Київський національний університет імені Тараса Шевченка
Факультет комп'ютерних наук та кібернетики
Кафедра математичної інформатики

«До захисту допущено»
Завідувач кафедри
В.М.Терещенко _____
(підпис)

«___ » _____20 __р.

Дипломна робота

на здобуття ступеня бакалавра

за спеціальністю 122 Комп'ютерні науки

на тему:

Мобільна медична діагностика

Виконала студентка 4-го курсу
Гаврилюк Оксана Михайлівна _____
(підпис)

Науковий керівник:
професор, доктор фіз-мат наук
Терещенко Василь Миколайович _____
(підпис)

Засвідчую, що в цій курсовій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Київ – 2021р.

РЕФЕРАТ

Обсяг роботи 62 сторінки, 40 ілюстрацій, 35 джерел посилань та 18 таблиць.

КОМП’ЮТЕРНИЙ ЗІР, МЕДИЧНА ІНФОРМАТИКА, РОЗПІЗНАВАННЯ ОБЛИЧЧЯ, CIELAB, ANDROID

Об'єктом дослідження є медична діагностика за фотографією.

Предметом дослідження є виявлення і класифікація видимих симптомів лиця.

Метою роботи є створення додатку як практичного інструмента для попереднього діагнозу.

Методи дослідження базуються на алгоритмах комп’ютерного зору, аналізі існуючих методів діагностики.

Основними засобами дослідження є мова програмування Python, бібліотеки Dlib, OpenCV та Android Studio.

У роботі було розглянуто метод медичної діагностики, який базується на створенні шкали відхилень та адаптації цієї шкали під користувача. Були приведені алгоритми локалізації лиця та виявлення ключових рис обличчя. Методи були реалізовані на мові Python після чого було створено додаток за допомогою Android Studio.

В роботі розглядається спосіб, який на відміну від вже існуючих методів, фокусується на попередній діагностиці за видимими симтомами обличчя за створеною шкалою відхилень, яка є адаптовною до користувача та приводиться програмна реалізація.

Результати цього дослідження можуть бути застосовані як допоміжний засіб при медичній діагностиці та як складова частина системи догляду за здоров’ям і для підвищення ефективності процесу лікування.

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

CNN – Convolutional Neural Network

NLP – Natural Language Processing

ЕКГ – Електрокардіографія

HOG – Histogram of Oriented Gradient

SVM – Support Vector Machine

MTCNN – Multi-Task Cascaded Convolutional Neural Network

ВСТУП

Актуальність роботи та підстави для її виконання. Фотографії обличчя пацієнтів можуть бути дуже інформативними при діагностиці та лікуванні. Відомо, що професійні лікарі, які працюють у своїй сфері десятки років, можуть багато сказати тільки по обличчю людини. При цьому спеціалісти покладаються на власний досвід, який насправді є дуже обмеженим. Тут на допомогу приходять комп'ютери. Вони можуть містити гігабайти інформації та бути чутливими до крихітних змін. Зазвичай для діагностики використовують набір зображень, на якому навчають штучний інтелект. Можна виконувати пошук наявності чи відсутності симптомів певних захворювань.

Для виявлення симптоматичних ознак стану здоров'я людини за проявом змін певних характеристик обличчя необхідно локалізувати ключові риси обличчя здорової людини та відслідковувати їх зміну та прояв у випадку тієї чи іншої хвороби. Потрібно знайти певні закономірності зміни характеристик у відповідності з діагностуванням відповідної хвороби. Для цього необхідно провести порівняльний аналіз хворих і здорових людей та створити механізм виявлення діагнозу та реалізувати його в додатку. Для знаходження ключових рис обличчя використовують алгоритми комп'ютерного зору.

Оцінка сучасного стану об'єкта розробки. Для пошуку оптимальних рішень, сформульованої вище проблеми, було проведено ряд досліджень. Хен Ким, Со Юнг Ким, Юнг Хо Ким і Парк Кванг [1] представили систему діагностики паралічу лицевого нерву для мобільних пристроїв. Джейн Рейлі Деланоу і Томас Вард [2] запропонували систему, засновану на алгоритмах комп'ютерного зору, для автоматичного вимірювання здатності пацієнтів посміхатися з метою реабілітаційної терапії. Фелікс Аулав, Джудіт Мек, Ліндсі МакДональд та Теренс Ленг [3] розробили додаток для діагностики жовтяниці у новороджених дітей. Шень Лінь, Джиган Лі, Боуен Фу та інші

дослідники створили інструмент для первинної діагностики серцевих захворювань[4]. Інші застосунки призначені для дерматологічних цілей, таких як лікування акне та алергічних висипань [5].

Отже, як можна побачити, не існує додатку, який би об’єднував у собі діагностику більшості видимих симптомів обличчя. Та чи існує спосіб для його створення за відсутності великих наборів фотографій?

Для того, щоб відповісти на це запитання, був проведений пошук робочих алгоритмів, які базуються на статистичних даних та не прив’язуються до конкретних симптомів. Було знайдено роботу Куан Вана та Цзебо Ло [6], які класифікують захворювання, орієнтуючись на шкалу відхилень.

Мета й завдання роботи. Метою дипломної роботи є розробка методів діагностики зовнішніх та деяких внутрішніх хвороб людини за змінами характеристик обличчя.

Можливі сфери застосування. Результати цього дослідження можуть бути застосовані як практичний інструмент для попереднього діагнозу. Воно також може бути використане як складова частина системи догляду за здоров’ям і підвищувати ефективність процесу лікування. Важливо запримітити, що алгоритми описані в роботі, мають бути застосовані як допоміжний засіб при медичній діагностиці, а не як повна її заміна.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ 3

ВСТУП 4

РОЗДІЛ 1. ПОСТАНОВКА ПРОБЛЕМИ 8

1.1. Розвиток медичної інформатики 8

1.2. Обмеженість набору даних 9

1.3. Порівняння з існуючими методами 9

РОЗДІЛ 2. ОГЛЯД ІСНУЮЧИХ АЛГОРИТМІЧНИХ ІНСТРУМЕНТІВ ДЛЯ ДІАГНОСТИКИ 13

2.1. Методи локалізації обличчя на зображенні 13

2.2. Локалізація ключових рис обличчя 16

2.3. Простір кольорів CIELAB 18

2.4. Метод виявлення контурів 20

РОЗДІЛ 3. ПОПЕРЕДНЯ ДІАГНОСТИКА ВИДИМИХ СИМПТОМІВ ОБЛИЧЧЯ 23

3.1. Загальний опис алгоритму 23

3.2. Діагностика мішків під очима 24

3.3. Діагностика синюватості губ 26

3.4. Діагностика червонуватості очей 27

3.5. Діагностика набряків та асиметрії обличчя 28

3.6. Діагностика опущених кутиків губ 31

3.7. Діагностика сухості та лущення губ 32

РОЗДІЛ 4. МЕДИЧНИЙ ЩОДЕННИК 35

4.1. Адаптація під користувача 35

4.2. Медичний щоденник зміни кольору обличчя 36

4.3. Медичний щоденник червонуватості обличчя 39

4.4. Медичний щоденник випадіння брів 43

4.5. Практичні рекомендації 46

РОЗДІЛ 5. ПРОГРАМНА РЕАЛІЗАЦІЯ 48

5.1. Адаптація Python під Android Studio 48

5.2. Результати роботи програми 48

ВИСНОВКИ 52

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ 53

Додаток А 57

РОЗДІЛ 1. ПОСТАНОВКА ПРОБЛЕМИ

1.1. Розвиток медичної інформатики

Розвиток медичної інформатики припав на 50-ті роки XX століття спочатку в США, потім в Європі та інших розвинених східних країнах. Реальне застосування інформатики у сфері охорони здоров'я тісно пов'язане з розвитком обробки даних. Поєднуючи медицину з технологіями планувалось створення методології для кращої та швидшої діагностики та ефективного лікування[7].

Загалом розвиток можна поділити на такі періоди:

1) Перший період з 1955 по 1965 рр. В основному характеризується експериментами та вивченням нових технологій в медицині.
2) Другий період з 1965 по 1975 рр. Під час цього періоду були введені більш ефективні способи автоматичної обробки даних.
3) Третій період з 1975 по 1985 рр. відзначається значним прогресом розвитку комп'ютерної техніки, що спричинило її дешевизну, що приводить до дуже інтенсивного розвитку інформаційних систем у сфері охорони здоров'я. Збільшується інтерес мед працівників до області медичної інформатики.
4) Четвертий період з 1985 по 1995 рр. Розвиток інформатики в сфері охорони здоров'я відповідав високим стандартам завдяки новому способу обробки та стандартизації знань. Були проведені інтенсивні дослідження по вдосконаленню штучного інтелекту, які включали в себе розробку і застосування експертної системи в медичній діагностиці та терапії. Штучний інтелект був введений у якості окремої дисципліни медичної інформатики і почав використовуватись на практиці.
5) П'ятий період розпочався в 1995 р. та триває по наш час. Зараз важко відділити розвиток медичної інформатики та розвиток

комп’ютерних технологій. Відбувається значне розширення методик застосування технологій у медицині та поширення цих знань у маси.

1.2. Обмеженість набору даних

Задача подолання обмеженості набору даних є основною в межах цієї дипломної роботи. Для ефективної діагностики необхідно застосовувати алгоритми, які є адаптивними, легко інтерпретуються та не потребують великої кількості тренувальних даних людей із діагностованими видимими симптомами обличчя. Також необхідно зібрати тренувальні дані, провівши аналіз наявності чи відсутності видимих симптомів обличчя вручну. Потрібно провести перевірку чіткості та однозначності у наявності чи відсутності певних симптомів.

Ще однією важливою задачею є гнучкість побудованої моделі до певного користувача. Необхідно забезпечити зміну шкали діагностики під індивідуальні особливості кожної людини. Зовнішність кожної людини є унікальною, тому при класифікації потрібно враховувати можливі відхилення від середнього і типового для більшості людей. Потрібно створити зручний та швидкий алгоритм для такої модифікації.

1.3. Порівняння з існуючими методами

Face2Gene

Додаток Face2Gene створений допомогти лікарям діагностувати рідкісні захворювання (в даному випадку дисморфізм рис обличчя). Фотографії пацієнтів аналізуються за допомогою алгоримів локалізації рис обличчя і глибокого навчання для виявлення фенотипів, які корелюють з рідкісними генетичними захворюваннями. Платформа в даний час доступна тільки навченим клініцистам для запобігання помилковому діагнозу і підтримує більше 7500 захворювань [8].

VisualDX

Додаток призначений для надання допомоги фахівцям в прийнятті більш точних клінічних рішень на основі великої бази даних зображень з високою роздільною здатністю і інтуїтивно зрозумілих функцій пошуку[9]. VisualDx використовує велику ретельно підібрану бібліотеку медичних зображень безлічі захворювань, навіть найрідкісніших і за допомогою алгоритмів машинного навчання надає допомогу терапевтам для розпізнавання, розуміння та лікування шкірних захворювань. Використовуючи штучний інтелект і машинне навчання, додаток аналізує тип ураження, а потім задає прості питання, щоб швидко перейти до диференціальної діагностики [10].

Мобільний додаток попередньої діагностики захворювань

Додаток являє собою систему рекомендацій, призначену для встановлення попереднього діагнозу пацієнта і нагадування про необхідність прийому ліків. Система рекомендацій виконує такі основні функції: попередній медичний діагноз за обраними симптомами; формування нагадування про прийом ліків; формування анамнезу прийому ліків [11].

Діагностика патологій

Прагнучи підвищити швидкість і точність діагностики, група дослідників з Медичного центру Бет Ізраїль і Гарвардської медичної школи використали методи глибинного навчання для розпізнавання зображень для діагностики пухлин. У порівнянні з лікарями-патологами, результати дослідження показали, що показник успішності діагностики склав 92 відсотки; на чотири відсотки нижче, ніж у лікарів. Однак якщо об'єднати алгоритм і результати, отримані людиною, точність склала 99,5% [12].

Діагностика у сфері онкології

Дослідники Стенфордського університету розробили алгоритм для діагностики раку шкіри з використанням глибокого навчання, зокрема глибоких згорткових нейронних мереж (CNN). Алгоритм був навчений виявляти рак шкіри або меланому з використанням 130 000 зображень

шкірних пошкоджень, які представляли понад 2 000 різних захворювань. Стенфордський алгоритм глибинного навчання був протестований за участю 21 сертифікованого дерматолога, які переглянули 370 зображень і їх запитали, чи будуть вони проводити біопсію або лікування або заспокоють пацієнта. Результати показали, що алгоритм мав ті ж можливості, що і 21 дерматолог, у визначенні найкращого алгоритму дій для всіх зображень [12].

