Для реалізації було обрано метод з використанням 68-точкової моделі обличчя, оскільки метод з використанням глибинного навчання потребує розмічений датасет для тренування мережі для виявлення втоми, а в поточний момент часу такого датасету не існує в світі. Метод з використанням каскадів Хаара не є доцільним для проведення дослідження оскільки використовує файли каскадів с документованою точність і тому можливо спрогнозувати точність цього метода.

4.2 Вибір мови програмування

Мовою програмування для додатку було обрано Python, а середовищем написання – Python Idle 3.10.

Python – це високорівнева, інтерпретована, інтерактивна та об'єктно-орієнтована скриптована мова. Python призначений для того, щоб бути легкочитаємим. Він часто використовує ключові слова, і має менше синтаксичних конструкцій, ніж інші мови.

- Python є інтерпретованою мовою

Python обробляється під час виконання інтерпретатором. Вам не потрібно складати свою програму, перш ніж її виконати. Це схоже на Perl та PHP.

- Python є інтерактивним

Ви можете насправді сидіти за підказкою Python і взаємодіяти з інтерпретатором безпосередньо, щоб написати свої програми.

- Python є об'єктно-орієнтованим

Python підтримує об'єктно-орієнтований стиль або техніку програмування, що інкапсулює код у об'єктах.

Python -мова початківців

Python-це чудова мова для програмістів на рівні початківців і підтримує розробку широкого спектру додатків від простої обробки тексту до браузерів www до ігор.

4.2 Dlib

DLIB - це сучасний інструментарій C ++, що містить алгоритми машинного навчання та інструменти для створення складного програмного забезпечення в C ++ для вирішення проблем реального світу. Він використовується як в галузі, так і в наукових колах у широкому спектрі доменів, включаючи робототехніку, вбудовані пристрої, мобільні телефони та великі високоефективні обчислювальні середовища. Ліцензування з відкритим кодом DLIB дозволяє використовувати його в будь -якій програмі, безкоштовно.

DLIB-ML реалізує численні алгоритми машинного навчання: SVM, K-засоби кластеризації, Байєсівські мережі, і багато інших.

DLIB також має функціональність корисності, включаючи багатопоточність, роботу з мережами, числові алгоритми, обробку зображень, та алгоритми стиснення даних та перевірки цілісності даних.

DLIB включає широке покриття тестування модулів та приклади їх реалізацій. Кожен клас і функція в бібліотеці є задокументованою. Цю документацію можна знайти на домашній сторінці бібліотеки. DLIB забезпечує хорошу основу для розробки програм машинного навчання в C ++.

DLIB дуже схожий на DMTL тим, що він забезпечує загальний високоефективний інструментарій машинного навчання з багатьма різними алгоритмами, але DLIB нещодавно оновлений і має більше прикладів. DLIB також містить набагато більш підтримуючу функціональність.

Що робить DLIB унікальним, це те, що вона розроблена як для використання у дослідженнях, так і для створення програм машинного навчання в C ++.

4.4 OpenCV

OPENCV (Open Source Computer Vision Library) - це відкрита бібліотека для роботи з алгоритмами комп'ютерного зору, машинним навчанням та обробкою зображень. Написана на C ++, але також існує для Python, JavaScript, Ruby та інших мов програмування. Вона працює в Windows, Linux та MacOS, iOS та Android.

Де використовується OPENCV

OPENCV можна використовувати там, де необхідний комп'ютерний зір. Ця сфера IT -індустрії працює з технологіями, які дозволяють пристрою "бачити", розпізнавати та описувати зображення. Комп'ютерне бачення дає точну інформацію про те, що показано на малюнку, з описом, характеристиками та розміром (з певним ступенем надійності).

Бібліотека також працює з машинним навчанням - галуззю, яка навчає алгоритмів діяти так чи інакше.

OPENCV використовується в:

1) робототехніці - для орієнтації робота в космосі, розпізнавання предметів та взаємодія з ними; Медичні технології-створити точні методи діагностики, наприклад, 3D-візуалізація органу під час МРТ;
2) Промислові технології - для автоматизованого контролю якості, читання етикеток, сортування продуктів тощо;
3) Безпеці - створити «розумні» камери спостереження, які реагують на підозрілі дії, для читання та розпізнавання біометрики;

4) Мобільних фотографіях - створити фільтри краси, які змінюють обличчя додатків;
5) В транспорті - для розвитку автопілотів.

Функції OPENCV:

- Робота зі структурами даних
- Модифікація зображень
- Додавання ефектів
- Малювання зверху зображення
- Розпізнавання об'єктів

