НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

ННІ ЕНЕРГЕТИКИ, АВТОМАТИКИ І ЕНЕРГОЗБЕРЕЖЕННЯ

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ
В.о. завідувача кафедри
автоматики та робототехнічних систем
ім. акад. І.І. Мартиненка
(назва кафедри)

к.т.н., доц. _____ О.О. Опришко
(підпис) (ПІБ)

"____" _____ 2025 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему "БІОМЕДИЧНА СИСТЕМА ДІАГНОСТИКИ ЗАХВОРЮВАНЬ З ВИКОРИСТАННЯМ ШТУЧНОГО ІНТЕЛЕКТУ"

Спеціальність: 163 - "Біомедична інженерія"

Гарант освітньої програми

д.т.н., професор _____ Никифорова Л.Є.
(науковий ступінь та вчене звання) (підпис) (П.І.Б.)

Керівник бакалаврської кваліфікаційної роботи

к.т.н., доцент _____ Кіктєв М.О.
(науковий ступінь та вчене звання) (підпис) (П.І.Б.)

Виконав _____ Сорочан В.Ю.
(підпис) (П.І.Б.)

КИЇВ – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

ННІ ЕНЕРГЕТИКИ, АВТОМАТИКИ І ЕНЕРГОЗБЕРЕЖЕННЯ

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
автоматики та робототехнічних систем
ім. акад. І.І. Мартиненка
(назва кафедри)

к.т.н., доц. О.О. Опришко
(підпис) (ПІБ)

"____" _____ 2025 р.

ЗАВДАННЯ
на виконання бакалаврської кваліфікаційної роботи студентці Сорочана Владислава Юрійовича
(прізвище, ім'я, по батькові)

Спеціальність: 163 - "Біомедична інженерія"
1. Тема бакалаврської кваліфікаційної роботи: "Біомедична система діагностики захворювань з використанням штучного інтелекту ",
затверджена наказом ректора НУБіП України від "12" 11 2024 р. № 2023 "С"
2. Термін подання завершеної роботи на кафедру "31" травня 2025 р.
3. Вихідні дані до бакалаврської кваліфікаційної роботи:
   3.1. Завдання кафедри на виконання бакалаврської кваліфікаційної роботи.
   3.2. Нормативні документи по проектуванню біомедичних систем
   3.3. Наукова література з тематики бакалаврської кваліфікаційної роботи.
4. Перелік питань, які необхідно розробити:
   4.1. Аналіз стану медичної проблеми діагностики захворювань шкіри
   4.2. Дослідження захворювань шкіри обличчя як біологічного об’єкту.
   4.3. Огляд відомих математичних моделей обробки зображень для діагностування захворювань шкіри.
   4.4. Вибір технічних засобів для побудови біомедичної системи .
   4.5. Програмні засоби для розпізнавання захворювань шкіри
   4.5. Схеми біомедичної системи (структурна, функціональна, принципова електрична).
   4.6. Кошторисні розрахунки.
   4.7. Техніка безпеки і охорона праці.

5. Перелік графічних документів:
5.1. Структурна і функціональна схеми біомедичної системи діагностування захворювань шкіри на обличчі
5.2. Алгоритм розпізнавання захворювань шкіри на обличчі з використанням штучних нейронних мереж

Дата видачі завдання "19" грудня 2024 року

Керівник бакалаврської кваліфікаційної роботи _____ Кіктєв М.О.
(підпис) (П.І.Б.)

Завдання прийняв до виконання _____ Сорочан В.Ю.
(підпис) (П.І.Б.)

РЕФЕРАТ

Дипломна робота за темою «Біомедична система візуалізації динаміки змін серцевого ритму» виконана студенткою кафедри автоматизації та робототехнічних систем ННІ енергетики, автоматики та енергозбереження *Сорочаном Владиславом Юрійовичем* зі спеціальності *163 «Біомедична інженерія»* та складається зі: вступу; 3 розділів (медико-технічне обґрунтування проекту; структурна схема візуалізації динаміки змін серцевого ритму; розробка алгоритмічного, програмного та технічного забезпечення), висновків до кожного з цих розділів; загальних висновків; списку використаних джерел, який налічує 26 джерел, та 2 додатків. Кількість таблиць – 12, ілюстрація - 18. Загальний обсяг роботи 72 сторінок.

У цій дипломній роботі був розроблений зручний **веб-сервіс**, який допомагає виявляти проблеми зі шкірою обличчя, а саме - висипання. Цей сервіс працює на основі даних, отриманих після сканування обличчя користувача. Головна мета розробки полягала в тому, щоб користувачі могли легко завантажити зображення свого обличчя через веб-додаток та отримати швидкий результат щодо можливих шкірних захворювань.

Об'єктом дослідження є медична задача діагностування захворювань шкіри обличчя з використанням штучних нейронних мереж.

Предметом дослідження є комплекс програмних та технічних засобів для розпізнавання зображень акне по фотознімкам.

Ключові слова: акне, розпізнавання захворювань шкіри обличчя, згорткова нейронна мережа, біомедична система, функція активації, web-додаток.

ABSTRACT

The thesis on the topic "Biomedical system for visualizing the dynamics of heart rate changes" was completed by a student of the Department of Automation and Robotic Systems of the National Research Institute of Power Engineering, Automation and Energy Saving, *Vladyslav Sorochan,* majoring in 163 "Biomedical Engineering" and consists of: introduction; 3 sections (medical and technical justification of the project; structural diagram of visualization of the dynamics of heart rate changes; development of algorithmic, software and hardware support), conclusions for each of these sections; general conclusions; list of sources used, which includes 26 sources, and 2 appendices. The number of tables is 12, illustrations - 18. The total volume of the work is 72 pages.

In this thesis, a convenient web service was developed that helps to identify problems with facial skin, namely rashes. This service works on the basis of data obtained after scanning the user's face. The main goal of the development was to allow users to easily upload an image of their face via a web application and get a quick result regarding possible skin diseases.

The object of the research is the medical task of diagnosing facial skin diseases using artificial neural networks.

The subject of the research is a set of software and hardware tools for recognizing acne images from photographs.

Keywords: acne, recognition of facial skin diseases, convolutional neural network, biomedical system, activation function, web application.

ЗМІСТ

Вступ

Акне – це не просто косметична проблема. Це одне з найпоширеніших захворювань шкіри, що вражає як підлітків, так і дорослих, і має значний вплив на їхнє самопочуття та якість життя. Люди, що страждають від висипань, часто мають ослаблений імунітет через постійні запальні процеси. Це може призвести до загального невдоволення життям, погіршення психічного здоров'я та, особливо у підлітків, до низької самооцінки. Крім того, висипання на шкірі нерідко сигналізують про глибинні проблеми в організмі, такі як порушення роботи кишківника, печінки, підшлункової залози, жовчного міхура та інших органів травної системи. Неправильний спосіб життя, такий як вживання алкоголю, куріння, низька фізична активність та недостатнє споживання води, також може бути причиною шкірних проблем. До того ж, неправильний догляд за шкірою – відсутність регулярного очищення, тонізації та зволоження – значно погіршує ситуацію з висипаннями.

Рішення та його реалізація. Щоб допомогти у вирішенні цих проблем, ми розробили веб-сервіс, який може сканувати обличчя на наявність акне. В основі цього сервісу лежить нейронна мережа, навчена виявляти різні захворювання шкіри. Після того, як користувач завантажує фотографію свого обличчя, і, можливо, додає опис своїх симптомів, система аналізує дані. На основі цього аналізу, сервіс надає персоналізовані рекомендації щодо товарів та послуг, які можуть допомогти покращити стан шкіри.

Застосування сервісу. Цей інструмент є універсальним і може бути корисним як у косметології, так і в медичній сфері. Його використання допоможе не тільки діагностувати проблеми, а й сприятиме покращенню загального самопочуття людей, надаючи їм своєчасну інформацію та рекомендації для догляду за шкірою.

Мета дипломної роботи полягала у створенні зручного онлайн-сервісу. Завдяки йому люди, які мають висипання на обличчі, змогли б швидко зрозуміти, що це за проблема (отримати попередній "діагноз"), а також знайти та придбати необхідні засоби для її вирішення.

Об'єктом дослідження були, в першу чергу, самі люди, які стикаються з висипаннями на обличчі, а також методи запобігання таким проблемам.

Предметом дослідження став детальний розгляд різних типів висипань, зокрема акне, а також аналіз ефективності різноманітних препаратів, які допомагають у боротьбі з цими шкірними захворюваннями.

Для реалізації цього сервісу з розпізнавання акне та надання рекомендацій за допомогою штучного інтелекту (а саме згорткової нейронної мережі) ми виконали такі задачі:

- Розробили спеціальні алгоритми та моделі, які здатні "навчатися" та точно виявляти шкірні захворювання.
- Порівняли різні підходи до виявлення акне, щоб знайти найефективніший.
- Визначили, які моделі штучного інтелекту забезпечують найвищу точність розпізнавання.
- Створили зручний веб-додаток, щоб користувач міг легко завантажити своє фото та отримати всі потрібні поради та дані.
- Провели ретельне тестування та перевірку роботи створеного веб-сервісу, використовуючи реальні знімки шкірних проблем.
- Дослідили, як ця система може бути застосована в різних сферах – від косметології та охорони здоров'я до фармацевтики.

РОЗДІЛ 1
МЕДИКО-ТЕХНІЧНЕ ОБҐРУНТУВАННЯ ПРОЕКТУ

1.1. Актуальність та ідея веб-сервісу для шкіри

Багато людей, які страждають від проблем зі шкірою, наприклад, від акне, часто не мають можливості звернутися до косметолога чи дерматолога. Через це вони нерідко починають лікуватися самостійно, не маючи повного розуміння своєї проблеми, що може бути небезпечним або навіть погіршити їхній стан. Тому виникла нагальна потреба у простому, швидкому та доступному способі самостійно визначити тип висипань і отримати рекомендації для покращення стану шкіри.

Крім того, ми бачимо, що людям птрібні персональні поради щодо догляду, адже проблеми зі шкірою не завжди пов'язані лише з підлітковим віком. Часто вони є наслідком неправильного догляду або нездорового способу життя (погане харчування, стрес, недосипання). Саме тому є велика потреба у веб-сайті, який даватиме поради не тільки про косметичні засоби, а й про здоровий спосіб життя в цілому.

Якщо говорити просто, **веб-сервіс** — це така собі програма, яка "живе" в інтернеті і вміє спілкуватися з іншими програмами. Вона має чітко визначені "правила" спілкування (описані зазвичай у спеціальних файлах, як XML), щоб будь-яка інша програма могла до неї звернутися і отримати потрібну інформацію чи виконати якусь дію.

Часто веб-сервіси працюють за принципом "клієнт-сервер", де ваш комп'ютер чи телефон (клієнт) надсилає запит, а віддалений сервер (де "живе" веб-сервіс) відповідає. І хоча іноді їх опис потрібен для автоматичної роботи, загалом, це просто спосіб для програм "розуміти" одна одну.

Завдяки інтернету, веб-сервіси значно спрощують взаємодію між людьми та різними інформаційними системами. Вони використовують цілий набір

стандартних правил, щоб дані можна було обмінювати, а бізнес-процеси ставали доступними через інтернет з будь-якого пристрою.

Зокрема, веб-сервіси дозволяють:

- Описати, що саме вони вміють робити, і дозволити іншим програмам ними користуватися.
- Допомогти зацікавленим сторонам знайти потрібний сервіс.
- Використати цей сервіс після того, як він знайдений.
- Отримати зрозумілий результат взаємодії.

Отже, веб-сервіси створюють відкриту інформаційну "інфраструктуру", завдяки якій різні організації можуть:

- Об'єднувати свої внутрішні процеси.
- Швидко налагоджувати зв'язок зі своїми партнерами.
- Пропонувати свої послуги іншим як готові "блоки", якими можна користуватися на певних умовах.

По суті, веб-сервіси — це фундамент для ще більших змін, які перетворюють Інтернет на універсальне ділове середовище, де легко відбувається "публікація", "знаходження" та "використання" різноманітних бізнес-послуг.

Веб-сервіси — це, по суті, новий спосіб створювати логіку для програм і з'єднувати між собою абсолютно різні додатки, використовуючи спільні стандарти. Завдяки їм будь-яка функція програми стає доступною через інтернет.

Їх робота базується на кількох простих принципах:

- Той, хто створює веб-сервіс, чітко визначає, як до нього надсилати запити і в якому форматі він буде відповідати.
- З будь-якого комп'ютера, підключеного до інтернету, можна надіслати запит до цього веб-сервісу.
- Веб-сервіс виконує задані дії і відправляє назад результат.

1.2. Аналіз методів виявлення захворювань шкіри обличчя

1.2.1. Спосіб виявлення захворювань шкіри

Коли ми намагаємося розпізнати проблеми зі шкірою за цифровими знімками, важливо розуміти, що не всі фотографії однаково корисні. Умовно кажучи, ми можемо виділити три типи "якості" зображень, від яких залежить, наскільки точно ми зможемо дати пораду чи поставити попередній діагноз:

1. **Погані фото:** Це знімки, де є багато засвітів, сильних фільтрів чи інших спотворень. За ними майже неможливо щось зрозуміти.
2. **Відео:** Відеозаписи обличчя, хоч і динамічні, теж мають свої особливості, які можуть впливати на точність аналізу.
3. **Хороші фото:** Найкраще працюють свіжі, якісні фотографії без фільтрів. Вони дозволяють отримати найточніші рекомендації. Чим краща якість фото, тим надійнішим буде результат.

Щоб зробити дослідження акне простішими та точнішими, ми зібрали новий, покращений набір фотографій, який назвали **AcneSCU**. На відміну від старих баз даних, AcneSCU містить зображення з вищою якістю, вже "відкалібровані" (нормалізовані), а також має дуже детальні позначки щодо різних видів акне, що робить анотації більш точними.

Ми помітили, що коли ми обрізаємо зображення високої роздільної здатності з AcneSCU, іноді частини висипань можуть опинитися "за кадром" (на контурі обрізки). Це може заважати навчанню нашої системи. Щоб уникнути такої неточності, ми придумали простий, але ефективний спосіб підготовки даних – так зване "масковане обрізання" (masked crop), яке гарантує, що всі часткові ураження будуть враховані або коректно оброблені.

