МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інтелектуальних
інформаційних систем, д-р техн. наук, проф.
_____Ю. П. Кондратенко
«____»_____2022 р.

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

ДОСЛІДЖЕННЯ МЕТОДІВ МАШИННОГО ЗОРУ ДЛЯ АВТОМАТИЗОВАНОГО ДІАГНОСТУВАННЯ ХВОРОБ ШКІРИ

Спеціальність 122 «Комп'ютерні науки»

122 – МКР – 601.21610117

Студент _____В. І. Петрович
«__» _____ 2022 р.

Консультант _____Г. В. Кондратенко
к.т.н., доцент
«__» _____ 2022 р.

Миколаїв – 2022

Чорноморський національний університет ім. Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Освітньо-кваліфікаційний рівень **магістр**
Галузь знань **12 «Інформаційні технології»**
(шифр і назва)
Спеціальність **122 «Комп'ютерні науки»**
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри інтелектуальних інформаційних систем, д-р техн. наук, проф.
_____ Ю. П. Кондратенко
«____» _____2021р.

ЗАВДАННЯ
на магістерську кваліфікаційну роботу

Видано студенту групи _601_ факультету комп'ютерних наук

Петровичу Валентину Івановичу.
(прізвище, ім'я, по батькові)

1. Тема магістерської кваліфікаційної роботи Дослідження методів машинного зору для автоматизованого діагностування хвороб шкіри.

Керівник роботи__канд. техн. наук, доцент Галина Володимирівна Кондратенко
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
Затв. наказом Ректора ЧНУ ім. Петра Могили від «___» ____ 20__ р. № _____

2. Строк подання студентом роботи «___» ___ 20__р.

3. Вхідні (початкові) дані до роботи: набір з більше ніж 25 тисяч зображень різних захворювань шкіри

Очікуваний результат роботи: розроблена згорткова нейронна мережа для діагностування захворювання шкіри та точного визначення типу захворювання

4. Зміст пояснювальної записки (перелік питань, які потрібно розглянути): дослідження та аналіз сучасного стану використання систем комп'ютерного зору, головні типи систем класифікацій, архітектурні рішення, розробка відповідної системи

5. Перелік графічних матеріалів __презентація_____
_____

6. Завдання до спеціальної частини: оцінка умов праці та забезпечення безпеки при надзвичайних ситуаціях персоналу ІТ – компанії «Postindustria»_____
_____

7. Консультанти:

Розділ	**Прізвище, ініціали та посада консультанта**	**Підпис**
Спеціальна частина з охорони праці	к.т.н., доцент Ю. Г. Щербак	
Методична частина	к.т.н., доцент Г. В. Кондратенко	

Керівник роботи _____канд. техн. наук, доцент Кондратенко Г.В._____
(наук. ступінь, вчене звання, прізвище та ініціали)

_____
(підпис)

Завдання прийнято до виконання _____Петрович В. І.___
(прізвище та ініціали)

_____
(підпис)

Дата видачі завдання «____» _____ 20__ р.

КАЛЕНДАРНИЙ ПЛАН
Виконання магістерської кваліфікаційної роботи

Тема: Дослідження методів машинного зору для автоматизованого діагностування хвороб шкіри_____.

№	Найменування роботи	Початок	Закінчення	Примітки
1	Визначення керівника і теми МКР. Подання заяви на затвердження теми МКР	01.09.2021	10.10.2021	
2	Отримання завдання на виконання МКР	19.10.2021	22.10.2021	
3	Складання календарного плану на період виконання МКР	23.10.2021	26.10.2021	
4	Огляд літератури за темою дослідження	27.10.2021	10.11.2021	
5	Проходження переддипломної практики, збір та аналіз матеріалів до МКР	22.11.2021	11.12.2021	
6	Аналіз предметної області та розробка технічного завдання. Моделювання результатів	16.12.2021	12.01.2022	
7	Опис фахової частини МКР, зокрема дослідження публікацій щодо класифікації зображень, огляд існуючих алгоритмів.	13.01.2022	25.01.2022	
8	Розробка спеціальної частини з охорони праці та методичної частини	26.01.2022	30.01.2022	
9	Попередній захист МКР на засіданні комісії кафедри	31.01.2022	31.01.2022	
10	Корегування роботи за результатами попереднього захисту	01.02.2022	03.02.2022	
11	Остаточне оформлення пояснювальної записки та слайдів доповіді для захисту	04.02.2022	06.02.2022	
12	Подання МКР рецензенту	09.02.2022	10.02.2022	
13	Рецензування МКР	11.02.2022	12.02.2022	
14	Подання МКР, її електронної копії та інших документів (відгуку, рецензії) до захисту	14.02.2022	15.02.2022	
15	Захист МКР перед екзаменаційною комісією (ЕК)	21.02.2022	22.02.2022	

Розробив студент __Петрович Валентин Іванович_____ _____

(прізвище та ініціали) (*підпис*)

Керівник роботи канд. техн. наук, доцент Кондратенко Г. В, _____

(наук. ступінь, вчене звання, прізвище та ініціали) *(підпис)*

*«*23*»* жовтня 2021 р.

АНОТАЦІЯ

до магістерської кваліфікаційної роботи
студента групи 601 ЧНУ ім. Петра Могили

Петровича Валентина Івановича

на тему: **“ДОСЛІДЖЕННЯ МЕТОДІВ МАШИННОГО ЗОРУ ДЛЯ АВТОМАТИЗОВАНОГО ДІАГНОСТУВАННЯ ХВОРОБ ШКІРИ”**

Об’єкт дослідження – методи машинного зору.

Предмет дослідження – штучні нейронні мережі для автоматизованого діагностування хвороб шкіри.

Мета роботи – діагностування захворювань шкіри на основі розроблених згорткових мереж різних архітектур з дослідженням впливу параметрів налаштування нейронної мережі на точність прогнозування (діагностування).

Магістерська робота складається з фахового розділу та спеціальних частин: розділ охорони праці та методичний розділ. Пояснювальна записка складається зі вступу, п’яти розділів та висновків.

У першому розділі проводиться аналіз сучасного стану задач автоматизованого діагностування хвороб шкіри. Виконується постановка проблеми та огляд предметної області. Другий розділ розкриває тему налаштування середовища розробки для роботи з обраним набором даних. Описано використані програмні застосунки та бібліотеки, які використовуються в роботі. У третьому розділі описуються результати досліджень набору даних та попередня обробка зображень. Виконано опис основних характеристик. Застосовано користувацьке масштабування для зміни розміру зображення. Використано формат даних HDF5. Зроблено розподіл даних на для тренування та для тестування та генерація додаткових зображень. Четвертий розділ описує процес навчання моделей різних архітектур. А саме використовуються користувацькі модифікації таких моделей: MiniVGG, AlexNet, VGG16. Для кожної архітектури застосовується підбір оптимальних параметрів навчання. У п’ятому розділі проведено дослідження різних параметрів навчання на точність навчання.

В результаті роботи створено різні архітектури згорткових нейронних мереж для вирішення проблеми діагностування хвороб шкіри. Найкращий з досягнутих результатів – 68.52% точності на тестовому наборі.

Магістерська робота містить __ сторінок, __ рисунків, __ джерел та __ додатків.

Ключові слова: згорткові нейроні мережі, машинний зір, хвороби шкіри, автоматизоване діагностування.

ABSTRACT

to the master's qualification work
student of group 601 of Petro Mohyla National University

Petrovych Valentyn

The object of research is the methods of machine vision.

The subject of research - artificial neural networks for automated diagnosis of skin diseases.

The aim of the work is to diagnose skin diseases on the basis of developed convolutional networks of different architectures with the study of the influence of neural network settings on the accuracy of forecasting (diagnosis).

The master's thesis consists of a professional section and special parts: the section of labor protection and methodical section. The explanatory note consists of an introduction, five chapters and conclusions.

The first section analyzes the current state of tasks of automated diagnosis of skin diseases. The problem statement and review of the subject area is performed. The second section describes how to configure the development environment to work with the selected dataset. Describes used software applications and libraries used in the work. The third section describes the results of data set research and image pre-processing. The description of the main characteristics is executed. Custom scaling is used to resize the image. HDF5 data format used. The distribution of data for training and for testing and generation of additional images is made. The fourth section describes the process of learning models of different architectures. Namely, custom modifications of the following models are used: MiniVGG, AlexNet, VGG16. The selection of optimal learning parameters is used for each architecture. The fifth section examines the various learning parameters for learning accuracy.

As a result, various architectures of convolutional neural networks have been created to solve the problem of diagnosing skin diseases. The best of the achieved results is 68.52% accuracy on the test set.

The master's thesis contains __ pages, __ drawings, __ sources and __ appendices.

Key words: convolutional neural networks, machine vision, skin diseases, automated diagnosis.

ЗМІСТ

Пояснювальна записка

до магістерської кваліфікаційної роботи

на тему:

ДОСЛІДЖЕННЯ МЕТОДІВ МАШИННОГО ЗОРУ ДЛЯ АВТОМАТИЗОВАНОГО ДІАГНОСТУВАННЯ ХВОРОБ ШКІРИ

Спеціальність 122 «Комп’ютерні науки»

122 – МКР – 601.21610117

Студент _____ В. І. Петрович
«__» _____ 2022 р.

Консультант _____ Г. В. Кондратенко
к.т.н., доцент
«__» _____ 2022 р.

Миколаїв – 2022

ВСТУП

Питання здоров’я та медицини завжди відігравали ключову роль в житті людей. Здоров’я – основа для щасливого життя людей. Це питання є актуальним навіть у розвинутих країнах, оскільки існує багато різноманітних хвороб, які тяжко виявити. Тому тема медицини є доволі актуальної і на сьогодні.

З поширенням методів штучного інтелекту їх почали впроваджувати і в сферу медицини. Деякі питання вже були вирішені: електроні картки пацієнтів, хірургічні роботи, автоматизовані системи діагностування. Але саме зі збільшенням обчислювальної потужності та використанням методів машинного навчання з'явилися нові методи вирішення багатьох проблем, які раніше було складно або неможливо розв'язати.

Найбільш поширений метод машинного навчання – нейроні мережі. Саме його найчастіше використовують в розв'язані сучасних проблем. Хоча технологія вже довгий час існує, вона залишається актуальною та ефективною. Незважаючи на це її методи постійно вдосконалюються та знаходять більш ефективні методи. Складність полягає в величезній кількості параметрів під час налаштуванні під конкретну задачу. Але висока еластичність та швидкість роботи роблять нейроні мережі дуже зручним інструментом.

А з появою згорткових нейронних мереж почався надзвичайно активний розвиток напрямку комп’ютерного зору. Згорткові мережі допомагають розбивати зображення на маленькі деталі і досліджувати їх особливості. А отримання якоїсь інформації з зображень або відео – дуже актуальна технологія, яка буде актуальною ще багато років.

Поєднуючи нейроні мережі та тему медицини ми отримуємо актуальну та суспільно-корисну роботу. Оскільки вона використовує останні дослідження в сфері нейромережевих технологій і комп’ютерного зору та має можливість використання в сучасному світі та навіть рятувати життя людей.

Тому згідно з актуальністю теми медицини та штучного інтелекту **об'єктом дослідження** обрано методи машинного зору, як найбільш популярний та

прогресивний напрям штучного інтелекту. **Предметом дослідження** є штучні нейронні мережі для автоматизованого діагностування хвороб шкіри.

Мета роботи – діагностування захворювань шкіри на основі розроблених згорткових мереж різних архітектур з дослідженням впливу параметрів налаштування нейронної мережі на точність прогнозування (діагностування).

Для досягнення поставленої мети необхідно вирішити наступні завдання:

- проаналізувати сучасний стан задачі діагностування захворювань шкіри;
- описати існуючі технології для вирішення поставленої задачі;
- реалізація штучної нейронної мережі та її налаштування для діагностування захворювань шкіри з використанням різних архітектур;
- навчання нейронної мережі, використовуючи різні підходи;
- порівняльний аналіз навчених моделей;
- дослідження впливу різних параметрів на навчання.

1 АНАЛІЗ СУЧАСНОГО СТАНУ ЗАДАЧІ АВТОМАТИЗОВАНОГО ДІАГНОСТУВАННЯ ХВОРОБ ШКІРИ

1.1 Постановка проблеми

Основне призначення комп’ютерів – полегшення життя людей. Поширення комп'ютерів в усіх сферах життя продовжується навіть зараз. Найбільш гостро постає питання використання комп'ютерів в медицині, адже це найголовніше питання для усього сучасного світу. Прикладом такого використання є електроні картки пацієнтів, автоматизовані системи діагностування захворювань по симптомам в смартфоні, сканування зображень для виявлення пухлин. Подібні системи вже використовуються та вдосконалюються для більш точних результатів та швидшої роботи. Отже питання здоров’я було, є і буде актуальним завжди.

З ростом обчислювальної потужності росте й складність завдань, які поставлені комп'ютеру. Те що людина буде обчислювати годинами, а можливо й днями, процесор може розрахувати за декілька секунд. Згідно з цим подальша автоматизація медицини – це лише питання часу.

Але автоматизація медицини має і негативні сторони. Процесор виконує лише команди, які записані в нього. Він може помилятися, тому що людина, яка його програмувала, може помилитися. І помилка може привести до погіршення стану пацієнту. Безсумнівно обчислювальні машини допомагають в медицині, але вони являються лиш засобом підтримки прийняття рішень і останнє слово завжди за лікарем. Комп'ютери не можуть усунути потребу в людині.

Раніше для вирішення подібних систем не вистачало обчислювальної потужності. Але разом з ростом тактових частот та технологій відкриваються нові можливості. Саме з появою більш швидких комп’ютерів стало можливим використання методів машинного навчання.

Складності для людини в медицині базуються на тому, що людина не має безмежної пам'яті та не може обробляти велику кількість інформації за один раз на відміну від комп'ютера. Наприклад симптоми у деяких вірусів можуть бути

схожими, а кількість аналізів може бути занадто велика, щоб зробити відразу висновок та відрізнити віруси. Передбачення хвороби та забезпечення здоров'я людини перша і найважливіша мета для кожної країни. Тому тема медицини з використанням інтелектуальних систем була і буде актуальна ще багато років.

Найбільш поширений метод машинного навчання – нейроні мережі. Нейроні мережі – метод штучного інтелекту, який являється прямою імітацією нервової системи людини, а саме роботи мозку. Саме такий метод допомагає вирішити багато задач, які раніше були недоступні. Мережі штучного інтелекту представляють нелінійні системи, що дозволяють набагато краще класифікувати дані, ніж звичайні лінійні методи. Завдяки ним в медицині сталася революція в сфері діагностування хвороб серця, зору, печінки та навіть злоякісних ракових пухлин [1]. Саме його найчастіше використовують в розв'язані сучасних проблем. Але оскільки технологія відносно нова, її постійно вдосконалюють та шукають більш ефективні методи. Складність полягає в величезній кількості параметрів під час налаштування під конкретну задачу. Але висока еластичність та швидкість роботи роблять нейроні мережі дуже зручним інструментом.

Штучні нейроні мережі надають важливі особливості програмам. Перша особливість надає змогу не тримати величезну базу даних в пам'яті, лише матрицю ваг. Це можливо тільки після навчання мережі, однак дає величезні переваги, навіть можуть бути перенесені в вбудовані системи розумного дому. Друга – самонавчання на даних, нейрона мережа сама адаптується на умови, які ставить їй людина, при правильній підготовці надається можливість класифікувати навіть рідкісні випадки. Третя – швидкість роботи, якщо мережа вже навчена. Не потрібно робити порівняння даних з величезними базами.

Нейроні мережі дають величезні можливості в відтворені моделі мозку живих істот. Він являється вищою формою розумного, саме тому його і хочуть відтворити. Як приклад мозок бджоли має 1 мільйон нейронів [2], відтворити таку нейрону мережу можуть сучасні комп'ютери. Однак для відтворення роботи мозку людини потрібні дуже потужні суперкомп'ютери. Як приклад в 2016 році була

розроблена мережа на 160 мільярдів нейронів [3], це навіть більше ніж у Google та Національної лабораторії США, в них 11.2 та 15 мільярдів відповідно. Ця нейронна мережа була розроблена фірмою Digital Reasoning у США, яка займається когнітивними дослідженнями. Вважається, що саме завдяки дослідженням штучних нейронних мереж, можна повністю зрозуміти роботу мозку.

Нейронні мережі вже обходять людину в вирішені багатьох задач. Існує тільки одна проблема. Вони виконують задані інструкції та заточені під конкретну задачу. На відміну від мозку людини. Початком публічного резонансу вважається гра Каспарова проти комп'ютера Deep Blue в шахи [4]. За цією подією слідкував увесь світ: вчені, програмісти, шахісти та просто зацікавлені люди. Першу гру (1996 р.) комп’ютер програв, але через рік все змінилося, та комп’ютер виграв. Тим самим творці суперкомп'ютеру показали наскільки швидко розвиваються комп'ютерні технології.

Хоча нейроні мережі отримали популярність майже у всіх галузях сучасного життя. Найбільшу увагу все ж таки приділяють провідним проблемам суспільства. Одна з найголовніших проблем – хвороби та їх діагностування. В наш час дуже гостро постає проблема шкірних захворювань, тому була обрана тема діагностування хвороб шкіри. Головна мета – навчити модель класифікувати зображення захворювань шкіри. Для цього потрібно навчити різні моделі та дослідити результати.

Поєднуючи нейроні мережі та тему медицини ми отримуємо актуальну та суспільно-корисну роботу. Оскільки вона використовує останні дослідження в сфері нейромережевих технологій та має можливість використання в сучасному світі та навіть рятувати життя людей.

1.2 Аналіз досліджень та публікацій

Комп'ютери використовуються в медицині майже у всіх областях, будь то діагностика, лікування, дослідження чи управління даними. Не дивлячись на

важливість кожного процесу діагностика завжди відігравала провідну роль. Оскільки без визначення конкретної хвороби неможливо почати ефективне лікування. В сучасному світі завдяки великій обчислювальній потужності персональних комп'ютерів можливо з вражаючою точністю сказати чи хвора людина, чи ні. Сучасна діагностика складається з 3 основних етапів: отримання історії хвороб, огляд хворого фактично та проведення лабораторних досліджень і усі ці етапи залучають до роботи обчислювальні можливості процесорів. І по кожному напрямку доволі багато публікацій та досліджень.

А тема захворювань шкіри дуже популярна, оскільки з 2016 року International Skin Imaging Collaboration (ISIC) [5] проводять щорічне випробування по методам штучного інтелекту для діагностування різних хвороб шкіри, де головна мета команд – досягнення найкращої точності діагностування.

За цим напрямом проводяться найрізноманітніші дослідження і їх вже накопилося доволі багато, тому в аналізі *будуть описані найновіші*.

Оскільки в датасетах ISIC зазвичай не рівномірна кількість даних для кожного класу, то потрібно використовувати різноманітні методи генерації даних. Одна з таких робіт з використанням генеративно змагальної мережі (GAN) – публікація Beynek, Bora, Evren, Ugur[6]. Вони змогли налаштувати мережу для створення нових даних та провели тест Тюрінга з медичними працівниками на те, чи зможуть вони відрізнити штучне зображення від справжнього.

Cassidya, Kendricka, Brodzickib, Jaworek-Korjakowska та Hoon Yapa [7] провели дослідження датасетів за усі роки. Вони знайшли дублікати в наборах даних для тестування та для навчання. Також привели багато порад для досягнення кращих результатів.

Також багато досліджень на проведення сегментації пухлин на зображеннях. Одним з таких є публікація Yang, Ren, Huang, Chuang, Chang[8]**.** Вони змогли покращити сегментацію попередніх дослідників та досягти кращих результатів в сегментації.

Ще один популярним метод дослідження – використання масок для класифікації лише певної частини зображення. Це доволі популярна техніка і вона використовується в багатьох роботах. Одна з крайніх – робота Reshma1, Al-Atroshi, Kumar Nassa, Geetha, Sunitha, Gouse Galety and Neelakandan[9]. Також маски використовуються в роботі Ünlü, Çınar[10]. Вони використовують модифікації популярних згорткових мереж для досягнення кращих результатів.

Інша група вчених розробляють додаток для iOS для виявлення злоякісних утворень шкіри. Додаток називається Melatec. Він описаний в роботі Meel, Bodepudi [11]. Оскільки даних доволі мало, то вони використовували додаткову генерацію використовуючи DeepAugment.

1.3 Огляд предметної області та постановка задачі

Предметна область включає в себе методи класифікації зображень. Основний керуючий метод усієї класифікації – згорткові нейроні мережі. Саме завдяки ним можна проходити все зображення для пошуку певних шаблонів. Але головне в згорткових нейронних мережах – архітектура. Саме вона вирішує ефективність класифікації. Адже різні архітектури мають різні результати, через те, що деякі мережі адаптовані на пошук маленьких шаблонів, інші на пошук великих, деякі використовуються для визначення 2 класів, інші для 1000 різних класів. Тобто основна мета в предметній області – визначення найбільш ефективної мережі.

Так само як і в штучних нейронних мережах, важливим фактором є корегування параметрів, таких як коефіцієнт навчання. Треба експериментально досліджувати вплив кожного параметру та знаходити оптимальний.

Через складність задачі, а саме нерівномірного розподілу класів, потрібно використовувати генерування додаткових даних. Разом з цим потрібно використовувати методи роботи з великими даними, які не можуть поміститися в оперативній пам’яті. Робота з форматом HDF5.

Поєднуючи нейроні мережі та тему медицини ми отримуємо актуальну та суспільно-корисну роботу. Оскільки вона використовує останні дослідження в сфері нейромережевих технологій і комп'ютерного зору та має можливість використання в сучасному світі та навіть рятувати життя людей.

Тому згідно з актуальністю теми медицини та штучного інтелекту **об'єктом дослідження** обрано методи машинного зору, як найбільш популярний та прогресивний напрям штучного інтелекту. **Предметом дослідження** є штучні нейронні мережі для автоматизованого діагностування хвороб шкіри.

Мета роботи – діагностування захворювань шкіри на основі розроблених згорткових мереж різних архітектур з дослідженням впливу параметрів налаштування нейронної мережі на точність прогнозування (діагностування).

Для досягнення поставленої мети необхідно вирішити наступні завдання:

- проаналізувати сучасний стан задачі діагностування захворювань шкіри;
- описати існуючі технології для вирішення поставленої задачі;
- реалізація штучної нейронної мережі та її налаштування для діагностування захворювань шкіри з використанням різних архітектур;
- навчання нейронної мережі, використовуючи різні підходи;
- порівняльний аналіз навчених моделей;
- дослідження впливу різних параметрів на навчання.

Висновки до розділу 1

Було виконано дослідження з використання штучних нейронних мереж в медицині. Описано цілі та завдання, які потрібно вирішити. Проведено аналіз досліджень та публікацій по темі виявлення захворювань шкіри. Виявлено основні тенденції розвитку в цьому питані.

Досліджено предметну область та технологій, з якими буде потрібно працювати для вирішення завдання.

2 НАЛАШТУВАННЯ СЕРЕДОВИЩА РОЗРОБКИ, БІБЛІОТЕК ШТУЧНОГО ІНТЕЛЕКТУ ТА ОБРОБКИ ЗОБРАЖЕНЬ

2.1 Середовище розробки

Різних серед розробки штучного інтелекту та мов програмування для них існує доволі багато. Найчастіше використовується Python, R, Matlab згідно дослідження [12]. Але бібліотеки машинного навчання існують для усіх найпопулярніших мов програмування.

