НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:
Завідувач кафедри

_____ Сергій СТІРЕНКО
(підпис)

“__” _____2021 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою “Комп’ютерні системи та мережі”

спеціальності 123 “Комп’ютерна інженерія”

на тему: «Система розпізнавання атипових плям на шкірі»

Виконав (-ла): студент (-ка) IV курсу, групи ІВ-71

Музика Олександр Андрійович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник асист. Сергієнко А.А. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант (нормоконтроль) проф., д.т.н. Сімоненко В.П. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Київ – 2021 р.

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалавр)

Освітньо-професійна програма

“Комп’ютерні системи та мережі”

спеціальність 123 “Комп’ютерна інженерія”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Сергій СТІРЕНКО
(підпис)
“__” _____2021 р.

ЗАВДАННЯ

на бакалаврський дипломний проєкт студента

Музики Олександра Андрійовича

1. Тема проєкту *Система розпізнавання атипових плям на шкірі*

керівник проєкту *асистентка Сергієнко Анастасія Анатоліївна*,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від *«11» травня* 2021 року № *1139-с*

2. Термін здачі студентом закінченого проєкту *1 червня 2021 р.*

3. Вихідні дані до проєкту *технічна документація. теоретичні та статистичні дані*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які розробляються) *аналіз предметної області, огляд технологій для розробки системи, реалізація програмного забезпечення, аналіз та тестування програмного продукту.*

5. Перелік графічного матеріалу (з точним позначенням обов'язкових креслень) *блок-схема роботи програми, діаграма класі програми, схема роботи програми, схема архітектури нейронної мережі.*

6. Консультант проєкту, з вказівкою розділів проєкту, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Нормоконтроль	*Сімоненко В. П., професор, д.т.н.*		

7. Дата видачі завдання_____

Календарний план

№ п/п	Найменування етапів дипломного проєкту	Терміни виконання етапів проєкту	Примітки
1.	*Затвердження теми проєкту*	*10.12.2020-15.12.2020*	
2.	*Вивчення та аналіз завдання*	*15.12.2020-15.03.2021*	
3.	*Розробка архітектури та загальної структури системи*	*15.03.2021-25.03.2021*	
4.	*Розробка структур окремих підсистем*	*25.03.2021-5.04.2021*	
5.	*Програмна реалізація системи*	*5.04.2021-15.04.2021*	
6.	*Оформлення пояснювальної записки*	*15.04.2021-20.05.2021*	
7.	*Захист програмного продукту*	*25.04.2021*	
8.	*Передзахист*	*23.05.2021*	
9.	*Захист*	*15.06.2021*	

Студент-дипломник _____
(підпис)

Керівник проєкту _____
(підпис)

АНОТАЦІЯ

В дипломній роботі бакалавра розроблено систему для розпізнавання атипових плям на шкірі за допомогою згорткової нейронної мережі з веб-додатком в якості інтерфейсу для користувача. Для навчання моделі використано датасет HAM10000. Система створена для оцінки ризиків пігментних уражень шкіри, яка може використовуватись медичним персоналом для попередньої діагностики.

Точність моделі на навчальній вибірці склала 76.9%, а точність на тестовій вибірці склала 74.5%. Для оцінки користі моделі необхідно оцінити її на практиці та розробити протоколи використання.

Веб-додаток розроблено на мові програмування Python, за допомогою мікрофреймворку Flask. Для дизайну веб-сторінок використано набір інструментів Bootstrap.

ANNOTATION

In bachelor's degree project, a system for identifying of atypical moles on the skin using a convolutional neural network with a web application as a user interface is developed. The HAM10000 dataset is used to train the model. The system is designed to evaluate the risks of pigmented skin lesions, which can be used by junior medical staff for preliminary diagnosis.

The accuracy of the model in the training sample was 76.9%, and the accuracy in the test sample was 74.5%. To assess the benefits of the model, it is necessary to evaluate it in practice and develop protocols of use.

The web application is developed in the Python programming language, using the Flask microframework. The Bootstrap library is used to design web pages.

Опис альбому

до дипломного проєкту

на тему: «Система розпізнавання атипових плям на шкірі»

Київ – 2021 р.

Справки	Формат	*Значення*	*Найменування*	Кіл. листів	№ екземпляра	*Додаток*
			Документація загальна			
			Знову розроблена			
	А4	***ІАЛЦ.467200.001 ОА***	Система розпізнавання	1		
			атипових плям на шкірі			
			Опис альбому			
	А4	***ІАЛЦ.467200.002 ТЗ***	Система розпізнавання	3		
			атипових плям на шкірі			
			Технічне завдання			
	А4	***ІАЛЦ.467200.003 ПЗ***	Система розпізнавання	64		
			атипових плям на шкірі			
			Пояснювальна записка			
	А4	***ІАЛЦ.467200.004 Д1***	Система розпізнавання	1		
			атипових плям на шкірі			
			Додаток А			
	А4	***ІАЛЦ.467200.005 Д2***	Система розпізнавання	1		
			атипових плям на шкірі			
			Додаток Б			
	А4	***ІАЛЦ.467200.006 Д3***	Система розпізнавання	1		
			атипових плям на шкірі			
			Додаток В			
	А4	***ІАЛЦ.467200.007 Д4***	Система розпізнавання	4		
			атипових плям на шкірі			
			Додаток Г			

					ІАЛЦ.467200.001 ОА
Зм.	*Арк.*	*№ докум.*	*Підп.*	*Дата*	
Розроб.		*Музика О.А.*			*Система розпізнавання атипових плям на шкірі* *Опис альбому*
Перев.		*Сергієнко А.А.*			
Н.контр.		*Сімоненко В.П.*			
Затв.		*Сергієнко А.А.*			

Літ.	*Аркуш*	*Аркушів*
Т	*1*	

НТУУ «КПІ ім. Ігоря Сікорського» ФІОТ
Група ІВ-71

Технічне завдання

до дипломного проєкту

на тему: «Система розпізнавання атипових плям на шкірі»

Київ – 2021 р.

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Найменування: система розпізнавання атипових плям на шкірі.

Область застосування: система широкого використання застосування для оцінки ризиків пігментних плям.

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання роботи кваліфікаційно-освітнього рівня бакалавр «комп'ютерної інженерії», затверджене кафедрою обчислювальної техніки Національного технічного Університету України «Київський Політехнічний Інститут імені Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ

Метою даної проектної роботи є побудова й навчання нейронної мережі для класифікації атипових плям на шкірі та розробка веб-додатку для зручного використання моделі користувачами.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом розробки є науково-технічна література з теорії та практики аналізу даних та комп'ютерного зору, технічна документація, публікації в періодичних виданнях, довідники, публікації в Інтернеті з даних питань.

5. ТЕХНІЧНІ ВИМОГИ

5.1 Вимоги продукту розробки

- Зручний інтерфейс для використання алгоритму класифікації
- Швидке розгортання системи на хості

- Легка інтеграція моделі в систему в разі покращення її якості

5.2 Вимоги програмного забезпечення

- MS Windows 7/8/10 або Linux OS
- Python 3.7.2 або вище
- Docker
- Веб-браузер з графічним інтерфейсом

5.3 Вимоги апаратної частини

- 32-розрядний (x86) або 64-розрядний (x64) процесор із тактовою частотою 1 ГГц або швидший
- Вільний простір жорсткого диску 4 ГБ або більше
- Оперативна пам'ять 2 ГБ або більше

6. ЕТАПИ РОЗРОБКИ

Етап	Дата
Затвердження теми проєкту	15.12.2020
Вивчення та аналіз завдання	15.03.2021
Розробка архітектури та загальної структури системи	25.03.2021
Розробка структур окремих підсистем	05.04.2021
Програмна реалізація системи	15.04.2021
Оформлення пояснювальної записки	21.05.2021

Пояснювальна записка

до дипломного проєкту

на тему: «Система розпізнавання атипових плям на шкірі»

Київ – 2021 р.

ЗМІСТ

СПИСОК УМОВНИХ СКОРОЧЕНЬ

ANN	Artificial Neural Networks – штучні нейронні мережі
CNN	Convolutional Neural Networks – згорткові нейронні мережі
R-CNN	Region Based Convolutional Neural Networks
ReLU	Rectified linear unit – функція активації нейрону
IDE	Integrated development environment

ВСТУП

Інформаційні технології широко використовуються в медичній практиці: організація та оптимізація роботи закладів охорони здоров'я, програмне забезпечення для медичного обладнання, системи обробки та зберігання медичної інформації, застосування обчислювальних можливостей для проведення клінічних досліджень тощо.

Серед таких задач, над вирішенням яких досі працюють дослідники, знаходиться аналіз медичних зображень — мікроскопічні зображення тканин та клітин людини, макрофотографії тканин та патологій, томографічні знімки тощо. Застосування аналізу медичних зображень може покращити якість діагностики та зменшити ймовірність помилки, адже при оцінці даних спеціалістом класифікація зображень за допомогою алгоритмів машинного навчання може виступати альтернативною думкою (а в разі досягнення високої точності може автоматизувати певні етапи діагностики).

Однією з можливих проблем, навколо якої сконцентрована дана робота, є аналіз дерматологічних зображень.

Дерматоскопія — неінвазійна медична процедура, під час якої за допомогою спеціального інструменту, дерматоскопу, отримуються зображення (або ураження оцінюються безпосередньо крізь збільшувальне скло), які аналізує лікар-дерматолог. Дерматоскопія більш точний метод діагностики ніж клінічне обстеження неозброєним оком [1]. Дана процедура так широко застосовується тому що це оптимальний метод (з точки зору співвідношення точності та ресурсозатратності) для онкологічного скринінгу шкіри — вчасного виявлення меланоми, злоякісного новоутворення на шкірі меланоцитарного генезу. Також дана процедура стає в нагоді для діагностики багатьох інших шкірних захворювань та пігментних уражень.

Метою даної роботи є розробка алгоритму розпізнавання пігментних новоутворень та створення зручного інтерфейсу для користування системою.

Розпізнавання образів здійснюється за допомогою згорткових нейронних мереж, а в якості інтерфейсу для користувача створено клієнт-серверний вебзастосунок. Дані передаються в безпечному вигляді і після відповідних обчислень видаляються на стороні сервера.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Використання штучного інтелекту в охороні здоров'я привертає увагу все дедалі більшої кількості інвесторів, що не дивно, адже цей сектор є перспективним. В 2019 році за підрахунками експертів розмір інвестицій сягнув 4 млрд доларів. [2]

Попри стрімкий розвиток сфери штучного інтелекту, наразі алгоритми машинного навчання здатні виконувати лише доволі вузькі завдання. Тому дослідження проводяться в тісній співпраці фахівців біомедичної сфери, які ставлять вузьку задачу, та фахівців в сфері технічних наук, які будують відповідні математичні моделі та шукають рішення. Зокрема триває пошук застосувань штучного інтелекту для прогнозування перебігу хвороби; прийняття персоналізованих медичних рішень (підбір індивідуальної схеми лікування); діагностики, базуючись на томографічних зображеннях (класифікація та сегментація рентгенівських, КТ-, МРТ-, Пет-КТ-знімків), мікроскопічних зображеннях (визначення відносної лейкоцитарної формули, класифікація та сегментація гістологічних препаратів), об'єктивній інформації (набір об'єктивних мірок стану пацієнта та добре структурованого анамнезу), сигналах датчиків медичної апаратури (зокрема, застосування методів аналізу часових рядів для інтерпретації електрокардіограми, електроенцефалограми) тощо.

В даній роботі буде розглянуто одну з таких проблем — розпізнавання пігментних плям на дерматоскопічних зображеннях, тому даний розділ містить короткий огляд застосування розпізнавання образів в медицині та аналіз основних методів і алгоритмів, які себе добре в цьому показали.

1.1 Огляд основних задач комп'ютерного зору

Комп'ютерний зір — це інтегральна дисципліна, яка стоїть на перетині комп'ютерних наук (теорія алгоритмів, комп'ютерна графіка, чисельні

методи), математики (лінійна алгебра, теорія ймовірності, тензорний аналіз, методи оптимізації, машинне навчання), фізики (оптика) та інших інженерних наук.

Зокрема виділяють наступні популярні задачі комп'ютерного зору:

- класифікація зображень;
- виявлення об'єктів;
- виявлення відношень між об'єктами
- сегментація зображень;
- та інші.

Ці задачі та основні методи їх вирішення буде розглянуто в даному підрозділі.

1.1.1. Класифікація зображень

Класифікації зображень — задача визначення класу зображення з відповідного чітко зазначеного переліку категорій, як, наприклад, показано на рис. 1.1. Значна частина алгоритмів комп'ютерного зору (наприклад виявлення об'єктів) зводиться саме до вирішення цієї задачі.

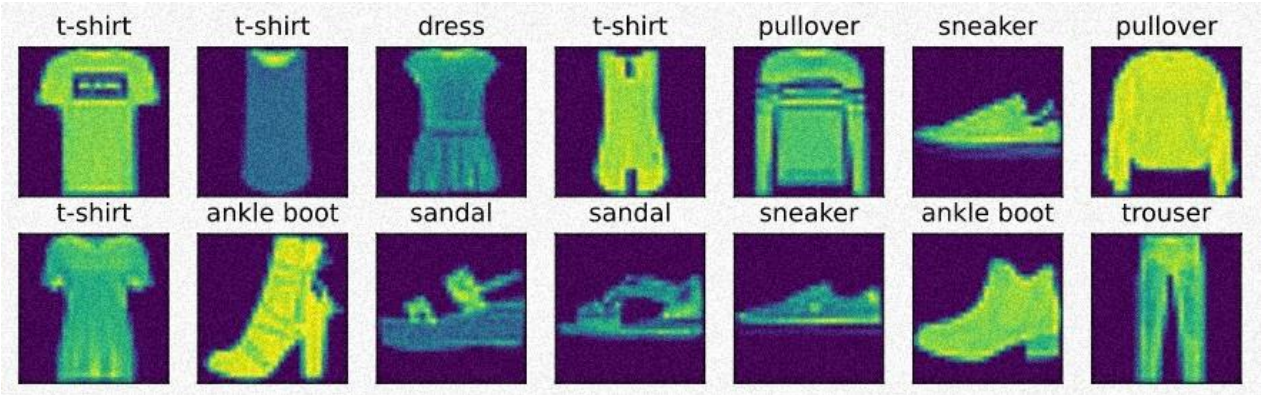


Рис. 1.1 – Класифікація зображень з набору *MNIST «LeCun et al., 1998»* [3]

Нехай X – множина ознак об'єкту, а Y – множина класів. Існує деяке невідоме відображення y: $X \longrightarrow Y$, таке, що пари (x, y) відомі тільки для навчальної вибірки $X^m = \{(x_1, y_1), \ldots, (x_n, y_n)\}$. Вирішення задачі класифікації полягає в побудові алгоритму a: $X \longrightarrow Y$ такого, що здатен класифікувати довільний об'єкт $x \in X$.

Для алгоритму можна визначити функцію помилки, щоб кількісно оцінити наскільки результат класифікації алгоритму відрізняється від справжнього невідомого відображення. Навчання алгоритму полягає в вирішенні задачі мінімізації, де цільовою функцією є функція помилки, а аргументами є параметри алгоритму.

В класифікації зображень елементом множини ознак X є двовимірне растрове зображення, представлене матрицею чисел, що кодують кожен піксель (чорно-біле зображення) або тензором (кольорове зображення).

Виділяють декілька видів класів, до яких може належати об'єкт:

- Класи, що не перетинаються. Об'єкт може одночасно належати лише до одного з класів
- Класи, що перетинаються. Об'єкт може належати одразу до декількох класів
- Нечіткі класи. До кожного класу об'єкт належить лише в певній мірі

За кількістю можливих класів задачу класифікації поділяють на:

- Бінарна класифікація – задача, де можливих класів лише два. Як правило ці два класи шифруються як 0 та 1.
- Багатокласова класифікація – задача, де можливих класів декілька.

Задача класифікації зображень, як і будь-яка інша задача класифікації, може бути вирішена одним із наступних алгоритмів машинного навчання[4]:

- Лінійні класифікатори (логістична регресія, алгоритм «наївний Байєс», перцептрон) – навчання таких алгоритмів доволі швидке, але вони рідко застосовуються в області розпізнавання образів, адже не здатні вивчити складні нелінійні залежності
- Метод опорних векторів
- Ядерні оцінювачі (k найближчих сусідів)
- Нейронні мережі (глибокі нейронні мережі, згорткові нейронні мережі та інші). На даний момент є лідерами в області розпізнавання образів

- Дерева рішень – можуть знаходити найскладніші залежності, однак дуже складно досягти балансу при підборі гіперпараметрів
- Алгоритмічні ансамблі

Такого стрімкого розвитку сфера класифікації зображень зазнала завдяки тому, що зі збільшенням обчислювальних потужностей комп'ютерів стало можливим навчання глибоких нейронних мереж з великою кількістю параметрів, які здатні відтворювати складні нелінійні залежності, на відміну від інших алгоритмів.

1.1.2 Виявлення об'єктів

Виявлення об'єктів – це технологія з області комп'ютерного зору, яка займається виявленням об'єктів на зображенні чи відеоряді, які відносяться до певного класу. Дана технологія застосовується в розпізнаванні облич, аналізі записів відеоспостереження, розпізнаванні об'єктів з датчиків та камер при проєктуванні машин на самокеруванні, аналізі томографічних знімків багажу в прикордонних службах та пропускних пунктах та інші. Як правило, метою вирішення задачі є знаходження меж прямокутника, в якому знаходиться об'єкт, як показано на рис. 1.2.

Методи виявлення об'єктів можна поділити на два класи: ті, що засновані на нейронних мережах і ті, що не засновані на нейронних мережах. Для другого класу важливо перед застосуванням алгоритму провести попередню обробку зображення для виокремлення ознак, які і будуть використовуватись для навчання. Згорткові нейронні мережі, в свою чергу, здатні працювати з необробленими даними.

Алгоритми, засновані на нейронних мережах:

- You Only Look Once (YOLO)
- Деформовані згорткові мережі
- Нейронна мережа одноразового уточнення для виявлення об'єктів (RefineDet)

- Retina-Net

Алгоритми, що не базуються на нейронних мережах:

- Метод Віоли-Джонса
- Масштабно-інваріантне перетворення ознак (SIFT)
- Гістограма орієнтовних градієнтів (HOG)

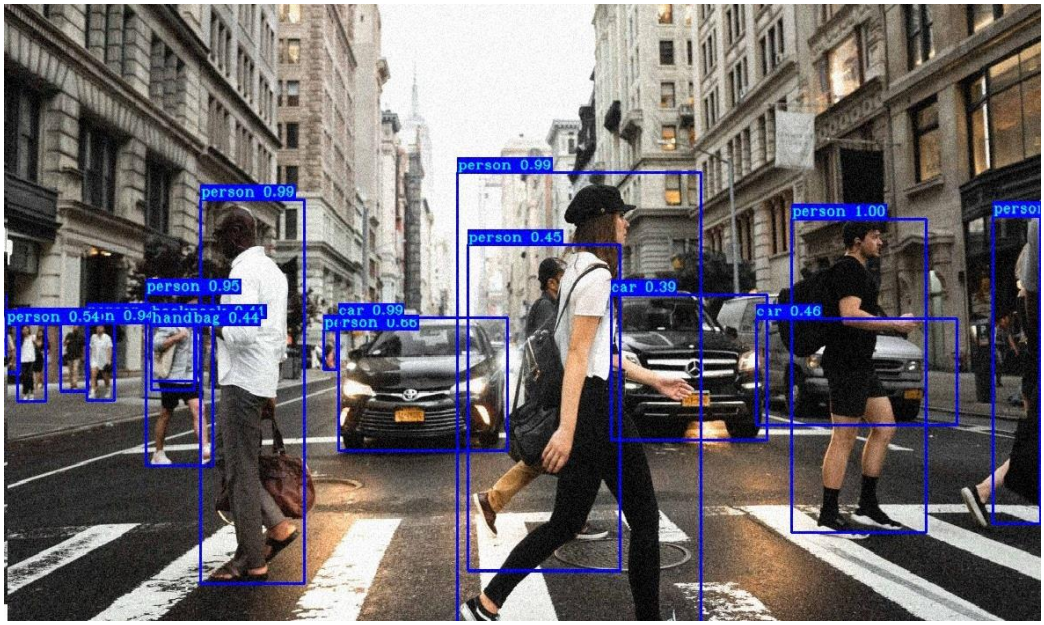


Рис. 1.2 – Об’єкти, виявлені та анотовані за допомогою алгоритму YOLO [5]

1.1.3 Сегментація зображень

Сегментація зображень – це процес розбиття зображення на сегменти (група пікселів, що утворює певний об’єкт на зображенні). Як правило використовується для виявлення об’єктів та отримання чітких контурів. Якщо говорити більш точно, то вирішення задачі сегментації зображення – це визначення, до якого класу відноситься кожен піксель на зображенні. Результатом сегментації є набір сегментів, що покривають ціле зображення, або набір контурів.

