МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інтелектуальних інформаційних систем

_____ Євген СІДЕНКО

«____»_____ 2026 р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

ІНТЕЛЕКТУАЛЬНА СИСТЕМА КЛАСТЕРИЗАЦІЇ ОЗНАК ЗОБРАЖЕНЬ КРОВ'ЯНИХ КЛІТИН ЗА ДОПОМОГОЮ ЗГОРТКОВИХ НЕЙРОННИХ МЕРЕЖ

Спеціальність 122 Комп'ютерні науки

Освітня програма «Комп'ютерні науки»

Здобувач _____ Богдан СОМРЯКОВ

«___»_____ 2026 р.

Керівник канд. техн. наук, доцент _____ Євген СІДЕНКО

«___»_____ 2026 р.

Миколаїв – 2026

Чорноморський національний університет імені Петра Могили
(повне найменування закладу вищої освіти)

Факультет	Комп’ютерних наук
Кафедра	Інтелектуальних інформаційних систем
Рівень вищої освіти	Перший (бакалаврський)
Освітній ступень	Бакалавр
Спеціальність	122 Комп’ютерні науки
Освітня програма	Комп’ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем

_____ Євген СІДЕНКО

«____» _____ 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу здобувача

Сомрякова Богдана Олеговича
(прізвище, ім’я, по батькові здобувача)

1. Тема кваліфікаційної роботи «Інтелектуальна система кластеризації ознак зображень кров’яних клітин за допомогою згорткових нейронних мереж».
Керівник роботи Сіденко Євген Вікторович, завідувач кафедри інтелектуальних інформаційних систем, канд. техн. наук, доцент.
Затв. наказом Ректора ЧНУ ім. Петра Могили від «25» грудня 2025 р. № 353.
2. Строк представлення кваліфікаційної роботи студентом «__» _____ 2026 р.
3. Вхідними даними для кваліфікаційної роботи є мічений набір із 17 092 зображень кров’яних клітин, отриманих за допомогою аналізатора CellaVision DM96 у лабораторії Hospital Clinic of Barcelona. До складу набору входять зображення восьми типів нормальних периферичних клітин: нейтрофілів, еозинофілів, базофілів, лімфоцитів, моноцитів, незрілих гранулоцитів, еритробластів та

тромбоцитів. Усі зображення мають роздільну здатність 360×363 пікселі та були промарковані фахівцями – клінічними патологами.

4. Перелік питань, що підлягають розробці.

– аналіз сучасного стану задачі обробки та кластеризації медичних зображень;

– огляд існуючих методів і підходів до аналізу зображень та виділення ознак у задачах комп'ютерного зору;

– дослідження застосування алгоритмів штучного інтелекту для екстракції ознак і кластеризації медичних даних;

– розробка інформаційної системи для автоматизованої обробки, аналізу та кластеризації зображень клітин крові з використанням сучасних методів глибокого навчання.

– розробка вебзастосунку з інтегрованою базою даних для збереження датасетів та забезпечення зручної взаємодії користувача з системою.

5. Перелік графічного матеріалу: презентація.

Керівник роботи _____ Євген СІДЕНКО
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Здобувач _____ Богдан СОМРЯКОВ
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Дата видачі завдання «22» грудня 2025 р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема: Інтелектуальна система кластеризації ознак зображень кров'яних клітин за допомогою згорткових нейронних мереж

№	Найменування роботи	Початок	Закінчення	Примітки
1	Отримання завдання на виконання КР	21.12.2025	24.12.2025	Виконано
2	Аналіз предметної області та постановка задачі	25.12.2025	30.01.2026	Виконано
3	Огляд літературних джерел за темою кваліфікаційної роботи, зокрема огляд публікацій та аналогічних систем, щодо кластеризації ознак медичних зображень з використанням CNN	31.01.2026	01.03.2026	Виконано
4	Огляд існуючих архітектур штучних нейронних мереж для вирішення поставленої задачі	02.03.2026	01.04.2026	Виконано
5	Реалізація обраних технологій з аналізом отриманих результатів	02.04.2026	24.05.2026	Виконано
6	Перший попередній захист КР на засіданні комісії кафедри	25.05.2026	25.05.2026	Виконано
7	Корегування роботи за результатами попереднього захисту	26.05.2026	04.06.2026	Виконано
8	Другий попередній захист КР на засіданні комісії кафедри	05.06.2026	05.06.2026	Виконано
9	Доробка та остаточне оформлення КР	06.06.2026	14.06.2026	Виконано
10	Подання КР, її електронної копії та інших документів (відгуку, рецензії) до захисту	15.06.2026	19.06.2026	Виконано

Керівник роботи _____ Євген СІДЕНКО
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Здобувач _____ Богдан СОМРЯКОВ
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Дата складання календарного плану
«29» січня 2026 р.

АНОТАЦІЯ

до кваліфікаційної роботи
здобувача групи 401 ЧНУ ім. Петра Могили

Сомрякова Богдана Олеговича

на тему: **“ ІНТЕЛЕКТУАЛЬНА СИСТЕМА КЛАСТЕРИЗАЦІЇ ОЗНАК ЗОБРАЖЕНЬ КРОВ’ЯНИХ КЛІТИН ЗА ДОПОМОГОЮ ЗГОРТКОВИХ НЕЙРОННИХ МЕРЕЖ”**

Кваліфікаційну роботу присвячено розробці інтелектуальної системи для кластеризації ознак зображень кров’яних клітин із використанням інструментарію глибокого навчання. Актуальність теми полягає у зростаючій потребі автоматизованих систем попереднього аналізу медичних візуальних даних, зокрема кластеризації клітин крові для їх подальшої діагностичної інтерпретації.

Об’єктом кваліфікаційної роботи є процес аналізу та структурування ознак об’єктів на медичних зображеннях.

Предметом кваліфікаційної роботи є методи й програмні засоби кластеризації ознак зображень кров’яних клітин на основі згорткових нейронних мереж.

Метою кваліфікаційної роботи є створення інтелектуальної системи, що забезпечує кластеризацію ознак клітин крові шляхом автоматичного виділення їх ознак зображень за допомогою згорткових нейронних мереж та подальшого групування методами машинного навчання.

Для досягнення поставленої мети визначено такі завдання:

– дослідити сучасні підходи до аналізу та кластеризації медичних зображень;

– виконати огляд інструментів та фреймворків глибокого навчання, придатних для виділення ознак;

– провести аналіз існуючих архітектур згорткових нейронних мереж та обґрунтувати вибір моделі для екстракції ознак;

– реалізувати процес автоматичного виділення ознак зображень кров’яних клітин;

– виконати кластеризацію відібраних ознак та провести їх візуалізацію;

– здійснити тестування та оцінку роботи розробленої інтелектуальної системи.

Кваліфікаційна робота складається з фахової частини і спеціальної частини з охорони праці. Фахова частина в свою чергу поділяється на вступ, чотири розділи, висновки та переліку джерел посилання.

У вступі наведено обґрунтування актуальності теми, мету, завдання та предмет дослідження.

У першому розділі здійснено аналіз предметної області, розглянуто особливості медичних зображень клітин крові, а також наведено огляд існуючих підходів до їх обробки та аналізу.

У другому розділі описано теоретичні основи дослідження, зокрема методи екстракції ознак за допомогою глибокого навчання та алгоритми кластеризації даних.

У третьому розділі представлено реалізацію навчання моделей, підготовку та обробку даних, а також проведено аналіз отриманих результатів із використанням відповідних метрик якості.

У четвертому розділі наведено програмну реалізацію системи, включаючи архітектуру вебзастосунку, серверну частину та інтеграцію з базою даних для збереження результатів.

У висновках подано узагальнення результатів дослідження, оцінку ефективності розробленої системи та окреслено можливі напрями її подальшого розвитку.

Кваліфікаційна робота викладена на 74 сторінок, містить 4 розділи, 34 ілюстрацій, 15 таблиць, 64 джерел в переліку посилань.

Ключові слова: *Python, комп'ютерний зір, кластеризація, машинне навчання, медичні зображення, згорткові нейронні мережі, CNN, KMeans, PCA, глибоке навчання.*

ABSTRACT

to the qualification work by the student of the group 401 of Petro Mohyla Black Sea National University

Somriakov Bohdan

“ INTELLIGENT SYSTEM FOR CLUSTERING IMAGE FEATURES OF BLOOD CELLS USING CONVOLUTIONAL NEURAL NETWORKS ”

The qualification work is dedicated to the development of an intelligent system for clustering image features of blood cells using deep learning methods, particularly convolutional neural networks. The relevance of the topic lies in the growing need for automated tools capable of performing preliminary analysis of medical visual data, including clustering blood cell images for further diagnostic interpretation.

The object of the qualification work is the process of analyzing and structuring feature representations of objects in medical images.

The subject of the qualification work is the methods and software tools for clustering feature representations of blood cell images using convolutional neural networks.

The aim of the qualification work is to create an intelligent system that performs clustering of blood cell features by automatically extracting them through convolutional neural networks and grouping them using machine learning algorithms.

To achieve the stated goal, the following tasks were defined:

– to study modern approaches to the analysis and clustering of medical images;

– to review deep learning tools and frameworks suitable for feature extraction;

– to analyze existing convolutional neural network architectures and justify the choice of a model for feature extraction;

– to implement an automatic feature extraction process for blood cell images;

– to perform clustering of the extracted features and visualize the results;

– to test and evaluate the developed intelligent system.

The qualification work consists of a professional part and a special section on occupational safety. The professional part is divided into an introduction, four chapters, conclusions, and a list of references.

The introduction presents the justification of the topic relevance, the aim, tasks, and subject of the study.

The first chapter provides an analysis of the subject area, describes the characteristics of medical blood cell images, and reviews existing approaches to their processing and analysis.

The second chapter outlines the theoretical foundations of the study, including deep learning-based feature extraction methods and clustering algorithms.

The third chapter presents the implementation of model training, data preparation and processing, as well as an analysis of the obtained results using relevant evaluation metrics.

The fourth chapter describes the software implementation of the system, including the web application architecture, backend component, and database integration for result storage.

The conclusions summarize the results of the study, evaluate the effectiveness of the developed system, and outline possible directions for further improvement.

The qualification work is presented on 74 pages, contains 4 chapters, 34 figures, 15 tables, and 64 sources in the list of references.

Keywords: Python, computer vision, clustering, machine learning, medical imaging, convolutional neural networks, CNN, KMeans, PCA, deep learning.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

AHP – analytic hierarchy process
AJAX – asynchronous JavaScript And XML
API – application programming interface
CNN – convolutional neural network
CPU – central processing unit
CSS – cascading style sheets
DB – database
DL – deep learning
GPU – graphics processing unit
HTML – hypertext markup language
HTTP – hypertext transfer protocol
IIS – intelligent information system
JS – JavaScript
JSON – JavaScript object notation
ML – machine learning
ORM – object-relational mapping
PCA – principal component analysis
ReLU – rectified linear unit
REST – representational state transfer
SQL – structured query language
UI/UX – user interface user experience
XML – extensible markup language
WSGI – web server gateway interface

ВСТУП

Розробка інтелектуальної системи для аналізу зображень клітин крові є актуальною задачею сучасного штучного інтелекту та медичної інформатики, оскільки автоматизація морфологічного аналізу дозволяє суттєво підвищити швидкість і точність первинної діагностики. Клітини крові відіграють ключову роль у функціонуванні організму, а їх форма, розмір, структура ядра та співвідношення цитоплазми є важливими діагностичними ознаками, що використовуються для виявлення інфекцій, запальних процесів, анемій та онкологічних захворювань.

Разом із тим, різні типи клітин крові можуть мати високу візуальну схожість, що ускладнює їх розпізнавання навіть для фахівців-лаборантів. Додатковими ускладнюючими факторами є якість мікроскопічних зображень, шум, варіації фарбування та перекриття клітин. Саме тому використання методів глибокого навчання є особливо доцільним, оскільки вони здатні автоматично виділяти приховані ознаки, недоступні для класичних підходів аналізу зображень.

У межах реалізації система спрямована на забезпечення повного циклу обробки даних — від завантаження зображень до отримання інтерпретованих результатів кластеризації. Це дозволяє не лише розпізнавати відомі типи клітин, але й аналізувати внутрішню структуру даних, виявляючи приховані закономірності без використання міток.

Функціональні можливості системи включають:

– завантаження та адміністрування декількох датасетів одночасно, що забезпечує універсальність системи та можливість роботи з різними наборами медичних даних без зміни програмної архітектури;

– автоматичне виділення глибоких морфологічних ознак клітин із використанням згорткової нейронної мережі MobileNetV2, яка формує компактні та інформативні векторні представлення зображень;

– виконання кластеризації методом K-Means із можливістю самостійного вибору кількості кластерів користувачем, що дозволяє адаптувати аналіз під різні експериментальні сценарії;

– проєкцію багатовимірних ознак у простір зниженої розмірності за допомогою методу головних компонент (PCA) для наочної інтерпретації результатів кластеризації;

– взаємодію з користувачем через сучасний веб-інтерфейс на базі фреймворку Flask, що забезпечує зручний доступ до функціоналу системи та інтуїтивний UI/UX;

– надійне збереження завантажених датасетів та результатів обробки в інтегрованій базі даних, що забезпечує їх подальше використання та аналіз;

– підвищення рівня безпеки системи шляхом тестування на вразливості, зокрема SQL-ін'єкції та інші типові веб-загрози.

Завдяки використанню якісно підготовленого медичного датасету та сучасних методів глибокого навчання, очікується, що система здатна ефективно виявляти приховану структуру даних і формувати кластери, які відповідають різним типам клітин крові.

Це є особливо важливим, оскільки кластеризація ознак дозволяє виявляти природні групи даних без попередньої розмітки, що є критичним у медичних задачах, де анотація даних є дорогою та обмеженою. Крім того, такий підхід може сприяти виявленню нетипових або патологічних зразків як окремих кластерів, що підвищує потенціал ранньої діагностики.

1 АНАЛІЗ СФЕР ЗАСТОСУВАННЯ ШІ

1.1 Сучасний стан інструментів ШІ для кластеризаційних задач CV

Сучасні методи кластеризації у сфері штучного інтелекту активно використовують глибокі нейронні мережі, зокрема конволюційні нейронні мережі (CNN). Вони дозволяють автоматично виділяти важливі ознаки об'єктів без потреби ручного визначення характеристик. Кожен об'єкт у наборі даних може бути представлений у вигляді вектора ознак, що обчислюється за допомогою CNN, як показано у формулі (1.1).

$$f_i = \mathrm{CNN}(x_i)\ , \tag{1.1}$$

де f_i – вектор ознак і-го об'єкта;

x_i – вхідне зображення (або об'єкт) з набору даних;

$\mathrm{CNN}(\quad)$ – згорткова нейронна мережа, що виконує виділення ознак.

Отримані вектори ознак слугують основою для алгоритмів кластеризації. Один із найпоширеніших підходів – це алгоритм K-Means, де об'єкти групуються навколо центроїдів кластерів (рис. 1.1).

