МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління

БОЖИГОРА Юрій Володимирович

Програмний модуль класифікації шкірних захворювань з використанням штучних нейронних мереж / Software module for classification of skin diseases using artificial neural networks

спеціальність 122 – Комп'ютерні науки
освітньо-професійна програма – Комп'ютерні науки

Кваліфікаційна робота

Виконав студент групи КН-41
Ю. В. Божигора

Науковий керівник:
к.т.н., доцент І. В. Турченко

Кваліфікаційну роботу
допущено до захисту:
«___»_____ 20___ р.

Завідувач кафедри
_____ М. П. Комар

ТЕРНОПІЛЬ – 2024

Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «бакалавр»
спеціальність 122 – Комп'ютерні науки
освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ М.П. Комар
«____» _____2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
БОЖИГОРІ Юрію Володимировичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: Програмний модуль класифікації шкірних захворювань з використанням штучних нейронних мереж / Software module for classification of skin diseases using artificial neural networks
керівник роботи Турченко Ірина Василівна, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від 12 грудня 2023 р. № 753.
2. Строк подання студентом закінченої кваліфікаційної роботи 15 травня 2024 р.
3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.
4. Основні питання, які потрібно розробити:

- провести опис предметної області;
- проаналізувати існуючі рішення для класифікації шкірних захворювань з використанням штучних нейронних мереж;
- зробити постановку задачі дослідження;
- представити загальну структуру і розробити алгоритм програмного модуля класифікації шкірних захворювань;
- розробити архітектуру згорткової нейронної мережі для класифікації захворювань шкіри за вхідним зображенням веб-камери комп'ютера;
- розробити та протестувати програмний модуль для класифікації захворювань шкіри.

5. Перелік графічного матеріалу в роботі:

- схема алгоритму модуля попередньої обробки зображення;
- схема алгоритму програмного модуля класифікації шкірних захворювань;
- схема алгоритму навчання згорткової нейронної мережі для програмного модуля класифікації шкірних захворювань.

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 12 грудня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Затвердження теми кваліфікаційної роботи, ознайомлення з літературними джерелами та складання плану роботи	до 01.01. 2024 р.	
2	Написання 1 розділу кваліфікаційної роботи	до 01.03. 2024 р.	
3	Написання 2 розділу кваліфікаційної роботи	до 01.04.2024 р.	
4	Написання 3 розділу кваліфікаційної роботи	до 01.05. 2024 р.	
5	Представлення попереднього варіанту кваліфікаційної роботи, перевірка та внесення змін керівником	до 15.05.2024 р.	
6	Опрацювання зауважень та представлення завершеного варіанту кваліфікаційної роботи. Підготовка супроводжуючих документів.	до 20.05.2024 р.	
7	Перевірка кваліфікаційної роботи на оригінальність тексту у системі «Unicheck».	до 10.06.2024 р.	
8	Оформлення кваліфікаційної роботи та отримання допуску до захисту	до 14.06.2024 р.	
9	Подання кваліфікаційної роботи до захисту на засіданні атестаційної комісії.	до 14.06. 2024 р.	

Студент_____ Ю.В. Божигора
(підпис) (прізвище та ініціали)

Керівник кваліфікаційної роботи_____ І.В. Турченко
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота на тему «Програмний модуль класифікації шкірних захворювань з використанням штучних нейронних мереж» на здобуття освітнього ступеня «бакалавр» зі спеціальності 122 «Комп’ютерні науки» освітньої програми «Комп’ютерні науки» написана обсягом в 59 сторінок і містить 21 ілюстрацію, 3 таблиці, 4 додатки та 29 використаних джерел.

Метою кваліфікаційної роботи є розробка програмного модуля для класифікації шкірних захворювань з використанням штучних нейронних мереж.

У роботі використовувалися методи аналізу і синтезу, порівняльного аналізу, методи глибокого навчання, обробки зображень та підготовки даних, логічного узагальнення результатів.

У результаті виконання кваліфікаційної роботи було розроблено програмний модуль, який можна використовувати в телемедицині, наприклад його можна інтегрувати в телемедичні платформи для надання дистанційної діагностики та рекомендацій щодо лікування; у первинній медичній допомозі, лікарі загальної практики можуть використовувати цей модуль, щоб допомогти визначити типові захворювання шкіри під час звичайних відвідувань пацієнтів; в освіті, студенти-медики можуть навчатися на цих моделях, щоб вивчити різні шкірні захворювання та їх візуальні прояви.

Ключові слова: ШКІРНІ ЗАХВОРЮВАННЯ, КЛАСИФІКАЦІЯ, ШТУЧНІ НЕЙРОННІ МЕРЕЖІ, КОМП'ЮТЕРНИЙ ЗІР, ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ, PYTHON, ПРОГРАМНИЙ МОДУЛЬ

ANNOTATION

Qualification work on the topic «Software module for classification of skin diseases using artificial neural networks» for obtaining a bachelor's degree in the specialty 122 «Computer Science» of the educational program «Computer Science» is written on 59 pages and it contains 21 figures, 3 tables, 4 annexes and 29 references.

The goal of the qualification work is to develop a software module for the classification of skin diseases using artificial neural networks.

The work used methods of analysis and synthesis, comparative analysis, methods of deep learning, image processing and data preparation, logical generalization of results.

As a result of qualification work, a software module was created that can be used in telemedicine, for example, it can be integrated into telemedicine platforms to provide remote diagnosis and treatment recommendations; in primary care, general practitioners and doctors can use this module to help identify common skin conditions during routine patient visits; in education, medical students can study on these models to learn various skin diseases and their visual manifestations.

Keywords: SKIN DISEASE, CLASSIFICATION, ARTIFICIAL NEURAL NETWORKS, COMPUTER VISION, CONVOLUTIONAL NEURAL NETWORKS, PYTHON, SOFTWARE MODULE

ЗМІСТ

ВСТУП

Дерматологічні захворювання можуть бути складними та важкими для діагностики навіть для досвідчених медичних працівників. Традиційні методи діагностики захворювань шкіри часто покладаються лише на візуальний огляд, який може зайняти багато часу та бути суб’єктивним. Програмне забезпечення для класифікації захворювань шкіри використовує розширені алгоритми для аналізу зображень шкіри, допомагаючи досягти більш точної та послідовної діагностики. Програмне забезпечення для класифікації може прискорити діагностичний процес, забезпечуючи швидкий аналіз зображень шкіри, дозволяючи лікарям приймати більш обґрунтовані рішення. Отже, тема кваліфікаційної роботи є актуальною.

Об'єктом дослідження є процес класифікації захворювань шкіри.

Предметом дослідження є програмний модуль класифікації шкірних захворювань з використанням штучних нейронних мереж.

Методи дослідження базуються на підході згорткової нейромережевої класифікації та використанні технологій комп’ютерного зору.

Метою кваліфікаційної роботи є розробка програмного модуля для класифікації шкірних захворювань з використанням штучних нейронних мереж.

Для досягнення поставленої мети необхідно виконати наступні завдання:

– проаналізувати предметну область;

– проаналізувати методи розпізнавання дефектів шкіри на зображенні чи у відеопотоці, дослідити переваги та недоліки, що дозволить сформувати вимоги до програмного модуля класифікації захворювань шкіри;

– представити загальну структуру і розробити алгоритм програмного модуля класифікації шкірних захворювань;

– розробити архітектуру згорткової нейронної мережі для класифікації захворювань шкіри за вхідним зображенням веб-камери комп’ютера;

– розробити та протестувати програмний модуль для класифікації захворювань шкіри.

Практичне значення отриманих результатів: програмний модуль можна

використовувати в телемедицині, наприклад його можна інтегрувати в телемедичні платформи для надання дистанційної діагностики та рекомендацій щодо лікування; у первинній медичній допомозі, лікарі загальної практики можуть використовувати цей модуль, щоб допомогти визначити типові захворювання шкіри під час звичайних відвідувань пацієнтів; в освіті, студенти-медики можуть навчатися на цих моделях, щоб зрозуміти різні шкірні захворювання та їх візуальні прояви. Також у регіонах з обмеженим доступом до дерматологів цей модуль може служити цінним ресурсом для попередньої діагностики.

Публікації та апробація кваліфікаційної роботи.

Результати дослідження опубліковано та апробовано в матеріалах VII міжнародної науково-практичної конференції «Сучасні світові тенденції розвитку науки та інформаційних технологій», м. Одеса, 30-31.05.2024 р. (Додаток Г).

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Класифікація шкірних захворювань

Шкірні захворювання є поширеними проблемами зі здоров’ям у всьому світі. Також вони є одними з найбільш заразних захворювань. ВООЗ повідомляє, що у 2022 році було діагностовано понад 14 мільйонів випадків і 9,6 мільйонів смертей у світі. Причинами шкірних захворювань є віруси, бактерії, алергія або грибкові інфекції [1]. Небезпека інфекцій непомітна, що спричиняє погіршення як фізичного здоров'я, так і психологічного (багато людей відчуває психологічний дискомфорт при наявності помітних проблем зі шкірою). У важких випадках це може спричинити рак шкіри. Діагностика шкірних захворювань за клінічними зображеннями є одним із найскладніших завдань аналізу медичних зображень. Крім того, при традиційному огляді медичними експертами діагностика шкірних захворювань вимагає багато часу та є суб’єктивною.

Існують різні види ураження шкіри. Вони відрізняються за симптомами та ступенем тяжкості. Деякі з них постійні, а деякі тимчасові і можуть бути безболісними або болючими. Серед цих захворювань шкіри меланома є найбільш смертоносним і небезпечним видом. Проте приблизно 95% пацієнтів із шкірними захворюваннями можуть одужати, якщо їх виявити на початковому етапі.

Між дерматологами та пацієнтами зі шкірними захворюваннями існує величезний розрив, оскільки багато людей не знають типів, симптомів і стадій шкірних захворювань. Іноді потрібен тривалий час, щоб проявились ознаки. Для цього потрібне раннє та швидке виявлення. Але правильно діагностувати шкірні захворювання, визначити тип і стадію захворювання, може бути важко і дорого. Автоматичний програмний модуль, створений на основі машинного навчання, дозволить точно та швидко класифікувати шкірні захворювання.

Класифікація шкірних захворювань є важливою сферою досліджень і розробок у галузі дерматології та медичних зображень. Вона охоплює застосування обчислювальних методів для автоматичної класифікації та ідентифікації різних

дерматологічних захворювань на основі візуальних характеристик, отриманих із зображень шкіри або даних пацієнтів. Основна мета класифікації шкірних захворювань полягає в тому, щоб допомогти лікарям у точному діагностуванні та лікуванні шкірних розладів, тим самим покращуючи результати для пацієнтів і якість медичної допомоги.

Точна діагностика має першочергове значення в дерматології, оскільки багато захворювань шкіри мають схожі візуальні симптоми, але вимагають абсолютно різних підходів до лікування. Помилковий діагноз або запізніле встановлення діагнозу може привести до неефективного лікування, прогресування захворювання та потенційно серйозних наслідків для здоров'я пацієнтів. Системи класифікації захворювань шкіри спрямовані на вирішення цих проблем, надаючи об'єктивну, послідовну та своєчасну діагностичну підтримку медичним працівникам.

Класифікація шкірних захворювань представляє кілька проблем, які необхідно вирішити для розробки ефективних і надійних систем класифікації.

Захворювання шкіри демонструють широкий спектр візуальних характеристик, включаючи відмінності в кольорі, текстурі, морфології та розподілі уражень. Ця мінливість ускладнює розробку алгоритмів, які можуть точно розрізняти різні ознаки [2].