Виділяють дві групи сегментації зображень:

- Семантична сегментація, де метою є встановлення до якого класу об’єктів відноситься кожен піксель зображення (рис. 1.3, зліва)
- Сегментація екземплярів, де метою є виділення окремих об’єктів,

незважаючи на те, відносяться вони до одного класу чи до різних (рис. 1.3, справа)

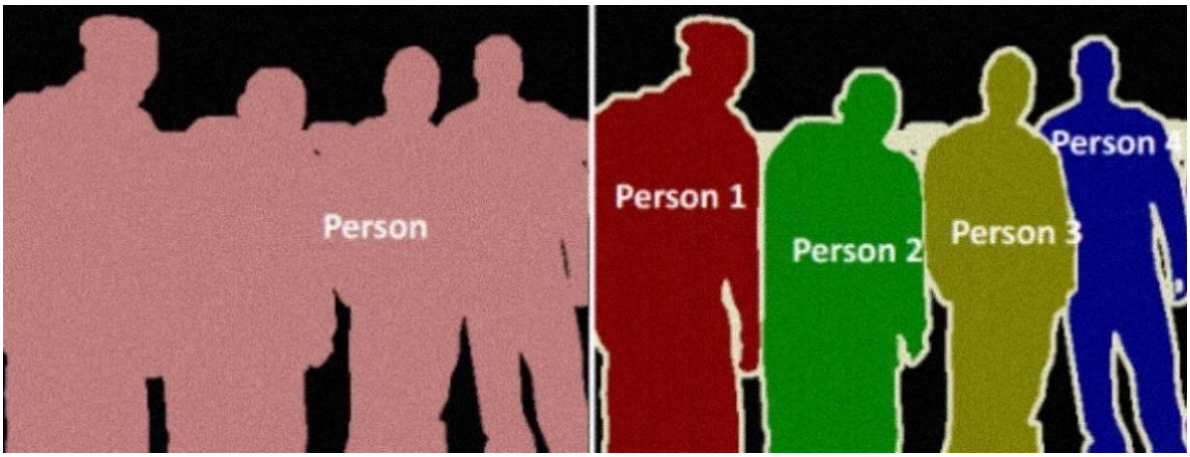


Рис. 1.3 – Групи сегментації зображень: семантична сегментація (зліва) та сегментація екземплярів (справа) [6]

Для сегментації зображень використовуються як традиційні алгоритми комп'ютерного зору так і алгоритми, засновані на нейронних мережах. Серед популярних алгоритмів можна виділити наступні [7]:

- Регіональна сегментація (Region-Based Segmentation) – ділить зображення на різні фрагменти базуючись на порогових значеннях. Цей алгоритм є дуже простим і швидким, але вимагає високої контрастності об'єктів.
- Сегментація через виявлення країв (Edge Detection Segmentation) – використовує локальні особливості для визначення країв об'єкту. Високу якість результатів показує тільки на контрастних зображеннях.
- Сегментація на основі кластеризації (Segmentation based on Clustering) – розділяє пікселі зображення на гомогенні кластери. Як правило алгоритми кластеризації в аналізі зображень дуже часозатратні.
- Mask R-CNN – алгоритм на основі нейронної мережі. Для кожного об'єкту визначає межі, маску та клас, до якого він відноситься. Як

правило має велику кількість параметрів і вимагає багато часу для навчання.

1.2 Використання нейронних мереж для класифікації зображень

1.2.1 Нейронна мережа

На створення штучних нейронних мереж дослідників надихнула нервова система. Основною одиницею штучної нейронної мережі є штучний нейрон, що є моделлю нейрону в мозку живої істоти. За аналогією, штучний нейрон здатний приймати сигнали від нейронів, що підключені до нього, обробляти їх та передавати сигнал в наступні елементи, до яких він підключений. Сигналами, що подаються на вхід нейрону, виступають дійні числа, а сигналом, що подає сам нейрон виступає певна нелінійна функція, яка застосовується до зваженої суми сигналів.

Нейрон в штучній нейронній мережі має структуру, зображену на рис. 1.4. На вхід подаються сигнали x_i, далі обчислюється скалярний добуток вектору вхідних сигналів $x = \langle x_1, \ldots, x_m \rangle$ на вектор ваг $\omega = \langle \omega_1, \ldots, \omega_m \rangle$ та додається зміщення b, після чого до отриманого числа застосовується нелінійна функція активації φ.

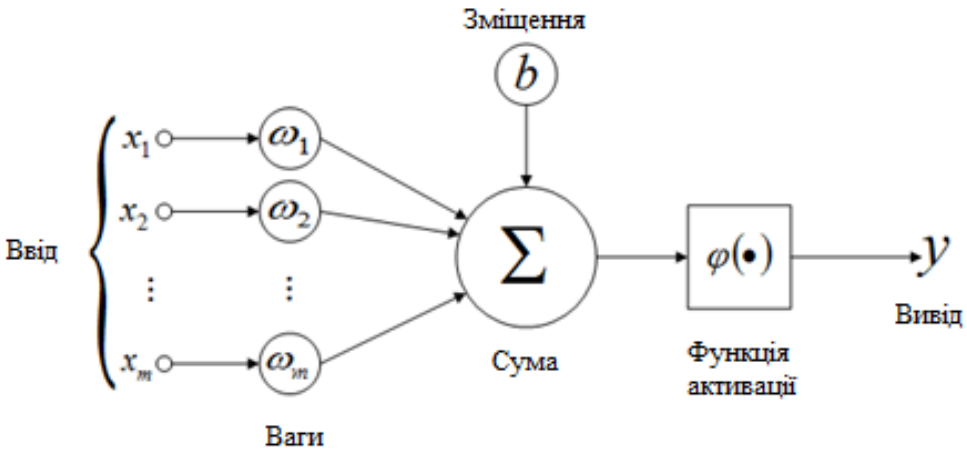


Рис. 1.4 – Математична модель нейрона

Таким чином, вихідний сигнал нейрона, якому на вхід подається m вхідних сигналів, обчислюється за наступною формулою:

$$y = \varphi\left(b + \sum_{i=1}^{m} x_i \cdot \omega_i\right) \quad (1)$$

В якості функції активації може виступати будь-яка нелінійна функція, однак зазвичай використовуються такі функції як сигмоїда, гіперболічний тангенс, ReLU, Leaky ReLU, та інші. Причому на внутрішніх шарах як правило використовується функція ReLU, а такі функції як сигмоїда, гіперболічний тангенс та softmax використовуються на зовнішньому шарі в задачах класифікації. Це пов'язано з тим, що функція ReLU обчислюється значно швидше і, якщо значення наближаються до 0, це не зменшує швидкість навчання.

Декілька таких нейронів утворюють шар, а декілька шарів утворюють нейронну мережу, як показано на рис. 1.5.

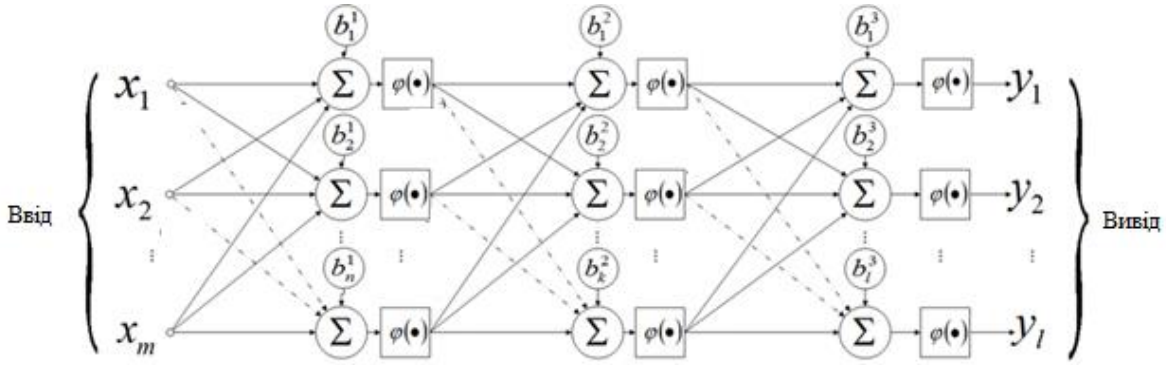


Рис. 1.5 – Декілька шарів нейронної мережі

1.2.2 Функція активації для багатокласової класифікації

Для багатокласової класифікації неперетинних класів на останньому шарі нейронної мережі використовується нормалізована експоненційна функція в якості функції активації, або, як її ще називають, функція softmax.

Нехай K – кількість класів, а $z = \langle z_1, \ldots, z_K \rangle$ – вектор, де z_i $(i = 1, \ldots, K)$ – сума добутку вхідних сигналів на ваги та зміщення $z_i = b + \sum_{j=1}^{m} x_j \cdot \omega_{ij}$, тоді функція softmax σ: $\mathbb{R}^K \to [0,1]^K$ визначена формулою

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_j^K e^{z_j}} \quad (2)$$

Далі, залежно від того яке саме представлення результату релевантне для даної задачі, можна в якості результату використовувати як вектор ймовірностей належності до певного класу або можна до отриманого вектору застосувати функцію argmax, яка повертає індекс елементу з найбільшою ймовірністю.

В якості функції помилки для оптимізації мережі з таким вихідним шаром може використовуватись перехресна ентропія.

1.2.3 Згорткові нейронні мережі

Коли мова йде про аналіз зображень, то в традиційних глибоких нейронних мережах 2D-зображення (представлене матрицею або тензором) аналізують як одновимірний масив, отриманий за рахунок згладження вхідного тензору. Даний підхід має проблему – вхідний вектор буде дуже великим (наприклад, зображення 1 000x1 000 дасть 3 000 000 вхідних ознак), що породжує надвелику кількість параметрів, які треба навчити.

Компромісним рішенням в даному випадку є використання операції згортки, яка містить значно менше параметрів, що треба навчати, і в той же час дає змогу виокремити з зображення важливі ознаки.

Згортковими нейронними мережами називаються такі штучні нейронні мережі, де обов’язково присутні згорткові шари та опціонально присутні агрегаційні шари.

В згортковому шарі під час прямого ходу кожен фільтр виконує операцію згортки (фільтр проходить в ширину та висоту з заданим кроком, обчислює добуток Фробеніуса, та записує результат в відповідну комірку двовимірної карти активації як показано на рис. 1.6), а двовимірні карти активації складаються в тензор. В результаті навчання мережа засвоює такі параметри фільтрів, які дозволяють їй при прямому ході виявляти певний тип

об’єкта на вхідному тензорі.

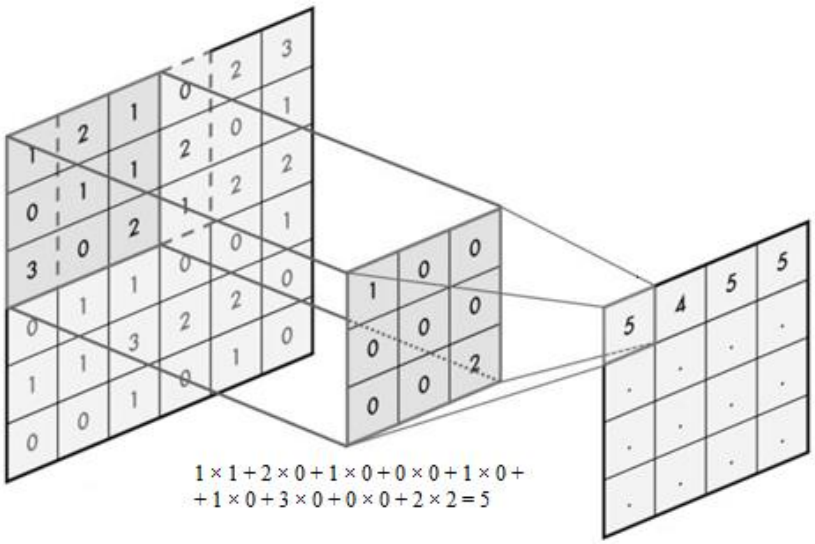


Рис. 1.6 – Демонстрація операції згортки на прикладі вхідної матриці 6x6 та фільтру згортки 3x3 з кроком 1

Також в згорткових мережах є агрегаційні шари (pooling layers), задачею яких є нелінійне зменшення розмірності вхідного тензору. Можуть використовуватись операції пошуку максимального елементу (приклад на рис. 1.7) або пошуку середнього значення елементів, але пошук максимального елементу є більш поширеним. Особливо актуальним є застосування агрегаційних шарів, якщо вхідні зображення дуже великого розміру. Також агрегаційні шари виконують функцію регуляризації – запобігання перенавчання моделі.

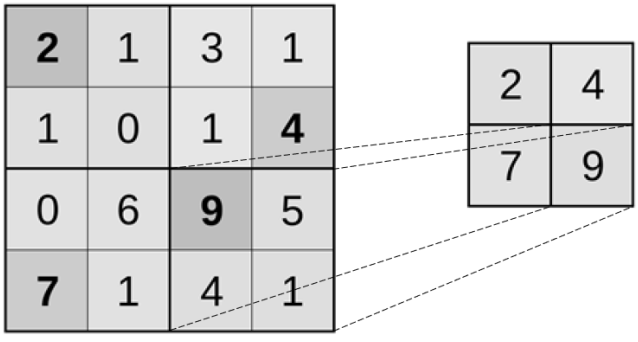


Рис. 1.7 – Демонстрація агрегаційного шару: за допомогою операції пошуку максимального елементу відбувається нелінійне зменшення розмірності вхідного тензору

Також в згорткових мережах є агрегаційні шари (pooling layers), задачею яких є нелінійне зменшення розмірності вхідного тензору. Можуть використовуватись операції пошуку максимального елементу (приклад на рис. 1.7) або пошуку середнього значення елементів, але пошук максимального елементу є більш поширеним. Особливо актуальним є застосування агрегаційних шарів, якщо вхідні зображення дуже великого розміру. Також агрегаційні шари виконують функцію регуляризації – запобігання перенавчання моделі.

1.2.4 Методи регуляризації

Глибокі нейронні мережі здатні віднаходити складні нелінійні залежності між предикторами та цільовою міткою. З одного боку це дає змогу будувати складні моделі, які здатні вирішувати доволі широкий спектр задач. А з іншого боку це призводить до проблеми перенавчання моделі (рис. 1.8).

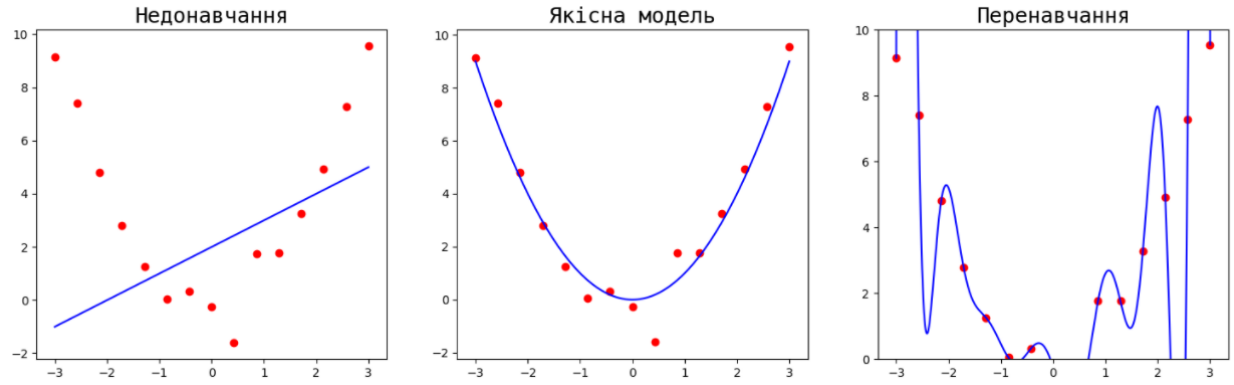


Рис. 1.8 – Недонавчання, якісна модель та перенавчання на прикладі вирішення задачі регресії з однією змінною

Для уникнення перенавчання моделі використовуються різні методи регуляризації моделі:

- L1-регуляризація
- L2-регуляризація
- Виключення (Dropout)
- Нормалізація
- Об’єднуючі шари

Одним із симптомів перенавчання нейронної мережі є дуже великі коефіцієнти в повноз’єднаних та згорткових шарах. Для того щоб уникнути такої ситуації використовується L1 та L2 регуляризації. При обчисленні функції помилки, яка мінімізується під час навчання, до неї додається сума модулів коефіцієнтів (3) або сума квадратів коефіцієнтів (4), помножені на коефіцієнт регуляризації λ, який є гіперпараметром.

L1-регуляризація:

$$\lambda \sum_i |a_i| \tag{3}$$

L2-регуляризація:

$$\lambda \sum_i a_i^2 \tag{4}$$

Також на повноз’єднаних шарах можна використовувати виключення випадкових нейронів (dropout). Цей метод запобігає коадаптації нейронів на тренувальних даних та пришвидшує навчання. Суть цього методу полягає в тому, що під час навчання деякі з нейронів, обрані випадковим чином, "вимикаються" (рис. 1.9). Ймовірність виключення кожного нейрону на ітерації визначає гіперпараметр коефіцієнт виключення.

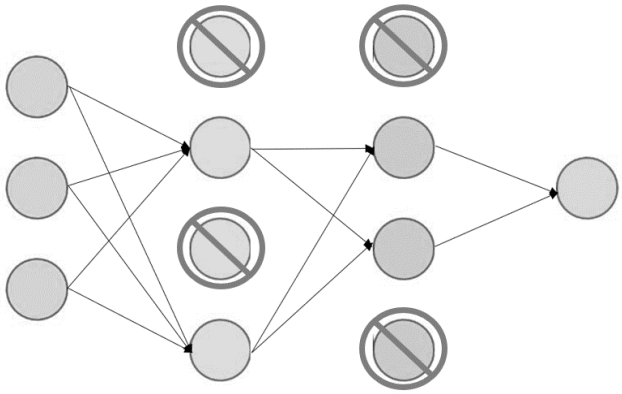


Рис. 1.9 – Приклад роботи регуляризації Dropout

Нормалізація вхідних параметрів та об’єднуючі шари не є методами регуляризації, однак їх використання невеликою мірою допомагає уникати перенавчання. Також варто зазначити, що використання нормалізації вхідних параметрів пришвидшує навчання та допомагає уникнути "затухання", якщо в

якості функції активації використовується сигмоїда або тангенсоїда.

1.3 Застосування розпізнавання образів в медицині

Застосування штучного інтелекту в медицині обіцяє бути такою ж революційною зміною як власне колись був розвиток медичної візуалізації. Однак, поки що широке впровадження машинного навчання в медичній сфері ще довго не відбудеться, через ряд причин, серед яких:

- Недостатня кількість даних
- Правове регулювання обробки персональних даних
- Складнощі інтерпретації результатів
- Велика варіативність медичних кейсів

По-перше, великою проблемою в даній сфері є складність збору та підготовки великої кількості даних, яку вимагає глибоке навчання для високої точності. Це обумовлено тим, що класи зображень та сегментаційні маски мають бути підтверджені або додатковим дослідженням (результати якого однозначно верифікують результат), або командою експертів, які незалежно один від одного розмічають зображення і воно включається в набір тільки за умови якщо оцінки експертів співпадають, що необхідно для мінімізації фактору людської помилки.

По-друге, нормативно-правові акти обмежують вільне використання медичної інформації в дослідницьких цілях. Наприклад, відповідно до закону України [8] збір та обробка персональних даних про здоров’я, політичні та релігійні погляди й інші допускається лише за умови інформованої згоди володільця (та інших підстав, які не застосовні в даному випадку). Тому процедура збору даних в інших сферах значно простіша, як мінімум з точки зору закону.

По-третє, визначити фактори та причини, через які алгоритм машинного навчання (зокрема нейронна мережа) дав той чи інший результат, неможливо. Математична модель зрозуміла і очевидна, але внутрішні параметри не

можливо інтерпретувати людині, яка її використовує.

По-четверте, кожен випадок занадто індивідуальний (адже людське тіло занадто різне, як і патологічні процеси) і навіть на великій кількості прикладів дуже складно побудувати якісну узагальнюючу модель.

Однак, на дослідження в цій сфері виділяється велика кількість коштів і в деяких задачах штучний інтелект починає показувати релевантні результати.

1.3.1 Діабетична ретинопатія

Діабетична ретинопатія – часте ускладнення цукрового діабету, яке проявляється ураженні сітківки, що згодом призводить до втрати зору. Нажаль, цей процес не зворотній і сучасні методи лікування можуть тільки сповільнити процес. Тому в превентивній медицині при ведені пацієнтів з цукровим діабетом дуже важливо регулярно проводити скринінги, аби помітити захворювання на ранній стадії. Такі скринінги вимагають ексклюзивний час лікаря-офтальмолога, фінансові затрати на процедуру і, що важливіше, мають ризик помилково негативного результату.