Крім того, ми розробили нову структуру, названу **SADH**, щоб вирішити проблему "плаского градієнта впевненості" в класифікації (це коли системі важко "чітко" визначитися з діагнозом). У нашій системі класифікація (визначення типу проблеми) та локалізація (де саме вона знаходиться)

обробляються різними частинами програми, що робить "рішення" системи більш чіткими та впевненими.

Для того, щоб точніше передбачати, де саме знаходяться висипання, ми додали спеціальний блок, що використовує **NWD** (Normalized Wasserstein Distance) разом із втратою **SBCE**. Ми довели, що цей підхід не тільки покращує зв'язок між тим, наскільки система "впевнена" у своїй класифікації, і наскільки точно вона вказує на місце ураження (IoU), а й загалом підвищує ефективність виявлення проблем.

А) Б)

В)

Рисунок 1.1 – Приклади детекції захворювань шкіри

1.3. Порівняльний аналіз існуючих детекторів акне

1.3.1. Як виявити захворювання шкіри за зображеннями

Коли ми намагаємося розпізнати проблеми зі шкірою за допомогою фотографій або відео, дуже важливо враховувати якість вихідного матеріалу. Можна виділити три основні категорії зображень, які по-різному впливають на точність аналізу і вимагають різних підходів:

- **Фото з фільтрами або засвітами:** Знімки, які сильно оброблені (наприклад, з фільтрами Instagram) або мають пересвічені ділянки, значно спотворюють реальний стан шкіри. Використовувати такі фото для діагностики дуже складно, оскільки нейромережа не побачить справжньої картини. Тому ми відразу розуміємо, що такий матеріал не підходить для точного розпізнавання. Дослідження показали, що складніший двошаровий алгоритм розпізнавання акне на основі глибокого навчання працює значно ефективніше, ніж простіші (одношарові) рішення, підвищуючи швидкість і точність виявлення.
- **Відео в реальному часі:** Хоча зйомка відео в реальному часі здається привабливою, вона підходить лише за умови використання дуже якісної камери без засвітів. Інакше виникнуть проблеми з обробкою відеопотоку та отриманням достовірних даних.
- **Свіжі, необроблені фотографії:** Найкращим варіантом для точної діагностики є нещодавно зроблені фотографії без будь-яких фільтрів або засвітів. Саме такі знімки найбільш правдиво відображають стан шкіри. Аналізуючи такі фото, ми можемо надавати максимально точні рекомендації щодо догляду та відповідних засобів. Наші випробування з використанням детектора Coarse-to-Fine Network (CFN) показали високу середню точність у 89%, а швидкість обробки становила лише 0,3 секунди.
-

1.3.2. Як ми формуємо рекомендації по догляду

Процес надання рекомендацій розбито на кілька етапів. Однією з головних проблем є те, що на одному зображенні може бути багато уражених ділянок, і буває важко забезпечити, щоб кожна з них була повністю "обрізана" для аналізу. Якщо ми просто відкинемо або, навпаки, спробуємо використати частково обрізані ураження, це призведе до великої кількості неточних даних і знизить якість роботи системи.

Щоб вирішити цю проблему, ми запропонували досить простий, але ефективний підхід: "маскування" всіх частково обрізаних уражень. Кожне вихідне зображення розбивається на менші частини (підзображення) розміром 1024×1024 пікселів. Щоб не пропустити жодної ділянки і включити всю область зображення в навчання, ми обрізаємо їх так, щоб вони трохи перекривалися. Це робиться спочатку по горизонталі, а потім по вертикалі. Завдяки такому перекриттю ми мінімізуємо ризик пропустити навіть найдрібніші ураження. Після цієї підготовки даних ми застосовуємо спеціальні алгоритми, такі як AcneSCU та RPN, для подальшого аналізу.

1.3.3. Аналіз підходів до розпізнавання шкірних проблем та підбору догляду

Коли ми намагаємося "діагностувати" стан шкіри за цифровими фотографіями, дуже важливо, якого вони якості. Умовно, можна виділити три типи зображень, від яких сильно залежить точність результату:

Невдалі фото: Це знімки, де багато зайвого світла (засвіти) або використані сильні фотофільтри. Такі фотографії дуже спотворюють реальний вигляд шкіри, тому нашій системі буде важко правильно її оцінити. Використання таких зображень значно знижує точність роботи нейронної мережі, тому для нашої системи вони не підходять. Наші дослідження показали, що складніші алгоритми розпізнавання акне (двошарові) на основі глибокого

навчання працюють значно краще, ніж простіші (одношарові), забезпечуючи вищу швидкість і точність виявлення.

Відео в реальному часі: Хоча зйомка відео в реальному часі здається зручною, вона буде ефективною лише за умови використання дуже якісної камери без засвітів. В іншому випадку, обробка такого відео може призвести до проблем з точністю.

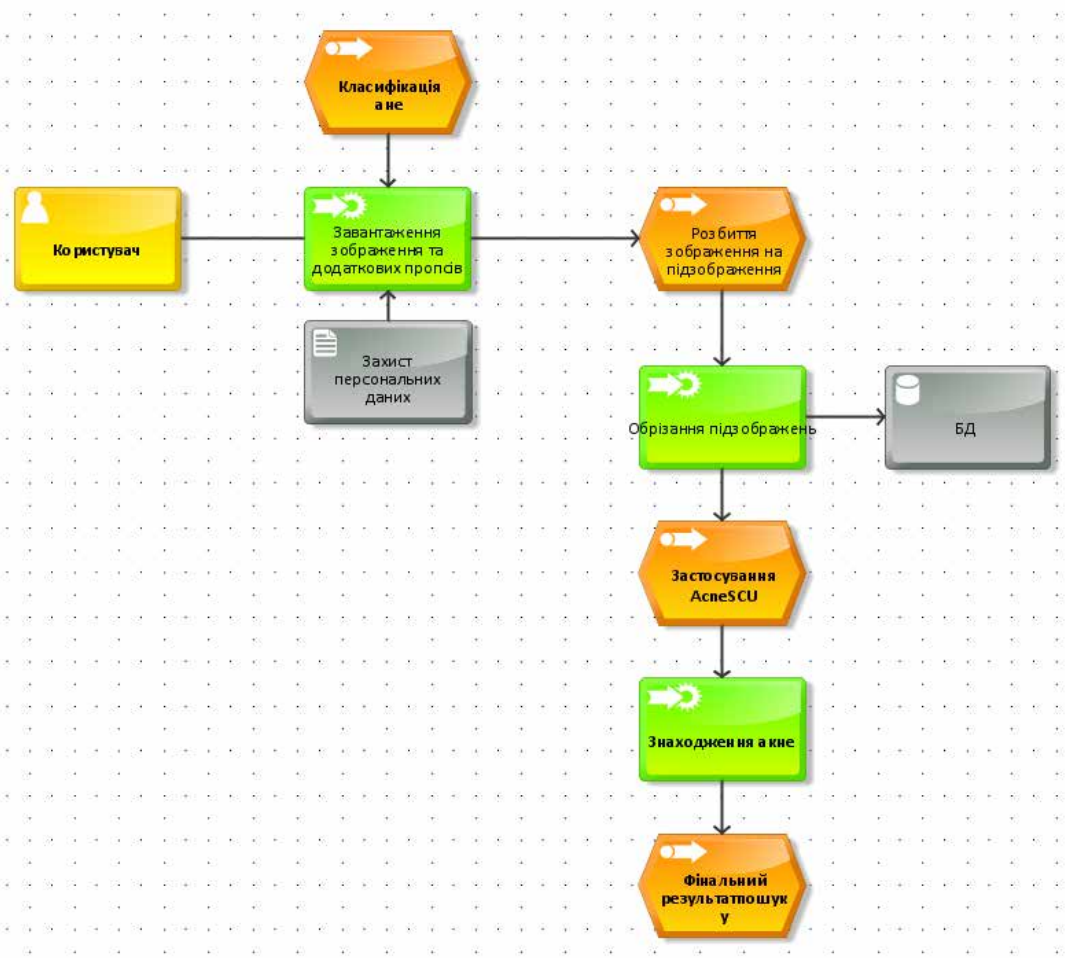


Рисунок 1.2. Модель об’єкта дослідження

Свіжі, необроблені фото: Найкращий варіант – це нещодавно зроблені фотографії, які не оброблялися фільтрами та не мають засвітів. Вони найбільш точно передають реальний стан шкіри. Аналізуючи такі якісні знімки, наша система може надати максимально точний список рекомендованих засобів та

порад з догляду. Наприклад, детектор Coarse-to-Fine Network (CFN), який ми вивчали, показав дуже високу точність (89%) і швидку роботу (0,3 секунди на аналіз).

1.3.4. Як ми формуємо рекомендації щодо догляду

Процес надання рекомендацій базується на певних етапах. Наприклад, ми розбиваємо кожне завантажене зображення на кілька менших частин (підзображень). Однак, коли на фото багато уражень, важливо, щоб кожне з них було повністю "видно" і не обрізалося. Якщо ми будемо ігнорувати або неправильно обробляти частково обрізані ураження, це призведе до неточних результатів.

Щоб уникнути цього, ми розробили простий, але ефективний метод: ми "маскуємо" всі частково обрізані ураження. Кожне таке підзображення має розмір 1024x1024 пікселі. Щоб переконатися, що вся ділянка шкіри на фотографії буде врахована при навчанні, ми створюємо достатню кількість таких підзображень, які трохи перекривають одне одного. Спочатку ми робимо це по горизонталі, потім по вертикалі. Завдяки такому перекриттю ми мінімізуємо ймовірність того, що якісь ураження будуть пропущені. Після такої підготовки даних ми використовуємо спеціальні алгоритми, такі як **AcneSCU** та **RPN**, для подальшого аналізу.

Згорткові нейронні мережі (ЗНМ) – це основа нашого підходу. Вони складаються з багатьох "шарів", які працюють разом. **Згорткові шари** шукають локальні закономірності на зображенні (наприклад, краї або текстури), "зважуючи" пікселі. **Шари об'єднання** потім спрощують дані, зменшуючи їх розмір, що допомагає системі фокусуватися на головному. В кінці зазвичай стоїть **класифікатор**, який за допомогою кількох повністю пов'язаних шарів і функції softmax визначає, до якої категорії належить виявлена проблема.

Ми порівнювали різні типи ЗНМ, щоб знайти найкращий.

RPN (Region Proposal Network): Це дуже популярний елемент у системах виявлення об'єктів. Він, по суті, є спеціальною нейромережею, яка допомагає знаходити потенційні ділянки на зображенні, де можуть бути об'єкти (в нашому випадку – висипання). RPN має дві "гілки": одна для класифікації (що це?) і одна для локалізації (де це?). Були різні підходи до того, як саме RPN прогнозує розташування: хтось передбачає верхній лівий і нижній правий кути (LAW), хтось додав центральну точку (Cornernet), хтось оцінює ступінь збігу рамки з реальним ураженням (MEA), а хтось досліджує зв'язки між різними рівнями деталей зображення (LOU). Маска R-CNN, наприклад, навіть додала функцію сегментації, щоб точно виділяти контури об'єктів. Цікаво, що дослідження (WUA) показали: повністю з'єднані мережі краще справляються з класифікацією, тоді як LAW краще працює для визначення точного розташування.

AcneNet: Це спеціальна глибока нейронна мережа, яку створили саме для розпізнавання акне на обличчі. Її "навчили" на величезній кількості зображень (10 000 фото облич з акне та без). AcneNet використовує відому архітектуру ResNet і досягає дуже високої точності – понад 90%.

MobileNetv2: Це "легка" згорткова мережа, розроблена спеціально для роботи на мобільних пристроях. Вона використовує хитрі "глибинно розділені" згортки, які дозволяють значно зменшити обсяг пам'яті, необхідний для роботи. Ця мережа має менше нелінійності (що, за словами розробників, навіть покращує її роботу) і використовує "інвертовані пропускні з'єднання", що ще більше зменшує кількість обчислень.

DeepAcne: Ще одна глибока нейронна мережа, призначена для розпізнавання акне. Вона також навчалася на великому наборі даних (понад 10 000 зображень) і активно використовує попередню обробку фотографій, таку як обрізання, розмиття або підсвічування, щоб покращити результати розпізнавання.

GoogleNet: Це досить глибока нейронна мережа (22 шари), але при цьому вона має дуже мало параметрів для навчання (лише 4 мільйони, порівняно з 60

мільйонами в AlexNet). Її головна "фішка" – використання так званих "початкових шарів" (inception layers). Це означає, що вона одночасно застосовує фільтри різного розміру, а потім об'єднує їхні результати. Ідея полягає в тому, що фільтри різного розміру можуть виявляти різні візерунки на вході.

DenseNet201: Ця нейронна мережа відома своєю потужністю і високою продуктивністю. Її назва походить від того, що кожен шар у мережі пов'язаний з *усіма* попередніми шарами. Така архітектура допомагає краще передавати інформацію всередині мережі та заохочує повторне використання вивчених особливостей. DenseNet201 дуже конкурентоспроможна з іншими передовими мережами, при цьому вимагаючи менше параметрів для навчання, ніж, наприклад, AlexNet.

Ми також розглянули вже існуючі рішення для виявлення акне. Наприклад, **SkinVision** – це мобільний додаток, який дозволяє користувачам аналізувати зображення своєї шкіри не тільки для виявлення пігментних плям і зневоднення, а й акне. Цей додаток використовує штучний інтелект для надання точних результатів діагностики.

Рисунок 1.3. Приклад конкурентів

У цьому розділі ми розглянемо кілька вже існуючих програм та сервісів, які так чи інакше працюють зі шкірою або розпізнаванням облич, щоб зрозуміти, які є підходи та чого досягли інші.

MDacne: Це мобільний додаток, який спеціалізується на виявленні акне. Він аналізує фотографії обличчя і не тільки допомагає визначити наявність висипань, а й надає індивідуальні поради щодо догляду за шкірою, враховуючи

її тип та ступінь важкості акне. Це як особистий помічник у кишені для догляду за проблемною шкірою.

SkinVisionary: Цей онлайн-сервіс також використовує технології штучного інтелекту, але його фокус ширший – він діагностує різні дерматологічні проблеми, включаючи акне. Користувачі просто завантажують фотографії своєї шкіри і отримують детальний звіт про її стан. Це такий собі "віртуальний дерматолог", доступний онлайн.