Отримання наборів даних зображень шкіри, які адекватно представляють різноманіття дерматологічних захворювань, може бути складним завданням. Такі проблеми, як конфіденційність даних, дисбаланс даних і мінливість якості зображення, створюють значні перешкоди для підготовки надійних моделей класифікації. Приклади зображень шкірних захворювань представлені на рисунку 1.1.

Деякі шкірні захворювання мають подібні візуальні характеристики, що призводить до плутанини між класами. Наприклад, відрізнити доброякісні та злоякісні ураження шкіри або розрізнити різні типи висипу може бути складно навіть для досвідчених дерматологів.

Моделі класифікації, навчені на одному наборі даних або одному пацієнті,

можуть погано узагальнюватися на погано видимі дані або різні групи пацієнтів. Досягнення потужних можливостей узагальнення має вирішальне значення для практичного розгортання систем класифікації в реальних умовах.

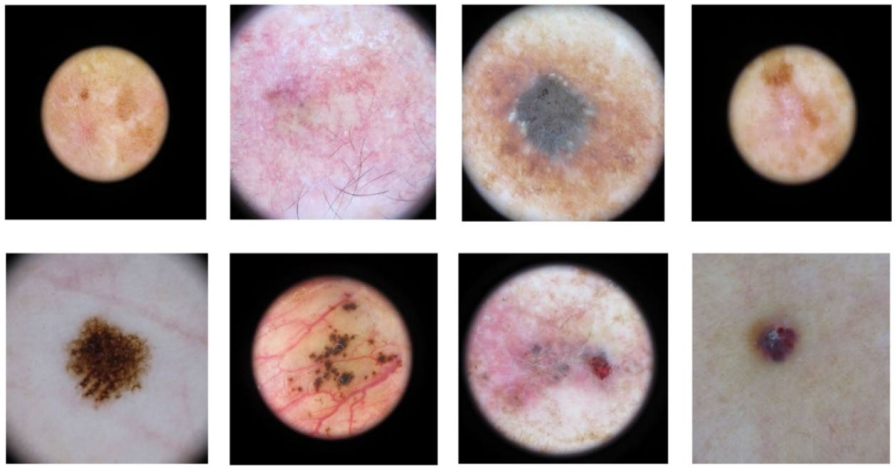

Рисунок 1.1 – Приклади шкірних захворювань

Існують різні підходи до вирішення проблем класифікації захворювань шкіри. Традиційні підходи, засновані на функціях, виділяють релевантні візуальні характеристики із зображень шкіри, такі як кольорові гістограми, дескриптори текстури та особливості форми. Потім ці функції використовуються для навчання класифікаторів машинного навчання, таких як опорні векторні машини або дерева рішень, для класифікації захворювань шкіри.

Методи глибокого навчання, зокрема згорткові нейронні мережі (ЗНМ), зробили революцію в класифікації шкірних захворювань. Вони можуть автоматично вивчати ієрархічні представлення зображень шкіри, фіксуючи як низькорівневі характеристики (наприклад, краї, текстури), так і високорівневі представлення, пов’язані з різними дерматологічними захворюваннями.

Методи ансамблевого навчання поєднують передбачення з кількох базових класифікаторів для покращення загальної продуктивності та надійності. Методи ансамблю можуть допомогти пом’якшити наслідки мінливості даних і плутанини між класами шляхом агрегування різноманітних джерел інформації [3].

У підсумку, класифікація захворювань шкіри відіграє життєво важливу роль у сучасній дерматології, пропонуючи потенціал для підвищення діагностичної

точності, ефективності та доступності лікування. Незважаючи на виклики, прогрес у обчислювальних техніках і міждисциплінарна співпраця продовжують стимулювати прогрес у цій важливій галузі досліджень і розробок.

1.2 Застосування згорткових нейронних мереж та комп'ютерного зору у класифікації шкірних захворювань

Перш ніж розпізнати артефакт на шкірі, необхідно розрізнити особливості зображення. Після цього слід застосовувати будь-який метод класифікації. Для класифікації та розпізнавання потрібна велика кількість ознак (особливостей зображення).

Звичайні моделі розпізнавання образів обмежені у своїй здатності обробляти природні дані в їх вихідній формі. Десятиліттями побудова системи розпізнавання образів або машинного навчання вимагала ретельного проектування та значного досвіду в області розробки інструменту вилучення ознак, який перетворював би необроблені дані (наприклад, значення пікселів зображення) у відповідне внутрішнє представлення або вектор ознак, з яких підсистема навчання, часто класифікатор, може виявляти або класифікувати шаблони вхідних даних [4]. Таким чином, витяг функцій із необроблених даних вимагає величезних зусиль і не є автоматизованим.

Згорткові нейронні мережі (ЗНМ) – клас нейронних мереж глибокого навчання – містять згорткові шари, призначені для вивчення особливостей вхідних даних, а повноз'єднані шари можна використовувати для класифікації. Штучна нейронна мережа поєднує ці два кроки, щоб зменшити вимоги до пам’яті та обчислювальну складність, а також забезпечити кращу продуктивність [5].

ЗНМ, як клас глибоких штучних нейронних мереж прямого поширення, успішно застосовуються для аналізу візуальних зображень. Вони використовують тип багатошарового персептрона, і відносно невелику попередню обробку порівняно з іншими алгоритмами класифікації зображень. Це означає, що мережа вивчає фільтри, які в традиційних алгоритмах були створені вручну.

У літературі розглядається багато варіантів архітектур згорткових нейронних мереж. Однак їх основні компоненти дуже схожі. ЗНМ складається з вхідного та вихідного рівнів, а також кількох прихованих рівнів. Зазвичай вони складаються з трьох типів шарів: згорткового, агрегувального і повноз'єднаного. Приклад архітектури згорткової нейронної мережі [5], яка класифікує зображення томографії мозку, представлено на рисунку 1.2.

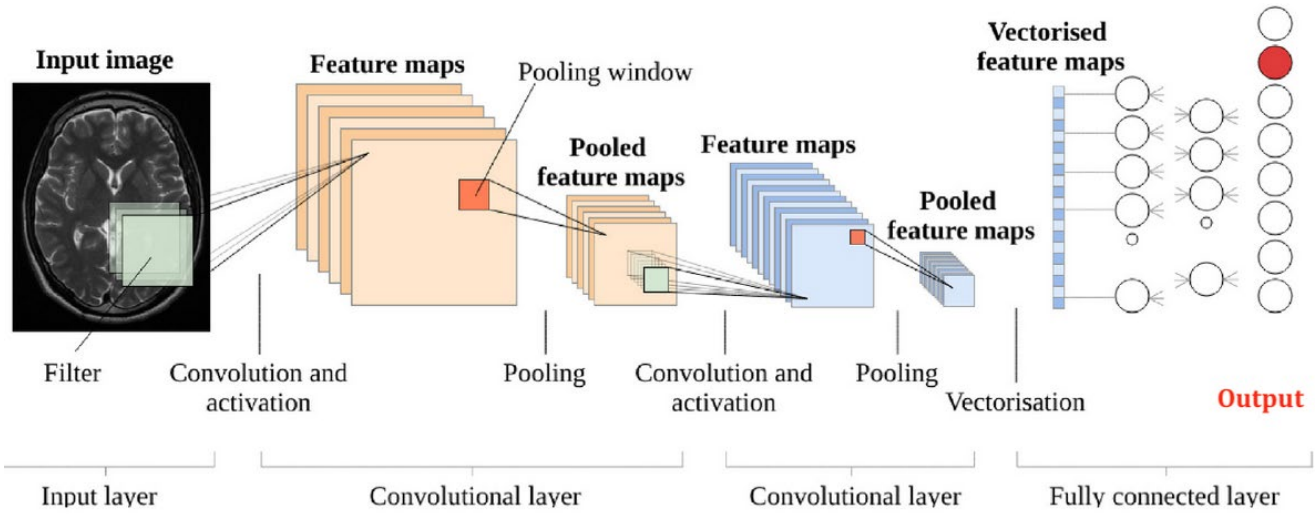

Рисунок 1.2 – Приклад згорткової нейронної мережі для розпізнання медичних зображень [5]

Згортковий шар є основним шаром згорткової нейронної мережі. Він містить фільтр для кожного каналу, ядро згортки якого обробляє попередній шар по фрагментах і використовується для розрахунку різних карт ознак. Вагові коефіцієнти ядра згортки невідомі і задаються в процесі навчання.

Особливістю згорткового шару є порівняно невелика кількість параметрів, які задаються під час навчання. Кожен нейрон карти ознак пов’язаний із областю сусідніх нейронів попереднього шару. Нову карту можна отримати шляхом спершу згортання вхідних даних із вивченим ядром, а потім застосування поелементної нелінійної функції активації до згорнутих результатів. Типовими функціями активації є sigmoid, tanh і ReLU [6].

Згорткові шари застосовують операцію згортки до вхідних даних, передаючи

результат до наступного шару.

Хоча повнозв’язні нейронні мережі прямого зв’язку можна використовувати як для вивчення ознак, так і для класифікації даних, застосування цієї архітектури до зображень є недоцільним. Потрібна буде дуже велика кількість нейронів, навіть у неглибокій (на відміну від глибокої) архітектурі через дуже великі розміри вхідних даних, пов’язаних із зображеннями, де кожен піксель є відповідною змінною. Наприклад, повноз'єднаний шар для маленького зображення 100×100 має 10 000 вагових коефіцієнтів. Операція згортки вирішує цю проблему, оскільки вона зменшує кількість вільних параметрів, дозволяючи мережі бути глибшою з меншою кількістю параметрів.

Шар агрегації – це нелінійне ущільнення карти ознак, за допомогою якого група пікселів (зазвичай розміром 2×2) ущільнюється до одного пікселя шляхом нелінійного перетворення. У результаті перетворення кожен прямокутник або квадрат, що не перетинаються, зменшується на один піксель, і вибирається піксель із максимальним значенням. Об'єднання дозволяє значно зменшити просторовий об'єм зображення.

Фільтрування деталей, які більше не потрібні, допомагає запобігти перенавчанню мережі. Шар об’єднання зазвичай вставляється після шару згортки перед наступним шаром згортки [7].

Згорткові мережі можуть включати локальні або глобальні рівні агрегації, які поєднують виходи кластерів нейронів одного шару в один нейрон наступного рівня [8, 9]. Наприклад, шар об’єднання використовує максимальне значення з кожного з кластерів нейронів на попередньому рівні.

На додаток до максимізації агрегації, вузли агрегації можуть використовувати інші функції, такі як об’єднання середніх значень і об’єднання L2-нормалізації. Через агресивне зменшення розміру представлення зображення тенденція спрямована на менші фільтри або повну відмову від рівня агрегації [10].

Після кількох шарів згортки та об’єднання може бути один або кілька повнозв’язаних шарів, які приймають усі виходи нейронів попереднього шару та з’єднують їх з кожним нейроном поточного шару для генерації глобальної

семантичної інформації [11]. Нейрони в повнозв'язаному шарі мають зв'язки з усіма збудженнями попереднього шару, як це можна побачити в звичайних нейронних мережах. Повністю пов'язаний шар не завжди необхідний, оскільки його можна замінити згортковим шаром 1×1 [12].

Останній рівень ЗНМ є вихідним. Для завдань класифікації зазвичай використовується функція активації нейронів Softmax [13]. Рівень втрати визначає, як навчання штрафує відхилення між прогнозованими та істинними мітками, і зазвичай є останнім рівнем.

Навчання ЗНМ є завданням глобальної оптимізації. Шляхом мінімізації функції втрат можна знайти найбільш відповідний набір параметрів. Стохастичний градієнтний спуск є найпоширенішим рішенням для оптимізації згорткової нейронної мережі [14].