Однак для задач комп'ютерного зору найбільше підходить саме Python [13]. Він має найпотужніші бібліотеки для обробки та роботи з зображеннями. Вони оптимізовані для максимально швидкої роботи. Оскільки зображень в наборі даних більше 25 тисяч, то оптимізація та швидкість роботи мають дуже важливе значення.

Python (див. рис. 2.1) — це інтерпретована мова програмування загального рівня високого рівня. Його філософія дизайну підкреслює читабельність коду з використанням значних відступів. Його конструкції, а також його об'єктно-орієнтований підхід спрямовані на те, щоб допомогти програмістам писати зрозумілий, логічний код для малих і великомасштабних проектів.

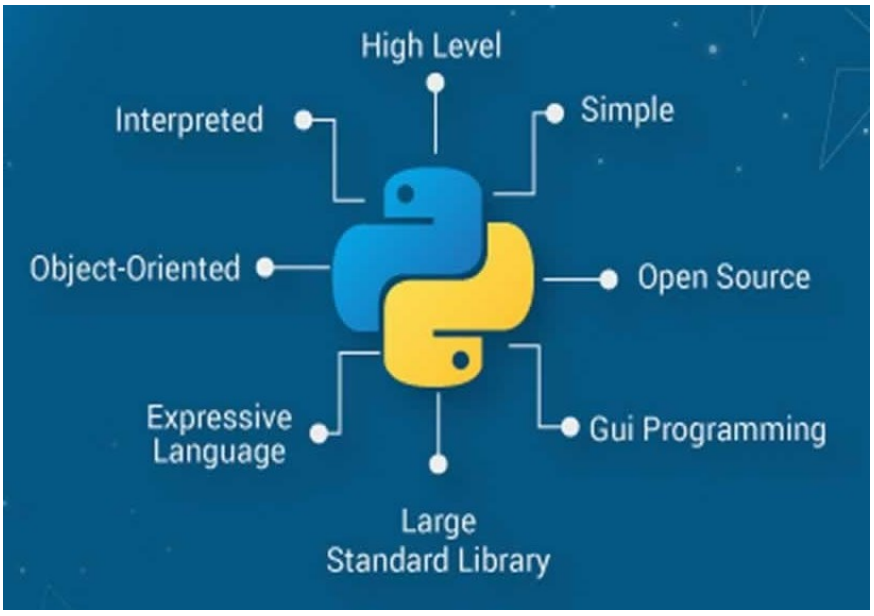

Рис. 2.1. Основні характеристики

Розробив цю мову Гвідо ван Россум наприкінці 1980-их років. Однак свою популярність мова отримала у 2010-их роках, коли тема штучного інтелекту набувала все більшої популярності.

Python підтримує модулі та пакунки, що сприяє модульності програм і повторному використанню коду. Інтерпретатор Python і велика стандартна бібліотека являються open-source продуктами і можуть вільно поширюватися.

Для більш зручного доступу до даних, можливості виводити графіки та використовувати повторно код використовується розширення для Python – **Jupyter.** Воно надає зручний графічний інтерфейс для роботи з кодом та даними.

Jupyter (див. рис. 2.2) [14] — це новітнє веб-інтерактивне середовище розробки коду та дослідження даних в спеціальних блокнотах. Його гнучкий інтерфейс дозволяє користувачам налаштовувати та організовувати робочі процеси в галузі науки про дані, наукових обчислень, обчислювальної журналістики та машинного навчання. Модульний дизайн пропонує розширення для розширення та збагачення функціональності.

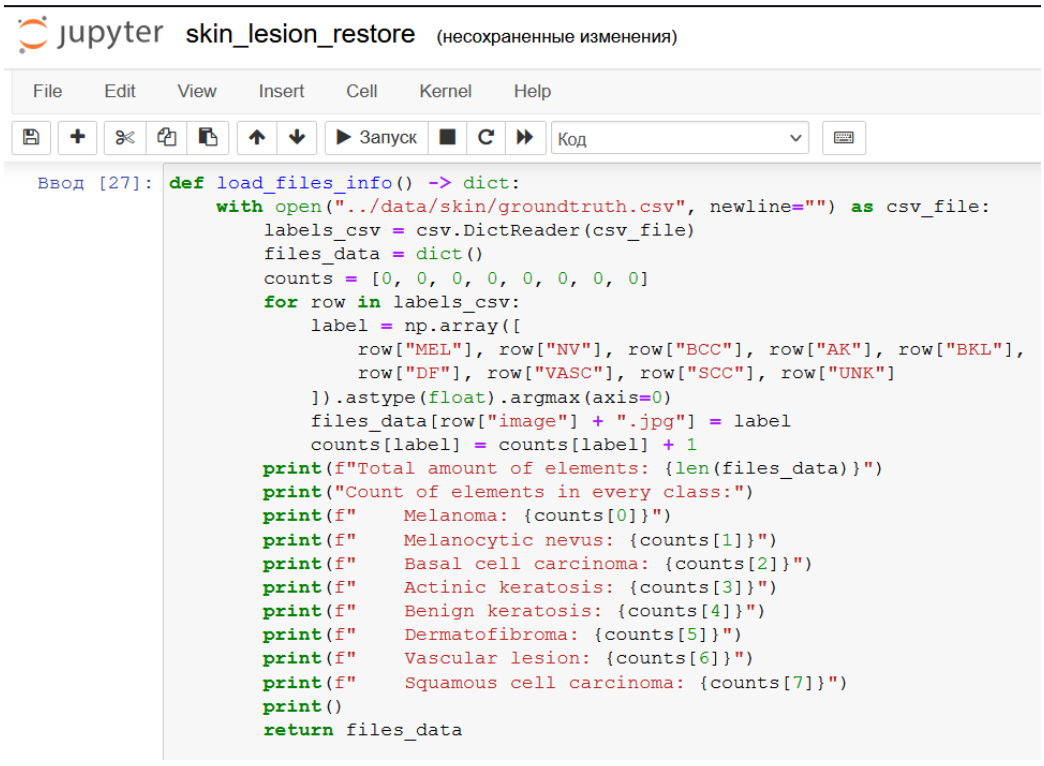

Рис. 2.2. Зовнішній вигляд блокноту в Jupyter

Також для кращого зв'язування Python та Jupyter використовуємо IDE **PyCharm**[15]. Це розробка фірми Jetbrains, яка являється найбільш популярною середою розробки на мові Python.

PyCharm доступна як кросплатформна програма. Вона сумісна з платформами Linux, macOS та Windows. Постачається з безліччю модулів, пакетів та інструментів для прискорення розробки Python, одночасно скорочуючи зусилля, необхідні для того, щоб зробити те ж саме. Крім того, в ній можна налаштувати усі параметри відповідно до вимог розробки та особистих уподобань.

Застосунок допомагає аналізувати ваші дані за допомогою Python. Просто потрібно створити науковий проект, додайте свої дані та можна досліджувати дані. Крім того остання версія надає змогу працювати с Jupyter блокнотами та досліджувати дані під час виконання. Також стало можливим робити графіки даних без будь-якого програмування та зовнішніх бібліотек. Для цього потрібно всього лиш відкрити Jupyter зміні та натиснути «побудувати графік». Див. рис. 2.3.

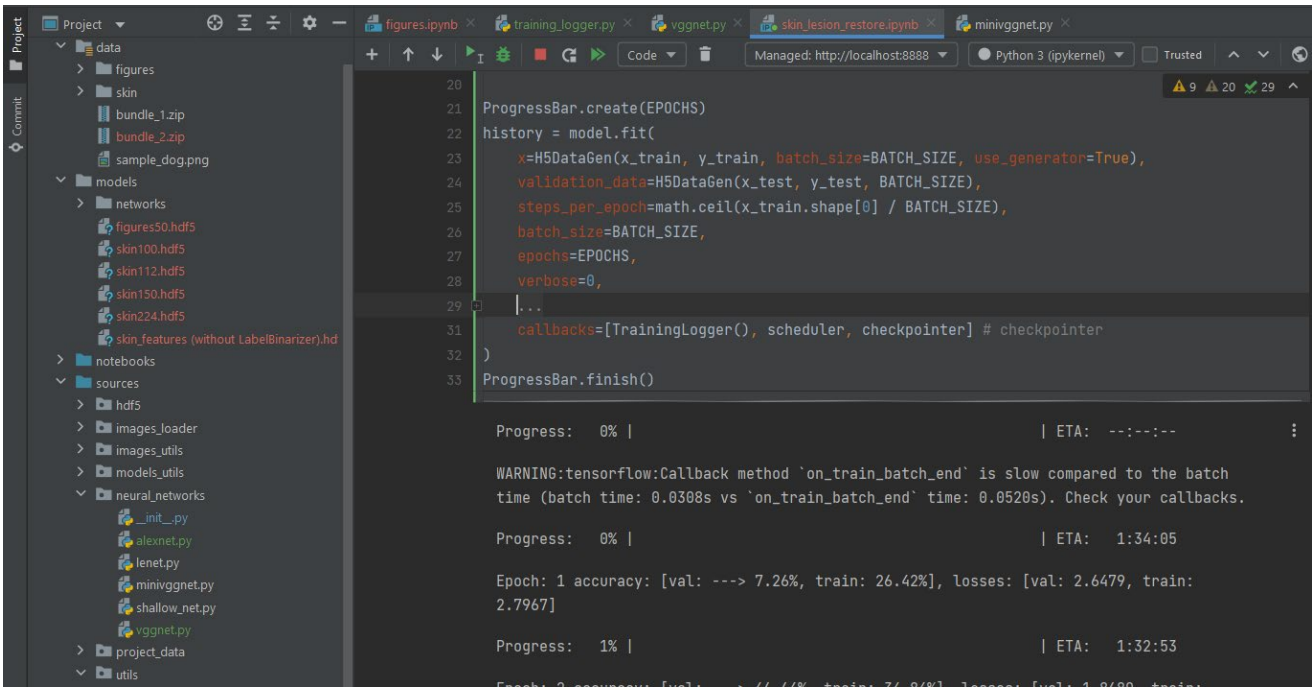

Рис. 2.3. Зовнішній вигляд робочої середовища Jupyter

2.2 Бібліотеки для роботи з зображеннями та штучним інтелектом

NumPy (див. рис. 2.4) [16] (скорочено від Numerical Python) — бібліотека з відкритим вихідним кодом мови програмування Python. Основне призначення – багатовимірні масиви (включаючи матриці) та математичні функції, призначені для роботи з багатовимірними масивами.

Основні характеристики бібліотеки:

- потужний та універсальний функціонал для будь-яких n-вимірних масивів будь якого формату даних;
- обчислювальні цифрові інструменти: комплексні математичні функції, генератори випадкових чисел, підпрограми лінійної алгебри, перетворення Фур'є та інші;
- сумісність з широким спектром апаратних та обчислювальних платформ;
- оптимізований для паралельного виконання на багатьох ядрах;
- простий в використані.

Служить основою для бібліотек у багатьох напрямках науки. Являється основою для будь-яких обчислень на Python через максимальну оптимізацію.

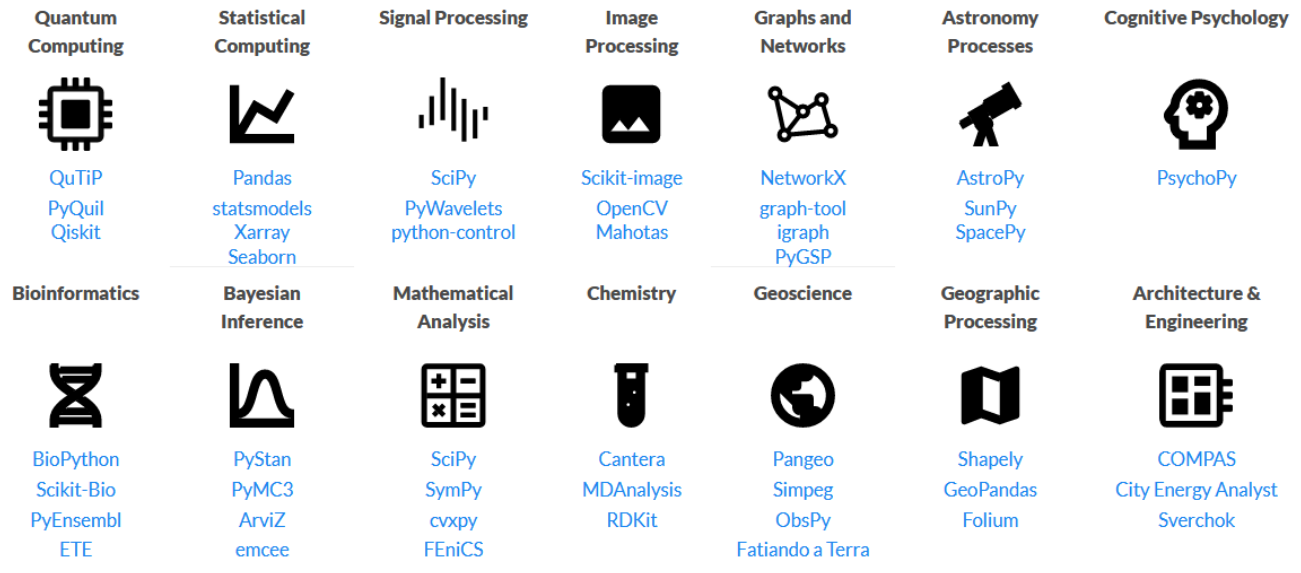

Рис. 2.4. Сфери використання NumPy

Sklearn [17] – бібліотека для підготовки даних. Має прості та оптимізовані інструменти для аналізу даних. Завдяки великій кількості параметрів робить ці інструменти придатними для великого спектру задач. Вона є стандартом в підготовці даних, незалежно від поставленої задачі, чи то задача кластеризації, чи задача комп'ютерного зору. Scikit-learn в основному написаний на Python і широко використовує NumPy для високопродуктивної лінійної алгебри та операцій з масивами. Крім того, деякі основні алгоритми написані на Cython для підвищення продуктивності.

Включає в себе алгоритми та методи для:

- класифікації;
- регресії;
- кластеризації;
- багатовимірного відображення;
- оптимізації моделей.

Але в нашому випадку в основному використовується для обробки та аналіз даних. Саме ця бібліотека використовується на початковому етапі роботі з набором даних – етапі дослідження та обробки даних.

OpenCV (див. рис. 2.5) [18] – бібліотека з відкритим кодом, яка включає кілька сотень алгоритмів комп'ютерного зору.

Він має модульну структуру, доступні такі модулі:

- core (центральна функціональність) – компактний модуль, що визначає основні структури даних, включаючи щільний багатовимірний масив Mat і основні функції, які використовуються всіма іншими модулями;
- imgproc – модуль обробки зображень, що включає лінійну та нелінійну фільтрацію зображень, геометричні перетворення зображення (зміна розміру, спотворення спорідненої та перспективи, загальне перемалювання на основі таблиці), перетворення колірного простору, гістограми тощо;
- video – модуль аналізу відео, який включає в себе алгоритми оцінки руху, віднімання фону та відстеження об'єктів;

– calib3d – основні алгоритми геометрії кількох переглядів, калібрування однієї та стереокамери, оцінка пози об'єкта, алгоритми стереовідповідності та елементи тривимірної реконструкції;

– features2d – визначні детектори ознак, дескриптори та відповідники дескрипторів;

– objdetect – виявлення об'єктів і екземплярів попередньо визначених класів (наприклад, обличчя, очі, гуртки, люди, автомобілі тощо);

– highgui – простий у використанні інтерфейс для простих можливостей інтерфейсу користувача;

– videoio – простий у використанні інтерфейс для запису відео та відеокодеків.

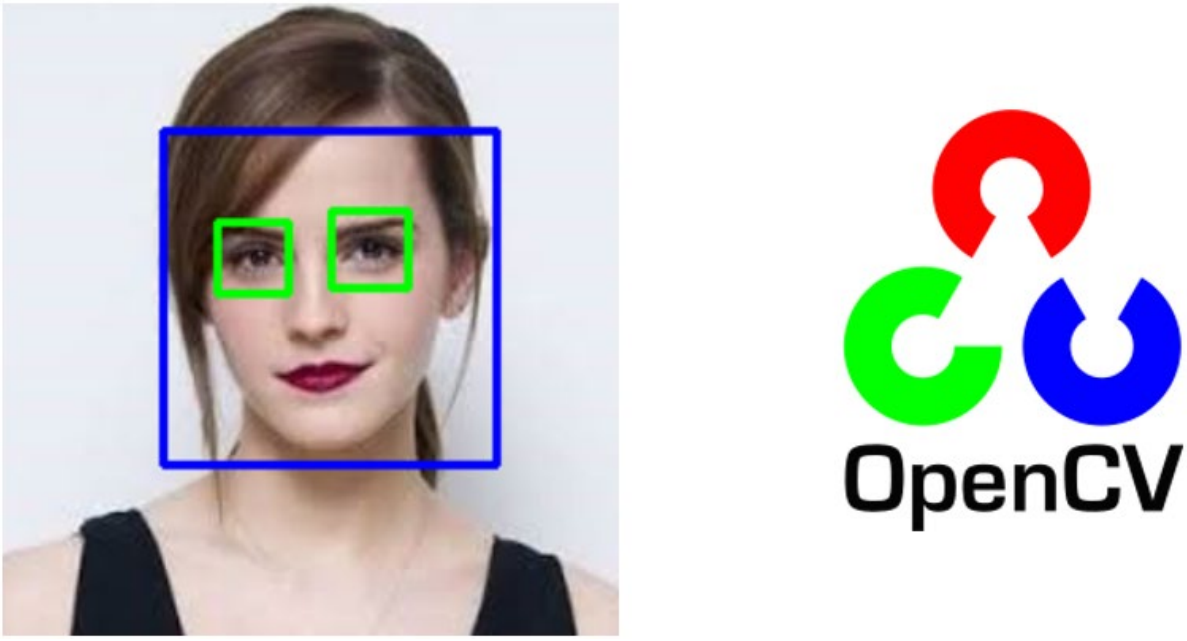

Рис. 2.5. Приклад використання OpenCV

Tensorflow [19] (разом з **Keras**[20]) – це безкоштовна бібліотека програмного забезпечення з відкритим вихідним кодом для машинного навчання та штучного інтелекту. Його можна використовувати для виконання ряду завдань, але він зосереджений на навчанні та використані глибоких нейронних мереж.

TensorFlow був розроблений командою Google Brain для внутрішнього використання Google у дослідженнях і виробництві. Початкова версія була випущена під ліцензією Apache License 2.0 у 2015 році. Google випустила оновлену версію TensorFlow під назвою TensorFlow 2.0 у вересні 2019 року.

Особливості TensorFlow:

– автодиференціація — це процес автоматичного обчислення вектору градієнта моделі щодо кожного з її параметрів, завдяки якій вдається досягати максимальної оптимізації;

– розумний розподіл на різні девайси з різними стратегіями поширення. Ці розподілені обчислення часто можуть прискорити виконання навчання та оцінки моделей TensorFlow і є звичайною практикою в області штучного інтелекту;

– функції втрат. TensorFlow надає набір функцій втрат. Для різних наборів даних і моделей різні втрати використовуються для визначення пріоритету певних аспектів продуктивності;

– різноманітні показники. TensorFlow надає API доступ до часто використовуваних показників. Приклади включають різні показники точності (двійкові, категоріальні, розріджені категоріальні) разом з іншими показниками, такими як точність, відкликання та перетин.

– TF.nn – модуль для налаштування власних моделей. Деякі з цих шарів включають варіації згорток (1/2/3D), функції активації (Softmax, RELU, GELU, Sigmoid тощо);

– оптимізатори. TensorFlow пропонує набір оптимізаторів для навчання нейронних мереж, включаючи ADAM, ADAGRAD і SGD. А еластичні параметри їх налаштування дають змогу покращити показник збіжності.

Складові частини зображені на рисунку 2.6.

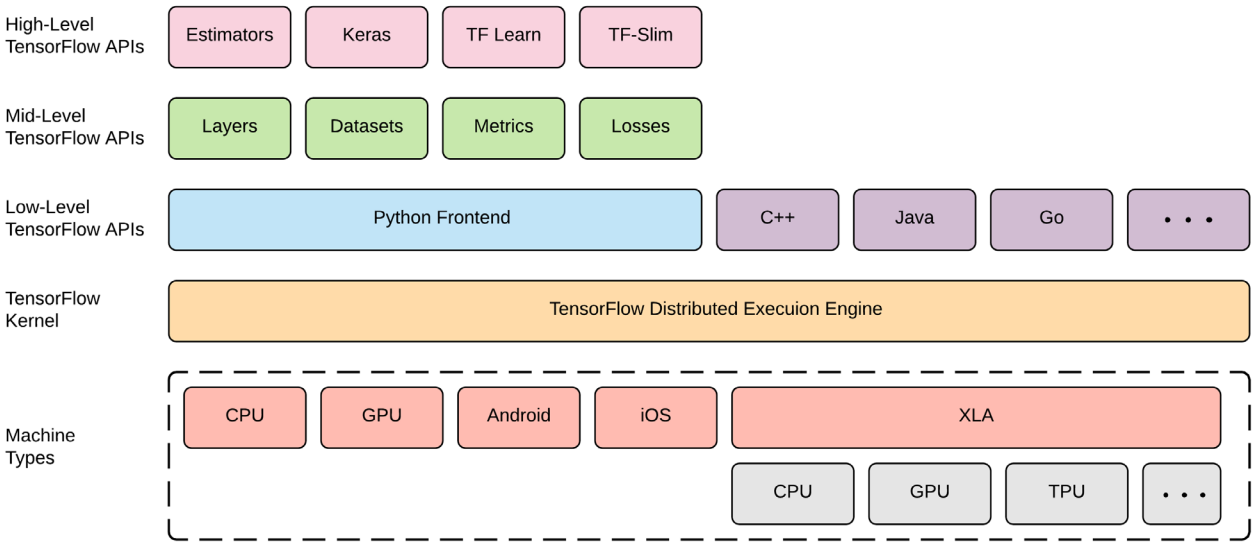

Рис. 2.6. Складові частини TensorFlow

Висновки до розділу 2

Проведено завантаження та налаштування усіх потрібних інструментів для навчання моделей на обраному наборі даних. Виконано аналіз потрібних бібліотек та придатної мови програмування для заданої задачі. Налаштовано середу розробки та описано основні характеристики таких бібліотек як: NumPy, Sklearn, OpenCV, TensorFlow.

3 ДОСЛІДЖЕННЯ ТА ОБРОБКА НАБОРУ ДАНИХ

Індивідуальне завдання полягає в дослідженні набору даних для подальшої правильної обробки та передачі в модель. Дослідження та обробка датасету перед використанням дуже важливий етап будь якої класифікації. Адже завдяки правильній обробці вдається досягти набагато кращих результатів.