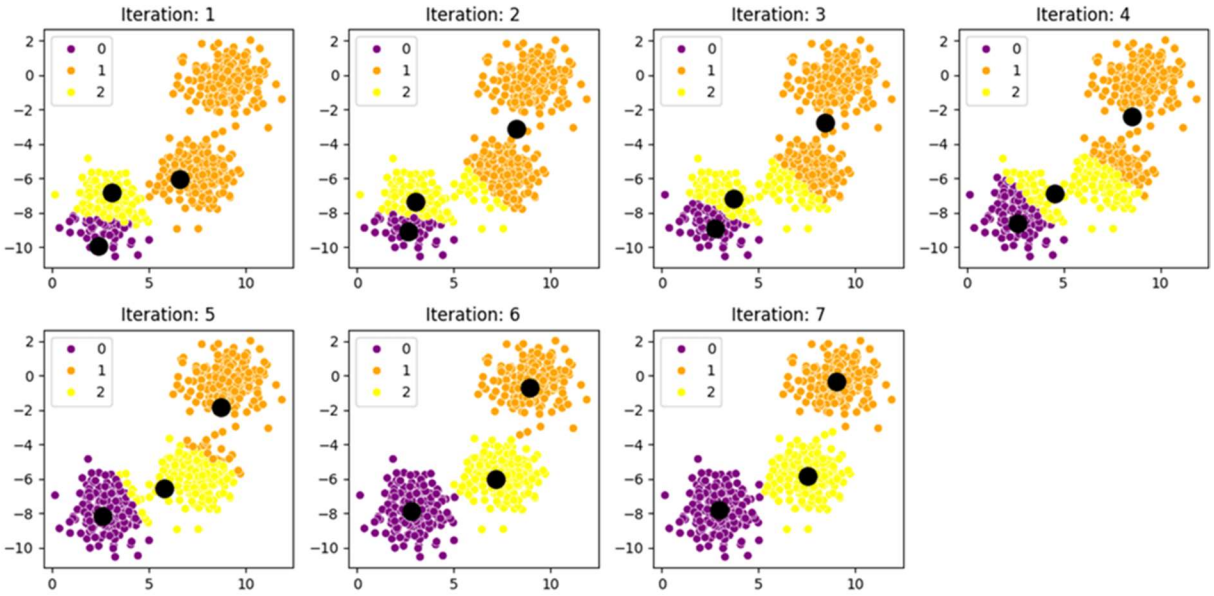


Рисунок 1.1 – Візуалізація K-Means кластеризації [1]

Центроїди розраховуються як середнє всіх векторів ознак у межах кластера (1.2), а відстань між об'єктом і центроїдом визначається як евклідова норма (1.3). Ці підходи дозволяють ефективно об'єднувати подібні об'єкти у групи.

$$\mu_k = \frac{1}{|C_k|}\sum_{x_i \in C_k} f_i \ , \tag{1.2}$$

де μ_k – центроїд k-го кластера;

C_k – множина об'єктів, що належать k-му кластеру;

$|C_k|$ – кількість об'єктів у кластері;

f_i– вектор ознак і-го об'єкта.

$$d(f_i, \mu_k) = |f_i - \mu_k|_2 \ , \tag{1.3}$$

де $d(f_i, \mu_k)$ – відстань між об'єктом та центроїдом кластера;

f_i – вектор ознак і-го об'єкта;

μ_k – центроїд k-го кластера;

$||_2$– евклідова норма (L2-відстань).

Для покращення обчислювальної ефективності та візуалізації сучасні системи часто застосовують методи зменшення розмірності, наприклад PCA (Principal Component Analysis) (рис.1.2).

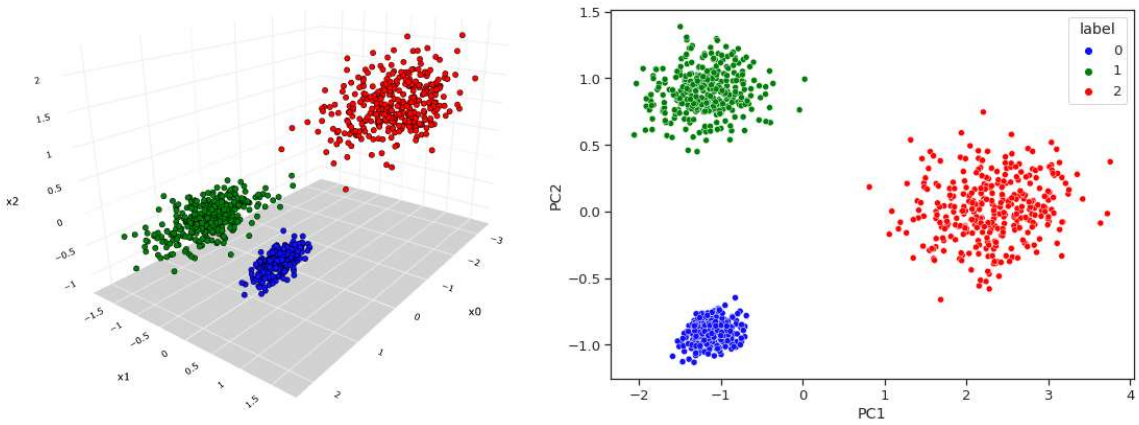


Рисунок 1.2 – Візуалізація роботи PCA [2]

PCA дозволяє проектувати багатовимірні вектори ознак у простір низької розмірності, що дозволяє оцінювати структуру кластерів у двовимірному або тривимірному просторі, як показано у формулі (1.4).

$$f_i^{(d)} = W^T f_i \,, \tag{1.4}$$

де $f_i^{(d)}$– вектор ознак і-го об'єкта у просторі зниженої розмірності d;

W – матриця проєкції (матриця головних компонент);

W^T – транспонована матриця проєкції;

f_i – вихідний багатовимірний вектор ознак і-го об'єкта.

Ще один важливий аспект сучасних методів кластеризації – це попередня обробка даних, яка включає нормалізацію пікселів і аугментацію даних. Нормалізація забезпечує стабільність навчання мереж і підвищує якість кластеризації (1.5), тоді як аугментація допомагає зробити моделі стійкими до різних трансформацій об'єктів, таких як обертання або масштабування, що представлено у формулі (1.6).

$$x_i' = \frac{x_i - \mu}{\sigma} \,, \tag{1.5}$$

де x_i'– нормалізоване значення ознаки і-го об'єкта;

x_i – початкове значення ознаки;

μ – середнє значення вибірки;

σ – стандартне відхилення вибірки.

$$x_i^{aug} = T(x_i'), \quad T \in \{\text{rotation, scaling, translation, flip}\} \,, \tag{1.6}$$

де x_i^{aug}– аугментований (перетворений) об'єкт;

x_i' – нормалізований вхідний об'єкт;

$T(\quad)$ – оператор аугментації;

T – множина геометричних перетворень (обертання, масштабування, зсув, віддзеркалення).

Сучасні інструменти для кластеризації також передбачають оптимізацію центроїдів шляхом мінімізації внутрі-кластерної дисперсії (1.7). Після проведення кластеризації важливо оцінити якість отриманих груп, для чого використовують різні метрики, зокрема силует-коефіцієнт (1.8) або інші індекси, які чисельно оцінюють щільність та відокремленість кластерів.

$$J = \sum_{k=1}^{K} \sum_{x_i \in C_k} | f_i - \mu_k |_2^2 \,, \tag{1.7}$$

де J – цільова функція кластеризації (внутрішньокластерна дисперсія)

K – кількість кластерів;

C_k – множина об'єктів k-го кластера;

f_i – вектор ознак i-го об'єкта;

μ_k – центроїд k-го кластера;

$||$ – евклідова норма.

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))} \,, \tag{1.8}$$

де $s(i)$ – силует-коефіцієнт і-го об'єкта;

$a(i)$ – середня відстань від і-го об'єкта до інших об'єктів у його кластері;

$b(i)$ – мінімальна середня відстань від і-го об'єкта до об'єктів іншого найближчого кластера.

Сучасні фреймворки для ШІ, такі як TensorFlow, PyTorch або scikit-learn, дозволяють об'єднувати CNN для виділення ознак з класичними алгоритмами кластеризації. Завдяки можливості обробки пакетів даних та багатопоточного виконання можна ефективно працювати з великими обсягами даних, що представлено у формулі (1.9).

$$F = \{f_1, f_2, \dots, f_n\}, \quad L = \{c_1, c_2, \dots, c_n\} \,, \tag{1.9}$$

де F – множина векторів ознак об'єктів;

f_i – вектор ознак і-го об'єкта;

L – множина міток (кластерів або класів);

c_i– мітка (кластер) і-го об'єкта;

n – кількість об'єктів у наборі даних.

Нині також активно розвиваються гібридні підходи, які поєднують CNN із автоенкодерами або графовими нейронними мережами для створення більш щільного та інформативного латентного простору. Це дозволяє досягати кращої структури кластерів та підвищує точність групування (1.10).

$$L_{total} = L_{cluster} + \alpha L_{reconstruction}\ , \tag{1.10}$$

де L_{total} – загальна функція втрат моделі;

$L_{cluster}$ – втрата кластеризації (помилка групування об'єктів);

$L_{reconstruction}$ – втрата реконструкції вхідних даних (автоенкодер);

α – ваговий коефіцієнт, що визначає внесок реконструкційної складової.

Згорткові нейронні мережі виконують ключову операцію згортки для виділення локальних ознак зображення (рис. 1.3).

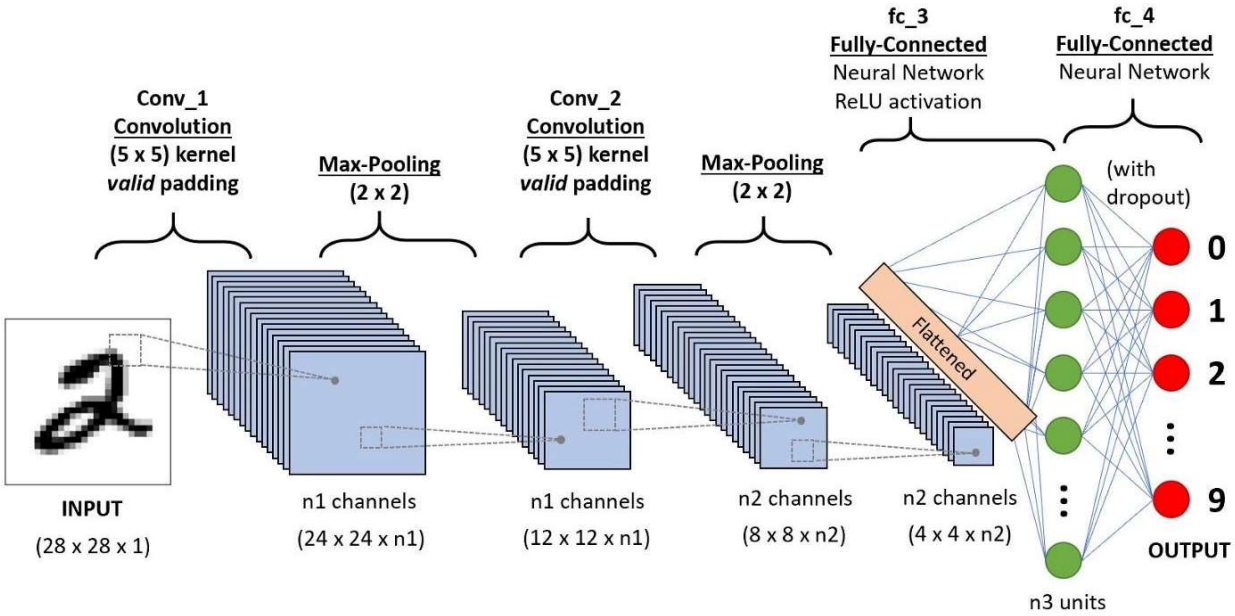


Рисунок 1.3 – Схематичне зображення згорткової нейромережі [3]

У кожному згортковому шарі формується карта ознак, де значення кожного елемента обчислюється як зважена сума елементів вхідного зображення з використанням ядра згортки та додаванням зміщення, як показано у формулі (11). Це дозволяє мережі виявляти базові патерни, такі як контури та текстури.

$$h_{i,j}^{(l)} = \sum_{m,n} W_{m,n}^{(l)} \cdot x_{i+m,j+n}^{(l-1)} + b^{(l)} , \quad (1.11)$$

де $h_{i,j}^{(l)}$ – значення карти ознак у позиції (i,j) на l-му шарі;

$x_{i+m,j+n}^{(l-1)}$ – значення вхідної карти ознак з попереднього шару;

$W_{m,n}^{(l)}$ – ваги ядра згортки (фільтра) на l-му шарі;

$b^{(l)}$ – зміщення (bias) для l-го шару;

(m, n) – зміщення (bias) для l-го шару;

Після виконання згортки застосовується нелінійна функція активації, яка вводить нелінійність у модель і дозволяє їй апроксимувати складні залежності у даних. Однією з найпоширеніших функцій є ReLU, яка обнуляє від'ємні значення та зберігає додатні, що формалізується у формулі (12).

$$a_{i,j}^{(l)} = \max\left(0, h_{i,j}^{(l)}\right), \quad (1.12)$$

де $a_{i,j}^{(l)}$ – активоване значення нейрона у позиції (i,j) на l-му шарі;

$h_{i,j}^{(l)}$ – значення після згорткової операції у позиції (i,j);

$\max(0, value)$ – функція активації ReLU, що зануляє від'ємні значення.

Для зменшення розмірності ознак і виділення найбільш значущих характеристик застосовуються операції підвибірки, зокрема max-pooling. Цей процес дозволяє зменшити розмірність даних і підвищити стійкість моделі до невеликих змін у вхідних даних, що описано у формулі (1.13).

$$p_{i,j}^{(l)} = \max_{(m,n)\in R} a_{i+m,j+n}^{(l)} , \quad (1.13)$$

де $p_{i,j}^{(l)}$ – значення після операції підвибірки (pooling) у позиції (i,j) на l-му шарі;

$a_{i+m,j+n}^{(l)}$ – значення активацій у відповідній локальній області;

R – область (вікно) підвибірки (pooling window), над якою виконується операція max-pooling.

Після проходження через декілька згорткових і пулінгових шарів формується узагальнений вектор ознак, який компактно представляє вхідне зображення. Часто для цього використовується глобальний середній пулінг, який агрегує інформацію з усієї карти ознак, як показано у формулі (1.14).

$$f_i = \frac{1}{N}\sum_{k=1}^{N} a_k \,, \tag{1.14}$$

де f_i – узагальнений вектор ознак і-го об'єкта;

a_k – значення k-го елемента карти ознак після згорткових/пулінгових шарів;

N – загальна кількість елементів у карті ознак.

Для забезпечення стабільності та коректності подальших обчислень вектор ознак може бути додатково нормалізований. Це дозволяє уникнути впливу масштабу значень і покращує якість кластеризації, що формалізується у формулі (1.15).

$$\widehat{f_i} = \frac{f_i}{|f_i|}\,, \tag{1.15}$$

де $\widehat{f_i}$– нормалізований вектор ознак і-го об'єкта;

f_i – вихідний вектор ознак;

$|f_i|$ – норма вектора (зазвичай L2-норма), що використовується для масштабування.

Серед сучасних архітектур згорткових нейронних мереж важливе місце займає MobileNet, яка оптимізована для ефективної роботи при обмежених обчислювальних ресурсах. Її ключовою особливістю є використання глибинно-

роздільних згорток, що дозволяє розділити процес виділення просторових та міжканальних ознак, як показано у формулі (1.16).

$$y_{i,j,k} = \sum_{m,n} W_{m,n,k} \cdot x_{i+m,j+n,k} \ , \tag{1.16}$$

де $y_{i,j,k}$– значення вихідної карти ознак у позиції (i,j) для k-го каналу;

$x_{i+m,j+n,k}$ – значення вхідного зображення у позиції (i+m,j+n) для k-го каналу;

$W_{m,n,k}$ – ваги згорткового ядра для просторових координат (m,n) у k-му каналі.

Додатково архітектура включає інверсні залишкові блоки та вузькі шари, що забезпечують збереження інформації при зменшенні розмірності простору ознак, що формалізується у формулі (1.17).

$$z = W_{1\times1} \cdot y \ , \tag{1.17}$$

де z – вихідний вектор ознак після застосування 1×1 згортки;

$W_{1\times1}$ – вагова матриця 1×1 згорткового шару;

y – вхідний вектор ознак.

Завдяки цьому MobileNet дозволяє отримувати якісні вектори ознак для подальшої кластеризації при значно меншій кількості параметрів і обчислень (рис. 1.4).

