Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет __Комп'ютерної інженерії та управління_____
(повна назва)

Кафедра Автоматизації проєктування обчислювальної техніки_____
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти ____другий магістерський)_____

Методи моніторингу концентрації людини за допомогою нейронних мереж.
(тема)

Виконав: здобувач другого року навчання,
групи СКСм-23-2 Носик Д. О._____
(прізвище, ініціали)

Спеціальність 123 Комп'ютерна інженерія
(код і повна назва спеціальності)

Тип програми __освітньо-професійна____
(освітньо–професійна або освітньо–наукова)

Освітня програма Комп'ютерна інженерія
(повна назва освітньої програми)

Керівник доцент Рожнова Т.Г._____
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ _Чумаченко С.В
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет_____Комп'ютерної інженерії та управління _____

Кафедра_____Автоматизації проєктування обчислювальної техніки__

Рівень вищої освіти____другий (магістерський)_____

Спеціальність_____123_ Комп'ютерна інженерія _____
(шифр і назва)

Тип програми _____Освітньо-професійна_____
(освітньо-професійна або освітньо–наукова)

Освітня програма _____Спеціалізовані комп'ютерні системи_____
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«__02__»____09_____ 2024 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві_____Носику Данилу Олександровичу_____
(прізвище, ім'я, по батькові)

1. Тема роботи Методи моніторингу концентрації людини за допомогою нейронних мереж. _____

затверджена наказом по університету від _"08"_ ___11_____2024 р. № __1189 Ст____

2. Термін подання студентом роботи до екзаменаційної комісії ____15.01.2025_____.

3. Вихідні дані до роботи _____

Операційна система Android _____

Мова розробки Kotlin _____

Середовище розрбки Android Studio _____

Бібліотека нейронної мержі Firebas ML Kit _____

Бібліотека нейронної мержі Mediapipe _____

4. Перелік питань, що потрібно опрацювати в роботі _____

Визначення релевантних фізіологічних параметрів для оцінки концентрації _____

Дослідження принципу роботи нейронних мереж для розпізнавання обличчя _____

Розробка механізму оцінки концентрації за допомогою нейронних мереж _____

Оцінка ефективності роботи різних нейронних мереж _____

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) __18 слайдів, pptx_____

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

7. Дата видачі завдання ____02.09.2024_____

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Видача теми дослідження, узгодження і затвердження теми	02.09.2024 – 05.09.2024	
2	Аналіз проблемної галузі, постановка задачі,	05.09.2024 – 24.10.2024	
3	Проектування досліджуваної системи	24.10.2024 – 10.11.2024	
4	Реалізація моделі	10.11.2024 – 01.12.2024	
5	Проведення експериментів	01.12.2024 –07.12.2024	
6	Збір данних	07.12.2024 –10.12.2024	
7	Оформлення пояснювальної записки	10.12.2024 – 30.12.2024	
8	Перевірка виконаного проекту керівником,	01.01.2025 – 15.01.2025	
9	Захист проекту	15.01.2025 –25.01.2025	

Здобувач _____
(підпис)

Керівник роботи _____ ___доцент Рожнова Т. Г _____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка містить 56 сторінок, 13 рисунків, 14 формул, 1 таблиця та 17 джерел за переліком посилань.

НЕЙРОННІ МЕРЕЖІ, КОНЦЕНТРАЦІЯ, ФІЗІОЛОГІЧНІ ПОКАЗНИКИ, ANDROID, FIREBASE, MEDIAPIPE, KOTLIN, СМАРТФОН

Метою кваліфікаційної роботи є дослідження методів оцінки концентрації людини в реальному часі за допомогою нейронних мереж.

Об'єктом дослідження є процеси моніторингу рівня концентрації людини, зокрема методи, що дозволяють визначати й оцінювати зміни зосередженості за допомогою мобільних пристроїв.

Предметом дослідження є методи і алгоритми нейронних мереж, що застосовують для автоматизованого аналізу відеопотоку, отриманого з камери телефону, з метою визначення рівня концентрації.

Для досягнення мети проведено аналіз фізіологічних параметрів, таких як рухи очей та кут повороту голови, що вказують на ступінь концентрації. Використовуючи нейронні мережі та камеру смартфона, було створено систему, здатну безперервно відстежувати концентрацію людини. Результати дослідження показують, що система може бути корисною для зменшення кількості помилок, своєчасного виявлення ознак втоми та підтримки високої концентрації, що сприятиме покращенню ефективності праці.

ABSTRACT

The explanatory note contains 56 pages, 13 figures, 14 formulas, 1 table, and 17 sources in the reference list.

NEURAL NETWORKS, CONCENTRATION, PHYSIOLOGICAL INDICATORS, ANDROID, FIREBASE, MEDIAPIPE, KOTLIN, SMARTPHONE

The purpose of the qualification work is to study methods for assessing human concentration in real time using neural networks.

The object of the study is the processes of monitoring the level of human concentration, in particular, methods that allow determining and assessing changes in concentration using mobile devices.

The subject of the study is methods and algorithms of neural networks used for automated analysis of the video stream obtained from the phone camera in order to determine the level of concentration.

To achieve the goal, an analysis of physiological parameters, such as eye movements, pupil diameter and facial activity, indicating the degree of concentration, was carried out. Using neural networks and a smartphone camera, a system was created that can continuously monitor human concentration. The results of the study show that the system can be useful for reducing the number of errors, timely detection of signs of fatigue and maintaining high concentration, which will contribute to improving work efficiency.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Firebase ML Kit – набір інструментів машинного навчання від Google для мобільних додатків, що підтримує різноманітні функції, такі як розпізнавання тексту, облич, штрихкодів тощо.

MediaPipe – платформа від Google для створення мультимодальних рішень комп'ютерного зору, яка забезпечує інструменти для розпізнавання облич, рухів рук, тіла тощо.

Kotlin – сучасна мова програмування, яка широко використовується для розробки Android–додатків.

Pitch – кут, що визначає нахил голови вперед або назад відносно горизонтальної осі.

Yaw – кут, що визначає поворот голови вліво або вправо відносно вертикальної осі.

Roll – кут, що визначає нахил голови вліво або вправо відносно поздовжньої осі.

CameraX – бібліотека для розробки камерних функцій у мобільних додатках на платформі Android.

RAM – оперативна пам’ять пристрою, яка використовується для тимчасового зберігання даних під час роботи програм.

CPU – центральний процесор, основний елемент обчислювальної системи, що виконує більшість розрахунків і обробки даних.

ВСТУП

У сучасному світі, де швидкість інформаційних потоків та навантаження на когнітивні здібності людини стрімко зростають, проблема підтримки високої концентрації уваги стає все більш актуальною. Ефективна концентрація є ключовою складовою продуктивності в багатьох сферах діяльності, включаючи освіту, роботу, спорт та творчість. Відволікання, стрес і втома негативно впливають на здатність до зосередження, що, в свою чергу, знижує якість виконуваних завдань і загальну ефективність [1].

Система моніторингу концентрації людини є інноваційним інструментом, який дозволяє оцінювати рівень уваги в режимі реального часу, ідентифікувати фактори, що впливають на її зміну, та впроваджувати корективи для підтримання оптимального стану. Така система може базуватися на різних методах збору та аналізу даних, включаючи фізіологічні показники (серцебиття, активність мозку), поведінкові фактори (рухи очей, міміка) та зовнішні умови (освітлення, шум).

Метою кваліфікаційної роботи є дослідження принципів роботи систем моніторингу концентрації людини, їхнього практичного застосування та перспектив розвитку. У роботі розглянуто основні методи та технології, що використовуються для оцінки концентрації, проаналізовано їхню ефективність та можливості інтеграції в різні сфери діяльності. Окрему увагу приділено аналізу переваг і недоліків існуючих рішень [2].

Актуальність теми обумовлена зростаючою потребою в інструментах, що сприяють підвищенню продуктивності та збереженню здоров'я в умовах сучасного темпу життя. Системи моніторингу концентрації можуть стати важливим елементом персоналізованих стратегій управління увагою, що дозволить підвищити ефективність навчання, роботи та інших видів діяльності, забезпечуючи при цьому гармонійний розвиток особистості та

збереження її психоемоційного стану. Також є необхідністю створення доступних та ефективних інструментів для моніторингу концентрації, які можуть бути застосовані для забезпечення безпеки на робочих місцях, підвищення продуктивності та мінімізації ризиків, пов’язаних із відволіканням уваги. Наприклад, у професії водія, де втрата концентрації може призвести до дорожньо-транспортних пригод, система може забезпечити постійний моніторинг стану уваги, попереджаючи про можливі ознаки втоми чи відволікання. У авіації система може допомогти пілотам залишатися уважними під час довготривалих рейсів. В медицині вона здатна підтримувати високу концентрацію хірургів під час складних операцій, попереджаючи про зниження рівня уваги [3].

У рамках дослідження використано бібліотеки MediaPipe та Firebase ML Kit, які надають готові моделі для аналізу фізіологічних параметрів, таких як положення голови, рухи очей, та ступінь відкриття повік.

Реалізація системи включала розробку алгоритму для інтеграції даних із зазначених бібліотек, обчислення показників уваги на основі отриманих параметрів, а також тестування моделі в умовах різних сценаріїв. Особливу увагу приділено оптимізації алгоритмів для роботи на мобільних пристроях із обмеженими ресурсами та адаптації моделей для роботи в реальному часі.

1 ПОСТАНОВКА ЗАДАЧІ ТА СФЕРА ВИКОРИСТАННЯ

1.1 Постановка проблеми

У сучасних умовах високого інформаційного навантаження і постійної зміни стимулів люди все частіше стикаються з труднощами у підтримці концентрації на виконанні конкретних завдань. Це породжує необхідність у розробці автоматизованих систем, які б могли моніторити і оцінювати рівень уваги користувача у реальному часі. Особливо актуальним є використання мобільних пристроїв для цього завдання, оскільки вони забезпечують портативність та доступність технології.

Використання нейронних мереж для моніторингу концентрації на мобільних платформах стикається з низкою технічних викликів. По–перше, обробка відеопотоку з камери телефону потребує високої обчислювальної потужності, особливо при роботі з конволюційними нейронними мережами, що аналізують мікровирази обличчя та рухи очей. По–друге, на мобільних пристроях обмежені апаратні ресурси, зокрема, процесорні та енергетичні обмеження, що ускладнює використання складних моделей. Це вимагає оптимізації алгоритмів для забезпечення плавної роботи та економії заряду батареї.

Сучасні технології глибокого навчання на мобільних пристроях дозволяють створювати оптимізовані алгоритми для визначення ступеня концентрації, здатні працювати у реальному часі з мінімальними витратами ресурсів. Розробка такого алгоритму включає використання легких моделей нейронних мереж, здатних аналізувати мікровирази обличчя та рухи очей, щоб визначити зміни рівня уваги. При цьому важливо забезпечити баланс між точністю та ефективністю роботи алгоритму на мобільній платформі. Це потребує застосування методів компресії, а також розробку спеціалізованих

алгоритмів, які забезпечать швидке й точне визначення ступеня концентрації, не перевантажуючи апаратні ресурси пристрою.

1.2 Мета та задачі дослідження

Мета даного дослідження полягає в детальному вивченні методів моніторингу ступеню концентрації людини за допомогою сучасних мобільних пристроїв та нейронних мереж, для розробки системи моніторингу концентрації людини, яка буде сприяти підвищенню рівня безпеки на робочих місцях. Для досягнення цієї мети, перед дослідженням ставляться наступні задачі:

– дослідити біологічні параметрів, які можуть відображати ступінь зосередженості людини;

– визначити за допомогою яких апаратно–програмних засобів можливо розробити систему аналізу концентрації людини;

– розробити систему моніторингу концентрації людини за допомогою сучасних програмно–апаратних засобів;

– дослідити, ефективність роботи системи в різних умовах.

Ці задачі допоможуть отримати повне розуміння проблематики систем моніторингу концентрації людини, а також дадуть можливість розробити рекомендації щодо впровадження та використання таких систем для підвищення безпеки людей на робочих місцях.