Cognitec: Ця компанія пропонує потужні рішення для розпізнавання облич (FRS - Face Recognition System). Їхня система "FaceVacs" є гнучкою і масштабованою, що робить її ідеальною для великих організацій, які шукають рішення для безпеки або контролю великих скупчень людей. Cognitec займає значну частку ринку розпізнавання облич і може надати ефективні рішення з мінімальними витратами на налаштування. Це більше про безпеку, ніж про шкіру.

DeepVision AI: Ця компанія також спеціалізується на рішеннях для розпізнавання облич (FRS), але орієнтована на маркетинг, планування, а також на компанії, яким потрібна перевірка обличчя з метою безпеки. DeepVision найкраще підходить для забудовників, роздрібних компаній та маркетингових агентств. Їхня система зі штучним інтелектом може використовуватися для контролю потоку людей, виявлення інцидентів та навіть розпізнавання транспортних засобів, надаючи автоматичний аналіз відео. Це ідеальне рішення для середніх та великих компаній, яким потрібні гнучкі та масштабовані можливості.

Face++: Цей сервіс відмінно підходить для порівняння обличчя з уже існуючими збереженими зображеннями. Завдяки детальному набору функцій і гнучким ціновим моделям, Face++ є чудовим вибором для компаній, які тільки починають інтегрувати технології розпізнавання облич у свої процеси.

Рисунок 1.4. Приклад конкрентів

1.4. Постановка задачі

Наша головна мета в цій роботі – створити зручний **онлайн-сервіс**. Ми хочемо, щоб будь-хто, хто стикнувся з висипаннями на обличчі, міг швидко зрозуміти, що це за проблема, і одразу ж отримати рекомендації щодо потрібних засобів для її вирішення. Крім того, сервіс буде давати персональні поради, як уникнути акне в майбутньому – наприклад, скільки води пити на день чи скільки кроків проходити.

Щоб наш сайт виглядав сучасно та був зручним для користувачів (це називається front-end), ми обрали технологію **React** разом з бібліотекою **Next.js**, які базуються на мові програмування **JavaScript**. Для красивого оформлення ми використовували **SCSS** (це такий "просунутий" CSS) і методику **BEM** для порядку в стилях.

Для зберігання всієї інформації (бази даних) ми вибрали **PostgreSQL**, оскільки це надійний і добре підходящий варіант. А для "мозку" сервісу, який обробляє запити і виконує складні операції (це називається back-end), ми також використали **Next.js**.

Серцем нашої системи, яка власне й розпізнає акне, є потужна **згорткова нейронна мережа RPN** у поєднанні з алгоритмом **AcneSCU**.

Основні вимоги до програми

Ми визначили, що наш сервіс повинен вміти робити, а також яким він має бути.

Функціональні вимоги:

- Запускати процес аналізу зображення обличчя на наявність захворювань.
- Визначати (розпізнавати) ці захворювання.
- Давати можливість користувачеві завершити процес розпізнавання.
- Мати гнучкий інтерфейс, який можна було б змінювати.
- Показувати користувачеві список рекомендованих товарів.

- Надавати список загальних рекомендацій щодо догляду за шкірою обличчя.

Нефункціональні вимоги:

- Мати простий та зрозумілий у використанні дизайн.
- Мати зручний та логічний каталог товарів.
- Формувати чіткі та корисні рекомендації щодо товарів.
- Підтримувати українську та англійську мови інтерфейсу.

Зацікавлені сторони у проекті

Ми проаналізували, для кого наш сервіс буде найкориснішим і хто може бути в ньому зацікавлений.

Всередині нашої сфери:

- Косметологи, які зможуть використовувати сервіс для попередньої оцінки стану шкіри клієнтів.
- Біологи, які можуть використовувати дані для досліджень.
- Організації, що вивчають проблеми акне, для збору даних та аналізу.
- Фармацевти, для розуміння потреби у певних препаратах.
- Дерматологи, як додатковий інструмент для консультацій.

Ззовні (ширша аудиторія):

- Міністерство охорони здоров'я України, як потенційний інструмент для профілактики.
- Фармацевтичні компанії, для розуміння ринку та потреб споживачів.
- Інші компанії-конкуренти, які вже мають подібні рішення.
- Звичайні люди, які шукають самостійну допомогу та поради щодо догляду за шкірою.

Хто "за" і хто "проти" нашого сервісу.

- **Протагоністи (ті, хто підтримує):** Це ми, розробники системи, самі користувачі, які отримають користь, а також косметологи, які сканують проблеми з обличчям та використовують наш сервіс.

- **Антагоністи (ті, хто може мати заперечення):** Це, ймовірно, наші конкуренти, які вже пропонують схожі рішення для акне. Також це можуть бути законодавчі органи, які можуть встановлювати певні обмеження щодо використання нейронних мереж у медичних цілях.

Висновок до першого розділу

Отже, перший розділ цієї дипломної роботи повністю присвячений вступу до теми. Ми описали, де саме може бути використаний наш "детектор" акне, проаналізували різні методи виявлення шкірних проблем, а також визначили, хто є зацікавленими сторонами проекту. Ми розглянули вже існуючі схожі сервіси, дали короткі пояснення щодо типів акне, які ми будемо виявляти, і, звичайно, чітко сформулювали, що саме ми робимо: яка мета, що досліджуємо, які завдання ставимо, і які вимоги (як функціональні, так і нефункціональні) пред'являємо до нашого програмного продукту.

Розділ 2
РОЗРОБКА АРХІТЕКТУРИ ПІДСИСТЕМИ РОЗПІЗНАННЯ ЗАХВОРЮВАНЬ ШКІРИ

2.1. Функціональний аналіз

Сьогодні технології розпізнавання об'єктів стали набагато досконалішими. Це стосується і нашої сфери – виявлення таких проблем, як захворювання шкіри. Тепер існують методи, які дозволяють дуже точно "бачити" різні об'єкти (або, у нашому випадку, ознаки хвороб) без необхідності використовувати дороге чи складне обладнання.

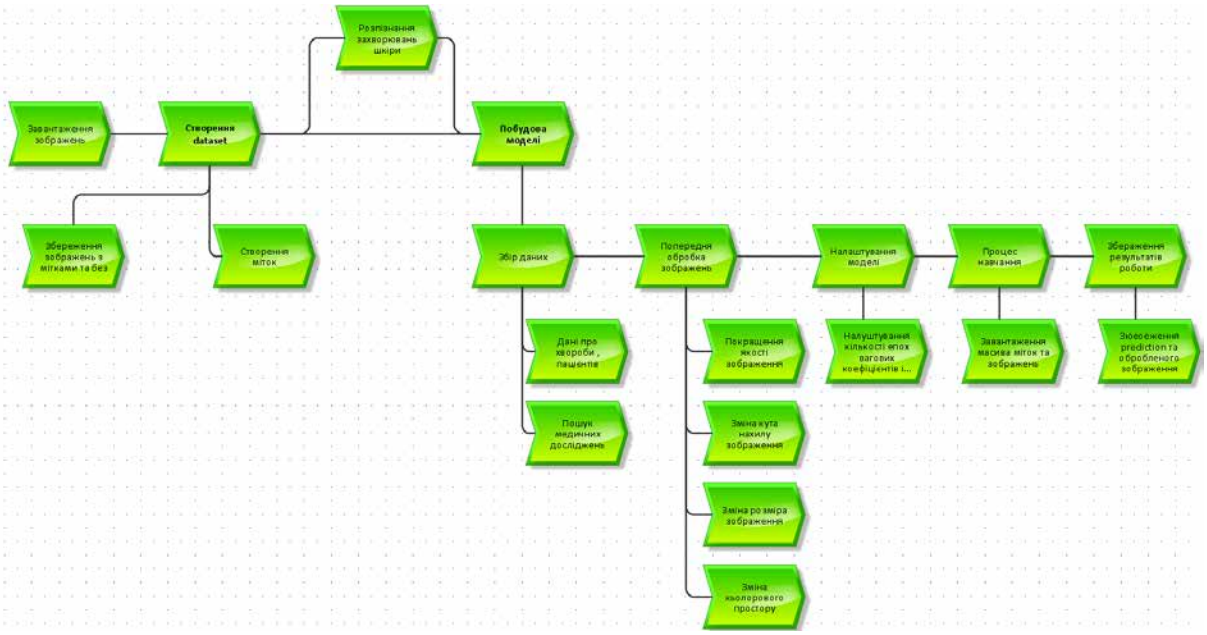


Рисунок 2.1 − Дерево функцій модуля acne detection

Ці методи, що належать до сфери **глибинного навчання** у **комп'ютерному зорі**, по суті, працюють у два основні етапи:

1. **"Навчання" системи:** Спочатку ми створюємо і тренуємо спеціальну "модель" (як мозок для комп'ютера), яка вчиться розпізнавати потрібні нам ознаки на величезній кількості прикладів.

2. **"Аналіз" зображень:** Після того, як модель навчена, вона може швидко і ефективно аналізувати нові зображення, виявляючи на них те, чому її навчили.

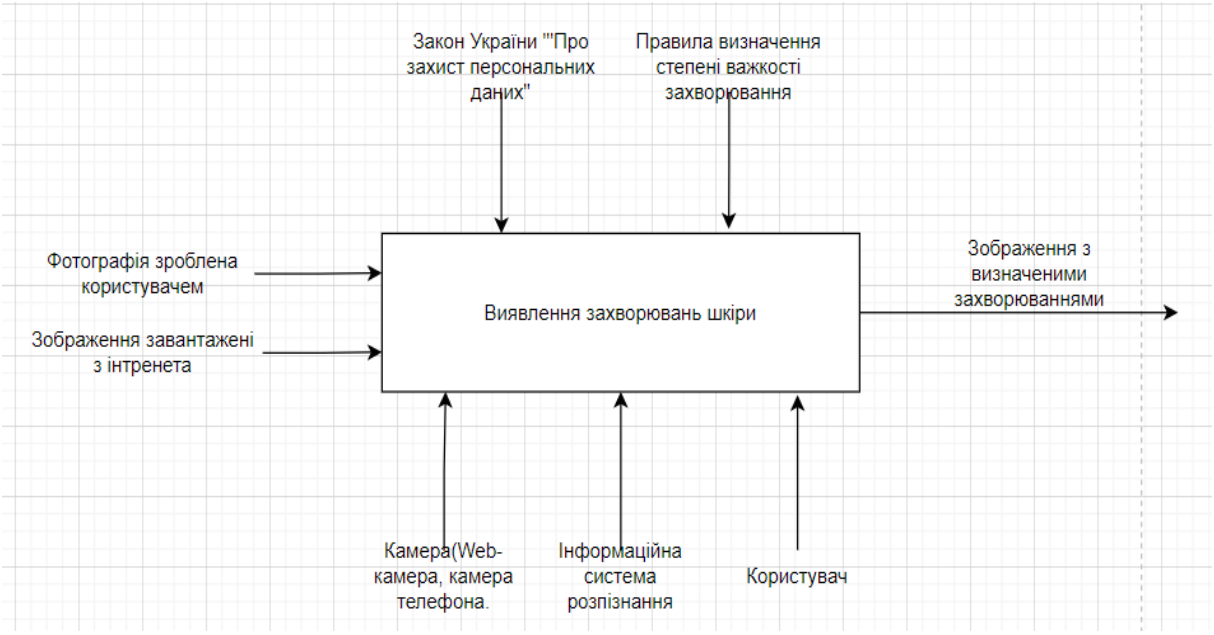


Рисунок 2.2 – IDEF0 ВЗШ (виявлення захворювань шкіри)