4.5 Реалізація

Додаток захоплює відео з вбудованої камери і для кожного окремого кадру визначає EAR для обох очей, рахує їх середнє арифметичне та повідомляє при значенні EAR нижче норми.

```
import cv2
import dlib
from scipy.spatial import distance
def eye_aspect_ratio(eye):
    p2_minus_p6 = distance.euclidean(eye[1], eye[5])
    p3_minus_p5 = distance.euclidean(eye[2], eye[4])
    p1_minus_p4 = distance.euclidean(eye[0], eye[3])
    ear = (p2_minus_p6 + p3_minus_p5) / (2.0 * p1_minus_p4)
    return ear

cap = cv2.VideoCapture(0)

hog_face_detector = dlib.get_frontal_face_detector()

dlib_facelandmark = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")
n1=0
while True:
    _, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    faces = hog_face_detector(gray)
    for face in faces:

        face_landmarks = dlib_facelandmark(gray, face)

        left_eye = []
        right_eye = []

        for n in range(36,42):
                x = face_landmarks.part(n).x
                y = face_landmarks.part(n).y
                left_eye.append((x,y))
        for n in range(42,48):
                x = face_landmarks.part(n).x
                y = face_landmarks.part(n).y
                right_eye.append((x,y))
        ear1 = eye_aspect_ratio(left_eye)
        ear2 = eye_aspect_ratio(right_eye)
        ear = (ear1+ear2)/2
        if ear < 0.24:
            for n in range(0, 68):
                x = face_landmarks.part(n).x
                y = face_landmarks.part(n).y
                cv2.circle(frame, (x, y), 2, (0, 0, 255), 1)
        else:
            for n in range(0, 68):
                x = face_landmarks.part(n).x
                y = face_landmarks.part(n).y
                cv2.circle(frame, (x, y), 1, (0, 255, 0), 1)
    cv2.imshow("Face Landmarks", frame)

    key = cv2.waitKey(1)
    if key == 27:
        break
cap.release()
cv2.destroyAllWindows()
```

Рисунок 4.1 – Код додатку

Приклад роботи додатку:

1) EAR вище або дорівнює норми:

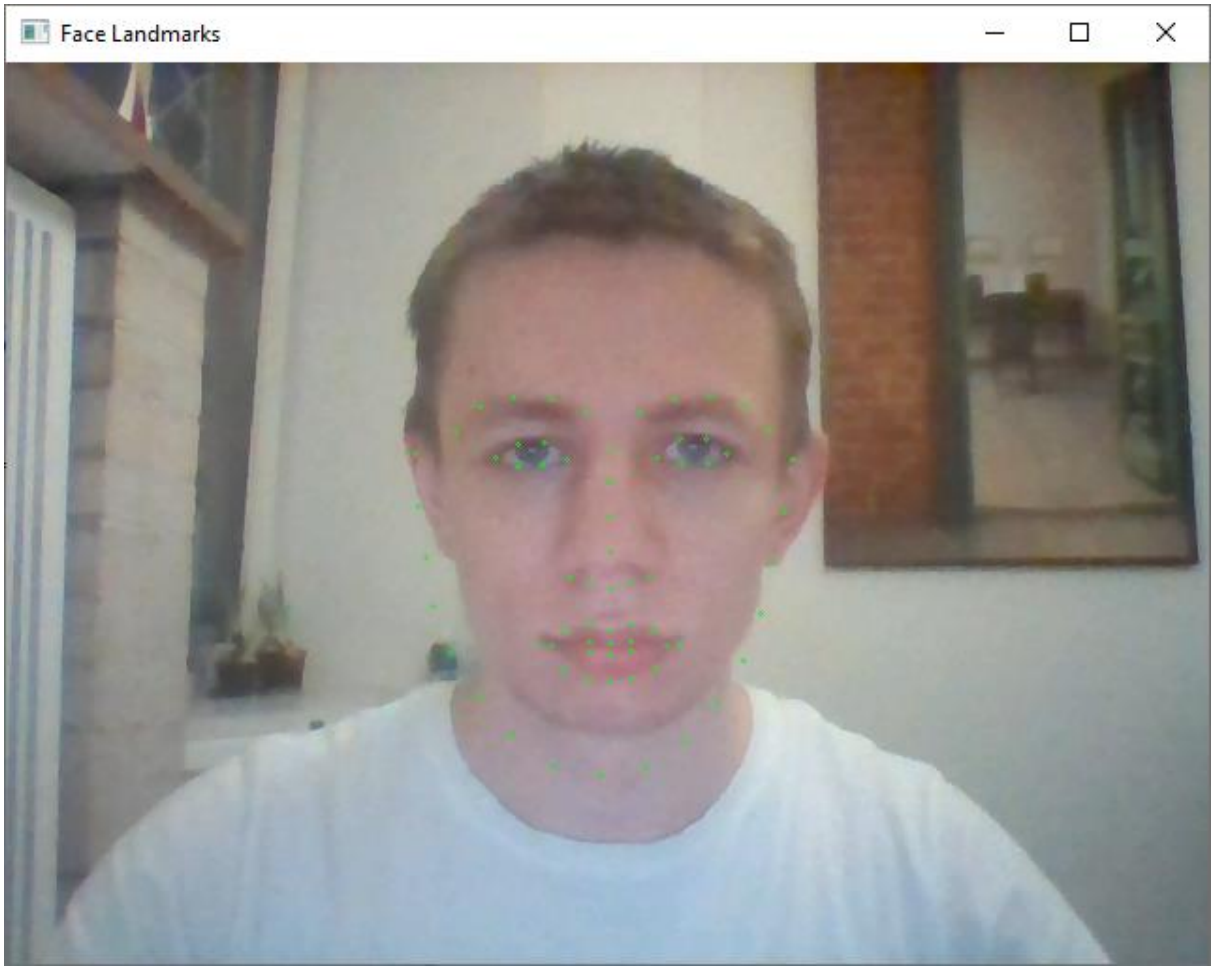

Рисунок 4.2 – Приклад роботи додатку з EAR вище норми

2) EAR нижче норми:

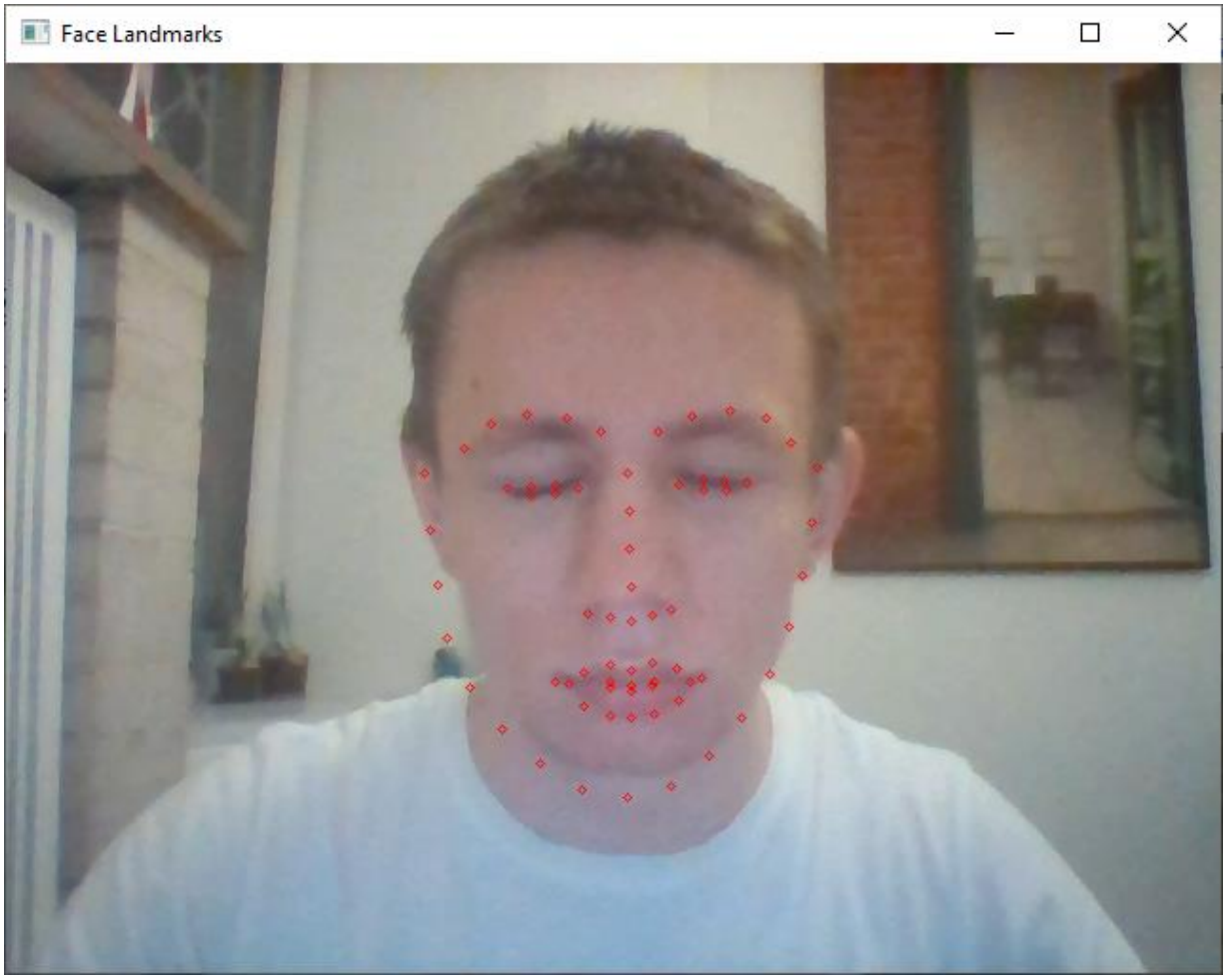

Рисунок 4.3 – Приклад роботи додатку з EAR нижче норми

4.6 Аналіз роботи додатку

Для визначення точності роботи додатку було проведено тестування в різних умовах освітлення, наявності окуляр і зміни нахилу голови. Так як на даний момент не існує розміченого датасету-еталону аналіз проводиться на основі власнозібраних відео 20 людей поділених на кадри. Стан втоми на кожному кадрі визначався людиною, якій належить відео.

Результати роботи додатку було порівняно з розміткою датасету і занесено в таблицю:

	Кількість позитивних результатів	Кількість негативних результатів	Точність
Без окуляр	18	2	90%
З окулярами	19	1	95%
Гарне освітлення	18	2	90%
Погане освітлення	16	4	80%
Нахил голови вперед	17	3	85%
Нахил голови назад	18	2	90%
Нахил голови вліво	15	5	75%
Нахил голови вправо	19	1	95%

Таблиця 4.1 – Результат роботи додатку

4.7 Висновок до розділу 4

У цьому розділі було розроблено додаток для контролю втоми користувача. Основною для нього став метод з використанням 68-точкової моделі обличчя. Для захоплення обличчя та нанесення точок на обличчя використовуються бібліотеки OpenCV і Dlib. Це бібліотеки з відкритим кодом і доступним для

всіх репозиторієм, які надають доступ для використання машинного навчання та обробки потокового відео.

Було проведено тестування додатку на власному датасеті, який складавься з в кадрів з відміченим станом втоми. На підставі дослідження було встановлено точність цього додатку у 87,5%. Можна відмітити що було досягнуто достатньо високої точності визначення стану втоми, але погане освітлення або зміна нахилу голови призводить до її зменшення на 2-5%. Для більш точної оцінки доцільно використовувати верифікований датасет, але в поточний момент часу такого не існує, оскільки галузь є нерозвиненою.

5 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

У даному розділі проведена оцінка основних функціональних та вартісних характеристик програмного продукту, розробленого для вирішення задачі визначення втоми користувача за допомогою засобів комп'ютерного зору в режимі поточного часу.