Babylon Health

Babylon Heath пропонує чат-бот під назвою Ask Babylon, який, як вони стверджують, може допомогти користувачам отримувати інформацію про свої симптоми за допомогою обробки природної мови (Natural Language Processing,NLP). Babylon Health стверджує, що користувачі можуть спочатку ввести свої симптоми в чат-бот, потім чат-бот задасть серію питань, пов'язаних з цими симптомами, щоб зібрати про них більш конкретну інформацію. Коли користувач відповідає на ці питання, чат-бот порівнює відповіді зі своєю базою даних. Якщо чат-бот зіставляє симптоми користувача з інформацією в базі даних Bablyon, то він надає користувачеві інформацію про його симптоми. Потім користувач може звернутися за порадою до лікаря у вікні чату бота. Якщо чат-бот не знаходить інформацію про симптоми користувача або знайдена інформація є особливо серйозною та загрозливою для здоров’я він порекомендує користувачеві або звернутися за порадою до свого лікаря, або проконсультуватися з одним з лікарів Babylon в штаті. Крім того, користувачі можуть вручну вводити свою медичну інформацію в систему Bablyon, щоб поліпшити рекомендації чат-бота для них [12].

AliveCor’s Kardia App

Kardia - додаток компанії AliveCor є додатком для ЕКГ для Apple Watch і смартфонів, яке дозволяє користувачам Apple Watch відстежувати власну частоту серцевих скорочень, використовуючи алгоритми машинного зору.

Під Apple Watch встановлений датчик, який дозволяє додатку Kardia підсвітлювати шкіру користувача і передавати візуальну інформацію в модель машинного навчання додатку Kardia [12].

Cognoa

Cognoa розробили платформу для допомоги в скринінгу поведінки дітей. Клієнти можуть заповнити інформацію про здоров'я і поведінку своєї дитини, потім додаток Cognoa відправляє результати скринінгу, в яких прогнозується рівень ризику затримок у розвитку або розвитку аутизму у їх дитини. Cognoa стверджує, що діагностика проводиться за допомогою машинного навчання [12].

Healthy.io

Healthy.io пропонує програму чат-бот для домашнього тестування сечі, який використовує комп'ютерний зір для сканування і порівняння тест-смужок. З панелі чату користувач може відкрити екран сканування, схожий на камеру, і потримати телефон над смугою. Через кілька секунд після сканування чат-бот видає користувачеві результати, які здаються найбільш імовірними на основі збігу кольорів. Він також дає користувачеві поради, наприклад, коли і чи варто йому звертатися за медичною допомогою. За заявою компанії, додаток також пропонує панель інструментів, на якій користувачі можуть бачити особливості результатів і наявність певних токсинів, які можуть бути в їхньому організмі [12].

В дипломній роботі розглядатиметься метод, який дозволить проводити попередню діагностику видимих симптомів обличчя, створюючи шкалу відхилень та адаптовуючись до особливостей зовнішньості кожного користувача. Буде створений додаток, який дозволить легко проводити діагностику та моніторити зміни і надавати за ними рекомендації.

РОЗДІЛ 2. ОГЛЯД ІСНУЮЧИХ АЛГОРИТМІЧНИХ ІНСТРУМЕНТІВ ДЛЯ ДІАГНОСТИКИ

2.1. Методи локалізації обличчя на зображенні

Існує два найбільш поширених способи локалізації обличчя на зображенні: за допомогою ознак Хаара [13] та гістограм направлених градієнтів [14, 15].

Метод ознак Хаара

Перший спосіб був розроблений Полом Віолою та Майклом Джонсом у 2001 році [16]. Він може бути застосований для пошуку різних об’єктів на фото, але використовується в основному для пошуку лиць.

Ознаки Хаара – ознаки цифрових зображень, які використовують при розпізнаванні образів. Ці ознаки складається з суміжних прямокутних областей. Вони позиціонуються на зображенні, далі підсумовуються інтенсивність пікселів областей, після чого обчислюється різниця цих сум. Ця різниця і буде значенням певної ознаки, певного розміру, певним чином позиціонованого на зображенні. Для прикладу розглянемо базу даних з людськими обличчями. Спільним для всіх зображень є той факт, що область в районі очей є темнішою за область щік. Отже, спільною ознакою Хаара для обличь будуть два суміжних прямокутних регіони, які лежать на очах та щоках. Далі використовуються **модель Адабуст та каскадні класифікатори**. Головною особливістю використання ознак Хаара є найбільша, у порівнянні з іншими ознаками, швидкість.

Метод гістограм направлених градієнтів.

HOG (Histogram of Oriented Gradient, що в перекладі гістограма направлених градієнтів) – це дескриптор ознак, який підраховує напрямки градієнтів в локальних точках зображення. Алгоритм розділяє зображення на маленькі квадрати, вираховує гістограму орієнтованих градієнтів для кожного квадрата, нормалізує результат та повертає дескриптор для кожного

квадрата. Розташування цих квадратів на зображеннях може далі бути використане для побудови моделі, наприклад SVM [17].

Linear SVM (Support Vector Machine, в перекладі метод опорних векторів) - це метод аналізу даних для класифікації та регресійного аналізу за допомогою моделей з керованим навчанням з пов'язаними алгоритмами навчання. Для заданого набору тренувальних зразків, кожен із яких відмічений як належний до однієї чи іншої з двох категорій, алгоритм тренування будує модель, яка відносить нові зразки до однієї чи іншої категорії, роблячи це бінарним лінійним класифікатором. Модель є представленням зразків як точок у просторі, відображених таким чином, що зразки з окремих категорій розділено чистою прогалиною, яка є щонайширшою. Нові зразки тоді відображуються до цього ж простору, й робиться передбачення про їхню належність до категорії на основі того, на який бік прогалини вони потрапляють.

Методи глибинного навчання. Існує ряд методів глибокого навчання для розпізнавання облич на фото. Однією з найбільш популярних моделей є багатозадачна каскадна згорткова нейронна мережа (Multi-Task Cascaded Convolutional Neural Network), скорочено MTCNN, описана Кайпен Джаном у 2016 році у роботі «Одночасне виявлення і вирівнення облич з використанням каскадних загорткових мереж». MTCNN користується популярністю, тому що в ній були отримані найсучасніші результати по ряду наборів контрольних даних, а також тому, що вона здатна також розпізнавати інші риси обличчя, такі як очі і рот. Мережа використовує каскадну структуру з трьома етапами: спочатку зображення масштабується до різних розмірів (так звана піраміда зображень), потім перша модель (Proposal Network або P-Net) виділяє кандидатні області для локалізації облич, друга модель (Refine Network або R-Net) фільтрує обмежуючі прямокутники, а третя модель (Output Network або O-Net) пропонує орієнтовні орієнтири на обличчі. Ці три моделі не пов'язані безпосередньо, замість цього вихідні дані попереднього етапу подаються в якості вхідних даних для наступного етапу.

Це дозволяє виконувати додаткову обробку між етапами. З кожний етапом збільшується складність мережі (рис. 1) [21].

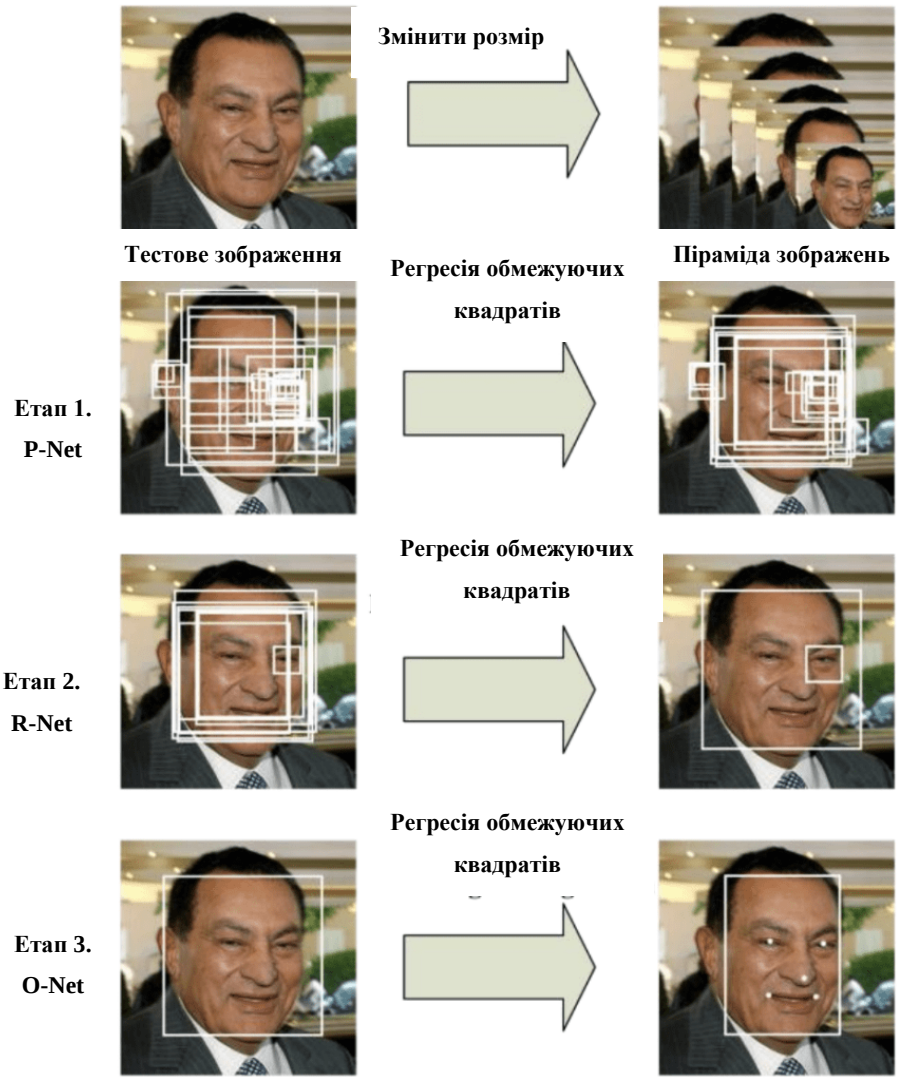

Рис. 1. Структура MTCNN

У роботі було використано метод гістограм направлених градієнтів, як достатньо швидкий та простий у реалізації метод, що полегшило створення застосунку діагностики. Більш складні методи глибинного навчання варто застосовувати при виявленні облич та їхніх рис з різних кутів та на різній відстані. Також метод гістограм направлених градієнтів дозволяє відслідковувати зміни освітленості пікселів, що означає, що для однієї людини ці зміни будуть досить схожими, незалежно від освітленості.

Оскільки у даній роботі діагностика проводиться на зображення обличчя, розташованих фронтально, то підвищення складності моделі не дає значних

покращень, а навпаки сповільнює діагностику, що є не бажаним при використанні додатку. Обмеженість наборів даних не дало змоги скористатися нейронними мережами для діагностики, тому було прийнято рішення робити усі підрахунки для діагностики на локалізованих рисах обличчя.

2.2. Локалізація ключових рис обличчя

Після отримання області обличчя стає можливим застосування механізмів виявлення орієнтирів обличчя. Всі методи, які існують для цього, намагаються локалізувати та позначити такі частини: рот, праву та ліву брови, праве та ліве око, ніс та щелепу. Для виявлення рис обличчя використовуються так звані орієнтири обличчя, які призвані локалізувати і представляти певні області лиця, такі як: очі, брови, ніс, рот та щелепу. Орієнтири обличчя успішно використовуються для вирівнювання обличчя, оцінки пози голови, обміну обличчями, виявлення підморгування та багато іншого [18].

Виявлення орієнтирів обличчя є похідним проблеми виявлення форми. Адже, отримавши вхідне зображення, модель виявлення форми намагається локалізувати ключові точки, які нас цікавлять, всередині цієї форми. В контексті орієнтирів обличчя головною метою є виявлення важливих рис обличчя з використанням методів виявлення форми. Існує багато реалізацій локалізації ключових рис обличчя. В роботі була використана робота “One Millisecond Face Alignment with an Ensemble of Regression Trees” [19]. Ця робота стосується проблеми вирівнювання лиць на зображенні. В ній демонструються як сукупність регресійних дерев можна використовувати для оцінки орієнтирів обличчя напряму з множини пікселів, досягнувши при цьому високої точності в реальному часі. Автори презентували загальний фреймворк, який базується на градієнтному методі для оптимізації сум квадратів похибок і роботи за відсутності частини вхідних даних. В книзі описується як правильне використання апріорних оцінок допомагає ефективно виділити риси обличчя. Також розглядаються різноманітні

способи додавання додаткових обмежень для запобігання перенавчання. Крім цього, проводиться аналіз впливу розміру тренувального набору даних на точність діагностики.