В свою чергу алгоритми машинного навчання досягли надвисокої точності в цій задачі (в деяких роботах точність сягала 99.18%, що перевищує точність обстеження лікарем) і витрати на автоматизовану діагностику були значно менші, ніж при мануальному тестуванні. [9]

1.3.2 Вірусна та бактеріальна пневмонія

Також однією з успішно вирішених проблем в 2020 році став аналіз рентгенівських знімків легень для диференціювання здорових легень, легень уражених бактеріальною пневмонією, легень уражених вірусною пневмонією та легень уражених вірусом SARS-CoV-2 (що є частинним випадком вірусної пневмонії, але завдяки відносно великій кількості прикладів алгоритмам вдалось визначити особливі патологічні патерни, специфічні саме для цього вірусу).

Дослідники з бразильського університету зуміли досягти 100% точності в визначенні ураження легень коронавірусною інфекцією. Для цього вони використали попередньо навчену модель ImageNet і провели подвійне донавчання: використали набір даних NIH ChestX-ray14 як проміжний етап і потім отриману модель донавчили на наборі даних COVID-19 Database.

Метою даного дослідження, як зазначалось в статті [10], було створення швидкої, дешевої і точної альтернативи ПЛР-тестуванню. Не зважаючи на високу точність моделі, використовувати рентгенівські знімки в якості масового тестування не дуже хороша ідея, адже з точки зору співвідношення інформації, яку можна отримати, та шкоди від діагностичної процедури, даний метод матиме високий показник частки неправдиво негативних результатів (через велику кількість випадків безсимптомного протікання хвороби). Однак, з точки зору аналізу даних такі результати є вражаючими.

Висновки до розділу 1

В даному розділі було розглянуто основні задачі комп'ютерного зору й алгоритми та технології, які використовуються для вирішення даних задач. Також було наведено приклади застосування комп'ютерного зору для медичної діагностики, де штучний інтелект показав високі результати.

Серед основних алгоритмів, які використовуються для вирішення задачі класифікації, що розглядається в даній роботі, обрано згорткові нейронні мережі глибокого навчання, адже вони себе добре показали в сфері аналізу зображень. Серед найбільших переваг можна виділити те, що в порівнянні з традиційними нейронними мережами CNN мають значно меншу кількість параметрів, але при цьому здатні навчати ядра для того, щоб виокремлювати із зображень закономірності та виявляти образи.

На даний момент йде велика кількість інвестицій в сферу глибокого навчання в медицині, однією з найбільших складових якої є аналіз медичних зображень.

РОЗДІЛ 2. ОГЛЯД ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ СИСТЕМИ

Система для розпізнавання пігментних уражень буде створена в форматі клієнт-серверного вебзастосунку. На серверній частині для класифікації зображень використовуватиметься нейронна мережа, що прийматиме на вхід зображення і видаватиме результат – вектор ймовірностей того, що зображення належить до одного з класів.

Відповідно, задачею даного розділу буде розглянути різні мови програмування, фреймворки, додаткові утиліти та сформувати стек технологій, що найкраще підходить для даної розробки.

2.1 Вибір фреймворку та мови для побудови нейронної мережі

Нейронні мережі глибокого навчання – складна математична модель, яка є композицією базових компонентів. Кожен раз розробляти з самого початку шари нейронної мережі, оптимізатори, оцінювачі помилок алгоритму та інтерфейс для користувача, що дозволить отримати візуалізації, абсолютно не раціонально, том у в сфері науки про дані використовують готові фреймворки для побудови нейронних мереж. Використання готових фреймворків замість розробки мережі з нуля є оптимальним рішенням, тому що:

- Переважна більшість фреймворків для нейронних мереж (та й в принципі будь-яких алгоритмів машинного навчання) написані на низькорівневих мовах, при розробці були обрані оптимальні алгоритми обчислень з точки зору використання ресурсів операційної системи, що дає змогу отримати високу швидкість навчання нейронних мереж, що є дуже важливим, адже навіть не складна модель може мати декілька мільйонів параметрів і час навчання такої моделі може займати від декількох годин до декількох днів і навіть місяців.
- Відсутність необхідності витрачати велику кількість часу на

розробку, дає змогу більше сконцентруватись на аналізі даних, дослідженні різних архітектур нейронних мереж, аналізі та обґрунтуванні різних підходів, надає можливість спробувати велику кількість різних рішень та обрати серед них ті, що дають найкращий (з точки зору кожної конкретної задачі) результат.

- Більшість фреймворків надає широкий набір інструментів аналітику для оцінки якості результатів, діагностики можливих помилок за допомогою статистичних метрик та візуалізацій.
- Дає змогу уникнути типових багів при створенні нейронних мереж з нуля, адже в нейронних мережах використовується велика кількість матричних обчислень.

2.1.1 Вибір мови програмування

Область машинного навчання має специфічні потреби, тому не кожна мова програмування може задовольнити їх в повній мірі. Серед таких потреб можна виділити наступні:

- Бібліотеки для наукових обчислень та візуалізацій
- Бібліотеки для роботи з даними (збір, очистка, препроцесинг)
- Легкість у використанні
- Можливість швидко робити складні математичні моделі

Через перелічені причини особливу популярність в області аналізу даних та машинного навчанні здобули такі мови як Python, R, Julia, Scala, Matlab.

Найпопулярнішою мовою програмування на даний момент є мова Python. Цю мову використовує 8,6 мільйонів розробників [11] по всьому світу, адже ця мова має дуже простий синтаксис та величезну спільноту розробників, що робить процес розробки на мові Python швидким та ефективним. Для даної мови написана велика кількість бібліотек, які застосовуються в машинному навчанні: numpy для швидких векторизованих обчислень, pandas для роботи з

табличними даними, scipy для наукових обчислень, sklearn для машинного навчання, seaborn та matplotlib для якісних візуалізацій та велика кількість бібліотек для глибокого навчання, що підтримуються в цій мові. Серед мінусів можна виділити те, що в цій мові динамічна типізація, тому є ймовірність великої кількості помилок.

Мова програмування R була створена спеціально для наукових та статистичних обчислень. Дана мова має багато бібліотек для візуалізацій (наприклад ggplot2), і велику кількість вбудованих функцій для статистики. Дана мова може бути кращим вибором, ніж Python, коли мова йде про специфічні наукові завдання, однак вона дуже повільна і не має спеціальних бібліотек, що можуть забезпечити швидкі обчислення.

Julia – доволі молода мова програмування, яка була створена спеціально для наукових обчислень. Вона відноситься до just-in-time компільованих мов, через що показує високу швидкість. Так само як і в Python, її синтаксис дуже простий для розуміння. Однак через свою «незрілість», вона має дуже малу спільноту та не дуже розвинену екосистему бібліотек для статистичних обчислень та машинного навчання.

Scala мультипарадигменна мова, що підтримує два підходи: об’єктно-орієнтовне програмування та функціональне програмування. В поєднанні з рушієм Spark дає змогу організувати високопродуктивні кластерні обчислення. Серед мінусів можна виділити те, що синтаксис і система типів дуже складні для вивчення початківцями.

Мова програмування Matlab також була створена спеціально для наукових обчислень, має велику спільноту розробників та надає можливість для створення якісних візуалізацій для обчислень. Серед головних мінусів можна виділити те, що вона є пропрієтарною.

Враховуючи перелічені плюси й мінуси різних мов програмування, найкращим вибором для даної роботи буде мова програмування Python, через зручність використання, простий синтаксис, велике спільноту розробників та

велику кількість фреймворків для глибокого навчання, статистичних візуалізацій та математичних обчислень.

Побудова нейронної мережі та аналіз даних відбуватимуться в спеціальному додатку – Jupyter Notebook. Його можна використовувати в різних мовах програмування (Python, R, C++, Julia та інші). Цей застосунок дає змогу створювати інтерактивні ноутбуки (рис. 2.1) – сторінки, які складаються з комірок або з програмним кодом, або текстовою інформацією (використовуючи мову розмітки markdown або html). Підтримується предметно-орієнтовану мову Latex в текстових клітинках для формул та можливість виводити візуалізації та результати під комірками, в яких виконувався код.

Рис. 2.1 – Скріншот додатку для створення інтерактивних ноутбуків Jupyter Notebook

Відносно новим конкурентом Jupyter Notebook в якості IDE можна назвати DataSpell. Це зручне середовище інтегрованої розробки, дизайн якого цілком відповідає потребам аналізу даних. Серед переваг можна відзначити сучасний дизайн, типовий для продуктів JetBrains, зручні інструменти налаштування віртуальних середовищ для виконання програмного коду та допоміжні інструменти написання коду, як-то автодоповнення. Головними недоліками є те, що ця IDE поки доступна тільки в тестовому режимі і

потенційно відпуститиметься під комерційною ліцензією.

2.1.2 Вибір фреймворку для розробки нейронної мережі

Серед найвідоміших фреймворків виділяють наступні: TensorFlow, Keras, PyTorch, Apache MXNet, та інші. Далі буде детально розглянуто перелічені фреймворки та обрано оптимальний.

а) TensorFlow та Keras

TensorFlow – відкрита бібліотека для машинного навчання, створена командою Google Brain Team на основі датафлоу парадигми та диференціального програмування. Доступна під ліцензією Apache License 2.0.

Переваги TensorFlow:

- Дана бібліотека є легкою в використанні з детальною документацією та великою кількістю туторіалів
- Часто виходять оновлення
- Велика спільнота розробників
- Має інструмент TensorBoard, що значно полегшує процес тренування та усунення багів
- Підходить для роботи з різними типами даних: табличні, текстові, зображення, відео та інші

Недоліки TensorFlow:

- В порівнянні з деякими іншими фреймворками може працювати повільніше

Keras – бібліотека з відкритим програмним кодом для нейронних мереж, яка може бути надбудовою до Tensorflow, Theano та деяких інших бібліотек. Як пояснюють розробники, вона розроблялась радше як інтерфейс до інших бібліотек, ніж самостійна бібліотека. З 2017 року офіційно підтримується TensorFlow. [12]

б) PyTorch

Ще одна популярна бібліотека для нейронних мереж, розроблена

компанією FaceBook. Підтримує мови Python та C++. Так як і TensorFlow підходить для створення обчислювальних графів.

Переваги PyTorch:

- Дружній до користувача дизайн
- Має інструменти для усунення багів, наприклад PyCharm debugger
- Має навчені моделі для донавчання

Недоліки PyTorch:

- На відміну від TensorFlow не має інструментів для візуалізації та моніторингу
- Через «незрілість» бібліотеки, спільнота не дуже велика

в) Apache MXNet

Переваги Apache MXNet:

- Швидкий та ефективний
- Підтримує багато мов програмування: Scala, R, Python, C++ та JavaScript

Недоліки Apache MXNet:

- Усунення багів та покращення бібліотеки можуть займати багато часів
- Порівняно з іншими бібліотеками має невелику спільноту розробників

Для даної роботи краще зупинитись на бібліотеці Keras, що працює як доповнення та інтерфейс для бібліотеки TensorFlow. По-перше, тому що ця бібліотека має дуже детальну та якісну документацію. По-друге через те, що ця бібліотека має широкий інструментарій для візуалізацій проміжних результатів навчання. По-третє, в даної бібліотеки велике співтовариство розробників, що дуже важливо при виникненні проблем в процесі розробки. По-четверте, в даної бібліотеки зручний та зрозумілий дизайн, що спрощує використання в розробці.

2.2 Вибір технології для серверної частини додатку

Для зручності користування системою багатьма користувачами, вона буде розроблена у форматі клієнт-серверного вебзастосунку. Для цього треба обрати стек технологій: мову програмування логіки на бекенді, інструменти створення зручного дизайну вебзастосунку на стороні фронтенду та вебфреймворк, за допомогою якого можна створити вебзастосунок швидко і якісно.

2.2.1 Вибір мови програмування

Найпоширенішими мовами для бекенд розробки наразі є такі мови як Java, Python, Ruby, PHP. Починає набирати популярність створена компанією Google у 2007 році мова Go.

Наведемо короткий огляд основних мов програмування в аспекті бекенд розробки, оцінивши їх переваги та недоліки:

a) Java

Дуже популярна мова програмування в області веброзробки для бекенд частини системи. Визначна тим, що центральною парадигмою цієї мови є ООП. Окрім веброзробки широко застосовується в мобільній розробці, банківській сфері та бекенд розробці. Відноситься до типобезпечних мов. Серед недоліків можна виділити те, що запуск застосунків потребує великої кількості ресурсів. Для швидкої та ефективної розробки використовуються такі фреймворки як Spring або Micronaut.

b) Python

Мова, популярність якої зростає останнім часом. Має велику спільноту розробників та екосистему бібліотек. Успішно застосовується в бекенд бо через простий синтаксис розробка займає доволі мало часу. Недоліком мови є динамічна типізація, що збільшує кількість потенційних багів під час розробки. Однак, цю проблему можна вирішити за допомогою бібліотек для типізації, наприклад Typing. Ще однією проблемою (чи перевагою) є GIL –

глобальне блокування інтерпретатора, суть якого в тому що в кожен момент часу лише один потік має доступ до інтерпретатора Python. Серед основних фреймворків, що використовуються для веброзробки можна виділити Flask, Django, Tornado, web2py та інші.

c) Ruby

Це доволі повільна мова, тому найменша помилка в розробці може коштувати багато часу. Ця мова занадто перевантажена синтаксичним цукром. Також можна зазначити, що там доволі нетривіальне ООП. Для побудови вебзастосунків використовується переважно Ruby on Rails.

d) C#

Мова програмування створена корпорацією Microsoft у 2000 році і зараз активно підтримується і розвивається. Дана мова використовує об'єктно-орієнтовну парадигму. Також варто відмітити велику кількість синтаксичного цукру, що в умілих руках може дуже допомогти в розробці. В цієї мови дуже велика спільнота і велика кількість матеріалів для вивчення мови. Ще з переваг можна відзначити строгу типізацію, що дає можливість уникати великої кількості помилок під час розробки. Серед недоліків можна виділити те, що мова дуже легко дизасемблюється. Попри те, що розробники активно працюють над забезпеченням кросплатформеності мови, вона тісно асоціюється з операційною системою Windows. C# - мова дуже потужна і на опанування всіх тонкощів може піти дуже багато часу.

e) JavaScript (Node.js)

Насправді Node.js – не фреймворк, а платформа, побудована на рушієві ChromeV8, що перетворює JavaScript з вузько призначеної мови в мову широкого застосування. Серед переваг можна виділити асинхронність, що керується подіями, що дає змогу розробляти ефективні застосунки вводу/виводу, легкість для опанування, величезна спільнота розробників, велику кількість бібліотек і сумісність з кодом клієнтської частини додатку. Серед недоліків можна виділити відсутність можливості масштабування, адже

платформа не дає можливості використовувати переваги багатопроцесорності, складнощі в роботі з реляційними базами даних та низьку швидкодію в CPU-інтенсивних задачах.

f) Golang

Мова програмування розроблена корпорацією Google в 2007 році. Останнім часом починає підніматися в рейтингу мов програмування. Серед переваг можна виділити високу швидкість (проєкти на Go швидко компілюються), сумісність з мовою C, багату бібліотеку (розширена бібліотека покриває широкий спектр специфічних вимог, а вебфреймворк є частиною стандартної бібліотеки) та зручне документування (розробники можуть генерувати документацію із коментарів в сирцевому коді). До недоліків можна віднести те, що в неї дуже мала екосистема і вона доволі складна для опанування новачками. Також варто зазначити, що ця мова досі не знайшла своєї основної області застосування, тому важко виділити якусь одну сильну сторону.

Оптимальним вибором для даної роботи буде мова програмування Python. По-перше, через те, що на ній зручно швидко розробляти додатки. По-друге, оскільки модель машинного навчання буде реалізована на мові Python, буде простіше організувати взаємодію бізнес-логіки та нейронної мережі

2.2.2 Вибір фреймворку для створення вебзастосунок

Оскільки в якості мови програмування обрано Python, в даному підрозділі розглядатимуться фреймворки для цієї мови.

a) Django

Найпопулярніший веб фреймворк на Python. Це дуже потужна бібліотека, що дає можливість робити розробку швидкою, а освоєння фреймворку дуже простим. Django надає інструменти для роботи з базами даних (за допомогою ORM – об'єктно-реляційного мапера), забезпечення авторизації та аутентифікації, маршрутизації ендпоінтів та міграції баз даних.

b) Flask

Мікро фреймворк, який дає змогу працювати з HTTP-запитами зі значно меншою кількістю абстракцій. Цей фреймворк надає доволі мало інструментів, але є можливість додавати туди все необхідне.

c) Tornado

Асинхронна бібліотека, яка при правильній конфігурації здатна обробляти більше ніж 10 000 одночасних підключень. Була створена для проєкту FeedFriend, пізніше її викупила компанія FaceBook, після чого його код став відкритим.

d) Aiohttp

Мікро фреймворк для створення асинхронних HTTP-клієнта та HTTP-сервера на мові програмування Python. Бібліотека є надбудовою над стандартною бібліотекою Asyncio, яка надає інструменти для ефективної розробки вебзастосунків.

Даний застосунок не потребує такого широкого спектру інструментів, які готовий надати Django і навіть Tornado буде занадто важким та надмірним для задачі, яка вирішується в даній роботі. До того ж в даній системі багато CPU-інтенсивних обчислень, тому асинхронність не дасть сильної переваги в швидкості обробки запитів. Тому оптимальним вибором буде Flask.

2.2.3 Вибір технологій для дизайну вебзастосунку

Як правило, для вебсторінок використовується мова розмітки HTML, для стилів CSS, а для інтерактивної поведінки мова програмування JavaScript. Для швидкого створення сторінок можна використовувати бібліотеку Bootstrap, яка надає готові шаблони для стилістичного оформлення вебсторінок та створення інтерактивного дизайну. В разі виникнення потреб, які неможливо задовольнити готовими шаблонами, які надає дана бібліотека, можна використати безпосередньо CSS та JavaScript.

2.3 Вибір технології для розгортання серверу

Як відомо, у кожного додатку є перелік бібліотек, які необхідні для того, щоб він міг виконувати свої функції. В даному випадку є як мінімум бібліотека для нейронної мережі, бібліотека для візуалізації, вебфреймворк та декілька бібліотек для вирішення дрібних задач.

Для коректної роботи додатку на іншому пристрої треба переконатись, що встановлений Python, після цього встановити всі необхідні бібліотеки, надати інструкцію як налаштувати середовище та розгорнути застосунок. Для вирішення цієї проблеми разом з додатком треба надавати дуже детальну інструкцію з підготовки, встановлення та розгортання додатку.

Цю проблему можна вирішити за допомогою технології Docker. Він дає змогу створити образ, в якому буде прописана вся необхідна інформація для налаштування, команди для підготовки середовище та розгортання додатку. За допомогою Docker процес розгортання додатку на новому хості зводиться до невеликих змін в файлі *docker-compose.yml* та запуску двох команд. Всі необхідні залежності та налаштування будуть встановлені при створенні контейнеру з образу.

Тому замість традиційного підходу, де встановлюється інтерпретатор Python, потім встановлюються бібліотеки і вносяться відповідні налаштування, буде використано технологію, яка зробить це автоматично.

2.4 Набір даних HAM10000

Для навчання алгоритму розпізнавання пігментних уражень необхідні тренувальні дані. В якості джерела розмічених зображень було обрано набір даних HAM10000, який складається із 10015 зразків (рис. 2.2), які представляють 7 поширених діагнозів. [13]

Набір даних розповсюджується під ліцензією Creative Commons Attribution-NonCommercial 4.0 International Public License, яка дає змогу використовувати його для некомерційних розробок. Для цього потрібно

вказувати посилання на автора та зміни, які застосовувались до вихідних даних.

Більше ніж 50% випадків, представлених у наборі даних, було підтверджено патологічно, а інші були підтверджені мікроскопією in-vivo або експертним консенсусом лікарів.

Набір даних був зібраний із двох джерел: особистої клінічної практики Кліффа Розендаля з Австралії та кафедри дерматології австрійського медичного університету у Вені. Частина даних зберігалися у презентаціях PowerPoint, де кожна презентація відповідала місячному періоду. Інша частина зберігалась у вигляді діапозитивів і була спеціально оцифрована для створення набору даних. В фотографіях було нормалізовано кольорову гаму аби вони були близькі до сучасної якості фотографій. Випадки з невизначеними або недостовірними діагнозами були виключені із набору даних. Також було виключено зображення дуже поганої якості, на яких були проблеми з фокусуванням або артефакти, які заважали візуально визначити патологічний процес.