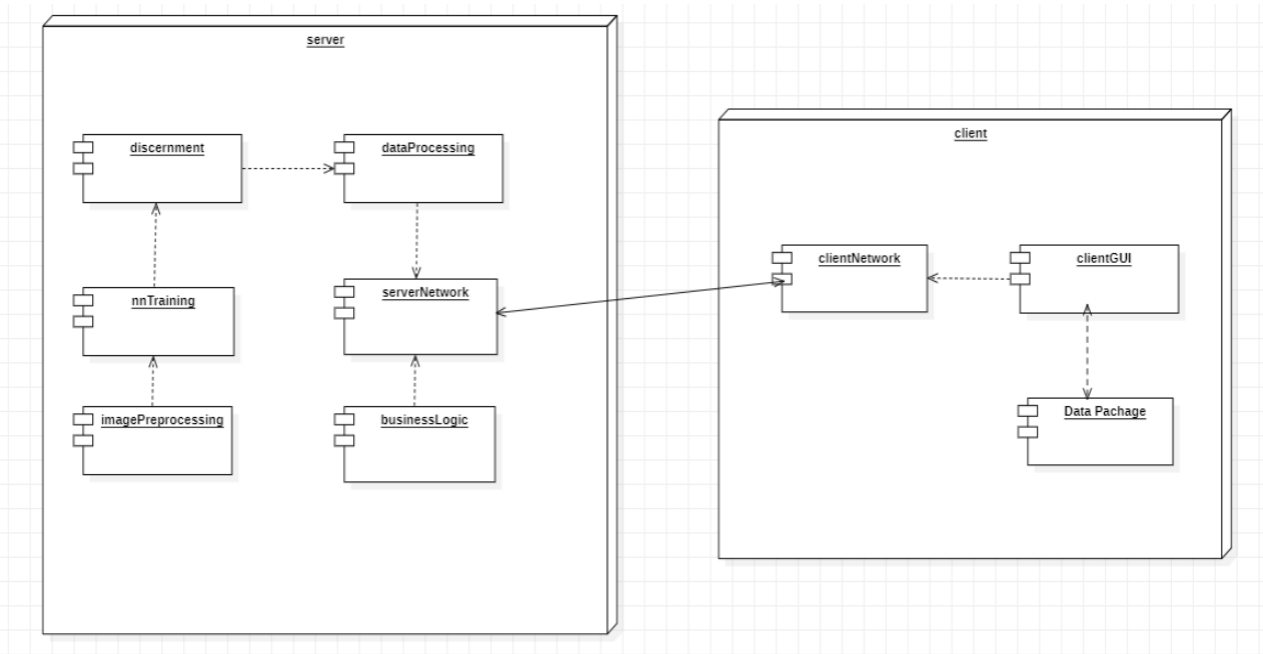


Рисунок 2.3 – Компонентна архітектура застосунку

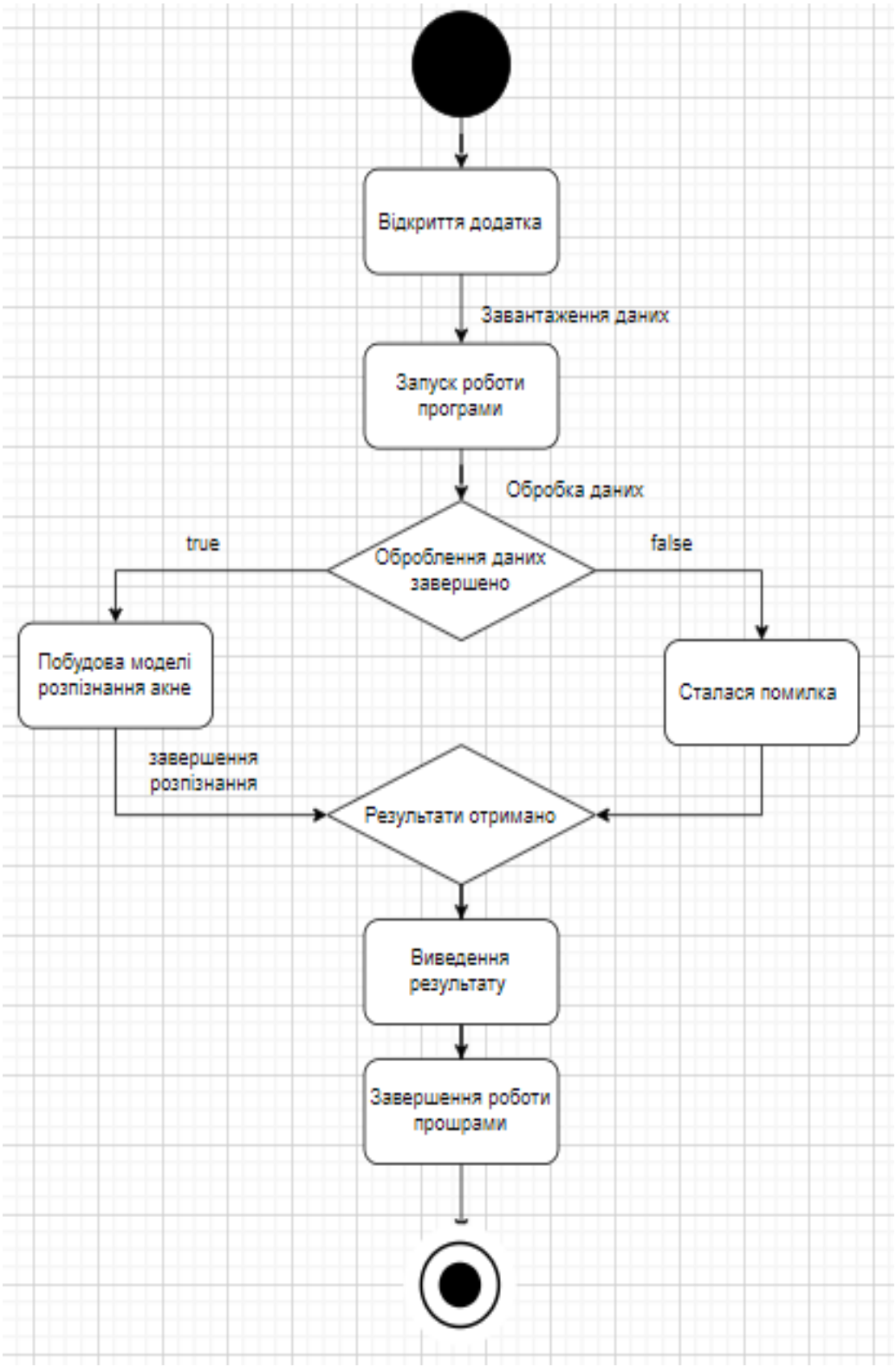


Рисунок 2.4 – Життєвий цикл додатку

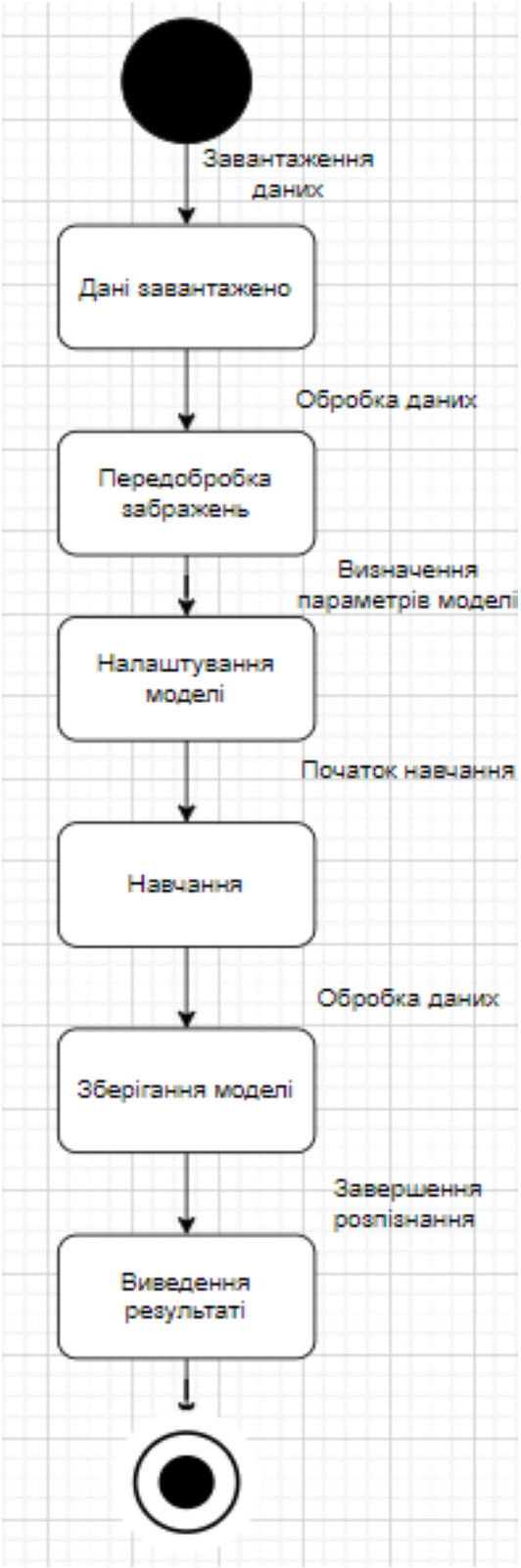


Рисунок 2.6 – Життєвий цикл навчання моделі

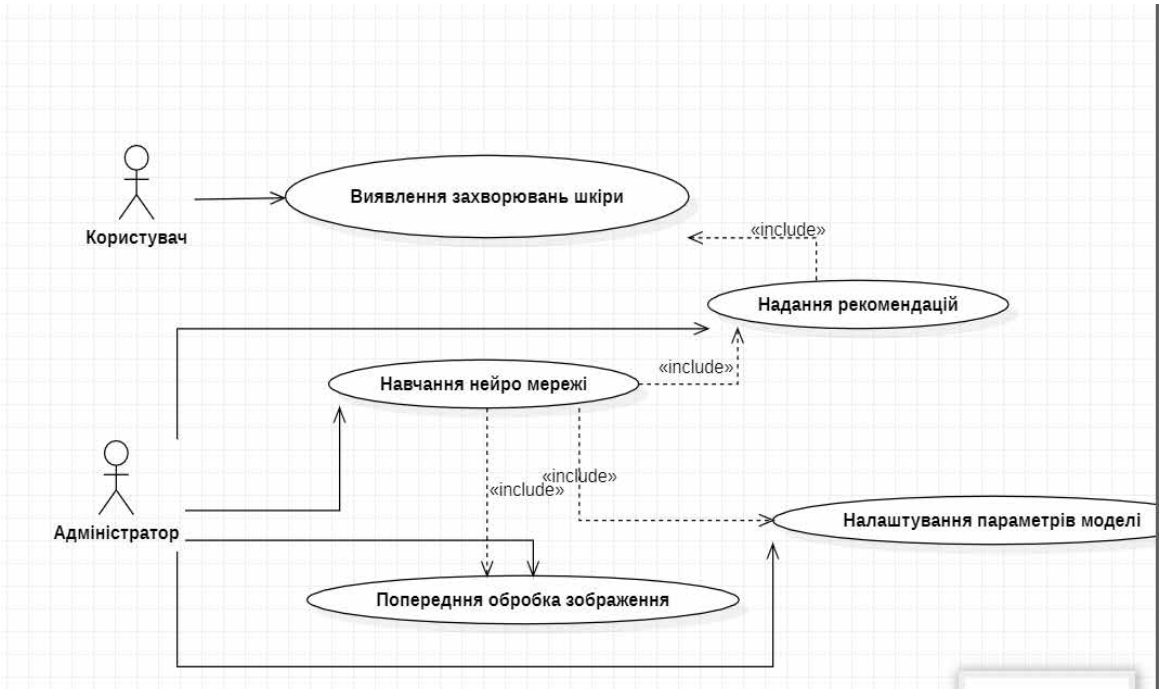


Рисунок 2.7 – UML діаграма процедентів

2.2. Узагальнена технічна архітектура

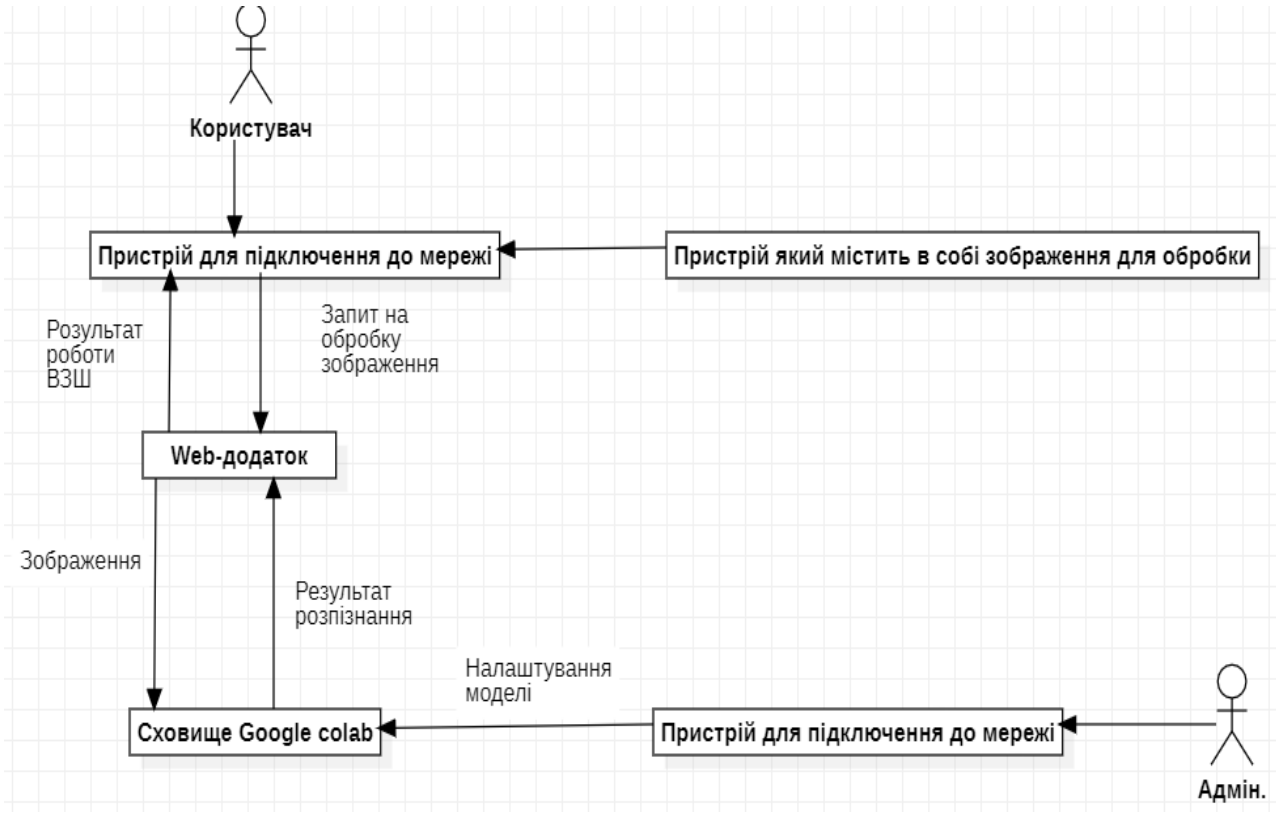
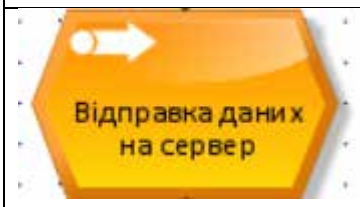
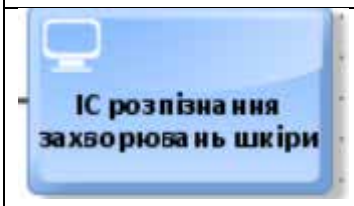
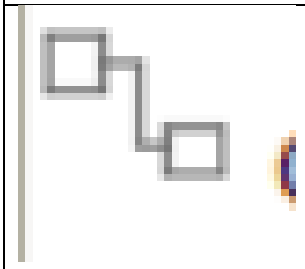


Рисунок 2.8 – Узагальнена архітектура системи

Таблиця 2.1 – Event-Driven Process Chain

	Activity – кроки, які необхідно виконати – описує, що відбувається під час процесу, тобто що саме робиться. Вони є основними елементами процесу. Приклад: завантажити зображення.
	Event– стан системи, процесу, що впливає на виконання функцій або керує ними. Наприклад: відправка даних на сервер
	IT–system (IT–система) – посилання на EDP (electronic data processing): дії можуть виконуватися вручну або автоматично. Автоматизовану діяльність здійснюють IT–системи. Приклад: AWS/EC2.
	Role – розподіл ролей для діяльності – ілюструє, хто виконує діяльність. Приклад: адміністратор.
	Потік даних – потік даних у процесі: вхідні та вихідні дані. Процес генерує дані або вимагає даних для продовження. Ці дані моделюються як вхідні або вихідні дані діяльності.