Метод використовує:

1) Навчальний набір даних із означеними орієнтирами лиця на фотографіях. Ці фото є означеними вручну і уточнюють конкретні координати x, y для регіонів, які оточують кожну структуру обличчя.
2) Ймовірності відстаней між парами вхідних пікселів

Отримавши навчальний набір даних, набір регресійних дерев навчають оцінювати розташування орієнтирів обличчя одразу з пікселів, тому ні про яке вилучення ознак мова не йде. Результатом цих маніпуляцій є детектор, який можна використовувати в реальному часі та з якісними прогнозами. Індекси 68 координат обличчя можна візуалізувати як показано нижче, рис. 2:

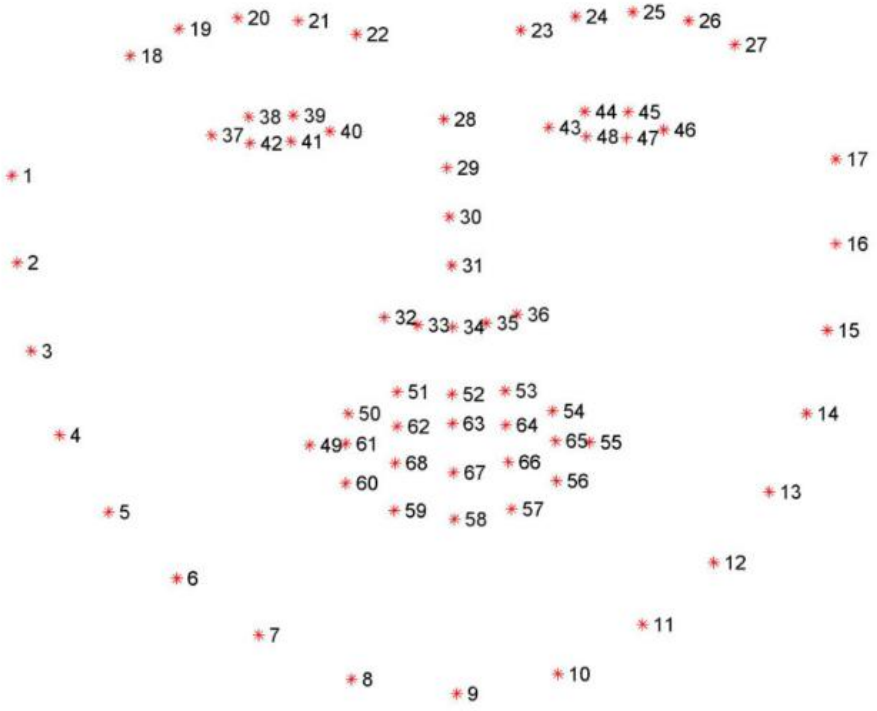

Рис. 2. Візуалізація точок ознак обличчя

2.3. Простір кольорів CIELAB

Для діагностики нам знадобляться поняття каналів a, b простору кольорів CIELAB для рис обличчя. Канали a та b дозволять створити приблизну шкалу червонуватості й жовтуватості рис обличчя для того, щоб виявити потенційні симптоми. CIELAB – це нелінійне перетворення XYZ в координати L, a, b. Метою CIELAB було створення простору кольорів, зміна кольорів в якому буде більш лінійною з точки зору людського сприйняття (порівняно з XYZ), тобто з тим, щоб однакова зміна значень координат кольорів у різних областях простору справляло однакове відчуття зміни кольору. Таким чином, математично б корегувалась нелінійність сприйняття кольору людиною. Це виходить за рахунок однакової перцептивної зміни для однакових змін координат L, a, b[22].

CIELAB-модель є абстрактною, тобто незалежною від пристрою. Модель визначає колір незалежно від способу його створення чи відображення. Оскільки колір визначається трьома параметрами, простір сам по собі є тривимірним простором дійсних чисел (де ми беремо евклідову відстань між координатами), що створює нескінченну кількість можливих кольорів. На практиці простір зазвичай відображається як тривимірний простір цілих чисел і тому L, a, b значення є зазвичай абсолютними, з наперед заданого діапазону[23, 24].

Координата L відповідає за освітленість, тобто перехід від чорного кольору до білого, а за перехід від зеленого до червоного і b за перехід від синього до жовтого. Освітленість L представляє найтемніше значення при L = 0 і найсвітліше при L = 100. Так звані канали кольорів a та b будуть визначати нейтральний сірий при значеннях a = 0 та b = 0. На осі a за перехід до зеленого відбувається у напрямку від’ємних координат, а на осі b при русі у додатному напрямку. Границі максимальних і мінімальних значень залежить від заданої реалізації (наприклад D65, D50) та зазвичай значення перебувають у діапазоні від -100 до 100, чи від -128 до 127 (беззнакове 8-бітне ціле число)[23, 24].

Простір кольорів CIELAB є похідним від CIE XYZ простору кольорів, який теж є незалежним від пристроїв та визначається невід'ємними координатами X, Y, Z. При створенні CIELAB враховувалась простота переходу до L, a, b координат. Саме тому кольори CIELAB пов'язані з CIEXYZ моделлю математичними формулами[22]:

$$L^* = 116\,\mathrm{f}\left(\frac{Y}{Y_n}\right) - 16$$

$$a^* = 500\,\mathrm{f}\left(\frac{X}{X_n}\right) - \mathrm{f}\left(\frac{Y}{Y_n}\right)$$

$$b^* = 200\,\mathrm{f}\left(\frac{Y}{Y_n}\right) - \mathrm{f}\left(\frac{Z}{Z_n}\right)$$

$\mathrm{f}(t) = \begin{cases} \sqrt[3]{t}, & t > \delta^3 \\ \frac{t}{3\delta^2} + \frac{4}{29}, & \text{інакше} \end{cases}$, де $\delta = \frac{6}{29}$, індекс n відповідає на нормованість.

Для реалізації D65 з нормуванням з $Y = 100$ A_n, B_n, C_n набувають таких значень:

$$X_n = 95{,}0489$$

$$Y_n = 100$$

$$Z_n = 108{,}8840$$

А для реалізації D50 таких:

$$X_n = 96{,}4212$$

$$Y_n = 100$$

$$Z_n = 82{,}5188$$

Зворотне перетворення використовує обернену функцію f^{-1}:

$$X = X_n \mathrm{f}^{-1}\left(\frac{L^* + 16}{116} + \frac{a^*}{500}\right)$$

$$Y = Y_n \mathrm{f}^{-1}\left(\frac{L^* + 16}{116}\right)$$

$$Z = Z_n \mathrm{f}^{-1}\left(\frac{L^* + 16}{116} + \frac{b^*}{200}\right)$$

де $f^{-1}(t) = \begin{cases} t^3, t > \delta \\ 3\delta^2\left(t - \frac{4}{29}\right), \text{інакше} \end{cases}$ і де $\delta = \frac{6}{29}$

2.4. Метод виявлення контурів

Методи виявлення контурів – це математичні методи, які дозволяють знаходити неоднорідні точки на зображенні, тобто точки, в яких різко змінюється яскравість. Точки, в яких яскравість зображення різко змінюється зазвичай об'єднують в набір вигнутих лінійних сегментів – контури. Виявлення контурів є один із основних елементів в обробці зображень, області комп'ютерного зору, особливо при виявленні ключових ознак [25]. В дипломній роботі було використано метод Кенні для виявлення контурів. Метод Кенні – це метод виявлення корисної структурної інформації з різних видимих об'єктів і значного зменшення об'єму даних, які необхідно при цьому обробити. Він широко застосовується в різних системах комп'ютерного зору завдяки достатньо якісній та надійній детекції. Джон Кенні, розробник алгоритму, виявив, що вимоги до застосування виявлення контурів є досить схожими, тому рішення було розроблене, щоб задовольняти широкий діапазон ситуацій [26].

Першим кроком алгоритму є зменшення кількості шуму на зображенні, який може дати похибку при виявленні контурів. Для цього застосовується фільтр Гауса, який згладжує зображення, формула для ядра розмірами

$(2k + 1) * (2k + 1)$ наведена нижче:

$$H_{ij} = \frac{1}{2\pi\sigma^2} exp(- \frac{(i - (k+1))^2 + (j - (k+1))^2}{2\sigma^2}$$

де $1 \leq i, j \leq (2k + 1)$

Використовується фільтр розмірності 5 * 5.

Наступним кроком є застосування ядра Собеля в горизонтальному та вертикальному напрямках, щоб отримати першу похідну в горизонтальному

G_x та вертикальному G_y напрямках. З двох отриманих зображень обчислюється кутовий градієнт і його напрямок для кожного пікселя за формулами:

$$\text{Кутовий градієнт } (G) = \sqrt{G_x^2 + G_y^2}$$

$$\text{Кут } (\theta) = (\tan(\frac{G_y}{G_x}))^{-1}$$

Кутовий градієнт завжди перпендикулярний до контурів. Він заокруглюється до одного з чотирьох кутів, які представляють собою вертикальний, горизонтальний та два діагональних напрямки. Після отримання величини та напрямку градієнта виконується повне сканування зображення для видалення будь-яких небажаних пікселів, які можуть не бути частиною контура. Для цього для кожного пікселя цей піксель перевіряється на те чи є він локальним максимумом в своєму околі напрямку градієнта, рис 3:

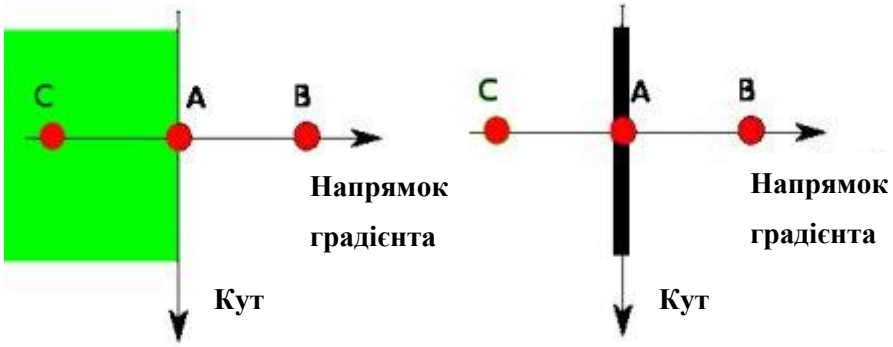

Рис. 3. Напрямок градієнта

Точка А знаходиться на краю у вертикальному напрямку. Напрямок градієнта перпендикулярний до краю. Точки В та С знаходяться в напрямку градієнта. Таким чином, точка А порівнюється з точками В та С для перевірки її на локальний максимум. Якщо це так, то точка розглядається в наступних етапах, інакше – обнулюється.

На останньому кроці вирішується чи всі контури є справді контурами. Для цього необхідні два порогових значення мінімуму та максимуму. Будь-які контури з градієнтом інтенсивності більшим за максимум обов’язково будуть контурами, а ті, що нижче за мінімум точно не будуть, тому автоматично відкидаються алгоритмом. Значення, які знаходяться між мінімумом і максимумом, класифікуються як ребра або як не ребра в залежності від їх зв’язності. Якщо вони зв’язані с пікселями чіткими контурами, то вважаються частиною цих контурів, інакше – відкидаються [27].

РОЗДІЛ 3. ПОПЕРЕДНЯ ДІАГНОСТИКА ВИДИМИХ СИМПТОМІВ ОБЛИЧЧЯ

3.1. Загальний опис алгоритму

Для подолання проблеми обмеженості тренувальних наборів даних фокус був направлений на збір середніх значень та стандартних квадратичних відхилень для наборів фото із відсутністю видимих симптомів обличчя. Тренувальні та тестові дані були відібрані вручну для фільтрації даних, які є неоднозначними чи не є чіткими. Для діагностики видимих симптомів обличчя необхідно спочатку локалізувати потрібну частину обличчя на фото. Для діагностики зміни кольору був застосований простір кольорів CIELAB. Це надало змогу створити модель незалежну від пристрою, адже в ній колір визначається незалежно від способу його створення чи відображення. Для діагностики асиметрії лиця були використані координати точок ознак обличчя. Для діагностики сухості губ – метод виявлення контурів.