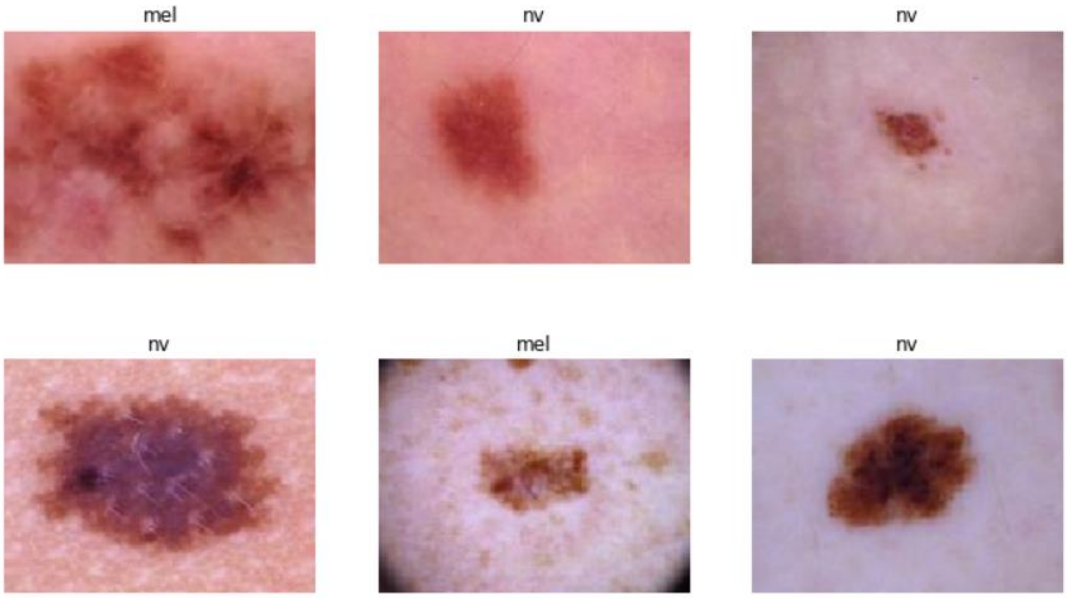


Рис. 2.2 – Зображення із набору даних HAM10000

В наборі даних представлено 7 класів зображень:

- Меланоцитний невус

- Меланома
- Актинічний кератоз та інтраепітельальна карцинома
- Базальноклітинна карцинома
- Дерматофіброма
- Себорейний кератоз
- Васкулярні утворення

До набору даних додається csv-файл з метаданими, в якому є інформація про вік пацієнтів, стать, місце локалізації ураження, метод діагностики, за допомогою якого було підтверджено діагноз, назва кожного зображення в архіві та унікальний ідентифікатор для кейсу (деякі ураження були представлені в декількох зображеннях і укладачі набору даних вирішили їх залишити, адже такі зображення фіксували ураження з різних ракурсів та з різним масштабом).

Висновки до розділу 2

В даному розділі було проаналізовано різні мови програмування та фреймворки окремо для побудови клієнт-серверного додатку та для побудови моделі класифікації на основі згорткової нейронної мережі.

Для обробки даних та побудови моделі класифікації було обрано мову програмування Python через те, що через простоту синтаксису вона дає можливість сконцентруватись на обробці та аналізі даних, експериментах з різними конфігураціями нейронних мереж для підбору оптимального варіанту. Фреймворком для нейронних мереж було обрано TensorFlow через його зручність, велике спільнота розробників та інструменти для візуалізації та моніторингу, що значно спрощує процес розробки.

Для створення клієнт-серверного додатку було обрано мову програмування Python, вебфреймворк Flask через його гнучкість та мінімальний набір функцій, що є найкращим варіантом для даної роботи, та бібліотеку шаблонів Bootstrap для швидкої розробки зручного та привабливого дизайну клієнтської частини.

Для зручності розгортання серверу на будь-якій платформі та швидке перенесення було обрано інструмент для контейнеризації Docker та інструмент оркестрації Docker-compose для легкого налаштування й запуску контейнерів.

Для навчання моделі використовуватиметься набір даних HAM10000, опублікований Гарвардським університетом у 2018 році.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Розробка алгоритму класифікації зображень

Як зазначалося в другому розділі, для навчання моделі класифікації використовуватиметься набір даних HAM10000. Перш ніж використовувати сам набір даних варто поглянути на метадані.

3.1.1 Аналіз набору даних

В метаданих набору даних, представлених у вигляді таблиці (рис. 3.1), окрім відповідної мітки класу (атрибут *dx*), міститься також додаткова інформація про кожне зображення, зокрема:

- унікальний ідентифікатор кожного кейсу (представлений атрибутом *lesion_id*), адже в деяких випадках одне й те саме ураження представлене різними знімками;
- метод клінічного підтвердження діагнозу (атрибут *dx_type*);
- вік пацієнта (атрибут *age*)
- стать пацієнта (атрибут *sex*)
- локалізація ураження (атрибут *localization*)

	lesion_id	image_id	dx	dx_type	age	sex	localization
0	HAM_0000118	ISIC_0027419	bkl	histo	80.0	male	scalp
1	HAM_0000118	ISIC_0025030	bkl	histo	80.0	male	scalp
2	HAM_0002730	ISIC_0026769	bkl	histo	80.0	male	scalp
3	HAM_0002730	ISIC_0025661	bkl	histo	80.0	male	scalp
4	HAM_0001466	ISIC_0031633	bkl	histo	75.0	male	ear

Рис. 3.1 – Скріншот таблиці з метаданими HAM10000

Найбільш представлений клас – *nv* (6705 зразків), трохи менш представлені фотографії меланоми *mel* (1113 зразків) та себорейного кератозу *bkl* (1099 зразків). Інші класи представлені мало (рис. 3.2), зокрема: актинічні кератози та інтраепітеліальна карцинома / хвороба Боуена *akiec* – 372 зразків,

дерматофіброма *df* – 115 зразків, базально-клітинна карцинома *bcc* – 514 зразків та васкулярні новоутворення *vasc* – 142 зразків.

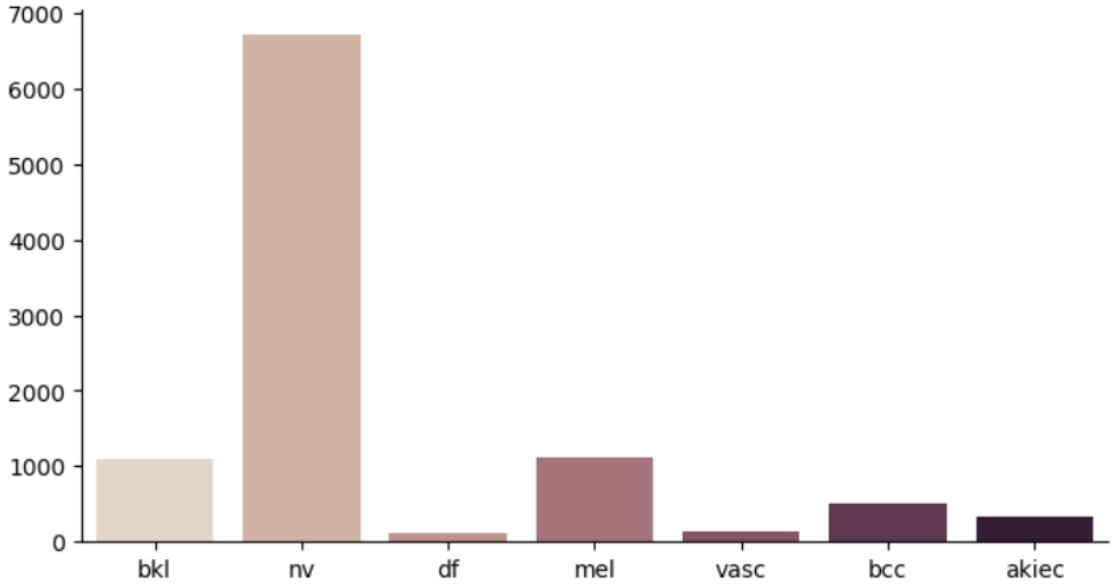


Рис. 3.2 – Кількість об'єктів кожного класу представлених в наборі даних

Такий сильний дисбаланс класів може дуже сильно погіршити якість моделі, тому раціональніше буде об’єднати мало представлені класи в збірний *undef*, який вказуватиме на те, шо точно визначити діагноз неможливо і потрібна додаткова експертна оцінка. Тому всі класи, які представлені менш ніж 1 тис. зображень будуть згруповані (рис. 3.3).

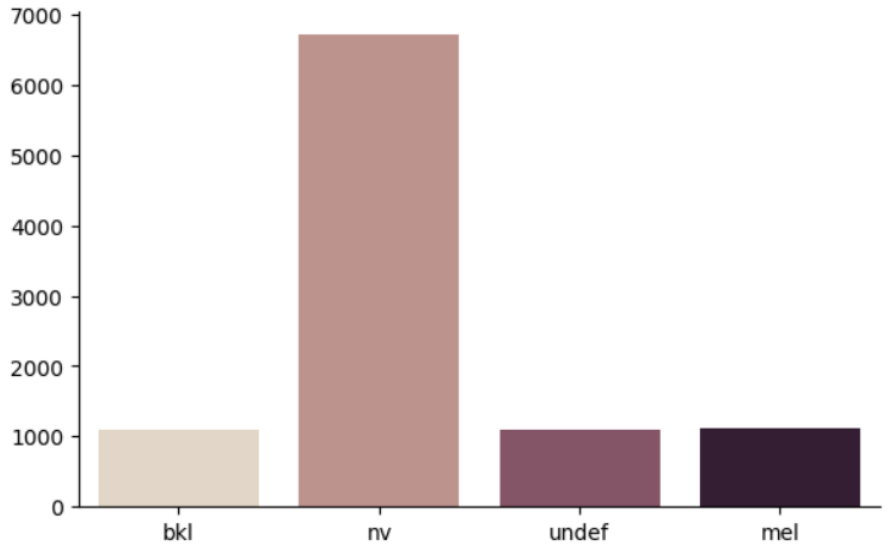


Рис. 3.3 – Кількість об’єктів в кожному класі після угрупування мало представлених класів в один збірний

Після групування мало представлених класів в один збірний отримано 4 класи, серед яких: *nv* – 6705 зразків, *mel* – 1113 зразків, *bkl* – 1099 зразків та *undef* – 1098 зразків. В такому вигляді набір даних досі є незбалансованим, але цій проблемі можна зарадити за допомогою методів аугментації та використанні зваженої функції оцінки помилки при навчанні моделі.

На рис. 3.4 показано розподіл віку пацієнтів залежно від діагнозу да допомогою коробкової діаграми. На графіку видно, що переважну більшість випадків звичайного меланоцитарного невуса можна відділити спираючись лише на вік. Загалом, згідно з клінічним статистичним даним, середній вік діагностування меланоми – 65 років[14], і при диференціальній діагностиці в клінічній практиці це враховується. Але в даному випадку врахування віку може негативно сказатись на повноті визначення меланоми, через те що даний клас мало представлений у даному наборі даних. Тому при розробці моделі цей атрибут враховувати не варто.

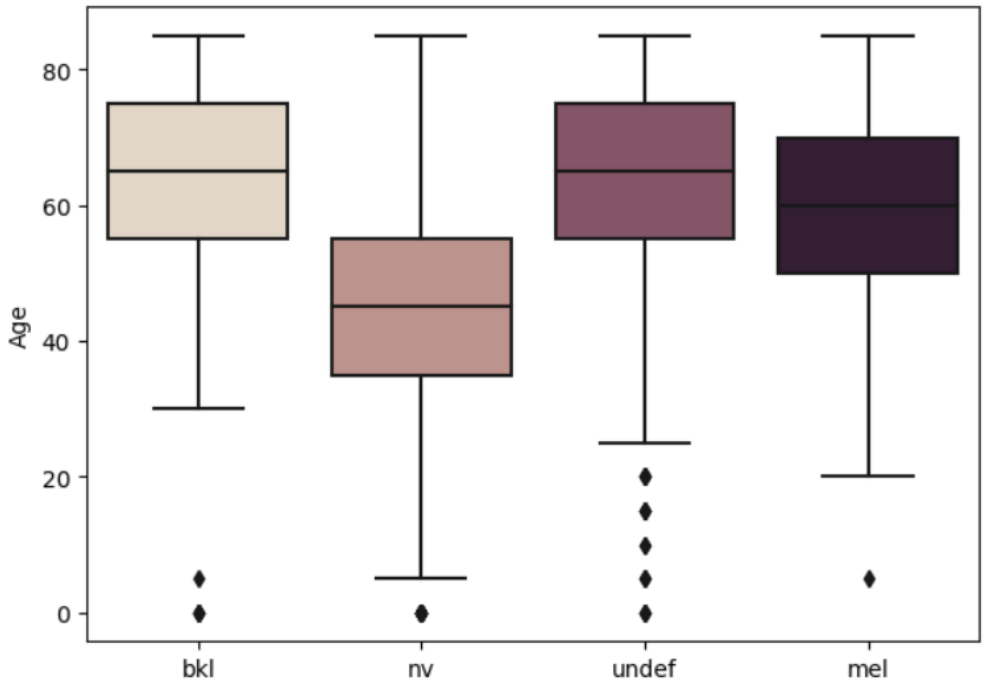


Рис. 3.4 – Коробкова діаграма розподілу віку за класами

Після аналізу розподілу захворювань за статтю пацієнтів (рис. 3.5) стає зрозуміло, що даний атрибут також не дає жодної нової інформації адже вибірка має невеликий гендерний дисбаланс, через що в кожному класі

спостерігається перевага в сторону чоловіків. Загалом такий результат навіть суперечить загальній статистиці захворюваності, адже, наприклад, в категорії до 50 років меланома більш поширена у жінок. [14]

Також варто зазначити, що різні фото одного й того ж ураження видалятись не будуть. Такі зображення хоч мають високу схожість, але приносять нову інформацію, що допомагає узагальнити модель.

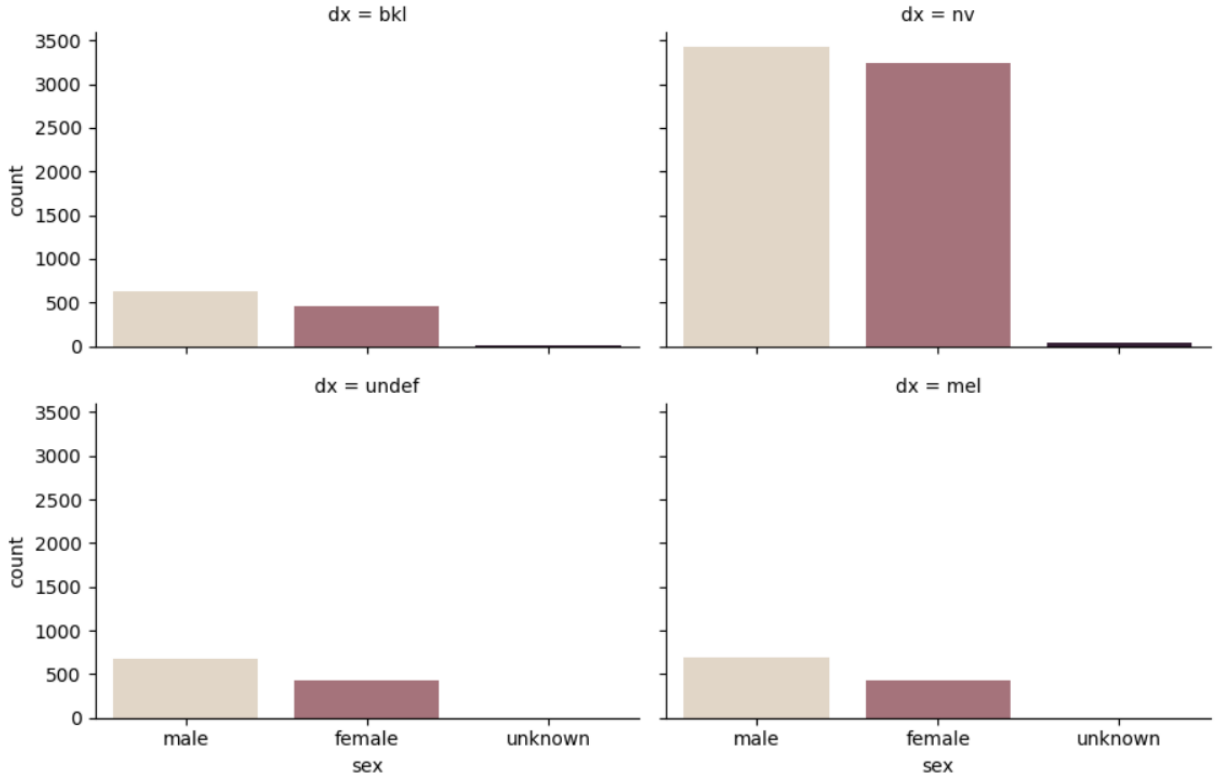


Рис. 3.5 – Порівняння кількості чоловіків та жінок залежно від діагнозу

Локалізація ураження теж не дає нової інформації (частину графіків наведено на рис. 3.6), співвідношення в класів в кожній групі локалізації майже однакове. Тому немає сенсу використовувати дану інформацію для навчання моделі.

В ході аналізу було визначено, що додаткові атрибути в метаданих не релевантні для поліпшення якості моделі, адже в більшості випадків вони не несуть ніякої нової інформації і є результатом незбалансованості набору даних відносно захворюваності відповідними хворобами. Більше того, розподіли деякої інформації суперечать статистичним даним з клінічної практики, тому

врахування таких даних може знизити точність моделі на нових даних.

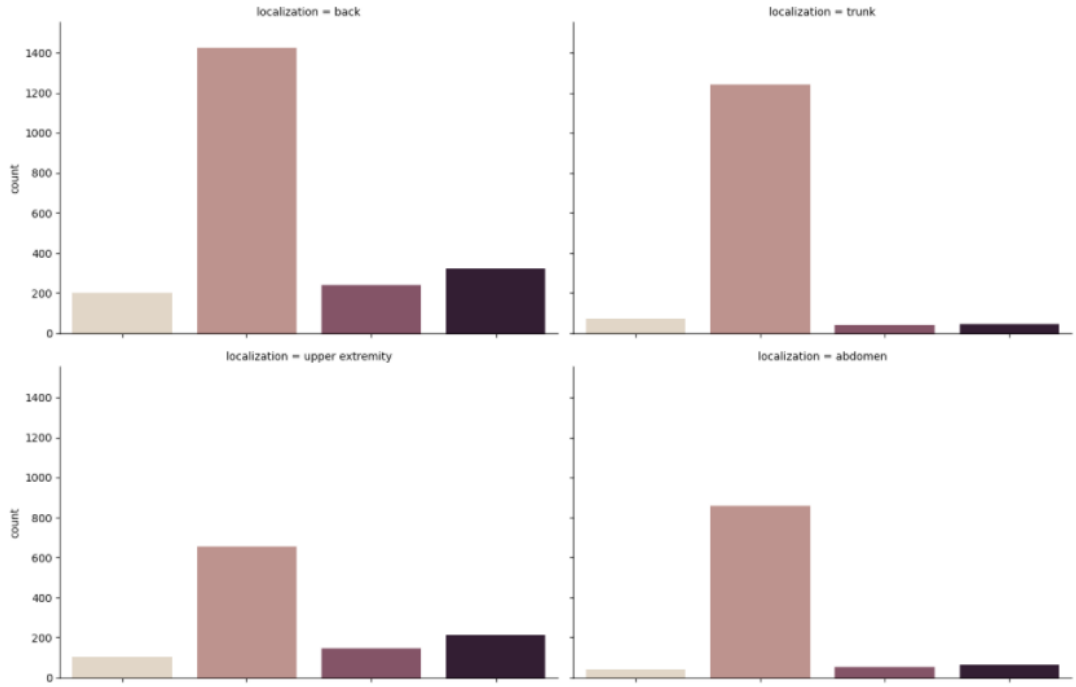


Рис. 3.6 – Порівняння кількості кейсів в різних групах локалізації ураження

3.1.2 Побудова нейронної мережі для класифікації зображень

Для класифікації зображень було вирішено використовувати згорткові нейронні мережі. Нейронна мережа складається з двох частин: виявлення ознак за допомогою згорткових шарів та аналіз ознак за допомогою повноз’єднаних шарів.

Розширення зображень в наборі даних складає 450x600. Для навчання нейронної мережі зображення було зменшено до розширення 120x160. Всі важливі патологічні акценти пігментних новоутворень було збережено, а менший розмір дозволить значно швидше навчати нейронну мережу.

Вибірку даних було розбито на дві підвибірки: навчальна та тестова, для того щоб була можливість оцінити якість моделі на нових даних, які не використовувались в навчанні. Тестові дані складають 7% від загальної вибірки, отже в навчальній вибірці 9313 зразків, а в тестовій – 702.

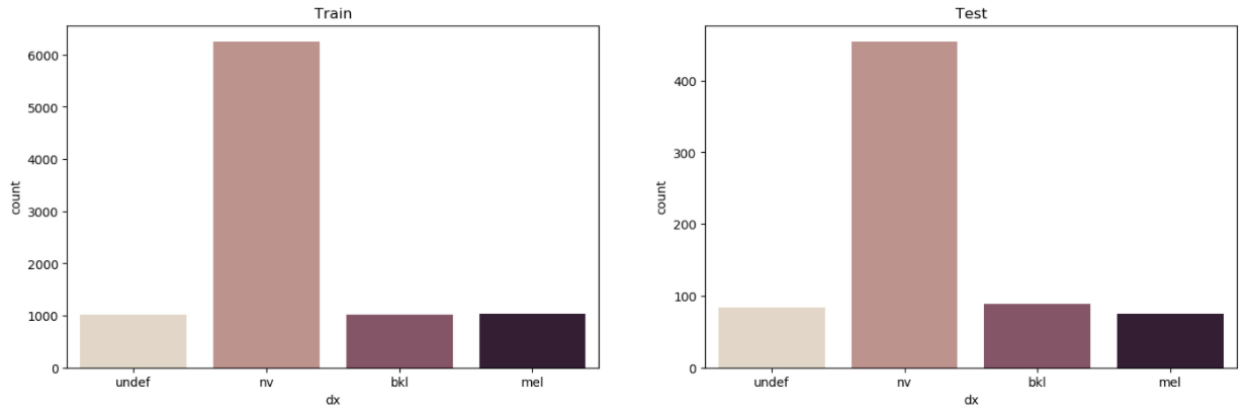


Рис. 3.7 – Представленість класів у тренувальній та навчальній вибірках

Оскільки під час розбиття даних на дві підвибірки використовувалось випадкове перемішування, розподіли в обох класах в результаті вийшли приблизно однаковими. Після розбиття даних в файлі метаданих, зображення з набору даних були переміщені в відповідні папки test та train, в яких було по 4 папки з відповідними назвами класів.