1.3 Огляд і аналіз існуючих рішень

Автори [15] запропонували модель згорткової нейронної мережі для класифікації захворювань шкіри на основі злиття моделей. Завдяки об'єднанню моделей, глибокому та поверхневому об'єднанню ознак, а також запровадженню модуля "уваги" було посилено здатність моделі виділяти ознаки. Крім того, було проведено серію дій, таких як попереднє навчання моделі, доповнення даних і точне налаштування параметрів, щоб покращити ефективність класифікації моделі. Результати експерименту показали, що під час роботи з нашим приватним набором даних, де переважають шкірні захворювання, подібні до вугрів, запропонована модель перевершила дві базові моделі DenseNet201 і ConvNeXt_L на 4,42% і 3,66% відповідно. У загальнодоступному наборі даних HAM10000 точність і показник f1 запропонованої моделі становили 95,29% і 89,99% відповідно, що також досягло хороших результатів порівняно з іншими сучасними моделями.

Вони навчили та протестували різні моделі ЗНМ, зокрема ResNet50, EfficientNet_B4, DenseNet201 та ConvNeXt_L, на своєму наборі даних, що характеризується малим розміром вибірки та дуже незбалансованими категоріями.

Результати показали, що DenseNet201 і ConvNeXt_L досягли найкращих показників класифікації з рівнем точності 92,12% і 92,88% відповідно. Для подальшого вдосконалення моделей класифікації дослідники запропонували вдосконалення моделей DenseNet і ConvNeXt.

Для DenseNet вони ввели блок Efficient Channel Attention (ECA) у внутрішній модуль для покращення вилучення функцій. Ця модифікація була спрямована на вирішення проблеми зникнення градієнта та покращення можливості представлення моделі. Подібним чином для ConvNeXt вони включили як ECA, так і новий модуль уваги Gated-Channel Transformation (GCT), щоб покращити агрегацію функцій і явно змоделювати зв'язки між каналами. Повна структура запропонованої ними моделі показана на рисунку 1.3.

У [16] автори запропонували комп'ютеризований процес класифікації шкірних захворювань за допомогою глибокого навчання на основі MobileNet V2 і довгострокової пам'яті (LSTM). Модель MobileNet V2 виявилася ефективною з кращою точністю, яка може працювати на легких обчислювальних пристроях. Ефективність порівнювалася з іншими найсучаснішими моделями, такими як точно налаштовані нейронні мережі (FTNN), згорткова нейронна мережа (ЗНМ), дуже глибокі згорткові мережі для розпізнавання великомасштабних зображень, розроблені Visual Geometry Group (VGG) і архітектуру згорткової нейронної мережі, яка розширилася з невеликими змінами. Використовується набір даних HAM10000, і запропонований метод перевершив інші методи з більш ніж 85% точністю. Його надійність у розпізнаванні ураженої області набагато швидше з майже вдвічі меншими обчисленнями, ніж звичайна модель MobileNet, приводить до мінімальних обчислювальних зусиль.

Крім того, створено мобільний додаток для миттєвої та правильної класифікації. Це допомагає пацієнту та дерматологам визначити тип захворювання за зображенням ураженої ділянки на початковій стадії захворювання шкіри. Це свідчить про те, що запропонована система може допомогти лікарям загальної практики ефективно та результативно діагностувати захворювання шкіри, тим самим зменшуючи подальші ускладнення та захворюваність.

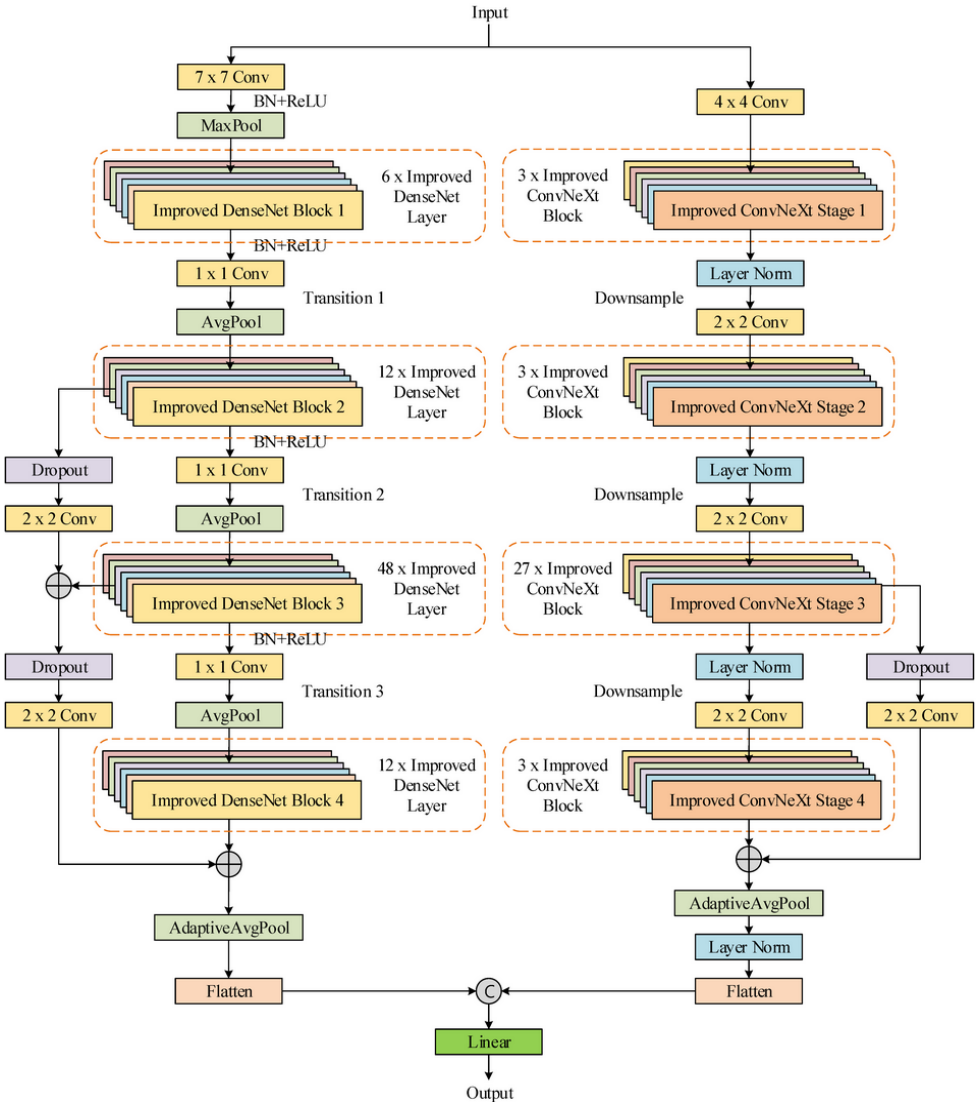

Рисунок 1.3 – Повна структура запропонованої моделі [15]

У [17] показана методика глибокого навчання для надійної діагностики типу меланоми на перших стадіях. Запропонована модель розрізняє злоякісне ураження, поверхневе поширення та вузлову меланому. Це дозволяє ранню діагностику вірусу та швидку ізоляцію та терапію, необхідну для зупинки подальшої передачі інфекції. Глибоке навчання і стандартний непараметричний метод машинного навчання представлені в топології глибокого шару згорткової нейронної мережі. Ефективність класифікатора ЗНМ була оцінена з використанням даних, отриманих з веб-сайту https://dermnetnz.org/. Результати експериментів показують, що запропонований метод перевершує діагностичну точність порівняно з методологіями, які на сьогодні вважаються найсучаснішими.

1.4 Постановка задачі дослідження

Аналіз предметної області та існуючих рішень показав, що автоматизація класифікації шкірних захворювань, зокрема проблема розпізнавання різних класів уражень шкіри, є дуже актуальною та затребуваною в багатьох галузях. Вирішення цієї проблеми дозволить прискорити процес діагностики, забезпечивши швидкий аналіз зображень шкіри, дозволяючи медичним працівникам приймати більш обґрунтовані рішення. Крім того, це може розширити можливості пацієнтів, надаючи їм інформацію про їхній стан шкіри. Пацієнти можуть використовувати програмне забезпечення, щоб дізнатися більше про стан здоров'я шкіри, зрозуміти можливі варіанти лікування та брати активну участь у прийнятті рішень щодо свого здоров'я. Великі набори даних зображень шкіри, зібрані програмним забезпеченням класифікації, можуть бути цінними ресурсами для дерматологічних досліджень. Дослідники можуть аналізувати ці набори даних, щоб отримати уявлення про поширеність, характеристики та результати лікування різних шкірних захворювань, що зрештою сприяє прогресу в цій галузі. Доступ до дерматологічної експертизи може бути обмежений у певних географічних регіонах або закладах охорони здоров'я. Програмне забезпечення для класифікації захворювань шкіри може подолати цю прогалину, надаючи доступ до діагностичної підтримки віддалено, дозволяючи постачальникам медичних послуг надавати якісну допомогу незалежно від місця розташування.

Розглянувши найпоширеніші методи розпізнавання ушкоджень шкіри та основні алгоритми виявлення шкірних артефактів у фото чи відеопотоці, для подальшої обробки можна висунути декілька вимог до реалізації програмного модуля, який буде розпізнавати ушкодження шкіри на вхідному зображенні. Також було розглянуто приклад програми для мобільних пристроїв, яка виконує артефакти шкіри, і виявилося, що не існує таких програмних модулів для комерційних цілей, які б працювали з веб-камерою ПК. Тому створення такого програмного модуля для комп'ютера є доцільним.

Оскільки програмний модуль, що буде розроблений, у першу чергу

орієнтований на сегмент персональних комп’ютерів або ноутбуків і повинен класифікувати захворювання шкіри з веб-камери персонального комп’ютера в реальному часі, найважливішим є швидкість алгоритму обробки вхідного відео та розпізнавання шкірних артефактів. Основним завданням є створення швидкого, точного та універсального програмного модуля для класифікації захворювань шкіри завдяки дослідженню існуючих методів захоплення кадрів, обробки зображень та розпізнавання шкірних артефактів.

Метою кваліфікаційної роботи є створення програмного модуля для класифікації уражень шкіри шляхом поєднання методів оптичного розпізнавання зображень, згорткової нейронної мережі, попередньої обробки зображення та аналізу отриманих результатів. Для досягнення поставленої мети необхідно вирішити такі завдання:

– провести опис предметної області;

– проаналізувати існуючі рішення для класифікації шкірних захворювань з використанням штучних нейронних мереж;

– зробити постановку задачі дослідження;

– представити загальну структуру і розробити алгоритм програмного модуля класифікації шкірних захворювань;

– розробити архітектуру згорткової нейронної мережі для класифікації захворювань шкіри за вхідним зображенням веб-камери комп’ютера;

– розробити програмний модуль для класифікації захворювань шкіри;

– протестувати програмний модуль і перевірити правильність розпізнавання.

2 ЗАГАЛЬНА СТРУКТУРА ТА АЛГОРИТМІЧНЕ ЗАБЕЗПЕЧЕННЯ ПРОГРАМНОГО МОДУЛЯ КЛАСИФІКАЦІЇ ШКІРНИХ ЗАХВОРЮВАНЬ

2.1 Загальна структура та алгоритм програмного модуля

Загальна структура програмного модуля класифікації шкірних захворювань може включати кілька основних компонентів, кожен з яких виконує специфічні функції для забезпечення ефективного аналізу та класифікації зображень шкіри.