1.3 Об'єкт та предмет дослідження

Об'єктом дослідження є процеси моніторингу рівня концентрації людини, зокрема методи, що дозволяють визначати й оцінювати зміни зосередженості за допомогою мобільних пристроїв. Основна увага зосереджена на використанні нейронних мереж для аналізу та інтерпретації

даних, отриманих з камери смартфона, що дозволяє створити портативну систему для відстеження концентрації.

Предметом дослідження є методи і алгоритми нейронних мереж, що застосовують для автоматизованого аналізу відеопотоку, отриманого з камери телефону, з метою визначення рівня концентрації. Це включає використання конволюційних нейронних мереж для обробки зображень обличчя, відстеження рухів очей та аналіз мікровиразів, що можуть сигналізувати про зміни в рівні уваги.

Дослідження охоплює технічні аспекти використання таких нейронних мереж, їх оптимізацію для ефективної роботи на мобільних пристроях, а також методи інтеграції цих моделей у реальні програми для мобільних платформ.

1.4 Актуальність теми

Сучасні нейронні мережі оптимізовані для мобільних пристроїв, що відкриває можливості для створення систем моніторингу концентрації, здатних працювати у реальному часі. Завдяки камерам мобільних пристроїв стає можливим зчитування фізіологічних маркерів уваги, таких як рухи очей та поворот голови. Це дозволяє реалізувати легкі моделі нейронних мереж, що ефективно працюють навіть на платформах з обмеженими обчислювальними ресурсами та забезпечують економію енергії.

Розробка таких систем, здатних безперервно оцінювати увагу користувача та надавати зворотний зв'язок, стає актуальною з огляду на широкий спектр їх можливих застосувань. Оптимізовані нейронні мережі, що працюють в режимі реального часу, дозволяють забезпечити точний моніторинг уваги, що важливо для підвищення продуктивності та безпеки в багатьох сферах.

2 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

2.1 Визначення релевантних фізіологічних параметрів

Здатність людини концентруватися, яку часто називають «увагою» – це розумове зусилля або когнітивний процес зосередження на певному завданні, думці чи діяльності протягом певного періоду часу, ігноруючи відволікаючі фактори. Ця здатність має вирішальне значення для навчання, вирішення проблем і ефективного виконання завдань.

Фізіологічні параметри можуть відображати ступінь концентрації людини. Ці параметри часто вимірюють за допомогою різних інструментів і методів у таких галузях, як неврологія, психологія та дослідження людського фактора [4].

Основні фізіологічні параметри, за допомогою яких можливо відстежити ступінь концентрації людини, це: рухи очей, діаметр зіниці, мімічна активність, частота миготіння, позиція голови, мікрожести.

Для розробки портативної системи контрою концентрації людини необхідно використати такі параметри, які можливо зчитати за допомогою доступних сучасних пристроїв. Одним з таких пристроїв є звичайний смартфон з камерою.

Використання звичайної камери смартфона та нейронних мереж дозволяє зчитувати низку фізіологічних параметрів для оцінки концентрації людини та інших аспектів її поведінки. Ось ключові параметри, які можна отримати за допомогою такого підходу:

Рухи очей (Eye Tracking). Камера смартфона у поєднанні з нейронними мережами може відстежувати рухи очей, включаючи фіксації та саккади. Це дозволяє визначати, чи людина фокусується на конкретному об'єкті чи перемикає увагу між різними точками (рис. 2.1).

Положення голови та напрямок погляду. Нейронні мережі аналізують положення голови та напрямок погляду людини. Постійні рухи голови або її нахили можуть вказувати на відволікання уваги, тоді як стабільне положення голови свідчить про зосередженість на завданні.

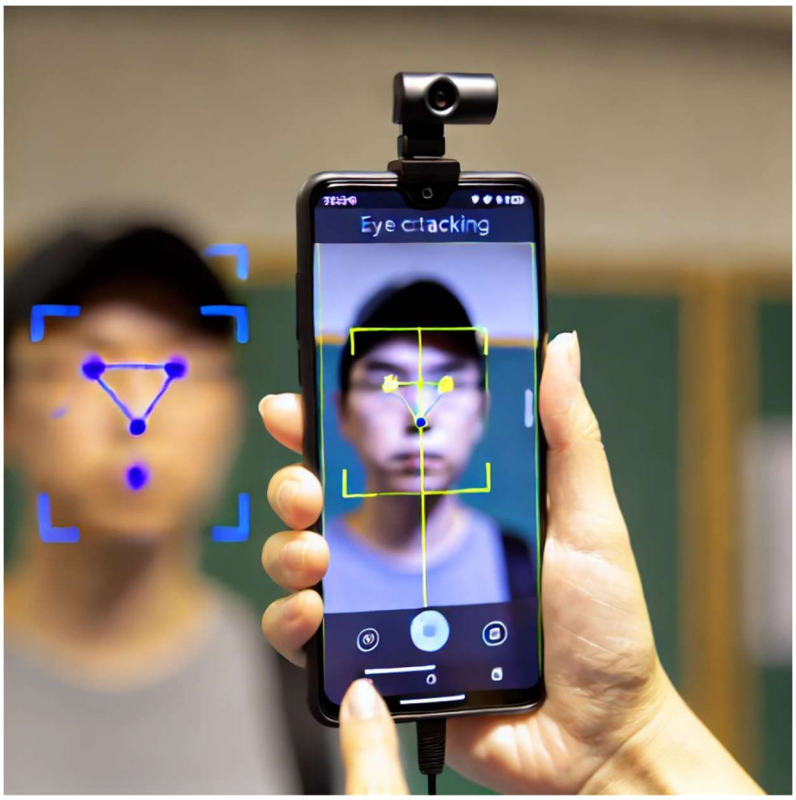


Рисунок 2.1 – Демонстрація можливості зчитування положення очей за допомогою нейронних мереж та смартфона

Використання звичайної камери смартфона та нейронних мереж відкриває широкі можливості для зчитування фізіологічних та поведінкових показників людини, дозволяючи ефективно оцінювати концентрацію у реальному часі без необхідності використання спеціалізованого обладнання.

2.2 Принцип роботи нейронних мереж для аналізу обличчя

Принцип роботи нейронних мереж для зчитування параметрів обличчя базується на обробці зображень обличчя, виявленні характерних ознак і прогнозуванні положення зіниць. Ось основні етапи цього процесу.

Збір даних. Зображення обличчя – збір великої кількості зображень облич з різних кутів і при різних освітленнях. Маркування даних обличчя – вказання точних координат очей або зіниць на кожному зображенні. Це необхідно для навчання нейронної мережі [5].

Попередня обробка даних:

– нормалізація зображень – зміна розміру зображень до стандартного формату, нормалізація яскравості та контрасту;

– видалення шуму – застосування фільтрів для зменшення шумів на зображеннях;

– аугментація даних – збільшення кількості тренувальних даних шляхом обертання, масштабування, зміни яскравості та інших перетворень.

Вибір архітектури. Найчастіше використовуються згорткові нейронні мережі (Convolutional Neural Networks, CNN), які добре підходять для обробки зображень [6].

Для згорткові нейронні мережі виділяють:

– шари згортки – використання згорткових шарів для виявлення основних ознак зображення (країв, кутів, текстур);

– шари підвибірки (Pooling) – зменшення розміру зображення, що знижує обчислювальні витрати та підвищує інваріантність до переміщення та масштабування;

– шари повного зв'язку – використання повнозв'язаних шарів для кінцевого прогнозування координат очей або зіниць (рис. 2.2).

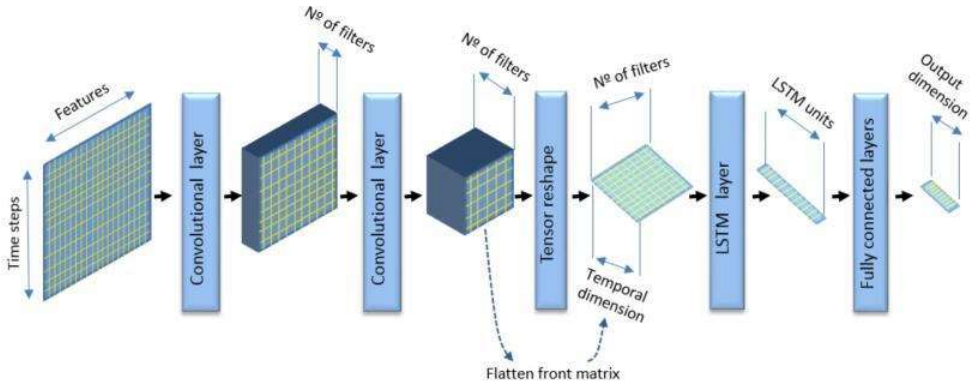


Рисунок 2.2 – Схема комбінації CNN та RNN

Навчання нейронної мережі складається з таких етапів.

Вибір функції втрат. Зазвичай використовуються функції втрат, такі як середньоквадратична похибка (Mean Squared Error, MSE), які порівнюють передбачені координати з реальними координатами очей.

Оптимізація. Використання алгоритмів оптимізації, таких як Adam або SGD, для мінімізації функції втрат.

Епохи та батчі. Навчання мережі на всьому наборі даних з використанням епох та батчів для поступового оновлення ваг.

Валідація та тестування. Валідаційний набір даних – використання окремого набору даних для перевірки точності моделі під час навчання. Тестовий набір даних – оцінка кінцевої точності моделі на нових, раніше не бачених зображеннях.

Реальне використання:

– захоплення зображення. Використання камери для зйомки обличчя в реальному часі;

– обробка зображення. Застосування попередньої обробки до захопленого зображення;

– прогнозування положення очей. Передача зображення через нейронну мережу для отримання координат очей або зіниць;

– відображення результату. Виведення координат на екран або використання їх для інших додатків, таких як відстеження погляду.

Нейронні мережі для зчитування положення очей працюють, обробляючи зображення та виділяючи ключові ознаки для точного визначення координат очей. Цей процес включає збір та підготовку даних, навчання моделі, її валідацію та реальне використання для прогнозування положення очей у режимі реального часу.

2.3 Методи оптимізації нейронних мереж для мобільних платформ

Оптимізація нейронних мереж для мобільних платформ є критично важливою через обмежені обчислювальні ресурси та енергоспоживання цих пристроїв. Основними методами оптимізації є стиснення моделей, квантування, прунінг та використання ефективних архітектур.

Стиснення моделей передбачає зменшення розміру нейронної мережі без значної втрати точності. Це досягається шляхом видалення надлишкових параметрів або шарів, що знижує вимоги до пам'яті та обчислювальних ресурсів.

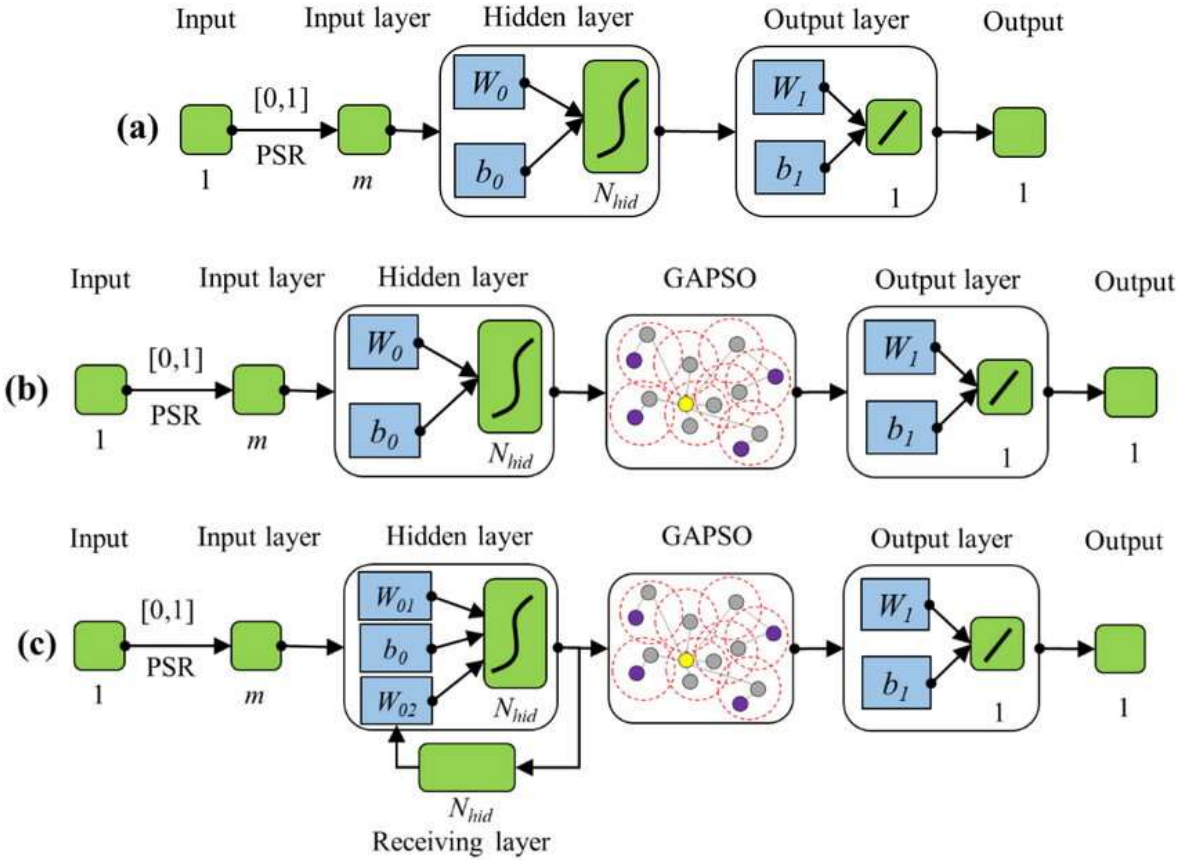
Квантування зменшує розрядність чисел, використовуваних для представлення ваг та активацій у мережі. Наприклад, перехід від 32–бітних до 8–бітних чисел може значно зменшити обсяг пам'яті та прискорити обчислення, з мінімальним впливом на точність моделі.