За допомогою функції *image_dataset_from_directory()* з пакету *keras.preprocessing* було створено об’єкт генератор, який подає данні батчами по 32 зображення. Такий підхід дає змогу уникнути переповнення оперативної пам’яті, бо 10 тис. зображень завантажуються порційно.

Як зазначалося в п. 3.1.1, набір даних дуже незбалансований і одні класи представлені більше ніж інші. Частково цю проблему було вирішено за допомогою об'єднання найменш представлених класів в один збірний – *undef*. Однак, наразі клас *nv* складає приблизно 66% вибірки, в той час як решта класів складає по 10-13%.

Для того щоб нівелювати цю проблему можна штучно збільшити різноманіття мало представлених класів та порахувати відповідні ваги класів, які будуть враховуватись при оцінці в функції помилки.

Для збільшення різноманіття в модель буде включено шар аугментації даних, який застосовує випадкову зміну або декілька змін до зображення (рис. 3.8). Цей шар активний тільки під час навчання моделі.

Для даної задачі зміна забарвлення, розтягування та будь-яка зміна

пропорцій можуть скомпрометувати зображення пігментного утворення. Наприклад для оцінки ризику злоякісної меланоми, згідно правила ABCDE, використовують такі характеристики як рівномірність кольору, симетричність та нерівні межі [15]. Тому важливо, аби при аугментації не змінювалась форма та забарвлення новоутворень.

```
In 17
data_augmentation = keras.Sequential(
    [
        layers.experimental.preprocessing.RandomRotation(0.15, fill_mode='nearest'),
        layers.experimental.preprocessing.RandomFlip("horizontal_and_vertical"),
        tf.keras.layers.experimental.preprocessing.RandomZoom(height_factor=(-0.1, 0.1),
                                                              width_factor=(-0.15, 0.15),
                                                              fill_mode='nearest')
    ], name="Augmentation"
)
```

Рис. 3.8 – Скріншот клітинки зі створенням шару для аугментації даних

Отже, для аугментації використовуватимуться віддзеркалювання зображення по вертикалі та по горизонталі, обертання зображення навколо центру на кут від 0 до 0.15·π радіан та випадкове наближення зображення (до 10% у висоту та до 15% у ширину). Утворені пробіли заповнюватимуться найближчими пікселями до межі пробілу.

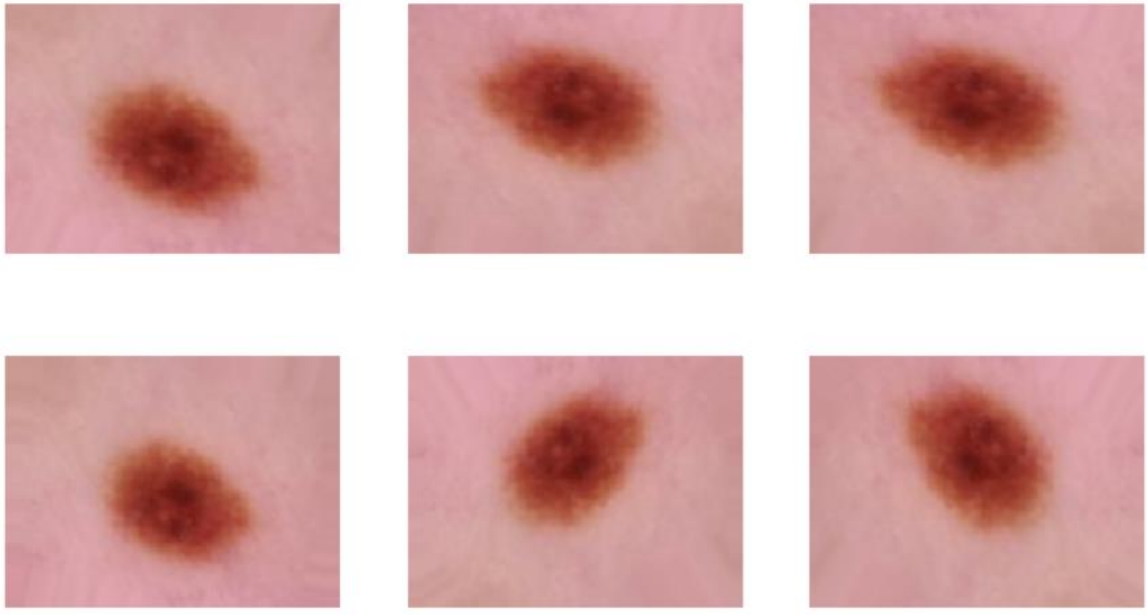


Рис. 3.8 – Приклад аугментації зображення з використанням шару *data_augmentation* в нейронній мережі

За допомогою функції *compute_class_weight()* з пакету *utils.class_weight* бібліотеки sklearn було отримано ваги класів, які балансуватимуть розрахунок помилки при навчанні – рахуватимуть помилку на мало представлених класах більшою.

Вхідні дані після аугментації масштабуються – з діапазону [0, 255] трансформуються в діапазон [0, 1], що є хорошою практикою для пришвидшення навчання та зменшення ймовірності потрапляння в локальний мінімум.

Далі один за одним йдуть 5 згорткових шарів, які складаються зі згортки, нормалізації даних та функції активації ReLU. На першому та останньому серед них розмір фільтра – 5, а в трьох посередині – 3. Крок, з яким зміщується фільтр при згортці – 2. Кількість фільтрів – 64, 128, 256, 512 та 1024 відповідно. Коефіцієнти ініціалізуються випадковими даними з нормального розподілу.

Після п’ятого згорткового шару вихідний тензор розгладжується, щоб отримати одновимірний масив. Після цього нормалізується.

Після згладжувального шару йдуть два повноз’єднаних шари, після кожного з яких дані нормалізуються.

Повно з’єднані шари під’єднані до останнього шару з 4 нейронами, де в якості функції активації softmax.

Візуалізація архітектури використаної нейронної мережі наведено у Додатку Г.

Функція помилки – категоріальна кросентропія, а для мінімізації даної функції під час навчання використовується оптимізатор Adam з кроком навчання 0.001.

Дана функція складається з 8 шарів (якщо не рахувати технічні шари як-то функція активації та нормалізація), тому є дуже високий ризик перенавчання моделі. Для уникнення перенавчання використовуватиметься 2 види регуляризації: комбінований регуляризатор L1 та L2 на всіх шарах та

виключення (dropout) на повноз’єднаних шарах в кінці моделі. Коефіцієнти регуляризації для комбінованого регуляризатора L1 та L2 становлять 0.01 та 0.01 відповідно, а для дропауту – 0.5.

На кожній епосі навчання визначатиметься точність та значення категоріальної кросентропії на навчальних та тестових даних.

3.1.3 Оцінка якості моделі

Модель навчалась впродовж 250 епох, де на кожному етапі фіксувались заміри двох метрик на тренувальних та тестових даних: точність та категоріальна крос-ентропія. Прикінцева точність для тренувальної вибірки склала 76.9%, а для тестової – 74.5%. Значення категоріальної кросентропії склало 4.7 для тренувальної вибірки та 4.8 для тестової.

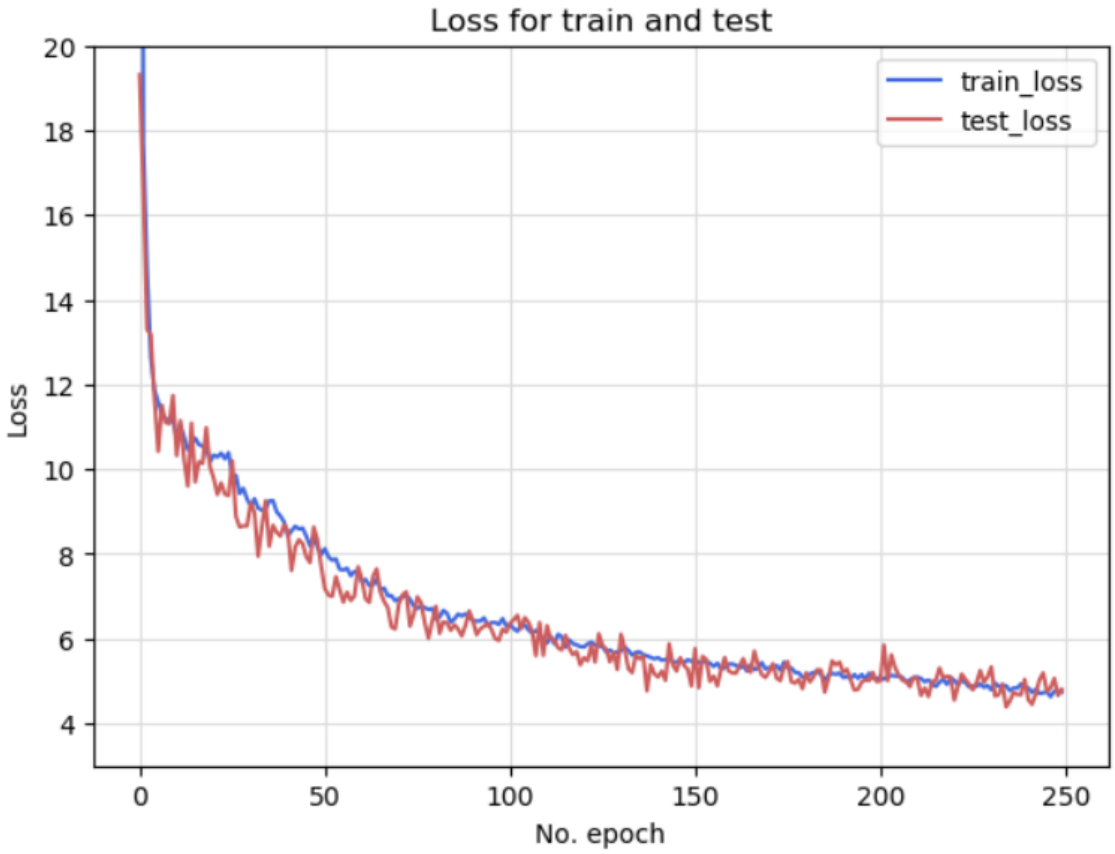


Рис. 3.9 – Категоріальна кросентропія на тестових і тренувальних даних

З рис. 3.9 видно, що зміщення мж категоріальною кросентропією навчальної та тестової вибірки майже відсутнє. На функції втрат на тестовій

вибірці спостерігається невеликий шум. На рис. 3.10 зображено точність на тренувальній та тестовій вибірках. Точність на тестовій виборці майже одразу стає високою в порівнянні з тестовою вибіркою, а далі випадково піднімаються та опускається з тенденцією до зростання. Згодом зашумленість точності стає меншою. Зашумленість як на графіку функції втрат так і на графіку точності можна пояснити тим, що в тестовій виборці константний набір даних в той час як тренувальна вибірка постійно незначним чином змінюється. Тому патерни, вивчені на новій аугментованій виборці, то погіршують то поліпшують точність на тестовій виборці. Поступове зменшення зашумленості відбувається через поступову генералізацію моделі.

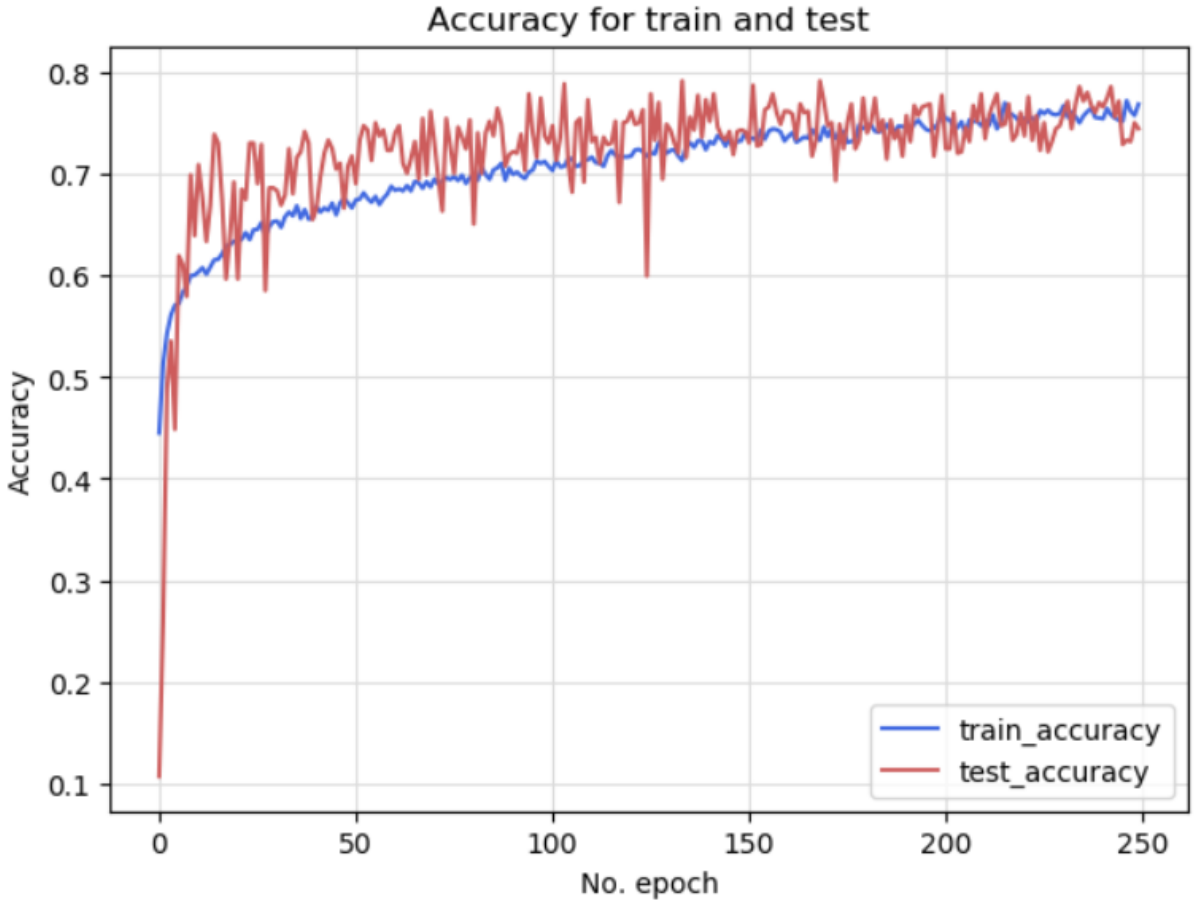


Рис. 3.10 – Точність моделі на тренувальних і тестових даних

На рис. 3.11 представлено матрицю невідповідностей для тренувальної вибірки нормалізовану по істинним міткам класів. Варто відзначити те, що значення представлені на головній діагоналі (повнота визначення класу)

мають відносно однакове значення і є доволі високими для класів, які були мало представлені в наборі даних.

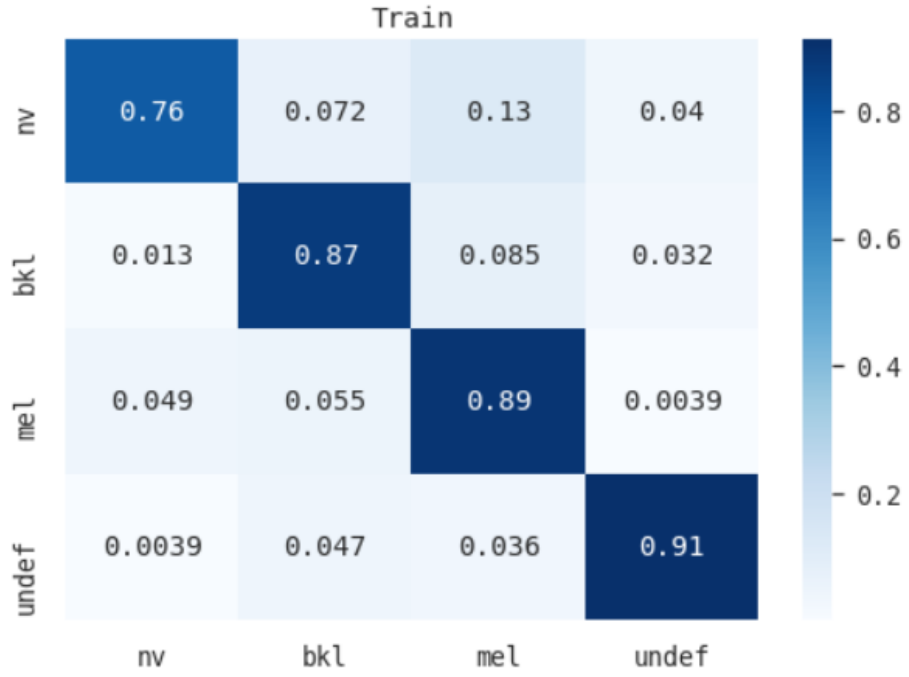


Рис. 3.11 – Матриця невідповідностей для тренувальних даних

На рис. 3.12 представлена матриця невідповідностей для тестових даних. Показники повноти для класів нижчі, ніж для тренувальних даних, однак теж мають приблизно однакові значення по діагоналі

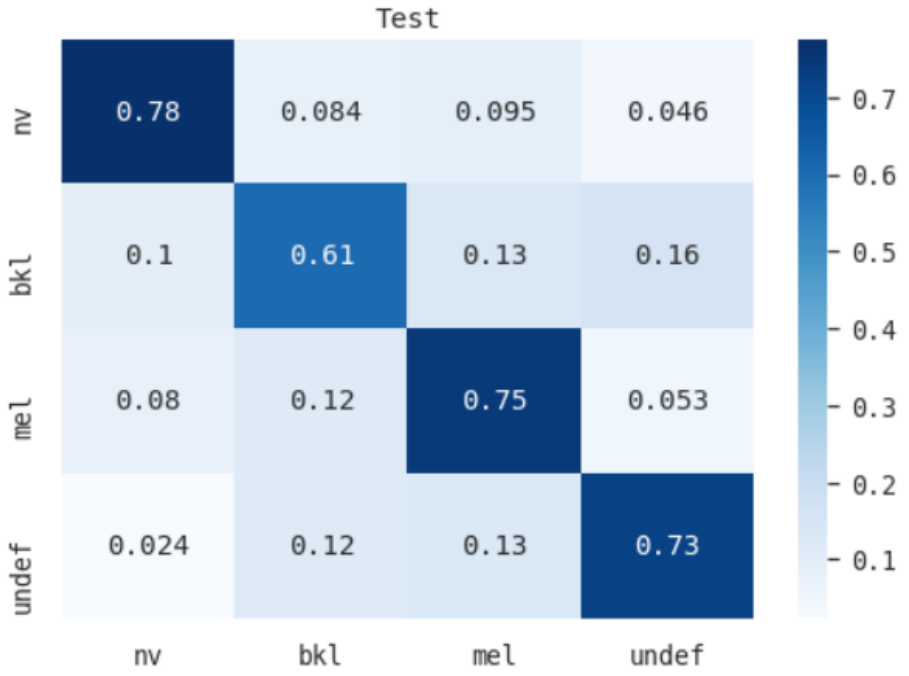


Рис. 3.12 – Матриця невідповідностей для тестових даних

Можна зробити висновок, що в моделі відбувається слабке перенавчання, однак вона має дуже високу узагальненість. Повнота визначення класів майже однакова для всіх класів, що говорить про те, що незважаючи на сильну незбалансованість набору даних, модель навчилась визначати різні класи однаково точно.

3.2 Розробка вебзастосунку для користування системою

Проєкт має модульну структуру (рис. 3.13). В кореневому каталозі знаходиться дві директорії – *project* та *tests*. Також файл app.py, в якому відбувається запуск сервера. В директорії *tests* знаходяться *unit*-тести, які запускаються після кожного коміту в віддалене сховище *GitHub*. В директорії *project* знаходяться основні компоненти роботи програми.