На тренувальних даних обчислювалося одна з описаних вище значень для кожної фотографії із відсутністю видимого симптому. Далі обчислювалося середнє значення та стандартне квадратичне відхилення. Для зображень із наявними видимими симптомами обличчя у більшості випадків значення є вищими або нижчими за ці ж значення обчислені для зображень із відсутніми симптомами. На валідаційних даних із наявними симптомами обличчя було обрано найкращу лінійну комбінацію середнього значення та стандартного квадратичне відхилення для кожної ознаки. Перехід за цю лінійну комбінацію вниз чи вверх (залежно від алгоритму) означає діагностику видимого симптому обличчя. Для прискорення обчислень середнього значення та стандартного квадратичного відхилення було застосовано технологію ПАРСК Java [28]. Це дало змогу скоротити час обчислень у декілька разів.

Для пошуку пошуку порогового значення поділу зображень на зображення з наявними і відсутніми видими ознаками було обрано алгоритм на основі пошуку найкрашьої лінійної комбінації середнього значення та стандартного квадратичного відхилення. Таким чином, порогове значення є прив'язаним до кожного симптому. Лінійна комбінація записується рівнянням $\mu + a * \delta$, де μ – середнє значення, а δ - середнє квадратичне відхилення. Далі алгоритм змінює значення коефіцієнта a від -5 до 5 з кроком 0,001. Обчислюються оцінки ефективності для кожного фіксованого порогового значення та за найвищими оцінками обирається значення коефіцієнта a для даного симптому. На графіку (рис. 4) можна побачити зміну оцінок ефективності в залежності від значення шуканого коефіцієнта:

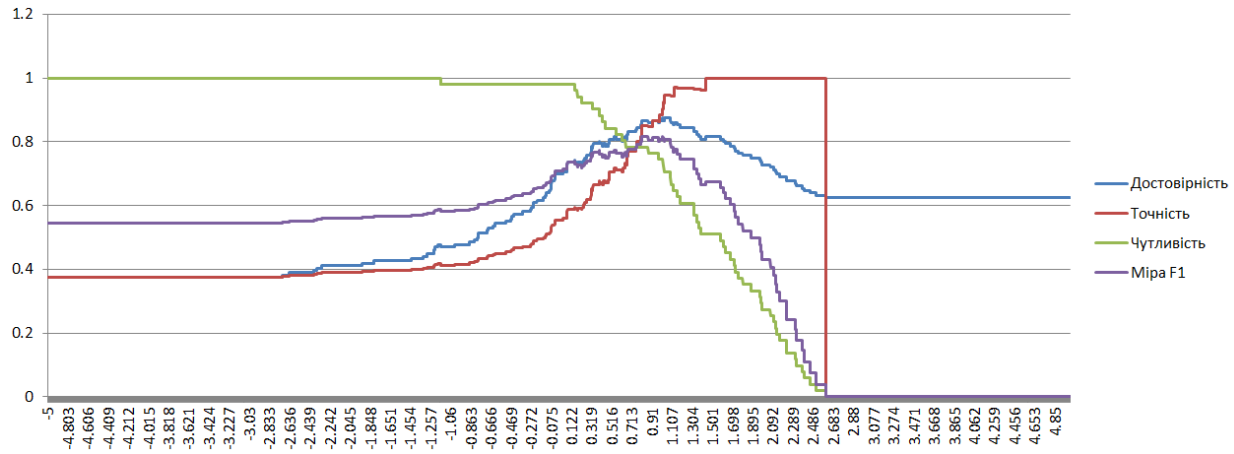

Рис. 4. Графік зміни оцінок ефективності при пошуку коефіцієнта поділу

3.2. Діагностика мішків під очима

Для діагностики мішків під очима треба спершу локалізувати очі на зображенні. Далі овал ока вписується в прямокутник, використовуючи крайні точки зверху, знизу, зліва та справа від ока. Наступним кроком є знаходження області під оком, в якій можна спостерігати наявність чи відсутність потемнінь. Для цього будується симетричний прямокутник до вже побудованого на оці, віддзеркалений відносно нижньої його сторони та зміщений на коефіцієнт. Цей коефіцієнт можна варіювати за необхідності, він призначений для усунення нижніх вій з прямокутника для діагностики.

Таким чином, була проведена локалізація ділянки, яка відповідає за мішки під очима. Для діагностики наявності чи відсутності потемнінь будується ще один прямокутник, симетричний до попереднього та віддзеркалений відносно нижньої сторони. Тут доцільно зсувати даний прямокутник на певний коефіцієнт, адже мішки під очима можуть бути більшими за розмір ока. Побудований прямокутник відповідає за ділянку шкіри, з якою буде порівнюватися область під очима. Після цього розглядається різниця у коефіцієнтів RGB між цими прямокутниками, поділені на кількість пікселів ока (код наведений у додатку А). Дані, з якими порівнюються знайдені коефіцієнти, наведені в таблиці, таб. 1:

Номер	**Компонента**	**μ (середнє значення)**	**δ (середнє квадратичне відхилення)**
1	r(очі) / Σ(очі)	0.1277055564508918	0.00798237586401697
2	g(очі) / Σ(очі)	0.13208364114346535	0.0890832192370234
3	b(очі) / Σ(очі)	0.14743503767390834	0.08785469727806773

Таб. 1. Коефіцієнти діагностики мішків під очима

Алгоритм діагностує наявність потемніння під очима, якщо виконується одна з умов:

- Відповідне дане значення більше за середнє значення компоненти 1
- Відповідне дане значення більше за середнє значення компоненти 2 плюс стандартне відхилення помножене на коефіцієнт 0.089 та одночасно відповідне дане значення більше за відповідне середнє значення компоненти 3 плюс стандартне відхилення помножене на коефіцієнт 0.088, рис. 5:

Рис.5. Діагностика мішків під очима

Оцінки ефективності для даного алгоритму виявились такими, таб.2:

Достовірність	**Точність**	**Чутливість**	**Міра F1**
0,881356	0,894737	0,918919	0,906667

Таб 2. Оцінки ефективності діагностики мішків під очима

3.3. Діагностика синюватості губ

Для діагностики синюватості губ треба спочатку локалізувати губи на зображенні. Далі рахується значення каналу a простору CIELAB губ поділений на кількість пікселів цієї ознаки. Провівши порівняння на тренувальних даних цієї компоненти для фотографій з та без синюватості губ, було виявлено, що компонента синіх губ завжди менша за середнє значення і для оцінки наскільки було обчислено стандартне відхилення різниць між середнім значенням для звичайних губ та для синюватих (код наведений у додатку А). Ці дані порівнюються з коефіцієнтами наведеними у таблиці, таб.3:

Номер	Компонента	μ (середнє значення)	δ (середнє квадратичне відхилення)
4	α(губи) / Σ(губи)	18.747098259964073	4.6907665761670164
5	α(губи) / Σ(губи) - α(сині губи) / Σ(сині губи)	3.382342778579465	2.152412921588228

Таб. 3. Коефіцієнти діагностики синюватості губ

Алгоритм діагностує синюватість губ, якщо компонента 4 є меншою за середнє значення звичайних губ мінус стандартне відхилення різниці компоненти 5 помножене на коефіцієнт 0.85.

Оцінки ефективності для даного алгоритму виявились такими, таб.4:

Достовірність	Точність	Чутливість	Міра F1
0,867647	0,851064	0,784314	0,816327

Таб. 4. Оцінки ефективності діагностики синюватості губ

3.4. Діагностика червонуватості очей

Для діагностики червонуватості очей треба спочатку локалізувати очі на зображенні. Далі рахується значення каналу b простору CIELAB кожного ока окремо поділені на кількість пікселів цієї ознаки (код наведений у додатку А). Ці дані порівнюються з коефіцієнтами наведеними у таблиці, таб. 5:

Номер	Компонента	μ (середнє значення)	δ (середнє квадратичне відхилення)
6	α(очі) / Σ(очі)	3.2140258841557534	3.143526693485413

Таб.5. Коефіцієнти діагностики червонуватості очей

Алгоритм визначає око червонуватим при компоненті 6 більшою за середнє значення плюс стандартне відхилення. Для діагностики

червонуватості очей достатньо, щоб одне з очей було червонуватим. Відповідність значення компоненти 6 та ступеню червонуватості можна бачити нижче, рис.6:

Комп. 6 = 6.139615880766551

Комп. 6 = 7.001433582272654

Комп. 6 = 8.755587573453383

Комп. 6 =10.466377097529708

Рис.6. Порівняння ступенів червонуватості очей

Оцінки ефективності для даного алгоритму виявились такими, таб. 6:

Достовірність	**Точність**	**Чутливість**	**Міра F1**
0,939394	0,842105	0,941176	0,888889

Таб 6. Оцінки ефективності діагностики червонуватості очей

3.5. Діагностика набряків та асиметрії обличчя

При діагностиці надмірної асиметрії обличчя було проведено пошук наявності асиметрії очей, рота та брів. Асиметрія вважається діагностованою, якщо асиметричними є або очі, або рот, або брови.

Для діагностики асиметрії очей спочатку потрібно локалізувати їх на зображенні, рис. 7. За знайденими контурами очей шукаються центроїди кожного з них. Далі знаходиться кут між ними. Як альтернативу можна використовувати різницю між y-координатами.

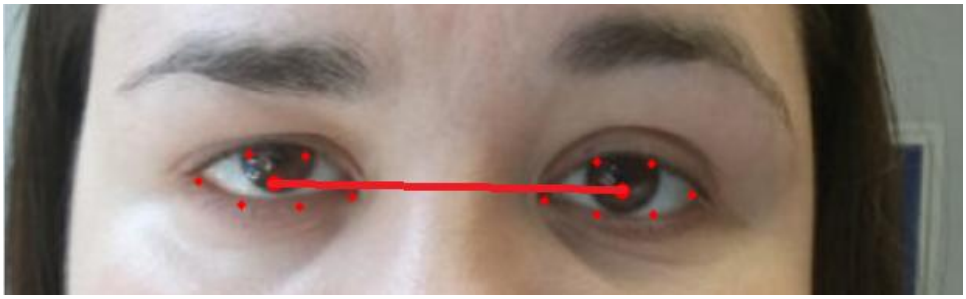

Рис.7. Діагностика асиметрії очей

Знайдений кут порівнюється з даними, наведеними у таблиці, таб. 7:

Номер	Компонента	μ (середнє значення)	δ (середнє квадратичне відхилення)
7	abs(центроїд(ліве око) – центроїд(праве око))	0.6672306321257245	0.6122106267178532

Таб.7. Коефіцієнти діагностики асиметрії очей

Алгоритм діагностує асиметрію очей, якщо кут між центроїдами очей більший за середнє значення плюс стандартне відхилення компоненти 7 помножене на коефіцієнт 2.14.

Для діагностики асиметрії рота спочатку потрібно локалізувати його на зображенні. Далі розглядаються лівий на правий кутики рота і різниця між їхніми y-координатами. Як альтернативу можна використовувати кут між кутиками, рис.8.

Рис.8. Діагностика асиметрії рота

Знайдена різниця порівнюється з даними наведеними у таблиці, таб.8:

Номер	Компонента	μ (середнє значення)	δ (середнє квадратичне відхилення)
8	abs(лівий кут(рот) – правий кут(рот))	2.0594594594594593	1.7893769804281592

Таб. 8. Коефіцієнти діагностики асиметрії рота

Алгоритм діагностує асиметрію рота, якщо різниця між у-координатами кутиків рота більша за середнє значення плюс стандартне відхилення компоненти 8 помножене на коефіцієнт 2.14.

Для діагностики асиметрії брів спочатку потрібно локалізувати їх на зображенні. Далі розглядається різниця між у-координатами найвищих точок кожної з брів, рис.9. Як альтернативу можна використовувати кут між цими точками.