Така структура забезпечуватиме системний підхід до розробки, навчання, тестування та розгортання програмного модуля для класифікації шкірних захворювань, що дозволить отримати надійний та ефективний інструмент для медичних працівників.

Структурна схема модуля класифікації хвороб шкіри включає наступні компоненти:

– вхідний відеопотік, який буде оброблено;

– веб-камера комп’ютера для захоплення вхідного відео;

– модуль обробки відеопотоку в реальному часі, який оброблятиме відеопотік у реальному часі;

– модель розпізнавання (прогнозування) уражень шкіри;

– результати класифікації, створені моделлю.

Кожен компонент послідовно з’єднаний стрілками, що вказують перехід від одного етапу до наступного, починаючи з відео-входу і закінчуючи результатами класифікації.

Спочатку необхідно отримати вхідне зображення. Для цього використовується веб-камера, неважливо, статичне це зображення або в русі (відеопотік). Веб-камера персонального комп’ютера передає зображення на блок обробки даних за допомогою блоку обробки потоку живого відео. У самому цьому блоці за допомогою нейронної мережі визначається, чи є на цьому зображенні один із потрібних об’єктів – ураження шкіри. Якщо ураження наявне, то нейронна мережа починає обробку цього кадру зображення, щоб визначити, яке захворювання шкіри присутнє в цьому кадрі зображення. Після проходження всіх

цих кроків формується відповідний результат, який відображається у відповідному вікні програмного модуля.

Існує кілька різних способів розпізнати об’єкт на зображенні, проте більшість із них не дуже точні та вимагають певних умов для успішного розпізнавання.

Найкращим серед цих методів є нейронна мережа, а саме згорткова нейронна мережа. Існує багато різних їх модифікацій, однак спільним для них є те, що вони вимагають досить тривалого попереднього навчання, яке сильно залежить від потужності машини, на якій воно буде проводитися. Навчання може зайняти від кількох годи, якщо машина досить потужна і необхідно навчитися розпізнавати лише невелику кількість об'єктів, до кількох тижнів або навіть місяців, якщо машина не надто потужна.

Модуль захоплення та попередньої обробки зображення є компонентом системи, що відповідає за графічний аналіз зображення та послідовне виділення в ньому структурних елементів зображення за запитом модуля розпізнавання шкірних захворювань. Алгоритм цього модуля представлено на рисунку 2.1.

Він отримує в запиті вказівку на появу очікуваного елемента. Тоді, під час обробки зображення, воно намагатиметься слідувати шляхом, який приведе до виявлення елемента шуканого типу. Після виконання завдання модуль повертає інформацію про виявлений елемент і точки його перетину з іншими елементами. Якщо це завдання виконати неможливо, повертається негативна відповідь.

Модуль попередньої обробки зображення (див. рисунок 2.1) починається із завантаження попередньо навченої моделі ЗНМ на даних про ураження шкіри. Потім він захоплює один кадр зображення з безперервного відеопотоку, що надходить із веб-камери пристрою. Він завантажує цей кадр у пам'ять модуля. Після цього зняте зображення розбивається на NxN комірок. Використовуючи нейронну мережу, цей модуль переглядає кожну клітинку та здійснює пошук будь-якого об’єкта в цих клітинках.

Далі виконується виявлення знайдених об’єктів і призначається ймовірність того, чи є цей об’єкт ураженням шкіри. Якщо ймовірність нижча за деякий

призначений поріг, таке ураження не виявляється у відеопотоці. В іншому випадку алгоритм вибирає комірку з найвищою ймовірністю та видаляє все навколо вибраного об’єкта. Після цього це зображення передається на наступний етап класифікації, який є розпізнаванням класу шкірного захворювання.

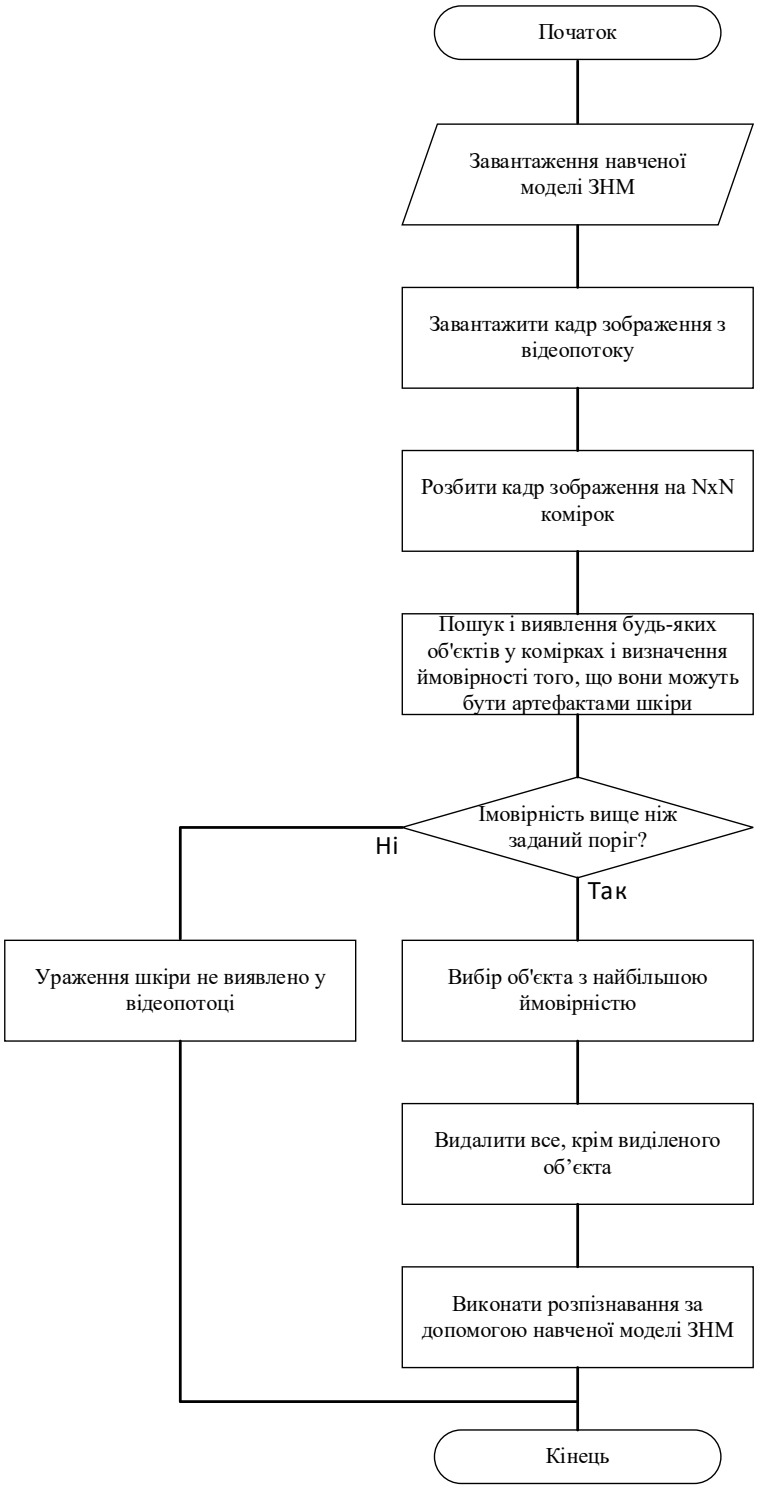

Рисунок 2.1 – Схема алгоритму модуля попередньої обробки зображення

Модуль захоплення та попередньої обробки зображення ураження шкіри з відеопотоку веб-камери – системний компонент, який відповідає за прийом відеопотоку з веб-камери, обробку зображення та підготовку його для подальшого розпізнавання типу захворювання шкіри за допомогою згорткової нейронної мережі. Підготовка в цьому контексті включає підготовку зображення шкіри перед його подальшим розпізнаванням за допомогою згорткової нейронної мережі. Цей етап може включати такі кроки:

а) приведення вхідного розміру зображення до відповідного стандарту;

б) нормалізація.

Після підготовки зображення його можна передати на вхід згорткової нейронної мережі для подальшої класифікації захворювань шкіри. Нейронну мережу зазвичай навчають розпізнавати певні ушкодження шкіри за допомогою навчального набору даних, і після навчання, модель може передбачити клас захворювання шкіри на основі вхідного зображення. Це описано в наступних підрозділах.

Алгоритм програмного модуля класифікації захворювань шкіри – це послідовність кроків і процедур, які виконує модуль для розпізнавання та інтерпретації уражень шкіри на зображенні, зробленому користувачем. Основні кроки алгоритму можуть включати: захоплення відеопотоку, попередню обробку, класифікацію шкірних уражень та інтерпретацію результатів [18].

Алгоритм програмного модуля класифікації шкірних захворювань представлений у додатку А. Він починається із завантаження навченої моделі ЗНМ та ініціалізації навчених параметрів і ваг з цієї моделі. Потім програмний модуль підключається до веб-камери ноутбука через розроблений інтерфейс. Починається запис відео. Якщо якість відеозображення недостатня, розпочнеться процес повторного калібрування камери, який виконуватиметься безперервно, доки якість зображення не стане прийнятною. Якщо якість зображення хороша, модуль визначить ROI (область інтересу) з об’єктом ураження шкіри за допомогою модуля попередньої обробки зображення, описаного раніше в цьому підрозділі. Перетворить захоплене зображення кадру відео в належний формат, щоб

використати його як вхідні дані для модуля прогнозування класу захворювань шкіри. Розпізнає отримані артефакти шкіри за допомогою навченої моделі ЗНМ. Після того, як це пошкодження шкіри буде розпізнано, зафіксує цей об’єкт і дасть ймовірність цього розпізнавання. Віднесе розпізнане ураження шкіри до певного класу та виведе на екран результати. В іншому випадку, якщо шкірне захворювання не розпізнано, модуль продовжує попередню обробку вхідного відеопотоку та виконує процедуру розпізнавання до досягнення хороших результатів.

2.2 Архітектура згорткової нейронної мережі для програмного модуля

Процес навчання згорткової нейронної мережі починається з підготовки набору даних. По-перше, потрібно мати набір даних, який містить вхідні зображення та відповідні мітки або категорії для цих зображень. Набір даних зазвичай поділяють на навчальні, перевірочні та тестові зразки. Ініціалізація мережі – створення згорткової архітектури нейронної мережі, що включає визначення кількості та розміру шарів, функцій активації, розмірів фільтрів тощо. Параметри мережі ініціалізуються випадковими значеннями або за певними правилами. Пряме поширення – це коли вхідні зображення поширюються мережею від вхідного рівня до вихідного. Кожен шар виконує функцію згортання, об’єднання, активації та передає результати на наступний рівень. Вихідні значення мережі порівнюються з очікуваними мітками для обчислення функції втрат. Розрахунок втрат – це коли функція втрат вимірює різницю між прогнозованими значеннями та очікуваними мітками [19].

Зворотне поширення помилки виконується після обчислення втрат, помилка поширюється від вихідного рівня до вхідного за допомогою алгоритму зворотного поширення. Це дозволяє розрахувати градієнти функції втрат на ваги та зміщення мережі. Потім виконується оптимізація ваг і зсувів. Градієнти, отримані в процесі зворотного поширення, використовуються для оновлення ваг і зсувів мережі. Це робиться за допомогою алгоритму оптимізації, наприклад стохастичного градієнтного спуску (SGD) або його варіантів. Алгоритм оптимізації регулює ваги

та зсуви, щоб мінімізувати функцію втрат.