	Правила – описують альтернативи робочого процесу і таким чином ілюструють можливі варіанти виконання. Доступні такі символи правил: АБО: можна використовувати будь–яку кількість шляхів. І: усі наступні/попередні шляхи застосовуються. Виключаюче АБО: може мати місце лише один з наступних/попередніх варіантів (як у цьому прикладі: або дані обробилися коректно, або виникла помилка).

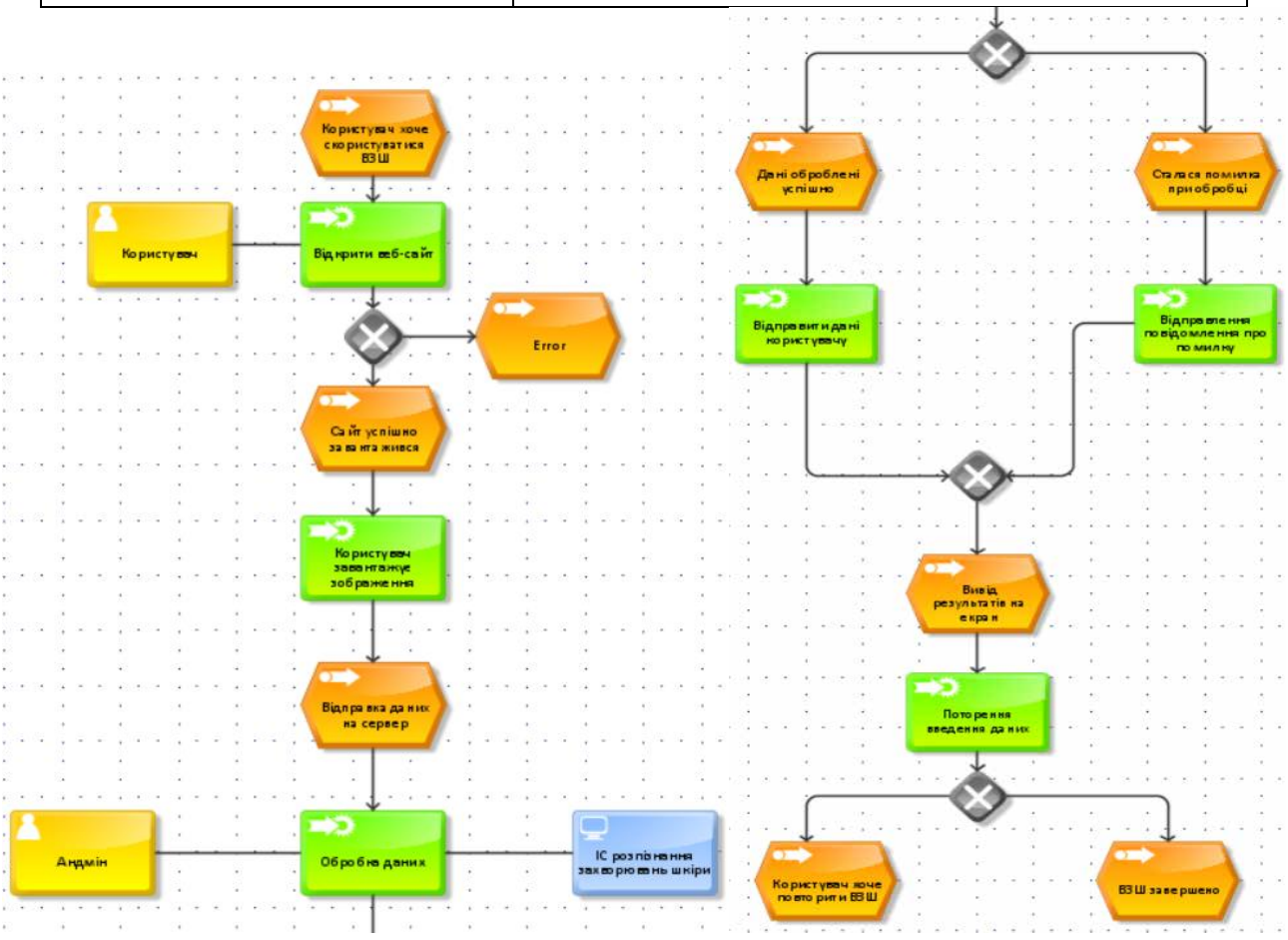


Рисунок 2.9 - Діаграма бізнес-процесу в (EPC)

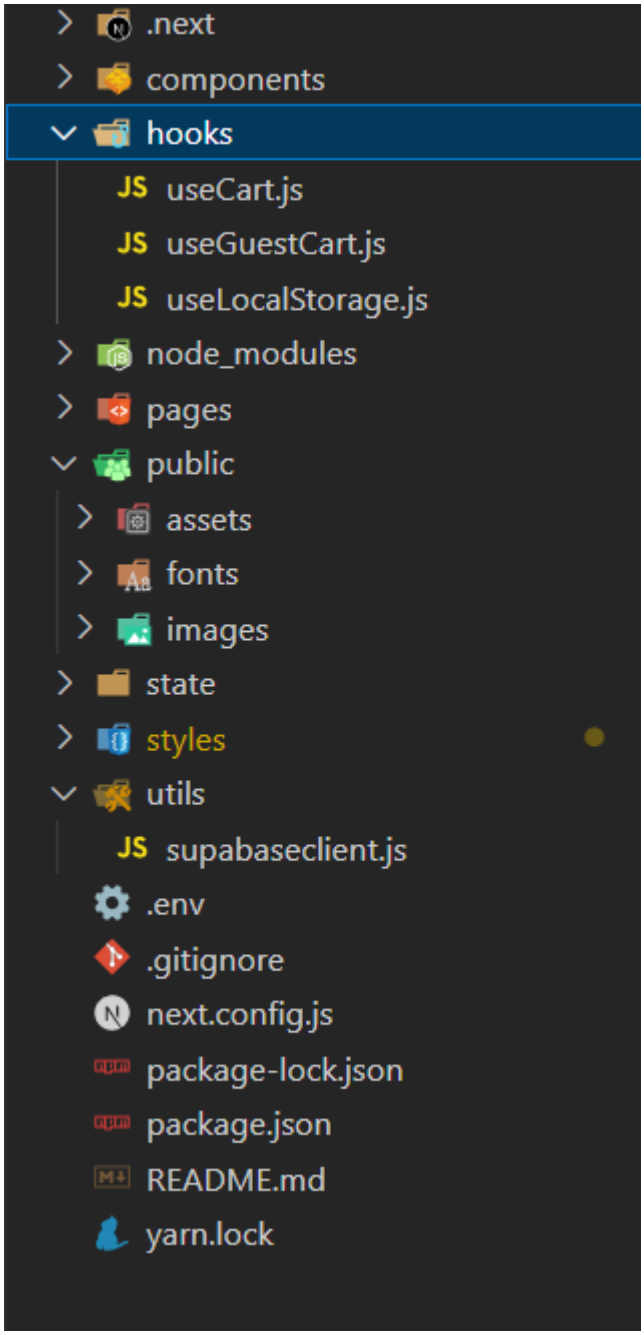


Рисунок 2.10 - Фізична архітектура системи

2.3. Вибір програмного середовища та інструментів реалізації для розпізнання акне - захворювань шкіри

1. Roboflow

У цьому розділі ми докладно розібралися з тим, як буде влаштований наш додаток.

Ми провели **функціональний аналіз**, щоб чітко зрозуміти, що саме має робити програма. Створили **"дерево функцій"**, яке візуально показує всі її можливості.

Далі ми продумали **загальну технічну "схему" системи**, визначили її основні "блоки" (підсистеми та модулі) та як вони будуть взаємодіяти між собою. Ми навіть намалювали цю архітектуру за допомогою стандарту IDEF0, а також побудували **бізнес-моделі** за допомогою EPC, щоб краще зрозуміти логіку роботи програми.

Щодо технічної реалізації, ми:

- Розробили структуру та основні функції нашого **клієнт-серверного додатку**.
- Для його створення використали мову програмування **Python**, а також бібліотеки **OpenCV** та **Pillow** для роботи з зображеннями.
- Зручний інтерфейс для користувача ми реалізували за допомогою бібліотеки **Streamlit**.
- Серверну частину додатка ми розмістили в "хмарі" – на платформі **Amazon Web Services**, що забезпечує її доступність та масштабованість.

РОЗДІЛ 3
РЕАЛІЗАЦІЇ ПРОГРАМНОЇ ПІДСИСТЕМИ

3.1. Реалізація програмної підсистеми розпізнання захворювань шкіри обличчя

3.1.1. Створення та експериментальні верифікації нейронної мережі для розпізнавання захворювань шкіри обличчя (акне)

Підготовка даних для навчання нейромережі: як ми "годуємо" Roboflow

Оскільки платформа Roboflow працює зі знімками у спеціальному форматі "зображення-мітки" (img-labels), для навчання нашої нейромережі ми використовували підготовлені фотографії з нашого набору даних (детальніше про це в розділі 1.9).

До кожного такого зображення ми додавали **JSON-файл**. Це такий текстовий файл, який містить всю важливу інформацію про об'єкти на фото:

- **Назву** самого зображення.
- **Координати "обмежувальних рамок"** (bounding boxes), які точно показують, де на зображенні розташовані висипання чи інші проблеми.
- Інформацію про **складність виявлення** цієї проблеми на конкретному знімку.

Таким чином, Roboflow отримує не просто картинки, а цілісні "пакети" даних, які дозволяють нейромережі точно зрозуміти, що і де шукати на зображеннях.

```
{
  "predictions": [
    {
      "x": 301.5,
      "y": 68,
      "width": 19,
      "height": 18,
      "confidence": 0.675,
      "class": "acne"
    },
    {
      "x": 168.5,
      "y": 364.5,
      "width": 15,
      "height": 19,
      "confidence": 0.653,
      "class": "acne"
    },
```


Рисунок 3.1 – Формат набору даних для foboflow

Платформа **Roboflow** дозволяє нам легко експериментувати з різними нейронними мережами та наборами даних для їхнього навчання. Це дуже зручно, адже ми можемо перемикатися між різними моделями, точно їх налаштовувати та порівнювати, яка з них працює найкраще.

У ході наших експериментів і перевірок ми виявили, що **найкращі результати** (тобто найвищу ефективність) показав метод, що базується на архітектурі **Faster R-CNN Inception-ResNet-v2**. Детальніше їх порівняння можна побачити в Таблиці 3.1.

Roboflow 2.0 Object Detection: наш інструмент для "навчання" системи

Roboflow 2.0 Object Detection — це оновлена, поліпшена онлайн-платформа, яка значно спрощує роботу з розпізнаванням об'єктів на зображеннях. Вона дозволяє нам швидко і ефективно підготувати дані, навчити нейронну мережу, а потім і використовувати її для розпізнавання різних об'єктів (у нашому випадку — ознак захворювань шкіри).

Таблиця 3.1. Порівняння різних типів нейронних мереж у Roboflow

Тип нейромережі	Витрати пам'яті (GB)	Загальний час тренування(год)	Точність
YOLOv5	0.250	2	0.75
EfficientDet	1	12	0.86
EfficientDet	0.100	1	0.72
Mask R-CNN	2	15	0.82
Roboflow 2.0 object detection	1	5	0.88

Процес роботи з цією платформою поділяється на кілька зрозумілих кроків:

1. **Завантаження даних:** Спочатку ми завантажуємо наші фотографії (у різних форматах, наприклад, JPEG чи PNG) на платформу.
2. **Підготовка даних:** Після завантаження Roboflow автоматично виконує "чорнову" роботу: зменшує розмір зображень, переводить їх у потрібний формат для навчання моделі і навіть створює додаткові зображення, трохи змінюючи існуючі (це називається **аугментацією**). Це допомагає моделі краще вчитися.
3. **Вибір "мозку" мережі:** Далі ми обираємо, яку архітектуру нейронної мережі використовуватимемо. Roboflow пропонує кілька готових, перевірених варіантів, з яких можна вибрати найбільш підходящий.
4. **Навчання моделі:** Після вибору архітектури починається сам процес навчання. Тут використовуються стандартні методи машинного навчання, щоб модель поступово "навчилася" розпізнавати потрібні об'єкти.
5. **Оцінка результатів:** Коли навчання завершено, ми перевіряємо, наскільки добре модель справляється зі своїм завданням на нових, "незнайомих" їй зображеннях. Для цього ми використовуємо різні показники, такі як точність, чутливість тощо.

6. **Налаштування параметрів:** Щоб зробити модель ще ефективнішою, ми можемо "підкручувати" її налаштування (гіперпараметри), наприклад, швидкість навчання або кількість "епох" (повних проходів даних через мережу).

Наші експериментальні налаштування навчання:

Ми підбирали ці параметри експериментальним шляхом, перевіряючи, що працює найкраще.

- **Кількість епох навчання: 384.** Це означає, що весь набір тренувальних даних (у нашому випадку, 284 зображення) був "пропущений" через нейронну мережу 384 рази. За кожну епоху відбувається багато дрібних кроків (ітерацій), кількість яких залежить від обсягу даних та розміру "батча" (пакету зображень, що обробляються одночасно).

Що саме робить розроблений вами додаток (по пунктах, як запитували):

Розроблений веб-додаток виконує наступні дії:

1. **Приймає фотографію обличчя** від користувача (бажано якісну, без фільтрів та засвітів).
2. **Аналізує завантажене зображення** за допомогою навченої нейронної мережі для **виявлення ознак акне** або інших шкірних висипань.
3. **Визначає (розпізнає) тип та ступінь** висипань на основі аналізу зображення.
4. **Формує персоналізовані рекомендації** щодо догляду за шкірою, виходячи з виявленої проблеми.
5. **Пропонує список рекомендованих товарів та послуг**, які можуть допомогти у вирішенні шкірних проблем.
6. Надає **загальні поради щодо здорового способу життя** (наприклад, норму води, кількість кроків), які впливають на стан шкіри.
7. Забезпечує **зручний користувацький інтерфейс** для взаємодії з сервісом, що дозволяє легко завантажувати фото, переглядати результати та рекомендації.
8. **Дозволяє користувачам брати участь у процесі покращення** моделі розпізнавання, надаючи нові дані для навчання та підвищення її точності.