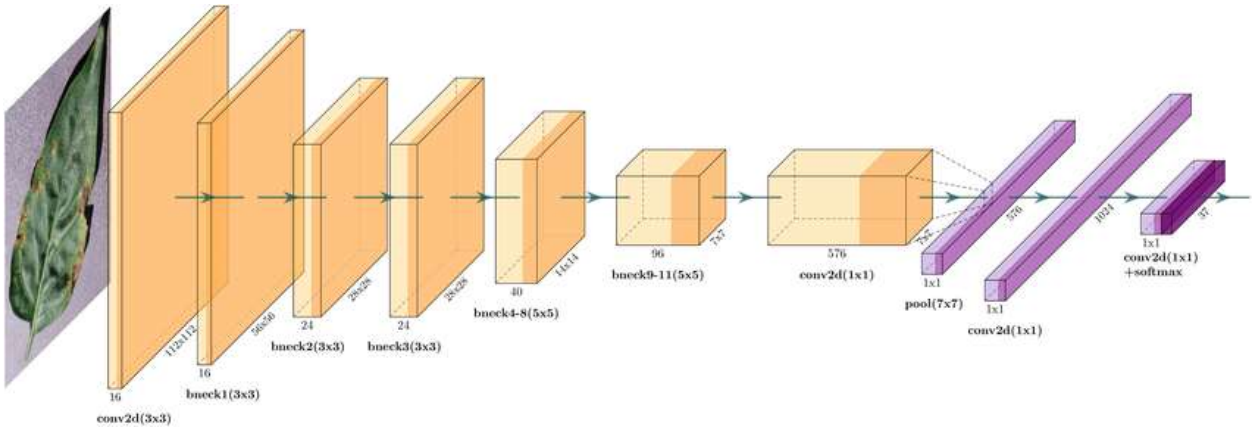


Рисунок 1.4 – Архітектура MobileNet-V3 Small [4]

MobileNet також відзначається своєю універсальністю та можливістю адаптації під різні обчислювальні ресурси. Завдяки наявності параметра ширини (width multiplier), який регулює кількість каналів у кожному шарі, та параметра роздільної здатності вхідного зображення, модель може масштабуватися шляхом зменшення або збільшення своєї складності залежно від вимог задачі та доступної обчислювальної потужності. Це дозволяє ефективно балансувати між точністю моделі та швидкістю обчислень, що є особливо важливим для застосувань у реальному часі.

Завдяки такій архітектурній гнучкості MobileNet може використовуватися як у високопродуктивних серверних системах, так і на мобільних пристроях або вбудованих платформах з обмеженими ресурсами, таких як IoT-пристрої чи системи edge computing. Окрему роль у цьому відіграє використання глибинно-роздільних згорток, які суттєво зменшують кількість параметрів моделі та обчислювальну складність без значної втрати якості представлення ознак.

Ще однією важливою особливістю MobileNet є її здатність до трансферного навчання. Попередньо навчена на великих і різноманітних наборах даних, таких як ImageNet, модель може бути адаптована до нових задач шляхом донавчання лише останніх шарів або використання її як фіксованого екстрактора ознак.

Застосування трансферного навчання дозволяє суттєво скоротити час навчання моделі, зменшити вимоги до обчислювальних ресурсів та водночас підвищити якість отриманих результатів за рахунок використання вже вивчених узагальнених ознак. Таким чином, MobileNet є ефективним компромісом між продуктивністю та ресурсозатратністю.

Узагальнюючи, послідовне застосування операцій згортки, функцій активації, підвибірки та нормалізації в межах CNN дозволяє ефективно формувати багаторівневі ознакові представлення. Такі представлення є стійкими до шуму та варіацій у вхідних даних і можуть бути успішно використані як основа для подальших задач кластеризації, класифікації та пошуку схожих об'єктів у великих масивах даних.

1.2 Огляд та аналіз публікацій і ІІS, присвячених засобам кластеризації ознак медичних зображень

Аналіз наявних наукових публікацій показує, що кластеризація ознак медичних зображень є активною областю досліджень, де застосовуються як класичні алгоритми, так і сучасні глибокі методи на базі нейронних мереж. Наприклад, у роботі "MedTransCluster: Transfer learning for deep medical image clustering" пропонується підхід до кластеризації медичних зображень грудної порожнини з використанням трансферного навчання та попередньо навчених CNN-моделей, що дозволяє значно покращити якість групування у порівнянні з класичними методами [5].

У літературі також існують роботи, що розробляють гібридні моделі глибокого навчання, комбінуючи алгоритми кластеризації з класичними архітектурами CNN для медичного аналізу без міток. Наприклад, дослідження "DeepMCAT: Large-Scale Deep Clustering for Medical Image Categorization" присвячено автоматичній кластеризації великих наборів медичних зображень МРТ без використання міток, що демонструє потенціал безнаглядного навчання у медичній практиці [6].

Інший напрямок пов'язаний із застосуванням глибоких словникових та розріджених моделей у кластеризації медичних зображень. У статті "A deep dictionary clustering approach for unsupervised image retrieval using convolutional sparse coding" запропоновано фреймворк, де ознаки витягуються глибокими CNN, а потім кластеризуються за допомогою багаторівневої словникової кластеризації для медичного пошуку зображень [7].

Окремо варто виділити підхід, що поєднує кластеризацію з напів-керованим навчанням у медичній класифікації (рис. 1.5). Стаття "Semi-Supervised Medical Image Classification Combined with Unsupervised Deep Clustering" демонструє, як кластеризація може бути інтегрована у класифікаційні моделі для покращення розпізнавання патологій, використовуючи як марковані, так і немарковані дані [8].

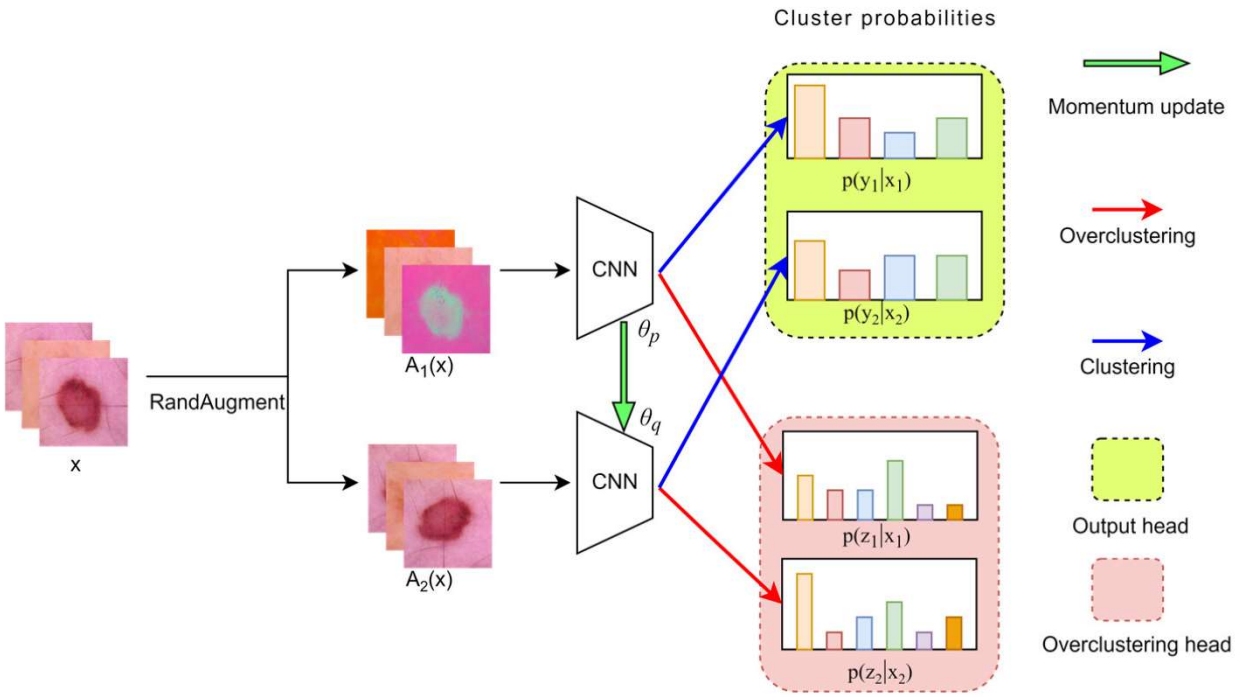


Рисунок 1.5 – Безнаглядна глибока кластеризація: ознаки зображень кластеризуються K-Means, over-clustering допомагає навчанні, для тесту використовується лише головна мережа [8]

Крім того, існують узагальнюючі огляди, що аналізують стан досліджень у глибокій кластеризації зображень. Наприклад, у "A comprehensive survey of image clustering based on deep learning" подано систематичний огляд сучасних методів глибокого кластерного навчання, включно із медичними застосуваннями, що дає змогу виявити ключові тенденції, сильні сторони та обмеження існуючих підходів [9].

Аналіз публікацій показує, що сучасні методи кластеризації медичних зображень все частіше інтегрують CNN або інші моделі глибокого навчання як потужні засоби автоматичного витягнення ознак. Комбінація таких представлень із алгоритмами кластеризації дозволяє більш точно і надійно виділяти групи подібних зображень, що є критично важливим для медичної діагностики та аналізу великих архівів даних.

У табл 1.1 наведено порівняльний аналіз сучасних наукових публікацій та підходів до кластеризації ознак медичних зображень, зокрема із застосуванням методів глибокого навчання та гібридних моделей.

Таблиця 1.1 – Порівняння публікацій та розроблених IIS, присвячених засобам кластеризації ознак зображень, зокрема в медичній сфері

Джерело	**Підхід**	**Основна ідея**	**Використані методи**	**Сильні сторони**	**Обмеження**
MedTransCluster [5]	Transfer learning + CNN	Кластеризація медичних зображень грудної порожнини з використанням попередньо навчених моделей	CNN, transfer learning, clustering	Висока точність кластеризації, покращення якості за рахунок pre-trained моделей	Залежність від якості попереднього навчання
DeepMCAT [6]	Deep clustering (unsupervised)	Автоматична кластеризація великих наборів медичних зображень без міток	Deep CNN, unsupervised clustering	Робота без розмічених даних, масштабованість	Високі обчислювальні витрати
Sparse Deep Dictionary Clustering [7]	Sparse coding + CNN features	Використання глибоких ознак та словникової кластеризації	CNN feature extraction, sparse coding, dictionary learning	Добра якість витягу ознак, ефективність для retrieval задач	Складність моделі
Semi-Supervised Clustering [8]	Hybrid (semi-supervised + clustering)	Поєднання кластеризації з частково розміченими даними	CNN, clustering, semi-supervised learning	Покращення точності за рахунок часткових міток	Потребує хоча б частково розмічених даних
Survey paper [9]	Огляд методів deep clustering	Систематизація сучасних методів глибокої кластеризації	CNN-based clustering, deep learning methods	Узагальнення підходів, аналіз тенденцій	Не пропонує нового алгоритму

Таблиця 1.1 демонструє, що сучасні дослідження переважно базуються на використанні згорткових нейронних мереж (CNN) у поєднанні з методами transfer learning, unsupervised та semi-supervised навчання. Це дозволяє підвищити якість кластеризації та забезпечити ефективну обробку великих обсягів медичних даних навіть за відсутності повної розмітки.

Аналіз таблиці 1.1 показує, що найбільш перспективними є гібридні підходи, які поєднують глибоке навчання з класичними алгоритмами кластеризації, оскільки вони забезпечують баланс між точністю, масштабованістю та здатністю працювати з неструктурованими медичними даними.

1.3 Постановка задачі

Отже, задачою є кластеризація великого набору зображень кров'яних клітин з метою групування їх за схожими ознаками без попередньої розмітки. Для цього планується використовувати глибоку нейронну мережу MobileNetV2 як засіб автоматичного виділення ознак із зображень, що дозволить уникнути ручного проєктування характеристик.

Ознаки, отримані з MobileNetV2, будуть подаватись на вхід алгоритму кластеризації без нагляду, зокрема K-Means, для розбиття набору даних на групи за подібністю.

У межах цієї роботи буде використаний публічний датасет Blood Cells Image Dataset (доступний на Kaggle) [10], який містить загалом 17 092 зображення окремих нормальних клітин, отримані за допомогою аналізатора CellaVision DM96 у Клінічній лабораторії при Hospital Clinic of Barcelona (рис. 1.5).

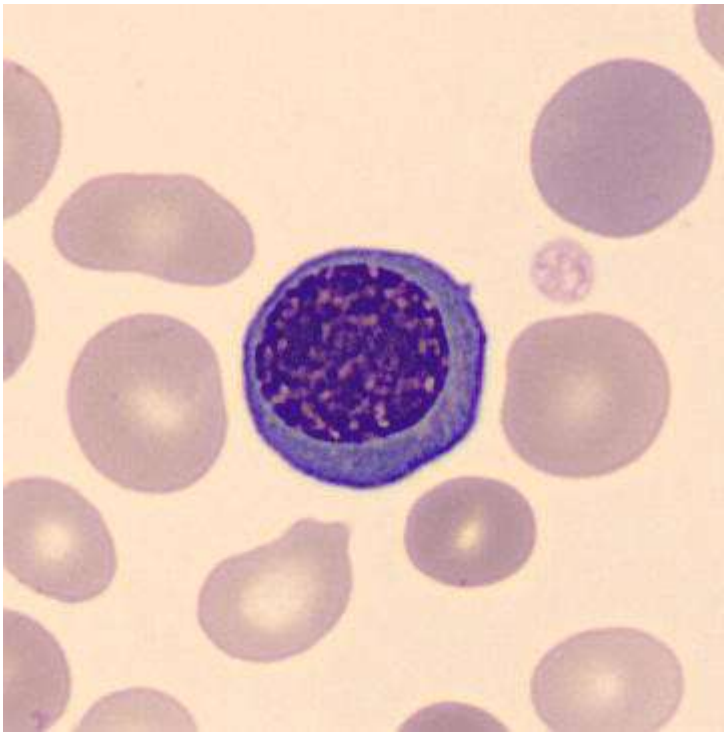


Рисунок 1.5 – Фрагмент датасету (клітина еритробласт)

Датасет містить вісім груп клітин, кожна з яких має важливу функцію. Нейтрофіли борються з бактеріями та вірусами, еозинофіли – з паразитами та беруть участь в алергіях, базофіли – у запальних процесах, лімфоцити – забезпечують специфічний імунітет, моноцити перетворюються на макрофаги для поглинання шкідливих частинок, незрілі гранулоцити – попередники дозрілих гранулоцитів, еритробласти – ранні форми червоних кров'яних клітин, а тромбоцити відповідають за згортання крові та загоєння ран.

Зображення мають розмір 360×363 пікселів у форматі JPG і були анотовані експертами-клінічними патологами. Всі зображення отримані від осіб без інфекційних, гематологічних або онкологічних захворювань і без будь-якого фармакологічного лікування на момент забору крові (рис. 1.6).