Ітерація та навчання є процесом зворотного поширення та оптимізації ваги, що повторюється протягом багатьох епох навчання. Кожна епоха включає передачу вперед, розрахунок втрат, зворотне поширення та оновлення ваг. Це дозволяє моделі покращувати свою передбачувану здатність з кожною ітерацією. Оцінка та тестування проводяться після завершення навчання, модель оцінюється на зразку валідації, щоб оцінити її точність і продуктивність. Крім того, модель можна протестувати на тестовому зразку, щоб оцінити її загальну здатність розпізнавати нові зображення.

Останні кроки: налаштування та вдосконалення. Залежно від результатів оцінювання та тестування, для покращення результатів можуть бути внесені зміни в архітектуру мережі, гіперпараметри або процес навчання. Цикл навчання-оцінювання-вдосконалення може тривати до досягнення задовільних результатів.

Алгоритм навчання ЗНМ для розроблюваного програмного модуля класифікації шкірних захворювань представлений у додатку Б. Він починається із завантаження попередньо навчених ваг ЗНМ і файлів конфігурації. Після цього вибираються зображення для навчальних цілей. Потім ці зображення розбиваються на навчальні та тестові набори. Якщо недостатньо зображень для навчання, необхідно згенерувати додаткові дані для навчання.

Існує кілька способів генерації додаткових даних для навчання нейронної мережі. Найпопулярнішим методом є аугментація (додавання) даних, який передбачає створення нових зображень шляхом застосування різноманітних перетворень до існуючих даних. Наприклад, зображення можна зміщувати, обертати, масштабувати, змінювати контрастність чи яскравість тощо. Це дозволяє розширити різноманітність даних і покращити загальну здатність моделі до узагальнення.

Після створення додаткових даних необхідно розділити навчальні дані на менші частини та розпочати навчання цієї моделі ЗНМ. Це є важливою технікою навчання нейронних мереж і робиться з кількох причин: обчислювальна ефективність, менша потреба в пам’яті, адаптивність градієнтного спуску,

регулярізація та уникнення перенавчання.

Налаштування навчальних ваг у процесі навчання нейронної мережі є основним кроком у пошуку оптимальних параметрів моделі і робиться з метою покращення здатності мережі прогнозувати та зменшення функції втрат.

Мережа продовжує навчання, доки не будуть оброблені всі зображення з одного батчу. ЗНМ обробляє всі батчі послідовно один за одним, фактично весь набір даних, до кінця епохи навчання. Далі алгоритм навчання оцінює помилку та зменшує швидкість (рейт) навчання, щоб перейти до наступної епохи навчання. Зменшення швидкості навчання нейронної мережі є важливою стратегією оптимізації, і вона використовується для різних цілей. Це приводить до підвищення стабільності навчання – великі значення швидкості навчання можуть привести до нестабільності навчання. Це веде до кращого узагальнення. Зменшення рейту навчання може допомогти уникнути перенавчання. При великій швидкості навчання модель може дуже точно відтворювати навчальні дані, але не зможе добре працювати на нових зображеннях [20]. Загалом зниження швидкості навчання дозволяє краще контролювати процес навчання, покращує стабільність, точність і збіжність моделі, а також допомагає уникнути перенавчання.

Після завершення всіх епох навчання останнім кроком є збереження навчених ваг моделі у файл. Він буде використаний у майбутніх прогнозах.

Архітектура згорткової нейронної мережі, яка використовується для програмного модуля класифікації захворювань шкіри, представлена на рисунку 2.2.

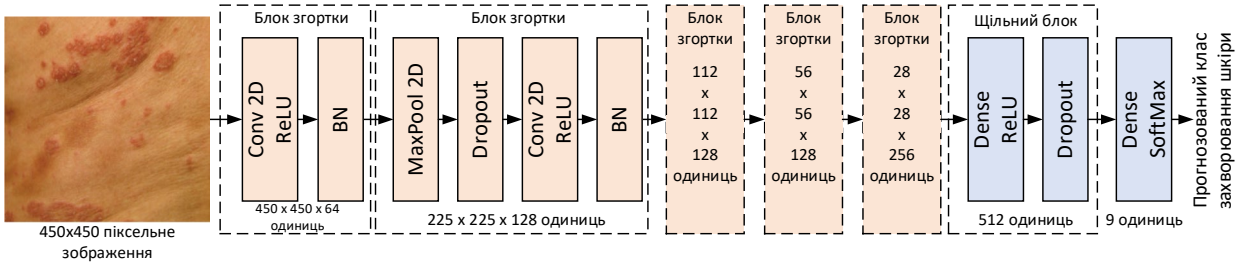

Рисунок 2.2 – Архітектура ЗНМ, що використовується для класифікації захворювань шкіри [27]

У ЗНМ карти функцій представляють вивчені особливості вхідного зображення або даних. Під час проходу вперед ЗНМ кожен згортковий рівень застосовує набір фільтрів до вхідних даних. Кожен фільтр виявляє певні візерунки або особливості, такі як грані, текстури або форми, у різних просторових розташуваннях у вхідних даних. Результатом кожного фільтра є карта ознак, яка підкреслює присутність і розташування певного об’єкта у вхідних даних.

Для першого згорткового блоку після обробки згорткового шару є 64 карт ознак розміром 450x450 пікселів, для другого блоку є 128 карт ознак розміром 225x225 пікселів, для третього блоку є 128 карт ознак розміром 112x112 пікселів, для четвертого блоку є 128 карт ознак розміром 56x56 пікселів та для п'ятого блоку є 256 карт ознак розміром 28x28 пікселів. Після цих блоків іде один щільний шар із 512 картами ознак, який перетворюється на щільний блок із 9 одиниць, який відображає 9 ймовірностей того, що шкірне захворювання може відноситись до певного класу. У результаті класифікації вибирається та відображається клас з найвищою ймовірністю. Кожен із цих блоків виконує певні перетворення кожної карти ознак.

Рівень Conv2D складається з набору фільтрів, які можна розглядати як невеликі підвибірки або шаблони. Кожен фільтр має ваги, які використовуються для згортки з вхідним зображенням.

Рівень Conv2D відіграє важливу роль у виявленні різних візерунків, меж і особливостей на зображеннях, дозволяючи нейронній мережі автоматично виконувати розпізнавання образів і вирішувати завдання комп’ютерного зору.

ReLU (Rectified Linear Unit), функція активації, яка широко використовується в штучних нейронних мережах встановлює вихідний сигнал на нуль для всіх від’ємних значень вхідного сигналу, тоді як для додатних значень вона передає їх без змін [27].

Нормалізація (BN) – техніка, яка використовується в штучних нейронних мережах для нормалізації вхідних даних між мережевими рівнями, що сприяє стабільному та швидкому навчанню мережі та покращує її загальну продуктивність [27].

MaxPool2D (максимальний рівень об’єднання) – операція, яка використовується в ЗНМ для зменшення розміру просторових розмірів (ширина та висота) ознак зображення, виконується шляхом вилучення найбільш значущих значень у кожному субрегіоні з вхідної області та використання їх як вихідних значень.

Dropout layer – це метод регулярізації, який використовується в нейронних мережах для запобігання перенавчанню. Він випадковим чином вимикає вихідні значення деяких нейронів під час навчання.

Рівень SoftMax – це функція активації, яка використовується на останньому рівні нейронних мереж для прогнозування ймовірностей для набору класів. Механізм її роботи полягає в перетворенні вектора значень в діапазон ймовірностей, де кожне значення відображає ймовірність відносної приналежності до певного класу [27].

Повнозв’язний шар з’єднує всі входи попереднього шару з кожним виходом наступного. У кожному нейроні цього шару виконується лінійна комбінація вхідних значень, вона включає ваги та зміщення, а наступний крок – результат передається через функцію активації.

2.3 Інформаційне, програмне та апаратне забезпечення програмного модуля

Skin Disease Classification Dataset [28] це колекція дерматоскопічних зображень поширених пігментних уражень шкіри. Він широко використовується в дослідженнях для розробки та оцінки алгоритмів машинного навчання та моделей глибокого навчання для класифікації та діагностики захворювань шкіри.

Цей набір даних містить загалом 900 дерматоскопічних зображень з роздільною здатністю 450x450 пікселів. Зображення в наборі даних класифікуються за дев’ятьма діагностичними категоріями, зокрема:

- актинічний кератоз;
- атопічний дерматит;

– доброякісний кератоз;

– дерматофіброма;

– меланоцитарний невус;

– меланома;

– плоскоклітинний рак;

– кандидоз стригучого лишаю;

– ураження судин.

Мітки, присвоєні зображенням, мають діапазон від 0 до 8, що представляють ці категорії захворювань шкіри. Приклади зображень із набору даних про шкірні захворювання показані на рисунку 2.3.

Для забезпечення високої точності прогнозу такої кількості навчальних зображень недостатньо. Для створення більшої кількості навчальних вибірок необхідно використовувати техніку аугментації.

Аугментація вхідних зображень розміром 450x450 пікселів – це процес штучного розширення навчальної вибірки шляхом створення додаткових зображень на основі існуючих. Метою такого доповнення є покращення продуктивності нейронної мережі за рахунок збільшення різноманітності даних і підвищення стійкості моделі до змін у вхідних даних. У контексті вхідних зображень захворювань шкіри 450x450 доповнення включає такі операції: обертання, зміщення, масштабування, зміна яскравості та контрастності, збільшення шуму.

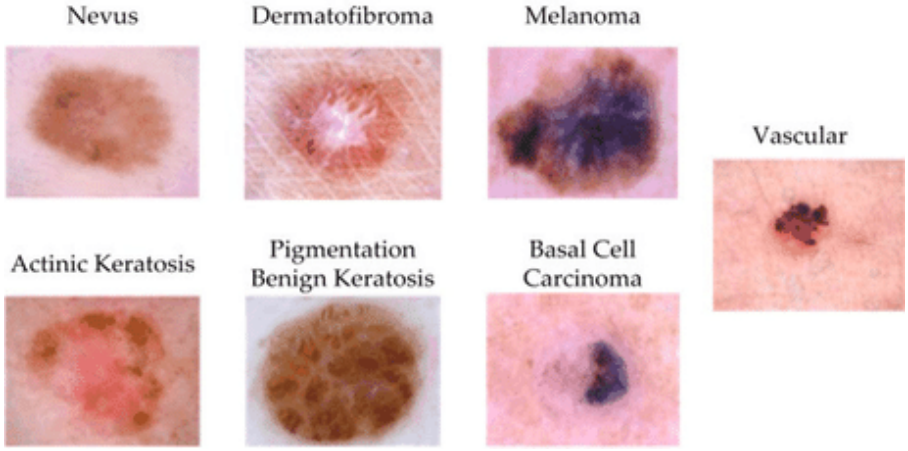

Рисунок 2.3 – Набір даних про захворювання шкіри

На рисунку 2.4 показано приклад техніки збільшення зображення ураження шкіри.

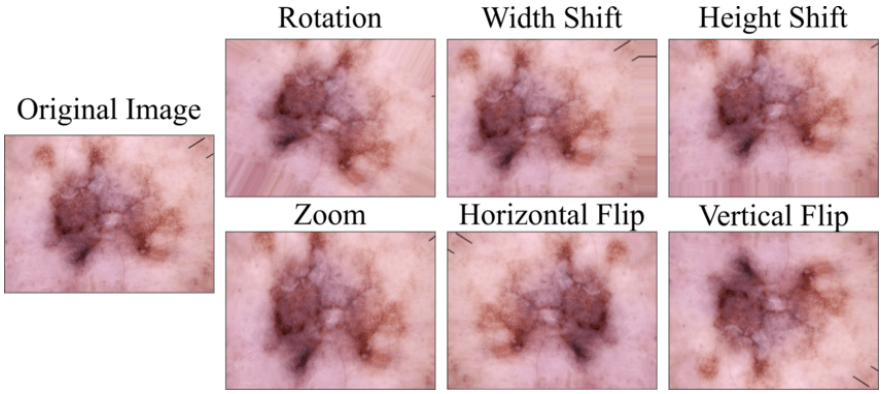

Рисунок 2.4 – Техніка аугментації

Потужна та популярна мова програмування Python широко використовується в машинному та глибокому навчанні. Простий і зрозумілий синтаксис робить її привабливим вибором для розробників, які хочуть будувати та навчати ЗНМ.