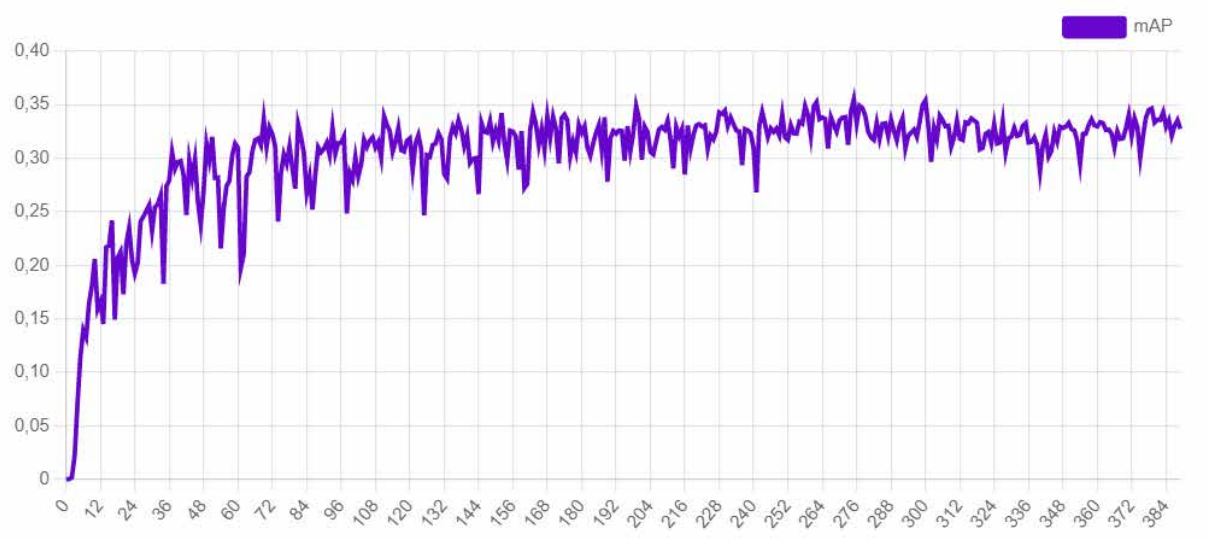


Рисунок 3.2 – Графіки зміни точності створеної моделі

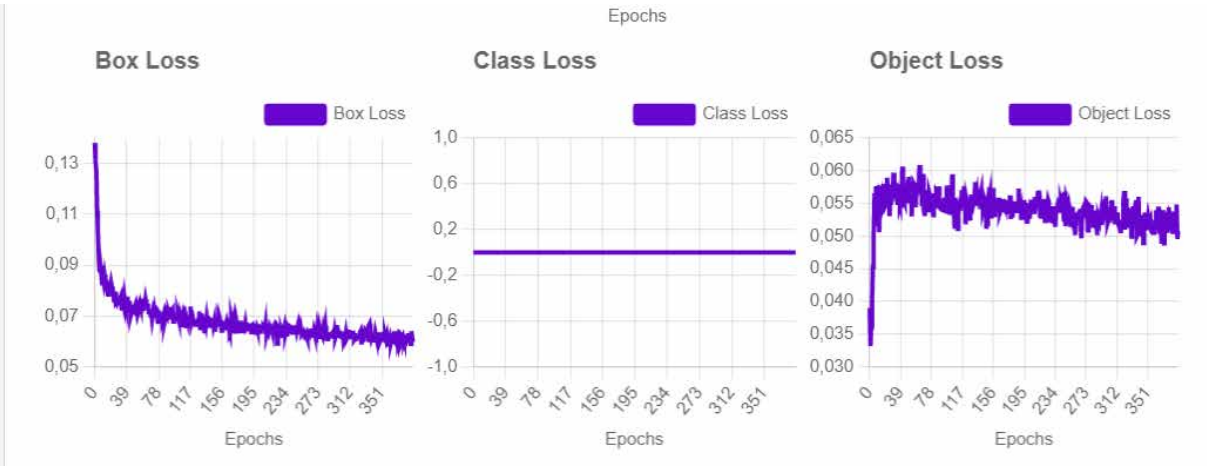


Рисунок 3.3 – Графіки зміни помилки створеної моделі

1.Box loss (втрата рамки) - це функція втрат, яка використовується в моделі об'єктного виявлення для підсумовування помилок в прогнозах рамок, які були зроблені моделлю під час навчання.

2.Box loss (втрата рамки) - це функція втрат, яка використовується в моделі об'єктного виявлення для підсумовування помилок в прогнозах рамок, які були зроблені моделлю під час навчання.

3. Object loss (втрата об'єкта) - це функція втрат, яка використовується в моделі об'єктного виявлення для підсумовування помилок в прогнозах наявності об'єктів на зображенні, які були зроблені моделлю під час навчання.

З графіків можна зрозуміти, що параметри нейронної мережі були належним чином налаштовані, що дозволило досягти точності моделі на рівні 80%. Новостворена модель буде протестована на перевірочному наборі даних (див. Рис. 3.4).

Рис 3.4 – Тестування створеної нейромережі

3.1.2. Реалізація основного модулю у Acne_detection

Для початку трема встановити та імортувати всі допоміжні бібліотеки :

Для цього скорестаємоця npm install

```
import Head from "next/head";
import Image from "next/image";
import { Inter } from "next/font/google";
import styles from "../styles/Home.module.scss";
import { useState, useRef, useEffect } from "react";
import { supabase } from "@/utils/supabaseclient";
import { useRouter } from "next/router";
import axios from "axios";
import ImageWithLabels from "./ImageWithLabels";
import Predictionmap from "./Predictionmap";
import img from "../public/logo.jpg";
```

Рисунок 3.5 – Список імпортованих бібліотек

useState – хук в react , який дозвояє працювати ці state

useRef – хук в react, який дозволяє робити ре-рендер тільки тих компонентів , які ти вкажеш

supabase – компонент , який потріб для роботи з БД

useRouter – бібліотека в nextJS , яка дозволяє створівати зручну та оптимізовану навігацію

axios – бібліотека для Js , яка дозволяє зручно працювати з асинхроними подіями

При натискані на кнопку submit та виборі користувачем зображення , яке він хоче обробити . Це зображення записується в БД supabase, для того, щоб далі передати його у вигляді url в нашу навчану модель нейромережі

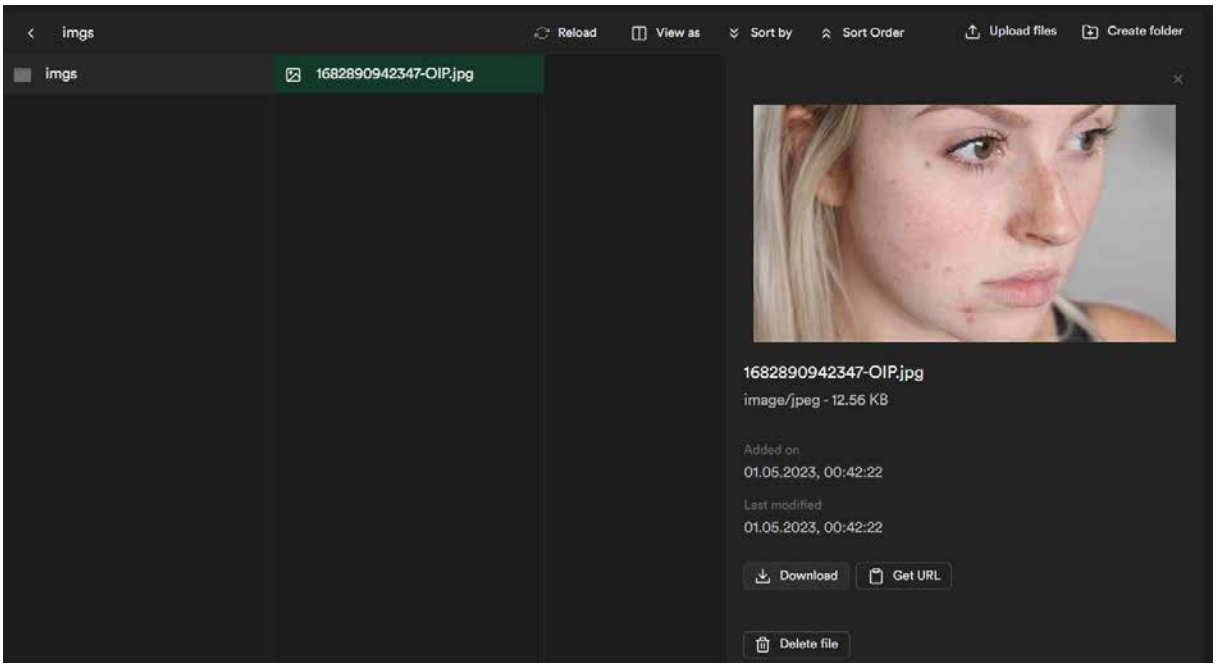


Рисунок 3.6 – Запис зображення в БД

Після чого ми повинні отрмати наше зображення у вигляді url в кієнтській частині

```
import { createBrowserSupabaseClient } from "@supabase/auth-helpers-nextjs";

export const supabase = createBrowserSupabaseClient();
```

```
  const uploadImage = async () => {
    const imageName = `${Date.now()}-${imageFile.name}`;

    const { data: list, error: err } = await supabase.storage
      .from("acne")
      .list("imgs");
```

```
    const filesToRemove = list.map((x) => `imgs/${x.name}`);
    const { data: data_acne, error: erroe_acne } = await supabase.storage
      .from("acne")
      .remove(filesToRemove);
    const { error } = await supabase.storage
      .from("acne")
      .upload(`imgs/${imageName}`, imageFile, {
        cacheControl: "3600",
        upsert: false,
      });

    if (error) return alert(error.message);


    const { data } = await supabase.storage
      .from("acne")
      .getPublicUrl(`imgs/${imageName}`);


    return new URL(data.publicUrl).href;
  };
export async function getServerSideProps({ params }) {
   const temp = [];
  const { data: list, error: err } = await supabase.storage
    .from("acne")
    .list("imgs");
  list.map((image) => {
    const { data } = supabase.storage
      .from("acne")
      .getPublicUrl(`imgs/${image.name}`);
    temp.push(data.publicUrl);
  });
  if (err) return alert("no products");
  const temp_url = temp[0];
  return {
    props: { acne: temp },
  };
}
```

Після того як ми отримали наше зображення у вишляді url нам треба з клієнської частини зробити запит на сервер де в нас знаходиться наша нейромережа.

```
 axios({
        method: "POST",
        url: "https://detect.roboflow.com/acne_detection/1",

        params: {
          api_key: "*******",
```

```
          image: publicUrl,
          confidence: 30,
          format: "JPEG",
        },
      })
```

Ми робимо POST запит в якому вказуємо в url- шлях до нашої нейромережі. Потім ми вказуємо параметри:

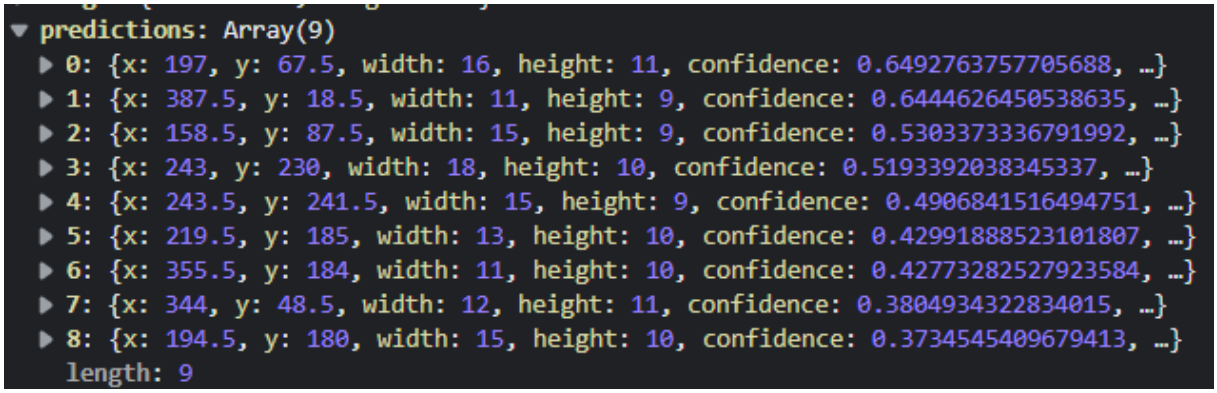
1.Api_key – ключ нашої моделі

2.image – наше зображення в url вигляді

3.confidence – мінімальне значення впевненості для моделі

4.Формат зображення ,яке ми отримаємо назад

Після чого ми отримуємо відповідь серевера у вигляді JSON в якому знаходиться масив об’єків



У такому вигляді ми отримуємо відповідь від сервера – це масив об’єктів кожен з цих об’єктів складається з наступних полів :

1.x – координати мітки по x

2.y – координати мітки по y

3.width – ширина мітки

4.heigth – висота мітки

Після чого нам потрібно відмалювати по цьому масиву об’єктів наші мітки

```
import { useRef, useEffect, useState } from "react";

function drawRect(ctx, x, y, w, h) {
  ctx.beginPath();
  ctx.rect(x, y, w, h);
  ctx.strokeStyle = "#02a16c";
  ctx.lineWidth = 2;
```

```
  ctx.stroke();
}
function drawText(ctx, x, y, text, index) {
  ctx.font = "14px Arial";
  ctx.fillStyle = "#000000";
  index = "N" + index + ": " + text;
  ctx.fillText(index, x, y);
}

export default function ImageWithLabels({
  src,
  imageWidth,
  imageHeight,
  labels_arr,

}) {
  const canvasRef = useRef(null);
  const [widthLabel, setWidth] = useState("");

  useEffect(() => {
    const canvas = canvasRef.current;
    const ctx = canvas.getContext("2d");

    const image = new Image();
    image.onload = () => {
      canvas.width = imageWidth;
      canvas.height = imageHeight;
      const size = imageWidth;

      console.log(size / 100);
      ctx.drawImage(image, 0, 0);

      // draw labels
      let index = 1;
      for (const item_arr of labels_arr) {
        let labelX = item_arr.x - item_arr.width / 2;
        let labelY = item_arr.y - item_arr.height / 2;
        let labelWidth = item_arr.width;
        let labelHeight = item_arr.height;
        setWidth(item_arr.width);

        drawRect(ctx, labelX, labelY, labelWidth, labelHeight);
        drawText(
          ctx,
          labelX,
          labelY - 5,
          item_arr.confidence.toFixed(2),
          index
        );
        index++;
      }
    };
```

```
    image.src = src;
  }, [src, imageWidth, imageHeight, widthLabel]);

  return <canvas ref={canvasRef} />;
}
```

Я створив окремий компонент ImageWithLabels в який передав наступні параметри :

1.imageWidth – ширина картинки

2.imageHeight – висота картинки

3.imageSize – загальний розмір картинки

4.labelX – розташування мітки по x

5.labelY – розташуання мітки по y

6.labelWidth – ширина мітки

7.labelheight – вистота мітки

8.labelConfidence – впевнееість розпізнання

За цими параметрами відмальовую labels .