3.1 ISIC Challenge

В роботі використовується набір даних International Skin Imaging Collaboration (ISIC) [5] challenge за 2019 рік. Починаючи з 2016 року, ISIC спонсорує щорічні виклики для комп'ютерної спільноти разом із провідними конференціями комп'ютерного зору. З роками проблеми зросли в масштабі, складності та участі, використовуючи високоякісні перевірені людиною навчальні та тестові набори з тисяч ліцензованих зображень і метаданих. Попередні проблеми були зосереджені насамперед на діагностичній точності для відрізнення меланоми від інших доброякісних і злоякісних уражень шкіри. До 2018 року діагностична продуктивність провідних алгоритмів стабільно перевершувала клініцистів у «дослідженні читачів». Додаткові завдання у 2019 та 2020 роках були розроблені для вирішення проблеми поза розповсюдженням та оцінки впливу клінічного контексту відповідно. У конкурсі 2020 року взяли участь 3300 учасників з усього світу. На додаток до щорічних Grand Challenges, ISIC проводить "живі випробування", які дозволяють дослідникам і студентам постійно порівнювати продуктивність своїх алгоритмів за допомогою зображень ISIC.

Усі зображення, що надходять до архіву ISIC, перевіряються як на конфіденційність, так і на гарантію якості. Більшість зображень мають пов'язані клінічні метадані, які були перевірені визнаними експертами з меланоми. Частина зображень була анотована та розмічена визнаними експертами з раку шкіри. Ці позначки включають дермаскопічні ознаки (тобто морфологічні елементи на зображенні, які, як відомо, розрізняють типи уражень шкіри).

3.2 Характеристики набору даних

Набір даних [21] складається с 25.331 зображення. Усі зображення виділяються по своїм кольоровим та розмірним характеристикам. Відразу можна побачити, що зображення були зроблені різним технічним обладнанням. В одних зображеннях кольори більш натуральні, в інших буває занадто великий баланс білого. Також можна побачити відмінності в загальній чіткості. Адже є як чіткі так і розмиті варіанти зображень.

Загалом в наборі даних 8 класів захворювань:

- меланома;
- меланоцитарний невус;
- базаліома;
- актинічний кератоз;
- доброякісний кератоз (сонячне лентиго / себорейний кератоз / кератоз, подібний до плоского лишаю);
- дерматофіброма;
- ураження судин;
- плоскоклітинний рак;
- жоден з вищевказаних.

Останній клас в нашому випадку не використовується. Адже він не має жодного зображення. Приклад зображень можна побачити на рисунку 3.1.

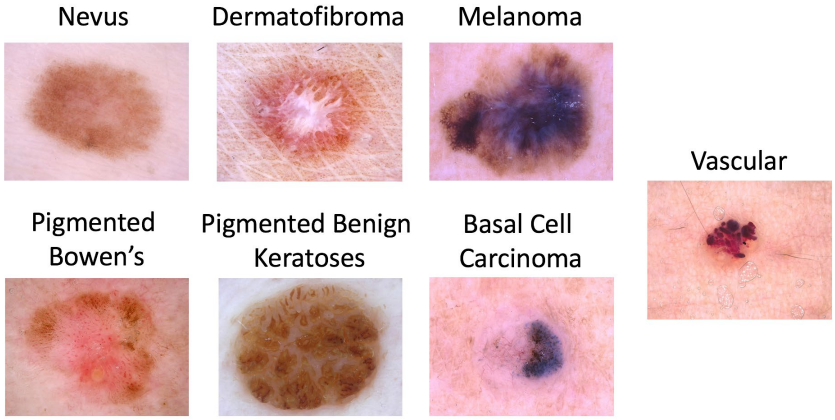

Рис. 3.1. Зображення з набору даних

3.3 Масштабування та розмір зображення

Перед початком роботи з даними їх потрібно привести до будь-якої нормалізованої форми. Першою проблемою є розмірність зображень. Вони всі мають різні пропорції та різну висоту та ширину. Написано скрипт для виявлення всіх варіантів розміру. Для більш зрозумілого вигляду відсортовано за кількістю зображень. Дивись **Додаток А**.

Для подачі зображення до нейронної мережі потрібно привести зображення до однакового розміру. Бувають різні вхідні формати для різних нейронних мереж, але найбільш поширені – 100x100, 112x112 та 224x224. Наша мета привести всі зображення до однакових пропорцій чи розміру, а лише потім вже зменшувати до розміру, який потрібен нейронній мережі. Для відображення як саме впливають на зображення різні методи зменшення та обрізання обрано одне зображення з навчального набору (див. рис. 3.2).

Перший метод зменшення – це звичайне перетворення до пропорцій 1:1 (див. рис. 3.3). Хоча він і являється дуже поширеним, він може дуже сильно спотворювати зображення. Особливо це буде помітно на цьому наборі даних, оскільки тут доволі багато різних пропорцій та розмірів. Також площа чорного фону (на інших зображеннях – шкіри) лише збільшилася при такому масштабуванні.

Ще один спосіб масштабування – додавання штучного краю (див. рис. 3.4), але як і попередній метод, зображення лише збільшило площу чорного фону та кількість не важливих елементів.

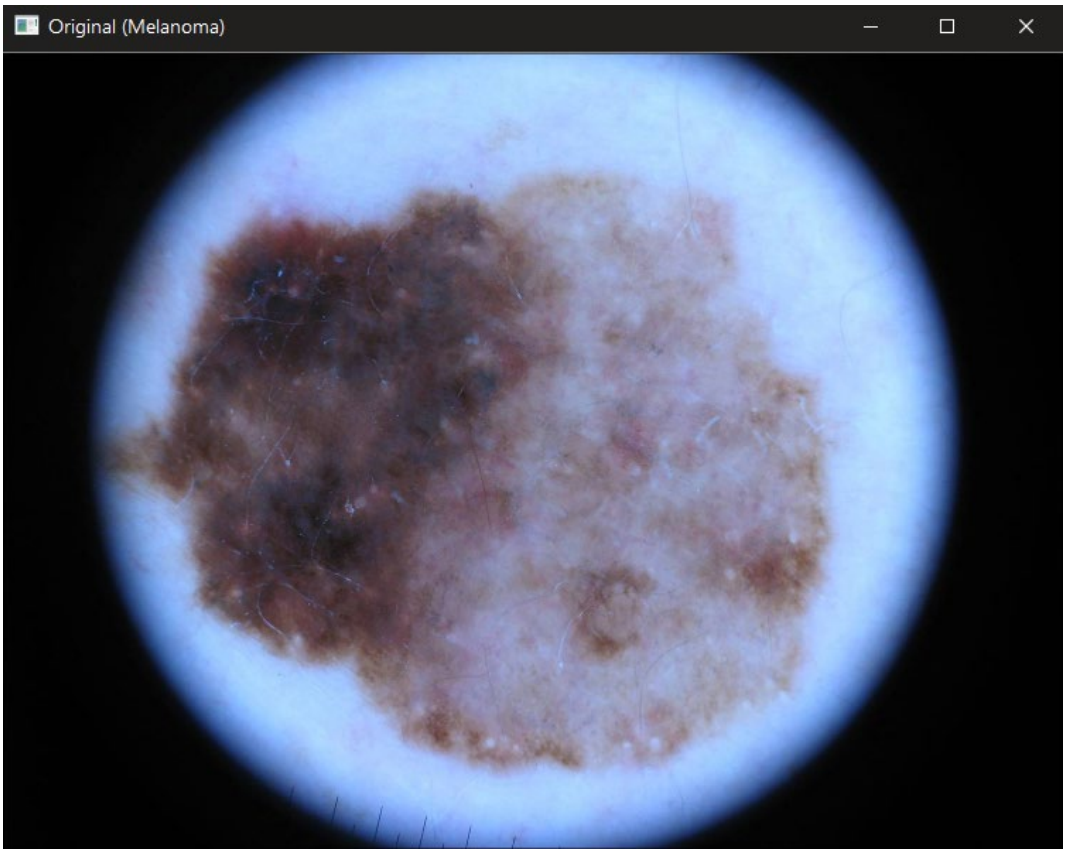

Рис. 3.2. Оригінал зображення

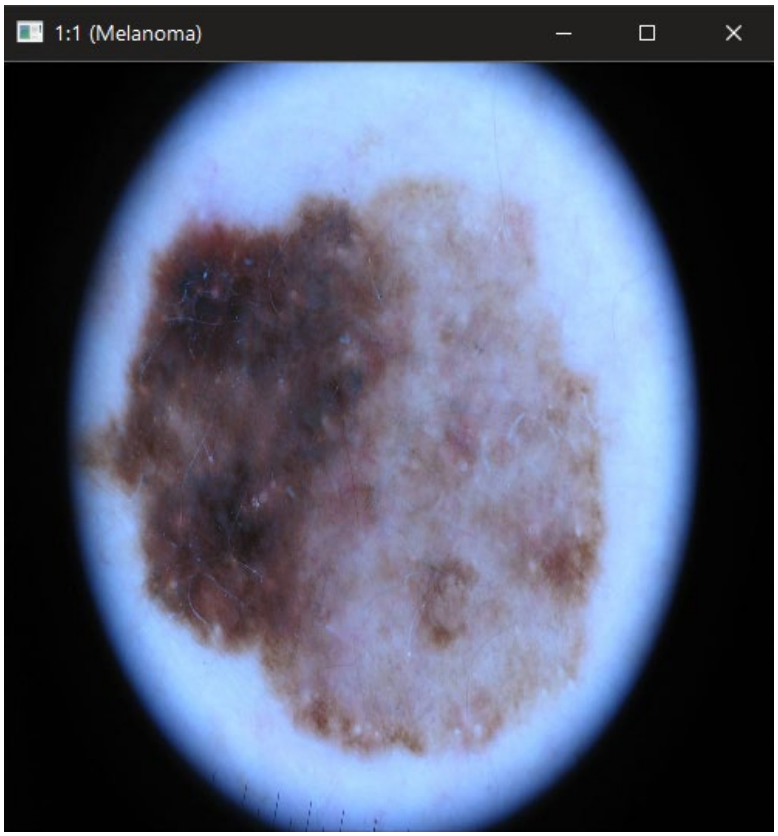

Рис. 3.3. Перетворення до пропорцій 1:1

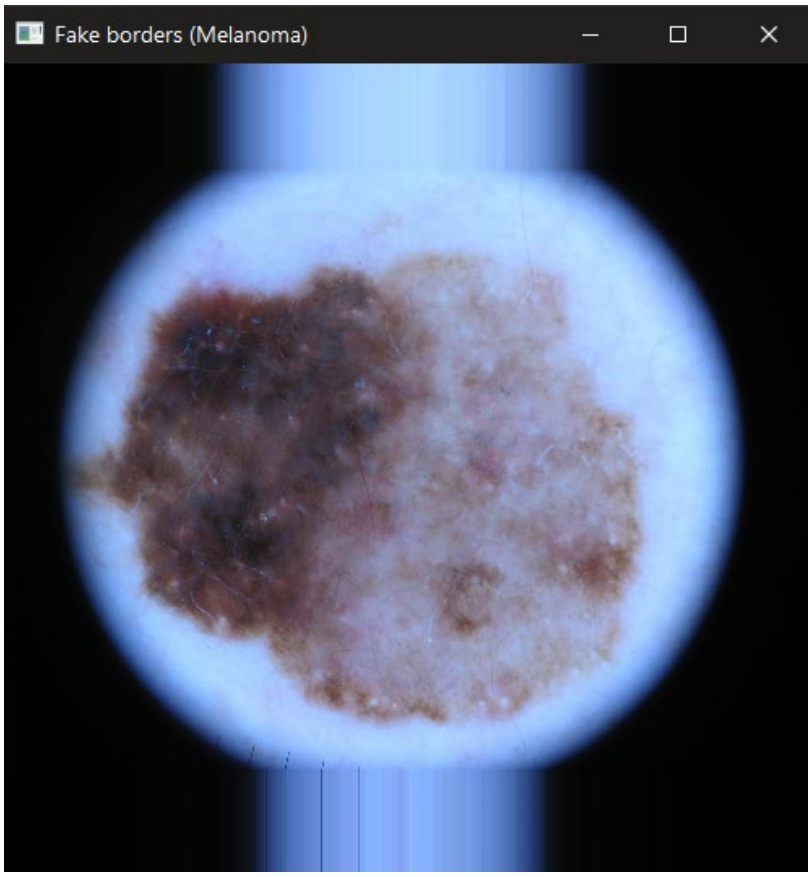

Рис. 3.4. Додавання штучного краю

Згідно аналізу зображень було вирішено використовувати свій метод масштабування (див. рис. 3.5). Алгоритм обробки зображень такий:

1. Обрізаємо 10 пікселів з кожної сторони, адже зазвичай це рамка;
2. Робимо зменшення зображення по найменшій стороні зберігаючи при цьому пропорції другої;
3. Обрізаємо зображення до відношення 1:1.

Саме такий підхід дозволяє обрізати якомога більше зайвого фону або здорової шкіри. Усі зображення перевірені на правильність обрізання. Подібне обрізання зображення може зачепити якусь незначну частину ділянки захворювання. Але завжди ця частка не має вирішального значення.

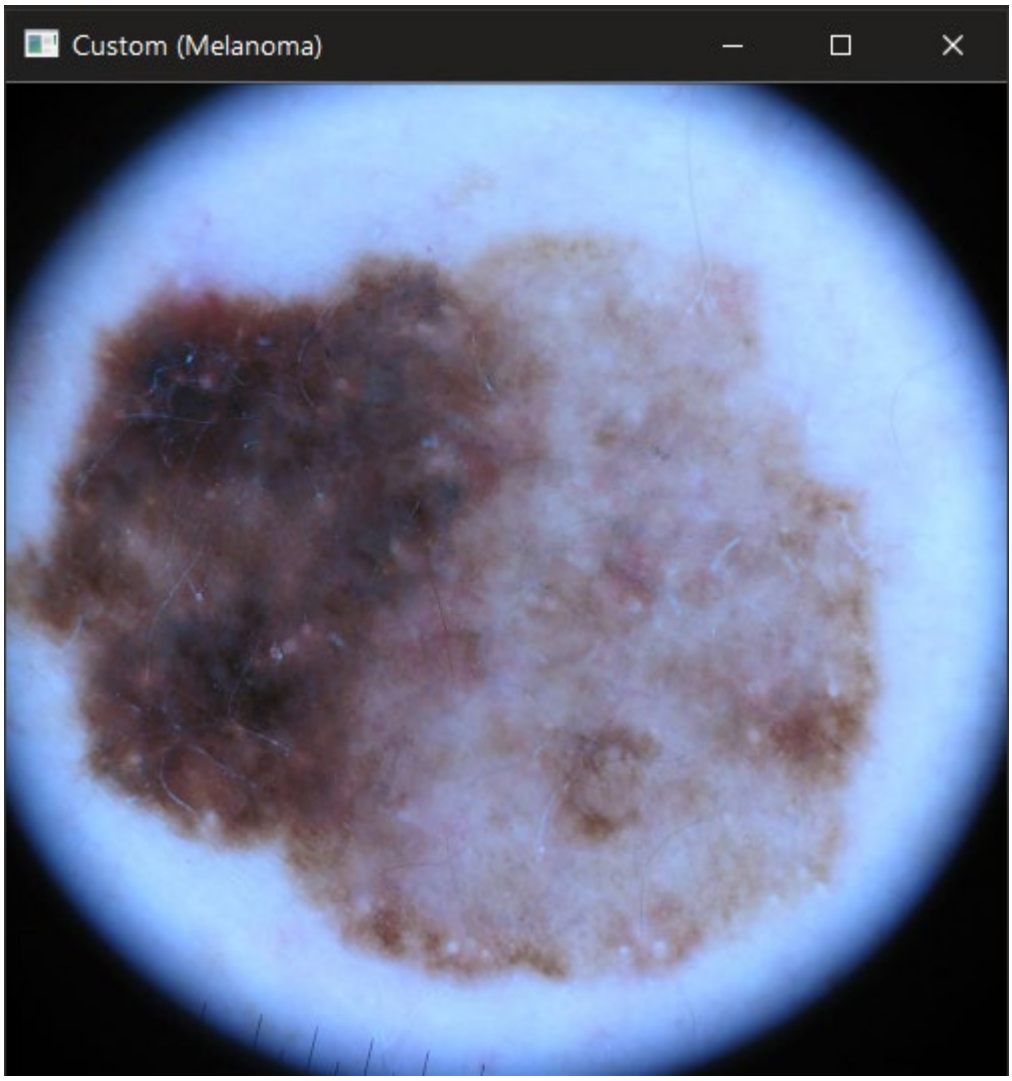

Рис. 3.5. Користувацьке масштабування

Неправильне зменшення та масштабування зображення буде знищувати пропорції і спотворювати характеристики, які можуть бути вирішальними в роботі класифікатору. Але користувацький спосіб масштабування не має таких недоліків. Він зберігає всі характеристики завдяки зменшенню, яке зберігає пропорції, та правильному обрізанню.

3.4 HDF5

Для більш швидкого навчання моделей потрібно використовувати відеокарту, але відео-пам’ять на чипі не без обмежень. А навчання на процесорі з оперативною пам’яттю займає в 20 разів більше часу ніж на відео-чипі. Тому для

навчання на відеокарті потрібно розмістити набір даних у відео-пам’яті. Але набір даних ISIC 2019 важить 9.12ГБ і при розподілені даних на штучні нейронні пласти, він буде ще більший.

В таких випадках використовують формат даних HDF5[22]. Цей формат являється відкритим та доволі розповсюдженим. Завдяки ньому можна працювати з даними на диску, як з даними в оперативній пам’яті, дозволяючи навіть редагувати дані. Він забезпечує швидку доступність та автоматичне завантаження наступних даних, використовуючи буфер.

Такий тип даних не дає 100% швидкодії, як і повне завантаження в оперативну пам’ять чи відео-пам’ять, але він є дуже гарною альтернативою при обмежених ресурсах. Hierarchical Data Format (HDF або ієрархічний формат даних) – назва формату файлів, розробленого для зберігання великого обсягу цифрової інформації. Спочатку був розроблений Національним центром комп'ютерних програм, зараз підтримується некомерційною організацією HDF Group. Складові можна побачити на рисунку 3.6.

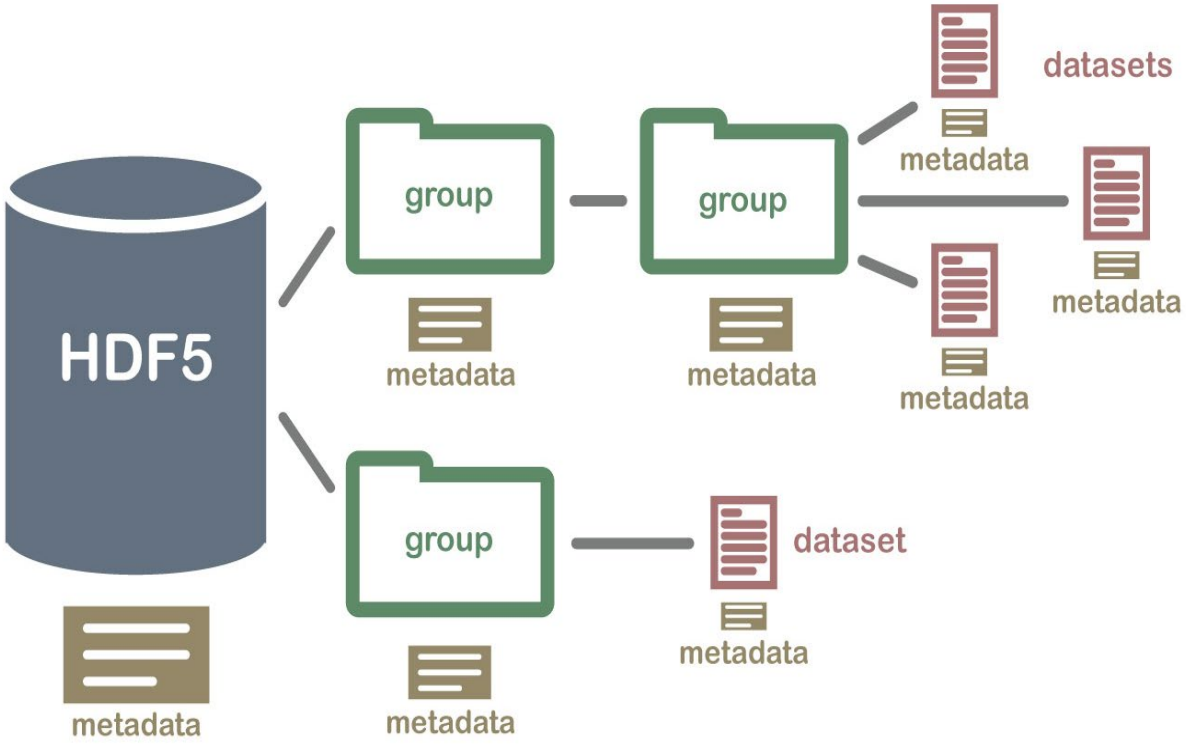

Рис. 3.6. Приклад структури файлу HDF5

Бібліотеки для роботи з форматом та пов'язані з ним утиліти доступні для використання під вільною ліцензією, схожою на ліцензію BSD. Формат HDF підтримується багатьма комерційними та некомерційними програмами, у тому числі є бібліотеки для роботи з ним у Java, Matlab, Scilab, Python, R та Julia. Пакет HDF, що вільно розповсюджується, складається з бібліотеки, утиліти командного рядка, вихідних текстів для тестування, інтерфейсу для Java і програми для перегляду HDF-файлів.

3.5 Розподіл даних для тренування та тестування

Оскільки зображень доволі багато, а саме 25.331 зображення, то вирішено зробити розподіл 80% для навчання та 20% для тестування. Але головна проблема тут полягає в тому, що розподіл даних дуже не рівномірний (див. рис. 3.7).

```
Loading new data...
Total amount of elements: 25331
Count of elements in every class:
    Melanoma: 4522
    Melanocytic nevus: 12875
    Basal cell carcinoma: 3323
    Actinic keratosis: 867
    Benign keratosis: 2624
    Dermatofibroma: 239
    Vascular lesion: 253
    Squamous cell carcinoma: 628
```

Рис. 3.7. Розподіл зображень по класам

Тому, щоб правильно розподілити дані, потрібно використовувати стратегію **stratify**, це розподіл даних згідно загальній кількості зображень в класі. Тобто, якщо в класі n зображень, то в тестову вибірку підуть 0.2 * n зображень і так для всіх класів.

Розміри тестових вибірок:

- Melanoma: 904;
- Melanocytic nevus: 2.575;
- Basal cell carcinoma: 666;
- Actinic keratosis: 173;
- Benign keratosis: 525;
- Dermatofibroma: 48;
- Vascular lesion: 51;
- Squamous cell carcinoma: 126.

3.6 Генерація даних

Оскільки даних деяких класів дуже мало, то потрібно використовувати техніку генерації додаткових даних (data augmentation[23]). Адже наприклад клас дерматофіброма та ураження судин мають усього по приблизно 250 зображень на клас. Це дуже мало відносно найбільшої категорії, меланометальний невус, яка має 12875 зображень.

Генерація даних – це метод генерування нових навчальних даних без зміни міток класів шляхом застосування деяких випадкових спотворень і перешкод. Основним мотивом розширення даних є підвищення узагальненості моделі, тому що якщо ми передаємо більше даних в нейронну мережу, вона може тренуватися точніше, використовуючи нові дані. Таким чином, модель може вивчати більш надійні характеристики, завжди переглядаючи нові дані. Ми повинні застосовувати збільшення даних лише під час навчання моделі, а не під час тестування. Після застосування генерації даних ми бачимо збільшення точності моделі в порівнянні з моделлю, яка навчається без розширення даних.