Програмний продукти призначено для використання на персональних комп’ютерах під управлінням будь-якої операційної системи.

Нижче наведено аналіз різних варіантів реалізації модулю з метою вибору оптимальної стратегії створення програмного продукту, враховуючи при цьому як економічні фактори, так і на характеристики продукту, що впливають на продуктивність роботи і на його сумісність з апаратним забезпеченням. Для цього було використано апарат функціонально-вартісного аналізу.

Функціонально-вартісний аналіз – це технологія оцінки реальної вартості продукту або послуги незалежно від організаційної структури компанії. Прямі та побічні витрати розподіляються по продуктам та послугам у залежності від потрібних обсягів ресурсів на кожному етапі виробництва. Виконані на цих етапах дії у контексті метода ФВА називаються функціями.

Мета ФВА полягає у забезпеченні найбільш оптимального розподілу ресурсів, що виділені на виробництво продукції або надання послуг, на прямі та непрямі витрати. У даному випадку – аналізу функцій програмного продукту й виявлення усіх витрат на реалізацію цих функцій.

Метод ФВА починається з визначення послідовності функцій, необхідних для виробництва продукту. Спочатку йдуть всі можливі функції, які розподіляються по двом групам: ті, що впливають на вартість продукту і ті, що не впливають. Також на цьому етапі оптимізується сама послідовність скороченням кроків, що не впливають на витрати.

Для кожної функції визначається повний обсяг річних витрат та кількість робочих часів. На основі даних оцінок визначається кількісна характеристика джерел витрат. Після визначення джерел витрат проводиться кінцевий розрахунок витрат на виробництво продукту.

5.1 Постановка задачі техніко-економічного аналізу

Методом FVA було проведено техніко-економічне обґрунтування для розробки системи виявлення та фільтрації образливих висловлювань у коментарях у соціальних мережах. Оскільки основні проектні рішення стосуються всієї системи, кожна окрема підсистема повинна їм задовольняти. Тому власне аналіз – це

аналіз функціональності програмних продуктів, призначених для обробки та фільтрації даних.

Технічні вимоги визначені для продукту :

1) можливість використання без підключення інтернет-мережі;
2) висока продуктивність;
3) зручний та інтуїтивно зрозумілий інтерфейс користувача;
4) витрата мінімальної кількість ресурсів.

5.1.1 Обґрунтування функцій програмного продукту

Основною функцією F0 є розробка програмного продукту, який вирішує проблему виявлення втоми у користувачів ПК. Залежно від конкретного використання можна виділити наступні основні функції програмного продукту:

F_1 – мова програмування;

F_2 – бібліотеки для використання комп’ютерного зору та машинного навчання;

F_3 – середовище розробки.

Кожна з основних функцій може мати декілька варіантів реалізації.

Функція F_1:

1) Python
2) C++

Функція F_2:

1) Python IDLE
2) Visual Studio

Функція F_3:

1) Dlib
2) OpenCV

5.1.2 Варіанти реалізації основних функцій

Варіанти реалізації основних функцій наведені у морфологічній карті системи на рисунку 4.1. Морфологічна карта відображує всі можливі комбінації варіантів реалізації функцій, які складають повну множину варіантів ПП.

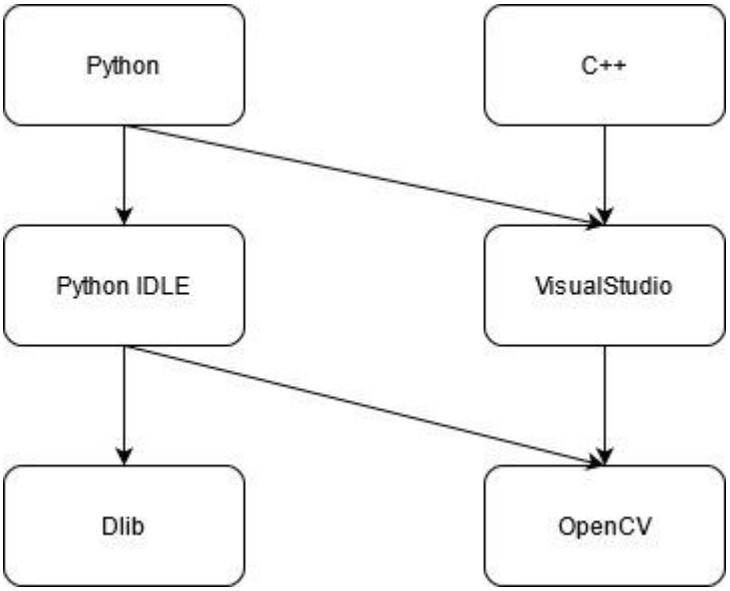

Рисунок 5.1 – Морфологічна карта варіантів реалізації функцій

На основі цієї карти побудовано позитивно-негативну матрицю варіантів основних функцій (табл. 3).

Таблиця 5.1 – Позитивно-негативна матриця варіантів основних функцій

Функція	Варіант реалізації	Переваги	Недоліки
F_1	А	Простота написання коду, невеликий час написання коду.	Значне використання сторонніх розробок, неможливість точного аналізу ПП.
	Б	Висока швидкодія ПП, можливість точного аналізу ПП.	Написання коду є складним, необхідність самостійної реалізації допоміжних засобів.
F_2	А	Простота використання.	Складність кастомізації та персонального налаштування, необхідність роботи з консольним терміналом.
	Б	Велика кількість можливостей для персонального налаштування	Складність використання
F_3	А	Велика кількість засобів машинного навчання	Мала кількість засобів обробки відеопотоку
	Б	Велика кількість засобів обробки відеопотоку	Мала кількість засобів машинного навчання

На основі аналізу позитивних і негативних матриць робимо висновок, що при розробці програмного продукту слід відкинути деякі варіанти реалізації функціональних можливостей, оскільки вони не відповідають цілям програмного продукту. Ці варіанти позначені на морфограмі.