Прунінг (pruning) включає видалення малозначущих ваг або нейронів з мережі, що зменшує її складність та покращує швидкодію. Цей метод дозволяє зменшити кількість обчислень та енергоспоживання під час інференсу.

Ефективні архітектури, такі як MobileNet та EfficientNet, спеціально розроблені для мобільних платформ. Вони використовують глибокі згорткові мережі з меншою кількістю параметрів та обчислень, що дозволяє досягти високої точності при низьких витратах ресурсів.

Додатково, методи динамічної адаптації мереж, такі як NetAdapt, дозволяють автоматично налаштовувати архітектуру нейронної мережі під конкретні апаратні обмеження мобільного пристрою, забезпечуючи оптимальний баланс між продуктивністю та точністю.

На рисунку 2.3 представлено порівняння різних методів оптимізації нейронних мереж для мобільних платформ, що демонструє їх вплив на розмір моделі та швидкодію.



a) ахітектура C-FNN нейронної мережі;

b) архітектура C-GAPSO-FNN нейронної мережі;

c) архітектура C-GAPSO-RNN нейронної мережі

Рисунок 2.3 – Архітектура нейронної мережі з різними методами оптимізації

2.4 Аналіз існуючих нейронних мереж

Зчитування стану очей та положення голови за допомогою нейронних мереж є актуальною темою досліджень, що має багато застосувань у різних галузях, таких як медицини, психології, маркетингу та інтерфейсів людина–комп'ютер. Нейронні мережі використовуються для аналізу та інтерпретації даних, отриманих з камер, які відстежують положення очей. Нижче наведено огляд основних типів нейронних мереж та підходів, що використовуються для цієї мети.

Для оцінки фізіологічних параметрів, таких як відсоток відкриття очей, положення голови існує кілька потужних бібліотек для нейронних мереж, які можна використовувати.

MediaPipe – це мультимодальний фреймворк від Google, який дозволяє створювати високопродуктивні рішення для обробки відео та зображень. MediaPipe має готові рішення для аналізу обличчя, рухів очей, пози та жестів. Наприклад, компонент Face Mesh дозволяє відстежувати до 468 точок на обличчі, що є корисним для оцінки міміки та рухів голови. Також MediaPipe Holistic об'єднує аналіз обличчя, рук та пози тіла, що робить його ідеальним для комплексного моніторингу концентрації за допомогою камери смартфона. MediaPipe добре оптимізований для Android і може працювати в режимі реального часу [7].

TensorFlow Lite – це полегшена версія TensorFlow, спеціально розроблена для мобільних пристроїв і вбудованих систем. TensorFlow Lite дозволяє розгортати складні нейронні мережі на Android для таких завдань, як розпізнавання рухів очей, відстеження положення голови та аналіз мімічної активності. Бібліотека оптимізована для низького енергоспоживання та високої швидкості обробки на мобільних платформах, що робить її ідеальною для додатків, які потребують високої продуктивності без втрати точності.

TensorFlow Lite підтримує попередньо треновані моделі та моделі, створені з нуля, які можуть використовуватися для різних завдань з аналізу відео [8].

Firebase ML Kit – це набір інструментів для машинного навчання, інтегрований з Firebase, який спрощує використання нейронних мереж на мобільних пристроях. Firebase ML Kit пропонує як хмарні, так і локальні рішення для аналізу зображень та відео, що можуть використовуватися для розпізнавання облич, визначення емоцій та аналізу рухів голови. Він також пропонує API для реального часу, які дозволяють відстежувати зміну виразів обличчя та інші фізіологічні показники. Firebase ML Kit легко інтегрується з Android–додатками та пропонує можливість використовувати як власні моделі TensorFlow Lite, так і готові рішення для нейронних мереж [10].

Dlib – це потужна бібліотека для машинного навчання та комп'ютерного зору, яка також підтримується на Android. Dlib включає алгоритми для виявлення та відстеження обличчя, що є особливо корисним для аналізу мімічних реакцій і рухів голови. Dlib відома своєю точністю при розпізнаванні ключових точок обличчя (facial landmarks), що робить її потужним інструментом для відстеження змін у виразах обличчя та інших показниках, пов'язаних з концентрацією. Ця бібліотека дозволяє створювати кастомні рішення на основі вже існуючих алгоритмів для Android–додатків [11].

OpenCV – це ще одна потужна бібліотека для комп'ютерного зору, яку можна використовувати на Android. Хоча OpenCV не є нейронною мережею сам по собі, вона часто використовується разом з TensorFlow або іншими бібліотеками для попередньої обробки зображень, таких як розпізнавання обличчя, очей або рук, перед їх передачею до нейронної мережі для подальшого аналізу. OpenCV може використовуватись для відстеження рухів очей, жестів та інших поведінкових параметрів, що впливають на концентрацію [12].

Таким чином, бібліотеки MediaPipe, TensorFlow Lite, Firebase ML Kit, Dlib та OpenCV забезпечують широкий спектр можливостей для аналізу

фізіологічних та поведінкових параметрів на Android. Вони оптимізовані для роботи на мобільних платформах і можуть використовуватись для створення комплексних рішень з моніторингу та оцінки концентрації людини в режимі реального часу.

2.5 Аналіз можливостей інтеграції

Інтеграція нейронних мереж у мобільні додатки для оцінки концентрації людини є ключовим аспектом розробки сучасних систем моніторингу. Проведено аналіз можливостей інтеграції найпоширеніших бібліотек, таких як MediaPipe, TensorFlow Lite, Firebase ML Kit, Dlib та OpenCV, з урахуванням їх особливостей, переваг і можливих складнощів.

Для роботи з MediaPipe необхідно отримати вихідний код бібліотеки через офіційний репозиторій та імпортувати його до проекту.

Інтеграція TensorFlow Lite починається з додавання бібліотеки до проекту. Наступним кроком є підготовка моделі у форматі TensorFlow Lite. Її слід розмістити у додатку, забезпечивши правильну обробку файлу без стискання. Після цього необхідно налаштувати ініціалізацію інтерпретатора TensorFlow Lite, який використовуватиме модель для обробки даних у додатку.

Інтеграція Firebase ML Kit вимагає підключення додатка до Firebase через його консоль, де створюється конфігураційний файл для вашого проекту. Потім бібліотека додається до проекту, після чого можна налаштовувати як використання вбудованих моделей, так і завантаження власних.

Для інтеграції Dlib у додаток спочатку потрібно зібрати її нативні бібліотеки для Android за допомогою Android NDK. Ці бібліотеки додаються до проекту і налаштовуються для коректної роботи через Gradle. Для взаємодії між Java–кодом і нативними функціями Dlib необхідно створити Java Native Interface (JNI), що забезпечить доступ до можливостей бібліотеки з основного додатка.

Інтеграція OpenCV передбачає додавання її SDK до проекту. Після імпорту модулів необхідно налаштувати середовище розробки для роботи з функціоналом бібліотеки, включаючи коректне підключення всіх необхідних файлів. Перед початком використання функцій OpenCV потрібно перевірити правильність ініціалізації бібліотеки в середовищі Android.

Аналізуючи можливості інтеграції бібліотек можна зробити висновок, що MediaPipe і Firebase ML Kit є найкращими виборами через їхню простоту налаштування, багатоплатформенність і готові рішення для аналізу фізіологічних показників у режимі реального часу. Ці бібліотеки дозволяють розробникам сконцентруватися на реалізації функціональності додатків, мінімізуючи час і ресурси, необхідні для інтеграції.

3 ПРОЄКТУВАННЯ ДОСЛІДЖУВАНОЇ СИСТЕМИ

3.1 Вибір нейронних мереж для дослідження

Для проведення дослідження з оцінки концентрації людини були обрані дві основні бібліотеки нейронних мереж – Firebase ML Kit та MediaPipe. Цей вибір базується на кількох ключових факторах, таких як простота інтеграції, можливість роботи на мобільних платформах, оптимізація для реального часу та широкі можливості для аналізу фізіологічних параметрів за допомогою камери смартфона.

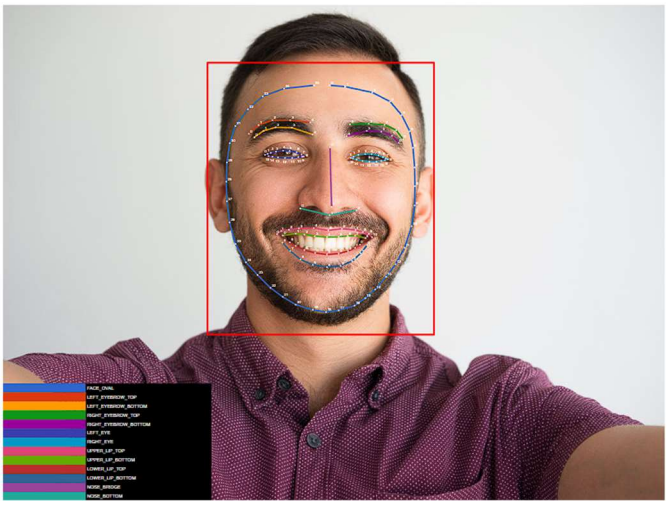


Рисунок 3.1 – Демонстрація зчитування параметрів обличчя за допомогою Firebase ML Kit

Firebase ML Kit був обраний завдяки своїй здатності легко інтегруватися з Android–додатками та пропонувати готові моделі для розпізнавання обличчя, аналізу мімічної активності та визначення виразів (рис 2.3). Він дозволяє здійснювати локальне розпізнавання, що забезпечує низьке енергоспоживання і високу продуктивність навіть на пристроях середнього класу. Окрім цього,

Firebase ML Kit підтримує можливість використовувати як власні моделі TensorFlow Lite, так і хмарні рішення для більш складних завдань. Це дає гнучкість у розробці та дозволяє адаптувати дослідження під різні сценарії застосування.

MediaPipe обраний за його високу продуктивність і можливість працювати в режимі реального часу з відеопотоком. MediaPipe надає готові пайплайни для аналізу рухів очей, положення голови, міміки та жестів, що дозволяє отримувати точні дані про фізіологічні показники людини без необхідності тренування власних моделей (рис. 3.4). Однією з ключових переваг MediaPipe є його оптимізація для мобільних платформ, що дозволяє досягати високої швидкості обробки зображень навіть на пристроях з обмеженими ресурсами. Крім того, MediaPipe дозволяє одночасно обробляти кілька параметрів, таких як рухи очей, обличчя та рук, що робить його ідеальним інструментом для комплексного дослідження.