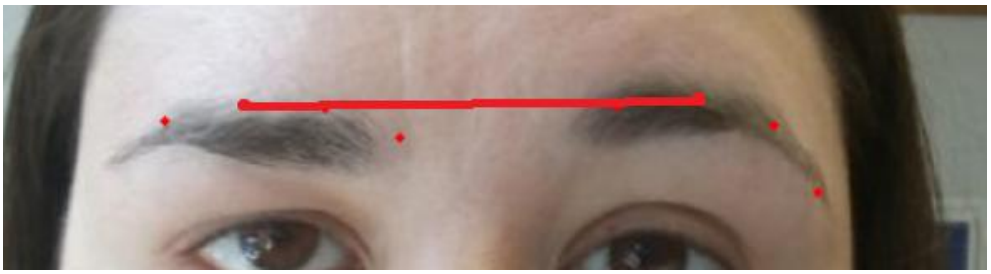

Рис.9. Діагностика асиметрії брів

Знайдена різниця порівнюється з даними наведеними у таблиці, таб.9:

Номер	**Компонента**	**μ (середнє значення)**	**δ (середнє квадратичне відхилення)**
9	abs(верхня точка(ліва брова) – верхня точка(права брова)	2.4918918918918	1.87766076122887

Таб. 9. Коефіцієнти діагностики асиметрії брів

Алгоритм діагностує асиметрію брів, якщо різниця між у-координатами найвищих точок брів більша за середнє значення плюс стандартне відхилення компоненти 9 помножене на коефіцієнт 1.88 (код наведений у додатку А).

Оцінки ефективності для даного алгоритму виявились такими, таб.10:

Достовірність	Точність	Чутливість	Міра F1
0,958333	0,941176	0,979592	0,96

Таб.10. Оцінки ефективності діагностики асиметрії обличчя

3.6. Діагностика опущених кутиків губ

Для діагностики опущених кутиків губ треба спочатку локалізувати рот на зображенні, рис.10. Далі знаходиться центроїд рота та шукаються різниці центроїда з крайніми правою та лівою точками рота. Далі обчислюється середнє значення цих різниць (код наведений у додатку А).

Опущені кутики губ Відсутність опущених кутиків губ

Рис.10. Діагностика опущених кутиків губ

Знайдене середнє значення порівнюється з даними, наведеними у таблиці, таб. 11:

Номер	Компонента	μ (середнє значення)	δ (середнє квадратичне відхилення)
10	mean(лівий кут(рот) - центроїд(рот), правий кут(рот) – центроїд(рот))	6.50454545454545	5.11725932264592

Таб.11. Коефіцієнти діагностики опущених кутиків губ

Алгоритм діагностує опущеність кутиків губ, якщо це середнє значення менше за середнє значення мінус стандартне відхилення компоненти 10 помножене на коефіцієнт 0.25.

Оцінки ефективності для даного алгоритму виявились такими, таб.12:

Достовірність	**Точність**	**Чутливість**	**Міра F1**
0,925	0,814815	0,956522	0,88

Таб.12. Оцінки ефективності діагностики опущених кутиків губ

3.7. Діагностика сухості та лущення губ

Для діагностики сухості та лущення губ знадобиться попередня обробка даних для виявлення відкритого рота. Це необхідно для того, щоб алгоритм не враховував зуби та внутрішній контур губ як тріщини та сухість. Для виявлення відкритого рота необхідно спочатку локалізувати рот на фото, рис. 11. Далі розглядається різниця y-координат, які відповідають за верхню та нижню середини внутрішньої частини рота відповідно.

Відкритий рот Закритий рот

Рис.11. Пошук відкрити рота на зображенні

На рис. 11 можна побачити, що різниця цих координат для закритого роту набагато менша за цю ж різницю для відкритого. Обчислена різниця порівнюється з даними наведеними у таблиці, які були обчислені для закритого рота, таб.13:

Номер	Компонента	μ (середнє значення)	δ (середнє квадратичне відхилення)
11	верхня частина внутрішньої середини рота – нижня частина внутрішньої середини рота	2.23913043478260 9	3.3112770596244 99

Таб. 13. Коефіцієнти пошуку відкритого рота на зображенні

Алгоритм визначає рот відкритим, якщо обчислена різниця більша за середнє значення плюс стандартне відхилення компоненти 11 помножене на коефіцієнт 2.

Оцінки ефективності для даного алгоритму виявились такими, таб.14:

Достовірність	Точність	Чутливість	Міра F1
0,964286	0,990476	0,945455	0,967442

Таб.14. Оцінки ефективності пошуку відкритого рота на зображенні

Далі для діагностики враховується був рот відкритим чи закритим.

Для визначення сухості та лущення губ використовується метод виявлення контурів бібліотеки OpenCv[27]. В алгоритмі розглядається коефіцієнт кількості ненульових компонент в масиві контурів. Тобто чим менша кількість ненульових компонент, тим більш сухими є губи. Відкритий рот має більше контурів за закритий, тому для точності діагностики даний коефіцієнт модифікується за даними наведеними у таблиці. Ці дані були пораховані як середні значення та стандартні квадратичні відхилення для різниць коефіцієнта кількості ненульових компонент в масиві контурів між відкритим та закритим ротом, таб.15:

Номер	Компонента	μ (середнє значення)	δ (середнє квадратичне відхилення)
12	edge detector(відкритий рот) – edge detector(закритий рот)	178.85595374253367	97.66810118612692

Таб.15. Коефіцієнти різниці компоненти діагностики між відкритим та закритим ротом

Для зображень з виявленим відкритим ротом від коефіцієнта кількості ненульових компонент в масиві контурів віднімається середнє значення та половина стандартного відхилення компоненти 12 (код наведений у додатку А). Після цієї попередньої обробки даний коефіцієнт порівнюється з даними наведеними у таблиці, таб. 16:

Номер	Компонента	μ(середнє значення)	δ (середнє квадратичне відхилення)
13	edge detector(губи)	437.59194208515015	93.44890149593988

Таб.16. Коефіцієнти діагностики сухості та лущення губ

Алгоритм діагностує сухість і лущення губ, якщо даний коефіцієнт більший за суму середнього значення і стандартного відхилення компоненти 13.

Оцінки ефективності для даного алгоритму виявились такими, таб. 17:

Достовірність	Точність	Чутливість	Міра F1
0,920635	0,777778	0,933333	0,848485

Таб 17. Оцінки ефективності діагностики сухості та лущення губ

РОЗДІЛ 4. МЕДИЧНИЙ ЩОДЕННИК

4.1. Адаптація під користувача

Для коректної попередньої діагностики важливим є орієнтир не тільки на загальну шкалу відхилень, але й на користувача. В кожної людини є свої особливості зовнішності, які не обов’язково є аномаліями і таким чином не мають бути діагностованими як видимий симптом. Тому доцільною є модифікація наведених вище алгоритмів під користувача. Користувач добровільно проходить ініціалізацію, завантажуючи свої фото обличчя для діагностики. Якщо програма діагностує якийсь видимий симптом, то перепитає у користувача про коректність діагностики. У випадку її некоректності відповідне порогове значення діагностики буде модифіковане як середнє значення порогового значення та обчисленого. Із більшою кількістю фото та некоректних діагностик це середнє значення буде ставати все ближче до необхідного для успішної класифікації. Обчислення саме середнього значення дозволяє уникнути великих похибок при потраплянні аномальних значень. На графіку (рис.12) можна бачити як покращується класифікація фотографій з відсутнім видимим симптомом мішків під очима із додаванням нових фото. При цьому діагностика наявного симптому не погіршується. Адаптація проводиться до усіх неправильно класифікованих симптомів із алгоритмів наведених раніше, окрім алгоритму діагностики опущених кутиків рота.

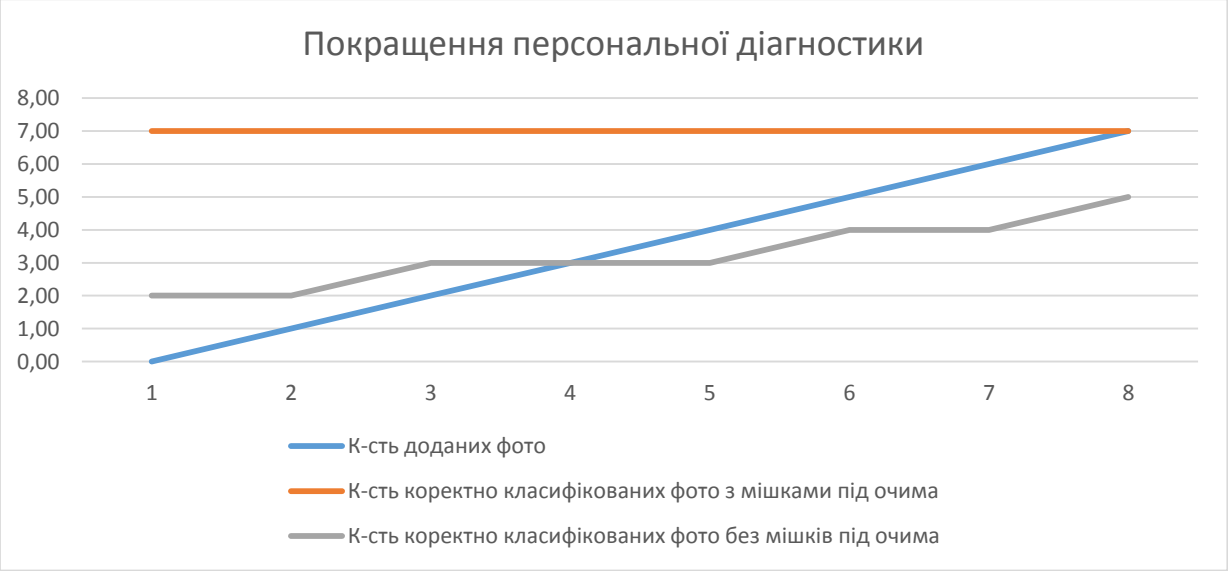

Рис.12. Графік покращення персональної діагностики

4.2. Медичний щоденник зміни кольору обличчя

Для діагностики зміни кольору обличчя не було проведено попереднє обчислення середнього значення та стандартного квадратичного відхилення, адже це давало б велику похибку і неоднозначні результати. Середнє значення буде обраховане для шкіри даної людини за її фотографіями.

Для діагностики зміни кольору шкіри треба спочатку локалізувати ділянки шкіри на зображенні. Локалізується три ділянки: центральна частина лобу та дві ділянки щік. Для локалізації центральної частини лобу спочатку локалізуються брови. Далі відбираються координати верхніх частин обох брів. Ці дві координати зсуваються вверх на коефіцієнт, щоб забезпечити відсутність частин брів в локалізованій ділянці. Будується прямокутник нижніми вершинами якого є ці дві координати. Верхні вершини знаходяться на деякій відстані від нижніх, але невеликій, щоб забезпечити відсутність волосся в локалізованій ділянці. Для локалізації ділянок щік відбувається схожа процедура на діагностику мішків під очима. Спершу локалізовуються очі на зображенні, рис.13. Далі овал ока вписується в прямокутник, використовуючи крайні точки зверху, знизу, зліва та справа від ока. Далі

будується симетричний прямокутник до вже побудованого на оці, віддзеркалений відносно нижньої його сторони та зміщений на коефіцієнт. Цей коефіцієнт дорівнює подвоєній ширині прямокутника, в який вписувалося око. Далі відбувається обчислення каналу a простору кольорів CIELAB (код наведений у додатку А).

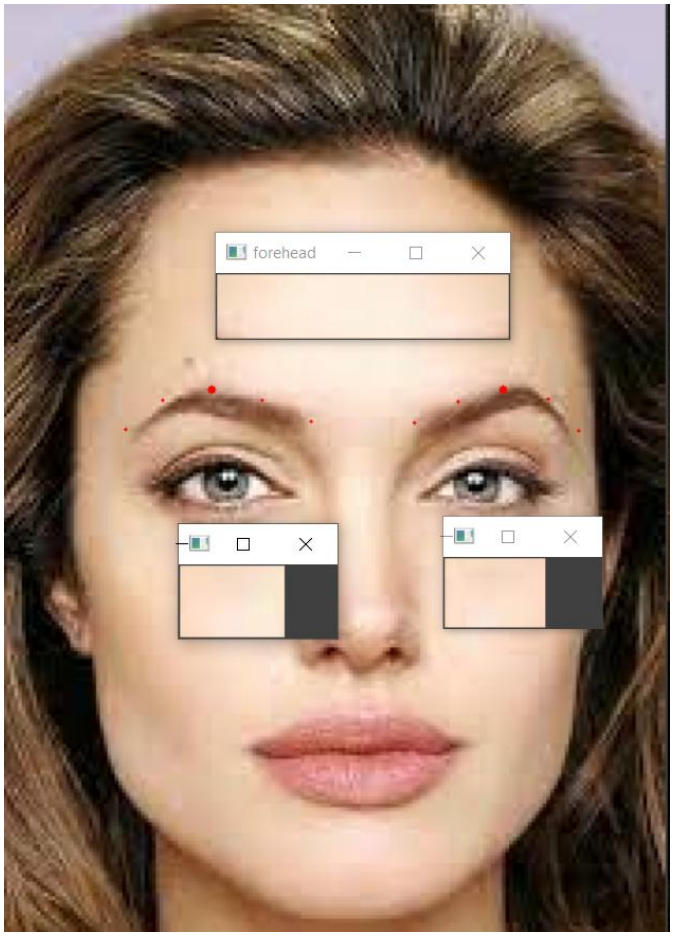

Рис.13. Локалізовані ділянки діагностики зміни кольору шкіри

Діагностика видимого симптому зміни кольору шкіри відбувається за таким алгоритмом:

1) Користувач додає першу фотографію на якій обчислюється дане значення
2) Далі обчислене значення на новій фотографії порівнюються із першим значенням, якщо нове значення більше за перше плюс деяких коефіцієнт, то алгоритм діагностує зміну кольору шкіри. Інакше нове обраховане значення запам’ятовується як останнє і також

обчислюється і запам’ятовується середнє значення між новим та першим. Таким чином, забезпечується збір середніх значень для кожного користувача індивідуально.