Таким чином, будемо розглядати такі варіанти реалізації програмного продукту:

- F_1(А) – F_2(Б) – F_3(Б)
- F_1(А) – F_2(А) – F_3(А)
- F_1(А) – F_2(А) – F_3(Б)
- F_1(Б) – F_2(Б) – F_3(Б)

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

5.2 Обґрунтування системи параметрів програмного продукту

5.2.1 Опис параметрів

На підставі даних про основні функції, що повинен реалізувати програмний продукт, вимог до нього, визначаються основні параметри виробу, що будуть використані для розрахунку коефіцієнта технічного рівня.

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

- X1 – час, який витрачається на виконання коду;
- X2 – об’єм пам’яті необхідний для роботи ПП;
- X3 – потенційний об’єм програмного коду.

5.2.2 Кількісна оцінка параметрів

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію програмного продукту як показано у таблиці 4.

Таблиця 5.2 – Основні параметри програмного продукту

Опис параметру	Умовні позначення	Одиниці виміру	Значення параметра		
			гірші	середні	кращі
Час на виконання коду	X1	мс	1000	500	200
Об’єм пам’яті	X2	Мб	128	64	32
Об’єм програмного коду	X3	Строки коду	2000	1000	500

5.2.3 Аналіз експертного оцінювання параметрів

Після детального обговорення та аналізу кожен експерт оцінює важливість кожного параметра для певної мети – розробки програмного продукту з максимально зручним інтерфейсом та зрозумілою взаємодією з користувачами.

Значення кожного параметра визначали шляхом попарного порівняння. Оцінку проводила експертна комісія з 5 осіб. Визначення коефіцієнта значущості включає:

1) визначення рівня значимості параметра шляхом присвоєння різних рангів;

2) перевірку придатності експертних оцінок для подальшого використання;

3) визначення оцінки попарного пріоритету параметрів;

4) обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 5.

Таблиця 5.3 – Результати ранжування показників

Параметр	Ранг параметра за оцінкою експерта					Сума рангів	Відхилення, Δ_i	Δ_i^2
	1	2	3	4	5			
X1	4	5	4	5	4	22	2	4
X2	5	5	5	4	4	23	3	9
X3	3	2	3	3	4	15	5	25
Разом	12	12	12	12	12	60	0	38

Для перевірки ступеню достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^{N} * r_{ij} = 60$$

де r_{ij} – ранг *i*-го параметра, визначений *j*-м експертом;

N – число експертів.

б) середня сума рангів *T*:

$$T = \frac{1}{n} R_i = 20.$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T.$$

г) загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{n} * \Delta_i^2 = 6.$$

д) коефіцієнт узгодженості (конкордації):

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 \cdot 38}{5^2(3^3 - 3)} = 0{,}76 > W_k = 0{,}67.$$

Ранжирування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0,67. Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 6. Числове значення, що визначає ступінь переваги *i*-го параметра над *j*-тим, a_{ij} визначається за формулою:

$$a_{ij} = \{$$

$$1{,}5 \;\; x_i > x_j;$$

$$1{,}0 \;\; x_i = x_j;$$

$$0{,}5 \;\; x_i < x_j.$$

$$\}$$

Таблиця 5.4 – Результати ранжування параметрів

Параметри	Експерти					Підсумкова оцінка	Числове значення коефіцієнтів переваги
	1	2	3	4	5		
X1,X2	<	=	<	>	=	=	1,0
X1,X3	>	>	>	>	=	>	1,5
X2,X3	>	>	>	>	=	>	1,5

З отриманих числових оцінок переваги складемо матрицю $A = \|a_{ij}\|$. Для кожного параметра розрахунок вагомості K_{B_i} проводиться за наступною формулою:

$$K_{B_i} = \frac{b_i}{\sum_{i=1}^{n} * b_i},$$

де $b_i = \sum_{j=1}^{N} a_{ij}$ – вагомість i-го параметра за результатами оцінок всіх експертів;

a_{ij} –коефіцієнт переваги i-го на j-тим параметром.

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%). На другому і наступних кроках відносні оцінки розраховуються за наступною формулою:

$$K_{B_i} = \frac{b_i'}{\sum_{i=1}^{n} * b_i'},$$

де $b_i' = \sum_{j=1}^{N} * a_{ij} b_j.$

Як видно з таблиці 7, різниця значень коефіцієнтів вагомості після другої ітерації не перевищує 1%, тому додаткові ітерації не потрібні.

Таблиця 5.5 – Розрахунок вагомості параметрів

i	j			Перша ітерація		Друга ітерація	
	X1	X2	X3	B_i	K_{B_i}	B_i^1	$K_{B_i}^1$
X1	1,0	1,0	1,5	3,5	0,39	11,25	0,40
X2	1,0	1,0	1,5	3,5	0,39	11,25	0.40
X3	0,5	0,5	1,0	2,0	0,22	5	0,20
Разом				9	1,0	27.5	1,0

5.3 Аналіз рівня якості варіантів реалізації функцій

– X1 – час, який витрачається на виконання коду;
– X2 – об’єм пам’яті необхідний для роботи ПП;
– X3 – потенційний об’єм програмного коду.