Уявімо, що ми збираємося навчити систему виявляти на зображенні певний об’єкт. Але на всіх зображеннях об’єкт у центрі зображення, згідно цього мережа не зможе розпізнавати цей же об’єкт в повернутому стані, чи у нижній частині зображення. Це стосується не тільки положення об’єкта, а й деякі інші поняття, як

наприклад освітлення, колір, контрастність. Таким чином, всі вищезазначені випадки відхилення можуть потрапляти в модель під час роботи, тому вона повинна вміти їх розрізняти також. Для цього генерується додатковий набір даних, щоб отримати належну модель, яка зможе працювати з усіма видами спотворень.

Головна мета генерації даних – створення нових на основі існуючих. Основна умова, щоб нові дані залишалися того ж самого класу.

Обрані характеристики для генератору даних:

1. Випадковий поворот на +-180 градусів. Оскільки нам немає значення в якому положенні було сфотографовано хворобу шкіри, то ми використовуємо будь-який поворот зображення при генерації даних. Адже будь-які хвороби від нахилу не перестають бути хворобами;
2. Випадкова зміна ширини та довжини на 30%. Допомагає моделі краще вивчати хвороби незалежно від їх площі. Також це допомагає знаходити ураження шкіри у будь-якому положенні, отже модель буде шукати саме особливі характеристики хворої ділянки шкіри, а не його розмір;
3. Випадковий зсув з коефіцієнтом 20. На відміну від зміни ширини та висоти він спотворює зображення фіксуючи одну сторону та розтягуючи іншу;
4. Випадкове прибільшення/віддалення на 20%. Схоже за принципом на зміну ширини та довжини, але збільшує та віддаляє зображення не спотворюючи пропорції;
5. Випадкове дзеркальне відображення;
6. Заповнення граней завдяки найближчим пікселям.

На зображеннях далі показано роботу генерації даних для різних класів. Як можна побачити, зображення змінюються доволі сильно, але основні характеристики незмінні.

Результати генерації даних дивись на рисунку 3.8-3.12.

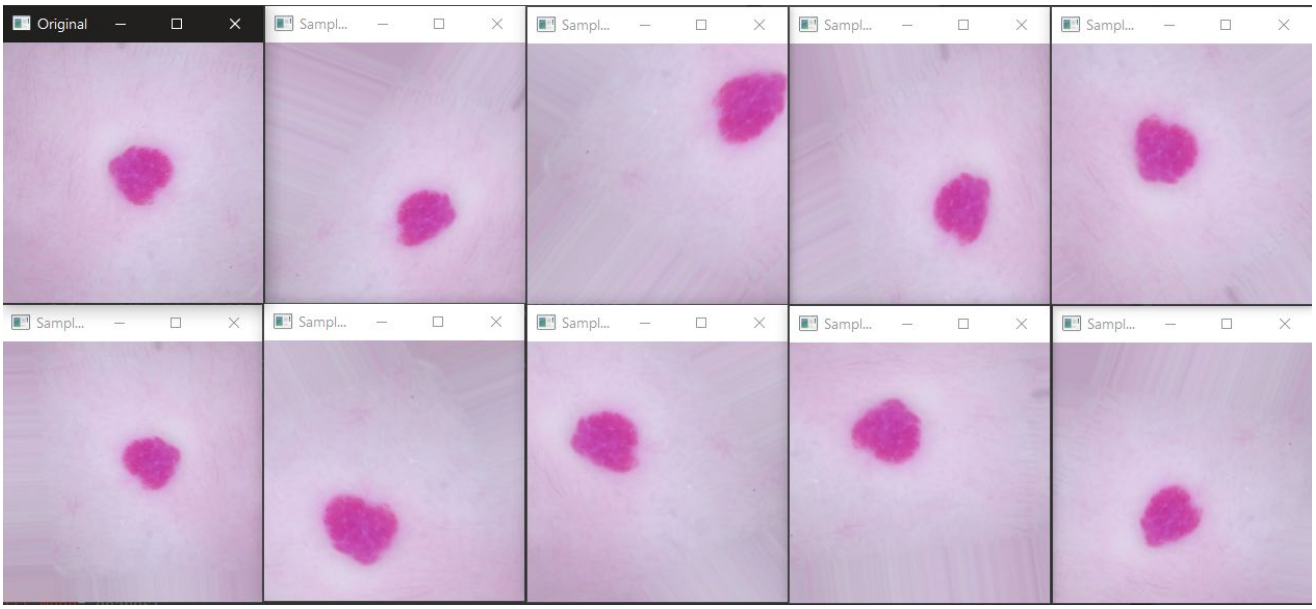

Рис. 3.8. Vascular lesion

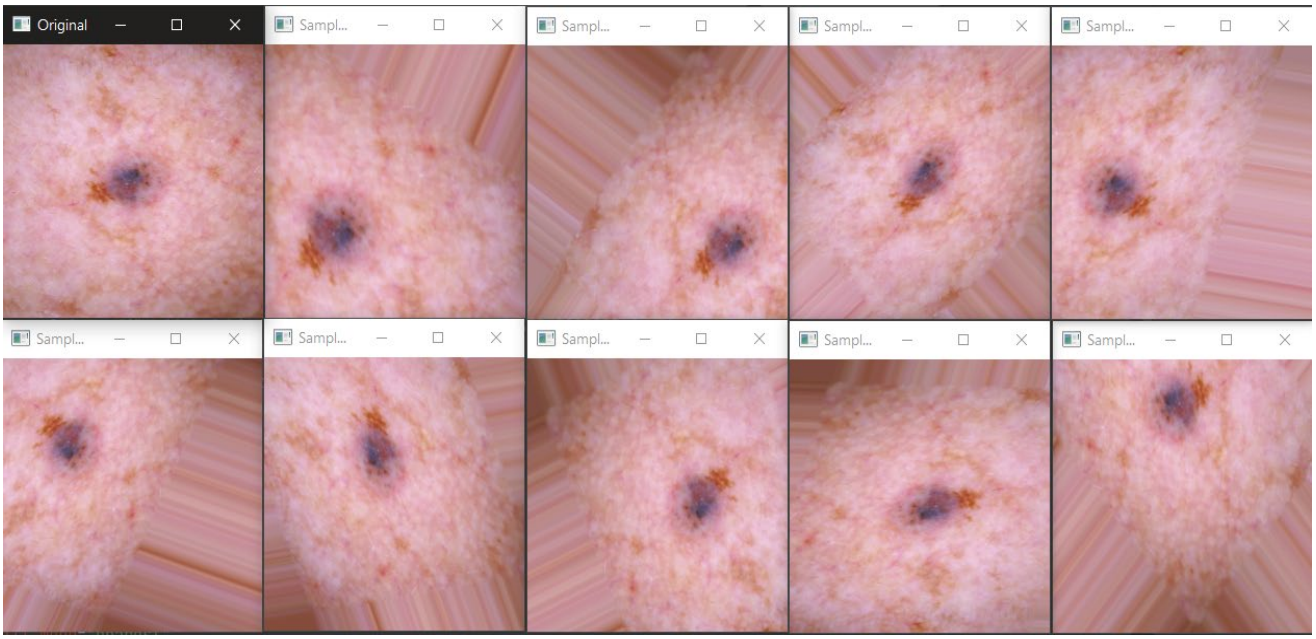

Рис. 3.9. Basal cell carcinoma

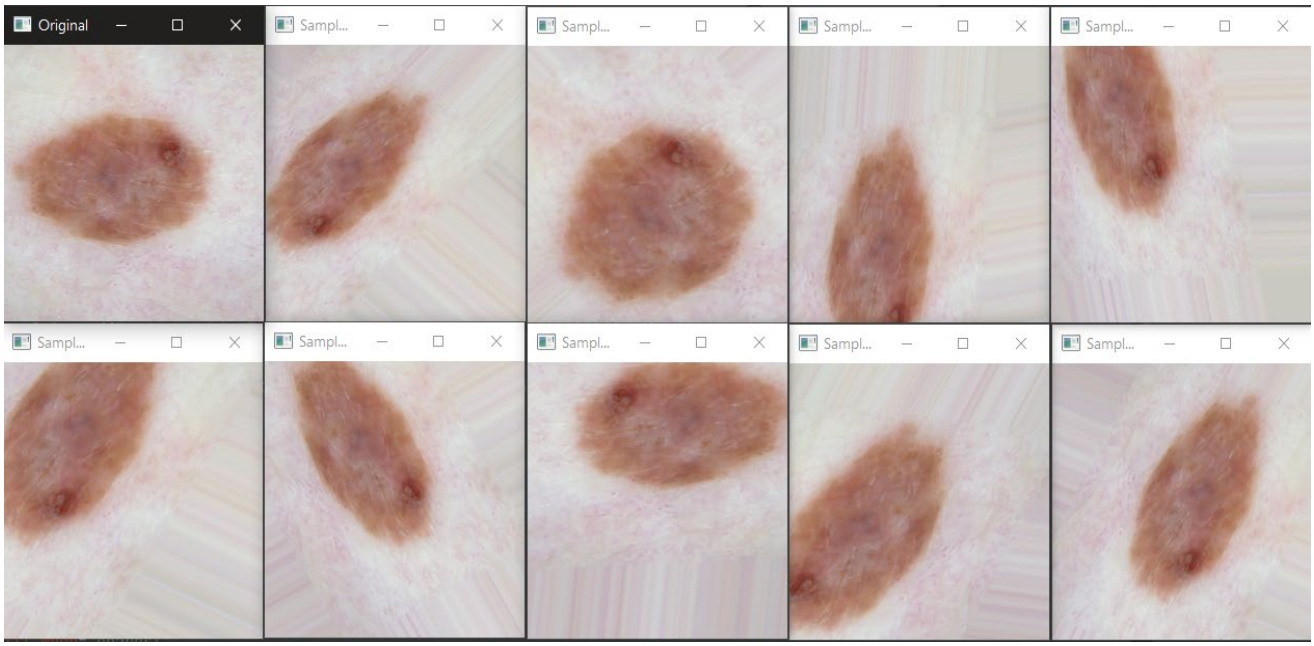

Рис. 3.10. Seborrheic keratosis

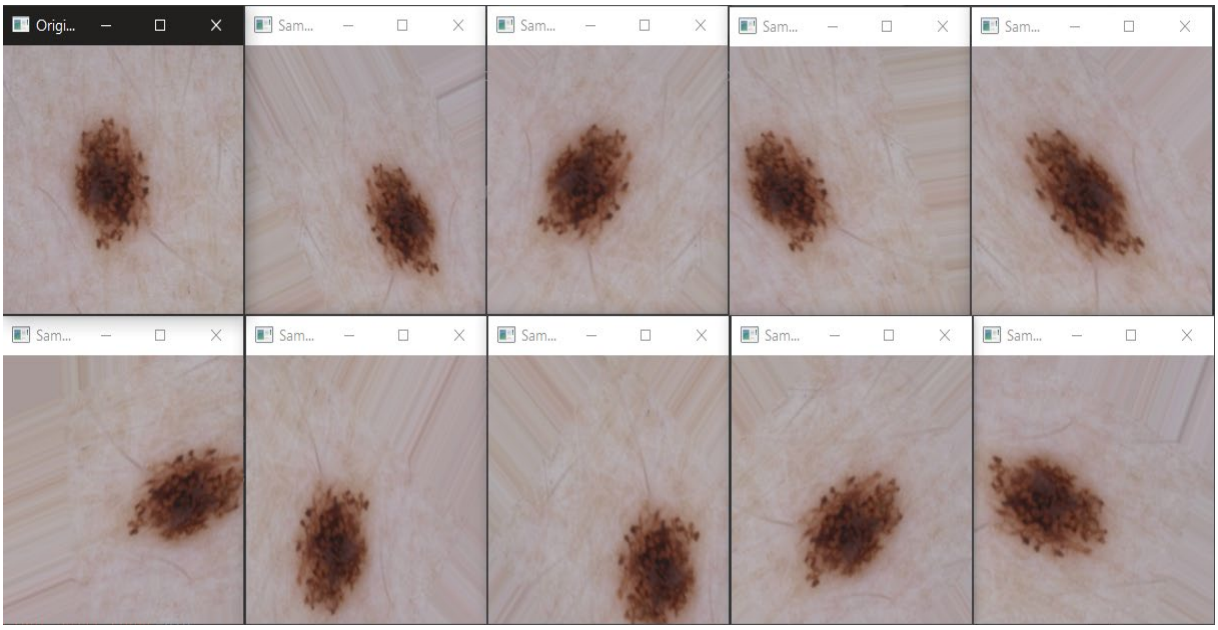

Рис. 3.11. Nevus

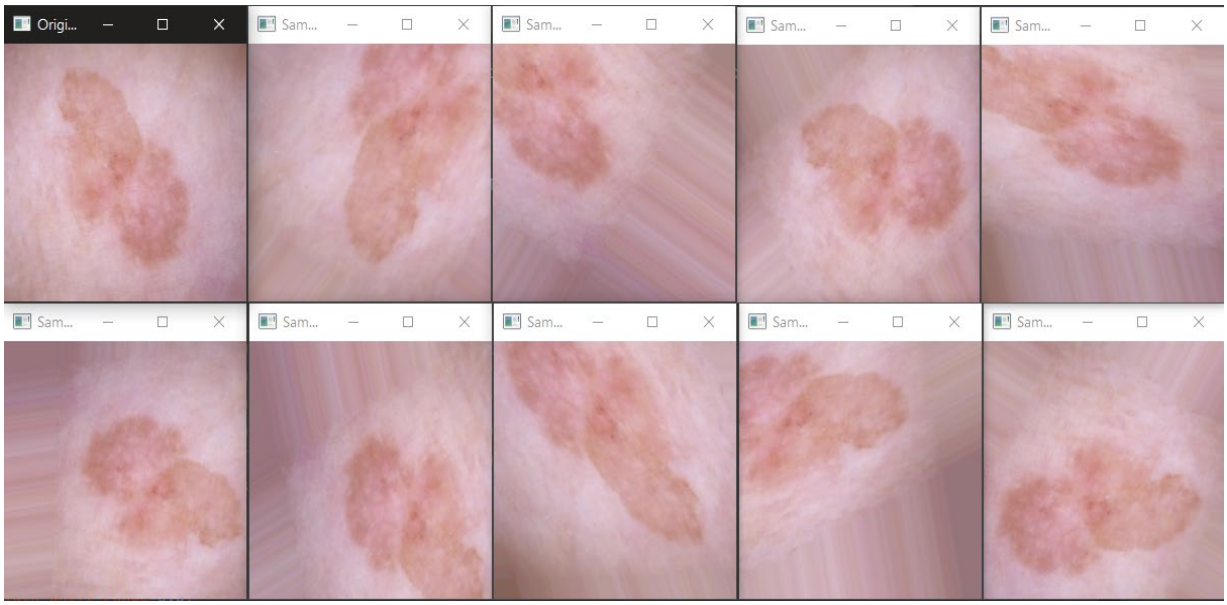

Рис. 3.12. Squamous cell carcinoma

Висновки до розділу 3

Описано організацію Internation Skin Imaging Collaboration, яка збирає дані для навчання моделей. Охарактеризовано основні особливості набору даних. Виконано дослідження найбільш оптимального алгоритму масштабування зображень. Використано HDF5 для можливості роботи з дуже великою кількістю зображень. Розподіллено дані для тестування та навчання. Виконано генерації даних.

4 НАВЧАННЯ МОДЕЛЕЙ

Вибір моделей здійснюється від менш глибоких до більш глибоких та складних.

4.1 Формат опису нейронних мереж

Оскільки TensorFlow дозволяє легко налаштовувати архітектури нейронних мереж, то щоб не шукати реалізацію кожної нейронної мережі в різних бібліотеках, вирішено створювати нейроні мережі власноруч.

Такий підхід має багато переваг, адже можна додавати та корегувати будь-який шар мережі. Завдяки цьому можна використовувати різні види активації. Також для збільшення точності можна використовувати шари нормалізації. А для покращення показника узагальнення знань мережі додавати шари з випадковим вимкненням нейронів.

Тому для кращого розуміння мереж, які будуть використовуватися в подальшому, та їх складових в цьому розділі буде проведено опис усіх шарів та їх назви в коді.

Шар активації. Цей шар підставляє суму зважених вхідних параметрів в зазначену функцію для кожного вузла. Приймає параметр з назвою функції активації. Це може бути «relu», «softmax», «sigmoid».

Довгі роки використовували функцію «sigmoid» та «tanh». Однак в задачах комп’ютерного зору найкращі результати показує «relu» функція та її похідні («leaky relu», «elu»).

Графік різних функцій активації можна побачити на рисунку 4.1.

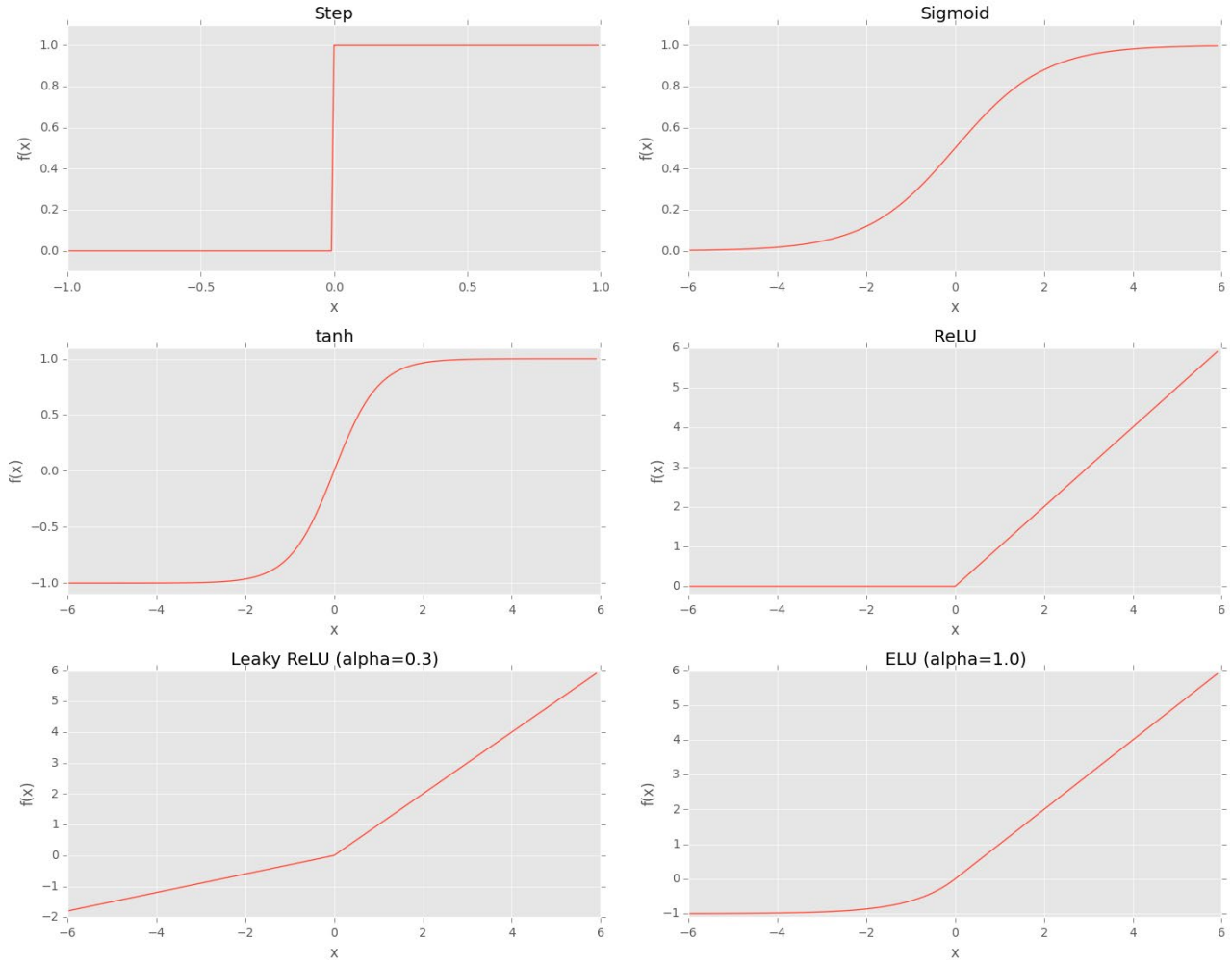

Рис. 4.1. Графіки різних функцій активації

Шар згортки (див. рис. 4.2)**.** Цей шар використовуючи певну маску дозволяє проходитися по всьому зображенню та знаходити подібності. Приймає параметри:

- кількість масок;
- розмірність масок;
- розмір вихідного зображення («same» – залишає той самий розмір);
- розмір вхідного сигналу – параметр для першого згорткового шару.

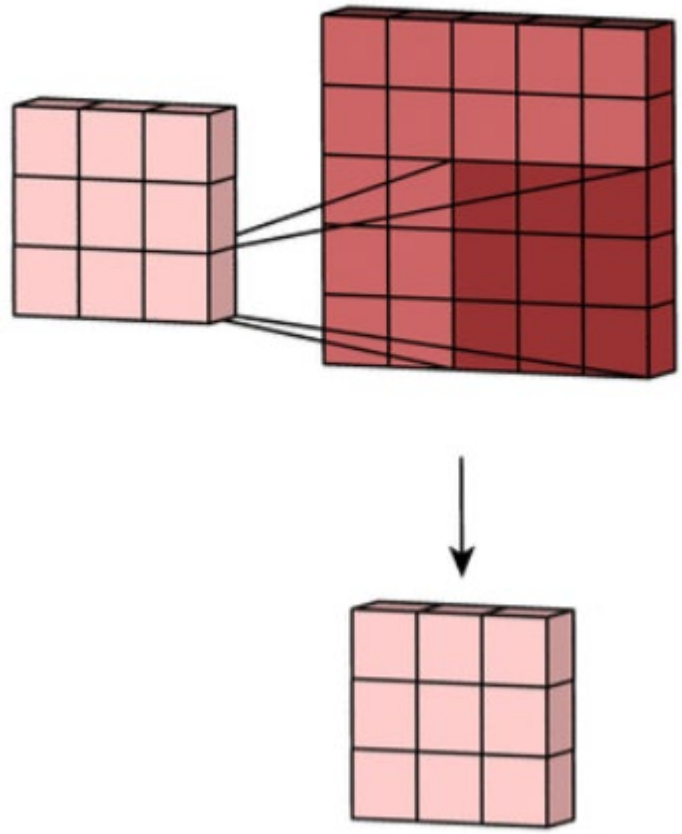

Рис. 4.2. Приклад роботи шару згортки

Шар нормалізації. Використовується для покращення результатів. Але доволі сильно сповільнює навчання. Цей шар приймає як параметр номер осі, в якій знаходяться шари. Це потрібно оскільки існують різні формати опису зображень (глибина кольору в першій осі, та глибина кольору в останній осі)

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

Шар активації для зменшення зображення. Використовується для зменшення зображення не втрачаючи при цьому признаки з найбільшими коефіцієнтами. Може використовувати різні функції, але зазвичай використовує функцію максимуму. Як параметр приймає розмір сітки для зменшення.