3) За відсутності діагностованої зміни кольору шкіри постійно оновлюється останнє знайдене значення та загальне середнє значення. Нові значення тепер порівнюються з останнім та з середнім (на першому кроці теж так, але це було одне і те саме число). Це дає змогу відслідковувати як і поступові зміни кольору шкіри, так і раптові.

На валідаційних даних було виявлено, що коефіцієнт різниці між новим та середнім значеннями та новим і останнім значеннями не мають перевищувати п’яти. На рис. 14- рис. 19 можна побачити як алгоритм діагностує зміну кольору шкіри. Так на рис. 15, 17 та рис. 19 діагностується блідуватість та жовтуватість шкіри. Також наведено зміну середнього й останнього значень із додаванням нових фото (рис. 14, 16 та рис. 18), де не було діагностовано зміну кольору шкіри.

Рис. 14.

Mean = last = -1.523981162826945

Рис. 15.

a = 5.347912645998754

Рис. 16.

Рис. 17.

Mean = 0.0666527985333394

a = 5.347912645998754

Last = 1.6572867598936238

Рис 18.

Рис 19.

Mean = -1.05843797996723423

a = 4.06518167508208

Last = -3.3086195369683815

4.3. Медичний щоденник червонуватості обличчя

Для діагностики червонуватості обличчя не було проведено попереднє обчислення середнього значення та стандартного квадратичного відхилення, адже це давало б велику похибку і неоднозначні результати. Середнє

значення буде обраховане для шкіри даної людини за її фотографіями. Для діагностики червонуватості лиця треба спочатку локалізувати шкіру на зображенні. Локалізувати лише ділянки лоба та щік буде не достатньо, адже почервоніння зазвичай виникають локально. Тому алгоритм потрібно модифікувати. Спочатку локалізується обличчя на зображенні, рис.20. Далі локалізуються губи та видаляються із зображення (замальовуються чорним кольором), щоб зміна кольору губ не заважала при діагностиці червонуватості обличчя (код наведений у додатку А).

Рис.20. Діагностика червонуватості обличчя

Наступним кроком є обчислення кількості пікселів, які підпадають під діапазон червонуватості на зображенні, рис.21:

Рис.21. Діапазон кольорів RGB діагностики червонуватості обличчя

Діагностика видимого симптому червонуватості обличчя відбувається за таким алгоритмом:

1) Користувач додає першу фотографію на якій обчислюється дане значення
2) Далі обчислене значення на новій фотографії порівнюються із першим значенням, якщо нове значення більше за перше плюс деяких коефіцієнт, то алгоритм діагностує червонуватість лиця. Інакше нове обраховане значення запам'ятовується як останнє і також обчислюється і запам'ятовується середнє значення між новим та першим. Таким чином, забезпечується збір середніх значень для кожного користувача індивідуально.
3) За відсутності діагностованої червонуватості шкіри постійно оновлюється останнє знайдене значення та загальне середнє значення. Нові значення тепер порівнюються з останнім та з середнім (на першому кроці теж так, але це було одне і те саме число). Це дає змогу відслідковувати як і поступові зміни червонуватості лиця, так і раптові.

Було зібрано валідаційні дані (рис.22- рис.27). На них виявлено, що коефіцієнт різниці між знайденим значенням і середнім та знайденим

значенням і останнім не повинна перевищувати 15 тисяч. Можна побачити як алгоритм діагностує червонуватість обличчя на рисунках 23, 25 та 27. Також наведено зміну середнього й останнього значень із додаванням нових фото (рис. 22, 24 та рис. 26), де не було діагностовано червонуватість обличчя.

Рис. 22.

Mean = 7312

Last = 7312

Рис. 23.

n = 29236

Рис. 24.

Mean = 7213

Last = 7114

Рис. 25.

n = 35356

Рис. 26.

Mean = 8977.3

Last = 12506

Рис. 27.

n = 26321

4.4. Медичний щоденник випадіння брів

Для діагностики випадіння брів не було проведено попереднє обчислення середнього значення та стандартного квадратичного відхилення, адже це давало б велику похибку і неоднозначні результати. Середнє значення буде обраховане для брів даної людини за її фотографіями. Для діагностики випадіння брів треба спочатку локалізувати брови на зображенні, рис.28. Далі будуються дві ділянки навколо брів (рис.29), які будуть розглядатися в подальшому. Наступним кроком є обчислення діапазону шкіри, який будемо шукати на ділянках брів. Це необхідно для виявлення кількості пікселів кольору шкіри людини на ділянці брів. Різке зменшення цієї кількості буде означати наявність випадіння брів.

Для локалізації ділянки шкіри скористаємося схожим алгоритмом як для діагностики зміни кольору шкіри. Локалізуємо центральну частину лобу. Для цього спочатку локалізуються брови. Далі відбираються координати верхніх частин обох брів. Ці дві координати зсуваються вверх на коефіцієнт, щоб забезпечити відсутність частин брів в локалізованій ділянці. Будується

прямокутник нижніми вершинами якого є ці дві координати. Верхні вершини знаходяться на деякій відстані від нижніх, але невеликій, щоб забезпечити відсутність волосся в локалізованій ділянці.

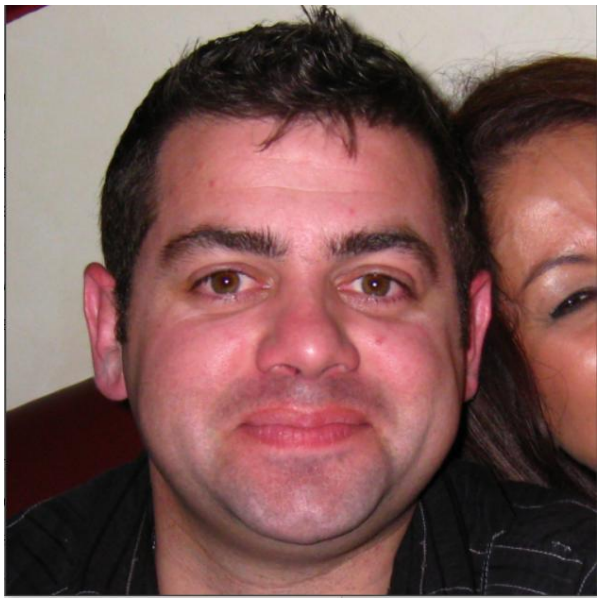

Рис.28.

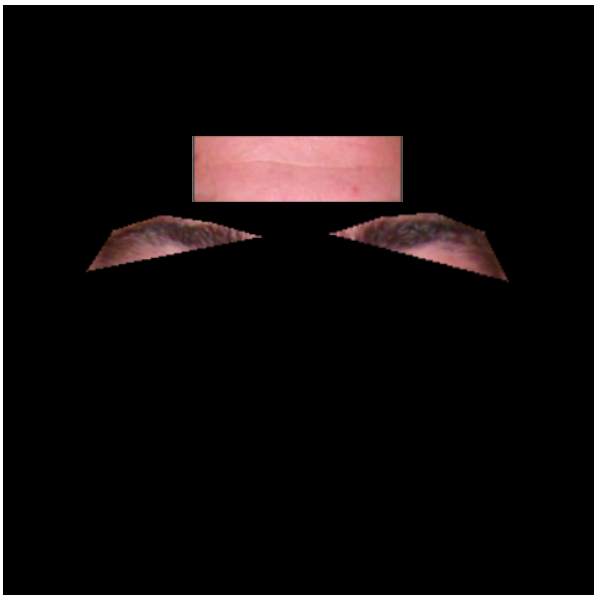

Рис.29.

Для визначення діапазону кольору шкіри спочатку знайдемо кольори RGB для кожного пікселя з діапазону центральної частини лобу. Далі для кожного каналу r, g, b порахуємо середнє значення та стандартне квадратичне відхилення. Діапазон буде визначатися так: для кожного каналу нижньою межею є середнє значення мінус стандартне квадратичне відхилення, а верхньою – середнє значення плюс стандартне квадратичне відхилення. Наступним кроком алгоритму буде обчислення кількості пікселів діапазону кольору шкіри на ділянках брів (код наведений у додатку А).

Діагностика видимого симптому випадіння брів відбувається за таким алгоритмом:

1) Користувач додає першу фотографію на якій обчислюється дане значення
2) Далі обчислене значення на новій фотографії порівнюються із першим значенням, якщо нове значення більше за перше плюс деяких коефіцієнт, то алгоритм діагностує випадіння брів. Інакше нове

обраховане значення запам’ятовується як останнє і також обчислюється і запам’ятовується середнє значення між новим та першим. Таким чином, забезпечується збір середніх значень для кожного користувача індивідуально.

3) За відсутності діагностованого випідання брів постійно оновлюється останнє знайдене значення та загальне середнє значення. Нові значення тепер порівнюються з останнім та з середнім (на першому кроці теж так, але це було одне і те саме число). Це дає змогу відслідковувати як і поступові зміни випадіння брів, так і раптові.
4) На валідаційних даних було виявлено, що коефіцієнт різниці між знайденим значенням і середнім та знайденим значенням і останнім не повинна перевищувати 150.

Для рисунків 30, 31 обчислена кількість пікселів, яка дорівнює 2. Для рисунків 32, 33 вона дорівнює 197. Із зображень можна побачити, що брови на рисунках 32, 33 і справді є рідшими, а оскільки різниця між першим та другим значеннями більша за 150, то алгоритм діагностував би випадіння брів.

Рис. 30

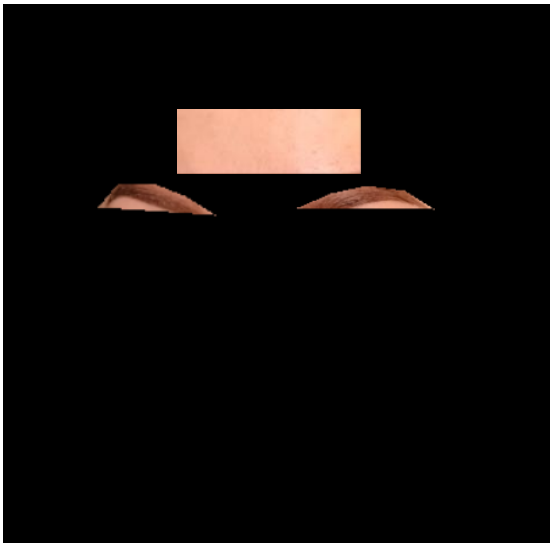

Рис. 31

Рис. 32

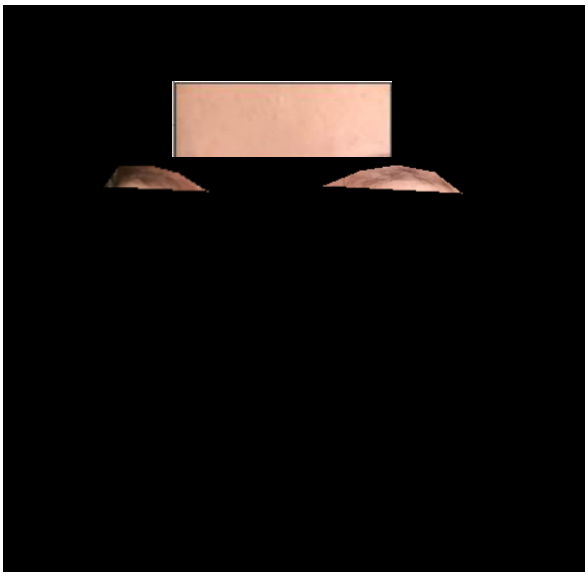

Рис. 33

4.5. Практичні рекомендації

На основі виявлених симптомів обличчя можна робити попередню діагностику захворювань, порекомендувавши на що варто звернути увагу[29, 30, 31, 32, 33]. Виявлені симптоми та можливі недуги наведені у таблиці 18:

Симптом	**Захворювання**
Мішки під очима	Хронічна алергія, гіпотериоз, сонне апное, захворювання нирок, ураження щитоподібної залози зі зниженням її функції
Синюваті губи	Переохолодження, цианоз, круп, при вагітності – дефіцит заліза, хвороба серця або легень
Червонуваті очі	Алергічний кон’юктивіт, язва роговиці, синдром сухого ока, інфекційний кератит, блефарит, інтоксикація, аутоімунні захворювання, грип, ГРВІ

Надмірна асиметрія лиця	Перша ознака інсульту, лицевий параліч
Опущені кутики губ	Депресія
Сухі губи, що лущаться	Зневоднення, діабет, порушення роботи щитовидної залози, дефіцит вітаміна B12 чи заліза
Зміна кольору шкіри (блідість, пожовтіння)	Анемія, ураження печінки, жовчних шляхів
Почервоніння обличчя	Проблеми з травленням (наприклад, при целіакії, непереносимості глютену), алергія, екземи та розацеа.
Випадіння брів	Вогнищева алопеція

Таб. 18. Відповідність симптомам захворювань

РОЗДІЛ 5. ПРОГРАМНА РЕАЛІЗАЦІЯ

5.1. Адаптація Python під Android Studio

Для програмної реалізації алгоритмів було вирішено використати мову Python та наявні там бібліотеки. Локалізація обличчя та ключових ознак обличчя відбувається з допомогою бібліотеки face_recognition та наявних там методів face_locations та face_landmarks. Перший метод повертає список лиць, локалізованих на фото та точки прямокутника, в яке вписане локалізоване лице. Другий метод повертає список словників локалізованих частин лиця. У словнику є наявні такі ознаки: ліва та права брови, ліве та праве око, ніс, губи та підборіддя [34]. Ця бібліотека дозволяє з легкістю локалізовувати частини обличчя та є просто адаптованою під Android Studio завдяки технології Chaquopy.

Chaquopy – технологія, яка дозволяє найпростішим способом використовувати Python в додатку для Андроїд. Вона забезпечує користувача всіма необхідними включеннями компонентів Python в додаток під Андроїд, в тому числі:

1) Повна інтеграція зі стандартною системою збірки рішення Gradle Android Studio.
2) Прості у використанні API для виклику коду Python з Java чи Kotlin чи навпаки.
3) Широкий спектр сторонніх пакетів Python, включаючи SciPy, OpenCV, TensorFlow, face_recognition та багато інших [35].

5.2. Результати роботи програми

На головній сторінці додатку наявні чорити кнопки: ініціалізації, діагностики, пробної діагностики та очищення усіх даних. Ініціалізація дозволяє підлаштувати діагностику під особливості зовнішньості кожної людини, туди можна загружати безліч фото для покращення точності діагностики. Далі додаток дозволяє проводити попередню діагностику із модифікованими параметрами, отриманими з ініціалізації. Також наявна функція пробної діагностики без урахування особливостей конкретної людини. В кожному меню є кнопки обрання фото, проведення діагностики (обчислень), виведення результатів та кнопка назад.рис. 34 - 37:

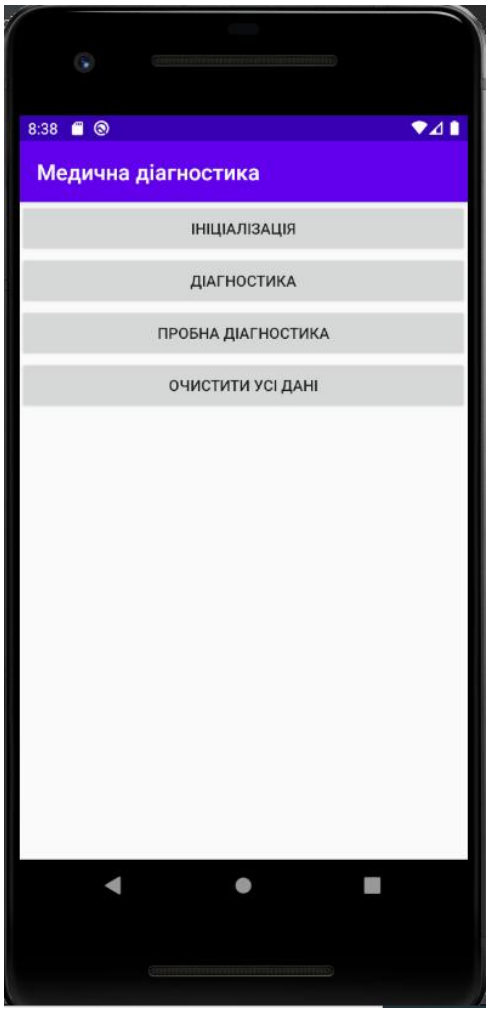

Рис. 34

Рис. 35

Рис. 36

Рис. 37

Діагностику можна проводити за окремими параметрами (рис. 38-39) після чого будуть виведені результати та рекомендації (рис. 40).

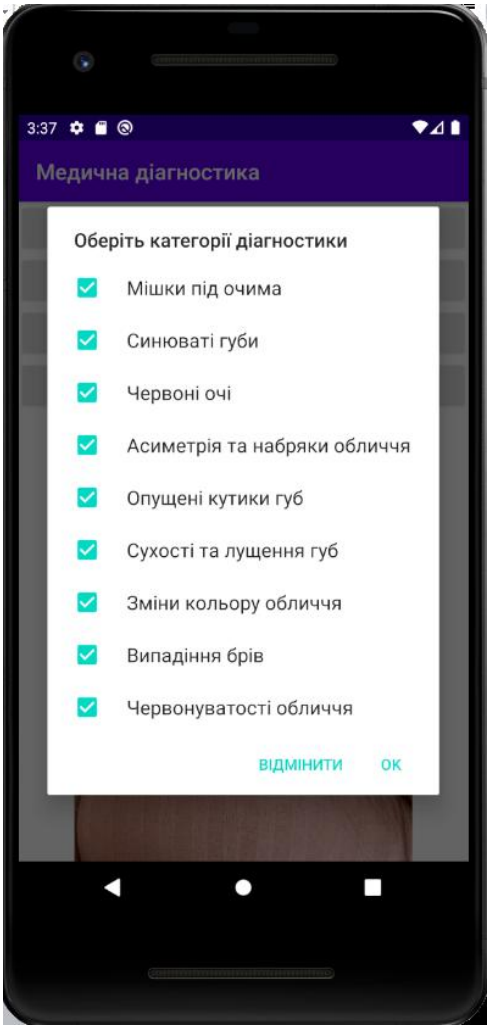

Рис. 38

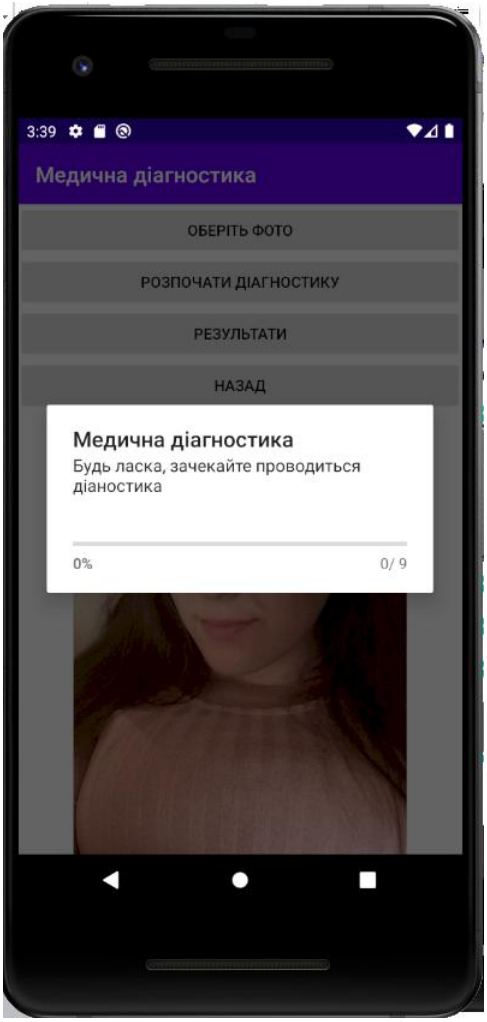

Рис. 39

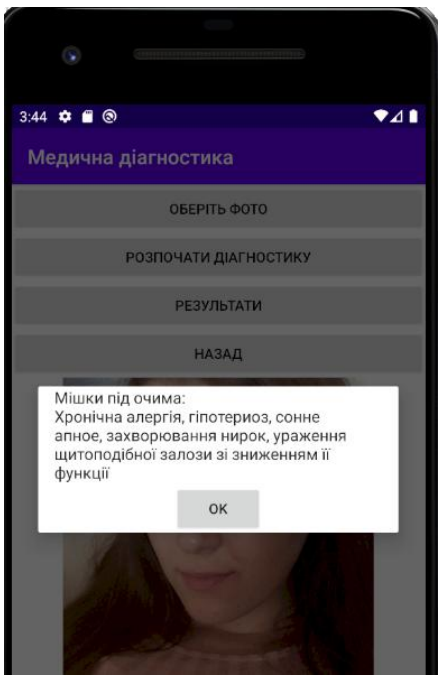

Рис. 40

ВИСНОВКИ

У роботі було розглянуто метод медичної діагностики, який базується на створенні загальної шкали відхилень. Було наведено алгоритми, які не потребують великого набору тренувальних даних для діагностики видимих симптомів обличчя. Достатньо високі показники ефективності вказують, що навіть за обмеженого набору даних можна створити гнучкий і адаптивний механізм для поставленої задачі. Узагальнення симптомів надало можливість попередньої діагностики великого переліку хвороб в одній програмі.

Було створено застосунок, який дозволяє спостерігати за видимими змінами обличчя. Було запропоновано орієнтуватися не лише на загальні властивості симптомів, проте і на індивідуальну специфіку кожного користувача. Ведення медичного щоденника дозволяє розширити можливості попередньої діагностики і адаптуватися під користувача.

Результати цього дослідження можуть бути застосовані як допоміжний засіб при медичній діагностиці та як складова частина системи догляду за здоров’ям та для підвищення ефективності процесу лікування.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. A Smartphone-Based Automatic Diagnosis System for Facial Nerve Palsy [Електронний ресурс] / Hyun Seok Kim, So Young Kim, Young Ho Kim, Kwang Suk Park. – 2015. – Режим доступу до ресурсу: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4634507/.
2. Delannoy JR, Ward TE. Proceedings of the Signals and Systems Conference (ISSC 2010) Cork: IET Irish; 2010. A preliminary investigation into the use of machine vision techniques for automating facial paralysis rehabilitation therapy
3. Felix Outlaw, Judith Meek, Lindsay MacDonald, Terence Leung. Screening for Neonatal Jaundice with a Smartphone
4. Feasibility of using deep learning to detect coronary artery disease based on facial photo Shen Lin, Zhigang Li, Bowen Fu, Sipeng Chen. European Heart Journal, Volume 41, Issue 46, 7 December 2020, Pages 4400–4411, https://doi.org/10.1093/eurheartj/ehaa640
5. Digital Skin Care: Top 8 Dermatology Apps [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://medicalfuturist.com/digital-skin-care-top-8-dermatology-apps/.
6. Detecting Visually Observable Disease Symptoms from Faces [Електронний ресурс] / Kuan Wang, Jiebo Luo. – 2016. – Режим доступу до ресурсу: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5007273/
7. Izet Masic “Five periods in Development of Medical Informatics” 1-5, 2014
8. Machine Learning for Medical Diagnostics – 4 Current Applications [Електронний ресурс] / Daniel Faggella. – 2020. – Режим доступу до ресурсу: https://emerj.com/ai-sector-overviews/machine-learning-medical-diagnostics-4-current-applications/.