Бібліотека Keras [21] є однією з найпопулярніших бібліотек глибокого навчання високого рівня для Python, яка забезпечує зручний та інтуїтивно зрозумілий інтерфейс для розробки та навчання ЗНМ, вона дозволяє швидко та легко визначати архітектуру ЗНМ, додавати шари, налаштовувати функції активації та інші параметри моделі.

Крім того, Python і Keras мають багато бібліотек підтримки, наприклад NumPy для роботи з числовими обчисленнями, Matplotlib для візуалізації даних і результатів, Pandas для роботи з даними. Загалом, Python і Keras надають потужні інструменти для розробки та навчання штучних нейронних мереж (ШНМ). Їх поєднання дозволяє досліджувати різні архітектури ШНМ, експериментувати з параметрами моделі та тренувати моделі на великих наборах даних.

Visual Studio Code, розроблений Microsoft, є популярним інтегрованим середовищем розробки, яке широко використовується для програмування на мові Python і побудови згорткових нейронних мереж.

Для навчання ЗНМ у роботі використовується відеокарта NVIDIA GeForce RTX 3060 [22], яка є потужним графічним прискорювачем, особливо в порівнянні з інтегрованими графічними рішеннями або старими моделями відеокарт.

3 ПРОГРАМНО-ТЕХНОЛОГІЧНЕ ЗАБЕЗПЕЧЕННЯ ПРОГРАМНОГО МОДУЛЯ КЛАСИФІКАЦІЇ ШКІРНИХ ЗАХВОРЮВАНЬ

3.1 Навчання моделі згорткової нейронної мережі для класифікації шкірних захворювань

Щоб реалізувати програмний модуль класифікації шкірних захворювань, спочатку необхідно навчити згорткову нейронну мережу, яка класифікуватиме ці ураження шкіри до певних класів. Для навчання було використано мову програмування Python з бібліотекою Keras для нейронних мереж.

Початковий етап підготовки даних для навчання передбачає перетворення та зміну форми піксельних даних із набору даних у формат зображення, що дозволяє обробити його алгоритмом. Підготовка даних для згорткової нейронної мережі включає кілька кроків, щоб забезпечити відповідний формат і оптимальні умови для навчання мережі.

При підготовці даних навчальний та тестовий набори нормалізуються із застосуванням ділення на 255. Як було описано раніше в підрозділі огляду набору даних, кожен піксель зображення шкіри має значення сірого від 0 до 255. Цей процес нормалізує значення в діапазоні від 0 до 1, покращуючи процес навчання та покращуючи продуктивність моделі. Як правило, зображення у відтінках сірого представляють значення пікселів як цілі числа від 0 до 255. Тут 0 позначає мінімальну інтенсивність (чорний), а 255 означає максимальну інтенсивність (білий). Нормалізуючи значення пікселів, вони приведені до стандартизованого діапазону, забезпечуючи послідовність і полегшуючи навчання моделі.

Наступним кроком буде генерація деяких додаткових даних для навчання за допомогою аументації. Код генератора даних представлено на рисунку 3.1. ImageDataGenerator() — це клас, наданий бібліотекою Keras у Python, який дозволяє доповнювати дані в реальному часі під час навчання нейронних мереж, зокрема для даних зображень.

Клас ImageDataGenerator пропонує різноманітні методи трансформації та аугментації зображень: обертання, зсув по ширині/висоті,

горизонтальне/вертикальне відображення, масштабування, тощо.

```
img_size = (450, 450)

def data_preprocessing(path):
    datagen = tf.keras.preprocessing.image.ImageDataGenerator(rescale=1.0/255,
                                                              featurewise_center=False,
                                                              samplewise_center=False,
                                                              featurewise_std_normalization=False,
                                                              samplewise_std_normalization=False,
                                                              zca_whitening=False,
                                                              rotation_range=10,
                                                              zoom_range = 0.1,
                                                              width_shift_range=0.1,
                                                              height_shift_range=0.1,
                                                              horizontal_flip=False,
                                                              vertical_flip=False)

    generator = datagen.flow_from_directory(
    path,
    batch_size=25,
    class_mode='categorical',
    target_size=img_size,
    color_mode="rgb"
    )

    return generator
```

Рисунок 3.1 – Генератор даних

ImageDataGenerator має кілька параметрів, які дозволяють застосовувати різні трансформації та доповнення зображення в процесі навчання. Усі параметри, які не використовуються, дорівнюють False. Ось описи операцій, які використовуються для доповнення даних дерматоскопії:

а) rotation_range: цей параметр визначає діапазон випадкових поворотів, які можна застосувати до зображень. Його встановлено на 10, щоб зображення можна було випадково обертати в діапазоні від -10 до +10 градусів;

б) zoom_range: цей параметр визначає діапазон випадкового масштабування, яке можна застосувати до зображень;

в) width_shift_range: цей параметр контролює діапазон випадкового горизонтального зсуву зображень;

г) height_shift_range: такий же, як width_shift_range, цей параметр встановлює діапазон випадкового вертикального зсуву зображень.

Код, який визначає архітектуру нейронної мережі, представлений на рисунку 3.2.

Model = Sequential() створює послідовну модель у Keras: лінійний стек шарів, де кожен шар додається один за одним. Це дозволяє побудувати нейронну мережу,

просто склавши шари в тому порядку, у якому необхідно їх виконувати.

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(filters=64, kernel_size=3, activation=act_func, padding='same', input_shape=(img_size[0], img_size[1], 3), kernel_initializer='he_normal'),
    tf.keras.layers.BatchNormalization(axis=-1),
    tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),
    tf.keras.layers.Dropout(0.3),

    tf.keras.layers.Conv2D(filters=128, kernel_size=5, activation=act_func, padding='same', kernel_initializer='he_normal'),
    tf.keras.layers.BatchNormalization(axis=-1),
    tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),
    tf.keras.layers.Dropout(0.3),

    tf.keras.layers.Conv2D(filters=128, kernel_size=5, activation=act_func, padding='same', kernel_initializer='he_normal'),
    tf.keras.layers.BatchNormalization(axis=-1),
    tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),
    tf.keras.layers.Dropout(0.3),

    tf.keras.layers.Conv2D(filters=256, kernel_size=3, activation=act_func, padding='same', kernel_initializer='he_normal'),
    tf.keras.layers.BatchNormalization(axis=-1),
    tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),
    tf.keras.layers.Dropout(0.3),

    tf.keras.layers.Conv2D(filters=512, kernel_size=3, activation=act_func, padding='same', kernel_initializer='he_normal'),
    tf.keras.layers.BatchNormalization(axis=-1),
    tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),
    tf.keras.layers.Dropout(0.3),

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(256, activation=act_func),
    tf.keras.layers.BatchNormalization(axis = -1),
    tf.keras.layers.Dropout(0.4),
    tf.keras.layers.Dense(256, activation=act_func),
    tf.keras.layers.BatchNormalization(axis = -1),
    tf.keras.layers.Dropout(0.4),

    tf.keras.layers.Dense(9, activation='softmax')

])
```

Рисунок 3.2 – Визначена архітектура ЗНМ

У цьому випадку є шари: Conv2D, MaxPooling2D, Dense, Dropout тощо, які були описані в попередньому підрозділі. Після визначення моделі було скомпільовано її з відповідною функцією втрат, оптимізатором і показниками, а потім навчено її на даних за допомогою методу fit().

Останнім кроком буде компіляція та збереження навчальної моделі у багаторазовому форматі .h5. На рисунку 3.3 представлено код для компіляції та збереження моделі. Визначаються оптимізатор, функція втрат і метрики.

```
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy', Precision(), Recall()])
history_model = model.fit(train_generator, epochs = 30, validation_data = validation_generator)
model.save('skin_detect_model.h5')
model.save_weights('skin_detect_model_weights.h5')
```

Рисунок 3.3 – Компіляція та збереження моделі

Оптимізатор ADAM (Adaptive Moment Estimation - оцінка адаптивного моменту) – це алгоритм оптимізації, який зазвичай використовується для навчання нейронних мереж.

У контексті навчання нейронної мережі метрика «точність» забезпечує міру того, наскільки добре модель працює з точки зору правильного прогнозування класів вхідних даних. Вона обчислюється шляхом порівняння прогнозованих міток класів, створених моделлю, з істинними мітками класів і підрахунку кількості правильних прогнозів.

Після компіляції моделі отримано візуалізовану архітектуру розробленої моделі ЗНМ і загальну кількість навчених параметрів за допомогою інструментів Keras (рис. 3.4), а також підсумок про кожну епоху навчання (рис. 3.5).

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 450, 450, 64)      1792

 batch_normalization (BatchN  (None, 450, 450, 64)     256
 ormalization)

 max_pooling2d (MaxPooling2D  (None, 225, 225, 64)     0
 )

 dropout (Dropout)           (None, 225, 225, 64)      0

 conv2d_1 (Conv2D)           (None, 225, 225, 128)     204928

 batch_normalization_1 (Batc  (None, 225, 225, 128)    512
 hNormalization)

 max_pooling2d_1 (MaxPooling  (None, 112, 112, 128)    0
 2D)

 dropout_1 (Dropout)         (None, 112, 112, 128)     0

 conv2d_2 (Conv2D)           (None, 112, 112, 128)     409728
...
Total params: 27,856,649
Trainable params: 27,853,449
Non-trainable params: 3,200
```