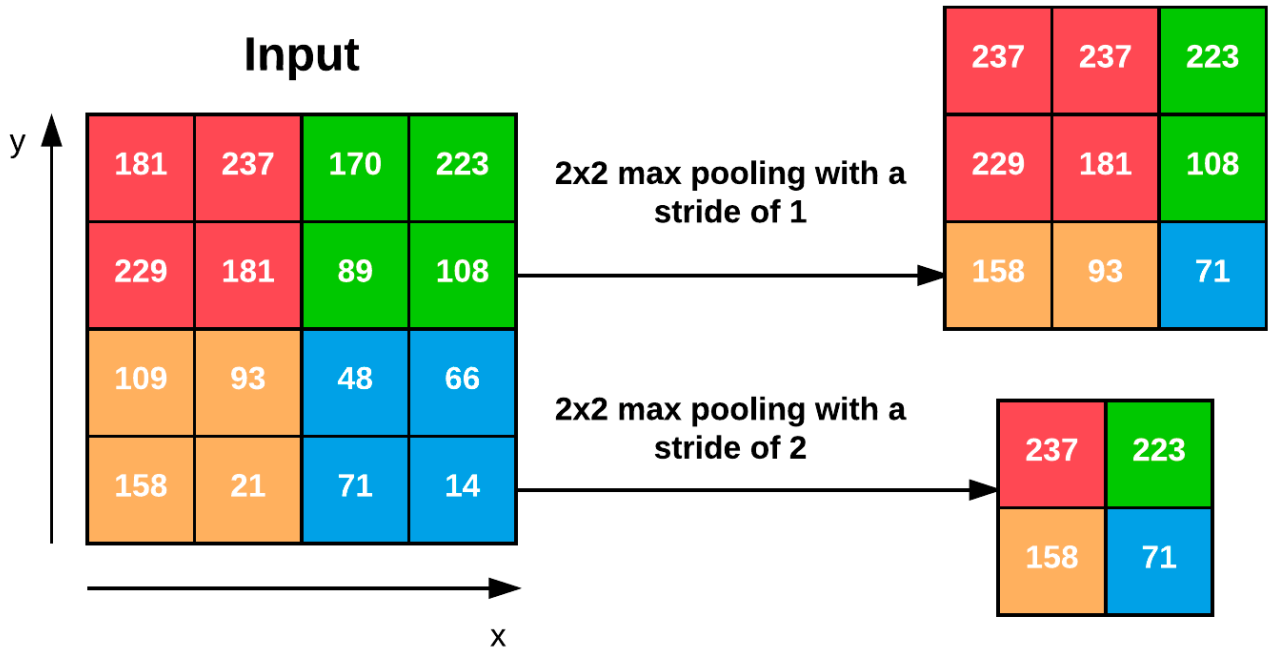

Рис. 4.3. Приклад зменшення зображення

Шар випадкового вимкнення нейронів. Використовується для покращення узагальнених знань мережі. Завдяки вимкненню нейронів різні нейрони вчяться розпізнавати одні й ті самі шаблони, тим самим покращуючи точність. Як параметр приймає відсоток для вимкнення.

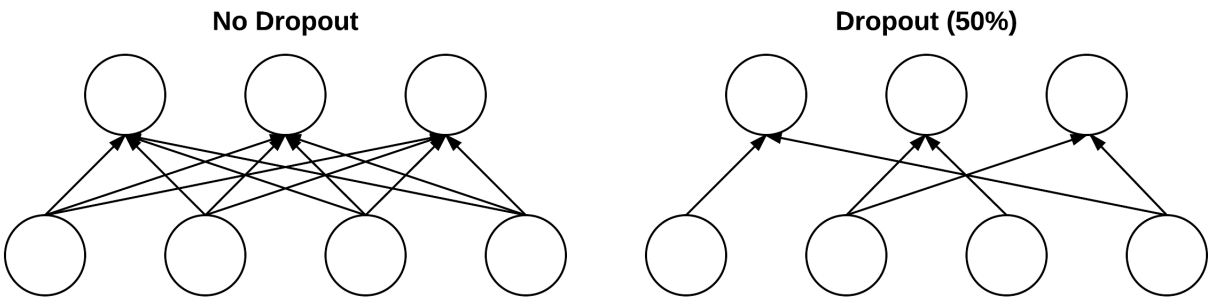

Рис. 4.4. Принцип роботи шару

Шар для перетворення матриці з багатьма осями в вектор. Використовується для передачі в штучній нейроні мережі.

Приклад роботи. На вхід подається зображення 100x100x3. Тоді результат буде 30.000 вхідних параметрів.

Шар зі штучною нейронною мережею. Можна використовувати багато різних послідовно. Як параметр приймає кількість вузлів.

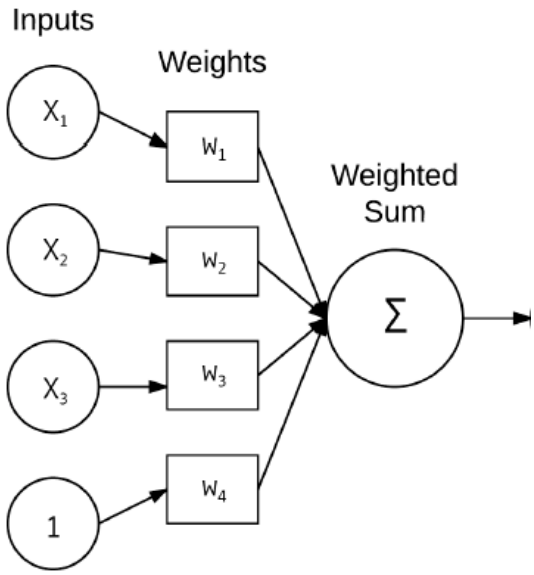

Рис. 4.5. Принцип роботи одного нейрону

4.2 MiniVGG

Першою моделлю для навчання обрано MiniVGG [24]. Ця мережа не найпростіша, але більш прості немає сенсу використовувати, адже задача, яку потрібно вирішити доволі складна.

Архітектура мережі:

```
Conv2D(32, (3, 3), padding="same", input_shape=input_shape)
Activation("relu")
BatchNormalization(axis=channels_dim)
Conv2D(32, (3, 3), padding="same")
Activation("relu")
BatchNormalization(axis=channels_dim)
MaxPooling2D(pool_size=(2, 2))
Dropout(0.25)
```

```
Conv2D(64, (3, 3), padding="same")
Activation("relu")
BatchNormalization(axis=channels_dim)
Conv2D(64, (3, 3), padding="same")
Activation("relu")
BatchNormalization(axis=channels_dim)
MaxPooling2D(pool_size=(2, 2))
Dropout(0.25)

Flatten()
Dense(512)
Activation("relu")
BatchNormalization()
Dropout(0.5)

Dense(classes)
Activation("softmax")
```

Особливістю VGG-подібних архітектур є те, що використовуються згорткові шари з маскою 3x3 та по два шари одразу, а лише після цього йде об’єднання. В даному випадку 2 шари по 32 маски, об’єднання 2x2, 2 шари по 64 маски, об’єднання 2x2.

Розроблена версія MiniVGG має свої модифікації, адже було додано шари BatchNormalization та Dropout. Це допоможе досягти найкращих результатів цієї мережі.

Мережа MiniVGG не глибока, тому використовувати популярний розмір 224x224 не бажано, тому треба брати менші розміри.

Параметри першої моделі:

- алгоритм оптимізації – зворотне поширення помилки;
- функція витрат – categorical crossentropy;
- розмір групи (batch size) – 64;
- коефіцієнт навчання – 0.0001;
- розмір зображення 100x100.

Результатом навчання мережі після 200 епох.

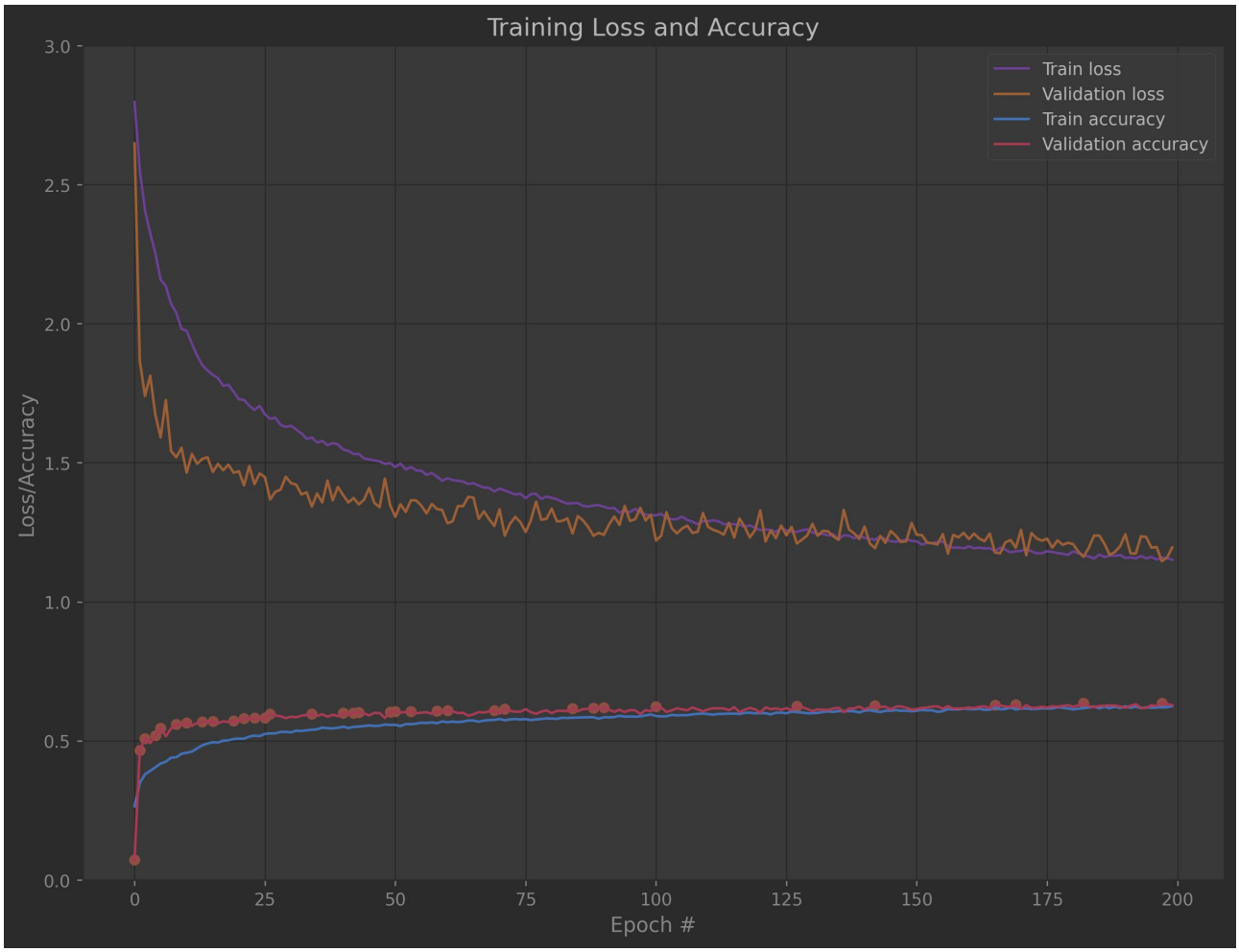

Рис. 4.6. Графік навчання

	precision	recall	f1-score	support
0	0.67	0.31	0.42	904
1	0.79	0.83	0.81	2575
2	0.47	0.66	0.55	665
3	0.24	0.25	0.25	173
4	0.32	0.47	0.38	525
5	1.00	0.02	0.04	48
6	0.82	0.35	0.49	51
7	0.27	0.07	0.11	126
accuracy			0.63	5067

Рис. 4.7. Точність навчання

Мережа навчилася з точністю **63%** класифікувати хвороби шкіри. Але як можна побачити по графіку навчання, точність зростає дуже повільно. Навіть зупинилося зростання тренувальної точності при досягнені 63 та більше відсотків. 200 епох замало для заданих параметрів. Надалі точність класифікації тренувальних даних буде зростати, але все повільніше і повільніше, а тестова не буде змінюватися. Даний процес називається – **недонавчання**. Причиною цього може бути дуже низький коефіцієнт навчання.

Тому для вирішення проблеми використовуємо ті самі характеристики моделі, але з коефіцієнтом навчання 0.001. Наша мета – досягти такого коефіцієнту, щоб нейрона мережа могла вивчати тренувальні дані й надалі, а не зупинятися на 63%.

Результат такого навчання можна побачити на рисунках 4.8-4.9. Збільшення коефіцієнту допомогло мережі навчати тренувальну вибірку до 95% та не зупиняти зростання на 63%. А тестова точність зросла до **68.52%**. Хоча на це знадобилося 240 епох.

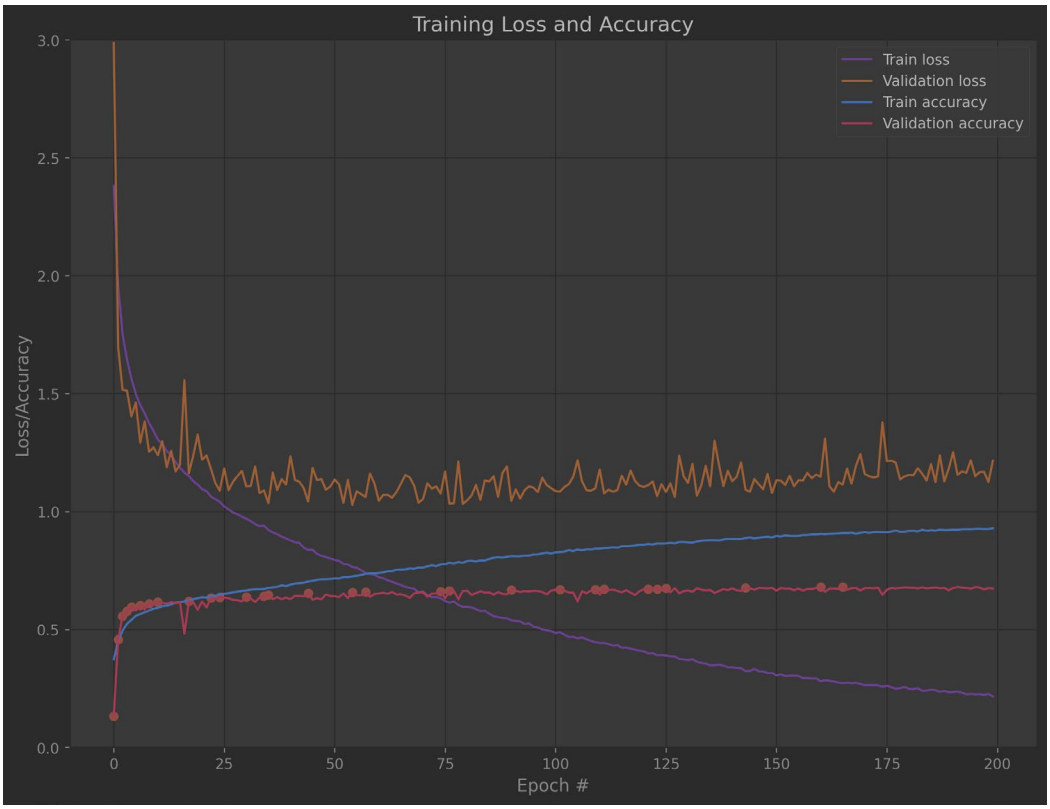

Рис. 4.8. Графік навчання

	precision	recall	f1-score	support
0	0.66	0.42	0.52	904
1	0.77	0.89	0.83	2575
2	0.59	0.58	0.59	665
3	0.34	0.27	0.30	173
4	0.40	0.48	0.44	525
5	0.35	0.19	0.24	48
6	0.90	0.51	0.65	51
7	0.38	0.20	0.26	126
accuracy			0.67	5067

Рис. 4.9. Точність навчання

Спробуємо створити модель з використанням вагових коефіцієнтів для класів. Точно ті самі параметри, що й у минулої моделі, але з вагами.

Вагові коефіцієнти для класів – це масив значень для корегування важливості навчання будь-якого класу. Найчастіше використовується для навчання моделей, де класифікація одного класу має більше значення, ніж класифікація іншого. Зазвичай використовується з оптимізатором Adam. Однак може мати негативні наслідки, як погане навчання класів з меншим ваговим коефіцієнтом, тому потрібно це відслідковувати.

В нашому випадку вони використовуються для врегулювання нерівномірності даних в наборі. Хоча алгоритм зворотного поширення помилки і сам вміє врегульовувати розмір помилки, вагові коефіцієнти будуть додатковим важелем для врегулювання нерівномірності даних.

Ваги для заданого датасету можна побачити на рисунку 4.10. Вони обраховуються за рахунок поділу кіл-ті даних в навчальному наборі на кількість в цьому класі.

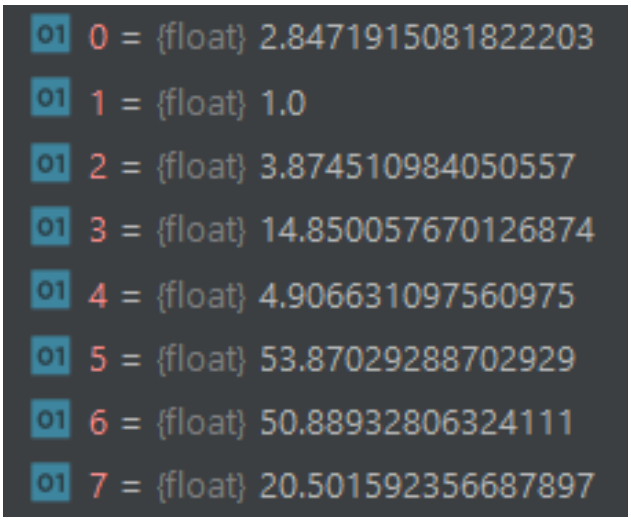

Рис. 4.10. Ваги для класів

Рис. 4.11. Графік навчання

	precision	recall	f1-score	support
0	0.61	0.44	0.51	904
1	0.79	0.86	0.83	2575
2	0.61	0.60	0.60	665
3	0.33	0.35	0.34	173
4	0.42	0.48	0.45	525
5	0.23	0.23	0.23	48
6	0.69	0.75	0.72	51
7	0.38	0.26	0.31	126
accuracy			0.67	5067

Рис. 4.12. Точність навчання

Завдяки використанню ваги для класів вдалося досягти приблизно такої ж точності (**68.28%**). Але по графіку навчання видно, що loss значення для тестового набору занадто нестабільне, а для тренувальних даних, як і очікувалось, воно збільшене в 5 разів. Тобто вагові коефіцієнти для класів не дають ніякого покращення. Вони не підходять для алгоритму зворотного поширення.

4.3 Модифікований MiniVGG

Спробуємо зробити MiniVGG модель більш глибокою. Збільшимо кількість масок на першій парі згорткових шарів до 64 (було 32), а на другій парі до 128 (було 64). Також додаємо один шар нейронної мережі з 1024 вузлами, перед тим з 512 вузлів.

Детальний опис модифікованої MiniVGG.

```
Conv2D(64, (3, 3), padding="same", input_shape=input_shape)
Activation("relu")
BatchNormalization(axis=channels_dim)
Conv2D(64, (3, 3), padding="same")
Activation("relu")
BatchNormalization(axis=channels_dim)
MaxPooling2D(pool_size=(2, 2))
```

Dropout(0.25)

Conv2D(*128*, (3, 3), padding="same")
Activation("relu")
BatchNormalization(axis=channels_dim)
Conv2D(*128*, (3, 3), padding="same")
Activation("relu")
BatchNormalization(axis=channels_dim)
MaxPooling2D(pool_size=(2, 2))
Dropout(0.25)

Flatten()
Dense(1024)
Activation("relu")
BatchNormalization()
Dropout(0.5)

Dense(512)
Activation("relu")
BatchNormalization()
Dropout(0.5)

Dense(classes)
Activation("softmax")

Час виконання збільшився в 2 рази через складність мережі. Тому на навчання мережі 200 епохами уходить вже 4 години. Найкращий результат, який вдалося отримати – **65.29%**.

Це гірший результат в порівнянні з попередньою менш глибокою моделлю. Причиною цього може бути занадто маленький коефіцієнт навчання. Зменшуємо коефіцієнт навчання до 0.01.

	precision	recall	f1-score	support
0	0.73	0.30	0.43	904
1	0.74	0.90	0.81	2575
2	0.53	0.54	0.54	665
3	0.29	0.34	0.31	173
4	0.39	0.41	0.40	525
5	0.25	0.10	0.15	48
6	0.53	0.53	0.53	51
7	0.26	0.17	0.20	126
accuracy			0.64	5067

Рис. 4.13. Результат навчання

Зменшення допомогло досягти попереднього результату точності, а саме **67.83%** точності. Це такі ж самі результати, як і в попередній моделі. Але складність обчислень в цій моделі набагато більша, тому попередня модель більш оптимізована.

	precision	recall	f1-score	support
0	0.60	0.46	0.52	904
1	0.79	0.87	0.83	2575
2	0.58	0.65	0.61	665
3	0.42	0.25	0.32	173
4	0.42	0.43	0.42	525
5	0.22	0.17	0.19	48
6	0.55	0.59	0.57	51
7	0.44	0.15	0.22	126
accuracy			0.67	5067

Рис. 4.14. Результат навчання

4.4 AlexNet

AlexNet [25] розробили Алекс Крижевський та інші. у 2012 році для участі в конкурсі ImageNet [26]. Загальна архітектура дуже схожа на LeNet-5, хоча ця модель значно більша та глибша. Саме ця модель зайняла перше місце на конкурсі ImageNet 2012 року. Успіх цієї моделі переконав велику частину спільноти комп'ютерного зору серйозно поглянути на більш глибокі моделі для завдань комп'ютерного зору.

Модель навчали як велику глибоку згорткову нейронну мережу, щоб класифікувати 1,2 мільйона зображень з високою роздільною здатністю на 1000 різних класів. За даними тесту було досягнуто показників помилок у першій і п'ятій ланках 37,5% та 17,0%, що значно краще, ніж у будь-яких моделей до цього в ImageNet змаганні. Нейронна мережа, яка має 60 мільйонів параметрів і 650 000 нейронів, складається з п'яти згорткових шарів, за деякими з яких слідують шари з максимальним об'єднанням, і три повністю пов'язані шари з остаточним 1000-стороннім softmax. Щоб пришвидшити навчання, було використано нейрони, що не насичаються, і дуже ефективну реалізацію операції згортки на графічному процесорі. Щоб зменшити перенавчання застосовувався нещодавно розроблений метод регуляризації Dropout, який виявився дуже ефективним.