Рівень якості кожного варіанту виконання основних функцій визначається окремо.

Абсолютні значення параметрів X1(час, який витрачається на виконання коду) та X2 (об’єм пам’яті необхідний для роботи ПП;) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра X3 (потенційний об’єм програмного коду) обрано найкращим варіантом, тобто 200 строк.

Коефіцієнт технічного рівня якості для кожного варіанта реалізації ПП розраховується за формулою:

$$K_{TP} = \sum_{i=1}^{n} * K_{B_i} B_i,$$

де n – кількість параметрів;

K_{B_i} – коефіцієнт вагомості і–го параметра;

B_i – оцінка i–го параметра в балах.

Розрахунок показників рівня якості представлено відповідно в таблиці 8.

Таблиця 5.6 – Розрахунок показників якості

Основні функції	Варіант реалізації	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F_1	А	500	5	0,4	2
	Б	200	10	0,4	4
F_2	А	64	5	0,4	2
	Б	32	10	0,4	4
F_3	А	1000	5	0,2	1
	Б	500	10	0,2	2

За цими даними визначаємо рівень якості кожного з варіантів:

- $F_1(А) – F_2(А) – F_3(А) = 2 + 2 + 1 = 5$
- $F_1(А) – F_2(Б) – F_3(Б) = 2 + 4 + 2 = 8$
- $F_1(А) – F_2(Б) – F_3(А) = 2 + 4 + 1 = 7$
- $F_1(Б) – F_2(Б) – F_3(Б) = 4 + 4 + 2 = 10$

Отже, найкращим є четвертий варіант, для якого коефіцієнт технічного рівня має найбільше значення.

5.4 Економічний аналіз варіантів розробки програмного продукту

Для визначення вартості розробки програмного продукту спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;

2. Розробка алгоритму збору даних.

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3.

Для реалізації завдання 1 використовує інформацію у вигляді даних, а завдання 2 використовує стандартні методи збору даних. Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань. Загальна трудомісткість обчислюється за формулою.

$$T_O = T_P \cdot K_П \cdot K_{СК} \cdot K_М \cdot K_{СТ} \cdot K_{СТ.М}$$

де T_P – трудомісткість розробки програмного продукту;

$K_П$ – поправочний коефіцієнт;

$K_{СК}$ – коефіцієнт на складність вхідної інформації;

$K_М$ – коефіцієнт рівня мови програмування;

$K_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм;

$K_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення.

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: $T_P = 78$ людино-днів. Поправочний коефіцієнт, який враховує вид вхідної інформації для першого завдання: $K_П = 1{,}7$. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації рівний $K_{СК} = 1$. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $K_{СТ} = 0{,}8$. Тоді загальна трудомісткість програмування першого завдання дорівнює:

$$T_1 = 78 \cdot 1.7 \cdot 0.8 = 106{,}08 \text{ людино-днів.}$$

Проведемо аналогічні розрахунки для другого завдання, в якому використовується алгоритм третьої групи складності зі ступенем новизни Б, тобто $Т_Р = 25$ людино-днів, $К_П = 0{,}9$, $К_{СК} = 1$, $К_{СТ} = 0{,}8$:

$$Т_2 = 25 \cdot 0.9 \cdot 0.8 = 18 \text{ людино-днів.}$$

Оскільки загальна трудомісткість усіх варіантів реалізації збігаються, їх можна об’єднати в одну групу.

Загальна трудомісткість складає:

$$Т_0 = (106{,}08 + 18) \cdot 8 = 996{,}64 \text{ людино-годин;}$$

В розробці беруть участь програміст з окладом 20000 грн., один аналітик в області даних з окладом 25000 грн, та один інженер даних з окладом 18000 грн. Визначимо середню зарплату за годину за формулою:

$$С_Ч = \frac{M}{T_m \cdot t} \text{ грн.,}$$

де M – місячний оклад працівників;

T_m – кількість робочих днів на місяць;

t – кількість робочих годин в день.

$$С_Ч = \frac{20000 + 25000 + 18000}{3 \cdot 20 \cdot 8} = 131{,}25 \text{ грн.}$$

Тоді, розрахуємо заробітну плату за формулою:

$$С_{ЗП} = С_Ч \cdot T_i \cdot К_Д,$$

де $С_Ч$ – величина погодинної оплати праці програміста;

T_i – трудомісткість відповідного завдання;

$К_Д$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

$$С_{ЗП} = 131,25 \cdot 979,2 \cdot 1,2 = 127\,315,59 \text{ грн.}$$

Відрахування на соціальний внесок становить 22%:

$$С_{СВ} = С_{ЗП} \cdot 0,22 = 127\,315,59 \cdot 0,22 = 28\,009,43 \text{ грн.}$$

Тепер визначимо витрати на оплату однієї машино-години. Оскільки одна ЕОМ обслуговується одним інженером апаратного забезпечення з окладом 10000 грн. та коефіцієнтом зайнятості $К_З = 0,2$ то для однієї машини отримаємо:

$$С_Г = 12 \cdot М \cdot К_З = 12 \cdot 10\,000 \cdot 0,2 = 24\,000 \text{ грн.}$$

З урахуванням додаткової заробітної плати:

$$С_{ЗП} = С_Г \cdot (1 + К_З) = 24\,000 \cdot (1 + 0,2) = 28\,800 \text{ грн.}$$