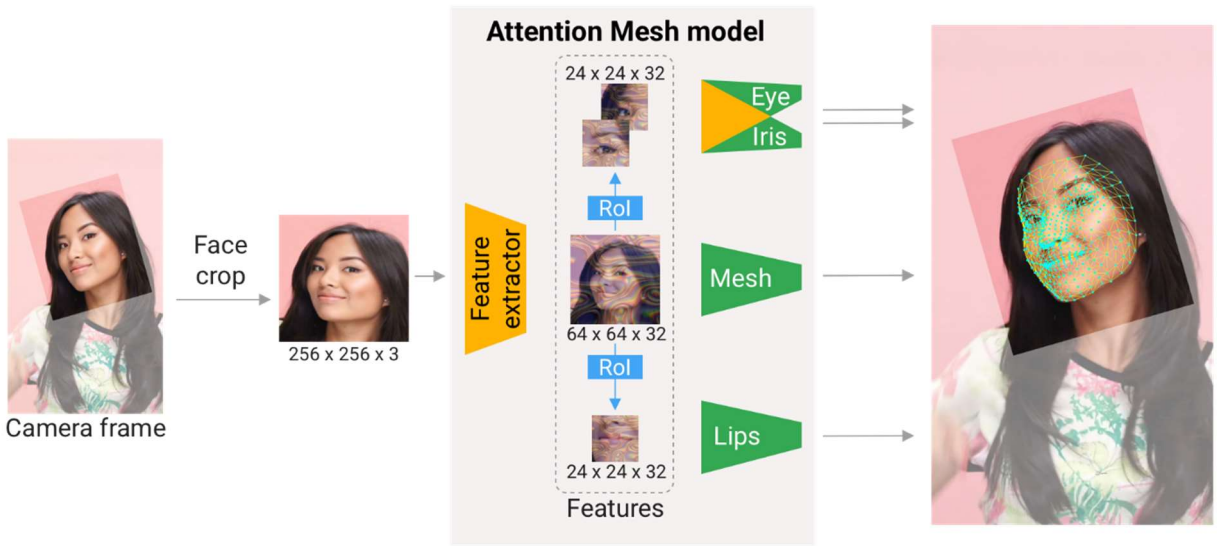


Рисунок 3.2 – MediaPipe огляд архітектури Attention Mesh моделі

Таким чином, вибір Firebase ML Kit та MediaPipe обумовлений їхньою простотою використання, високою продуктивністю на Android, а також можливістю працювати з реальними фізіологічними показниками в режимі

реального часу. Це дозволяє зосередитися на вирішенні завдань дослідження без необхідності витрачати значні ресурси на розробку або оптимізацію власних моделей нейронних мереж.

3.2 Дослідження принципу роботи MediaPipe

Бібліотека MediaPipe, особливо її компонент Face Mesh, є інструментом для високоточного виявлення обличчя та побудови його 3D–моделі. Вона використовує сучасні підходи до обробки зображень, базуючись на алгоритмах глибокого навчання та оптимізації для реального часу. Бібліотека написана переважно мовами C++ та Python, що забезпечує її ефективність та універсальність. Вона також надає API для використання на інших платформах, зокрема Android і iOS, через інтерфейси мовою Java/Kotlin.

Для виявлення обличчя MediaPipe застосовує модель BlazeFace, яка є легкою нейронною мережею, спеціально розробленою для обробки у реальному часі. Основою роботи BlazeFace є:

– конволюційні нейронні мережі (CNN), які виконують просторовий аналіз зображення;

– регресія координат межевих рамок (bounding box regression), яка визначає область, де розташоване обличчя.

Формула регресії координат:

$$Box_{pred} = CNN(I), \tag{3.1}$$

де I – вхідне зображення, а CNN – набір конволюційних шарів, що прогнозують координати (x_{min}, y_{min}, x_{max}, y_{max}) обличчя.

Після виявлення обличчя виконуються етапи масштабування та нормалізації зображення, що забезпечує ефективність моделі:

Масштабування обличчя до фіксованого розміру. Зображення обличчя обрізається відповідно до межевої рамки та масштабується до розміру W×H (наприклад, 192*192 пікселів). Для цього використовується формула:

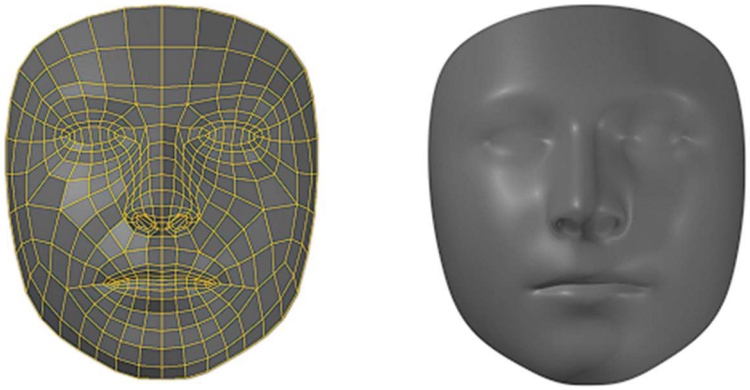
$$ScaledPixel(i,j) = \frac{OriginalPixel(i,j) - MinPixel}{MaxPixel - MinPixel}, \quad (3.2)$$

де i, j – координати пікселів.

Нормалізація. Піксельні значення зображення перетворюються в діапазон [0,1][0, 1][0,1] або [−1,1][–1, 1][−1,1], що покращує збіжність моделі. Формула нормалізації:

$$NormalizedPixel(i,j) = \frac{\text{ScaledPixel}(i,j)}{255.0}, \quad (3.3)$$

Модель Face Mesh прогнозує 468 ключових точок на обличчі (landmarks) із 3D–координатами . Це досягається за допомогою архітектури регресії, яка на основі попередньо обробленого зображення обчислює положення кожної точки [16].



(a) (b)

a) прогнозована топологія сітки;

b) 3D модель сітки;

Рисунок 3.3 – Прогнозована топологія сітки (a) та її 3D модель (b)

Основний принцип роботи – це регресія координат ключових точок:

$$L = Regressor(I), \tag{3.4}$$

де L={(x1, y1, z1),…,(x_{468}, y_{468}, z_{468})} – набір координат точок, а Regressor – легковажна модель на основі CNN.

MediaPipe використовує моделі, попередньо натреновані на великих наборах даних, що дозволяє досягти високої точності навіть за обмежених обчислювальних ресурсів.

Оптимізація моделей:

– Model Quantization. Зменшення розміру моделей через перетворення параметрів до менш точних форматів (наприклад, float32 до int8).

– Edge TPU Optimization. Використання апаратних прискорювачів (GPU, TPU) для реального часу.

BlazeFace здатна працювати з обличчями різного розміру завдяки адаптації межевих рамок до масштабів зображення. Для визначення глибини обличчя (z–координати) використовується елементальний підхід тріангуляції на основі камерної матриці.

Обробка потоків. MediaPipe реалізує обробку даних кадр за кадром із використанням Directed Acyclic Graphs (DAGs), що дає змогу паралельно обчислювати результати різних етапів.

Мови реалізації:

– C++: основний компонент для високопродуктивних обчислень.

– Python API: для інтеграції в наукові проєкти.

– Java/Kotlin: для мобільних додатків.

3.3 Дослідження принципу роботи Firebase Face Detection

Firebase Face Detection – це сервіс для ідентифікації та аналізу облич на зображеннях, який надається в рамках платформи Firebase ML Kit. Він використовує машинне навчання для швидкого та точного виявлення облич, підтримуючи функції визначення рис обличчя, відстеження пози та ключових точок. Бібліотека написана мовою C++ із підтримкою API для інтеграції на Java, Kotlin (Android) та Swift, Objective–C (iOS).

Firebase Face Detection забезпечує визначення облич на основі моделі, що поєднує класичні методи обробки зображень та сучасні алгоритми глибокого навчання. Виявлення облич виконується через створення межевих рамок (bounding boxes) для кожного виявленого обличчя та оцінку ключових рис обличчя (наприклад, очей, носа, рота) для кожного обличчя в межах рамки.

Формула для визначення межевих рамок:

$$Box_{pred} = FeatureExtractor(I), \tag{3.5}$$

де I – вхідне зображення, а FeatureExtractor – модуль, який обчислює ключові ознаки та їх координати.

Firebase Face Detection дозволяє прогнозувати такі параметри обличчя:

– позиція та орієнтація голови: оцінюється нахил (tilt), обертання (yaw) та кут повороту (roll);

– виявлення відкритості очей і рота: прогнозуються бінарні значення для визначення стану (відкритий/закритий);

– положення ключових точок: обчислюються координати 12 ключових точок, таких як кути очей, основа носа, середина рота.

Формула для регресії ключових точок:

$$L = Regressor(B), \tag{3.6}$$

де $L=\{(x_1, y_1),\ldots,(x_{12}, y_{12})\}$ – набір координат точок, а B – межева рамка, виділена на попередньому етапі.

Для забезпечення стабільності роботи моделі Firebase Face Detection застосовує масштабування зображення до фіксованого розміру. Це зменшує обчислювальне навантаження. Нормалізація піксельних значень до діапазону [0,1], що покращує продуктивність моделі:

$$NormalizedPixel(i,j) = \frac{PixelValue(i,j)}{255}, \quad (3.7)$$

Для відео Firebase Face Detection може відстежувати обличчя в реальному часі, використовуючи міжкадрові зв'язки. Ідентифікація обличчя у послідовності кадрів із присвоєнням унікальних ідентифікаторів дозволяє враховувати зміщення положення обличчя між кадрами для забезпечення стабільності.

Firebase Face Detection базується на таких натренованих моделях:

– BlazeFace: ефективна легка нейронна мережа для виявлення меж обличчя, оптимізована для мобільних пристроїв. Вона забезпечує високу продуктивність та точність навіть на обмежених ресурсах;

– DeepLabv3: використовується для покращення сегментації обличчя, що дозволяє більш точно визначати межі та ключові точки;

– Face Landmark Models: моделі регресії, спеціально навчені на великих наборах даних для прогнозування 12 ключових точок на обличчі.

– MobileNet: легка архітектура глибоких нейронних мереж, яка використовується для виділення ознак обличчя;

– Firebase Face Detection використовує попередньо натреновані моделі, що скорочує час обробки та забезпечує високу точність.

Обличчя спочатку визначаються на основі класичних методів виявлення, таких як каскади Гаара, а потім уточнюються за допомогою моделі нейронної

мережі. Позиції рис обличчя прогнозуються на основі глибоких регресійних моделей. Моделі підтримують квантування для зменшення розміру та підвищення продуктивності на мобільних пристроях.

3.4 Проєктування алгоритму визначення концентрації людини

Для проектування алгоритму визначення концентрації людини було обрано підхід, що базується на аналізі двох основних груп параметрів: положення голови та стану очей. Ці параметри дозволяють оцінювати ступінь зосередженості користувача, використовуючи сучасні технології комп'ютерного зору та нейронні мережі.

Градус повороту голови (pitch, yaw, roll) – характеризує положення голови користувача щодо осі тіла. Значні відхилення голови в будь–якому напрямку можуть вказувати на зниження рівня уваги. Відсоток відкриття очей (лівого і правого) – відкриття або часткове закриття очей є важливим індикатором концентрації, оскільки втома або зменшення уваги часто супроводжується зменшенням відкриття очей.

Алгоритм визначення концентрації людини передбачає нормалізацію параметрів повороту голови та відсотка відкриття очей. Параметри повороту голови (pitch, yaw, roll) нормалізуються до значень, де максимальне допустиме відхилення (45°) відповідає 100% втрати концентрації, а відсутність відхилення (0°) вказує на максимальну увагу (100% концентрації). Відсоток відкриття очей безпосередньо використовується у вигляді значень від 0 до 100%, де 100% – це повністю відкриті очі [15].

Формули для обчислення концентрації по кожному параметру повороту голови виглядають так:

$$\begin{aligned} H_{\text{pitch}} &= \left(1 - \frac{|\text{ pitch }|}{45}\right) \times 100 \\ H_{\text{yaw}} &= \left(1 - \frac{|\text{ yaw }|}{45}\right) \times 100, \\ H_{\text{roll}} &= \left(1 - \frac{|\text{ roll }|}{45}\right) \times 100 \end{aligned} \tag{3.8}$$

Де |pitch|, |yaw|, |roll| – абсолютні значення відхилення голови від нульової позиції (прямого погляду вперед).