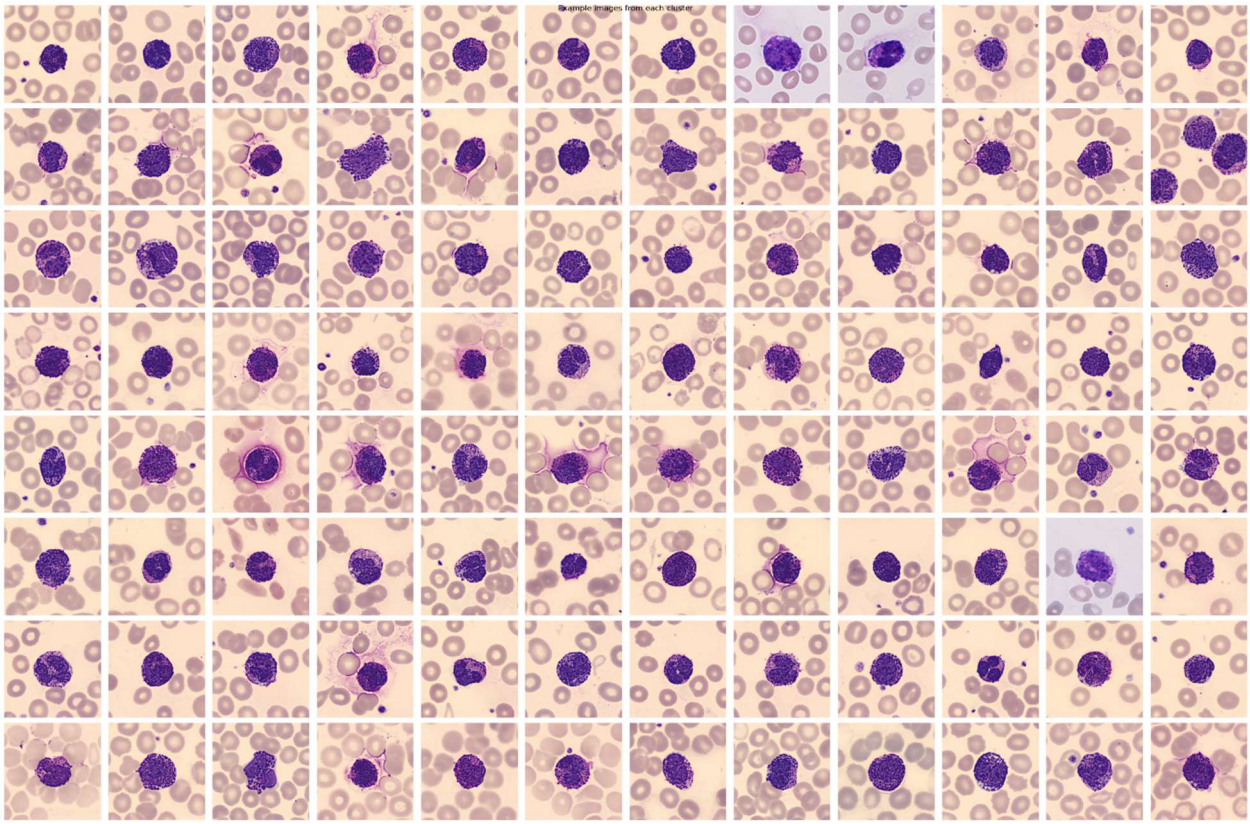


Рисунок 1.6 – Превью датасету кров'яних клітин

Для подальшого аналізу структури ознак буде проведено зменшення розмірності за допомогою PCA, що дозволить відобразити розподіл кластерів у двовимірному просторі та оцінити якість виділених груп.

Результати кластеризації планується відображати у вигляді інтерактивних графіків та групованих наборів зображень, що дозволить візуально оцінити подібність між клітинами у кожному кластері.

В рамках реалізації система буде мати можливість:

– завантажувати та адмініструвати декілька датасетів одночасно, забезпечуючи універсальність для різних типів медичних даних;

– автоматично виділяти глибокі морфологічні ознаки клітин за допомогою нейронної мережі MobileNetV2;

– виконувати кластеризацію методом K-Means з можливістю самостійного вибору кількості груп користувачем;

– проводити проєкцію ознак у простір низької розмірності через алгоритм PCA для наочної візуалізації результатів;

– взаємодіяти з користувачем через сучасний веб-інтерфейс на базі Flask, що забезпечує зручний UI/UX для дослідника;

– надійно зберігати датасети інтегрованій базі даних;

– гарантувати стабільність та захищеність системи завдяки тестуванню на вразливості до SQL-ін'єкцій та інших атак.

Завдяки використанню якісно анотованого медичного датасету та сучасних методів глибокого навчання, очікується, що система зможе ефективно розгортати структуру ознак у кластери, що відповідають різним типам клітин, навіть у випадках зображень поганої якості або з обмеженою контрастністю.

Висновки до розділу 1

У першому розділі розглянуто сучасні підходи до кластеризації зображень із використанням методів глибокого навчання. Основну увагу приділено поєднанню згорткових нейронних мереж для автоматичного виділення ознак та алгоритму K-Means для безнаглядного групування даних.

Проаналізовано принципи роботи CNN, формування векторів ознак та методи зменшення розмірності, зокрема PCA, що дозволяє візуалізувати структуру кластерів. Розглянуто сучасні архітектури, зокрема MobileNet, як ефективний інструмент для швидкої та ресурсоефективної екстракції ознак.

Також описано етапи попередньої обробки даних, включаючи нормалізацію та аугментацію, які підвищують якість кластеризації. Наведено математичні основи алгоритму K-Means та підходи до оцінювання якості кластерів.

Окремо проаналізовано наукові публікації у сфері кластеризації медичних зображень, що підтверджує актуальність використання безнаглядних методів навчання для роботи з великими обсягами даних. Визначено, що сучасні підходи часто поєднують CNN з іншими моделями глибокого навчання для підвищення точності кластеризації.

Описано обраний датасет кров'яних клітин та його особливості, а також сформовано вимоги до майбутньої системи, включаючи використання бази даних, реалізацію веб- та десктопного застосунків і інтерактивну візуалізацію результатів.

Таким чином, сформовано теоретичну основу та визначено ключові технології для подальшої реалізації системи кластеризації медичних зображень.

2 ТЕХНОЛОГІЇ ВИРІШЕННЯ ПОСТАВЛЕНОЇ ЗАДАЧІ

2.1 Мова програмування Python

Python – це високорівнева інтерпретована мова програмування загального призначення, яка відзначається простотою синтаксису, зрозумілістю коду та великою гнучкістю у застосуванні. Мова була створена в початку 1990-х років і з того часу набула широкого поширення у різних сферах розробки програмного забезпечення та науки. Основна ідея Python полягає у забезпеченні максимальної читабельності та мінімальної складності запису алгоритмів, що робить її доступною для початківців і водночас потужною для професійних програмістів.

Однією з ключових переваг Python є його багата екосистема готових бібліотек і модулів, що значно скорочує час розробки та дозволяє використовувати вже реалізовані алгоритми без необхідності писати їх з нуля. Мова підтримує кілька парадигм програмування, зокрема об'єктно-орієнтоване, процедурне та функціональне програмування, що забезпечує гнучкість у побудові різноманітних програмних систем. Завдяки інтерпретованості Python дозволяє відразу запускати та тестувати код, що сприяє швидкому прототипуванню та експериментуванню з алгоритмами.

Python є кросплатформеною мовою, що дозволяє запускати програми на різних операційних системах без істотних змін у коді. Крім того, активна спільнота розробників забезпечує постійну підтримку, оновлення бібліотек і численні приклади реалізації практично будь-яких алгоритмів. Простота синтаксису, зрозумілість конструкцій та наявність великої кількості документації роблять Python однією з найпопулярніших мов програмування у світі і дозволяють ефективно вирішувати широкий спектр задач у науці, освіті та промисловості.

2.2 Бібліотека Tensorflow

TensorFlow – це потужна бібліотека для машинного навчання та глибокого навчання, розроблена компанією Google. Вона призначена для створення, навчання

та застосування моделей штучних нейронних мереж різного рівня складності, починаючи від простих моделей регресії до складних глибинних мереж. TensorFlow підтримує як високорівневий, так і низькорівневий підхід до побудови моделей.

Основною особливістю TensorFlow є використання графів обчислень (рис. 2.1), що дозволяє ефективно виконувати складні математичні операції над багатовимірними масивами даних. Це забезпечує високий рівень оптимізації та продуктивності, а також підтримку паралельних обчислень на центральному процесорі (CPU) та графічному процесорі (GPU), що особливо важливо при роботі з великими наборами даних і складними нейронними мережами.

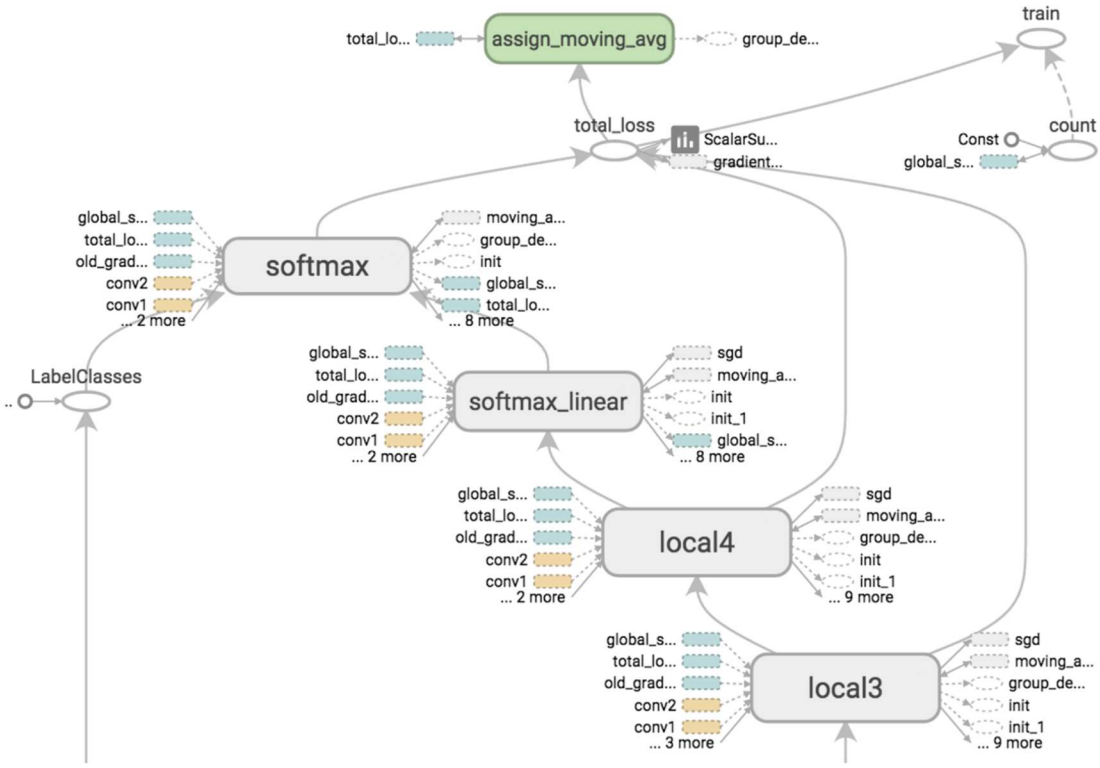


Рисунок 2.1 – Граф обчислень [11]

TensorFlow надає широкий спектр інструментів і модулів для різних завдань машинного навчання, включаючи обробку зображень, обробку природної мови,

розпізнавання голосу та роботу з часовими рядами. Бібліотека підтримує роботу з високорівневим API Keras, що дозволяє швидко створювати та навчати нейронні мережі без необхідності вручну визначати всі параметри та шари моделі. Крім того, TensorFlow забезпечує можливість збереження і завантаження навчальних моделей, що робить її зручною для використання у виробничих проєктах.

Однією з переваг TensorFlow є його кросплатформеність та інтеграція з різними мовами програмування, перш за все з Python, що робить бібліотеку доступною як для науковців, так і для розробників програмного забезпечення. Активна спільнота користувачів та велика кількість документації дозволяють швидко опанувати основи роботи з бібліотекою і ефективно застосовувати її для різних завдань машинного навчання.

Завдяки своїй потужності, гнучкості та можливостям масштабування TensorFlow став однією з найпопулярніших бібліотек у світі штучного інтелекту та глибокого навчання, забезпечуючи створення складних моделей і автоматизацію аналізу великих даних у різних сферах діяльності.

2.3 Бібліотека Keras

Keras – це високорівнева бібліотека для створення та навчання нейронних мереж, яка працює поверх TensorFlow і значно спрощує процес розробки моделей глибокого навчання. Вона орієнтована на швидке прототипування та зручність використання, дозволяючи будувати складні архітектури нейронних мереж за допомогою зрозумілого та компактного коду.

Основною перевагою Keras є її модульність і простота. Користувач може легко створювати моделі, додаючи шари послідовно або використовуючи більш складні підходи, такі як функціональний API. Це дає змогу реалізовувати як прості нейронні мережі, так і складні архітектури, зокрема згорткові нейронні мережі для обробки зображень.

Keras надає вбудовані інструменти для роботи з шарами, функціями активації, оптимізаторами та функціями втрат, що дозволяє гнучко налаштовувати

процес навчання моделей. Крім того, бібліотека підтримує використання попередньо навчених моделей, таких як MobileNetV2, що дає можливість застосовувати трансферне навчання для ефективного виділення ознак із зображень без необхідності навчати модель з нуля.

Ще однією важливою особливістю є можливість легкої інтеграції з іншими компонентами TensorFlow, що забезпечує доступ до обчислювальних ресурсів, таких як CPU та GPU. Це дозволяє значно прискорити процес навчання та обробки великих обсягів даних.

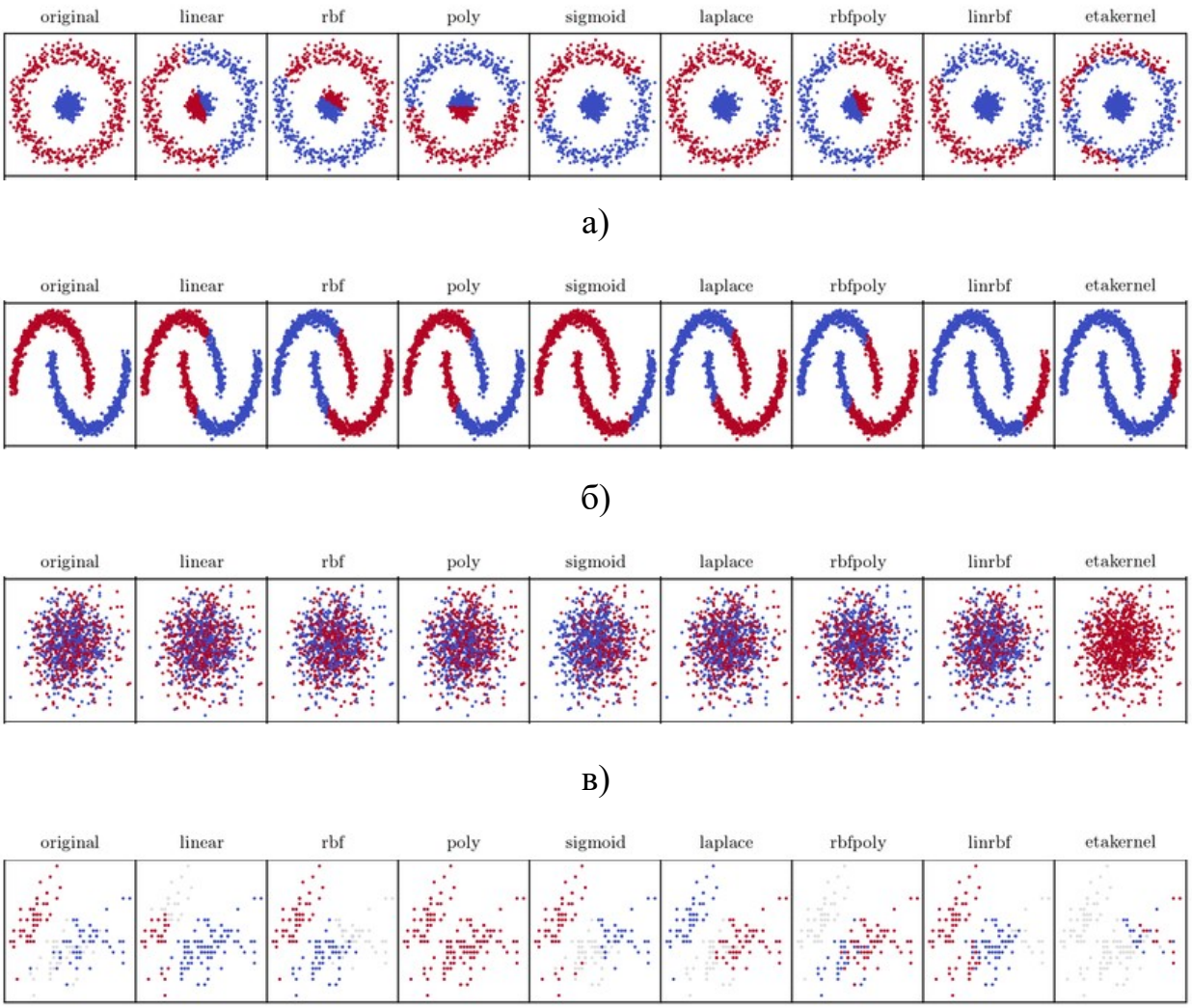
Таким чином, Keras є зручним інструментом для розробки моделей глибокого навчання, який поєднує простоту використання з широкими можливостями, що робить його ефективним рішенням для задач обробки та кластеризації зображень.