Тепер перейдемо до створення нашого інтерфейса , з усими данними ,які ми вже отримали

```
    <>
      <Head>
        <title>Create Next App</title>
        <meta name="description" content="Generated by create next app" />
        <meta name="viewport" content="width=device-width, initial-scale=1" />
        <link rel="icon" href="/favicon.ico" />
      </Head>
      <header className={styles.header}>
        <Image src={img}></Image>
        <div>
          {" "}
          <h1>Дипломна робота студента групи КН-42</h1>
          <h1>Комарніцького Іллі</h1>
        </div>
      </header>
      <div className={styles.main}>
        <div className={styles.wrapper}>
          {data && (
            <ImageWithLabels
              imageWidth={data.data.image.width}
              imageHeight={data.data.image.height}
              labels_arr={data.data.predictions}
```

```
            src={imageUrl}
          />
        )}
        {data && <Predictionmap predictions={data.data.predictions} />}
      </div>

      <form
        onSubmit={(e) => onSubmit(e)}
        className={styles.addCategory__form}
      >
        <label htmlFor="image">Зображення категорії</label>
        <input
          accept="image/*"
          type="file"
          name="image"
          id="image"
          ref={imageFileRef}
          onChange={handleImageChange}
        />

        <button type="submit"> submit</button>
      </form>
    </div>
  </>
```

3.2. Інструктивний матеріал з експлуатації виявлення захворювань шкіри

3.2.1. Інструктивний матеріал з експлуатації виявлення захворювань шкіри для користувача

Завдяки створенні web-додатка було максимально спрощено взаємодую додатка та користувачів (косметолог, дерматолог), які повинні просто зайти на веб-сайт (рис. 3.7).

Для того, щоб здійснити виявлення акне необхідно натиснути на кнопку «Вибір файлу». Після чого вибрати зображення та натиснути на кнопку «submit».

Після того як ви обрали зображення ви можете побачити результат роботи програмного модулю.

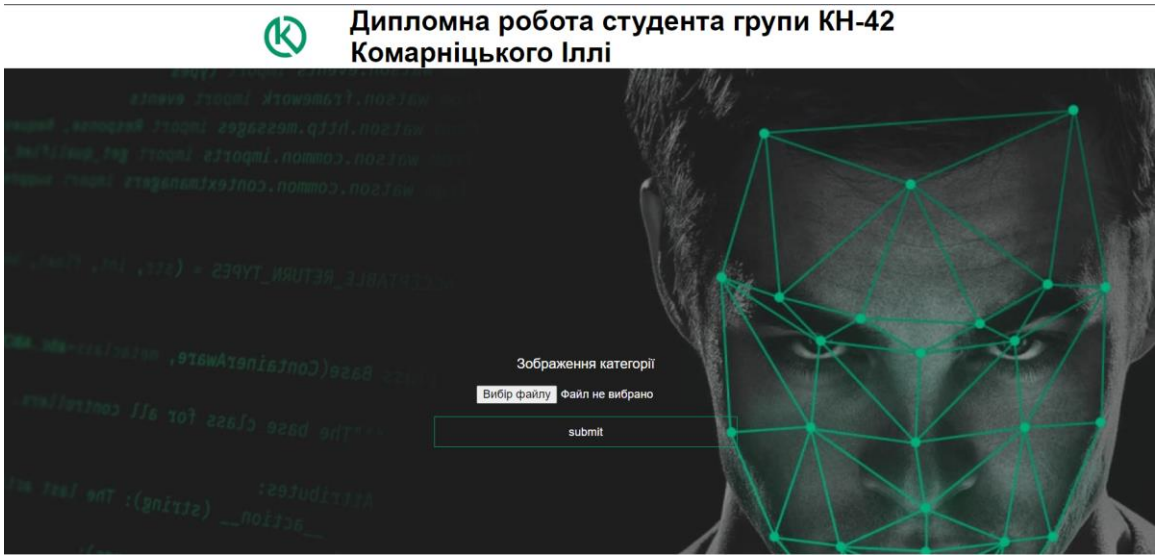


Рисунок 3.7 – Вигляд web-додатка

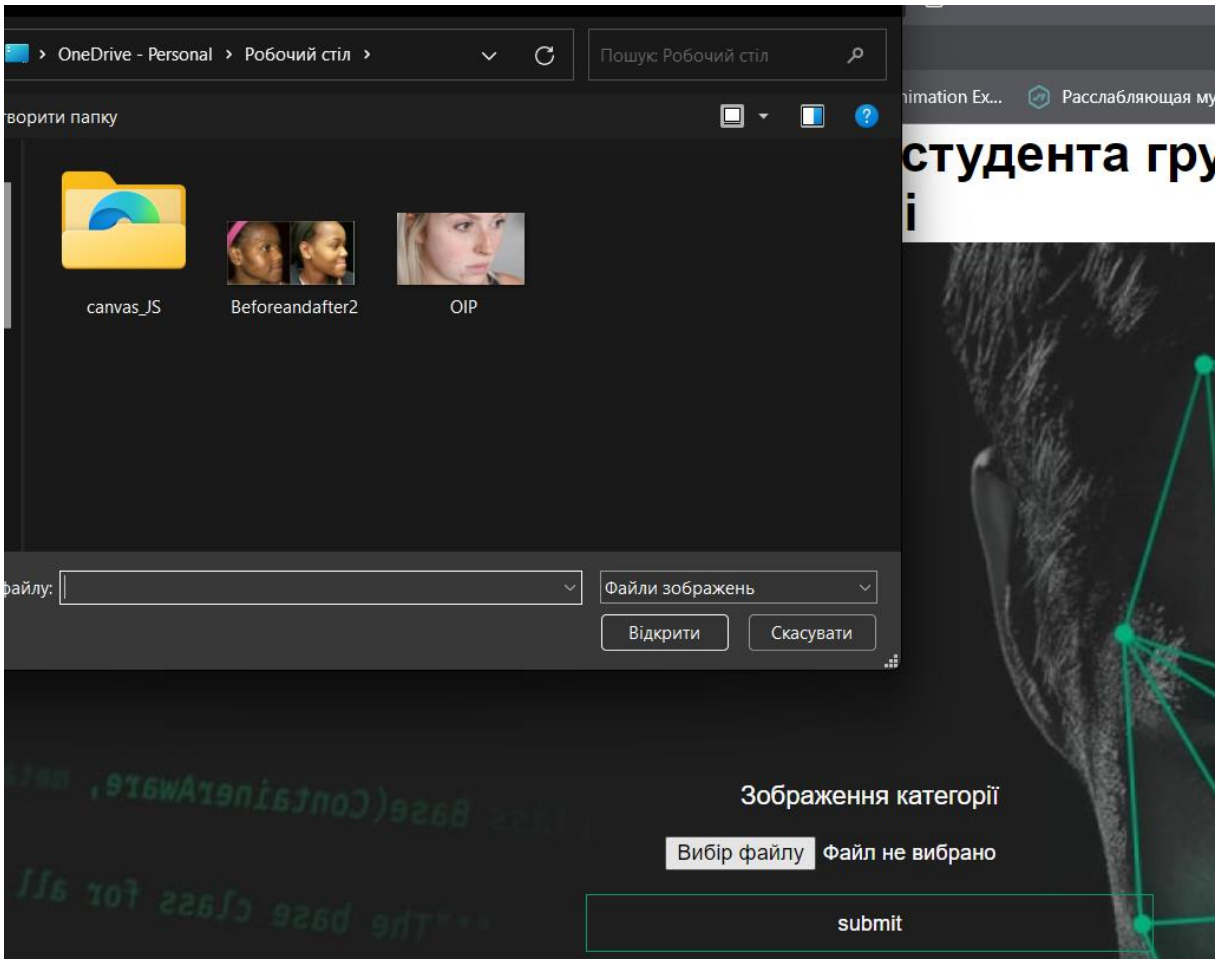


Рисунок 3.8 – Вибір зображення

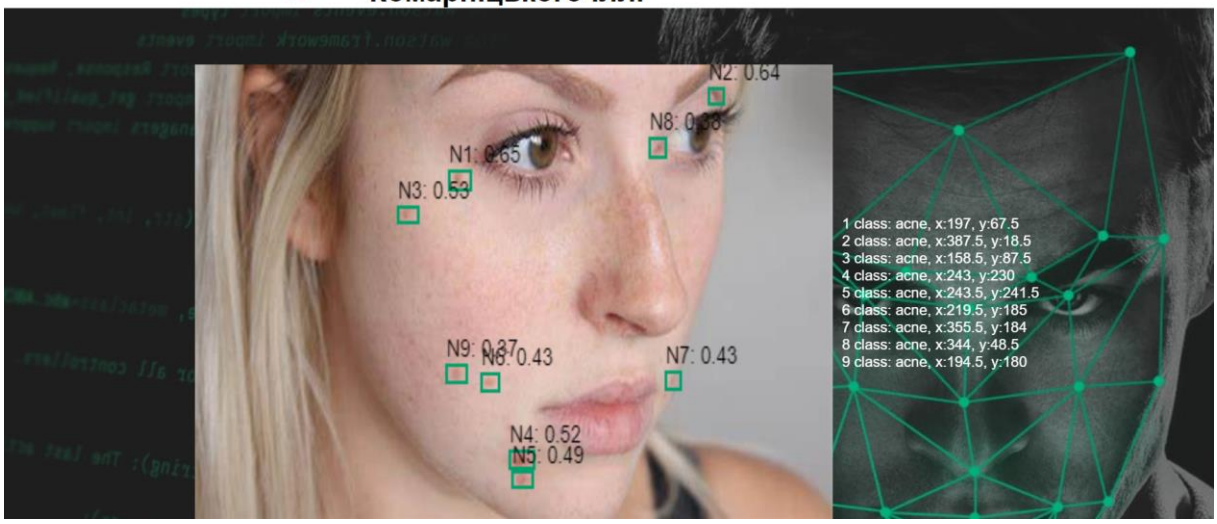


Рисунок 3.9 – Виконання прогнозу

3.3. Тестування та аналіз модулю виявлення захворювань шкіри

Розглянемо можливі випадки, які можуть трапитися під час виявлення акне. Було вирішено виділити наступні випадки:

1.Якщо при виявленні захворювань шкіри на зобрженні буде здорова людина та хвора

Рисунок 3.10 – Виконання прогнозу на декількох людях

Результат: як можна побачити програма відпрацювала коректно та відмітила мітки тільки на хворі людині

2.Перевірити роти програми на людей з різної етнічної раси

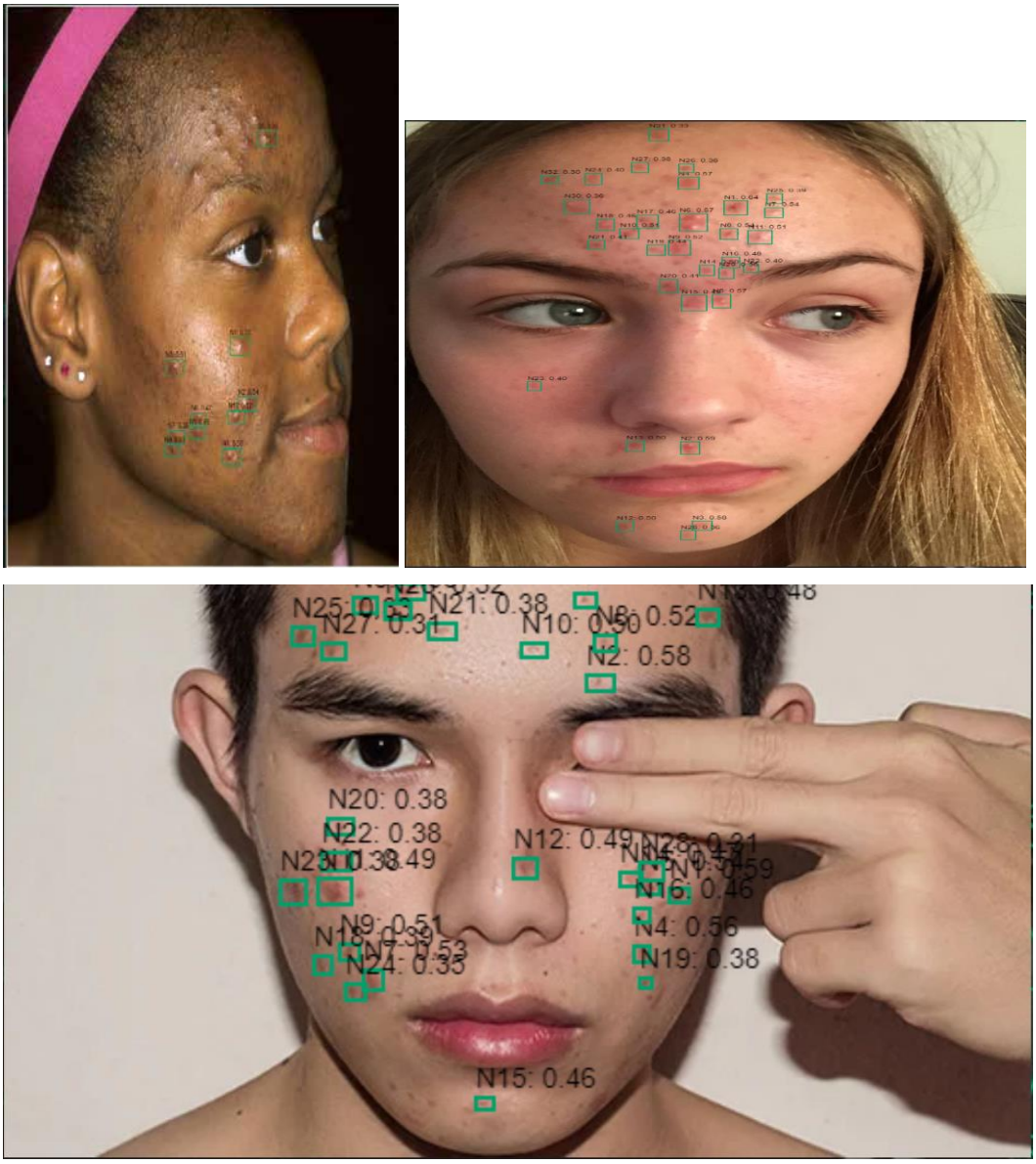


Рисунок 3.11 – Виконання прогнозу на людях різної раси

Результат: як можна побачити програмний код відпрацював вірно на людях різної раси.