Концентрація по очах:

– ліве око Oleft – відсоток відкриття лівого ока;

– праве око Oright – відсоток відкриття правого ока.

Середнє значення для концентрації:

$$C = \frac{H_{\text{pitch}} + H_{\text{yaw}} + H_{\text{roll}} + O_{\text{left}} + O_{\text{right}}}{5}, \tag{3.9}$$

Формула розрахунку концентрації:

$$C = \frac{\begin{array}{c}\left(1-\frac{|\text{ pitch }|}{45}\right)\times 100+\left(1-\frac{|\text{ yaw }|}{45}\right)\times 100 \\ +\left(1-\frac{|\text{ roll}}{45}\right)\times 100+O_{\text{left}}+O_{\text{right}}\end{array}}{5}. \tag{3.10}$$

Якщо значення С (Формула 4) наближається до 100%, це свідчить про високу концентрацію користувача. Якщо ж значення знижується, це вказує на зменшення уваги та можливе відволікання або втому.

Алгоритм враховує одночасно кілька важливих факторів:

– положення голови, яке може свідчити про відволікання уваги в бік;

– відкриття очей, що вказує на втомленість.

4 РЕАЛІЗАЦІЯ ДОСЛІДЖУВАНОЇ МОДЕЛІ ТА ПРОВЕДЕННЯ ДОСЛІДЖЕННЯ

4.1 Особливості інтеграції Mediapipe

Бібліотека Mediapipe Face Mesh надає інструменти для отримання тривимірних координат ключових точок обличчя, які можна використати для обчислення параметрів орієнтації голови і стану очей. Детально розглянуто алгоритми обчислення цих параметрів, а також особливості інтеграції Mediapipe з CameraX для обробки відеопотоку в реальному часі.

CameraX – це бібліотека для роботи з камерою на Android, створена Google. Вона забезпечує простий та ефективний API для доступу до функцій камери, обробки зображень і відео, адаптуючись до різних пристроїв. CameraX значно спрощує роботу з камерою порівняно зі старими API, такими як Camera1 та Camera2.

Кадри з CameraX передаються у Mediapipe FaceMesh через ImageProxy. Спочатку кадр конвертується у формат Bitmap, який може бути оброблений Mediapipe (лістниг 4.1).

Лістниг 4.1 – Інтеграція Mediapipe FaceMesh та CameraX

```
    private fun processImageProxy(imageProxy: ImageProxy) {
        val mediaImage = imageProxy.image ?: return
        val rotationDegrees = 
imageProxy.imageInfo.rotationDegrees
        val bitmap = ImageUtils.imageToBitmap(mediaImage, 
rotationDegrees)
        val faceMeshResult = faceMesh.process(bitmap)
        if (faceMeshResult.multiFaceLandmarks().isNotEmpty()) {
            val landmarks = 
faceMeshResult.multiFaceLandmarks()[0].landmarkList
            calculateParameters(landmarks)
        }
```

```
        imageProxy.close()
    }
```

Кадр з ImageProxy перетворюється на Bitmap із урахуванням повороту. faceMesh.process() викликається для обробки кадру і отримання ключових точок.

Для роботи з моделлю обличчя Mediapipe використовую механізм Face Landmarks. Face Landmarks це набір із 468 точок на обличчі, кожна з яких представлена тривимірними координатами (x,y,z). Кожна координата нормалізована відносно розміру обличчя, що дозволяє працювати з будь–яким масштабом. Отримання даних для yawDegrees, pitchDegrees і rollDegrees Для обчислення орієнтації голови використовуються специфічні точки: noseTip (точка кінчика носа), chin (точка підборіддя), leftEyeOuter і rightEyeOuter (зовнішні кути очей). Для кожної точки використовується унікальний індекс з допомогою якого можна отримати їх з масиву faceLandmarks:

```
val noseTip = faceLandmarks[1]
val leftEye = faceLandmarks[33]
val rightEye = faceLandmarks[263]
val chin = faceLandmarks[152]
val forehead = faceLandmarks[10]
```

Дані про landmarks отримуються з об'єкта FaceMeshResult. За допомогою координат цих точок обчислюються кути:

– yaw (кут повороту по горизонталі) розраховується на основі різниці координат зовнішніх точок очей;

– pitch (нахил вперед/назад) визначається за різницею координат носа та підборіддя;

– roll (нахил голови набік) використовується нахил лінії між зовнішніми точками очей;

Для розрахунку ступеня відкриття очей у відсотках використовуються тривимірні координати точок верхньої та нижньої повіки. Нормалізація базується на ширині очей, яка виступає опорною відстанню.

Для обчислення вертикальної відстані між повіками, використовується «Евклідова відстань»:

$$h_{\text{eye}} = \sqrt{\left(x_{\text{upper}} - x_{\text{lower}}\right)^2 + \left(y_{\text{upper}} - y_{\text{lower}}\right)^2 + \left(z_{\text{upper}} - z_{\text{lower}}\right)^2} \quad (4.1)$$

де h_{eye} – вертикальна відстань між верхньою (upperLidupper) та ижньою (lowerLid) точками повіки, x,y,z – нормалізовані координати landmarks, які представляють положення точки в тривимірному просторі.

Для обчислення ширини ока використовується відстань між внутрішнім і зовнішнім куточками ока:

$$w_{\text{eye}} = \sqrt{(x_{\text{outer}} - x_{\text{inner}})^2 + (y_{\text{outer}} - y_{\text{inner}})^2 + (z_{\text{outer}} - z_{\text{inner}})^2} \quad (4.4)$$

де w_{eye}: горизонтальна ширина ока. Вимірюється як тривимірна евклідова відстань між зовнішнім (outerCornerouter) і внутрішнім (innerCornerinner) куточками ока.

Розрахунок відсотка відкриття очей обчислюється, як відношення вертикальної відстані між повіками до ширини ока, помножене на 100.

$$\text{Eye Openness (\%)} = \left(\frac{h_{\text{eye}}}{w_{\text{eye}}}\right) \times 100 \quad (4.3)$$

Mediapipe обробляє кожен кадр з камери, отримує landmarks і передає їх у функції для обчислення yawDegrees, pitchDegrees, rollDegrees та EyeOpenness.

Для зменшення затримок обробки використовується стратегію ImageAnalysis.STRATEGY_KEEP_ONLY_LATEST, щоб уникнути черг кадрів, які можуть збільшувати затримку. Використовується окремий потік для аналізу кадрів, щоб не блокувати основний UI–потік. Для оптимізаціх використання пам’яті ImageProxy закривається після кожного виклику processImageProxy для уникнення витоку пам’яті. Для реалізації багатопоточності використовується механізм Coroutines мови Kotlin, таким чином кінцева функція для отримання параметрів концентрації представлена на лістингу 4.2.

Лістниг 4.2 – Інтеграція Mediapipe FaceMesh та CameraX

```
facemesh!!.setResultListener { faceMeshResult:
FaceMeshResult ->
    lifecycleScope.launch {
        try {
            val concentration =
MediapipeConcentration(faceMeshResult).calculate()
                Log.d(
                    "MediapipePreview",
                    "result: $concentration ${
                        timesList.firstOrNull {
it.same(faceMeshResult.timestamp().toInt()) }
                            ?.calculateTime()
                    }"
                )
        } catch (e: Exception) {
            Log.e("MediapipePreview",
"setupStreamingModePipeline: ${e.message}")
        }
    }
    glSurfaceView!!.setRenderData(faceMeshResult)
    glSurfaceView!!.requestRender()
}
```

4.2 Особливості інтеграції Firebase

Firebase Face Detection є частиною Firebase ML Kit, що забезпечує широкий набір інструментів для машинного навчання, оптимізованих для мобільних додатків. Для розрахунку таких характеристик, як кут повороту голови (Yaw, Pitch, Roll) та відсоток відкриття очей, необхідно інтегрувати Firebase Face Detection SDK у додаток для операційної системи Android, використовуючи мову програмування Kotlin [17].

Робота з Firebase Face Detection починається з налаштування CameraX, який забезпечує захоплення кадрів з камери у режимі реального часу. Дані з камери передаються у вигляді об’єктів ImageProxy, які містять самі зображення та метадані, такі як розмір, формат і кут повороту.

Для обробки зображення у режимі реального часу CameraX налаштовується так, щоб використовувати ImageAnalysis із стратегією STRATEGY_KEEP_ONLY_LATEST. Це дозволяє уникнути затримок у обробці кадрів і підтримувати плавність роботи. На етапі обробки зображення об’єкт ImageProxy конвертується у MlImage шляхом отримання MediaImage та врахування кута повороту, який задається через rotationDegrees.

Firebase Face Detection дозволяє визначати ключові параметри, включаючи кути обертання голови (Yaw, Pitch, Roll) та ймовірності відкриття очей. Значення кутів повертаються у градусах і дозволяють оцінювати положення голови відносно вертикальної, горизонтальної та поздовжньої осей. Нижче наведено приклад використання Firebase для аналізу параметрів (лістинг 4.3).

Обробка даних у реальному часі реалізується за допомогою Kotlin Coroutines, які дозволяють виконувати обчислення асинхронно, що мінімізує затримки і забезпечує плавну роботу додатку.

Лістинг 4.3 – Отримання параметрів з MlImage

```
detector.process(mlImage)
        .addOnSuccessListener { faces ->
            for (face in faces) {
                val yaw = face.headEulerAngleY // Кут обертання навколо вертикальної осі
                val pitch = face.headEulerAngleX // Нахил вперед/назад
                val roll = face.headEulerAngleZ // Нахил убік
                val leftEyeOpen = face.leftEyeOpenProbability ?: -1f
                val rightEyeOpen = face.rightEyeOpenProbability ?: -1f

                Log.d("FaceDetection", "Yaw: $yaw, Pitch: $pitch, Roll: $roll")
                Log.d("FaceDetection", "Left Eye: $leftEyeOpen, Right Eye: $rightEyeOpen")
            }
            imageProxy.close()
        }
        .addOnFailureListener {
            Log.e("FaceDetection", "Error: ${it.message}")
            imageProxy.close()        }
```

Використання параметрів headEulerAngleX, headEulerAngleY та headEulerAngleZ для кутів обертання, а також leftEyeOpenProbability та rightEyeOpenProbability для аналізу відкриття очей, дозволяє точно оцінювати положення голови та стан очей. Такий підхід забезпечує високу ефективність аналізу концентрації користувача в реальному часі.

4.3 Реалізація алгоритму розрахунку концентрації

Алгоритм розрахунку концентрації реалізується через аналіз параметрів положення голови та відкриття очей, які надаються Firebase Face Detection або Mediapipe Face Mesh. Абстрактний клас BaseConcentration забезпечує базову функціональність для обчислення індексу концентрації. Його метод calculateConcentration приймає значення кутів повороту голови (Pitch, Yaw, Roll) і відсоткові показники відкриття очей (leftEyeOpen, rightEyeOpen). Розрахунок починається з нормалізації кутів повороту голови до максимального значення в 45 градусів, що вважається допустимим порогом. Для цього використовується формула, яка визначає пропорційну зменшену концентрацію залежно від відхилення кута. Наприклад, якщо кут нахилу голови наближається до 45 градусів, його вплив на концентрацію стає мінімальним. Для параметрів відкриття очей значення передаються без змін, оскільки вони вже надаються у вигляді відсотків. Підсумкова концентрація розраховується як середнє арифметичне з урахуванням усіх параметрів (лістинг 4.4).

Лістинг 4.4 – Реалізація методу розрахунку концентрації

```
    protected fun calculateConcentration(pitch: Double,  yaw: 
Double,
    roll: Double, leftEyeOpen: Double,  rightEyeOpen: Double): 
Double {
    // Обмеження максимального кута повороту голови
    val maxAngle = 45.0

    // Розрахунок концентрації для повороту голови
    val hPitch = (1 - (Math.abs(pitch) / maxAngle)) * 100
    val hYaw = (1 - (Math.abs(yaw) / maxAngle)) * 100
    val hRoll = (1 - (Math.abs(roll) / maxAngle)) * 100
```

```
    // Підсумкова концентрація (середнє значення)
    return (hPitch + hYaw + hRoll + leftEye + rightEye) / 5
}
```

Інтеграція алгоритму в реальному часі забезпечується через використання CameraX для отримання зображень та Firebase Face Detection або Mediapipe Face Mesh для обробки облич. Дані з детектора передаються у методи обчислення, після чого повертається результат. Наприклад, після виявлення обличчя Firebase Face Detection надає параметри headEulerAngleX, headEulerAngleY, headEulerAngleZ, що відповідають кутам нахилу голови, а також leftEyeOpenProbability та rightEyeOpenProbability, які визначають імовірність відкриття очей. Ці параметри обробляються через спеціальний метод processFaceData, який адаптує значення для використання у розрахунках.