В директорії *analyze* знаходяться файли *__init__.py*, *forms.py* та *utils.py*. В файлі *__init__.py* реалізовані функції відображення для ендпоінтів. В файлі *forms.py* – знаходиться клас, що спадкується від класу *FlaskForm* з пакету *flask* для реалізації форми завантаження зображення для аналізу. В файлі utils.py реалізовано клас *AnalyzerManager*, який реалізує бізнес-логіку проєкту та клас *ImageAngPredictions*, який містить в собі результати класифікації – відносний шлях до зображення, що аналізувалось, відносний шлях до діаграми ймовірностей належності до класів, яка відображається користувачу системи, та словник з ймовірностями.

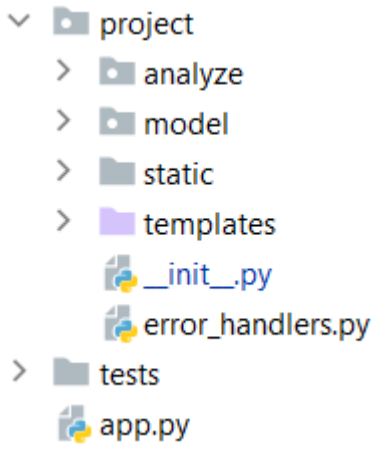


Рис. 3.13 – Структура проєкту

В директорії *model* розташований файл *__init__.py*, в якому міститься абстрактний клас *ImageClassifierAbstract* з абстрактним методом *get_prediction(self, mole_image: np.ndarray)*, який приймає на вхід зображення, перетворене в чотиривимірний numpy-масив. Цей клас з оголошеним абстрактним методом є інтерфейсом для використання класифікатора в даному проєкті. Також в даному файлі міститься конкретна реалізація абстрактного класу – *ImageClassifierCNN* і дата-клас *PredictionsForImage*.

В директорії *static* містяться файли для відображення на вебсторінках, а в директорії *templates* містяться *html*-документи, для розмітки вебсторінок.

В файлі *error_handlers.py* реалізовані функції відображення для сторінок з помилками запитів, а в фалі *__init__.py* міститься фабрика додатків (рис. 3.14), яка створює та налаштовує об'єкт Flask-додатку.

```
def create_app() -> Flask:
    """
    Flask application factory
    :return: Flask object
    """
    app = Flask(__name__)

    app.config["SECRET_KEY"] = environ["FLASK_SECRET_KEY"]

    @app.route('/')
    def index():
        return render_template("index.html")

    @app.route('/info')
    def info():
        return render_template("info.html")

    app.register_blueprint(analyze_bp)
    app.register_blueprint(error_pages_bp)

    return app
```

Рис. 3.14 – Скріншот програмного коду з функцією налаштування вебсерверу

При налаштуванні додатку в фабриці додатку із змінних середовища береться секретний ключ, який використовується *flask* для підпису кук і

роботи деяких розширень, наприклад *Flask-BCrypt*. Потім створюється дві функції відображення для головної сторінки та для сторінки з інформацією про проєкт. Реєструються блюпринти – ескізи додатку, які використовуються для легшого розбиття коду на модулі. Після налаштувань функція повертає готовий застосунок.

Основні ендпоінти додатку описані в модулі *analyze*. Функції представлення використовують об’єкт класу *AnalyzerManager*. Цей клас в якості інтерфейсу надає метод *make_predictions(pic_uploaded)*, який приймає на вхід завантажене користувачем зображення, обробляє його та повертає результат, упакований об’єкт в дата-класу *ImageAndPrediction*. Всередині класу *AnalyzerManager* використовуються псевдоприватні методи *__temporary_save_image(pic_uploaded)* для збереження зображення на сервері (щоб потім сформувати сторінку з результатом) та *__get_next_counter()* який повертає унікальний *id*, який використовується для формування назви збереженого зображення для кожного запиту. Псевдоприватне поле *__classifier* містить об’єкт класу класифікатора, нащадка абстрактного класу *ImageClassifierAbstract*, який використовується для класифікації зображення.

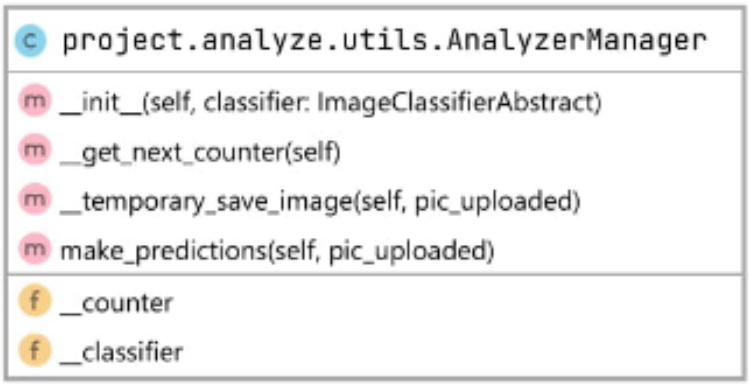


Рис. 3.15 – Діаграма класу *AnalyzerManager*

Для зручності використання класифікатору, було створено абстрактний клас *ImageClassifierAbstract* з публічною функцією *get_prediction(mole_image)*, яка приймає на вхід зображення представлене в форматі numpy-масиву і повертає передбачення, який слугує інтерфейсом для використання конкретних інтерфейсів в бізнес-логіці програми. Всередині класу

використовується метод *__get_next_counter()* для отримання унікального ідентифікатора, який використовуватиметься для збереження діаграми ймовірностей на сервері для подальшого використання. Функція *__make_plot_and_save(pred_dict)* приймає на вхід словник з ймовірностями і будує графік, для наочності результатів, після чого зберігає його на сервері та повертає відносний шлях до файлу з зображенням діаграми. Функція *__pred_array_to_dict(pred_array)* приймає на вхід масив з ймовірностями, який повертає нейронна мережа, та перетворює його на словник, де ключ – мітка класу, а значення – ймовірність. В псевдоприватному полі *__model* збрегігається об’єкт keras-моделі нейронної мережі, яка використовується для аналізу зображення. При ініціалізації класу параметри моделі зчитуються з файлу *serialized_model.h5*. Якщо буде отримана нова більш точна модель в ході дослідження, систему можна оновити просто замінивши файл з параметрами на новий. В псевдоприватному полі *__labels* міститься словник відповідності індексів в масиві ймовірностей, які повертає модель, та відповідних міток класів в форматі повної назви.

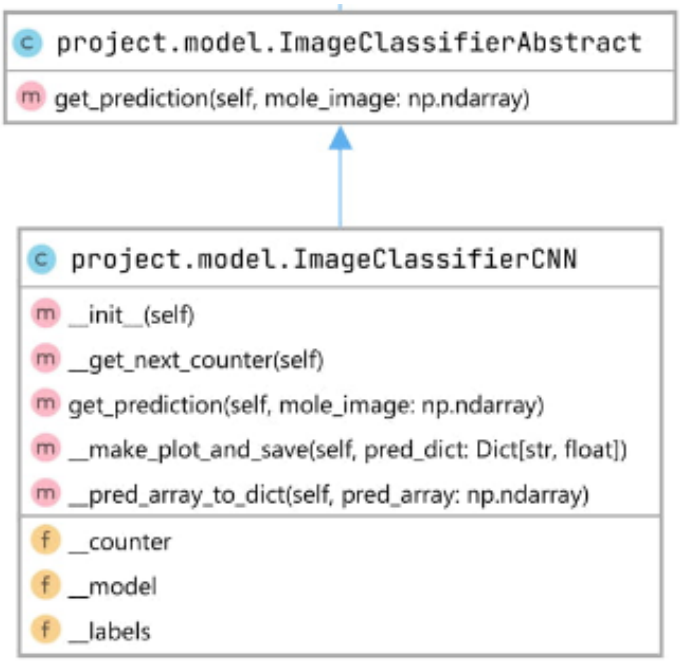


Рис. 3.16 – Діаграма класів *ImageClassifierAbstract* та *ImageClassifierCNN*

Зображення, які аналізує система мають видалятися одразу після того як вони стають непотрібні. Це обумовлено тим, що фотографії пігментних

уражень відносяться до чутливої інформації, яку не варто зберігати без згоди користувача та обґрунтованої причини. До того ж, якщо видаляти фото уражень та діаграми одразу після того, як вони стали не потрібні, це зекономить місце на сервері.

```python
@analyze_bp.route("/success")
def success_analysis():
    predictions: dict = session["predictions"]

    images: List[str] = [
        os.path.join(current_app.root_path, "static", *predictions["plot_path"].split("/")),
        os.path.join(current_app.root_path, "static", *predictions["image_path"].split("/"))
    ]

    del_thread: Thread = Thread(target=delay_delete, args=(60, images))
    del_thread.start()

    return render_template("analyze/success_page.html",
                           plot=predictions["plot_path"],
                           image=predictions["image_path"])


def delay_delete(delay: int, images: List[str]) -> None:
    sleep(delay)

    for img_path in images:
        os.remove(img_path)

    return
```

Рис. 3.17 – Скріншот реалізації функції представлення для ендпоінту */analysis/success* та функції *delay_delete()*

Для реалізації видалення зображень із серверу створено функцію *delay_delete(delay, images)*, яка приймає два параметри: *delay* – час, який очікуватиметься перед видаленням зображень і *images* – список абсолютних шляхів до об'єктів, які треба видалити. В функції представлення *success_analysis()* створюється потік, в який передається функція *delete_delay()* та відповідні аргументи для цієї функції. В якості часу очікування було обрано 60 секунд. Перед надсиланням результату користувачу потік активується. В заголовках відповіді зазначається, що зображення, які надсилаються клієнтові, не потрібно кешувати. Відносні шляхи до діаграми ймовірностей і до вхідного зображення передаються в якості іменованого аргументу в функцію

render_template(). Потім вони будуть підставлені в HTML-файл, який буде надіслано користувачу, за допомогою інструменту Jinja2.

3.3 Налаштування Docker

Для спрощення процесу розгортання серверу на різних хостах використовується Docker. Для цього в кореневій директорії проєкту було створено файл *Dockerfile* (рис. 3.18), в якому описані команди, які виконуватимуться для створення образу.

```
FROM python:3.7.2

ENV FLASK_APP=app
ENV FLASK_ENV=production

RUN mkdir /app
WORKDIR /app

COPY requirements.txt ./
RUN pip install --no-cache-dir -r requirements.txt

ADD ./project ./project
COPY app.py .
```

Рис. 3.18 – Скріншот *Dockerfile*

Командою *FROM* вказується батьківський образ, який використовуватиметься для побудови нового образу, в даному випадку це Python-3.7.2. Далі додаємо змінні середовища: *FLASK_APP*, яка вказує який файл необхідно запускати для старту серверу, коли виконується команда *flask run*, та *FLASK_ENV*, в якій міститься тип запуску – розробка чи продакшн. Далі створюється директорія, в які будуть міститися файли проєкту за допомогою команди *RUN mkdir app* і ця директорія робиться робочою директорією. Копіюється файл з залежностями проєкту *requirements.txt*, після чого ці залежності встановлюються за допомогою утиліти *pip install*. Після встановлення додаткових бібліотек, копіюються основні файли проєкту до директорії.

В файлі *docker-compose.yml (рис. 3.19)* прописуються сервіси, які створюються для цього додатку. В даному випадку тільки один сервіс – запуск сервера, однак такий підхід робить простішим потенційне розширення додатку. Цей файл може редагуватись адміністратором системи для налаштування запуску серверу на хості. В полі *image* вказується назва образу, який буде створено. В полі *command* вказується команда, якою запускається сервіс у вигляді списку, де перший елемент команда, а інші – опції. В полі *environment* вказуються змінні середовища. Значення змінної *FLASK_SECRET_KEY* береться із змінної середовища з таким же ім'ям. В полі *ports* вказуються порти, які будуть поставлені у відповідність в форматі *<порт хоста, на який надсилатимутися запити>:<порт який прослуховує сервер в середині контейнеру>*. Порт хоста можна змінити на той, який планується використовувати. Порт в середині хоста та порт, вказаний в аргументах команди запуску сервера, мають співпадати. В поле *restart* вписане значення *"always"* щоб сервіс перезапускався завжди після падіння.

```yaml
version: '3'

services:
  run-app:
    build: .
    image: diploma-project-image
    command: [ "flask", "run", "--host=0.0.0.0", "--port=5000" ]
    environment:
      - FLASK_APP=app
      - FLASK_ENV=production
      - FLASK_SECRET_KEY=${FLASK_SECRET_KEY}
    ports:
      - "5000:5000"
    restart: "always"
```

Рис. 3.19 – Скріншот *docker-compose.yml*

Висновки до розділу 3

В даному розділі було проаналізовано метадані, які були викладені в якості доповнення до набору даних HAM10000. Після ряду перевірок розподілів додаткової інформації та порівняння її із клінічними статистичними даними було прийняте рішення не використовувати ці дані для навчання моделі, а обмежитись лише мітками класів та відповідними зображеннями, адже майже вся додаткова інформація не несла жодної корисної інформації.

Для класифікації зображень було побудовано згорткову нейронну мережу, в якій 5 згорткових шарів, два повноз'єднаних шара та вихідний шар з функцією активації softmax. В якості цільової функції для оптимізації використовувалась категоріальна кросентропія, а в якості методу оптимізації обрано оптимізатор Адама. Окрім категоріальної кросентропії на кожному кроці розраховувалась точність моделі. Для регуляризації використовувався комбінований регуляризатор L1-L2 і випадкове виключення нейронів з ймовірністю 50% на повноз'єднаних шарах. Для вирішення дисбалансу класів було додано шар аугментації даних, а в функції помилки враховувались ваги класів.

Точність моделі, яку вдалося досягнути – 76.9% на тренувальній вибірці і 74.5% на тестовій. Незважаючи на сильну незбалансованість класів, повнота визначення кожного класу була приблизно на одному рівні.

Окрім моделі було наведено детальний опис розробленої системи, яка є інтерфейсом для використання моделі.

Для зручного розгортання серверу на хостах, було описано конфігураційні файли для створення docker-образу.

РОЗДІЛ 4. АНАЛІЗ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ

4.1 Запуск сервера

Для роботи серверу необхідно створити образ, який згодом запускатиметься в оркестрі (наборі взаємопов'язаних контейнерів, мультиконтейнері). За необхідності можна відреагувати файл *docker-compose.yml*, вказавши потрібні параметри. Для цього необхідно перейти в кореневу директорію проєкту ввести команду docker-compose built, яка створить образ мультиконтейнера. На рис. 4.1 наведено логи після запуску команди. Останній рядок вказує на успішне виконання та виводить унікальний ідентифікатор створеного образу.

```
(env) C:\edu\Diploma\SkinDiseasesServer>docker-compose build
Building run-app
failed to get console mode for stdout: The handle is invalid.
[+] Building 2.7s (13/13) FINISHED
 => [internal] load build definition from Dockerfile                    0.0s
 => => transferring dockerfile: 259B                                    0.0s
 => [internal] load .dockerignore                                       0.0s
 => => transferring context: 2B                                         0.0s
 => [internal] load metadata for docker.io/library/python:3.7.2         2.4s
 => [auth] library/python:pull token for registry-1.docker.io           0.0s
 => [internal] load build context                                       0.0s
 => => transferring context: 1.66kB                                     0.0s
 => [1/7] FROM docker.io/library/python:3.7.2@sha256:1eecfb3b9cee73af760e  0.0s
 => CACHED [2/7] RUN mkdir /app                                         0.0s
 => CACHED [3/7] WORKDIR /app                                           0.0s
 => CACHED [4/7] COPY requirements.txt ./                               0.0s
 => CACHED [5/7] RUN pip install --no-cache-dir -r requirements.txt     0.0s
 => CACHED [6/7] ADD ./project ./project                                0.0s
 => [7/7] COPY app.py .                                                 0.0s
 => exporting to image                                                  0.1s
 => => exporting layers                                                 0.0s
 => => writing image sha256:405468cedd4fc6fcfaac884fdd4b281d9126bf05accd2  0.0s
 => => naming to docker.io/library/diploma-project-image                0.0s
Successfully built 405468cedd4fc6fcfaac884fdd4b281d9126bf05accd2097afd479efa6927994
```

Рис. 4.1 – Скріншот логів створення образу

Після цього образ мультиконтейнера буде збережений в локальному реєстрі. Для того щоб розгорнути сервер потрібно запустити образ, який

створить активний мультиконтейнер з сервером. При запуску контейнера, локальний порт та внутрішній порт, вказані в файлі *docker-compose.yml*, з'єднаються і сервер відповідатиме на запити.

4.2 Демонстрація функціоналу програмного продукту

При звернені до системи за адресою, на якій розгорнуто сервер (в даному випадку *localhost*), користувачу буде представлена домашня сторінка (рис. 4.2). В верхньому меню є посилання на головну сторінку, посилання на сторінку з інформацією про систему та сторінку, де можна проаналізувати зображення. Також на сторінці наведено короткий опис проєкту та кнопка-запрошення для завантаження зображення на аналіз.

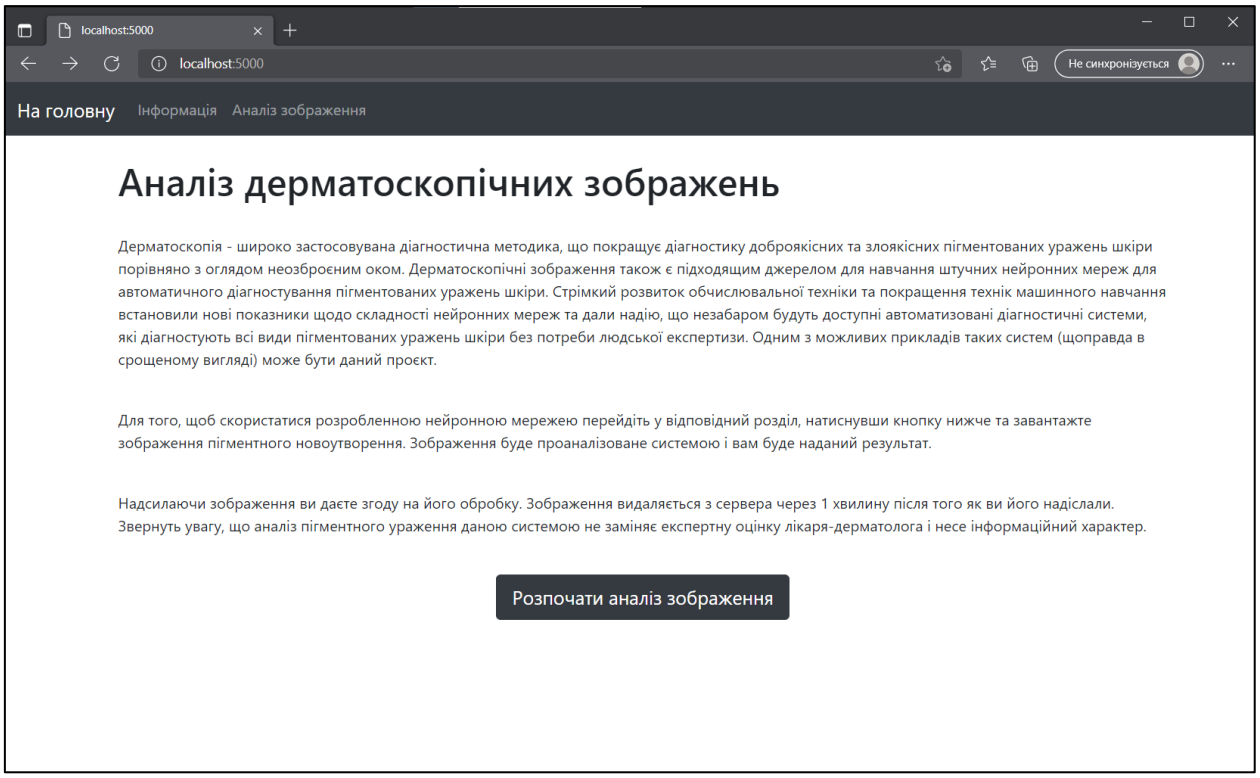


Рис. 4.2 – Головна сторінка

При переході на сторінку для завантаження зображення користувачу демонструється сторінка з формою для прикріплення файлу та кнопка підтвердження. Якщо кнопка «Проаналізувати зображення» буде натиснута без додавання файлу, користувач залишиться на тій же сторінці і йому буде показано попередження про те, що це обов'язкова дія. Над кнопкою наведено

застереження про те, що дана система не є заміною візиту до лікаря і містить виключно інформативний характер. Також наведено інформацію про видалення файлу з серверу.