2.4 Бібліотека Sklearn

Scikit-learn (sklearn) – це бібліотека для машинного навчання мовою Python, яка надає набір інструментів для аналізу даних, побудови моделей та оцінки їх ефективності. Вона орієнтована на реалізацію класичних алгоритмів машинного навчання та широко використовується як у наукових дослідженнях, так і в прикладних задачах.

Основною особливістю Scikit-learn є її уніфікований інтерфейс, який дозволяє однаково працювати з різними моделями. Бібліотека реалізує такі методи, як класифікація, регресія, кластеризація та зменшення розмірності даних. Зокрема, для задач кластеризації вона містить алгоритми, такі як K-Means, що дозволяє групувати об'єкти за схожістю ознак (рис. 2.2), а також методи на кшталт PCA для проєкції даних у простір меншої розмірності.

Scikit-learn тісно інтегрується з бібліотеками NumPy та SciPy, що забезпечує ефективну обробку числових даних та виконання математичних операцій. Крім того, вона надає засоби для попередньої обробки даних, включаючи нормалізацію, масштабування та підготовку вибірок, що є важливими етапами перед навчанням моделей.



а)

б)

в)

г)

Рисунок 2.2 – Використання K-Means кластерезації в sklearn для: а) кола, б) місяців, в) – classification, г) – iris [12]

Ще однією важливою перевагою є наявність інструментів для оцінювання моделей, таких як різні метрики якості, що дозволяють аналізувати результати кластеризації або класифікації. Завдяки простоті використання, зрозумілій документації та широкому набору алгоритмів, Scikit-learn є ефективним інструментом для реалізації задач аналізу даних і машинного навчання, зокрема для кластеризації ознак зображень.

2.5 Бібліотека Matplotlib

Matplotlib – це потужна бібліотека для створення 2D-графіків та візуалізації даних у мові програмування Python. Вона дозволяє будувати лінійні графіки, гістограми, діаграми розсіювання, стовпчикові та кругові діаграми, а також більш складні візуалізації, комбінуючи різні типи графіків у одному малюнку. Основною її перевагою є гнучкість і можливість тонкого налаштування практично будь-якого елемента графіка: осей, кольорів, маркерів, легенд, підписів та масштабів.

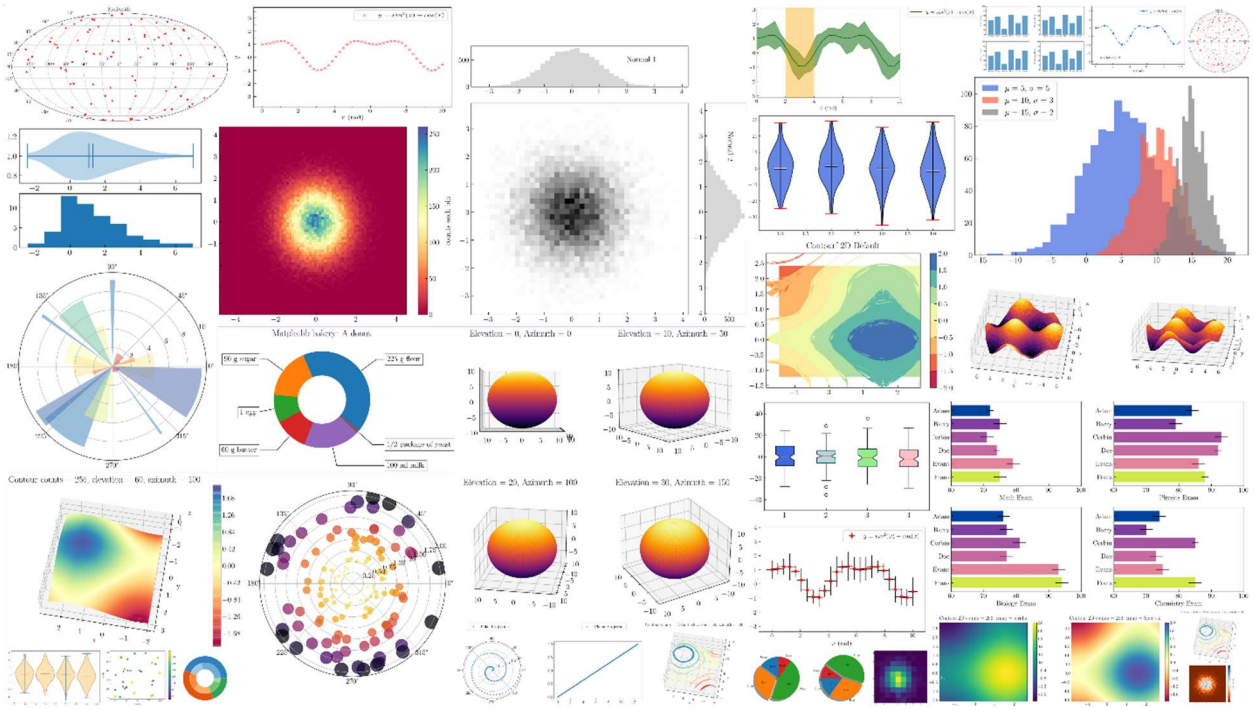


Рисунок 2.3 – Приклад графіків з Matplotlib

Matplotlib широко використовується у наукових дослідженнях, обробці зображень, аналізі часових рядів та машинному навчанні для візуалізації результатів. Часто її застосовують разом із бібліотеками NumPy і Pandas для обробки та подання даних, а також із TensorFlow та Scikit-learn для демонстрації результатів кластеризації, навчання моделей та аналізу помилок.

Бібліотека забезпечує декілька рівнів роботи: можна використовувати високорівневий API Pyplot, який імітує функціональність MATLAB і дозволяє

швидко будувати графіки, або низькорівневий об'єктно-орієнтований підхід, що дає більший контроль над створенням складних композицій графіків. Завдяки цьому Matplotlib є універсальним інструментом для будь-якого виду візуалізації даних у Python.

Як приклад, за допомогою Matplotlib можна будувати PCA-проекції кластерів, гістограми розподілу ознак, графіки зміни втрат під час навчання моделей або порівнювати кілька кластерів на двовимірних проекціях для оцінки якості кластеризації.

2.6 Бібліотека Numpy

NumPy – це базова бібліотека для наукових обчислень у мові програмування Python, яка забезпечує ефективну роботу з багатовимірними масивами даних і математичними операціями над ними. Вона є фундаментом для більшості інших бібліотек машинного навчання та аналізу даних, таких як Scikit-learn, TensorFlow і Matplotlib.

Основним елементом NumPy є ndarray – багатовимірний масив, що дозволяє зберігати великі обсяги числових даних і виконувати над ними операції значно швидше, ніж стандартні структури Python (рис. 2.4). Це досягається завдяки використанню оптимізованих реалізацій на мові C та підтримці векторизованих обчислень.

Завдяки механізму broadcasting (трансляції), NumPy дозволяє виконувати арифметичні операції над масивами різного розміру, що значно спрощує код та уникає використання повільних циклів for.

NumPy надає широкий набір функцій для виконання математичних операцій, включаючи лінійну алгебру, статистичні обчислення, генерацію випадкових чисел та роботу з матрицями. Важливою перевагою є можливість виконання операцій над цілими масивами без використання циклів, що значно підвищує продуктивність обробки даних.

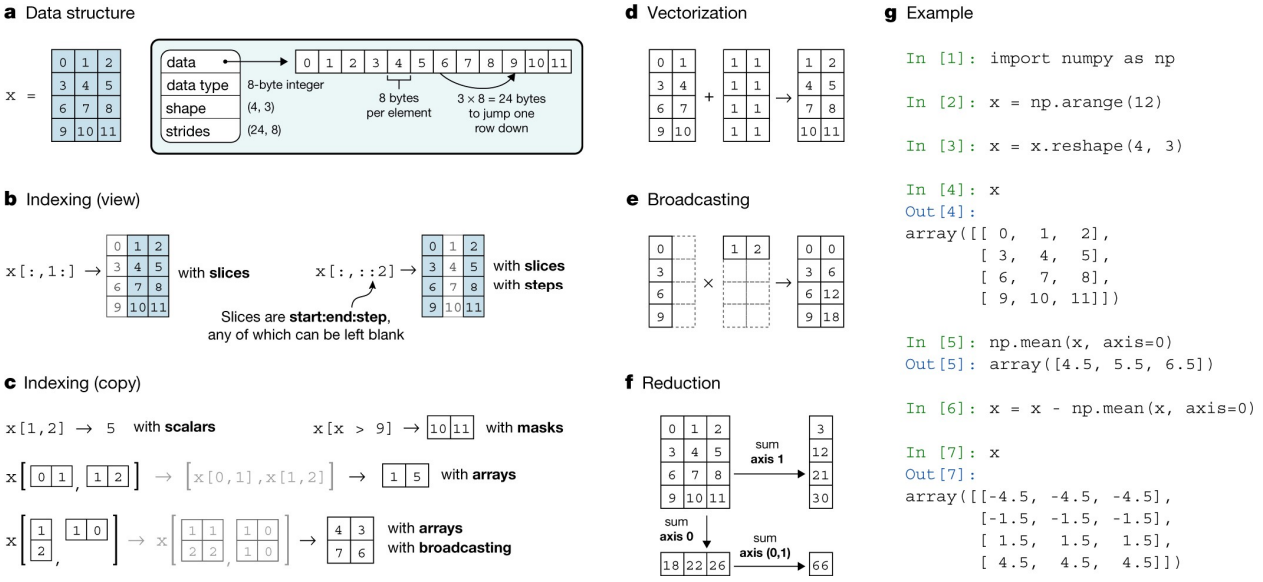


Рисунок 2.4 – Осноні концепції масивів в NumPy [14]

У задачах машинного навчання та кластеризації NumPy використовується для представлення ознак у вигляді векторів, обчислення відстаней між об'єктами, нормалізації даних та підготовки їх до подальшої обробки. У поєднанні з іншими бібліотеками NumPy забезпечує швидку та ефективну реалізацію алгоритмів аналізу даних.

Таким чином, NumPy є ключовим інструментом для роботи з числовими даними, який забезпечує основу для реалізації алгоритмів машинного навчання та обробки зображень.

2.7 Фреймворк Flask

Flask — це легкий веб-фреймворк для мови програмування Python, який використовується для створення веб-застосунків і REST API. Він побудований за принципом мінімалізму та надає базовий набір інструментів для розробки, залишаючи розробнику свободу у виборі додаткових компонентів, таких як робота з базами даних, авторизація або обробка запитів [15].

Основною перевагою Flask є його простота та гнучкість. Він не нав'язує жорсткої архітектури проєкту, що дозволяє створювати як невеликі навчальні застосунки, так і масштабовані веб-системи. Flask підтримує маршрутизацію

запитів, роботу з HTTP-методами (GET, POST тощо), а також використання шаблонів для формування веб-сторінок [16].

Фреймворк працює на основі WSGI (Web Server Gateway Interface), що забезпечує взаємодію між веб-сервером і Python-застосунком [17]. Запити від користувача обробляються через визначені маршрути (routes), після чого система виконує відповідну логіку і повертає результат у вигляді HTML-сторінки або JSON-відповіді.

У задачах машинного навчання та аналізу даних Flask часто використовується для створення веб-інтерфейсів, які дозволяють користувачам завантажувати дані, запускати моделі, отримувати результати кластеризації або класифікації та переглядати візуалізації. Завдяки цьому Flask виступає як зручний інтерфейс між алгоритмами обробки даних і кінцевим користувачем.

Таким чином, Flask є ефективним інструментом для розробки веб-застосунків, який поєднує простоту використання з достатньою функціональністю для реалізації складних систем, включаючи проєкти в галузі машинного навчання та аналізу даних.

2.8 Бібліотека SQLAlchemy та розширення Flask-SQLAlchemy

SQLAlchemy — це потужна бібліотека для Python, яка використовується для роботи з базами даних [18]. Вона реалізує підхід ORM (Object Relational Mapping), тобто дозволяє працювати з таблицями бази даних як із Python-об'єктами [19]. Замість написання SQL-запитів вручну, розробник описує моделі у вигляді класів, а бібліотека автоматично перетворює операції над цими об'єктами у SQL-команди [20]. SQLAlchemy підтримує складні запити, зв'язки між таблицями, транзакції та різні типи баз даних (SQLite, PostgreSQL, MySQL тощо). Завдяки цьому код стає більш структурованим, безпечним і зручним для підтримки.

Flask-SQLAlchemy — це розширення для фреймворку Flask, яке спрощує інтеграцію SQLAlchemy у веб-застосунки. Воно додає додатковий рівень абстракції, який робить налаштування бази даних простішим і менш громіздким.

Розробнику достатньо підключити розширення, задати рядок підключення до бази даних і створити моделі, після чого можна одразу працювати з даними через ORM. Flask-SQLAlchemy також забезпечує зручну інтеграцію з контекстом Flask-застосунку, включаючи роботу із запитами, сесіями та міграціями.

Основна різниця між ними полягає в тому, що SQLAlchemy є самостійною бібліотекою для роботи з базами даних, яка може використовуватись у будь-якому Python-проєкті, тоді як Flask-SQLAlchemy є надбудовою саме для Flask, яка спрощує її використання у веб-застосунках. SQLAlchemy дає більше контролю та гнучкості, а Flask-SQLAlchemy — простіший і швидший старт для розробки веб - систем [21].

2.9 Логіка вибору інструментів

У процесі проєктування системи було сформовано декілька популярних програмних екосистем (пакетів) для задач машинного навчання та аналізу даних (табл. 2.1). Метою було визначення найбільш доцільного середовища для реалізації задач класифікації та кластеризації зображень кров'яних клітин.

Таблиця 2.1 – Порівняння програмних екосистем (пакетів Python, R та Java для задач машинного навчання)

Категорія	Python пакет	R пакет	Java пакет
Core / мова	Python	R	Java
Web / API	Flask	plumber	Spring Boot
Робота з БД / ORM	SQLAlchemy / Flask-SQLAlchemy	DBI / RSQLite	Hibernate / JPA
Обробка даних	NumPy	dplyr	Apache Commons / Weka
Візуалізація	Matplotlib	ggplot2	JFreeChart
Machine Learning	Scikit-learn	caret / mlr	Weka / Java-ML
Deep Learning	Keras / TensorFlow	keras	Deeplearning4j

Для цього було проведено аналіз існуючих інструментів у різних мовах програмування, зокрема Python, R [22] та Java [23], з урахуванням їхніх можливостей у сфері обробки зображень та побудови ML та DL моделей. Окрему

увагу приділено тому, які саме бібліотеки та фреймворки доступні в кожній екосистемі та наскільки вони підходять для вирішення поставленої задачі.