У реалізації алгоритму важливо забезпечити обробку можливих випадків, коли деякі параметри можуть бути недоступними (наприклад, якщо обличчя частково закрите або система не змогла точно визначити стан очей). У таких ситуаціях використовуються значення за замовчуванням, щоб уникнути помилок. Наприклад, якщо ймовірність відкриття очей не визначена, замість неї використовується нульове значення. Для зручності роботи з даними створюються відповідні класи MediapipeConcentration та FirebaseConcentration, які наслідують BaseConcentration який виконує розрахунки на основі вхідних параметрів (рис 4.1).

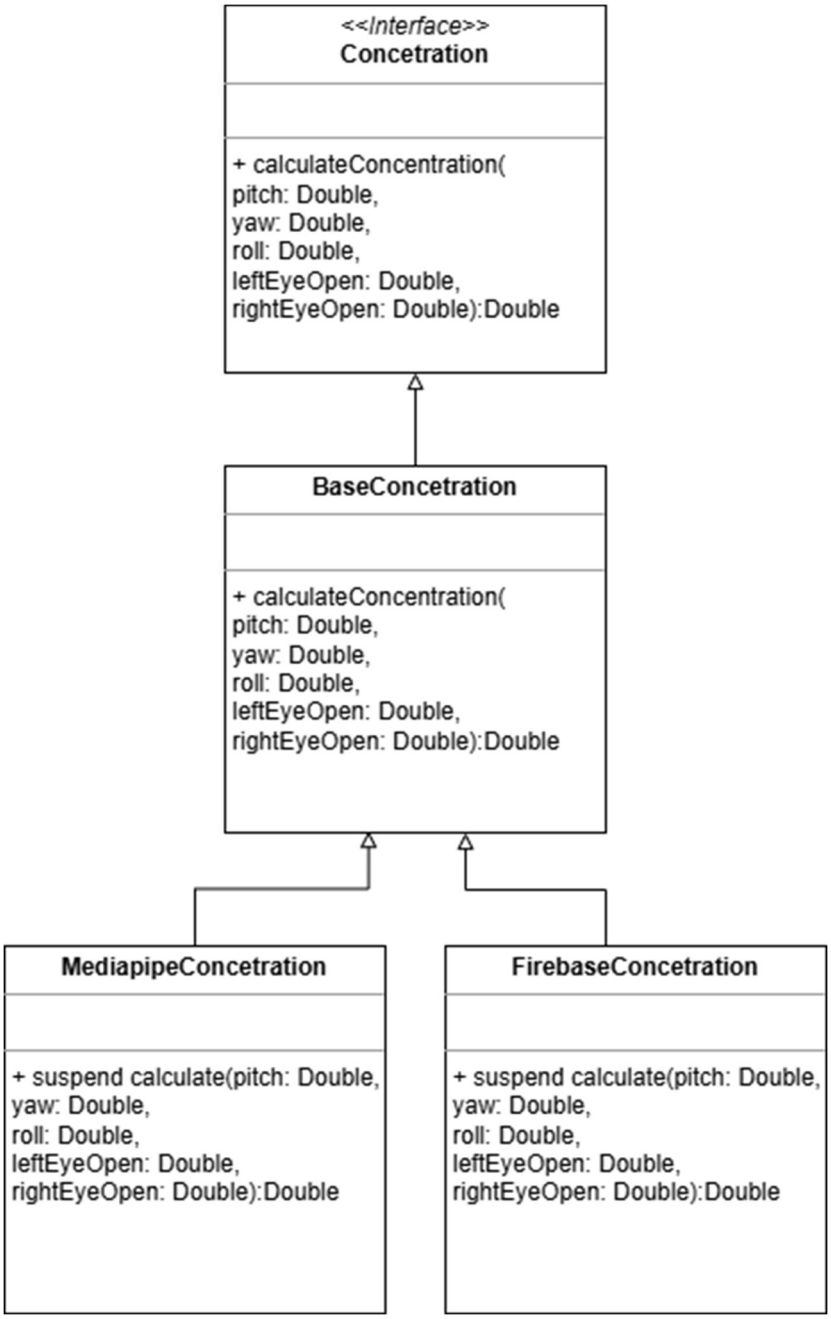


Рисунок 4.1 – Діаграма класів Concentration

4.4 Метод збору даних

Основним параметром для аналізу продуктивності бібліотек Mediapipe Face Mesh та Firebase Face Detection є час, витрачений на обробку кожного окремого кадру. Цей показник дозволяє оцінити швидкодію бібліотек у реальному часі. Для отримання часу обробки у кожної бібліотеки є свої особливості, що враховувалися при розробці методів збору даних.

Для бібліотеки Firebase Face Detection час обробки кадру визначався як інтервал між отриманням кадру у форматі MlImage і завершенням роботи методу detector.process(image), який повертає результат аналізу. Це дозволяє безпосередньо заміряти тривалість обробки за допомогою стандартних інструментів вимірювання часу у мілісекундах.

У реалізації використовувався механізм фіксації початкового часу (timeStart) у момент передачі кадру для обробки і кінцевого часу (timeEnd) у момент отримання результату. Різниця між цими значеннями давала тривалість обробки конкретного кадру. Приклад реалізації у коді (лістинг 4.5).

Лістинг 4.5 – Реалізація розрахунку часу для Firebase

```
    MlKitConcentration(imageProcessor!!, imageProxy,
graphicOverlay!!).let {
        lifecycleScope.launch(Dispatchers.IO) {
            try {
                val timeStart = Date().time
                val concentration = it.calculate()
                val timeEnd = Date().time
                withContext(Dispatchers.Main) {

findViewById<TextView>(R.id.tv_conc_result).text =
                        "MLKit = ${concentration}\nTime =
${timeEnd - timeStart}"
                }
            } catch (e: Exception) {
                Log.e("CameraXLivePreviewActivity",
"bindAnalysisUseCase: ${e.message}")
            }
        }
```

Для бібліотеки Mediapipe Face Mesh прямого способу вимірювання часу обробки кадрів не передбачено. Однак для реалізації заміру використовувався параметр timestamp, який доступний для кожного кадру через об’єкт TextureFrame.

Процес збору даних полягав у фіксації міток часу для кожного кадру на момент його отримання та моменту обробки. Різниця між часовими мітками дозволяла оцінити час, витрачений на обробку кадру. Цей підхід реалізовано за допомогою наступного коду – лістинг 4.6.

Лістинг 4.6 – Реалізація розрахунку часу для Mediapipe

```
    cameraInput!!.setNewFrameListener { textureFrame:
TextureFrame? ->
        Log.d("MediapipePreview", "setNewFrameListener:
${textureFrame}")
        textureFrame?.let {
            lifecycleScope.launch {
                mutex.withLock {
timesList.add(TextureTime(it.timestamp.toInt()))
                }
            }
        }
        facemesh!!.send(textureFrame)
    }
```

Мітка часу (timestamp) кожного кадру додається у список, що дозволяє обчислити час, необхідний для обробки. Хоча цей підхід менш прямий, ніж у випадку Firebase, він забезпечує достатню точність для оцінки продуктивності Mediapipe.

Метод збору даних для Firebase Face Detection є простішим завдяки можливості безпосереднього вимірювання часу роботи методу

detector.process(image). Для Mediapipe Face Mesh процес вимірювання є непрямим і вимагає додаткової обробки часових міток кадрів.

4.5 Опис експерименту

Метою експерименту є дослідження можливості оцінки концентрації людини в реальному часі за допомогою бібліотек Mediapipe Face Mesh і Firebase Face Detection на Android–пристроях. Основна увага приділяється порівнянню швидкодії, точності, стабільності роботи та споживання системних ресурсів обома бібліотеками. Для оцінки концентрації використовуються такі параметри, як положення голови (нахили, повороти) та відсоток відкритості очей.

Для проведення експерименту було Android додаток, який складається з двох модулів. Перший модуль використовує Mediapipe Face Mesh для аналізу ключових точок обличчя, таких як кути очей, повіки, ніс і підборіддя, що дозволяє розраховувати положення голови і відкритість очей. Другий модуль інтегрує Firebase Face Detection, який забезпечує аналіз обличчя з отриманням аналогічних параметрів. Обидва модулі мають вбудовану систему збору даних, яка записує час обробки кадрів, положення голови, відсоток відкритості очей, а також споживання ресурсів CPU і RAM.

Експеримент проводиться в умовах реального часу з залученням тестових користувачів, які виконують завдання на зосередження уваги. Учасникам пропонується слідкувати за рухомою точкою на екрані або утримувати фіксацію погляду на заданій області. Під час експерименту вимірюються такі параметри, як затримка між отриманням зображення з камери та результатами аналізу, стабільність роботи бібліотек, кількість пропущених кадрів, а також обчислювані значення концентрації на основі відкритості очей і положення голови.

Зібрані дані включають час обробки кожного кадру, середнє та пікове завантаження CPU і RAM, відсоток відкритості очей, положення голови, а також кількість збоїв і пропущених кадрів. Для аналізу використовується статистична обробка даних, що дозволяє порівняти ефективність Mediapipe Face Mesh і Firebase Face Detection у реальному часі. Для аналізу завантаженості CPU та RAM використовується інструмент Profiler вбудований в Android Studio.

На основі отриманих результатів зроблено висновки про ефективність використання кожної з бібліотек для задач оцінки концентрації в реальному часі.

4.6 Збір даних та аналіз результатів

Для оцінки ефективності роботи бібліотек Mediapipe Face Mesh та Firebase Face Detection було проведено збір даних щодо використання ресурсів смартфона в реальному часі. Тестування здійснювалося на пристрої Google Pixel 6a з операційною системою Android 14. Моніторинг системних ресурсів включав використання оперативної пам’яті (RAM) та завантаження центрального процесора (CPU) під час роботи кожної бібліотеки.

Для кожного типу бібліотек було розроблено мінімальний користувацький інтерфейс для візуалізації роботи додатку (рис. 4.2).

При роботі з бібліотекою Mediapipe Face Mesh середнє споживання оперативної пам’яті (RAM) склало 650 МБ, що зумовлено інтенсивним обчисленням і підтримкою великої кількості ключових точок обличчя (рис. 4.3). Завантаження центрального процесора (CPU) під час роботи Mediapipe становило в середньому 26% (рис. 4.3).