Рисунок 3.4 – Візуалізована архітектура моделі

```
Epoch 1/30
28/28 [==============================] - 412s 15s/step - loss: 2.2667 - accuracy: 0.3099 - precision: 0.3672 - recall: 0.2123 - val_loss: 23.6987 - val_accuracy: 0.1105 - val_precision: 0.1105 - val_recall: 0.1105
Epoch 2/30
28/28 [==============================] - 402s 14s/step - loss: 1.7165 - accuracy: 0.4333 - precision: 0.4943 - recall: 0.3128 - val_loss: 3.7420 - val_accuracy: 0.1602 - val_precision: 0.1679 - val_recall: 0.1271
Epoch 3/30
28/28 [==============================] - 398s 14s/step - loss: 1.6413 - accuracy: 0.4433 - precision: 0.5482 - recall: 0.3429 - val_loss: 3.1584 - val_accuracy: 0.2486 - val_precision: 0.3553 - val_recall: 0.1492
Epoch 4/30
28/28 [==============================] - 396s 14s/step - loss: 1.4716 - accuracy: 0.4735 - precision: 0.5611 - recall: 0.3687 - val_loss: 3.4971 - val_accuracy: 0.2320 - val_precision: 0.2778 - val_recall: 0.1657
Epoch 5/30
28/28 [==============================] - 397s 14s/step - loss: 1.4554 - accuracy: 0.4821 - precision: 0.5635 - recall: 0.3945 - val_loss: 2.6651 - val_accuracy: 0.2762 - val_precision: 0.3243 - val_recall: 0.1326
Epoch 6/30
28/28 [==============================] - 395s 14s/step - loss: 1.3557 - accuracy: 0.5108 - precision: 0.6029 - recall: 0.4247 - val_loss: 2.6600 - val_accuracy: 0.1934 - val_precision: 0.1954 - val_recall: 0.0939
Epoch 7/30
28/28 [==============================] - 397s 14s/step - loss: 1.3933 - accuracy: 0.5065 - precision: 0.5591 - recall: 0.4003 - val_loss: 2.4037 - val_accuracy: 0.2597 - val_precision: 0.2929 - val_recall: 0.1602
Epoch 8/30
28/28 [==============================] - 396s 14s/step - loss: 1.3554 - accuracy: 0.5251 - precision: 0.5981 - recall: 0.4462 - val_loss: 1.8157 - val_accuracy: 0.3646 - val_precision: 0.4524 - val_recall: 0.2099
Epoch 9/30
28/28 [==============================] - 397s 14s/step - loss: 1.1857 - accuracy: 0.5681 - precision: 0.6333 - recall: 0.4634 - val_loss: 2.0894 - val_accuracy: 0.3260 - val_precision: 0.3802 - val_recall: 0.2541
Epoch 10/30
28/28 [==============================] - 397s 14s/step - loss: 1.2196 - accuracy: 0.5538 - precision: 0.6182 - recall: 0.4390 - val_loss: 1.6285 - val_accuracy: 0.4696 - val_precision: 0.5465 - val_recall: 0.2597
Epoch 11/30
28/28 [==============================] - 395s 14s/step - loss: 1.1266 - accuracy: 0.5897 - precision: 0.6765 - recall: 0.4950 - val_loss: 1.9466 - val_accuracy: 0.3204 - val_precision: 0.3737 - val_recall: 0.2044
Epoch 12/30
28/28 [==============================] - 398s 14s/step - loss: 1.1777 - accuracy: 0.5882 - precision: 0.6679 - recall: 0.4993 - val_loss: 1.4380 - val_accuracy: 0.4530 - val_precision: 0.5825 - val_recall: 0.3315
Epoch 13/30
...
Epoch 29/30
28/28 [==============================] - 431s 15s/step - loss: 0.8445 - accuracy: 0.6786 - precision: 0.7396 - recall: 0.6112 - val_loss: 1.1669 - val_accuracy: 0.6077 - val_precision: 0.6867 - val_recall: 0.5691
Epoch 30/30
28/28 [==============================] - 442s 16s/step - loss: 0.7470 - accuracy: 0.7102 - precision: 0.7698 - recall: 0.6428 - val_loss: 1.0453 - val_accuracy: 0.6243 - val_precision: 0.6755 - val_recall: 0.5635
```

Рисунок 3.5 – Підсумок навчальних епох

Код model.save_weights('skin_detect_model_weights.h5') використовується для збереження навченої моделі нейронної мережі у файл у форматі Hierarchical Data Format 5 (HDF5).

Зберігаючи модель у файлі, можна легко перезавантажувати та повторно використовувати навчену модель без необхідності перенавчати її з нуля.

3.2 Реалізація програмного модуля

Для реалізації програмного модуля використано бібліотеки Python: CV2 (OpenCV) і TKinter. Перша розпізнає об’єкти і обробляє зображення, а друга використовується для створення інтерфейсу для програмного модуля.

Спочатку імпортуються необхідні бібліотеки Python (рисунок 3.6).

```
import tkinter as tk
from tkinter import simpledialog
import cv2 as cv
import os
import PIL.Image, PIL.ImageTk
import model
import camera
```

Рисунок 3.6 – Імпорт бібліотек для програмного модуля

Далі потрібно активувати камеру, використовуючи бібліотеку CV2 для отримання вхідного зображення. Функція cv.VideoCapture() в OpenCV ініціює процес захоплення відео з різних джерел, таких як камери або відеофайли. Якщо вказано ціле число, воно представляє індекс камери, підключеної до комп’ютера. Наприклад, 0 зазвичай стосується камери за замовчуванням (вбудованої або першої зовнішньої камери), 1 означає другу камеру тощо. Функція cv.VideoCapture().read використовується для читання наступного кадру з об’єкта захоплення відео, створеного cv.VideoCapture(), повертає два значення: логічне значення, яке вказує, чи кадр було успішно прочитано, і сам кадр, який представлений у вигляді масиву NumPy. Метод read() використовується в циклі для безперервного захоплення кадрів із джерела відео, доки не буде виконано умову завершення (наприклад,

користувач натискає клавішу, щоб отримати прогноз). Цей процес описано на рисунку 3.7.

```python
import cv2 as cv

class Camera:

    def __init__(self):
        self.camera = cv.VideoCapture(1)
        if not self.camera.isOpened():
            raise ValueError("Unable to open camera!")

        self.width = self.camera.get(cv.CAP_PROP_FRAME_WIDTH)
        self.height = self.camera.get(cv.CAP_PROP_FRAME_HEIGHT)

    def __del__(self):
        if self.camera.isOpened():
            self.camera.release()

    def get_frame(self):
        if self.camera.isOpened():
            ret, frame = self.camera.read()

            if ret:
                return (ret, cv.cvtColor(frame, cv.COLOR_BGR2RGB))
            else:
                return (ret, None)
        else:
            return None
```

Рисунок 3.7 – Активація камери за допомогою OpenCV

Для наступного кроку необхідно перевірити, як модель насправді прогнозує. Для цього на навчену модель необхідно подати зображення з тестового набору (рисунок 3.8).

```
model.load_weights(checkpoint_path)
predict_disease(model, "{}/Atopic Dermatitis/1_42.jpg".format(path_val))
```

```
1/1 [==============================] - 0s 165ms/step
Probability for each class: [('Atopic Dermatitis', '0.91'), ('Tinea Ringworm Candidiasis', '0.09'),
```

Рисунок 3.8 – Прогнозування тестового зразка зображення

‘model’ представляє модель нейронної мережі, яку вже навчено, ‘load_weights(checkpoint_path)’ завантажує ваги в модель із файлу, визначеного ‘checkpoint_path’. ‘checkpoint_path’ – це шлях, який вказує на місце, де зберігаються ваги моделі. ‘predict_disease(model, image_path)’, ця функція приймає два аргументи: модель і шлях до файлу зображення. Він використовує модель для прогнозування захворювання шкіри, представленого на зображенні.

Як можна побачити, тестовий зразок зображення шкірного захворювання був класифікований з достовірністю 91%. Тепер необхідно додати функцію прогнозування в програмний модуль. Цей код представлено на рисунку 3.9.

```
def predict(self, frame):
    frame = frame[1]
    cv.imwrite("frame.jpg", cv.cvtColor(frame, cv.COLOR_RGB2GRAY))

    img = tf.keras.utils.load_img("frame.jpg", target_size=(450, 450, 3), color_mode = 'rgb')
    array = tf.keras.utils.img_to_array(img)
    array = array / 255.0

    img_array = np.expand_dims(array, axis=0)


    predictions = self.model.predict(img_array)

    for prediction in predictions:
        formatted_predictions = [f'{value:.2f}' for value in prediction]

    top_prob_index = np.argmax(formatted_predictions)
    top_prob = round(float(formatted_predictions[top_prob_index].replace(",", "."))*100, 2)

    return top_prob_index, top_prob
```

Рисунок 3.9 – Функція прогнозування

Наданий фрагмент коду виконує попередню обробку зображення та робить прогноз за допомогою попередньо навченої моделі. Ось опис того, що робить код:

– прийом вхідного кадру відео: метод приймає вхідні дані під назвою «frame», які є масивом. Він бере другий елемент масиву (‘frame[1]’), який є кадром зображення;

– завантаження та попередня обробка зображення: збережений “frame.jpg” завантажується за допомогою утиліти TensorFlow Keras (tf.keras.utils.load_img) із цільовим розміром 450x450 пікселів і в режимі кольору RGB;

– завантажене зображення перетворюється в масив

(tf.keras.utils.img_to_array);

– масив зображення нормалізовано шляхом ділення на 255,0 для масштабування значень пікселів до діапазону [0, 1];

– розширення розмірів: масив зображень розгортається вздовж нової осі, щоб додати пакетний розмір (np.expand_dims(array, axis=0)). Це необхідно, тому що модель очікує пакет зображень як вхідні дані;

– робить прогноз: попередньо оброблений масив зображень передається в модель для прогнозування (self.model.predict(img_array)), створюючи список прогнозів;

– визначення найвищого прогнозу: індекс прогнозу з найвищою ймовірністю визначається за допомогою np.argmax(formatted_predictions);

– повернення результату: метод повертає масив, що містить індекс найвищого прогнозу (top_prob_index) і відповідну ймовірність (top_prob).

Далі за допомогою бібліотеки TKinter створюється фрейм інтерфейсу, який містить потік з веб-камери ноутбука та кнопку для прогнозування класу захворювання шкіри. Код цього фрейму представлено на рисунку 3.10.

```
class App:

    def __init__(self, window=tk.Tk()):

        self.window = window
        self.window_title = window.title("WUNU - Skin disease classification software module")

        self.counters = [1, 1]

        self.model = model.Model()

        self.auto_predict = False

        self.camera = camera.Camera()

        self.init_gui()

        self.delay = 15
        self.update()

        self.window.attributes("-topmost", True)
        self.window.mainloop()

    def init_gui(self):

        self.canvas = tk.Canvas(self.window, width=self.camera.width, height=self.camera.height)
        self.canvas.pack()

        self.btn_predict = tk.Button(self.window, text="Predict class of skin disease", width=50, command=self.predict)
        self.btn_predict.pack(anchor=tk.CENTER, expand=True)

        self.class_label = tk.Label(self.window, text="WELCOME!!!")
        self.class_label.config(font=("Arial", 20))
        self.class_label.pack(anchor=tk.CENTER, expand=True)

    def update(self):
        if self.auto_predict:
            print(self.predict())

        ret, frame = self.camera.get_frame()
```

Рисунок 3.10 – Код для створення інтерфейсу

На рисунку 3.11 представлено програмне вікно модуля класифікації захворювань шкіри. Він виконує відеопотік в режимі реального часу і готовий

класифікувати вхідні зображення після натискання користувачем кнопки.

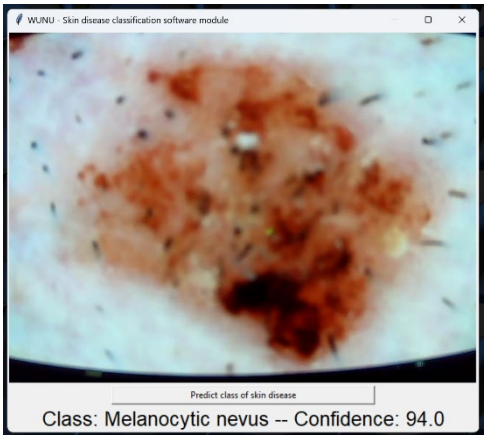

Рисунок 3.11 – Інтерфейс програмного модуля класифікації

Як можна побачити, програмний модуль працює належним чином, а інтерфейс простий і легкий у роботі. Деякі фрагменти програмного коду програмного модуля представлені в додатку В.

3.3 Тестування програмного модуля

Основною метою тестування є виявлення помилок, проблем, недоліків і некоректної роботи програмного модуля перед його використанням у реальних умовах.

Тестування програмного модуля для класифікації шкірних захворювань включає перевірку його здатності правильно ідентифікувати ураження шкіри відповідно до навченої моделі. Під час тестування важливо враховувати різні умови освітлення, фону та розміщення. Для цього було створено три різні тестові випадки.