Для обґрунтованого вибору оптимального варіанту було застосовано метод аналізу ієрархій (AHP), який дозволяє формалізувати процес прийняття рішення шляхом використання парних порівнянь та обчислення вагових коефіцієнтів [24 - 26].

Для виконання парних порівнянь використано шкалу відносної важливості Сааті (табл. 2.2), яка дозволяє кількісно оцінити перевагу одного елемента над іншим.

Таблиця 2.2 – Saaty Scale (шкала відносної важливості) [27]

Пряме значення	**Інтерпретація**	**Зворотне значення**
9	Абсолютна перевага	1/9
8	Між дуже сильною і абсолютною	1/8
7	Дуже сильна перевага	1/7
6	Між сильною і дуже сильною	1/6
5	Сильна (суттєва) перевага	1/5
4	Між помірною і сильною	1/4
3	Помірна перевага	1/3
2	Між рівною і помірною	1/2
1	Рівна важливість	1

На основі поставленої задачі було визначено критерії оцінювання (табл. 2.3) та альтернативи (табл. 2.4), між якими здійснюється вибір.

Таблиця 2.3 – Критерії для порівняння

Код	**Критерій**	**Короткий опис**
C1	Web / API development	Розробка веб-додатків, REST API та серверної логіки
C2	Data Science / Machine Learning	Обробка даних, статистичний аналіз та побудова ML моделей
C3	Scalability	Здатність мови підтримувати великі системи та високі навантаження

Таблиця 2.4 – Альтернативи

Код	Альтерантива
O1	Python пакет
O2	R пакет
O3	Java пакет

Далі для кожного критерію було побудовано матриці парних порівнянь альтернатив (табл. 2.5–2.7), які відображають відносну перевагу кожного програмного пакета за відповідним критерієм.

Таблиця 2.5 – Критерій C1 (Web / API development)

	Python пакет	**R пакет**	**Java пакет**
Python пакет	1	6	2
R пакет	1/6	1	1/4
Java пакет	1/2	4	1

Таблиця 2.6 – Критерій C2 (Data Science / Machine Learning)

	Python пакет	**R пакет**	**Java пакет**
Python пакет	1	2	4
R пакет	1/2	1	2
Java пакет	1/4	1/2	1

Таблиця 2.7 – Критерій C3 (Scalability)

	Python	**R**	**Java**
Python пакет	1	3	1/3
R пакет	1/3	1	1/9
Java пакет	3	9	1

Для визначення ваг критеріїв було сформовано матрицю парних порівнянь критеріїв (табл. 2.8).

Таблиця 2.8 – Критерії

	C1 (Web / API development)	**C2 (Data Science / Machine Learning)**	**C3 (Scalability)**
C1 (Web / API development)	1	1/3	1/2
C2 (Data Science / Machine Learning)	3	1	2
C3 (Scalability)	2	1/2	1

На основі отриманих матриць обчислюється власний вектор за формулою (2.1), який визначається як геометричне середнє елементів рядка. Далі виконується нормалізація власного вектора за формулою (2.2), в результаті чого отримується вектор пріоритетів, що відображає вагу кожного критерію.

$$w_i = \left(\prod_{j=1}^{n} a_{ij}\right)^{\frac{1}{n}}, \quad (2.1)$$

де w_i – компонента власного вектора (геометричне середнє і-го рядка);

a_{ij} – елемент матриці попарних порівнянь;

n – кількість критеріїв.

$$P_i = \frac{w_i}{\sum_{k=1}^{n} w_k}, \quad (2.2)$$

де P_i – нормалізована вага (вектор пріоритетів) і-го критерію;

w_i – компонента власного вектора, отримана як геометричне середнє;

n – кількість критеріїв;

$\sum_{k=1}^{n} w_k$ – сума всіх компонент власного вектора.

Наступним кроком є обчислення максимального власного значення матриці λ_{max} за формулою (2.3), яке використовується для перевірки узгодженості суджень.

$$\lambda_{max} = \sum_{i=1}^{n} a_{i,1} \cdot P_1 + \sum_{i=1}^{n} a_{i,2} \cdot P_2 + \cdots + \sum_{i=1}^{n} a_{i,n} \cdot P_n, \quad (2.3)$$

де $P_{\text{і}}$ – максимальне власне значення матриці попарних порівнянь;

$a_{i,j}$ – елемент матриці попарних порівнянь;

P_{j} – компонента нормалізованого вектора пріоритетів;

n – кількість критеріїв.

Для оцінки узгодженості обчислюється індекс узгодженості (2.4) та відношення узгодженості (2.5). Значення $C_R < 0.1$ свідчить про прийнятну узгодженість експертних оцінок.

$$C_I = (\lambda_{\max} - n) / (n - 1), \tag{2.4}$$

де C_I – індекс узгодженості;

$\lambda_{\max}$ – максимальне власне значення матриці;

n – кількість критеріїв.

$$C_R = \frac{C_I}{R_I} \cdot 100\ \%, \tag{2.5}$$

де C_R – відношення узгодженості;

C_I – індекс узгодженості;

R_I – випадковий індекс (Random Index), табличне значення;

n – кількість критеріїв.

Вибір значення випадкового індексу RI здійснюється залежно від розмірності матриці парних порівнянь. Для цього використовується табличне представлення стандартних значень, запропонованих Сааті (табл. 2.9).

Таблиця 2.9 – Значення Random Index (RI) [28]

n	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
RI	0.00	0.00	0.58	0.90	1.12	1.24	1.32	1.41	1.45	1.49

Для матриці парних порівнянь критеріїв (табл. 2.8) наведено приклад обчислення вагових коефіцієнтів із використанням методу аналізу ієрархій.

$$w_1 = \left(1 \cdot (1/3) \cdot (1/2)\right)^{1/3} = 0.55\,,$$

$$w_2 = (3 \cdot 1 \cdot 2)^{1/3} = 1.82\,,$$

$$w_3 = (2 \cdot (1/2) \cdot 1)^{1/3} = 1.00\,,$$

$$\sum w_i = 0.55 + 1.82 + 1.00 = 3.37\,,$$

$$P_1 = 0.55/3.37 = 0.163\,,$$

$$P_2 = 1.82/3.37 = 0.546\,,$$

$$P_3 = 1.00/3.37 = 0.297\,,$$

$$\lambda_{max} = 6 \cdot 0.163 + 1.8333 \cdot 0.540 + 3.5 \cdot 0.297 = 0.978 + 0.990 + 1.040 = 3.0092\,,$$

$$C_I = (3.0092 - 3)/(3 - 1) = 0.01/2 = 0.0046\,,$$

$$C_R = 0.0046/0.58 = 0.0079 \cdot 100 = 0.79\,\%\,.$$

У результаті виконаних розрахунків отримано вектори пріоритетів для критеріїв та альтернатив, які наведено в табл. 2.9.

Таблиця 2.9 – Вектори пріоритетів

Назва критерія	**Критерії**	**Назва альтренативи**	**C1 (Web / API)**	**C2 (DS/ML)**	**C3 (Scalability)**
C1 (Web / API)	0,16	O1 (Python пакет)	0,59	0,57	0,23
C2 (DS/ML)	0,54	O2 (R пакет)	0,09	0,29	0,08
C3 (Scalability)	0,30	O3 (Java пакет)	0,32	0,14	0,69
SUM	1,00	SUM	1,00	1,00	1,00

На основі отриманих даних також було обчислено показники узгодженості для всіх матриць парних порівнянь. Значення індексу узгодженості CI та відношення узгодженості CR наведено в табл. 2.10.

Таблиця 2.10 – Коефіцієнти узгодження

Тип	CI, %	CI
Критерії	0,79	0,005
C1 (Web / API)	0,79	0,005
C2 (DS/ML)	0,00	0,000
C3 (Scalability)	0,00	0,000

На завершальному етапі виконується синтез глобальних пріоритетів альтернатив відповідно до формули (2.6), що дозволяє визначити загальну перевагу кожної альтернативи з урахуванням усіх критеріїв.

$$P_i^G = \sum_{j=1}^{n} P_i^j \cdot P_j \ . \tag{2.6}$$

Розрахунки виконуються нижче:

$$P_1^G = 0.16 \cdot 0.59 + 0.54 \cdot 0.57 + 0.30 \cdot 0.23 = 0.0944 + 0.3078 + 0.069 = 0.4712 \ ,$$

$$P_2^G = 0.16 \cdot 0.09 + 0.54 \cdot 0.29 + 0.30 \cdot 0.08 = 0.0144 + 0.1566 + 0.024 = 0.1950 \ ,$$

$$P_3^G = 0.16 \cdot 0.32 + 0.54 \cdot 0.14 + 0.30 \cdot 0.69 = 0.0512 + 0.0756 + 0.207 = 0.3338 \ ,$$

$$\sum P_i^G = 0.4712 + 0.195 + 0.3338 = 1.0000 \ .$$

Результати розрахунків глобальних пріоритетів альтернатив наведено в табл. 2.11.

Таблиця 2.11 – Результати розрахунків глобальних пріоритетів альтернатив

Альтернатива	Глобальний пріоритет
O1 (Python пакет)	0,4712
O2 (R пакет)	0,1950
O3 (Java пакет)	0,3338
SUM	1,0000

Отримані значення свідчать про те, що найбільший глобальний пріоритет має альтернатива O1 (Python пакет), що вказує на її перевагу серед розглянутих варіантів.

Таким чином, саме Python пакет було обрано як основне середовище для реалізації задач класифікації та кластеризації зображень кров'яних клітин.

Достовірність отриманих результатів обґрунтовується використанням методу аналізу ієрархій (AHP), який є одним із класичних методів багатокритеріального прийняття рішень. Його перевагою є можливість формалізації експертних оцінок, їх математичної обробки та перевірки узгодженості суджень за допомогою індексу узгодженості CI та відношення узгодженості CR.

Оскільки у всіх матрицях парних порівнянь виконано умову $CR<0.1$, можна зробити висновок про достатню узгодженість експертних оцінок, а отже — про коректність та надійність отриманих результатів.

Висновки до розділу 2

У даному розділі було розглянуто основні технології та інструменти, обрані для реалізації задач аналізу зображень клітин крові. Проведений аналіз показав, що для ефективного вирішення задач класифікації та кластеризації доцільно використовувати сучасні засоби машинного навчання та глибокого навчання, які забезпечують автоматичне виділення інформативних ознак із даних.

У якості основної мови програмування обрано Python, що обґрунтовується його простотою, гнучкістю та наявністю розвиненої екосистеми бібліотек для обробки даних і побудови моделей. Було розглянуто ключові бібліотеки, зокрема TensorFlow та Keras для створення і навчання нейронних мереж, Scikit-learn для реалізації алгоритмів кластеризації та зменшення розмірності, NumPy для ефективної роботи з числовими масивами, а також Matplotlib для візуалізації результатів.

Окрему увагу приділено використанню фреймворку Flask, який може бути застосований для подальшої інтеграції розроблених моделей у веб-застосунок, а також бібліотеки SQLAlchemy для організації взаємодії з базою даних. Це створює передумови для розширення роботи до повноцінної інтелектуальної системи в майбутньому.

З метою обґрунтування вибору програмного середовища було проведено порівняльний аналіз альтернативних екосистем (Python, R, Java) із використанням методу аналізу ієрархій (AHP). У процесі аналізу враховувалися такі критерії, як можливості розробки веб-застосунків, підтримка задач машинного навчання та масштабованість. Результати розрахунків показали, що Python має найбільший глобальний пріоритет серед розглянутих альтернатив.

Крім того, було перевірено узгодженість експертних оцінок за допомогою показників CI та CR, значення яких не перевищують допустимий поріг. Це свідчить про надійність отриманих результатів та коректність проведеного аналізу.

Таким чином, обраний стек технологій є обґрунтованим і забезпечує необхідні інструменти для реалізації поставленої задачі. Використання зазначених бібліотек і фреймворків створює основу для подальшої реалізації та дослідження моделей класифікації і кластеризації зображень клітин крові, що буде розглянуто у наступному розділі.

3 НАВЧАННЯ МОДЕЛЕЙ, ОЦІНКА РЕЗУЛЬТАТІВ НАВЧАННЯ

3.1 Реалізація навчання моделі класифікації

Для реалізації процесу навчання використання стандартних *.py файлів є менш зручним, оскільки виникнення помилки призводить до зупинки всього процесу виконання. Натомість більш доцільним є застосування *.ipynb ноутбуків [29], які дозволяють виконувати код поетапно, у вигляді окремих блоків, та зберігати проміжні результати обчислень.

Процес навчання моделі було реалізовано у середовищі Google Colab, що дозволяє виконувати обчислення у хмарі та використовувати доступні обчислювальні ресурси, зокрема GPU [30]. Це значно спрощує роботу з великими обсягами даних та прискорює обробку моделей машинного навчання.

Для доступу до локальних файлів і збереження результатів використовувалося підключення Google Drive (рис. 3.1). Це реалізовано за допомогою відповідного модуля, що дозволяє працювати з файлами безпосередньо з хмарного сховища [31].

```
[ ]
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D, Dense, Dropout, Flatten, BatchNormalization
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.model_selection import train_test_split
from sklearn.metrics import silhouette_score, calinski_harabasz_score, davies_bouldin_score
import matplotlib.pyplot as plt
import matplotlib.pyplot as plt
import numpy as np
import os
from PIL import Image
import seaborn as sns
import time
import pickle
import random

[ ]
from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive

[ ]
!cp -r /content/drive/MyDrive/data_cells /content/
```

Рисунок 3.1 – Іморт бібліотек та завантаження датасету

Перед реалізацією задачі кластеризації ознак зображень було доцільно спочатку розглянути більш просту та контрольовану задачу — класифікацію. Це дозволяє перевірити якість ознак, що виділяються моделлю, а також оцінити загальну придатність даних для подальшого аналізу.

На рис. 3.2 підготовляються дані для навчання, валідації та тестування нейронної мережі. Для навчального набору зображення проходять попередню обробку та різні трансформації, такі як обертання, масштабування, зсуви та горизонтальне відображення, щоб зробити дані більш різноманітними і зменшити ризик перенавчання [32]. Для валідаційного та тестового наборів застосовується лише базова обробка, без аугментації, щоб оцінка моделі відображала реальні дані.

Дані організовані за папками відповідно до класів, і кожен набір зображень підбирається у потрібному розмірі. Навчальний набір перемішується для кращого навчання, а валідаційний та тестовий залишають порядок незмінним. В результаті модель отримує доступ до близько 12 тисяч зображень для навчання, понад 1,7 тисяч для валідації та понад 3,4 тисяч для тестування, усі з восьми класів, що забезпечує збалансовану підготовку та точну оцінку продуктивності.

```
real_size = (360,363)

train_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input,
    rotation_range=20,
    zoom_range=0.2,
    horizontal_flip=True,
    width_shift_range=0.1,
    height_shift_range=0.1
)

val_test_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input
)

train_generator = train_datagen.flow_from_directory(
    os.path.join(output_dir, 'train'),
    target_size=real_size,
    batch_size=32,
    class_mode='categorical',
    shuffle=True,
    seed=SEED
)
```

```
validation_generator = val_test_datagen.flow_from_directory(
    os.path.join(output_dir, 'val'),
    target_size=real_size,
    batch_size=32,
    class_mode='categorical',
    shuffle=False,
    seed=SEED
)

test_generator = val_test_datagen.flow_from_directory(
    os.path.join(output_dir, 'test'),
    target_size=real_size,
    batch_size=32,
    class_mode='categorical',
    shuffle=False,
    seed=SEED
)
```

```
Found 11959 images belonging to 8 classes.
Found 1705 images belonging to 8 classes.
Found 3428 images belonging to 8 classes.
```

Рисунок 3.2 – Створення тренувальної, валідаційної, та тестової вибірок

Після підготовки вибірок було створено програмний код для попереднього перегляду елементів датасету (рис. 3.3). Це дозволило візуально оцінити якість зображень, їх різноманітність та відповідність класам. Такий підхід допомагає своєчасно виявити проблемні дані, наприклад, зображення низької якості, пошкоджені файли або неправильно марковані класи. Крім того, попередній перегляд забезпечує краще розуміння структури датасету, що полегшує подальшу обробку, нормалізацію та підготовку до навчання моделей машинного навчання.