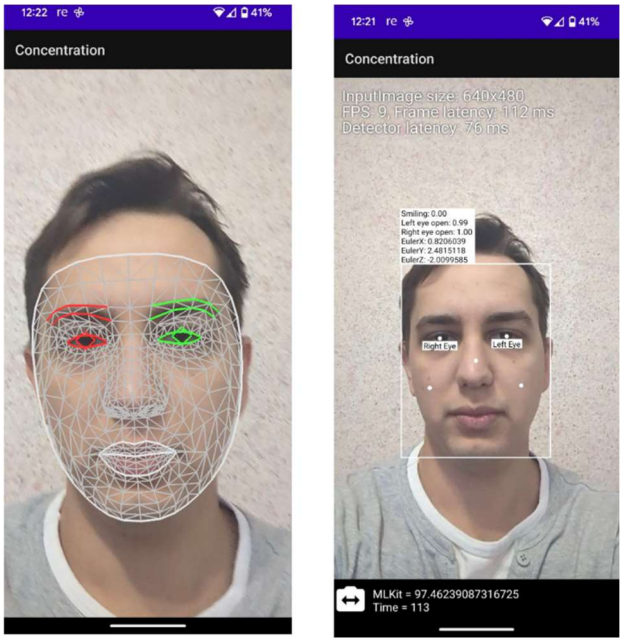


Рисунок 4.2 – Інтерфейс додатку для різних типів нейронних мереж

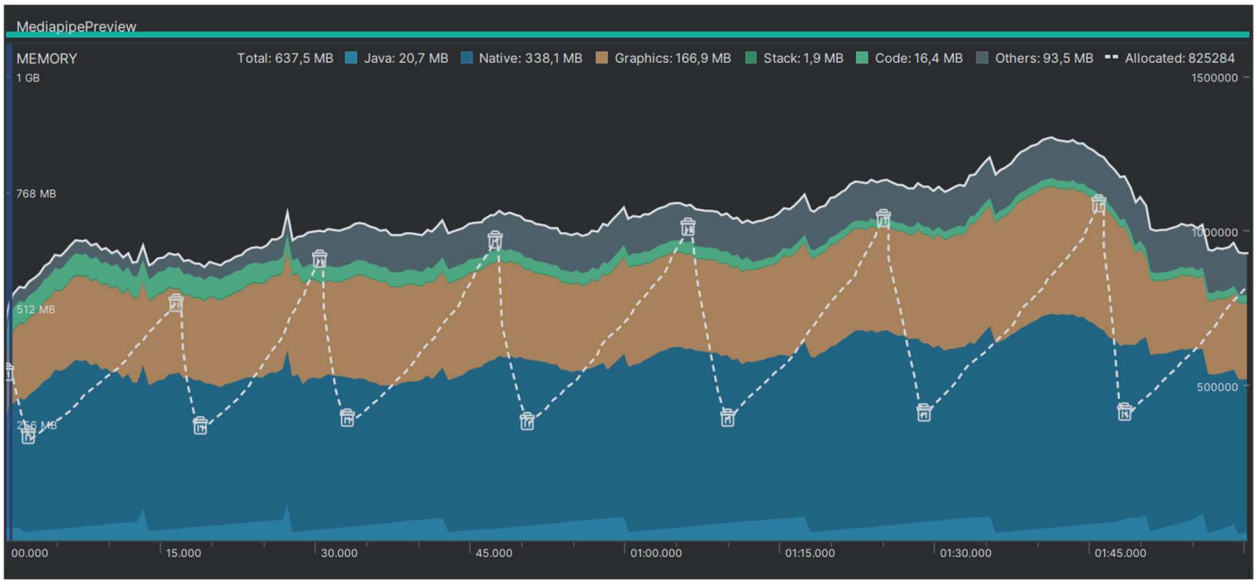


Рисунок 4.3 – Використання RAM для Mediapipe Face Mesh

У свою чергу, Firebase Face Detection демонструє значно менше споживання системних ресурсів:

– середнє значення використання оперативної пам’яті (RAM) склало 280 МБ (рис. 4.5), що пояснюється спрощеним підходом до обробки обличчя,

орієнтованим на виявлення базових параметрів, таких як відкритість очей і положення голови;

– завантаження центрального процесора (CPU) було на рівні 14% (рис. 4.6).

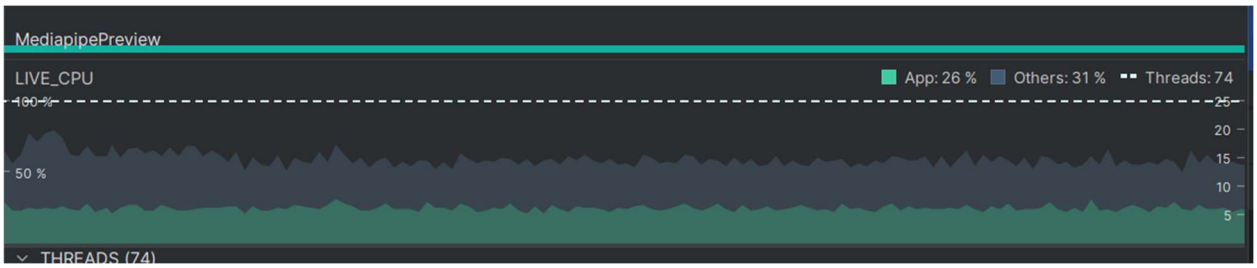


Рисунок 4.4 – Використання CPU для Mediapipe Face Mesh

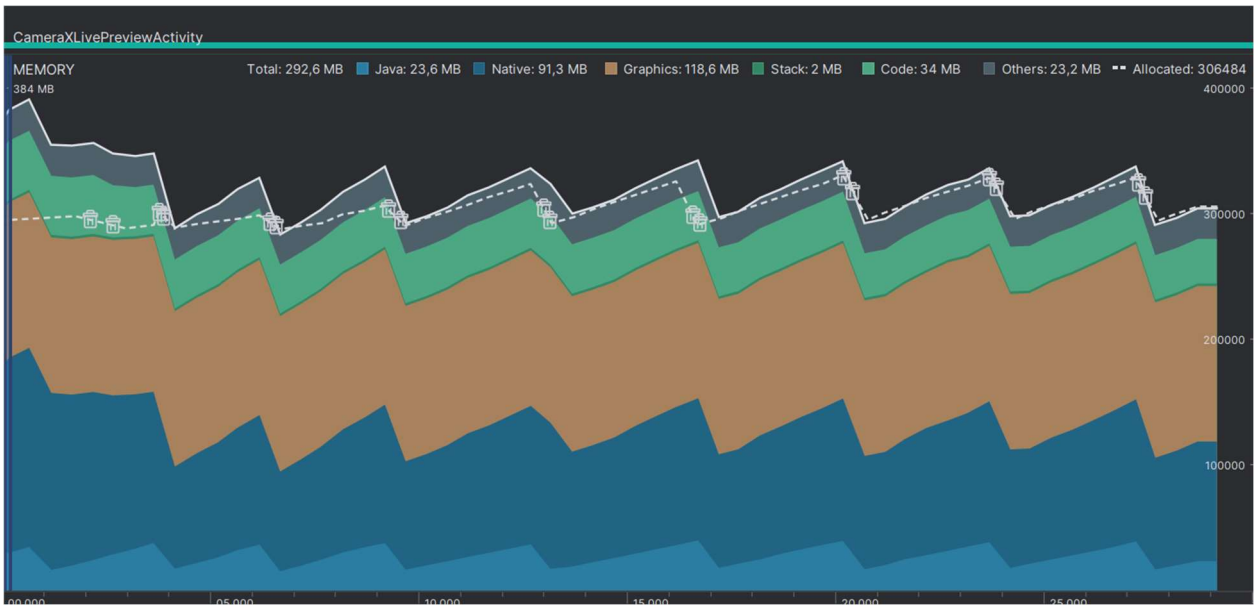


Рисунок 4.5 – Використання RAM для Firebase

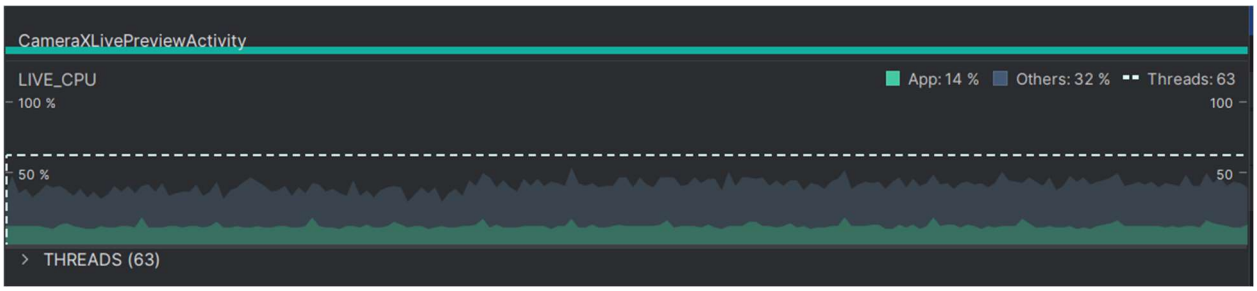


Рисунок 4.6 – Використання CPU для Firebase

Під час експерименту були зібрані дані щодо концентрації користувачів та часу розрахунку для двох бібліотек: Mediapipe Face Mesh та Firebase Face Detection. Всі дані впорядковані та структуровані для порівняння ефективності роботи бібліотек. Для кожної з бібліотек були розраховані такі значення:

– середнє значення концентрації;
– мінімальне значення концентрації;
– максимальне значення концентрації;
– середній час розрахунку;
– мінімальний час розрахунку;
– максимальний час розрахунку;
– кількість кадрів.

Результати вимірювань наведено у таблиці 4.1.

Таблиця 4.1 – Результати вимірювань для Mediapipe Face Mesh та Firebase Face Detection

Тип	Mediapipe Face Mesh	Firebase Face Detection
Середнє значення Концентрації (%)	73,55056861	87,41710648
Мінімальне значення Концентрації (%)	13,84383477	40,98079462
Максимальне значення Концентрації (%)	99,75098351	99,18626342
Середній час розрахунку (мс)	32,68666667	101,9453416
Мінімальний час розрахунку (мс)	20	91
Максимальний час розрахунку (мс)	70	164
Кількість кадрів	1801	806

Дослідження показало суттєві відмінності у використанні системних ресурсів між бібліотеками Mediapipe Face Mesh та Firebase Face Detection. Для

Mediapipe середнє споживання оперативної пам’яті (RAM) становить 650 МБ, що значно перевищує показники Firebase Face Detection, де RAM використовувалася в обсязі 280 МБ. Це пояснюється тим, що Mediapipe виконує більш деталізований аналіз обличчя, відстежуючи велику кількість ключових точок. Аналогічна тенденція спостерігається і для завантаження центрального процесора (CPU): Mediapipe вимагає 26%, тоді як Firebase – лише 14%.

Таким чином, Firebase Face Detection виявилася менш ресурсоємною, що робить її привабливішим варіантом для пристроїв із обмеженими ресурсами або задач, які не потребують високої деталізації аналізу.

Порівняння показників концентрації свідчить про вищі середні значення для Firebase Face Detection (87,42%) порівняно з Mediapipe Face Mesh (73,55%). Також Firebase демонструє кращі мінімальні значення концентрації (40,98% проти 13,84% для Mediapipe), що свідчить про більш стійке визначення навіть у складних умовах. Максимальні значення концентрації для обох бібліотек дуже близькі: Mediapipe – 99,75%, Firebase – 99,19%, що вказує на порівнянну здатність обох бібліотек визначати максимальний рівень уваги.

Це може пояснюватися тим, що Firebase використовує менш складний, але стабільніший підхід до аналізу, що зменшує ймовірність помилок в умовах швидких змін у положенні голови або відкритості очей.

Одна з ключових переваг Mediapipe – значно швидший час обробки кадру. Середній час розрахунку для Mediapipe складає 32,69 мс, у той час як для Firebase цей показник становить 101,95 мс, що більше ніж утричі перевищує час Mediapipe. Мінімальні та максимальні значення часу також свідчать на користь Mediapipe: 20 мс проти 91 мс мінімального часу і 70 мс проти 164 мс максимального часу відповідно.

Це вплинуло на загальну кількість оброблених кадрів. Mediapipe змогла обробити 1801 кадр, у той час як Firebase – лише 806 кадрів за однаковий час тестування. Mediapipe демонструє високу продуктивність, що робить її більш

придатною для завдань, які потребують швидкого аналізу та реакції в реальному часі.

Також аналізуючи графіки зміни часу вимірювання для Mediapipe Face Mesh (рис. 4.7) та Firebase Face Detection (рис 4.8) можна зробити висновок, що Mediapipe показує більш стабільну швидкість обробки так, як має менше піків збільшення часу, а саме має 6 значних відхилення на наборі даних в 1801 кадр. В той час, як Firebase Face Detection має 20 піків збільшення часу в наборі який складається з 785 кадрів.

Результати експерименту показують, що Mediapipe Face Mesh має перевагу у швидкості обробки кадрів та кількості оброблених даних, що робить її кращим вибором для задач, де важлива швидкодія та деталізація. Однак вона значно більш ресурсоємна, що може бути критичним фактором для пристроїв із обмеженими ресурсами.

Firebase Face Detection, у свою чергу, демонструє більшу стабільність у визначенні концентрації та значно менше використання ресурсів, що робить її придатною для менш інтенсивних завдань, де ключову роль відіграють енергоефективність та простота реалізації.