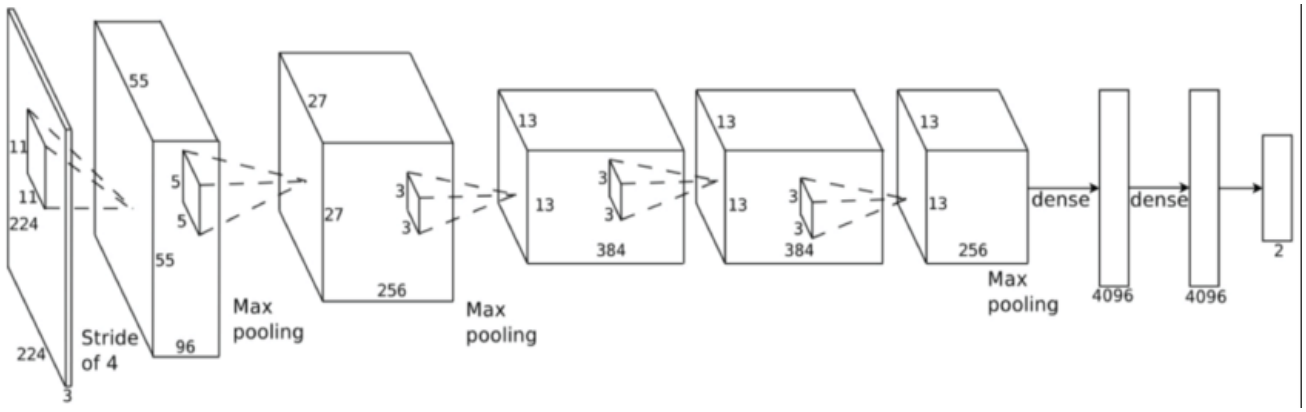

Рис. 4.15. Графічне зображення моделі

Детальний опис моделі:

```
Conv2D(96, (11, 11), padding="same", strides=4, input_shape=input_shape)
Activation("relu")
BatchNormalization(axis=channels_dim)
MaxPooling2D(pool_size=(3, 3), strides=2)
Dropout(0.25)

Conv2D(256, (5, 5), padding="same", strides=4)
Activation("relu")
BatchNormalization(axis=channels_dim)
MaxPooling2D(pool_size=(3, 3), strides=2)
Dropout(0.25)

Conv2D(384, (3, 3), padding="same", strides=4)
Activation("relu")
BatchNormalization(axis=channels_dim)

Conv2D(384, (3, 3), padding="same", strides=4)
Activation("relu")
BatchNormalization(axis=channels_dim)

Conv2D(256, (3, 3), padding="same", strides=4)
Activation("relu")
BatchNormalization(axis=channels_dim)

Flatten()

Dense(4096)
Activation("relu")
BatchNormalization()
Dropout(0.5)

Dense(4096)
Activation("relu")
BatchNormalization()
Dropout(0.5)

Dense(classes)
Activation("softmax")
```

Дана модель, на відміну від VGG-подібних має згорткові шари з розміром більше ніж 3x3, а саме 11x11 та 5x5 в перших шарах. Це допомагає шукати великі шаблони на перших шарах, а надалі вже розбивати їх на менші. Також вона не використовує Pooling, використовується згортка зі зменшенням розміру (strides).

Навчання моделі видало помилку в тестуванні та не досягло точності більше ніж 50%. Все це через маленький розмір зображення (100x100). Для навчання цієї моделі потрібно використати збільшення розміру зображення до 224x224 (як в оригінальній статті). Адже модель дуже глибока та сильно зменшує зображення на своїх шарах згортки.

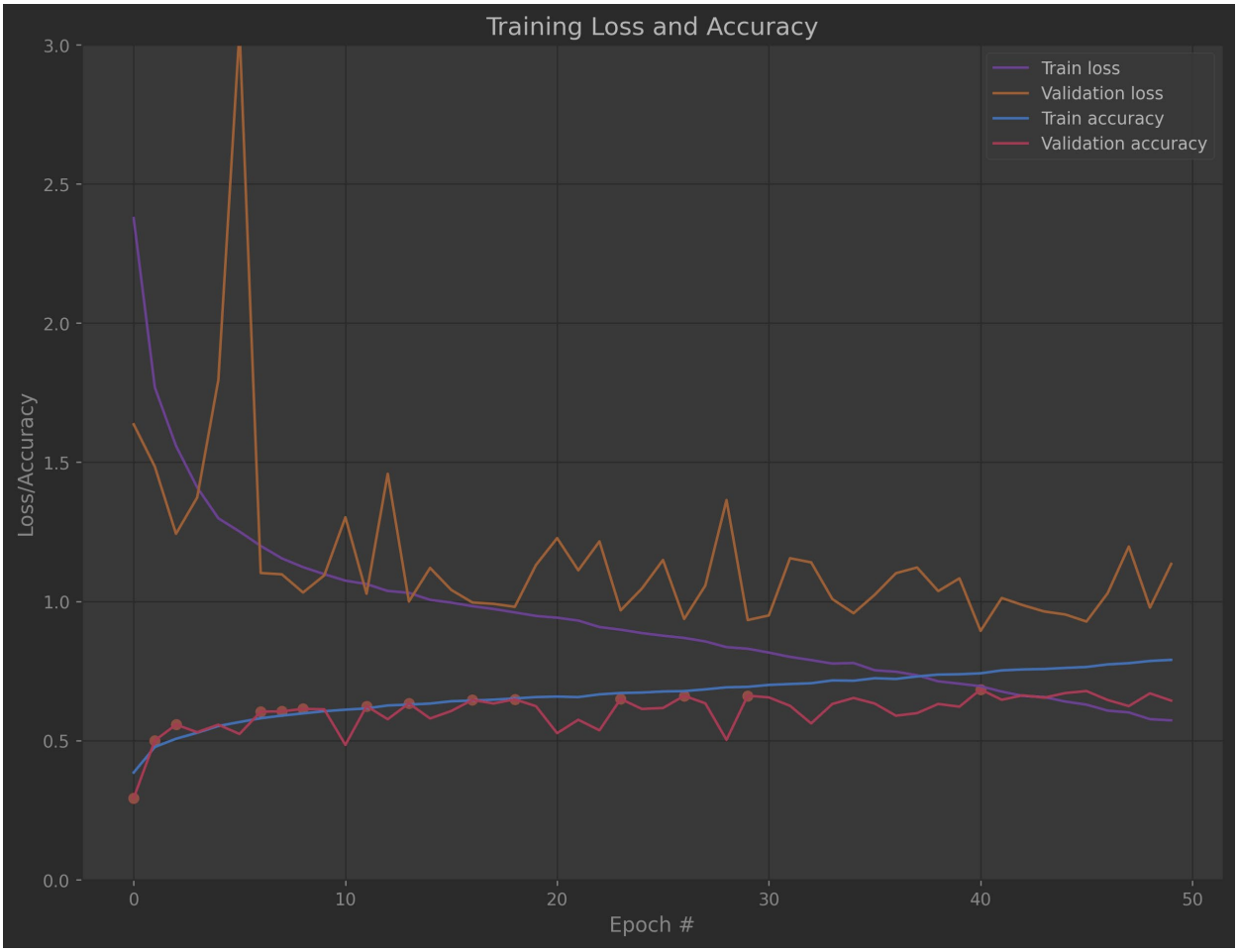

Рис. 4.16. Графік навчання

	precision	recall	f1-score	support
0	0.63	0.45	0.53	904
1	0.77	0.88	0.82	2575
2	0.57	0.71	0.63	665
3	0.41	0.31	0.35	173
4	0.49	0.39	0.44	525
5	0.47	0.15	0.22	48
6	0.74	0.67	0.70	51
7	0.52	0.19	0.28	126
accuracy			0.68	5067

Рис. 4.17. Результат навчання

В результаті навчання моделі вдалося досягти точності **68.34%**. Це приблизно така сама точність, яку змогли досягти з MiniVGG моделлю.

Оскільки мережа була створена для 1000 різних класів, вона може бути занадто складною для 8 класів. Тому робимо спробу зменшити кількість вузлів у двох Dense шарах до 512. Зменшення вузлів в штучних нейронних шарах допоможе зменшити кількість характеристик, які буде шукати модель, і тим самим дасть змогу краще вивчати їх.

Однак при цьому згорткова частина моделі не змінюється і її глибина залишається тією самою.

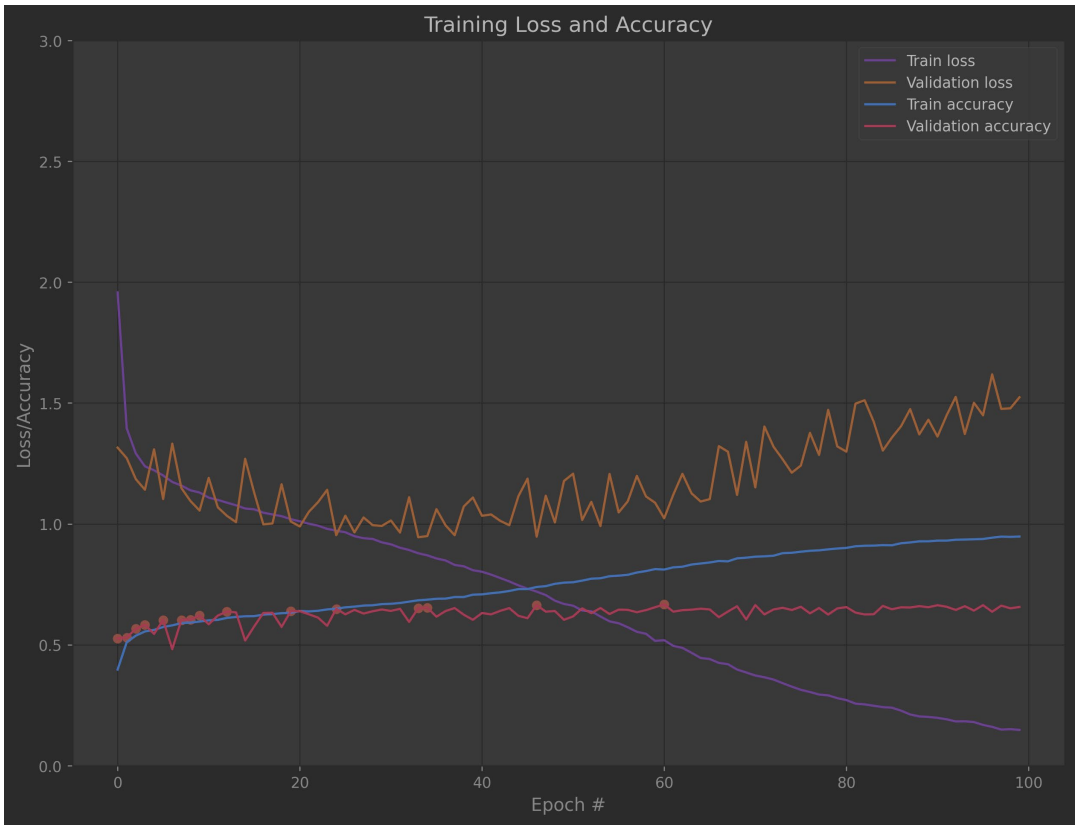

Рис. 4.18. Графік навчання

Згідно графіку навчання можна побачити перенавчання після 30 епохи. Що свідчить, про погане навчання моделі.

	precision	recall	f1-score	support
0	0.57	0.50	0.53	904
1	0.80	0.80	0.80	2575
2	0.53	0.73	0.61	665
3	0.34	0.33	0.34	173
4	0.43	0.40	0.41	525
5	0.29	0.04	0.07	48
6	0.65	0.63	0.64	51
7	0.40	0.21	0.28	126
accuracy			0.66	5067

Рис. 4.19. Результат навчання

Найкраща точність складає **66.65%**. Тобто модифікація не принесла ніяких покращень.

4.5 VGG16

VGG16 (див. рис. 4.20) [27] – це мережа, яка започаткувала поширення VGG-подібних мереж. Вона впроваджує техніку з використанням більшої кількіості згорткових фільтрів розміром 3x3, без використання більших фільтрів.

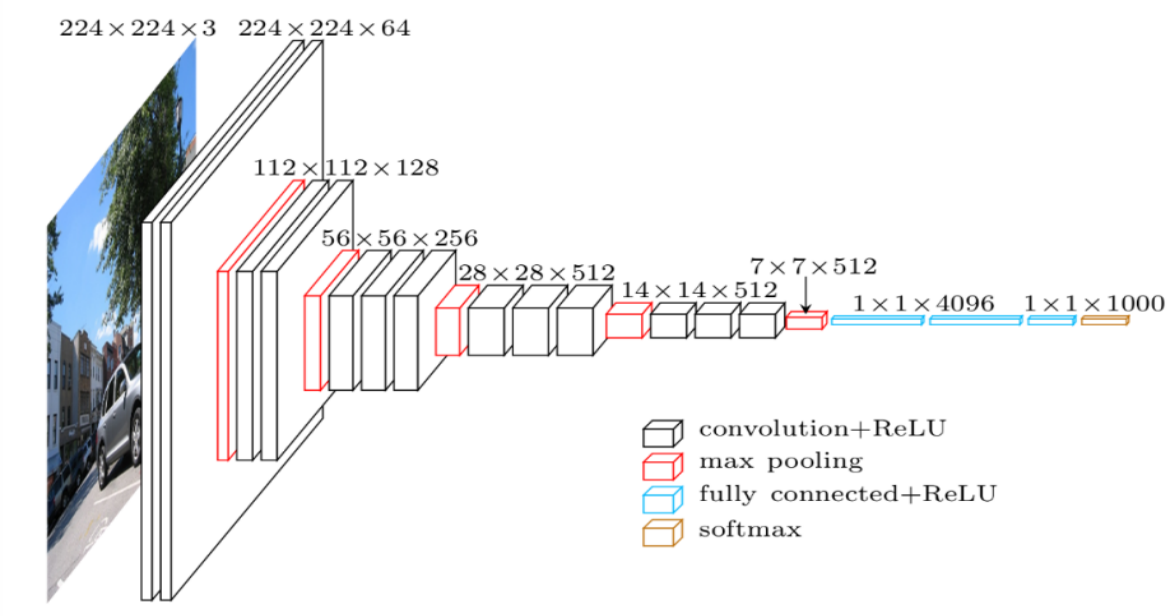

Рис. 4.20. Архітектура мережі

Її автори у своїй роботі досліджують вплив глибини згорткової мережі на її точність у налаштуваннях розпізнавання великомасштабних зображень. Основна особливість мережі – збільшення глибини з використанням архітектури з дуже малими (3x3) фільтрами згортки, що показує значне покращення в порівнянні з попередніми конфігураціями, яке можна досягти шляхом збільшення глибини. Ці висновки лягли в основу конкурсу ImageNet Challenge 2014, де мережа зайняла перше та друге місця відповідно до локалізації та класифікації. Також показано,

що ознаки добре узагальнюються на інші набори даних, де вони досягають найсучасніших результатів.

Через обмеження в можливостях технічного обладнання, *неможливо* використовувати повну оригінальну модель. Тому потрібно було зменшити кількість фільтрів та вузлів у нейронній мережі. А саме згорткові шари з 512 фільтрів були видалені, а замість них були додані початкові шари з 32 фільтрами. Просто їх видалити неможливо, адже разом з ними відбувається зменшення зображення в 2 рази, а якщо не зменшувати, то модель просто не може розміститися в пам’яті.

Детальний опис моделі:

```
Conv2D(32, (3, 3), padding="same", input_shape=input_shape)
Activation("relu")
BatchNormalization(axis=channels_dim)
Conv2D(32, (3, 3), padding="same")
Activation("relu")
BatchNormalization(axis=channels_dim)
MaxPooling2D(pool_size=(2, 2))
Dropout(0.20)

Conv2D(64, (3, 3), padding="same", input_shape=input_shape)
Activation("relu")
BatchNormalization(axis=channels_dim)
Conv2D(64, (3, 3), padding="same")
Activation("relu")
BatchNormalization(axis=channels_dim)
MaxPooling2D(pool_size=(2, 2))
Dropout(0.20)

Conv2D(128, (3, 3), padding="same", input_shape=input_shape)
Activation("relu")
BatchNormalization(axis=channels_dim)
Conv2D(128, (3, 3), padding="same")
Activation("relu")
BatchNormalization(axis=channels_dim)
MaxPooling2D(pool_size=(2, 2))
Dropout(0.20)
```

```
Conv2D(256, (3, 3), padding="same", input_shape=input_shape)
Activation("relu")
BatchNormalization(axis=channels_dim)
Conv2D(256, (3, 3), padding="same")
Activation("relu")
BatchNormalization(axis=channels_dim)
Conv2D(256, (3, 3), padding="same")
Activation("relu")
BatchNormalization(axis=channels_dim)
MaxPooling2D(pool_size=(2, 2))
Dropout(0.20)

Flatten()

Dense(1024)
Activation("relu")
BatchNormalization()
Dropout(0.5)

Dense(1024)
Activation("relu")
BatchNormalization()
Dropout(0.5)

Dense(512)
Activation("relu")
BatchNormalization()
Dropout(0.5)

Dense(classes)
Activation("softmax")
```

Використовуємо коефіцієнт навчання 0.01.

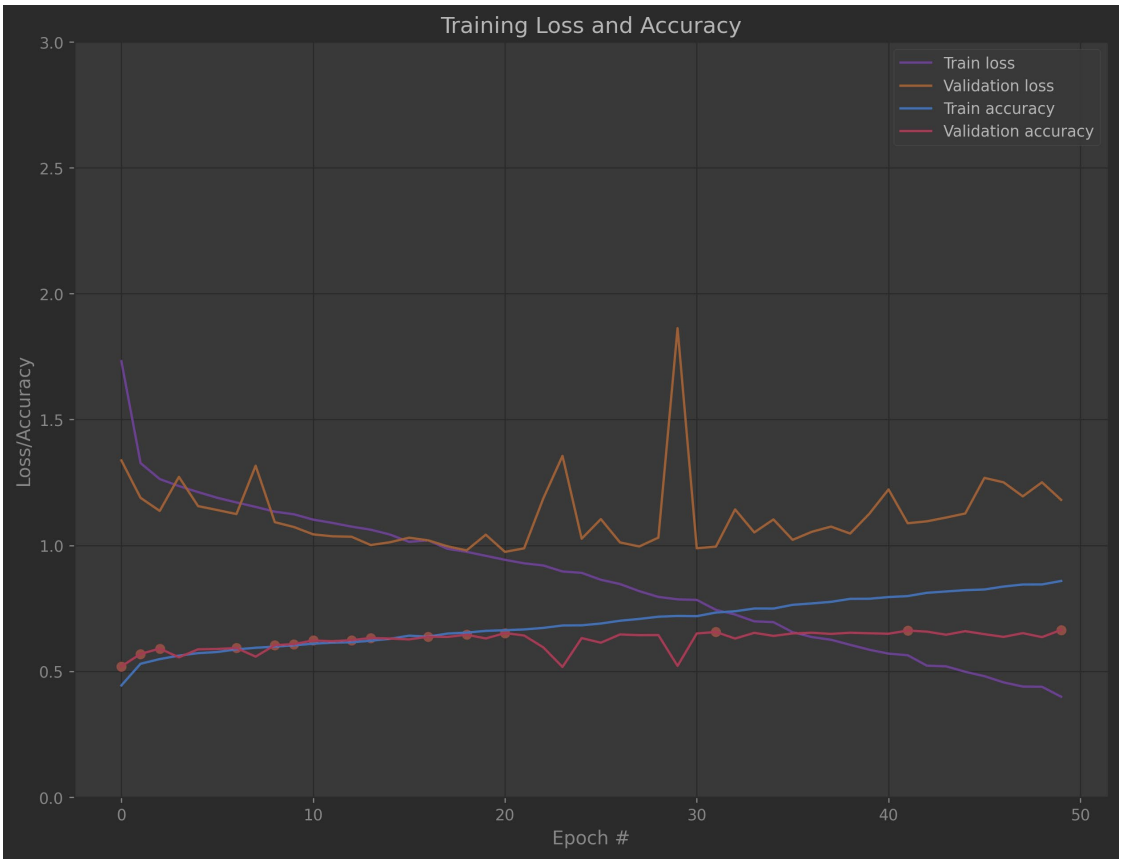

Рис. 4.21. Графік навчання

	precision	recall	f1-score	support
0	0.56	0.53	0.54	904
1	0.79	0.84	0.82	2575
2	0.56	0.65	0.60	665
3	0.42	0.21	0.28	173
4	0.38	0.36	0.37	525
5	0.47	0.19	0.27	48
6	0.63	0.47	0.54	51
7	0.34	0.18	0.24	126
accuracy			0.66	5067

Рис. 4.22. Результат навчання

Згідно навчання моделі вдалося отримати точність **66.43%**.

Висновки до розділу 4

Описано детальний формат опису нейронних мереж для кращого розуміння архітектурних рішень та створених моделей. Описано шари, які використовуються в моделі: Activation, Conv2D, BatchNormalization, MaxPooling2D, Dropout, Flatten, Dense.

Виконано опис створених моделей та результати їх навчання.

Таблиця 4.1

Точність моделей

Назва моделі	Особливості та параметри	Точність
MiniVGG	Розмір зображення: 100x100 Коефіцієнт навчання: 0.0001	63.23%
MiniVGG	Розмір зображення: 100x100 Коефіцієнт навчання: 0.001	68.52%
MiniVGG	Розмір зображення: 100x100 Коефіцієнт навчання: 0.001 З використанням ваги для класів	68.28%
Модифікований MiniVGG	Розмір зображення: 100x100 Коефіцієнт навчання: 0.001	65.29%
Модифікований MiniVGG	Розмір зображення: 100x100 Коефіцієнт навчання: 0.01	67.83%
AlexNet	Розмір зображення: 224x224 Коефіцієнт навчання: 0.01	68.34%
AlexNet	Розмір зображення: 224x224 Коефіцієнт навчання: 0.01 Зменшено кількість шарів в нейронній мережі до 512	66.65%
VGG16	Розмір зображення: 224x224 Коефіцієнт навчання: 0.01 Зменшено кількість шарів в нейронній мережі до 512	66.43%

Найкращі результати показали MiniVGG та AlexNet. Але згідно оптимізації та складності моделі, найбільш оптимальною є MiniVGG, адже вона, на відміну від AlexNet потребує менше ресурсів та працює з меншою роздільною здатністю, що робить її швидшою.

5 ДОСЛІДЖЕННЯ ВПЛИВУ ПАРАМЕТРІВ НАВЧАННЯ НА ТОЧНІСТЬ

5.1 Вплив коефіцієнту навчання

Для дослідження коефіцієнту навчання використовується модель MiniVGG, оскільки вона виявилася найбільш оптимальною, серед усіх використаних до цього. Також вона має найбільшу швидкість навчання на одну епоху, через те що вона не має дуже глибокої архітектури та роздільна здатність зображення не висока.

Параметри для навчання:

- розмір зображення: 100x100;
- розмір однієї групи (batch size): 32;
- оптимізатор: зворотне поширення помилки (SGD);
- автоматичне зменшення коефіцієнту навчання на 1% з кожною епохою;
- кількість епох: 30.

Використовуючи ці параметри та різні коефіцієнти навчання проведемо дослідження впливу коефіцієнту навчання на процес навчання.

Згідно дослідження графіків та результатів навчання (див. рис. 5.1-5.10) можна підтвердити загальновідомі факти:

- чим більший коефіцієнт навчання, тим скоріше навчається мережа;
- більший коефіцієнт навчання сприяє більш швидкому перенавчанню мережі (тренувальна точність продовжує зростати, а точність для тестування залишається на місці, або навіть спадає), через те що мережа занадто швидко вивчає ознаки саме тренувальних даних, а не узагальнені ознаки класу;
- більший коефіцієнт сприяє більш нестабільному та різкому зростанню Validation loss;
- хоча більший коефіцієнт показує кращі значення точності за коротший період часу, але подальше зростання може загалі зупинитися.