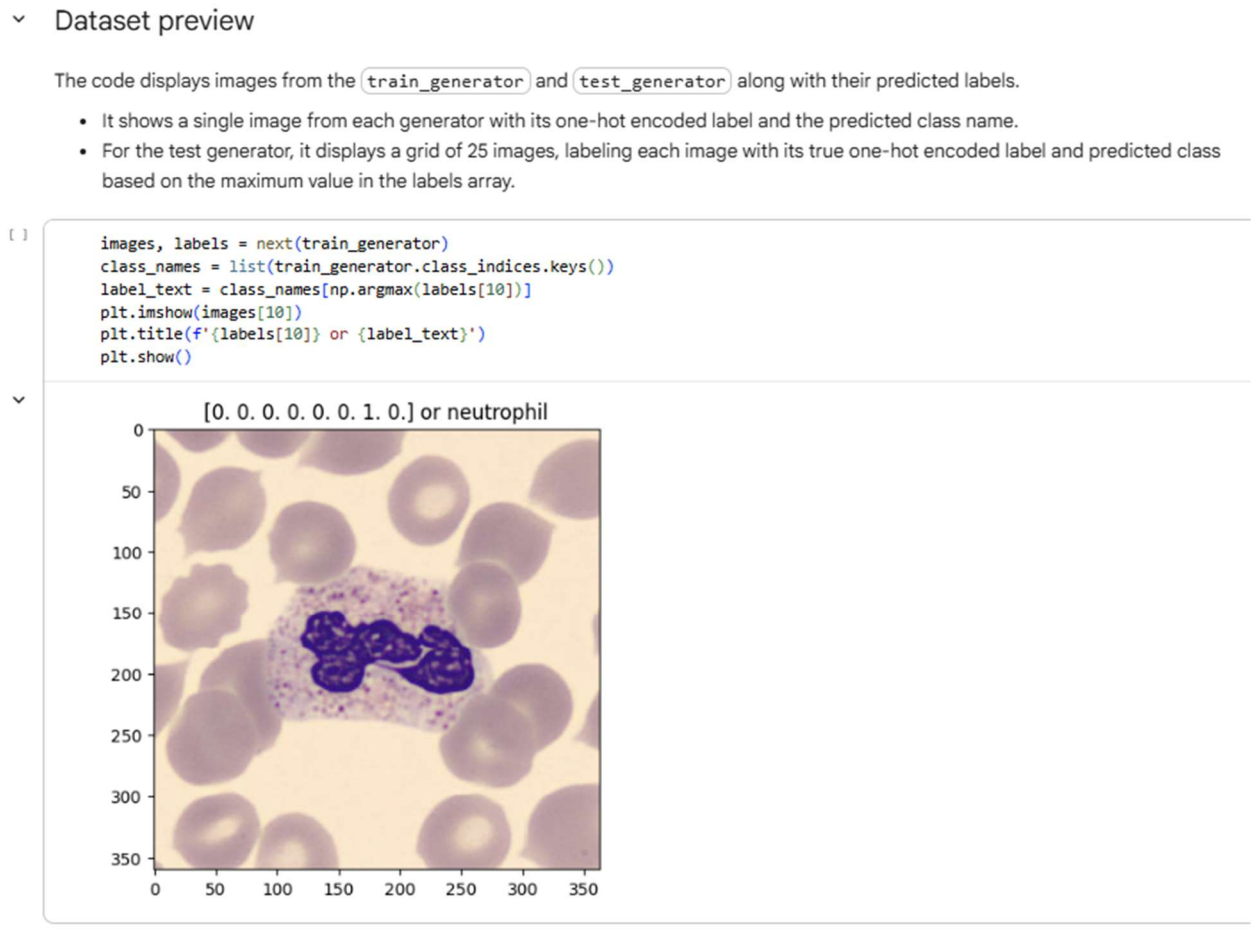


Рисунок 3.3 – Фрагмент коду для превью елементів датасету

На рис. 3.3 також можна побачити, що всі класи були перетворені у формат one-hot encoding, що дозволяє представляти категоріальні мітки у вигляді бінарних векторів для подальшого використання в нейронних мережах. Більш наочно це видно на рис. 3.4.

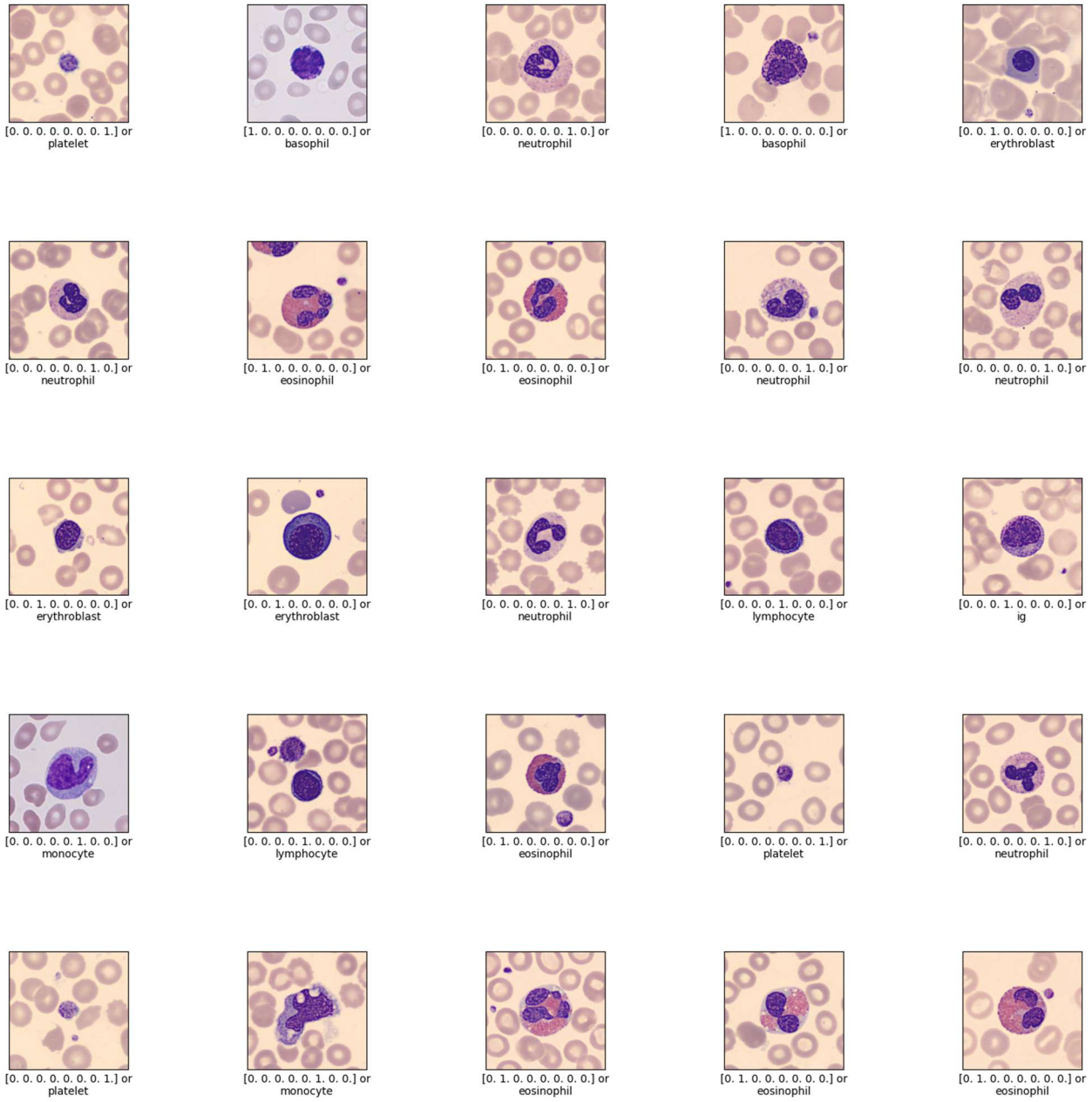


Рисунок 3.3 – Фрагмент датасету після one-hot encoding

Далі було розроблено модель, яка є згортковою нейронною мережею (CNN) для класифікації зображень кров'яних клітин (табл 3.1). Вона складається з чотирьох блоків Conv2D + BatchNormalization + MaxPooling2D для виділення ознак, після чого йде Flatten для перетворення багатовимірних ознак у вектор, два щільні шари Dense з нормалізацією та дроп-аутом для запобігання перенавчанню, і фінальний шар з Softmax для багатокласової класифікації [33-38].

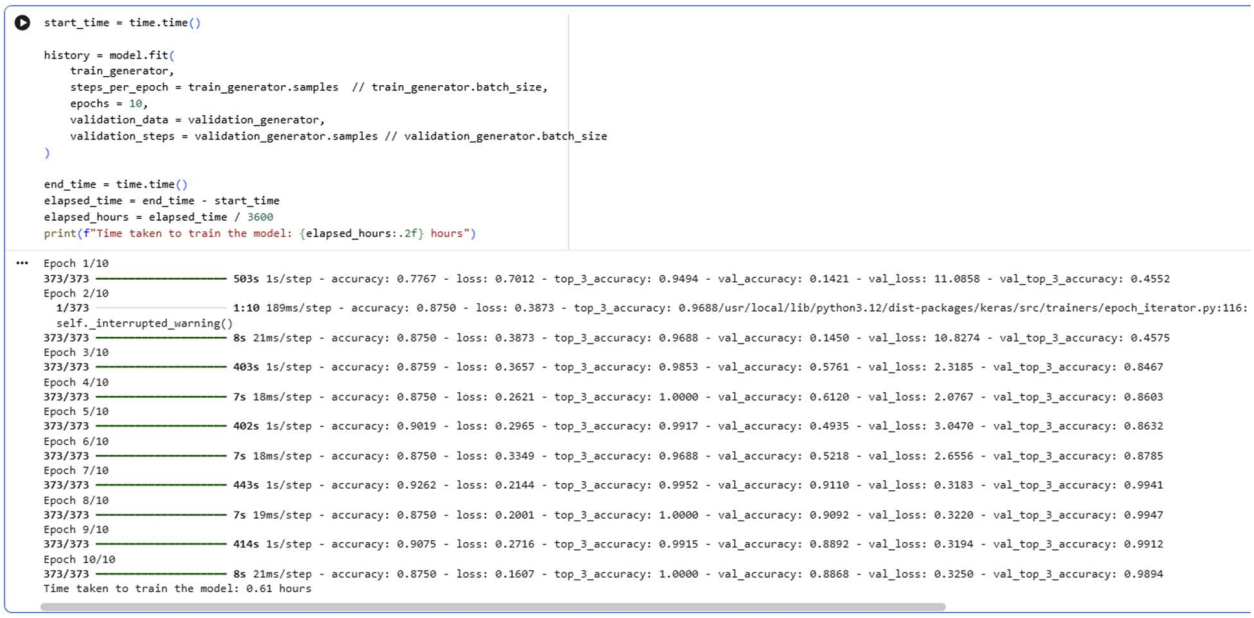
Таблиця 3.1 – Архітектура моделі класифікації зображень кров'яних клітин

Шар	**Тип**	**Вхід**	**Вихід**	**Активація / Особливості**
1	Conv2D	(360, 363, 3)	(358, 361, 32)	ReLU
2	BatchNormalization	(358, 361, 32)	(358, 361, 32)	-
3	MaxPooling2D	(358, 361, 32)	(179, 180, 32)	-
4	Conv2D	(179, 180, 32)	(177, 178, 64)	ReLU
5	BatchNormalization	(177, 178, 64)	(177, 178, 64)	-
6	MaxPooling2D	(177, 178, 64)	(88, 89, 64)	-
7	Conv2D	(88, 89, 64)	(86, 87, 128)	ReLU
8	BatchNormalization	(86, 87, 128)	(86, 87, 128)	-
9	MaxPooling2D	(86, 87, 128)	(43, 43, 128)	-
10	Conv2D	(43, 43, 128)	(41, 41, 128)	ReLU
11	BatchNormalization	(41, 41, 128)	(41, 41, 128)	-
12	MaxPooling2D	(41, 41, 128)	(20, 20, 128)	-
13	Flatten	(20, 20, 128)	51200	-
14	Dense	51200	512	ReLU
15	BatchNormalization	512	512	-
16	Dropout	512	512	rate=0.5
17	Dense	512	num_classes	Softmax

Модель продемонструвала високу та стабільну точність як на навчальній, так і на тестовій вибірках. Підсумковий показник точності (Accuracy) на навчанні становить 0.87, тоді як на тестових даних він навіть дещо вищий — 0.89. Це свідчить про хорошу здатність моделі до узагальнення та відсутність критичного перенавчання. Особливу увагу варто звернути на метрику Top-3 Accuracy, яка сягнула вражаючих 0.99, що підтверджує здатність алгоритму майже безпомилково вгадувати правильний клас у межах трьох найбільш імовірних варіантів.

Процес навчання тривав 10 епох (близько 36 хвилин) і супроводжувався поступовим зниженням функції втрат. Згідно з логами, вже на 7-й епосі модель

вийшла на високі показники валідації (val_accuracy: 0.91), які зберіглися до кінця тренування. Невелика різниця між тренувальною та тестовою точністю вказує на те, що модель була грамотно налаштована та ефективно розпізнає закономірності в даних, забезпечуючи надійні результати класифікації (рис. 3.4).



```
start_time = time.time()

history = model.fit(
    train_generator,
    steps_per_epoch = train_generator.samples  // train_generator.batch_size,
    epochs = 10,
    validation_data = validation_generator,
    validation_steps = validation_generator.samples // validation_generator.batch_size
)

end_time = time.time()
elapsed_time = end_time - start_time
elapsed_hours = elapsed_time / 3600
print(f"Time taken to train the model: {elapsed_hours:.2f} hours")
```

```
Epoch 1/10
373/373 ━━━━━━━━━━━━━━━━━━━━ 503s 1s/step - accuracy: 0.7767 - loss: 0.7012 - top_3_accuracy: 0.9494 - val_accuracy: 0.1421 - val_loss: 11.0858 - val_top_3_accuracy: 0.4552
Epoch 2/10
  1/373 ━━━━━━━━━━━━━━━━━━━━ 1:10 189ms/step - accuracy: 0.8750 - loss: 0.3873 - top_3_accuracy: 0.9688/usr/local/lib/python3.12/dist-packages/keras/src/trainers/epoch_iterator.py:116:
  self._interrupted_warning()
373/373 ━━━━━━━━━━━━━━━━━━━━ 8s 21ms/step - accuracy: 0.8750 - loss: 0.3873 - top_3_accuracy: 0.9688 - val_accuracy: 0.1450 - val_loss: 10.8274 - val_top_3_accuracy: 0.4575
Epoch 3/10
373/373 ━━━━━━━━━━━━━━━━━━━━ 403s 1s/step - accuracy: 0.8759 - loss: 0.3657 - top_3_accuracy: 0.9853 - val_accuracy: 0.5761 - val_loss: 2.3185 - val_top_3_accuracy: 0.8467
Epoch 4/10
373/373 ━━━━━━━━━━━━━━━━━━━━ 7s 18ms/step - accuracy: 0.8750 - loss: 0.2621 - top_3_accuracy: 1.0000 - val_accuracy: 0.6120 - val_loss: 2.0767 - val_top_3_accuracy: 0.8603
Epoch 5/10
373/373 ━━━━━━━━━━━━━━━━━━━━ 402s 1s/step - accuracy: 0.9019 - loss: 0.2965 - top_3_accuracy: 0.9917 - val_accuracy: 0.4935 - val_loss: 3.0470 - val_top_3_accuracy: 0.8632
Epoch 6/10
373/373 ━━━━━━━━━━━━━━━━━━━━ 7s 18ms/step - accuracy: 0.8750 - loss: 0.3349 - top_3_accuracy: 0.9688 - val_accuracy: 0.5218 - val_loss: 2.6556 - val_top_3_accuracy: 0.8785
Epoch 7/10
373/373 ━━━━━━━━━━━━━━━━━━━━ 443s 1s/step - accuracy: 0.9262 - loss: 0.2144 - top_3_accuracy: 0.9952 - val_accuracy: 0.9110 - val_loss: 0.3183 - val_top_3_accuracy: 0.9941
Epoch 8/10
373/373 ━━━━━━━━━━━━━━━━━━━━ 7s 19ms/step - accuracy: 0.8750 - loss: 0.2001 - top_3_accuracy: 1.0000 - val_accuracy: 0.9092 - val_loss: 0.3220 - val_top_3_accuracy: 0.9947
Epoch 9/10
373/373 ━━━━━━━━━━━━━━━━━━━━ 414s 1s/step - accuracy: 0.9075 - loss: 0.2716 - top_3_accuracy: 0.9915 - val_accuracy: 0.8892 - val_loss: 0.3194 - val_top_3_accuracy: 0.9912
Epoch 10/10
373/373 ━━━━━━━━━━━━━━━━━━━━ 8s 21ms/step - accuracy: 0.8750 - loss: 0.1607 - top_3_accuracy: 1.0000 - val_accuracy: 0.8868 - val_loss: 0.3250 - val_top_3_accuracy: 0.9894
Time taken to train the model: 0.61 hours
```

Рисунок 3.3 – Процес навчання CNN

Для оцінки якості класифікації було побудовано матрицю помилок [39], яка підтвердила високу точність моделі: більшість прогнозів зосереджена на головній діагоналі. Найкращі результати продемонстровано для класів eosinophil, neutrophil та platelet, що свідчить про ефективне розпізнавання їхніх ключових ознак.