Firebase Face Detection, у свою чергу, демонструє більшу стабільність у визначенні концентрації та значно менше використання ресурсів, що робить її придатною для менш інтенсивних завдань, де ключову роль відіграють енергоефективність та простота реалізації.

Ці результати дозволяють зробити висновок, що вибір бібліотеки залежить від конкретного сценарію використання. Mediapipe ідеально підходить для задач реального часу з високими вимогами до деталізації, тоді як Firebase більш ефективна в умовах обмежених ресурсів або задач зі зниженими вимогами до продуктивності.

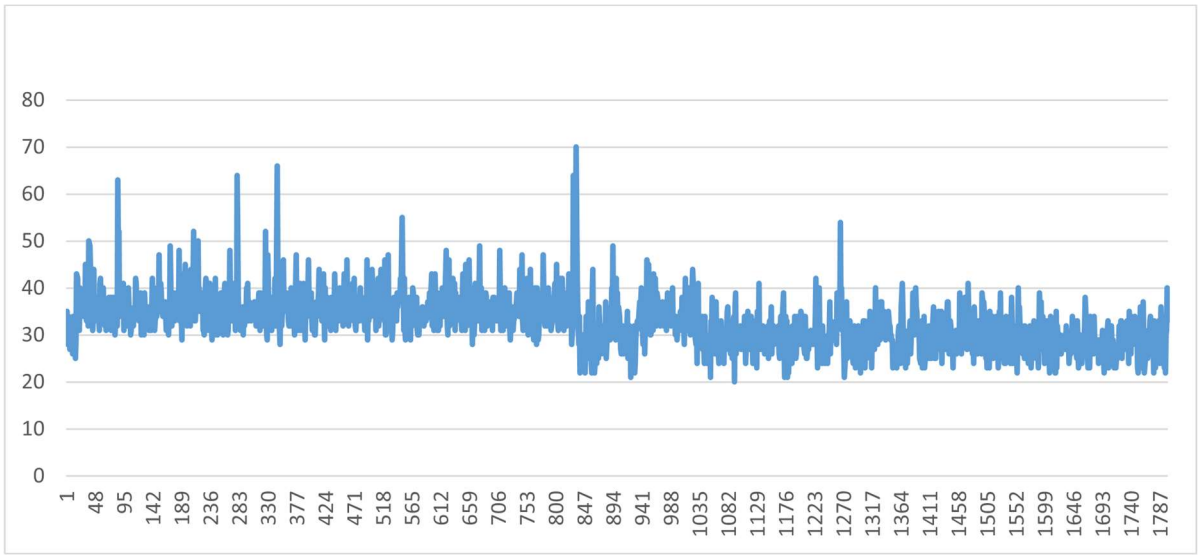


Рисунок 4.7 – Графік зміни часу вимірювання в Mediapipe Face Mesh, по осі x – номер кадру, по осі y – час (в мс)

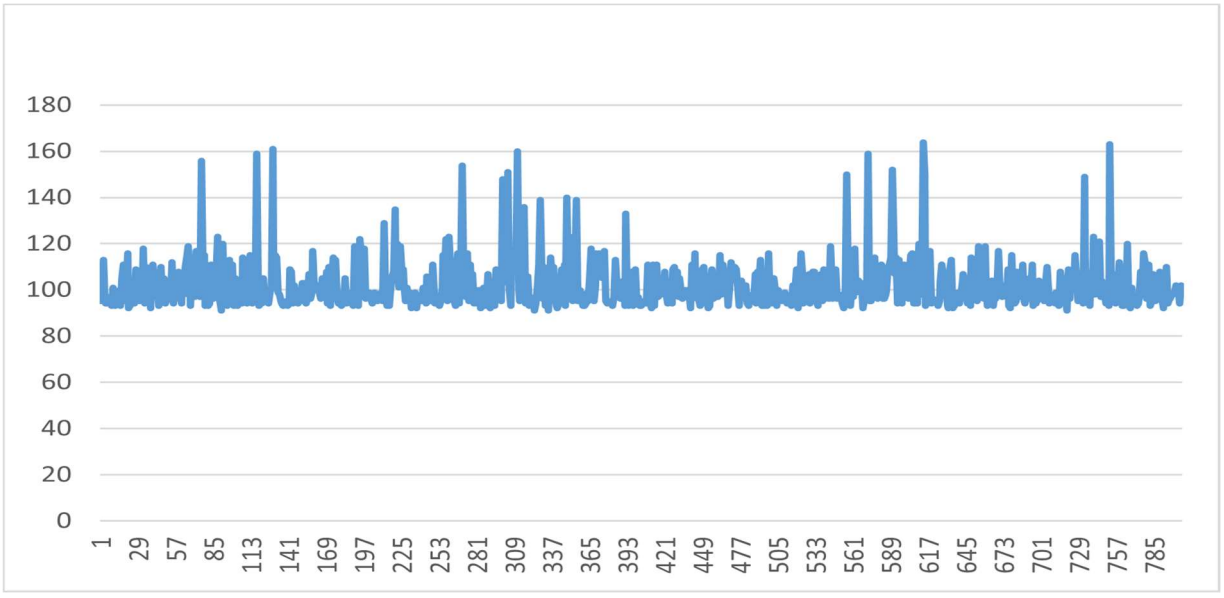


Рисунок 4.8 – Графік зміни часу вимірювання в Firebase Face Detection, по осі x – номер кадру, по осі y – час (в мс).

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було досліджено сучасні методи моніторингу концентрації людини, а також розроблено систему, яка поєднує використання нейронних мереж і мобільних платформ для оцінки фізіологічних параметрів.

Проведено аналіз фізіологічних параметрів, які корелюють із рівнем зосередженості, таких як рухи очей, міміка, частота миготіння та положення голови. Виявлено, що найперспективнішими маркерами для мобільних платформ є рухи голови та відсоток відкриття очей, які можуть бути оцінені за допомогою вбудованих камер смартфонів.

Створено мобільний додаток на платформі Android, який інтегрує дані з камери смартфона та нейронних мереж для реального часу аналізу концентрації. Проведені тести за допомогою Android додатка на оцінку концентрації підтвердили, що система забезпечує високу точність виявлення стану уваги і швидкість обробки даних із затримкою в середньому менш ніж 100мс.

Проведено порівняння бібліотек MediaPipe та Firebase ML Kit. MediaPipe показала кращі результати в режимі реального часу завдяки високій оптимізації, тоді як Firebase ML Kit відповідно до графіків заванатженності системи демонструє меншу завантаженість CPU та RAM.

Таким чином, можна зробити висновки, що впровадження системи оцінки концентрації людини на платформі Android використовуючи сучасні моделі нейронних мереж можливе за умов правильної інтеграції та оптимізації. В подальшому розроблена система може бути використана, для контролю концентрації людини на робочому місці, що зменшує вірогідність нещасних випадків.

Відповідно до проаналізованих результатів наведених в розділі 4.7 можна відмітити ефективність моделей нейронних мереж MediaPipe та Firebase ML Kit. З урахуванням отриманих даних можна зробити висновок, що Mediapipe для пристроїв, які мають обчислювальні ресурси тоді як Firebase більш ефективна в умовах обмежених ресурсів або задач зі зниженими вимогами до продуктивності.

Для подальшого вдосконалення системи можна розширити базу даних для тренування нейронних мереж, включивши користувачів із різними поведінковими моделями. Також впровадити адаптивні моделі, які динамічно налаштовуються на особливості конкретного користувача в реальному часі. Додати підтримку інших фізіологічних маркерів, таких як теплові зміни на обличчі або електродермальна активність, для більш точного визначення рівня концентрації.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. World Health Organization [Електронний ресурс] / WHO. – Режим доступу: www / URL: https://www.who.int/ru/health–topics/cardiovascular–diseases#tab=tab_1/– 01.06.2023 р. – Загол. з екрану.

2. Носик, Д. О. (2024). Методи визначення напрямку погляду за допомогою нейронних мереж. Тези доповідей міжнародної науково-практичної конференції "Сучасні підходи в науці та освіті", 20 грудня 2024 року. URL: https://archive.liga.science/index.php/conference-proceedings/issue/view/ukr-20.12.2024/111 (дата звернення: 25 грудня 2024 року).

3. Shao Liming, Ling Meining, Yan Ying, Xiao Guangnian. Research on Vehicle–Driving–Trajectory Prediction Methods by Considering Driving Intention and Driving Style. 2024. Т. 16, №19. С. 8417. DOI: 10.3390/su16198417. URL: https://www.researchgate.net/publication/384424848_Research_on_Vehicle–Driving–Trajectory_Prediction_Methods_by_Considering_Driving_Intention_and_Driving_Style (дата звернення: 13.10.2024).

4. Konecki Krzysztof, Płaczek Aleksandra, Tarasiuk Dagmara. Concentration as a Problem of the Socialized Mind. 2023. С. 199–228. DOI: 10.4324/9781003424284–10. URL: https://www.researchgate.net/publication/372634287_Concentration_as_a_Problem_of_the_Socialized_Mind (дата звернення: 13.10.2024).

5. Harvard Medical School. Focus on concentration [Електронний ресурс]. URL: https://www.health.harvard.edu/mind–and–mood/focus–on–concentration (дата звернення: 13.10.2024).

6. Kugler Karl G., Hackl Werner, Mueller Laurin Aj, Fiegl Heidi. The Impact of Sample Storage Time on Estimates of Association in Biomarker Discovery

Studies // Journal of Clinical Bioinformatics. 2011. Т. 1, №1. DOI: 10.1186/2043–9113–1–9. URL: https://www.researchgate.net/publication/51481371_The_Impact_of_Sample_Storage_Time_on_Estimates_of_Association_in_Biomarker_Discovery_Studies (дата звернення: 13.10.2024).

7. Бодянський Є. В., Руденко О. Г. Штучні нейронні мережі: архітектури, навчання, застосування : [монографія] / Є. В. Бодянський, О. Г. Руденко. – Харків : ТЕЛЕТЕХ, 2004. – 372 с. : іл. – ISBN 966–95416–2–2. – 40,00.

8. Purwono Purwono, Ma'arif Alfian, Rahmaniar Wahyu, Imam Haris. Understanding of Convolutional Neural Network (CNN): A Review // International Journal of Robotics and Control Systems. 2023. Т. 2, №4. С. 739–748. DOI: 10.31763/ijrcs.v2i4.888. URL: https://www.researchgate.net/publication/367157330_Understanding_of_Convolutional_Neural_Network_CNN_A_Review (дата звернення: 13.10.2024).

9. MediaPipe Solutions guide [Електронний ресурс]. URL: https://ai.google.dev/edge/mediapipe/solutions/guide (дата звернення: 13.10.2024).

10. TensorFlow [Електронний ресурс]. URL: https://www.tensorflow.org/ (дата звернення: 13.10.2024).

11. Firebase ML Kit. Detect faces [Електронний ресурс]. URL: https://firebase.google.com/docs/ml–kit/android/detect–faces (дата звернення: 13.10.2024).

12. dlib [Електронний ресурс]. URL: http://dlib.net/ (дата звернення: 13.10.2024).

13. OpenCV [Електронний ресурс]. URL: https://opencv.org/ (дата звернення: 13.10.2024).

14. Darwin I. F. Android Cookbook. 2nd ed. – O'reilly Media. – 2017. – 664 с.

15. Android Distribution Chart [Електронний ресурс]. URL: https://composables.com/android–distribution–chart (дата звернення: 13.10.2024).

16. Bjegojevic Bojana, Leva Maria Chiara, Cromie Sam D., Balfe Nora. Physiological Indicators for Real–Time Detection of Operator's Attention // European Conference on Safety and Reliability (ESREL 2022), Дублін, Ірландія, 2022. DOI: 10.3850/978–981–18–5183–4_J01–05–149–cd. URL: https://www.researchgate.net/publication/363567662_Physiological_Indicators_for_Real–Time_Detection_of_Operator's_Attention (дата звернення: 13.10.2024).

17. Носик, Д. О. (2024). Порівняльний аналіз підходів до використання багатопоточності для Android розробки на Kotlin. Тези доповідей міжнародної науково-практичної конференції "Сучасні підходи в науці та освіті", 20 грудня 2024 року. URL: https://archive.liga.science/index.php/conference-proceedings/issue/view/ukr-20.12.2024/111 (дата звернення: 25 грудня 2024 року).