Коефіцієнт навчання: 0.1 (див. рис. 5.1-5.2).

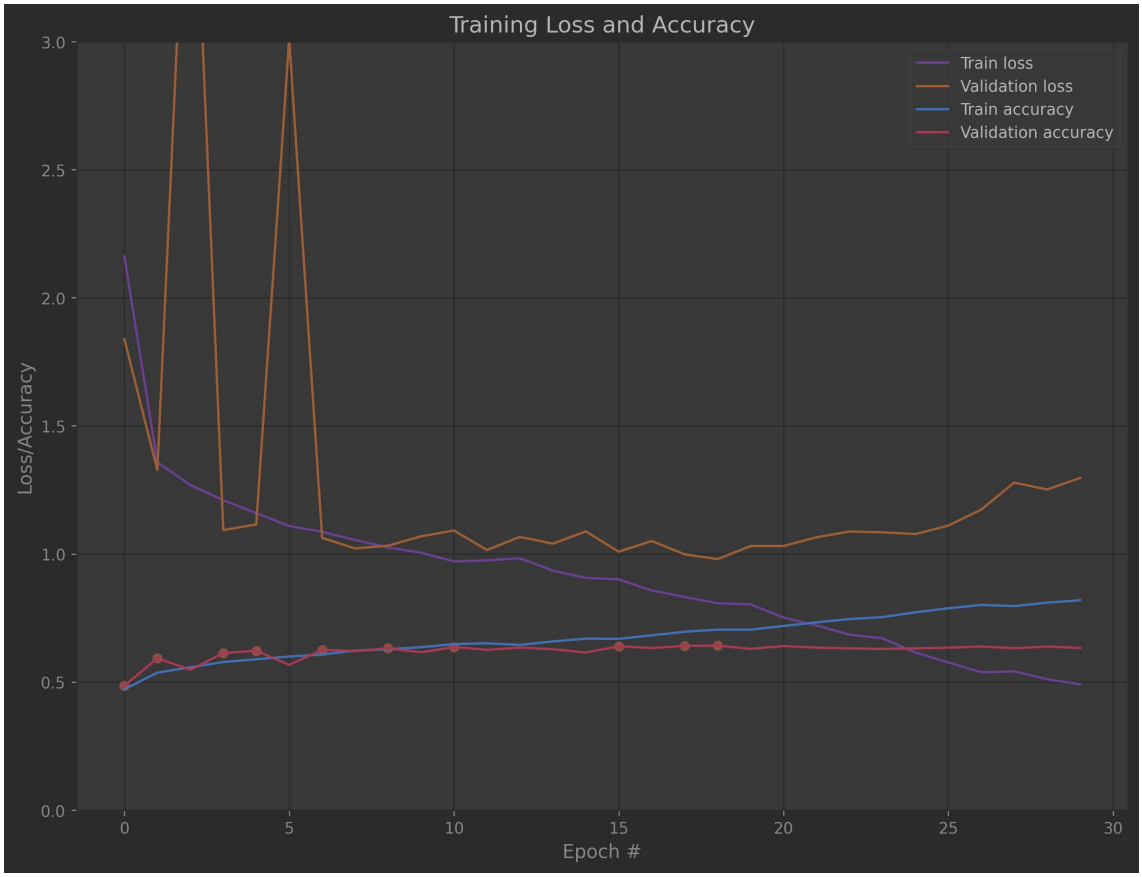

Рис. 5.1. Графік навчання

	precision	recall	f1-score	support
0	0.51	0.41	0.46	904
1	0.73	0.88	0.80	2575
2	0.49	0.63	0.55	665
3	0.35	0.10	0.15	173
4	0.38	0.20	0.26	525
5	0.33	0.06	0.11	48
6	0.63	0.33	0.44	51
7	0.42	0.06	0.11	126
accuracy			0.63	5067

Рис. 5.2. Результат навчання

Найкраща точність: 64.02%.

Коефіцієнт навчання: 0.01 (див. рис. 5.3-5.4).

Рис. 5.3. Графік навчання

	precision	recall	f1-score	support
0	0.58	0.45	0.51	904
1	0.75	0.89	0.81	2575
2	0.57	0.58	0.58	665
3	0.33	0.22	0.26	173
4	0.42	0.33	0.37	525
5	0.38	0.06	0.11	48
6	0.66	0.53	0.59	51
7	0.31	0.13	0.18	126
accuracy			0.66	5067

Рис. 5.4. Результат навчання

Найкраща точність: 65.92%.

Коефіцієнт навчання: 0.001 (див. рис. 5.5-5.6).

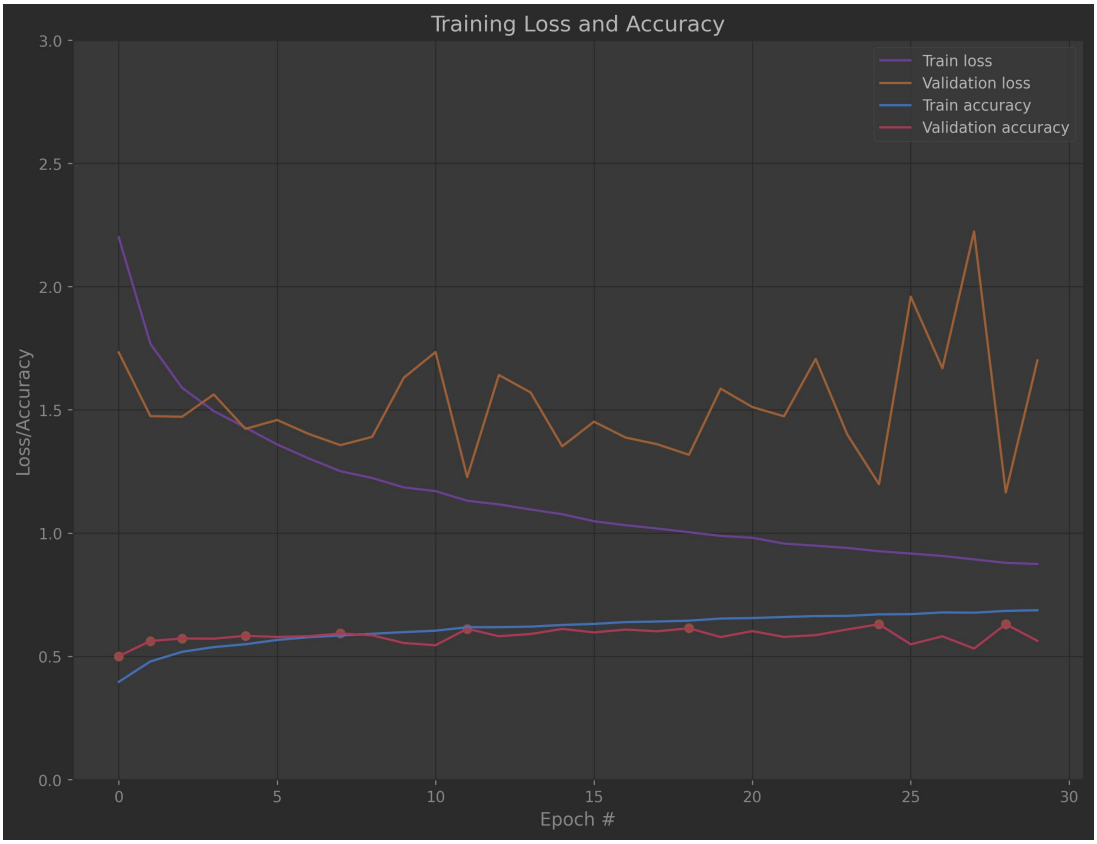

Рис. 5.5. Графік навчання

	precision	recall	f1-score	support
0	0.57	0.36	0.44	904
1	0.82	0.73	0.77	2575
2	0.47	0.42	0.44	665
3	0.15	0.40	0.22	173
4	0.28	0.49	0.35	525
5	0.15	0.04	0.07	48
6	0.56	0.35	0.43	51
7	0.14	0.21	0.17	126
accuracy			0.56	5067

Рис. 5.6. Результат навчання

Найкраща точність: 63.00%.

Коефіцієнт навчання: 0.0001 (див. рис. 5.7-5.8).

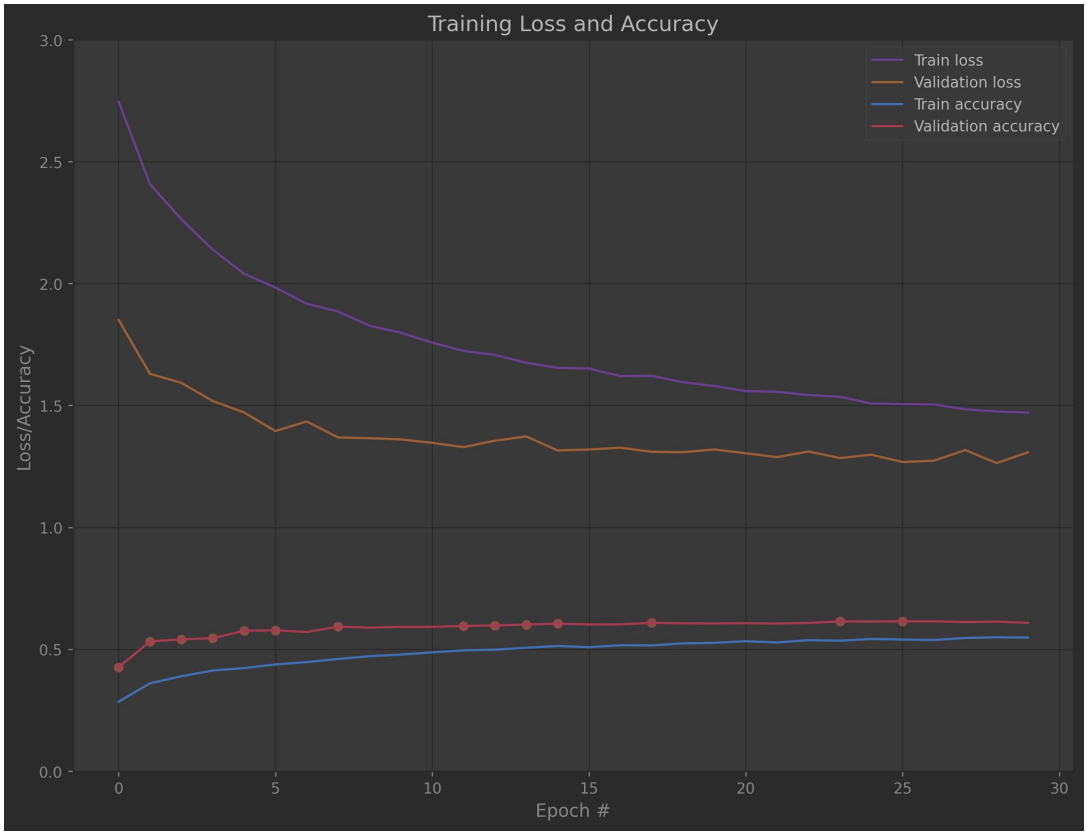

Рис. 5.7. Графік навчання

	precision	recall	f1-score	support
0	0.64	0.24	0.34	904
1	0.73	0.87	0.79	2575
2	0.39	0.72	0.50	665
3	0.21	0.06	0.10	173
4	0.38	0.22	0.28	525
5	0.00	0.00	0.00	48
6	0.38	0.12	0.18	51
7	0.12	0.02	0.04	126
accuracy			0.61	5067

Рис. 5.8. Результат навчання

Найкраща точність: 61.44%.

Коефіцієнт навчання: 0.00001 (див. рис. 5.9-5.10)

Рис. 5.9. Графік навчання

	precision	recall	f1-score	support
0	0.58	0.17	0.27	904
1	0.81	0.65	0.72	2575
2	0.29	0.80	0.43	665
3	0.09	0.05	0.06	173
4	0.21	0.31	0.25	525
5	0.00	0.00	0.00	48
6	0.00	0.00	0.00	51
7	0.05	0.01	0.01	126
accuracy			0.50	5067

Рис. 5.10. Результат навчання

Найкраща точність: 50.56%.

5.2 Вплив розміру групи (batch size) на результати навчання

Використовуючи ті ж самі параметри, тільки з коефіцієнтом навчання 0.001 та кількістю епох 20 досліджуємо вплив розміру групи.

Головне, на що повинен впливати цей параметр, це швидкість роботи та збалансованість та більш стабільне зростання точності. Під стабільністю мається на увазі відсутність різких спадів та підйомів з кожною групою.

Розмір групи: 5 (див. рис. 5.11).

Час виконання: 22:28.

Найкраща точність: 52.85%.

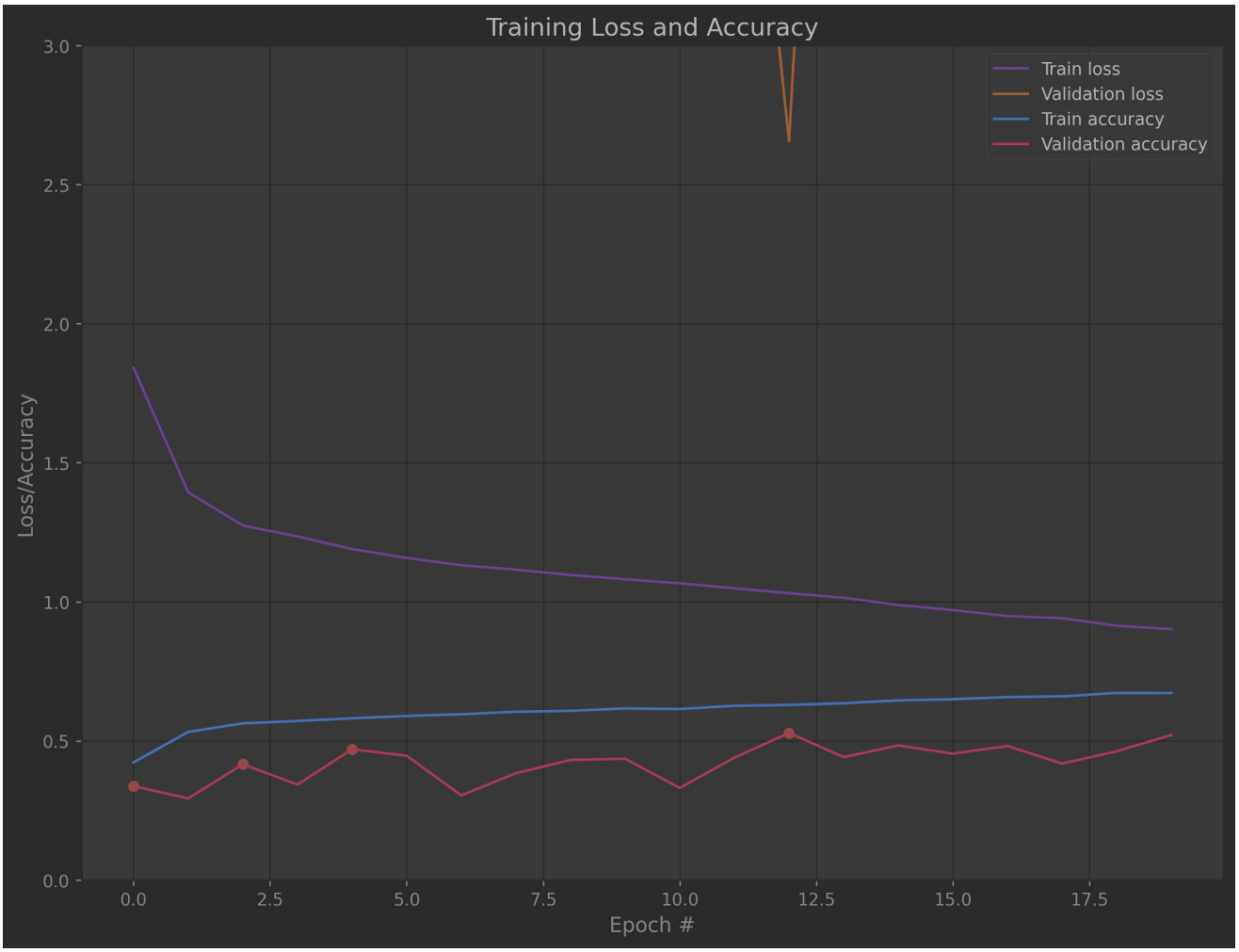

Рис. 5.11. Графік навчання

Розмір групи: 25 (див. рис. 5.12).

Час виконання: 11:24.

Найкращий результат точності: 60.79%.

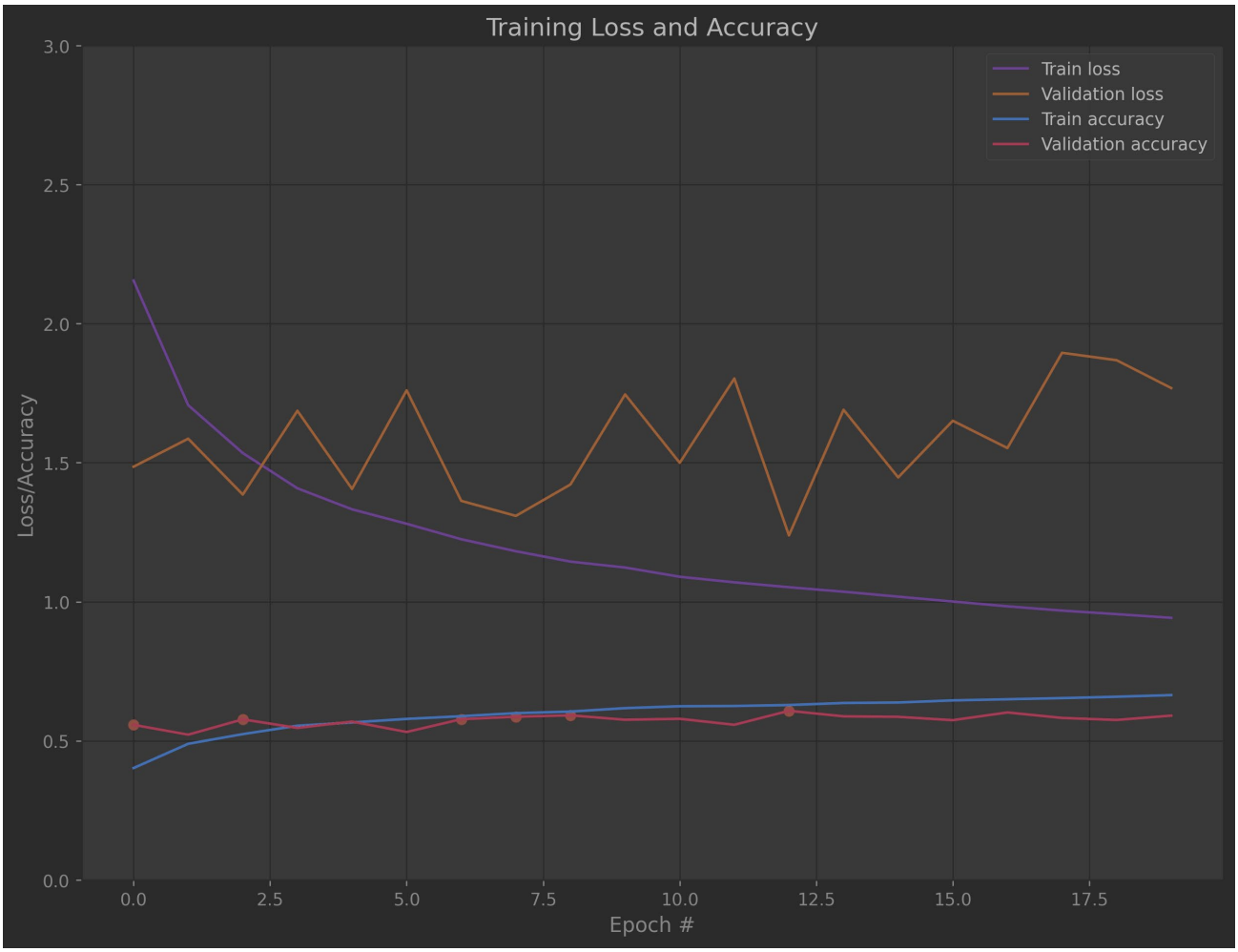

Рис. 5.12. Графік навчання

Розмір групи: 50 (див. рис. 5.13).

Час виконання: 10:29.

Найкраща точність: 62.76%.

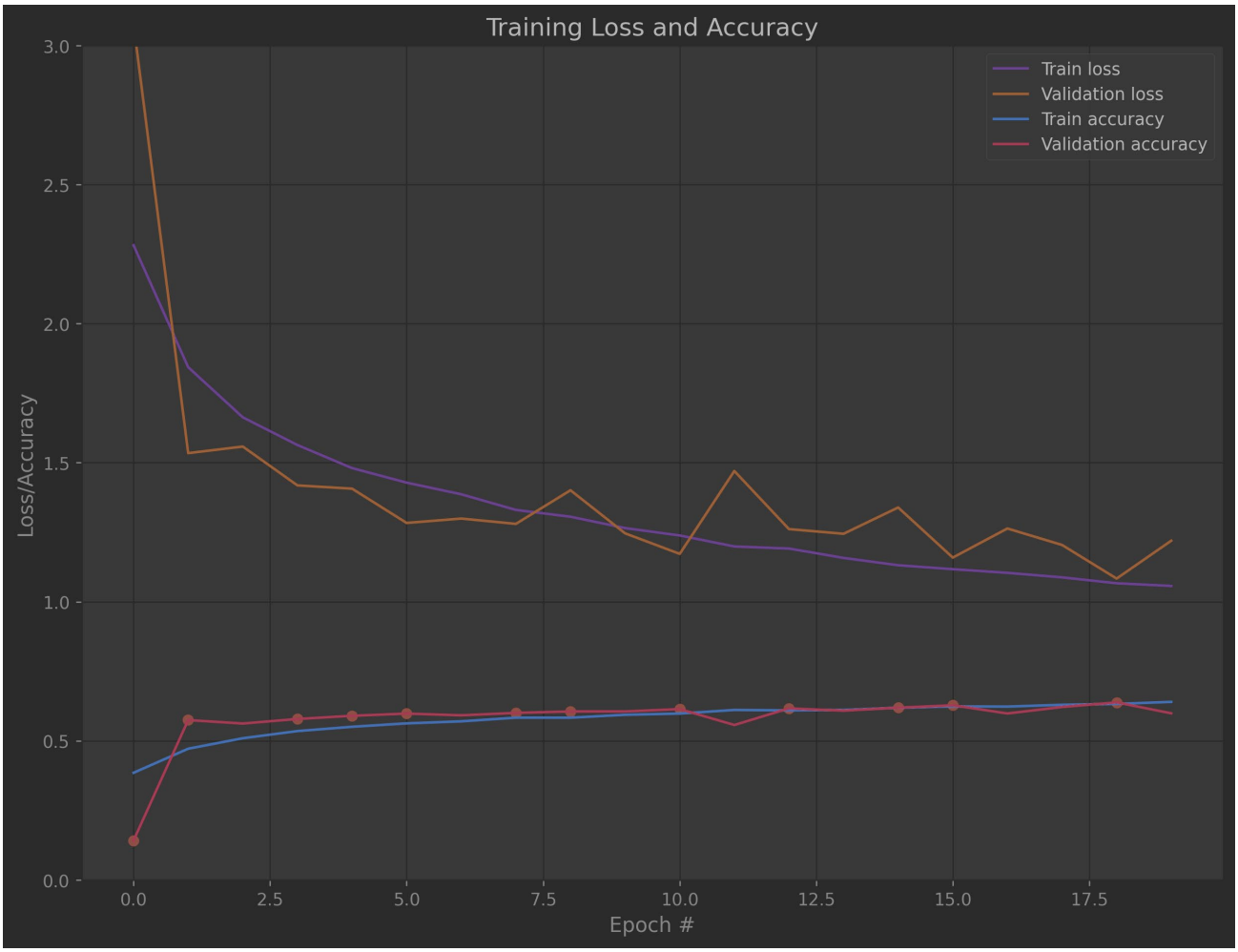

Рис. 5.13. Графік навчання

Розмір групи: 100 (див. рис. 5.14).