9. Most Popular Medical Apps for Doctors and Patients [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.byteant.com/blog/most-popular-medical-apps-for-doctors-and-patients/.
10. VisualDX [Електронний ресурс] – Режим доступу до ресурсу: https://www.visualdx.com/clinical-solutions/.
11. Mobile Application for Preliminary Diagnosis of Diseases / Edgars Vasil evskis, Iryna Dubyak, Taras Basyuk – Riga, Latvia; Lviv, Ukraine. – 12c.
12. Pamela Bump. AI for Mobile Medical Diagnostics – Current Applications [Електронний ресурс] / Pamela Bump. – 2019. – Режим доступу до ресурсу: https://emerj.com/ai-sector-overviews/ai-for-mobile-medical-diagnostics-current-applications/.
13. Lienhart, R. and Maydt, J., «An extended set of Haar-like features for rapid object detection», ICIP02, pp. I: 900—903, 2002
14. Histograms of Oriented Gradients for Human Detection Navneet Dalal and Bill Triggs 2-6.
15. Histogram of Oriented Gradients and Object Detection
16. Viola and Jones, "Rapid object detection using a boosted cascade of simple features", Computer cool Vision and Pattern Recognition, 2001
17. Cortes, Corinna; Vapnik, Vladimir N. (1995). "Support-vector networks" (PDF). Machine Learning. 20 (3): 273–297.
18. drian Rosebrock. Facial landmarks with dlib, OpenCV, and Python [Електронний ресурс] / Adrian Rosebrock. – 2017. – Режим доступу до ресурсу: https://www.pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/.
19. One Millisecond Face Alignment with an Ensemble of Regression Trees [Електронний ресурс] – Режим доступу до ресурсу: https://www.csc.kth.se/~vahidk/papers/KazemiCVPR14.pdf.
20. AGernot Hoffmann Contents CIELab Color Space 2019. 2-3.

21. How to Perform Face Detection with Deep Learning [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://machinelearningmastery.com/how-to-perform-face-detection-with-classical-and-deep-learning-methods-in-python-with-keras/.
22. The Lab Color Mode in Photoshop, Adobe TechNote [Електронний ресурс] – Режим доступу до ресурсу: https://web.archive.org/web/20081207061220/http:/kb.adobe.com/selfservice/viewContent.do?externalId=310838.
23. Whitepaper on understanding colors by X-rite.2016. 14-16
24. Color spaces in OpenCV (C++ / Python) [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://learnopencv.com/color-spaces-in-opencv-cpp-python/.
25. Umbaugh, Scott E (2010). Digital image processing and analysis : human and computer vision applications with CVIPtools (2nd ed.). Boca Raton, FL: CRC Press
26. Canny, J., A Computational Approach To Edge Detection, IEEE Transactions on Pattern Analysis and Machine Intelligence, 8(6):679–698, 1986.
27. Alexander Mordvintsev & Abid K. Canny Edge Detection [Електронний ресурс] / Alexander Mordvintsev & Abid K.. – 2013. – Режим доступу до ресурсу: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_canny/py_canny.html.
28. Дерев’янченко О.В. ПАРКС-JAVA система для паралельних обчислень на комп’ютерних мережах: Навчальний посібник для студентів факультету кібернетики. – Київ. – 2011. – 60с
29. 8 ознак небезпечних хвороб, які можна «прочитати» по обличчю [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://transkarpatia.net/transcarpathia/different/117839-8-oznak-nebezpechnih-hvorob-iak-mozhna-prochitati-po-oblichchiu.html.

30. Як прочитати на обличчі симптоми недуги [Електронний ресурс] – Режим доступу до ресурсу: https://zik.ua/news/2017/08/27/na_dobre_zdorovya_yak_prochytaty_na_oblychchi_symptomy_nedugy_1156437.
31. О чем свидетельствуют синие губы [Електронний ресурс] – Режим доступу до ресурсу: https://health.sarbc.ru/o-chem-svidetelstvuyut-sinie-guby.html.
32. Красные глаза: причина, что делать [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://glazik.com.ua/ru/articles/blog/krasnye-glaza-prichina-chto-delat/.
33. О каких болезнях могут сигнализировать сухие губы [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://www.infox.ru/usefull/287/227793-o-kakih-boleznah-mogut-signalizirovat-suhie-guby.
34. Face Recognition [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://face-recognition.readthedocs.io/en/latest/readme.html.
35. Chaquopy Python SDK for Android [Електронний ресурс] – Режим доступу до ресурсу: https://chaquo.com/chaquopy/.

Додаток А

Приклади коду алгоритмів

Діагностика мішків під очима

```
def eye_bags_detection(left_eye, right_eye, image):
```

Вписуємо очі в прямокутник, який опускаємо вниз:

```
    (x, y, w, h) = cv2.boundingRect(np.array([left_eye]))
    coeff = 5
    roi = image[y + h + coeff:y + 2 * h + coeff, x:x + w]
    red_right_eye, green_right_eye, blue_right_eye = get_rgb(roi)
    roi_skin = image[y + 2 * h + coeff:y + 4 * h + coeff, x:x + w]
    red_skin_right_eye, green_skin_right_eye, blue_skin_right_eye = get_rgb(roi_skin)

    (x, y, w, h) = cv2.boundingRect(np.array([right_eye]))
    coeff = 5
    roi = image[y + h + coeff:y + 2 * h + coeff, x:x + w]
    red_left_eye, green_left_eye, blue_left_eye = get_rgb(roi)
    roi_skin = image[y + 2 * h + coeff:y + 4 * h + coeff, x:x + w]
    red_skin_left_eye, green_skin_left_eye, blue_skin_left_eye = get_rgb(roi_skin)
```

Рахуємо різницю по кожній компоненті RGB:

```
    red_diff_right = abs(red_right_eye – red_skin_right_eye)
    red_diff_left = abs(red_left_eye – red_skin_left_eye)
    green_diff_right = abs(green_right_eye – green_skin_right_eye)
    green_diff_left = abs(green_left_eye – green_skin_left_eye)
    blue_diff_right = abs(blue_right_eye – blue_skin_right_eye)
    blue_diff_left = abs(blue_left_eye – blue_skin_left_eye)

    red_diff = red_diff_right + red_diff_left
    green_diff = green_diff_right + green_diff_left
    blue_diff = blue_diff_right + blue_diff_left
```

Перевіряємо чи виконується наша умова:

```
    if red_diff > eye_bags_red_mean or (green_diff > eye_bags_green_mean + 0.089 *
eye_bags_green_dev and blue_diff >
                                eye_bags_blue_mean + 0.088 * eye_bags_blue_dev):
        return True
    return False
```

Діагностика синюватості губ

def blue_lips_detection(lips, image):

Обрізаємо необхідну компоненту з фото:

img = crop_image(lips, image)

Допоміжна функція підрахунку каналу a:

a_lip, sum = calculate_a_sum(img)

Перевіряємо чи виконується наша умова:

```
if a_lip / sum < blue_lips_mean – (blue_lips_diff_mean + blue_lips_diff_dev) * 0.85:
    return True
return False
```

Діагностика червонуватості очей

def red_eyes_detection(left_eye, right_eye, image):

Обрізаємо необхідну компоненту з фото:

img = crop_image(left_eye, image)

Допоміжна функція підрахунку каналу a:

a_right_eye, sum_right_eye = calculate_a_sum(img)

Аналогічно для правого ока:

```
img = crop_image(right_eye, image)
a_left_eye, sum_left_eye = calculate_a_sum(img)
```

Рахуємо компоненту 6:

```
a_right = a_right_eye / sum_right_eye
a_left = a_left_eye / sum_left_eye
```

Перевіряємо чи виконується наша умова:

```
if a_right > red_eyes_mean + red_eyes_dev or a_left > red_eyes_mean + red_eyes_dev:
    return True
return False
```

Діагностика набряків та асиметрії обличчя

```
def asymmetric_face_detection(left_eye, right_eye, left_eyebrow, right_eyebrow, mouth_left, mouth_right):
```

Визначаємо центроїди очей:

```
left_eye_center = left_eye.mean(axis=0).astype("int")
right_eye_center = right_eye.mean(axis=0).astype("int")
```

Рахуємо кут між центроїдами:

```
dY = right_eye_center[1] – left_eye_center[1]
dX = right_eye_center[0] – left_eye_center[0]
angle_eyes = np.degrees(np.arctan2(dY, dX)) – 180
```

Допоміжна функція для визначення найвищої точки брів:

```
max_left_eyebrow = highest_point(left_eyebrow)
max_right_eyebrow = highest_point(right_eyebrow)
```

Рахуємо компоненти 7-9:

```
eyes = abs(180 – abs(angle_eyes))
mouth = abs(mouth_left – mouth_right)
eyebrows = abs(max_left_eyebrow – max_right_eyebrow)
```

Перевіряємо чи виконується наша умова:

```
if eyes > asymmetric_eyes_mean + 2.14 * asymmetric_eyes_dev or mouth > asymmetric_mouth_mean + 2.14 * \
    asymmetric_mouth_dev or eyebrows > asymmetric_eyebrows_mean + 1.88 * asymmetric_eyebrows_dev:
  return True
return False
```

Діагностика опущених кутиків губ

```
def depression_detection(mouth, mouth_right_edge, mouth_left_edge):
```

Обчислюємо центроїд рота:

```
mouth_center = mouth.mean(axis=0).astype("int")
```

Рахуємо компоненту 10:

```
mouthh = (mouth_center[1] - mouth_left_edge + mouth_center[1] - mouth_right_edge) / 2
```

Перевіряємо чи виконується наша умова:

```
if mouthh < depressed_mouth_mean - 1.28 * depressed_mouth_dev:
    return True
return False
```

Діагностика сухості та лущення губ

```
def dry_lips_detection(mouth, image, mouth_up, mouth_down):
```

Обрізаємо необхідну компоненту з фото:

```
img = crop_image(mouth, image)
```

Метод виявлення контурів:

```
edges = cv2.Canny(img, 100, 200)
mouth_diff = mouth_down - mouth_up
```

Допоміжна функція підрахунку ненульових компонент:

```
lips = calc_not_zero(edges)
```

Перевіряємо чи виконується наша умова:

```
if mouth_diff > dry_lips_open_mean + 2 * dry_lips_open_dev:
    lips -= dry_lips_open_mouth_diff_mean
    lips -= 0.5 * dry_lips_open_mouth_diff_dev
if lips > dry_lips_mean + dry_lips_dev:
    return True
return False
```

Діагностика зміни кольору обличчя

```
def skin_color_detection(left_eyebrow, right_eyebrow):
```

Обрізаємо частину лоба з фото та підраховуємо канал a CIELAB:

```
    coeff = 25
    roi = image[left_eyebrow[2][1] - coeff * 2:left_eyebrow[2][1] - 2,
left_eyebrow[2][0]:right_eyebrow[2][0]]
```

```
a_forehead, sum = calculate_a_sum(roi)
a_forehead /= sum
```

Обрізаємо частини щік з фото та підраховуємо канал a CIELAB:

```
left_eye = face_landmarks_dict.get('left_eye')
right_eye = face_landmarks_dict.get('right_eye')
(x, y, w, h) = cv2.boundingRect(np.asarray(left_eye))
coeff = 5
roi_skin_left = image[y + 2 * h + coeff:y + 4 * h + coeff, x:x + w]
a_left_skin, sum = calculate_a_sum(roi_skin_left)
a_left_skin /= sum
(x, y, w, h) = cv2.boundingRect(np.asarray(right_eye))
coeff = 5
roi_skin_right = image[y + 2 * h + coeff:y + 4 * h + coeff, x:x + w]
a_right_skin, sum = calculate_a_sum(roi_skin_right)
a_right_skin /= sum
```

Повертаємо середнє значення:

```
return (a_forehead + a_left_skin + a_right_skin) / 3
```

Діагностика червонуватості обличчя

```
def redness_detection(face_location, lips):
```

Залишаємо на зображенні лише обличчя:

```
top, right, bottom, left = face_location[0]
roi = image[top:bottom, left:right]
```

Замальовуємо губи на зображенні:

```
img = cross_out_feature(lips, roi)
bounds = [
   ([106, 93, 175], [160, 140, 230])
]
```

Обраховуємо скільки пікселів не входять в діапазон червонуватості:

```
not_zero = detect_color(bounds, img)
return not_zero
```

Діагностика випадіння брів

```
def eyebrows_alopecia_detection(left_eyebrow, right_eyebrow):
```

Обраховуємо діапазон кольору шкіри:

```
    coeff = 25
    roi = image[left_eyebrow[2][1] - coeff * 2:left_eyebrow[2][1] - 2,
left_eyebrow[2][0]:right_eyebrow[2][0]]
    bounds = get_color_range(roi)
```

Обраховуємо кількість пікселів шкіри в зоні брів:

```
    img = crop_image(left_eyebrow, image)
    not_zero = detect_color(bounds, img)
    img = crop_image(right_eyebrow, image)
    not_zero += detect_color(bounds, img)
   return not_zero
```