Відрахування на соціальний внесок становить 22%:

$$С_{СВ} = С_{ЗП} \cdot 0,22 = 57\,600 \cdot 0,22 = 6\,336 \text{ грн.}$$

Амортизаційні відрахування розраховуємо за формулою при амортизації 25% та вартості ЕОМ – 30 000 грн.:

$$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1,15 \cdot 0,25 \cdot 20000 = 8\,625 \text{ грн.}$$

де $К_{ТМ}$ – коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача;

$К_А$ – річна норма амортизації;

$Ц_{ПР}$ – договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо за формулою:

$$С_Р = К_{ТМ} \cdot К_Р \cdot Ц_{ПР} = 1{,}15 \cdot 0{,}05 \cdot 30000 = 1\ 725 \text{ грн.}$$

де $К_Р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$Т_{ЕФ} = (Д_К - Д_В - Д_С - Д_Р) \cdot t \cdot К_В, Т_{ЕФ} = (365 - 104 - 12 - 16) \cdot 8 \cdot 0{,}9 = 1\ 667{,}6 \text{ год.}$$

де $Д_К$ – календарна кількість днів у році;

$Д_В, Д_С$ – відповідно кількість вихідних та святкових днів;

$Д_Р$ – кількість днів планових ремонтів устаткування;

t – кількість робочих годин в день;

$К_В$ – коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$$С_{ЕЛ} = Т_{ЕФ} \cdot N_C \cdot К_З \cdot Ц_{ЕЛ} = 1\ 677{,}6 \cdot 0{,}65 \cdot 0{,}2 \cdot 1{,}68 = 366{,}38 \text{ грн.}$$

де N_C – середньо-споживча потужність приладу;

$К_З$ – коефіцієнтом зайнятості приладу;

$Ц_{ЕЛ}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$$С_Н = Ц_{ПР} \cdot 0{,}67 = 30\ 000 \cdot 0{,}67 = 20100 \text{ грн.}$$

Тоді, річні експлуатаційні витрати будуть складати:

$$С_{ЕК} = С_{ЗП} + С_{СВ} + С_А + С_Р + С_{ЕЛ} + С_Н, С_{ЕК} = 28\ 800 + 6\ 336 + 8\ 625 + 1\ 725 + 828{,}06 + 20\ 100 = 65\ 970{,}06 \text{ грн.}$$

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$С_{МГ} = \frac{С_{ЕК}}{Т_{ЕФ}} = \frac{65\ 970{,}06}{1\ 677{,}6} = 39{,}32 \text{ грн/год.}$$

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу складають:

$$С_{М} = С_{МГ} \cdot Т = 39{,}32 \cdot 996{,}64 = 39\ 187{,}40 \text{ грн.}$$

Накладні витрати складають 67% від заробітної плати:

$$С_{Н} = С_{ЗП} \cdot 0{,}67 = 127\ 315{,}59 \cdot 0{,}67 = 85\ 301{,}45 \text{ грн.}$$

Отже, вартість розробки програмного продукту за варіантами становить:

$$С_{ПП} = С_{ЗП} + С_{СВ} + С_{М} + С_{Н}, С_{ПП}$$
$$= 127\ 315{,}59 + 28\ 009{,}43 + 39\ 187{,}40 + 85\ 301{,}45$$
$$= 279\ 813{,}87 \text{ грн.}$$

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$К_{ТЕР} = \frac{К_{К}}{С_{ПП}} = \frac{7{,}16}{280\ 092{,}87} = 2{,}56 \cdot 10^{-5}$$

5.5 Висновок до розділу 5

В результаті виконання економічного розділу були систематизовані і закріплені теоретичні знання в галузі економіки та організації виробництва використанням їх для техніко-економічного обґрунтування розробки методом функціонально-вартісного аналізу.

На основі даних про зміст основних функцій, які повинен реалізувати програмний продукт, були визначені чотири найбільш перспективні варіанти реалізації продукту. Найбільш ефективним виявився третій варіант реалізації функцій ПП, який дає максимальну величину коефіцієнта техніко-економічного рівня, вартість витрат для нього становить $С_{ПП} = 279\ 813{,}87$ грн.

Цей варіант передбачає:

– мову програмування C++;
– середовище розробки VisualStudio;
– використання бібліотеки OpenCV;
– високу швидкодію;

Висновки

В даній роботі було проаналізовано 3 різних методи визначено користувача, а саме:

- алгоритм на основі визначення втоми за допомогою каскадів Хаара;

- алгоритм з використанням 68-точкової моделі обличчя;

- алгоритм з використанням глибинного навчання.

Для їх реалізації найбільш доцільним є використанням таких допоміжних бібліотек, як Dlib та OpenCv. Вони надають можливість використовувати засоби машинного навчання та обробляти потокове відео отримане з будь-якого пристрою.

Методи, вказані вище, було порівняно з точки зору продуктивності та простоти реалізації і виходячи з цього аналізу було прийняте рішення про розробку додатку, що базується на 68-точковій моделі обличчя.

Також було розроблено та протестовано додаток на основі 2-го методу на мові Python з використанням бібліотек Dlib та OpenCv. Його роботу було протестовано на власному датасеті з різними умовами тестування. У результаті було визначено приблизну точність визначення стану втоми додатком і зроблено припущення про можливі варіанти покращення його роботи.