Час виконання: 10:34.

Найкраща точність: 62.48%.

Рис. 5.14. Графік навчання

Згідно отриманих результатів можна побачити, що розмір групи ніяк не впливає на точність при кількості елементів в групі більше ніж 25, все в межах звичайного відхилення від норми. Але для 5 точність дійшла до 52.85% і перестала зростати.

Разом з цим швидкість роботи виростає. Так наприклад, якщо розмір групи 5, то час виконання 20 епох буде 22 хвилини, а якщо 25 – 10 хвилин для тих же 20 епох. Через обмеженість апаратних пристроїв швидкість не відрізняється для 25, 50 та 100 елементів в групі. Але на більш потужних пристроях швидкість навчання буде швидше для більших значень кількості елементів в групі.

Висновки до розділу 5

Проведено аналіз впливу кількості елементів в групі та коефіцієнту навчання на результати навчання.

ВИСНОВКИ

Під час виконання магістерської кваліфікаційної роботи було проведено роботу зі створення автоматизованої системи діагностування хвороб шкіри. Для цього проведено аналіз останніх досліджень та публікацій в цьому напрямку, досліджено основні напрямки розвитку цієї теми.

Для створення штучної нейронної мережі, яка зможе вирішити поставлену задачу, налаштовано середу розробки з використанням Python, Jupyter та PyCharm. Встановлено та налаштовано найсучасніші бібліотеки для роботи з матрицями, наборами даних, зображеннями та штучними нейронними мережами.

Проведено дослідження набору даних, його основні характеристики, опис класів. Для роботи з будь-якими типами штучних нейронних мереж використовується розроблена система масштабування зображень. Використано генерування додаткових даних. Оброблені дані збережено в спеціальному форматі HDF5, для того щоб мати можливість навчання на пристроях з маленькими системними вимогами. Зроблено розподіл даних для тестування та навчання.

Оскільки використовується користувацьке налаштування моделей та їх параметрів, то виконано опис усіх основних шарів, які використовуються для будування архітектурних рішень. Через обмеженість апаратних пристроїв не має можливості використовувати повні, або більш глибокі моделі, тому в навчанні використовувались не дуже глибокі моделі, а як у випадку з AlexNet та VGG16 то взагалі зменшені. Було навчено MiniVGG (+ модифікований варіант), AlexNet та VGG16 з використанням різних параметрів навчання. Серед кращих результатів 2 моделі AlexNet (0.01) та MiniVGG (0.001), вони змогли досягти 68% точності. Але оптимальною є MiniVGG, адже вона не така глибока та вхідне зображення менше, а це забезпечує більш швидку роботу. Також проведено дослідження впливів коефіцієнту навчання та розміру групи на швидкість та стабільність навчання.

Нажаль обрані варіанти мереж не змогли досягти достатньої точності для використання в реальному світі. Як варіанти для покращення точності потрібно використовувати більш глибокі мережі та можливо більш складну генерацію даних, що потребує набагато потужнішого апаратного забезпечення, на

поточному це неможливо зробити, навіть маленькі мережі потребують забагато часу на навчання. Також сучасним варіантом вирішення цієї проблеми є побудова маски хвороби, тобто перед класифікацією проводиться знаходження точної позиції ділянки шкіри з хворобою, а лише після цього йде використання згорткової мережі для обраного регіону. Саме таки метод допоможе досягти набагато кращих результатів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Їжов А., Чечеткін В. Нейроні мережі в медицині *OSP - Гід по технологіям цифрової трансформації* : наукова стаття. URL: https://www.osp.ru/os/1997/04/179201 (дата звернення: 23.11.2021).

2. Queen M. Bigger not necessarily better, when it comes to brains *Your source for the latest research news* : наукова стаття. URL: https://www.sciencedaily.com/releases/2009/11/091117124009.htm (дата звернення: 24.11.2021).

3. Гусев А. Искусственный интеллект в медицине *Система предиктивной аналитики и управления рисками в медицине на основе искусственного интеллекта Webiomed* : наукова стаття. URL: https://webiomed.ai/blog/iskusstvennyi-intellekt-v-meditsine/ (дата звернення: 25.11.2021).

4. Каспаров против Deep Blue | Матч, изменивший историю *Chess.com – шахматы* : стаття. URL: https://www.chess.com/ru/article/view/kasparov-protiv-deep-blue-match-izmenivshii-istoriiu#kasparov-deep-blue-1997-rematch (дата звернення: 25.11.2021).

5. ISIC Challenge *The International Skin Imaging Collaboration (ISIC)* : офіційний сайт. URL: https://challenge.isic-archive.com/ (дата звернення: 26.11.2021).

6. Synthetic Skin Cancer Image Data Generation Using Generative Adversarial Neural Network International Journal of Multidisciplinary Studies and Innovative Technologies » Submission » Synthetic Skin Cancer Image Data Generation Using Generative Adversarial Neural Network : наукова стаття. URL: https://dergipark.org.tr/en/pub/ijmsit/issue/65832/1010638 (дата звернення 29.11.2021).

7. Analysis of the ISIC image datasets: Usage, benchmarks and recommendations. *ScienceDirect Journals and Books* : наукова стаття. URL:

https://www.sciencedirect.com/science/article/pii/S1361841521003509 (дата звернення 29.11.2021).

8. Deep Hybrid Convolutional Neural Network for Segmentation of Melanoma Skin Lesion *Publishing Open Access research journals & papers | Hindawi* : наукова стаття. URL: https://www.hindawi.com/journals/cin/2021/9409508/ (дата звернення 30.11.2021).

9. Deep Learning-Based Skin Lesion Diagnosis Model Using Dermoscopic Images *Tech Science Press* : наукова стаття. URL: https://www.researchgate.net/profile/Vinay-Nassa/publication/354378696_Deep_Learning-Based_Skin_Lesion_Diagnosis_Model_Using_Dermoscopic_Images/links/613579a4c69a4e48798142a6/Deep-Learning-Based-Skin-Lesion-Diagnosis-Model-Using-Dermoscopic-Images.pdf (дата звернення 31.11.2021).

10. Lesion Detection on Skin Images Using Improved U-Net *5th International Students Science Congress* : наукова стаття. URL: https://www.researchgate.net/profile/Elif-Uenlue/publication/354197957_Lesion_Detection_on_Skin_Images_Using_Improved_U-Net/links/612dfbd838818c2eaf707ce1/Lesion-Detection-on-Skin-Images-Using-Improved-U-Net.pdf (дата звернення 01.12.2021).

11. Melatect: A Machine Learning Model Approach For Identifying Malignant Melanoma in Skin Growths *Cornel university* : наукова стаття. URL: https://arxiv.org/abs/2109.03310 (дата звернення 03.12.2021).

12. Data Science Programming Languages *Flatiron School* : наукова стаття. URL: https://flatironschool.com/blog/data-science-programming-languages (дата звернення 05.01.2022).

13. Python programming language *Python Language* : опис технології. URL: https://rb.gy/5fz6aj (дата звернення: 06.01.2022).

14. JupyterLab: A Next-Generation Notebook Interface *Jupyter technologies* : офіційна сторінка. URL: https://jupyter.org/ (дата звернення: 07.01.2022).

15. PyCharm scientific mode *PyCharm scientific tools* : опис технології. URL: https://www.jetbrains.com/help/pycharm/matplotlib-tutorial.html (дата звернення: 08.01.2022).

16. Numpy NumPy *The fundamental package for scientific computing with Python* : документація. URL: https://numpy.org/ (дата звернення: 09.01.2022).

17. Scikit-learn Machine learning in Python *Simple and efficient tools for predictive data analysis* : опис технології. URL: https://scikit-learn.org/stable/index.html (дата звернення: 10.01.2022).

18. OpenCV (Open Source Computer Vision Library) *OpenCV* : опис технології. URL: https://docs.opencv.org/4.5.5/d1/dfb/intro.html (дата звернення: 11.01.2022).

19. TensorFlow Core *TensorFlow documentation* : опис технології. URL: https://www.tensorflow.org/guide (дата звернення: 12.01.2022).

20. Keras. Simple. Flexible. Powerful. *Keras Official Site* : опис технології. URL: https://keras.io/ (дата звернення: 13.01.2022).

21. Skin Lesion Images for Melanoma Classification *Kaggle* : набір даних. URL: https://www.kaggle.com/andrewmvd/isic-2019 (дата звернення: 14.01.2022).

22. The HDF5® Library & File Format *The HDF Group* : опис стандарту. URL: https://www.hdfgroup.org/solutions/hdf5 (дата звернення: 15.01.2022).

23. What is Data Augmentation? Techniques, Benefit & Examples *AI Multiple* : стаття. URL: https://research.aimultiple.com/data-augmentation/ (дата звернення: 15.01.2022).

24. Only Numpy: Implementing Mini VGG (VGG 7) and SoftMax Layer with Interactive Code *Towards Data Science* : стаття. URL: https://towardsdatascience.com/only-numpy-implementing-mini-vgg-vgg-7-and-softmax-layer-with-interactive-code-8994719bcca8 (дата звернення: 16.01.2022).

25. Alex Krizhevsky, Ilya Seuskever, Geoffrey E. Hinton ImageNet Classification with Deep Convolutional Neural Networks : наукова стаття. URL:

https://www.nvidia.cn/content/tesla/pdf/machine-learning/imagenet-classification-with-deep-convolutional-nn.pdf (дата звернення: 17.01.2022).

26. ImageNet *ImageNet database and challenges* : опис змагання. URL: https://www.image-net.org/index.php (дата звернення: 18.01.2022).

27. Karen Simonyan, Andrew Zisserman Very Deep Convolutional Networks for Large-Scale Image Recognition : наукова стаття. URL: https://arxiv.org/abs/1409.1556 (дата звернення: 19.01.2022).

28. Рашид Т. Создаем нейронную сеть: навчальний посібник / пер. с англ. А. Г. Гузикевич. СПб, 2017, 272 с.

29. Державні санітарні правили і норми ДСанПіН 5.5.6.009-98 Влаштування і обладнання кабінетів комп'ютерної техніки в навчальних закладах та режим праці учнів на персональних комп'ютерах. Державні санітарні правила та норми(згідно з наказом від 30.12.1998 N 9).

30. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПІН 3.3.2.007-98(згідно з наказом від 10.12.98 № 7).

31. Гігієнічна класифікація праці за показниками шкідливості та небезпечності факторів виробничого середовища, важкості та напруженості трудового процесу //Охорона праці. – 2001. –№ 12. – С. 12-20.

32. Жидецький В.Ц. Основи охорони праці: Підручник. – Львів: УАД, 2006. – 336 с.

33. НПАОП 0.00-1.28-10. Правила охорони праці під час експлуатації електронно-обчислювальних машин;

34. Голінько В.І. Охорона праці в галузі інформаційних технологій : навч. посіб. /В.І. Голінько, М.Ю. Іконніков, Я.Я. Лебедєв ; М-во освіти і науки України, Нац. гірн. ун-т. – Д. : НГУ, 2015. – 246 с.

ДОДАТОК А
Розміри зображень

Формат запису – (ширина, довжина, глибина) : кількість зображень

(1024, 1024, 3) : 12414
(450, 600, 3) : 10015
(680, 1024, 3) : 1121
(768, 1024, 3) : 774
(682, 1024, 3) : 173
(685, 1024, 3) : 156
(764, 1024, 3) : 81
(681, 1024, 3) : 78
(576, 767, 3) : 68
(679, 1024, 3) : 68
(802, 919, 3) : 54
(683, 1024, 3) : 40
(684, 1024, 3) : 36
(677, 1024, 3) : 33
(767, 1022, 3) : 22
(678, 1024, 3) : 21
(686, 1024, 3) : 17
(719, 824, 3) : 16
(676, 1024, 3) : 14
(716, 1024, 3) : 10
(674, 962, 3) : 8
(542, 722, 3) : 7
(674, 1024, 3) : 6
(768, 576, 3) : 5
(673, 1024, 3) : 4
(750, 1024, 3) : 4
(722, 962, 3) : 3
(717, 1019, 3) : 3
(689, 1024, 3) : 3
(638, 959, 3) : 2
(675, 1024, 3) : 2
(542, 718, 3) : 2
(692, 1024, 3) : 2
(687, 1024, 3) : 2
(769, 1024, 3) : 1
(629, 963, 3) : 1
(724, 965, 3) : 1
(540, 722, 3) : 1
(640, 964, 3) : 1
(649, 965, 3) : 1
(649, 961, 3) : 1
(624, 965, 3) : 1
(720, 964, 3) : 1
(545, 722, 3) : 1
(724, 960, 3) : 1
(638, 966, 3) : 1
(1024, 857, 3) : 1
(711, 1024, 3) : 1
(669, 1024, 3) : 1
(690, 1024, 3) : 1
(706, 1024, 3) : 1
(543, 722, 3) : 1
(672, 1024, 3) : 1
(710, 1024, 3) : 1
(542, 725, 3) : 1
(670, 962, 3) : 1
(715, 1024, 3) : 1
(699, 1024, 3) : 1
(642, 958, 3) : 1
(626, 949, 3) : 1
(671, 962, 3) : 1
(645, 966, 3) : 1
(634, 959, 3) : 1
(640, 957, 3) : 1
(723, 957, 3) : 1
(672, 964, 3) : 1
(723, 959, 3) : 1
(639, 959, 3) : 1
(641, 962, 3) : 1
(723, 962, 3) : 1
(645, 960, 3) : 1
(676, 962, 3) : 1
(542, 724, 3) : 1
(782, 1024, 3) : 1
(858, 1024, 3) : 1
(930, 1024, 3) : 1
(946, 1024, 3) : 1
(914, 1024, 3) : 1
(961, 1024, 3) : 1
(830, 1024, 3) : 1
(798, 1024, 3) : 1
(751, 1024, 3) : 1
(878, 1024, 3) : 1
(861, 1024, 3) : 1
(995, 1024, 3) : 1
(818, 1024, 3) : 1
(711, 1007, 3) : 1
(713, 1011, 3) : 1
(711, 1008, 3) : 1
(682, 1016, 3) : 1
(682, 796, 3) : 1
(680, 853, 3) : 1
(750, 771, 3) : 1
(614, 838, 3) : 1
(566, 679, 3) : 1
(680, 833, 3) : 1
(602, 639, 3) : 1
(717, 1017, 3) : 1
(704, 1007, 3) : 1
(671, 1024, 3) : 1
(688, 1024, 3) : 1

ДОДАТОК Б
Лістинг коду

```
import csv
import math
import os

import cv2
import h5py as h5
import numpy as np
import tensorflow as tf
import tensorflow.keras.utils
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelBinarizer
from tensorflow.keras import Sequential, layers
from tensorflow.keras.applications import imagenet_utils
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.callbacks import ModelCheckpoint, BaseLogger, LearningRateScheduler
from tensorflow.keras.layers import BatchNormalization, Conv2D, MaxPooling2D, Activation, Flatten, Dropout,
Dense
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import load_model

from sources import PathsManager, ImagesUtils, FixGPU, PlotManager, MiniVGGNet, TrainingLogger
from sources.utils.progress_bar import ProgressBar

class H5DataGen(tf.keras.utils.Sequence):
    def __init__(self,
                 x_dataset: h5.Dataset,
                 y_dataset: h5.Dataset = None,
                 batch_size: int = 32,
                 use_generator: bool = False):
        self.x_dataset = x_dataset
        self.y_dataset = y_dataset
        self.batch_size = batch_size
        self.i = 0
        self.use_generator = use_generator
        if y_dataset is not None:
            self.with_y = True
```

```
        else:
            self.with_y = False
        self.generator = ImageDataGenerator(
            rotation_range=180,
            width_shift_range=0.3,
            height_shift_range=0.3,
            shear_range=5,
            zoom_range=0.2,
            horizontal_flip=True,
            fill_mode="nearest"
        )

    def __len__(self):
        return math.ceil(self.x_dataset.shape[0] / self.batch_size)

    def __getitem__(self, position):
        current_batch = position * self.batch_size
        x_part = self.x_dataset[current_batch:current_batch + self.batch_size]
        y_part = None
        if self.with_y:
            y_part = self.y_dataset[current_batch:current_batch + self.batch_size]

        if self.use_generator:
            flow = self.generator.flow(x_part, y_part, batch_size=self.batch_size)
            x = flow.x[:self.batch_size]
            if self.with_y:
                y = flow.y[:self.batch_size]
            else:
                raise IOError("WTF?!")
        else:
            x = x_part
            y = y_part
        if self.with_y:
            return x, y
        else:
            return x


def load_data(for_feature_extraction: bool = False):
    files_info = load_files_info()
```

```
    files_paths = PathsManager.images_list("../data/skin/input")

    images = []
    labels = []
    ProgressBar.create(len(files_paths), "Loading images")

    for (i, path) in enumerate(files_paths):
        img = cv2.imread("../data/skin/input/" + path)
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        if for_feature_extraction:
            img = ImagesUtils.resize_smallest_size_and_crop(img, (224, 224))
            img = imagenet_utils.preprocess_input(img)
        else:
            img = ImagesUtils.resize_smallest_size_and_crop(img, (IMG_SIZE, IMG_SIZE))
        images.append(img)
        labels.append(files_info[path])
        ProgressBar.increment()

        # if i == 1000:
        #     break

    ProgressBar.finish()
    images = np.array(images, dtype=np.float16) / 255.0
    labels = np.array(labels)
    print(f"Images array in memory: {images.nbytes / (1024 * 1000.0):.2f}MB")
    return images, labels


def load_split_data():
    path = MODELS + f"/skin{IMG_SIZE}.hdf5"
    if os.path.exists(path):
        print(f"File skin{IMG_SIZE}.hdf5 already exists")
        return h5.File(path)
    else:
        print("Loading new data...")
        (x, y) = load_data()
        (x_train, x_test, y_train, y_test) = train_test_split(
            x,
            y,
            test_size=0.20,
```

```
        random_state=42,
        stratify=y
    )
    binarizer = LabelBinarizer().fit(np.array([0, 1, 2, 3, 4, 5, 6, 7]))
    y_train = binarizer.transform(y_train)
    y_test = binarizer.transform(y_test)

    file = h5.File(MODELS + f"/skin{IMG_SIZE}.hdf5", "w")
    file.create_dataset("x_train", data=x_train)
    file.create_dataset("y_train", data=y_train)
    file.create_dataset("x_test", data=x_test)
    file.create_dataset("y_test", data=y_test)
    return file


def extract_features(dataset: h5.Dataset, file: h5.File, dataset_name: str):
    batch_size = 8
    vgg16 = VGG16(weights="imagenet", include_top=False)
    features_dataset = file.create_dataset(dataset_name, (dataset.shape[0], 7 * 7 * 512))
    for i in range(0, dataset.shape[0], batch_size):
        images = dataset[i:i + batch_size]
        features = vgg16.predict(images, batch_size=batch_size)
        features = features.reshape((features.shape[0], 7 * 7 * 512))
        features_dataset[i:i + batch_size] = features
    return features_dataset


def load_features_data():
    print("Loading new features...")
    (x, y) = load_data(for_feature_extraction=True)
    (x_train, x_test, y_train, y_test) = train_test_split(
        x,
        y,
        test_size=0.20,
        random_state=42,
        stratify=y
    )
    temp_file = h5.File(MODELS + "/temp_features_file.hdf5", "w")
    x_train = temp_file.create_dataset("x_train", data=x_train)
    y_train = temp_file.create_dataset("y_train", data=y_train)
```

```
    x_test = temp_file.create_dataset("x_test", data=x_test)
    y_test = temp_file.create_dataset("y_test", data=y_test)

    ### BE CAREFUL! This data doesn't use LabelBinarizer

    file = h5.File(MODELS + "/skin_features.hdf5", "w")
    extract_features(x_train, file, "x_train")
    extract_features(x_test, file, "x_test")
    file.create_dataset("y_train", data=y_train)
    file.create_dataset("y_test", data=y_test)

    del temp_file

    return file


def get_class_weights():
    return {
        0: 12875.0 / 4522,
        1: 12875.0 / 12875,
        2: 12875.0 / 3323,
        3: 12875.0 / 867,
        4: 12875.0 / 2624,
        5: 12875.0 / 239,
        6: 12875.0 / 253,
        7: 12875.0 / 628,
    }

print("Creating model...")
model = create_custom_model(IMG_SIZE, IMG_SIZE, 3, 8)
model.compile(
    loss="categorical_crossentropy",
    optimizer=SGD(learning_rate=LEARNING_RATE),
    # optimizer=tf.keras.optimizers.SGD(learning_rate=LEARNING_RATE),
    # optimizer=tf.keras.optimizers.Adam(),
    metrics=["accuracy"]
)

checkpointer = ModelCheckpoint(
    filepath=MODELS + "/networks/skin/" + MODEL_NAME + ".hdf5",
```

```
    # filepath=MODELS + "/networks/" + MODEL_NAME + "-{epoch:03d}-{val_accuracy:0.3f}.hdf5",
    monitor="val_accuracy",
    mode="max",
    save_best_only=True,
    verbose=0,
)


def step_decay(epoch):
    init_alpha = LEARNING_RATE
    factor = DROP_FACTOR
    drop_every = DROP_EVERY
    alpha = init_alpha * (factor ** np.floor((1 + epoch) / drop_every))
    return float(alpha)


scheduler = LearningRateScheduler(step_decay)

ProgressBar.create(EPOCHS)
history = model.fit(
    x=H5DataGen(x_train, y_train, batch_size=BATCH_SIZE, use_generator=True),
    validation_data=H5DataGen(x_test, y_test, BATCH_SIZE),
    steps_per_epoch=math.ceil(x_train.shape[0] / BATCH_SIZE),
    batch_size=BATCH_SIZE,
    epochs=EPOCHS,
    verbose=0,
    # class_weight=get_class_weights(),
    callbacks=[TrainingLogger(without_progress_bar=True), scheduler, checkpointer]
)
ProgressBar.finish()

# model = load_model(MODELS + "/networks/AlexNet.hdf5")
predictions = model.predict(x_test, batch_size=32)
report = classification_report(
    y_test[:].argmax(axis=1),
    predictions.argmax(axis=1),
    #  target_names=label_binarizer.classes_
)
print(report)
```