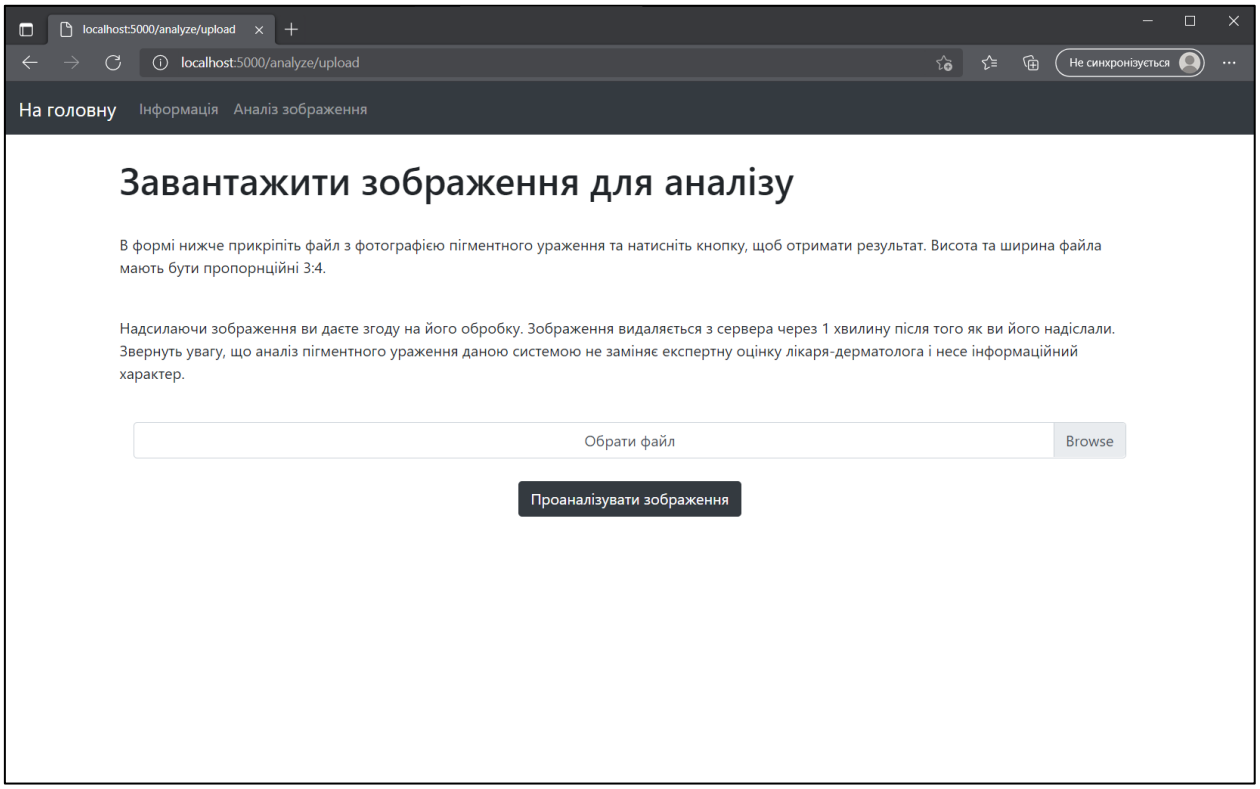


Рис. 4.3 – Сторінка для завантаження файлу

Після того, як користувач завантажив файл і натиснув кнопку, сервер проаналізує зображення, перенаправить користувача на сторінку з результатом виконання аналізу. На цій сторінці буде показане зображення, яке завантажив користувач з лівого боку та діаграма ймовірностей з правого боку. На діаграмі є 4 комірки, кожна з яких містить стовпець, довжина яких пропорційна ймовірності того, що зображення належить до одного з чотирьох класів. З лівого боку від стовпця наведено числове значення ймовірності (від 0 до 1). Сума ймовірностей дорівнює 1, адже система вирішує задачу багатокласової класифікації з класами, які не перетинаються.

На рис. 4.4 зображено результат виконання програми після завантаження фотографії меланоцитного невуса. Згідно результату, ймовірність того, що це меланоцитний невус складає 95.5%. Ймовірність належності до інших класів майже близька до 0%.

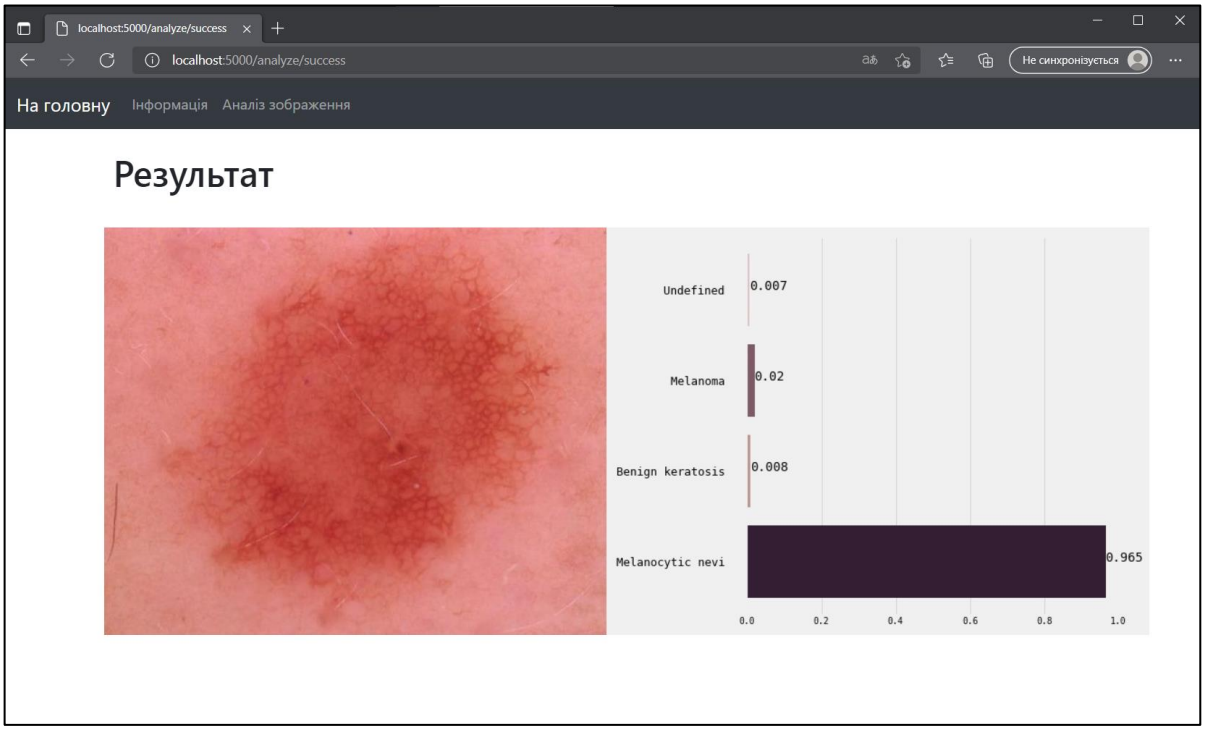


Рис. 4.4 – Результат аналізу зображення з меланоцитним невусом

На рис. 4.5 наведено результат аналізу зображення з меланомою. Згідно результату, ймовірність того, що дане ураження – саме меланома, складає 89.5%, ймовірність того, що це – меланоцитний невус – 8.1%. Ймовірність інших діагнозів близька до 0%.

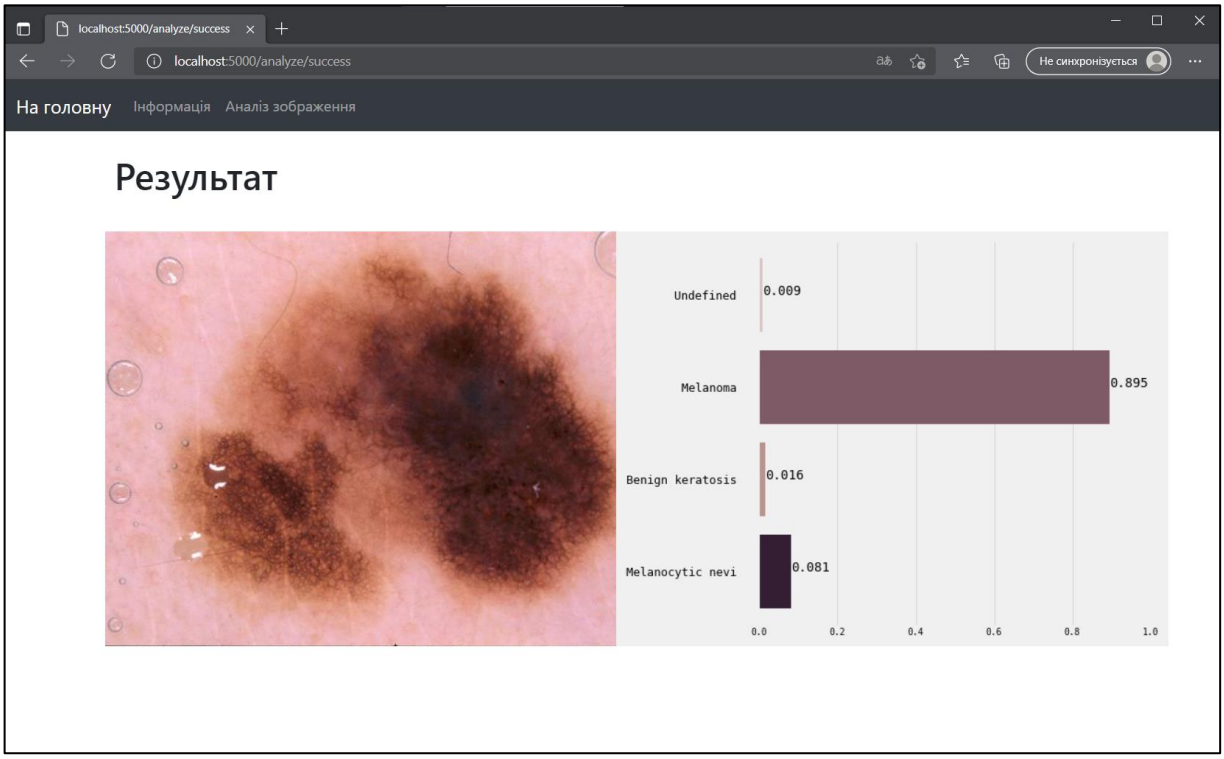


Рис. 4.5 – Результат аналізу зображення з меланомою

Якщо користувач звернеться до ендпоїнту, обробку якого не передбачено системою, користувачу буде показано сторінку з відповідним повідомленням (рис. 4.6). На цій сторінці буде верхнє меню, яке дозволить перейти до головної сторінки.

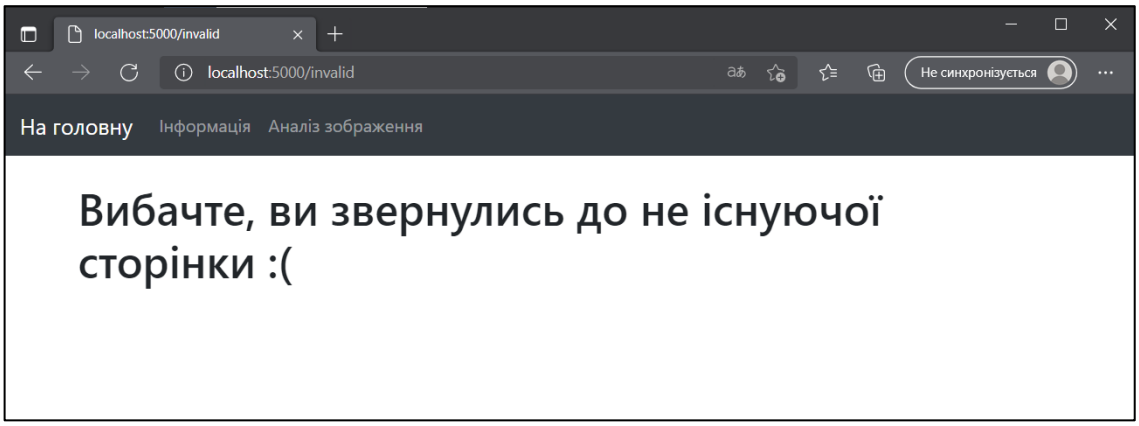


Рис. 4.6 – Повідомлення про те, що сторінки не існує.

Висновки до розділу 4

В даному розділі було наведено алгоритм розгортання серверу на локальному хості за допомогою Docker.

Також було розглянуто принципи взаємодії кінцевого користувача та розробленої системи.

Задачу, поставлену в даній роботі, було виконано: користувач має змогу аналізувати зображення пігментних уражень за допомогою зручного вебзастосунку; встановлення додатку з усіма залежностями не складає проблем, адже для цього достатньо створити docker-образ; інтеграцію нової моделі в систему можна здійснити замінивши файл з параметрами, а в разі змінення формату вихідних параметрів моделі достатньо створити новий клас для класифікатора, успадкований від абстрактного класу *ImageClassifierAbstract*.

ВИСНОВКИ

В першому розділі було розглянуто область комп'ютерного зору та сформовано задачу, яку необхідно вирішити в даній роботі, в термінах машинного навчання. Також було описано принцип роботи нейронних мереж, методи регуляризації для уникнення перенавчання та обґрунтовано доцільність використання згорткової архітектури для задачі класифікації зображень.

В другому розділі було проаналізовано фреймворки для побудови і навчання нейронної мережі, обґрунтовано вибір мови програмування, вебсерверної бібліотеки, додаткових інструментів для розробки клієнт-серверного додатку і інструмент для швидкого розгортання серверу на хості. Також було розглянуто набір даних HAM10000 для навчання нейронної мережі.

В третьому розділі було проаналізовано метадані обраного набору даних і обґрунтовано відмову у використанні додаткових даних для побудови моделі машинного навчання. В даних було виявлено проблему – незбалансованість вибірки і запропоновано методи вирішення: аугментація даних та використання зваженої функції помилки. Далі було детально описано архітектуру нейронної мережі і проаналізовано результати навчання метриками якості на тестовій і на навчальній вибірках на кожній епосі. Також було наведено детальний опис програмної структури серверного додатку, описано функції та взаємодію використаних класів. В кінці було описано конфігураційні файли для створення docker-образу.

В четвертому розділі було наведено інструкцію з розгортання серверу на хості та наведено приклад взаємодії кінцевого користувача з системою.

В результаті навчання моделі було досягнуто точності 76.9% для навчальної вибірки і 74.5% для тестової вибірки. Повнота визначення для кожного з класів була майже на одному рівні, що говорить про те, що проблема

з дисбалансом класом не сильно вплинула на точність розпізнавання окремих класів завдяки штучному збільшенню даних шляхом аугментації та використанню зваженої функції помилки. Також варто відмітити, що різниця в точності 2.5% між тестовою і тренувальною вибіркою говорить про хорошу узагальненість моделі.

Для використання моделі було розроблено зручний вебзастосунок. Завдяки гнучкій організації програми, є змога ітеративно поліпшувати якість моделі та оновлювати систему, замінивши файл з параметрами і перезапустивши сервер.

Дана модель розроблена для використання молодшим медичним персоналом для попередньої оцінки ризиків пігментних уражень на шкірі, що потенційно може зробити процедуру дерматоскопії доступнішою через зменшення собівартості. Для використання моделі так максимізації користі потрібно оцінити її на практиці та розробити відповідні протоколи інтерпретації результатів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Dermoscopy improves accuracy of primary care physicians to triage lesions suggestive of skin cancer / G. Argenziano et al. *Journal of clinical oncology*. 2006. Vol. 24, no. 12. P. 1877–1882. URL: https://doi.org/10.1200/jco.2005.05.0864 (дата звернення: 30.05.2021).

2. Landi H. Investors poured $4B into healthcare AI startups in 2019. Fierce Healthcare. URL: https://www.fiercehealthcare.com/tech/investors-poured-4b-into-healthcare-ai-startups-2019 (дата звернення: 30.05.2021).

3. The Image Classification Dataset URL: https://d2l.ai/chapter_linear-networks/image-classification-dataset.html (дата звернення: 30.05.2021).

4. James Le. The top 10 machine learning algorithms every beginner should know. 2019. URL: https://builtin.com/data-science/tour-top-10-algorithms-machine-learning-newbies (дата звернення: 30.05.2021).

5. YOLOv3 object detection in TensorFlow 2.x. URL: https://medium.com/analytics-vidhya/yolov3-object-detection-in-tensorflow-2-x-8a1a104c46a8 (дата звернення: 30.05.2021).

6. Vinorth Varatharasan, Hyo-Sang Shin, Antonios Tsourdos and Nick Colosimo. Improving learning effectiveness for object detection and classification in cluttered backgrounds. URL: https://www.researchgate.net/publication/339616270_Improving_Learning_Effectiveness_For_Object_Detection_and_Classification_in_Cluttered_Backgrounds (дата звернення: 30.05.2021).

7. Computer vision tutorial: a step-by-step introduction to image segmentation techniques (Part 1). URL: https://www.analyticsvidhya.com/blog/2019/04/introduction-image-segmentation-techniques-python/ (дата звернення: 30.05.2021).

8. Про захист персональних даних: Закон України від 01.06.2010 р. № 2297-VI: станом на 23 квіт. 2021 р. URL:

https://zakon.rada.gov.ua/laws/show/2297-17 (дата звернення: 30.05.2021).

9. Alyoubi W. L., Shalash W. M., Abulkhair M. F. Diabetic retinopathy detection through deep learning techniques: A review. *Informatics in Medicine Unlocked.* 2020. Vol. 20. P. 100377. URL: https://doi.org/10.1016/j.imu.2020.100377 (дата звернення: 30.05.2021).

10. Pedro R. A. S. Bassi, Romis Attux. A deep convolutional neural network for COVID-19 detection using chest x-rays. *Research on biomedical engineering.* 2021. URL: http://dx.doi.org/10.1007/s42600-021-00132-9 (дата звернення: 30.05.2021).

11. Liam Tung. Programming languages: Python developers now outnumber Java ones. URL: https://www.zdnet.com/article/programming-languages-python-developers-now-outnumber-java-ones/ (дата звернення: 30.05.2021).

12. Good news, Tensorflow chooses Keras! URL: https://github.com/keras-team/keras/issues/5050 (дата звернення: 30.05.2021).

13. The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. Sci Data. 2018. № 5. URL: https://doi.org/10.1038/sdata.2018.161 (дата звернення: 30.05.2021).

14. Melanoma skin cancer statistics. *American Cancer Society | Information and Resources about for Cancer: Breast, Colon, Lung, Prostate, Skin.* URL: https://www.cancer.org/cancer/melanoma-skin-cancer/about/key-statistics.html (дата звернення: 30.05.2021).

15. Signs of melanoma skin cancer | Symptoms of melanoma. *American Cancer Society | Information and Resources about for Cancer: Breast, Colon, Lung, Prostate, Skin.* URL: https://www.cancer.org/cancer/melanoma-skin-cancer/detection-diagnosis-staging/signs-and-symptoms.html (дата звернення: 30.05.2021).

Додаток А

СТРУКТУРНА СХЕМА РОБОТИ

до дипломного проєкту

на тему: «Система розпізнавання атипових плям на шкірі»

Київ – 2021 р.

					ІАЛЦ.467200.004 Д1		
Зм.	*Арк.*	*№ докум.*	*Підп.*	*Дата*			
Розроб.		*Музика О.А.*			*Система розпізнавання атипових плям на шкірі*	*Літ.*	*Аркуш*	*Аркушів*
Перев.		*Сергієнко А.А.*				Т	*1*	
					Структурна схема роботи	*НТУУ «КПІ ім. Ігоря Сікорського» ФІОТ Група ІВ-71*	
Н.контр.		*Сімоненко В.П.*					
Затв.		*Сергієнко А.А.*					

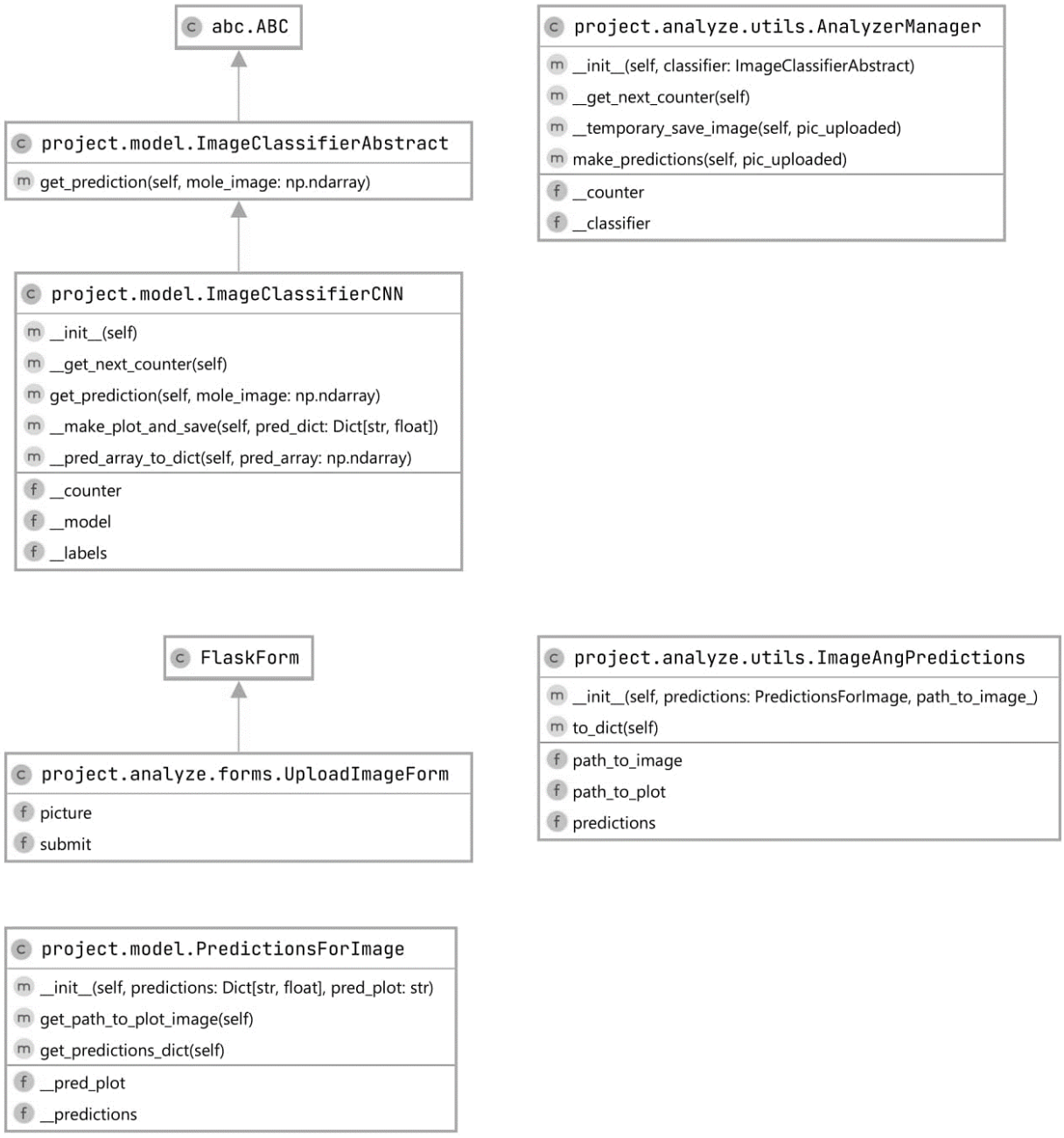
Додаток Б

ФУНКЦІОНАЛЬНА СХЕМА КЛАСІВ

до дипломного проєкту

на тему: «Система розпізнавання атипових плям на шкірі»

Київ – 2021 р.