Список використаних джерел

1. Real Time Drowsy Driver Detection Using Haarcascade Samples [Електронний ресурс] - Режим доступу до ресурсу: https://www.researchgate.net/publication/271376030_Real_Time_Drowsy_Driver_Detection_Using_Haarcascade_Samples

2. Research on a Real-Time Driver Fatigue Detection Algorithm Based on Facial Video Sequences [Електронний ресурс] - Режим доступу до ресурсу: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjlt4P_prL4AhWHgSoKHWXpDOgQFnoECAgQAQ&url=https%3A%2F%2Fwww.mdpi.com%2F2076-3417%2F12%2F4%2F2224%2Fpdf%3Fversion%3D1645437674&usg=AOvVaw2kR8vpvKQUo8Uk-6qFKDUu

3. Research on fatigue detection based on visual features [Електронний ресурс] - Режим доступу до ресурсу: https://ietresearch.onlinelibrary.wiley.com/doi/pdfdirect/10.1049/ipr2.12207

4. Zhang, G.; Yau, K.K.; Zhang, X.; Li, Y. Traffic accidents involving fatigue driving and their extent of casualties. [Електронний ресурс] - Режим доступу до ресурсу: http://doi.org/10.1016/j.aap.2015.10.033

5. Borghini, G.; Astolfi, L.; Vecchiato, G.; Mattia, D.; Babiloni, F. Measuring neurophysiological signals in aircraft pilots and car drivers for the assessment of mental workload, fatigue and drowsiness. Neurosci. Biobehav. [Електронний ресурс] - Режим доступу до ресурсу: http://doi.org/10.1016/j.neubiorev.2012.10.003

6. Davidovi, J.; Pei, D.; Lipovac, K.; Anti, B. The significance of the development of road safety performance indicators related to driver fatigue. [Електронний ресурс] - Режим доступу до ресурсу: http://doi.org/10.1016/j.trpro.2020.03.024

7. Cui, Z.; Sun, H.M.; Yin, R.N. Real-time detection method of driver fatigue state

based on deep learning of face video. Multimed. [Електронний ресурс] - Режим доступу до ресурсу: http://doi.org/10.1007/s11042-021-10930-z
8. Hu, X.; Lodewijks, G. Exploration of the effects of task-related fatigue on eye-motion features and its value in improving driver fatigue-related technology. Transp. Res. Part F Traffic Psychol. [Електронний ресурс] - Режим доступу до ресурсу: http://doi.org/10.1016/j.trf.2021.03.014
9. You, F.; Li, Y.-H.; Huang, L.; Chen, K.; Zhang, R.-H.; Xu, J.-M. Monitoring drivers' sleepy status at night based on machine vision. Multimed. [Електронний ресурс] - Режим доступу до ресурсу: http://doi.org/10.1007/s11042-016-4103-x
10. Ren, S.; Cao, X.; Wei, Y.; Sun, J. Face Alignment at 3000 FPS via Regressing Local Binary Features. In Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA
11. Zhang, K.; Zhang, Z.; Li, Z.; Qiao, Y. Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks. IEEE Signal Process. [Електронний ресурс] - Режим доступу до ресурсу: http://doi.org/10.1109/LSP.2016.2603342
12. Chaudhuri, A.; Routray, A. Driver Fatigue Detection through Chaotic Entropy Analysis of Cortical Sources Obtained From Scalp EEG Signals. IEEE Trans. Intell. Transp. Syst. [Електронний ресурс] - Режим доступу до ресурсу: http://doi.org/10.1109/TITS.2018.2890332
13. Zhang, C.; Ma, J.; Zhao, J.; Liu, P.; Cong, F.; Liu, T.; Li, Y.; Sun, L.; Chang, R. Decoding Analysis of Alpha Oscillation Networks on Maintaining Driver Alertness. Entropy [Електронний ресурс] - Режим доступу до ресурсу: http://doi.org/10.3390/e22070787
14. He, K.; Zhu, Y.; He, Y.; Liu, L.; Lu, B.; Lin, W. Detection of Malicious PDF Files Using a Two-Stage Machine Learning Algorithm. Chin. J. Electron. [Електронний ресурс] - Режим доступу до ресурсу: http://doi.org/10.1049/cje.2020.10.002
15. Hao, Z.; Wan, G.; Tian, Y.; Tang, Y.; Dai, T.; Liu, M.; Wei, R. Research on

Driver Fatigue Detection Method Based on Parallel Convolution Neural Network. In Proceedings of the 2019 IEEE International Conference on Power, Intelligent Computing and Systems (ICPICS), Shenyang, China.
16. Liu, W.; Sun, H.; Shen, W. Driver fatigue detection through pupil detection and yawing analysis. In Proceedings of the 2010International Conference on Bioinformatics and Biomedical Technology, Chengdu, China.
17. Wadhwa, A.; Roy, S.S. Driver drowsiness detection using heart rate and behavior methods: A study. Data Anal. Biomed. Eng. Healthc. 2021, 55, 163–177.
18. Villanueva, A.; Benemerito, R.L.L.; Cabug-Os, M.J.M.; Chua, R.B.; Rebeca, C.K.D.C.; Miranda, M. Somnolence Detection System Utilizing Deep Neural Network. In Proceedings of the 2019 International Conference on Information and Communications Technology (ICOIACT), Yogyakarta, Indonesia, 24–25 July 2019; pp. 602–607.
19. Savas, B.K.; Becerikli, Y. Real Time Driver Fatigue Detection System Based on Multi-Task ConNN. IEEE [Електронний ресурс] - Режим доступу до ресурсу: http://doi.org/10.1109/ACCESS.2020.2963960
20. Zhang, Y.; Han, X.; Gao, W.; Hu, Y. Driver Fatigue Detection Based On Facial Feature Analysis. Int. J. Pattern Recognit. Artif. Intell. [Електронний ресурс] - Режим доступу до ресурсу: http://doi.org/10.1142/S0218001421500348