По-перше, необхідно перевірити, чи чутливий програмний модуль до змін зовнішнього освітлення. У реальному середовищі умови освітлення можуть відрізнятися, наприклад, залежно від часу доби або місця зйомки і можуть суттєво змінити зовнішній вигляд артефактів шкіри на зображеннях. Якщо модуль не може працювати незалежно від різних умов освітлення, його ефективність і надійність

можуть бути обмежені. У таблиці 3.1 описані вимоги тестового випадку впливу навколишнього освітлення, а на рисунку 3.12 представлено результат цього тесту.

Таблиця 3.1 – Тестування впливу навколишнього освітлення

ID тестового випадку	Тест	Очікуваний вхід	Очікуваний результат	Фактичний вхід	Фактичний результат
1.	Вплив зовнішнього освітлення	Затемнене зображення	Правильна класифікація	Затемнене зображення	Правильна класифікація

Клас “Атопічний дерматит” прогнозується правильно з високою ймовірністю, отже, модуль не чутливий до навколишнього освітлення.

Рисунок 3.12 – Класифікація затемненого зображення

У другому тестовому випадку перевірено можливість програмного модуля класифікувати захворювання шкіри із зображення на папері чи цифровому пристрої, які пацієнти можуть надсилати з будь-якого місця, що дозволить дистанційні консультації. Різні умови освітлення можуть спричинити тіні, відблиски або нерівномірне освітлення на папері чи телефоні, що може приховати важливі деталі ураження шкіри. У таблиці 3.2 описані вимоги тестового сценарію

впливу джерела зображення ураження шкіри, а на рисунку 3.13 – його результат.

Таблиця 3.2 – Вплив джерела зображення ураження шкіри

ID тестового випадку	Тест	Очікуваний вхід	Очікуваний результат	Фактичний вхід	Фактичний результат
2.	Вплив джерела зображення ураження шкіри	Зображення з телефону	Правильна класифікація	Зображення з телефону	Правильна класифікація

Клас “Судинне ураження” прогнозується правильно з високою ймовірністю, що означає, що цей програмний модуль не реагує на різні джерела зображення.

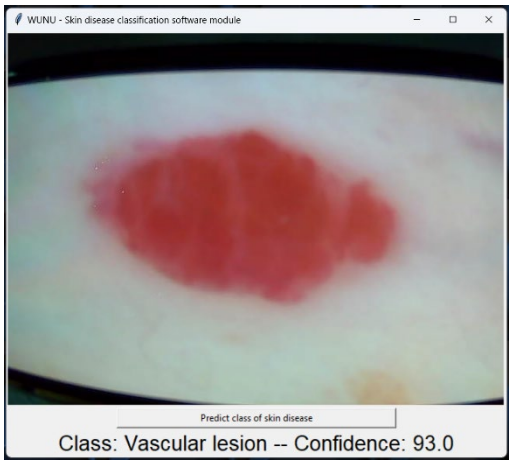

Рисунок 3.13 – Класифікація ураження шкіри з альтернативного джерела зображення

Третій тестовий випадок перевіряє здатність розробленого програмного модуля класифікувати лише пошкодження шкіри людини, а не будь-яку іншу поверхню.

У таблиці 3.3 описані вимоги тестового випадку і йогог результат наведений на рисунку 3.14.

Таблиця 3.3 – Тестування класифікації поверхні, відмінної від шкіри

ID тестового випадку	Тест	Очікуваний вхід	Очікуваний результат	Фактичний вхід	Фактичний результат
3.	Класифікація виключно шкіри	Зображення поверхні столу	Правильна класифікація	Зображення поверхні столу	Правильна класифікація

Повідомлення “Шкіра не виявлена” підтверджує, що програмний модуль реагує лише на зображення шкіри людини.

Рисунок 3.14 – Результат класифікації поверхні, відмінної від шкіри

Деякі незначні помилки були виявлені під час тестування програмного модуля, здебільшого щодо додатка бібліотеки CV2 та взаємодії з камерою. Після перевірки та виправлення цих помилок програмний модуль класифікації шкірних захворювань працює належним чином.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було:

– проаналізовано предметну область;

– проаналізувано існуючі рішення для класифікації шкірних захворювань з використанням штучних нейронних мереж, що дозволило сформулювати вимоги до програмного модуля класифікації захворювань шкіри;

– представлено загальну структуру і розроблено алгоритм програмного модуля класифікації шкірних захворювань;

– розроблено архітектуру згорткової нейронної мережі для класифікації захворювань шкіри за вхідним зображенням веб-камери комп’ютера;

– представлено алгоритм навчання згорткової нейронної мережі для програмного модуля класифікації шкірних захворювань;

– виконано навчання згорткової нейронної мережі з використанням фреймворку глибокого навчання Keras;

– розроблено програмний модуль для класифікації захворювань шкіри в режимі реального часу з використанням Python та бібліотеки комп’ютерного зору;

– протестовано програмний модуль та перевірено правильність класифікації.

Отже, було розроблено програмний модуль, який можна використовувати в телемедицині, наприклад його можна інтегрувати в телемедичні платформи для надання дистанційної діагностики та рекомендацій щодо лікування; у первинній медичній допомозі, лікарі загальної практики можуть використовувати цей модуль, щоб допомогти визначити типові захворювання шкіри під час звичайних відвідувань пацієнтів; в освіті, студенти-медики можуть навчатися на цих моделях, щоб зрозуміти різні шкірні захворювання та їх візуальні прояви [27].

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Yao P., Shen S., Xu M., Liu P., Zhang F., Xing J., & Xu R. X. Single model deep learning on imbalanced small datasets for skin lesion classification. IEEE transactions on medical imaging, 41(5), 2021. P. 1242-1254.

2. Cahan A., James J. C. A learning health care system using computer-aided diagnosis. Journal of medical Internet research 19.3, 2017.

3. Marcos A., Iury A. S. Classification models for skin tumor detection using texture analysis in medical images. Journal of Imaging 6.6, 2020.

4. LeCun Y. et al. Deep learning. 2015. URL:
https://www.nature.com/articles/nature14539

5. Roslan R., Razly I.N., Sabri N., & Ibrahim Z. Evaluation of psoriasis skin disease classification using convolutional neural network. IAES International Journal of Artificial Intelligence, 9, 2020. P. 349-355.

6. Simonyan K., Zisserman A. Very deep convolutional neural networks for large-scale image recognition Network. 2015. URL: https://arxiv.org/abs/1409.1556/

7. Wang Z. J., Turko R., Shaikh O., Park H., Das N., Hohman F., Chau D. H. P. CNN explainer: learning convolutional neural networks with interactive visualization. IEEE Transactions on Visualization and Computer Graphics, 27(2), 2020. P. 1396-1406.

8. Dan C., Meier U., Masci J., Gambardella L.M., Schmidhuber J. Flexible, High Performance Convolutional Neural Networks for Image Classification. Proceedings of the Twenty-Second international joint conference on Artificial Intelligence. Volume Two, 2011. P. 1237–1242.

9. Krizhevsky A., Sutskever I., Hinton G.E. ImageNet classification with deep convolutional neural networks. Commun. ACM 60, 6. June 2017. P. 84–90. URL: https://doi.org/10.1145/3065386

10. Girshick R. "Fast R-CNN," 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile. 2015. p. 1440–1448, doi: 10.1109/ICCV.2015.169.

11. Sagar A. S., Chen Y., Xie Y., & Kim H. S. MSA R-CNN: A comprehensive approach to remote sensing object detection and scene understanding. Expert Systems

with Applications, 241, 122788, 2024.

12. Zhang X., Xu J., Yang J., Chen L., Zhou H., Liu X., Ying Y. Understanding the learning mechanism of convolutional neural networks in spectral analysis. Analytica Chimica Acta, 1119, 2020. P. 41-51.

13. Maweu B. M., Dakshit S., Shamsuddin R., Prabhakaran B. CEFEs: a CNN explainable framework for ECG signals. Artificial Intelligence in Medicine, 115, 102059, 2021.

14. Yuan L., Chen D., Chen Y.L., Codella N., Dai X., Gao J., Zhang P. Florence: A new foundation model for computer vision. 2021. arXiv preprint arXiv:2111.11432.

15. Wei Mingjun & Wu Qiwei & Ji,Hongyu & Wang Jingkun & Lyu Tao & Liu Jinyun & Zhao Li. A Skin Disease Classification Model Based on DenseNet and ConvNeXt Fusion. Electronics. 2023. URL: 12. 438. 10.3390/electronics12020438.

16. Srinivasu P.N.; SivaSai J.G.; Ijaz M.F.; Bhoi A.K.; Kim W.; Kang J.J. Classification of Skin Disease Using Deep Learning Neural Networks with MobileNet V2 and LSTM. Sensors 2021, 21, 2852. https://doi.org/10.3390/s21082852

17. Allugunti V. R. A machine learning model for skin disease classification using convolution neural network. International Journal of Computing, Programming and Database Management, 2022, 3 (1), P.141-147.

18. Voulodimos A., Doulamis N., Doulamis A., Protopapadakis E. Deep Learning for Computer Vision: A Brief Review. Computational Intelligence and Neuroscience, vol. 2018. Article ID 7068349. 2018. URL: https://doi.org/10.1155/2018/7068349.

19. Fang W., Zhong B., Zhao N., Luo H., Xue J., Xu Sh. A Deep Learning-based Approach for Mitigating Falls from Height with Computer Vision: Convolutional Neural Network Accepted Version. Advanced Engineering Informatics. 39. 2019. P. 170-177. 10.1016/j.aei.2018.12.005.

20. Tan M., Le Q. Efficientnet: Rethinking model scaling for convolutional neural networks. In International conference on machine learning. 2019. P. 6105-6114.

21. Keras. URL: https://keras.io/

22. NVIDIA GeForce RTX 3060. URL: https://www.nvidia.com/en-us/geforce/graphics-cards/30-series/rtx-3060-3060ti/

23. Joseph F.J.J., Nonsiri S., Monsakul A. Keras and TensorFlow: A hands-on experience. Advanced Deep Learning for Engineers and Scientists: A Practical Approach. 2021. P. 85-111.

24. Wang Sh., Zhang Y. DenseNet-201-Based Deep Neural Network with Composite Learning Factor and Precomputation for Multiple Sclerosis Classification. ACM Trans. Multimedia Comput. Commun. Appl. 16, 2s, Article 60. 2020. URL: https://doi.org/10.1145/3341095

25. Traore B.B., Kamsu-Foguem B., Tangara F. Deep convolution neural network for image recognition. Ecological informatics, 48. 2018. P. 257-268.

26. Paoletti M.E., Haut J.M., Plaza J., Plaza A. A new deep convolutional neural network for fast hyperspectral image classification. ISPRS journal of photogrammetry and remote sensing, 145. 2018. P. 120-147.

27. Божигора Ю. В., Турченко І.В. Архітектура згорткової нейронної мережі для програмного модуля класифікації захворювань шкіри. Сучасні світові тенденції розвитку науки та інформаційних технологій: матеріали VII міжнародної науково-практичної конференції, м. Одеса, 30-31.05.2024 р. С. 54-56.

28. Skin Disease Classification Dataset. URL: https://www.kaggle.com/datasets/riyaelizashaju/skin-disease-classification-image-dataset

29. Комар М.П., Саченко А.О., Васильків Н.М., Гладій Г.М., Коваль В.С., Ліп’яніна-Гончаренко Х.В. Методичні рекомендації до виконання кваліфікаційної роботи з освітньо-професійної програми «Комп’ютерні науки» спеціальності 122 «Комп’ютерні науки» за першим (бакалаврським) рівнем вищої освіти. Тернопіль: ЗУНУ, 2024. 52 с.