3.Перевірка роботи на власних зображення зроблених на телефон

Результат:Як можна побачити модель вірно спрацьовує , якщо фото без фільтрів та не спрацьовує на пост акне

4.Превірка роботи програму на зображенні в темноті

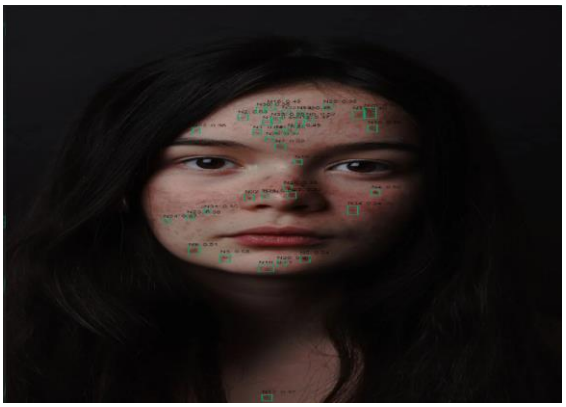


Рисунок 3.13 – Виконання прогнозу в ночі

Навіть на тестових знімках, зроблених у темний час доби, наша система справляється з розпізнаванням правильно.

Висновок до третього розділу

У цій частині роботи я детально розповів про те, **як влаштований мій веб-додаток** та **які моделі я використовував для розпізнавання проблем зі шкірою**.

- Я розібрав "по гвинтиках" кожну складову свого проекту, пояснивши, як працюють його різні частини: та, що "спілкується" з користувачем (клієнтська), та, що відповідає за логіку та обробку даних (серверна).
- Також я перерахував всі бібліотеки та компоненти, які були задіяні в проекті, щоб показати, на чому базується його робота.
- Крім того, я створив **докладну інструкцію**, яка допоможе будь-якому користувачеві легко розібратися, як працювати з веб-додатком. Я пояснив, як влаштований інтерфейс і як ним користуватися.
- Нарешті, я провів **ретельне тестування** програмного модуля в різних умовах і ситуаціях. Це дозволило мені виявити як сильні сторони моєї моделі, так і її слабкі місця, над якими можна буде попрацювати в майбутньому.

ВИСНОВКИ

Підводячи підсумки дипломної роботи, можна сказати, що ми:

1. Розробили працюючий програмний додаток, який вміє виявляти захворювання шкіри, що є першим кроком до їх подальшого лікування.

2. Провели глибокий аналіз предметної області, вивчивши вже існуючі підходи та рішення у сфері косметології та комп'ютерного зору для розпізнавання об'єктів на фото та відео.

3. Детально продумали структуру системи:

- Зробили функціональний аналіз і чітко визначили, що саме має робити програма.
- Створили архітектуру всього застосунку, продумали його бізнес-логіку, розбили на окремі підсистеми та модулі, а також налагодили зв'язки між усіма компонентами.

4. Втілили додаток у життя:

- Розробка відбувалася з використанням мови програмування JavaScript та бібліотеки Next.js.
- Ми використали нейронні мережі на основі моделей Roboflow та YOLOv5 для розпізнавання. *Важливо: це не ми створили ці моделі, а використали їх для нашого завдання.* Якщо ви створювали власну архітектуру, то саме її потрібно тут згадати.
- Для зручності користувачів та інших розробників ми підготували детальну документацію.

5. Що вміє розроблений додаток?

- Виявляти захворювання шкіри за фотографіями.
- Дозволяти користувачам брати участь у покращенні класифікаційної моделі (навчати її), щоб підвищувати точність виявлення акне.
- Надавати рекомендації щодо догляду за шкірою. *Додано на основі попередніх текстів.*

- Допомагати підбирати необхідні засоби та послуги. *Додано на основі попередніх текстів.*

6. Перспективи та результати:

- З технічної точки зору, програму ще можна вдосконалювати, оптимізуючи код і розширюючи набір даних для навчання нейромережі, що дозволить ще більше підвищити її точність.
- Досягнута точність навченої нейронної мережі становить 80%. Це дуже гарний результат, особливо якщо врахувати, що початковий набір даних для тренування був не ідеальним.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Daniel Nelson. How does Image Classification Work? – [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://www.unite.ai/how-does-image-classification-work/
2. What is image classification? Basics you need to know. – [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://www.superannotate.com/blog/image-classification-basics
3. Digital Image Processing: Edge Detection – [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://medium.com/ntust-aivc/digital-image-processing-edge-detection-29aa84a8fd60
4. Guide on Support Vector Machine (SVM) Algorithm – [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/
5. What Is a Support Vector Machine? – [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://www.mathworks.com/discovery/support-vector-machine.html
6. Convolutional Neural Networks: Understand the Basics. – [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-understand-the-basics/
7. Advantages and disadvantages of Convolutional Neural Network. – [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://aspiringyouths.com/advantages-disadvantages/convolutional-neural-network-cnn/
8. Introduction to Convolutional Neural Networks. – [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-

cnn/?utm_source=reading_list&utm_medium=https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-understand-the-basics/

9. Mark Sandler, Andrew Howard. MobileNetV2: Inverted Residuals and Linear Bottlenecks. – [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://arxiv.org/abs/1801.04381 1-6 с.
10. Barret Zoph, Vijay Vasudevan. Learning Transferable Architectures for Scalable Image Recognition. – [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://arxiv.org/abs/1707.07012 3-9 с.
11. Mingxing Tan, Quoc V. Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. – [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://arxiv.org/abs/1905.11946 2-8 с.
12. 17 – Sona college of technology. eye Computer Vision Project. — Електронні дані. – Набір даних. — Режим доступу : https://universe.roboflow.com/sona-college-of-technology-pbo8a/eye-ublew
13. 18 – What is Overfitting? – [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://aws.amazon.com/what-is/overfitting/
14. 19 – Geoffrey E. Hinton, Nitish Srivastava. Improving neural networks by preventing co-adaptation of feature detectors. – [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://arxiv.org/abs/1207.0580
15. MobileNet, MobileNetV2, and MobileNetV3 – Електрон. текстові дані. – Режим доступу : https://keras.io/api/applications/mobilenet/
16. EfficientNet B0 to B7 – Електрон. текстові дані. – Режим доступу : https://keras.io/api/applications/efficientnet/
17. Kiktev, N., Lendiel, T., & Korol, O. (2023). A Mobile Facial Recognition System Based on a Set of Raspberry Technical Tools. Selected Papers of the XX International Scientific Conference “Dynamical System Modeling and Stability Investigation" (DSMSI-2023). Conference Proceedings. Vol. 2: Computer Applied Mathematics. CEUR Workshop Proceedings, 2023, 3746, pp. 23–32. https://ceur-ws.org/Vol-3746/Paper_3.pdf

```
import Head from "next/head";
import Image from "next/image";
import { Inter } from "next/font/google";
import styles from "../styles/Home.module.scss";
import { useState, useRef, useEffect } from "react";
import { supabase } from "@/utils/supabaseclient";
import { useRouter } from "next/router";
import axios from "axios";
import ImageWithLabels from "./ImageWithLabels";
import Predictionmap from "./Predictionmap";
import img from "../public/logo.jpg";

const inter = Inter({ subsets: ["latin"] });

export default function Home({ acne }) {
  const [imageFile, setImageFile] = useState(null);
  const imageFileRef = useRef(null);
  const router = useRouter();
  const [data, setData] = useState("");
  const [imageUrl, setImageUrl] = useState("");
  // console.log(acne);
  const uploadImage = async () => {
    const imageName = `${Date.now()}-${imageFile.name}`;

    const { data: list, error: err } = await supabase.storage
      .from("acne")
      .list("imgs");
    const filesToRemove = list.map((x) => `imgs/${x.name}`);
    const { data: data_acne, error: erroe_acne } = await supabase.storage
      .from("acne")
      .remove(filesToRemove);
    const { error } = await supabase.storage
      .from("acne")
      .upload(`imgs/${imageName}`, imageFile, {
        cacheControl: "3600",
        upsert: false,
      });

    if (error) return alert(error.message);

    // const { data } = await supabase.storage
    //   .from("acne")
    //   .getPublicUrl(`imgs/${imageName}`);

    const { data } = await supabase.storage
      .from("acne")
      .getPublicUrl(`imgs/${imageName}`);

    // console.log(new URL(data.publicUrl).href);
    return new URL(data.publicUrl).href;
```

```
};

const handleImageChange = (e) => {
  const image = e.target.files[0];

  if (image.size > 2000000) {
    alert("Image size must be less than 2MB");
    return;
  }

  setImageFile(image);
};

const onSubmit = async (e) => {
  e.preventDefault();
  let publicUrl = "";

  if (imageFile) {
    publicUrl = await uploadImage();

    axios({
      method: "POST",
      url: "https://detect.roboflow.com/acne_detection/1",

      params: {
        api_key: "****",
        image: publicUrl,
        confidence: 30,
        format: "JPEG",
      },
    })
      .then(function (response) {
        console.dir(response);
        // const blob = new Blob([response.data], { type: "image/jpeg" });
        // const url = URL.createObjectURL(blob);

        setData(response);
        setImageUrl(publicUrl);
        console.log(data);
      })
      .catch(function (error) {
        console.log(error.message);
      });
  }

  router.replace(router.asPath);
};

return (
  <>
    <Head>
      <title>Create Next App</title>
```

```
        <meta name="description" content="Generated by create next app" />
        <meta name="viewport" content="width=device-width, initial-scale=1" />
        <link rel="icon" href="/favicon.ico" />
      </Head>
      <header className={styles.header}>
        <Image src={img}></Image>
        <div>
          {" "}
          <h1>Дипломна робота студента групи КН-42</h1>
          <h1>Комарніцького Іллі</h1>
        </div>
      </header>
      <div className={styles.main}>
        <div className={styles.wrapper}>
          {data && (
            <ImageWithLabels
              imageWidth={data.data.image.width}
              imageHeight={data.data.image.height}
              labels_arr={data.data.predictions}
              src={imageUrl}
            />
          )}
          {data && <Predictionmap predictions={data.data.predictions} />}
        </div>

        <form
          onSubmit={(e) => onSubmit(e)}
          className={styles.addCategory__form}
        >
          <label htmlFor="image">Зображення категорії</label>
          <input
            accept="image/*"
            type="file"
            name="image"
            id="image"
            ref={imageFileRef}
            onChange={handleImageChange}
          />

          <button type="submit"> submit</button>
        </form>
      </div>
    </>
  );
}

export async function getServerSideProps({ params }) {
  // const supabaseClient = createBrowserSupabaseClient()
  const temp = [];
  const { data: list, error: err } = await supabase.storage
    .from("acne")
    .list("imgs");
```

```
  list.map((image) => {
    const { data } = supabase.storage
      .from("acne")
      .getPublicUrl(`imgs/${image.name}`);
    temp.push(data.publicUrl);
  });
  if (err) return alert("no products");
  const temp_url = temp[0];
  return {
    props: { acne: temp },
  };
  // try {
  //   const response = await axios.post(
  //     "http://localhost:3000/api/acne",
  //     temp_url
  //   );
  //   return {
  //     props: { acne: response },
  //   };
  // } catch (error) {
  //   return {
  //     props: {
  //       error: error,
  //     },
  //   };
  // }
}

import { useRef, useEffect, useState } from "react";

function drawRect(ctx, x, y, w, h) {
  ctx.beginPath();
  ctx.rect(x, y, w, h);
  ctx.strokeStyle = "#02a16c";
  ctx.lineWidth = 2;
  ctx.stroke();
}
function drawText(ctx, x, y, text, index) {
  ctx.font = "14px Arial";
  ctx.fillStyle = "#000000";
  index = "N" + index + ": " + text;
  ctx.fillText(index, x, y);
}

export default function ImageWithLabels({
  src,
  imageWidth,
  imageHeight,
  labels_arr,

  //   labelX,
  //   labelY,
```

```
  //   labelWidth,
  //   labelHeight,
}) {
  const canvasRef = useRef(null);
  const [widthLabel, setWidth] = useState("");

  useEffect(() => {
    const canvas = canvasRef.current;
    const ctx = canvas.getContext("2d");

    const image = new Image();
    image.onload = () => {
      canvas.width = imageWidth;
      canvas.height = imageHeight;
      const size = imageWidth;

      console.log(size / 100);
      ctx.drawImage(image, 0, 0);

      // draw labels
      let index = 1;
      for (const item_arr of labels_arr) {
        let labelX = item_arr.x - item_arr.width / 2;
        let labelY = item_arr.y - item_arr.height / 2;
        let labelWidth = item_arr.width;
        let labelHeight = item_arr.height;
        setWidth(item_arr.width);

        drawRect(ctx, labelX, labelY, labelWidth, labelHeight);
        drawText(
          ctx,
          labelX,
          labelY - 5,
          item_arr.confidence.toFixed(2),
          index
        );
        index++;
      }
    };
    image.src = src;
  }, [src, imageWidth, imageHeight, widthLabel]);

  return <canvas ref={canvasRef} />;
}
```