					ІАЛЦ.467200.005 Д2		
Зм.	*Арк.*	*№ докум.*	*Підп.*	*Дата*			
Розроб.		*Музика О.А.*			*Система розпізнавання атипових плям на шкірі*	*Літ.*	*Аркуш*	*Аркушів*
Перев.		*Сергієнко А.А.*				T	*1*	
						НТУУ «КПІ ім. Ігоря Сікорського» ФІОТ Група ІВ-71		
Н.контр.		*Сімоненко В.П.*			*Функціональна схема класів*			
Затв.		*Сергієнко А.А.*						

Додаток В

ПРИНЦИПОВА СХЕМА РОБОТИ

до дипломного проєкту

на тему: «Система розпізнавання атипових плям на шкірі»

Київ – 2021 р.

					ІАЛЦ.467200.006 Д3		
Зм.	*Арк.*	*№ докум.*	*Підп.*	*Дата*			
Розроб.		*Музика О.А.*			*Система розпізнавання атипових плям на шкірі*	*Літ.*	*Аркуш*	*Аркушів*
Перев.		*Сергієнко А.А.*				Т	*1*	
					Принципова схема роботи	*НТУУ «КПІ ім. Ігоря Сікорського» ФІОТ Група ІВ-71*	
Н.контр.		*Сімоненко В.П.*					
Затв.		*Сергієнко А.А.*					

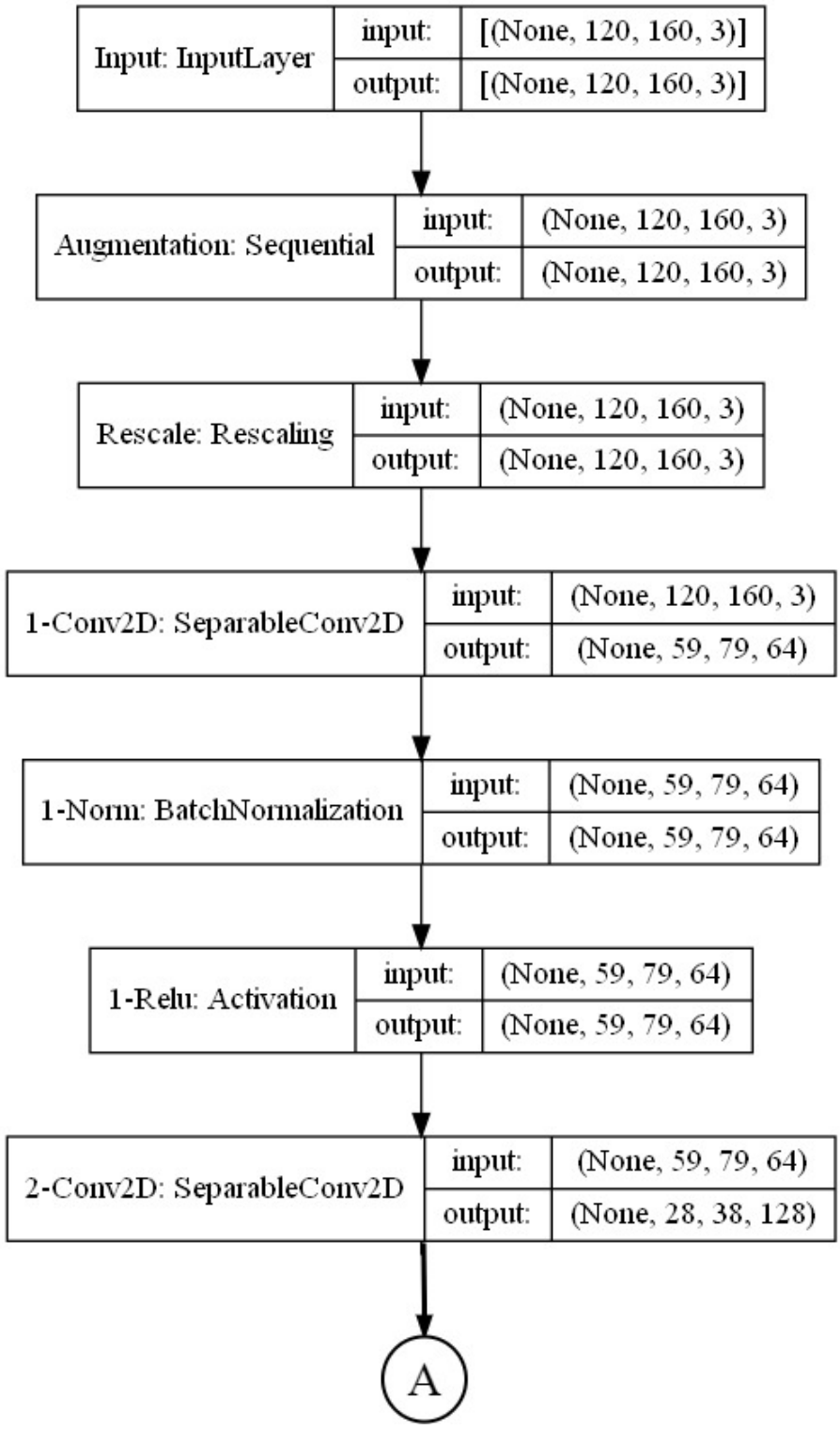
Додаток Г

АРХІТЕКТУРА НЕЙРОННОЇ МЕРЕЖІ

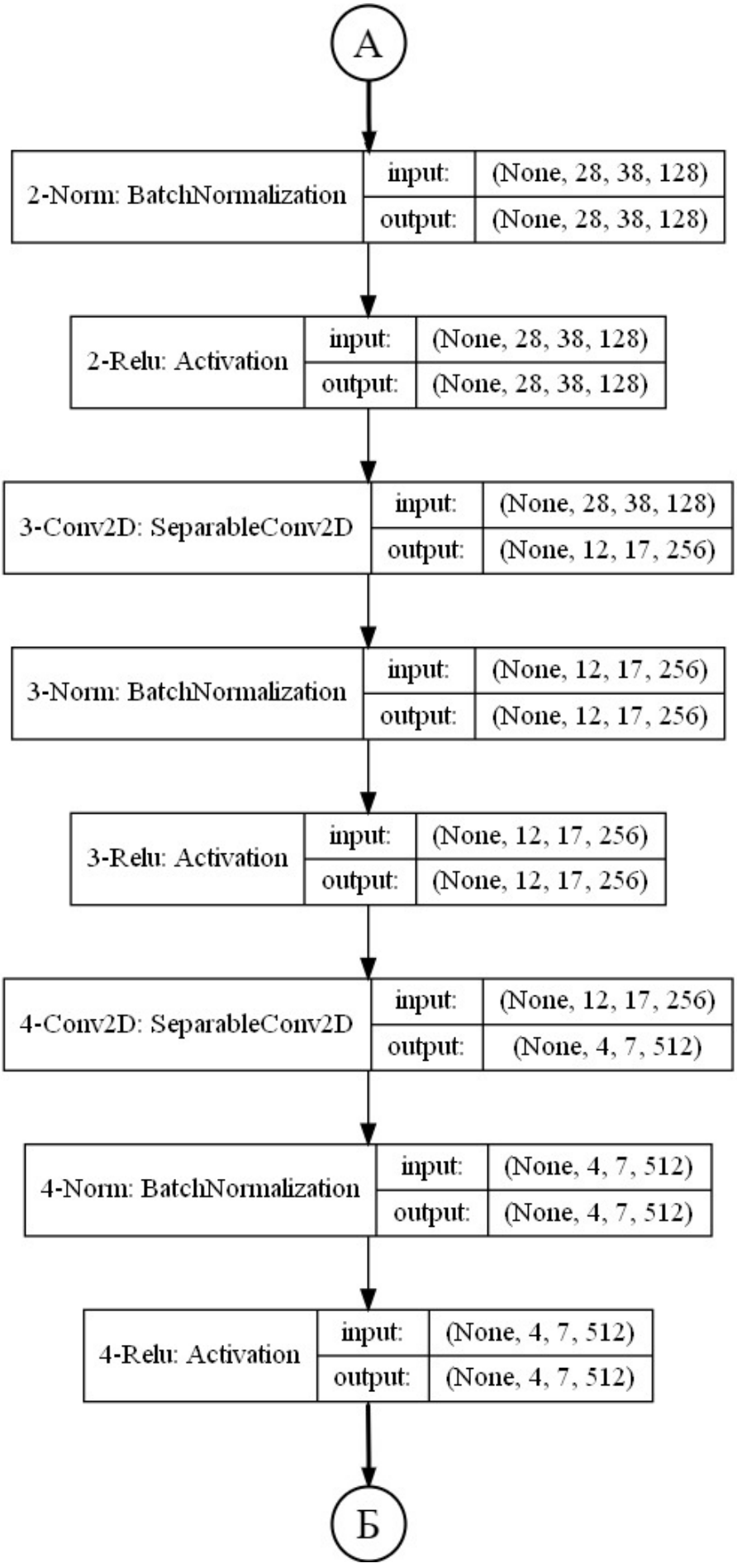
до дипломного проєкту

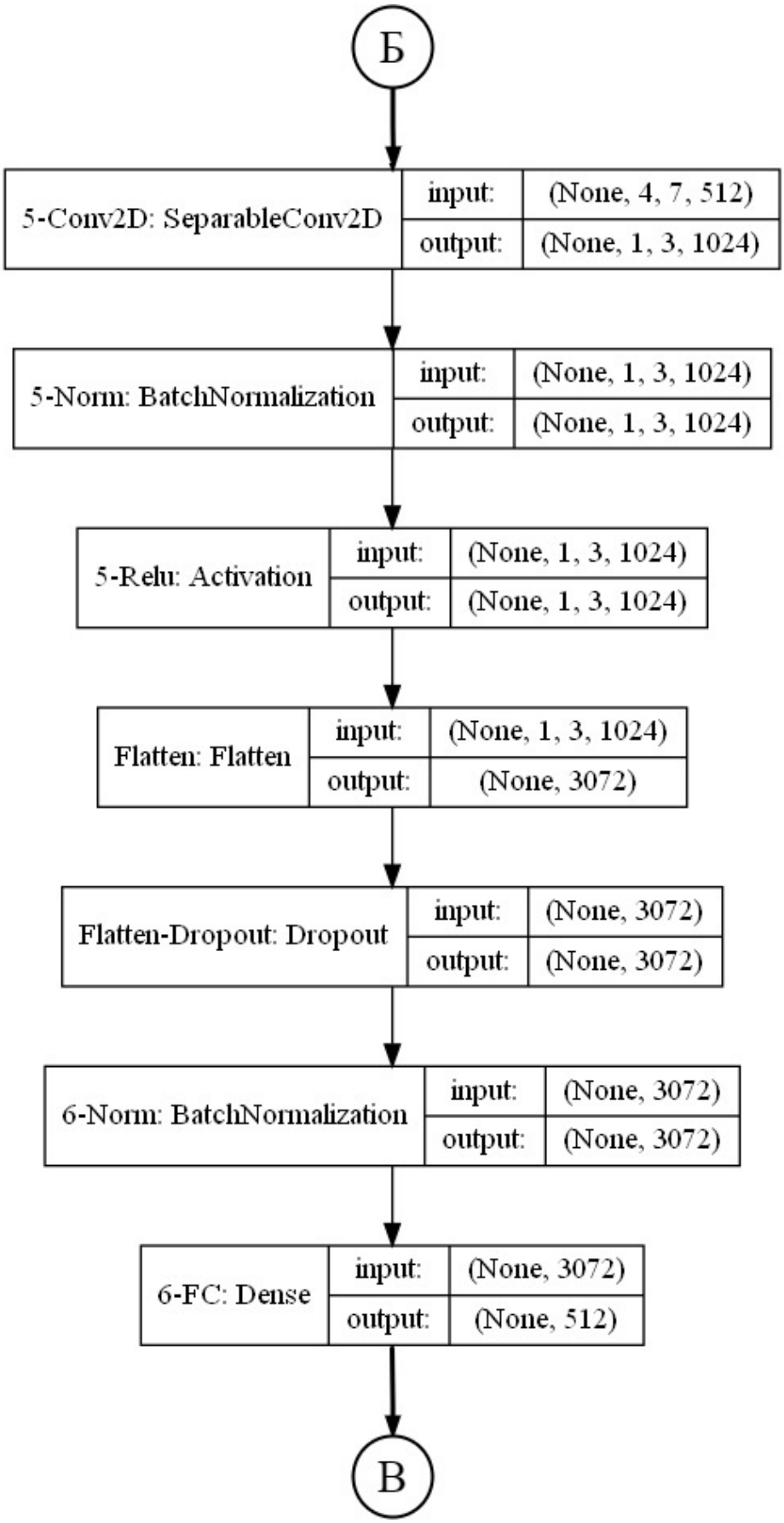
на тему: «Система розпізнавання атипових плям на шкірі»

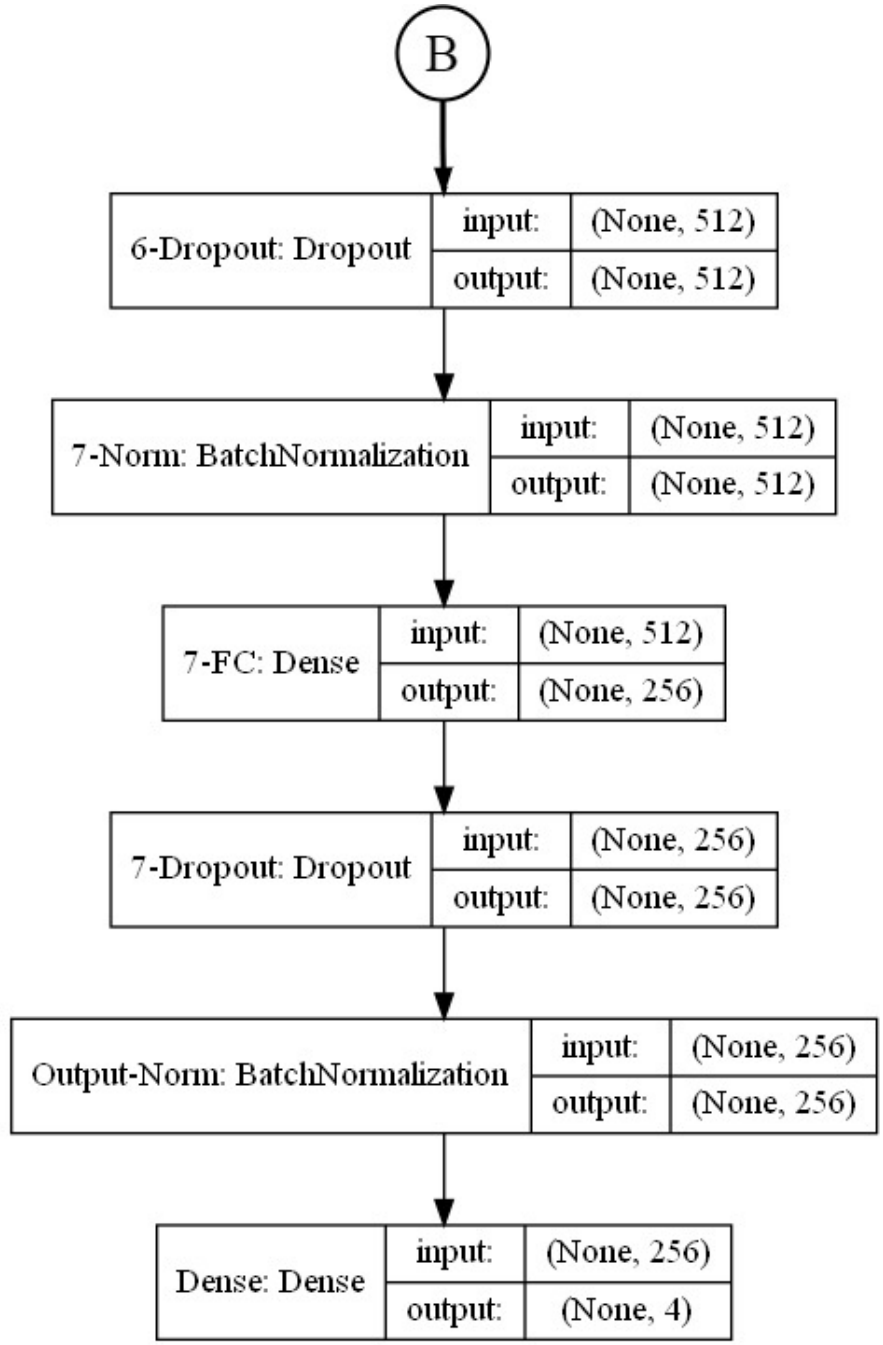
Київ – 2021 р.



					ІАЛЦ.467200.007 Д4		
Зм.	*Арк.*	*№ докум.*	*Підп.*	*Дата*			
Розроб.		*Музика О.А.*			*Система розпізнавання атипових плям на шкірі*	*Літ.*	*Аркуш*	*Аркушів*
Перев.		*Сергієнко А.А.*				Т	*1*	
					Архітектура нейронної мережі	*НТУУ «КПІ ім. Ігоря Сікорського» ФІОТ Група ІВ-71*	
Н.контр.		*Сімоненко В.П.*					
Затв.		*Сергієнко А.А.*					


А
2-Norm: BatchNormalization
input: (None, 28, 38, 128)
output: (None, 28, 38, 128)
2-Relu: Activation
input: (None, 28, 38, 128)
output: (None, 28, 38, 128)
3-Conv2D: SeparableConv2D
input: (None, 28, 38, 128)
output: (None, 12, 17, 256)
3-Norm: BatchNormalization
input: (None, 12, 17, 256)
output: (None, 12, 17, 256)
3-Relu: Activation
input: (None, 12, 17, 256)
output: (None, 12, 17, 256)
4-Conv2D: SeparableConv2D
input: (None, 12, 17, 256)
output: (None, 4, 7, 512)
4-Norm: BatchNormalization
input: (None, 4, 7, 512)
output: (None, 4, 7, 512)
4-Relu: Activation
input: (None, 4, 7, 512)
output: (None, 4, 7, 512)
Б


Б
5-Conv2D: SeparableConv2D
input: (None, 4, 7, 512)
output: (None, 1, 3, 1024)
5-Norm: BatchNormalization
input: (None, 1, 3, 1024)
output: (None, 1, 3, 1024)
5-Relu: Activation
input: (None, 1, 3, 1024)
output: (None, 1, 3, 1024)
Flatten: Flatten
input: (None, 1, 3, 1024)
output: (None, 3072)
Flatten-Dropout: Dropout
input: (None, 3072)
output: (None, 3072)
6-Norm: BatchNormalization
input: (None, 3072)
output: (None, 3072)
6-FC: Dense
input: (None, 3072)
output: (None, 512)
В


B
6-Dropout: Dropout
input: (None, 512)
output: (None, 512)
7-Norm: BatchNormalization
input: (None, 512)
output: (None, 512)
7-FC: Dense
input: (None, 512)
output: (None, 256)
7-Dropout: Dropout
input: (None, 256)
output: (None, 256)
Output-Norm: BatchNormalization
input: (None, 256)
output: (None, 256)
Dense: Dense
input: (None, 256)
output: (None, 4)