Основним недоліком є складність диференціації класу ig, який модель часто плутає з monocyte (85 випадків). Це вказує на візуальну схожість даних типів клітин (рис. 3.4).

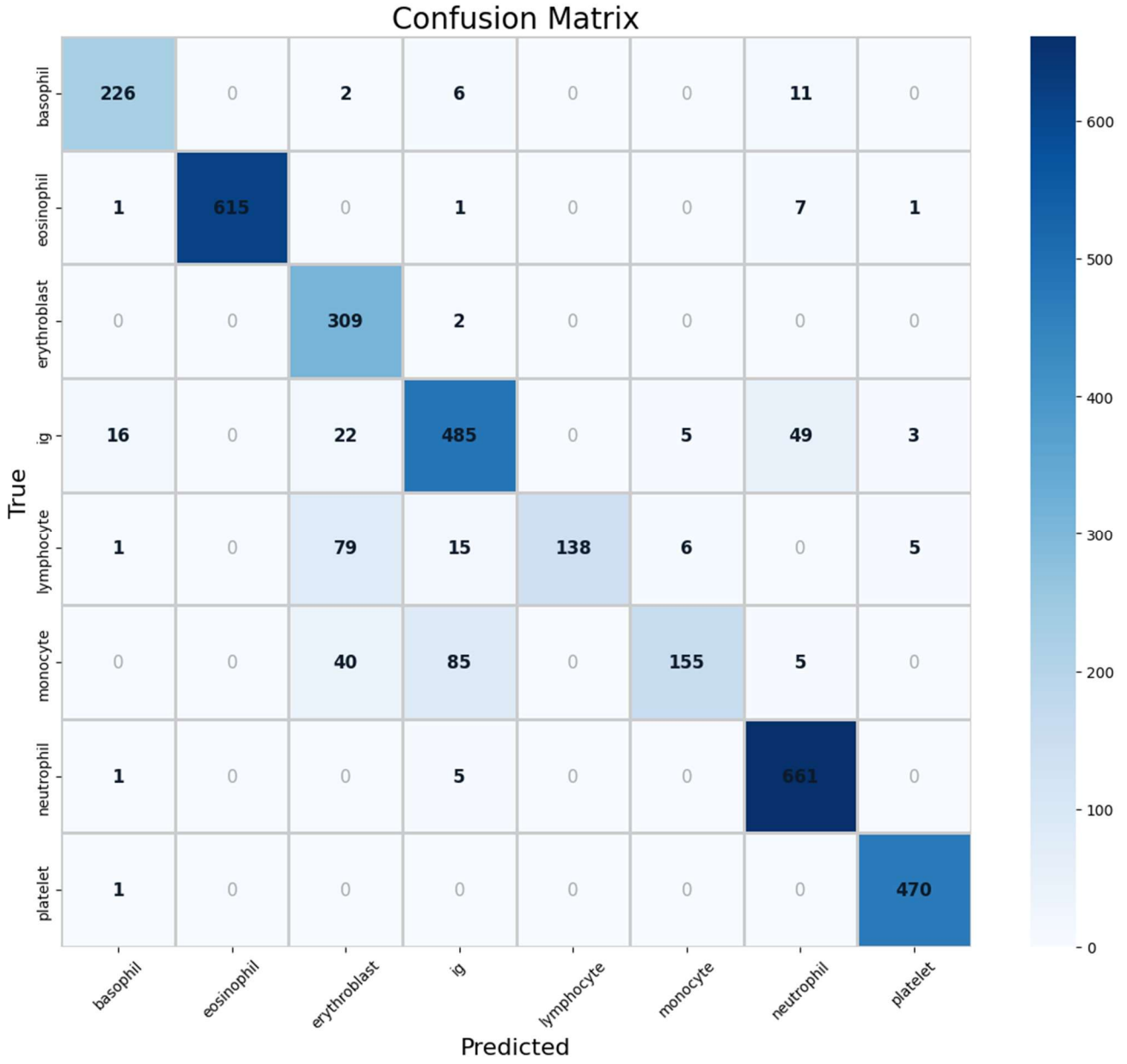


Рисунок 3.4 – Матриця помилок

Візуальне порівняння на рис. 3.5 пояснює складність класифікації: обидва типи клітин мають великі розміри, значний об'єм світлої цитоплазми та масивні несегментовані ядра. Подібність бобоподібного ядра моноцита та округлого ядра ig створює ідентичний силует, що збиває модель. Оскільки дрібна зернистість у цитоплазмі ig за кольором і текстурою майже не відрізняється від «пінистої» структури моноцита, нейромережа часто ідентифікує ці різні класи як один об'єкт.

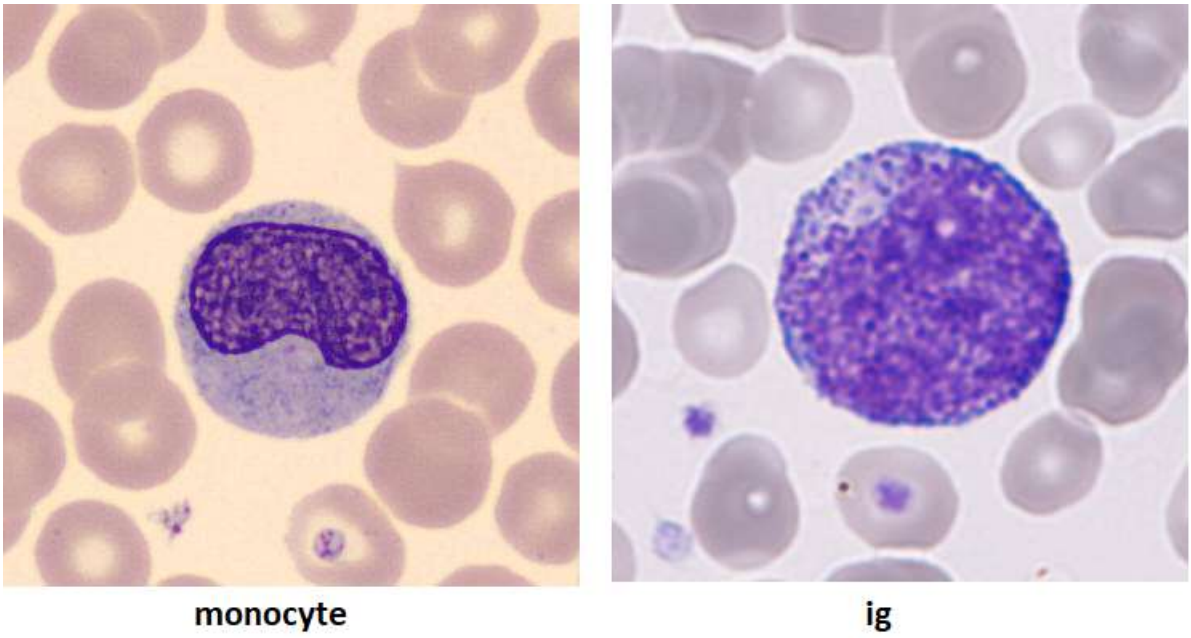


Рисунок 3.5 – Порівняння клітини monocyte та ig

3.2 Реалізація навчання моделі кластеризації

Було завантажено зображення з папки train_dir, приведено їх до розміру 360×363, переведено у формат RGB та згруповано по 50 штук у батчі (рис. 3.6).

```
[ ]
train_dir = '/content/data_cells'

train_dataGen = ImageDataGenerator(preprocessing_function=preprocess_input)

train_generator = train_dataGen.flow_from_directory(
    directory=train_dir,
    target_size=(360, 363),
    color_mode='rgb',
    batch_size=50,
    class_mode=None,
    shuffle=False
)

filenames = train_generator.filenames
print(f"Loaded {len(filenames)} images")

Found 17092 images belonging to 8 classes.
Loaded 17092 images
```

Рисунок 3.6 – Завантаження даних для кластеризації

Далі витягнуті ознаки зображень за допомогою моделі MobileNetV2 із попередньо навченими на ImageNet вагами, без верхнього класифікаційного шару та із середнім пулінгом для отримання векторів фіксованої довжини. Зображення з генератора train_generator пропустили через модель, отримавши матрицю ознак розміром 17092×1280, де кожен рядок відповідає вектору ознак одного зображення, що підготовлений для подальшої кластеризації (рис. 3.7).

```
feature_extractor = MobileNetV2(weights='imagenet', include_top=False, pooling='avg', input_shape=(360,363,3))

/tmp/ipython-input-2233695580.py:1: UserWarning: `input_shape` is undefined or non-square, or `rows` is not in
  feature_extractor = MobileNetV2(weights='imagenet', include_top=False, pooling='avg', input_shape=(360,363,3)
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v2/mobilenet_v2_we
9406464/9406464 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```

```
features = feature_extractor.predict(train_generator, verbose=1)
print("Extracted features shape:", features.shape)

/usr/local/lib/python3.12/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning
  self._warn_if_super_not_called()
342/342 ━━━━━━━━━━━━━━━━━━━━ 2125s 6s/step
Extracted features shape: (17092, 1280)
```

Рисунок 3.7 – Витагання ознак зображень за допомогою моделі MobileNetV2

Було обрано вісім кластерів, відповідно до кількості класів клітин у датасеті, і за допомогою алгоритму KMeans кожному зображенню призначено свій кластер.

```
[ ]
num_clusters = 8
kmeans = KMeans(n_clusters=num_clusters, random_state=42)
cluster_labels = kmeans.fit_predict(features)

print("Cluster labels assigned:", cluster_labels)

Cluster labels assigned: [0 5 2 ... 7 2 2]
```

Рисунок 3.8 – Тренування KMeans моделі

Далі ознаки було зменшено до двох вимірів за допомогою PCA для візуалізації, після чого результати було відображено на графіку, де кожен кластер позначено своїм кольором, що дозволяє наочно оцінити розподіл зображень між групами (3.9).

```
pca = PCA(n_components=2, random_state=42)
features_2d = pca.fit_transform(features)

print("Reduced features shape:", features_2d.shape)

plt.figure(figsize=(10, 8))

num_clusters = len(np.unique(cluster_labels))
colors = plt.cm.get_cmap('tab20', num_clusters)

for cluster_id in range(num_clusters):
    idx = cluster_labels == cluster_id
    plt.scatter(features_2d[idx, 0], features_2d[idx, 1],
                label=f'Cluster {cluster_id}', alpha=0.6, s=40,
                color=colors(cluster_id))
```

Рисунок 3.9 – Зменшено до двох вимірів за допомогою PCA

Результат PCA продемонстровано на рис. 3.10. Як видно, всі точки злилися в одну купу, і візуально чітких кластерів не спостерігається.

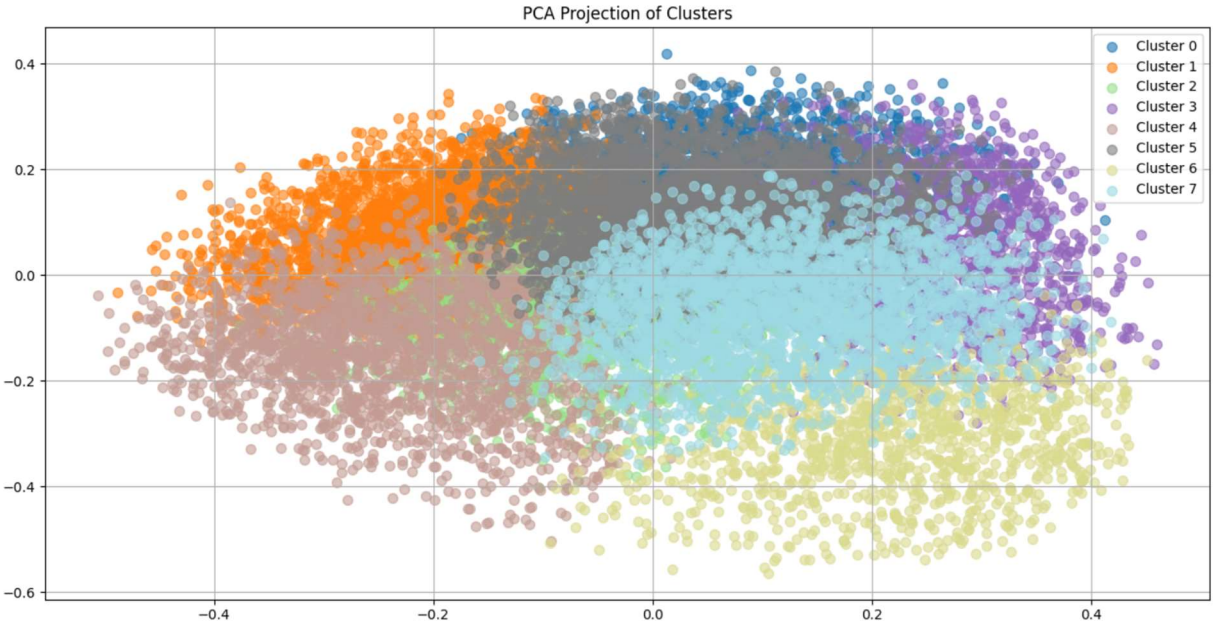


Рисунок 3.10 – PCA проекція кластерів кров'яних клітин

На рис. 3.11 продемонстровано порівняння кластерів з реальними цільовими мітками. Лівий графік показує кластери, середній – справжні класи, а правий поєднує колір (класи) та форму (кластери). Видно, що деякі класи клітин мають тенденцію групуватися разом у певних кластерах: наприклад, basophil та eosinophil частково збігаються у кластері 0 і кластері 1, що свідчить про подібність їх морфологічних ознак.

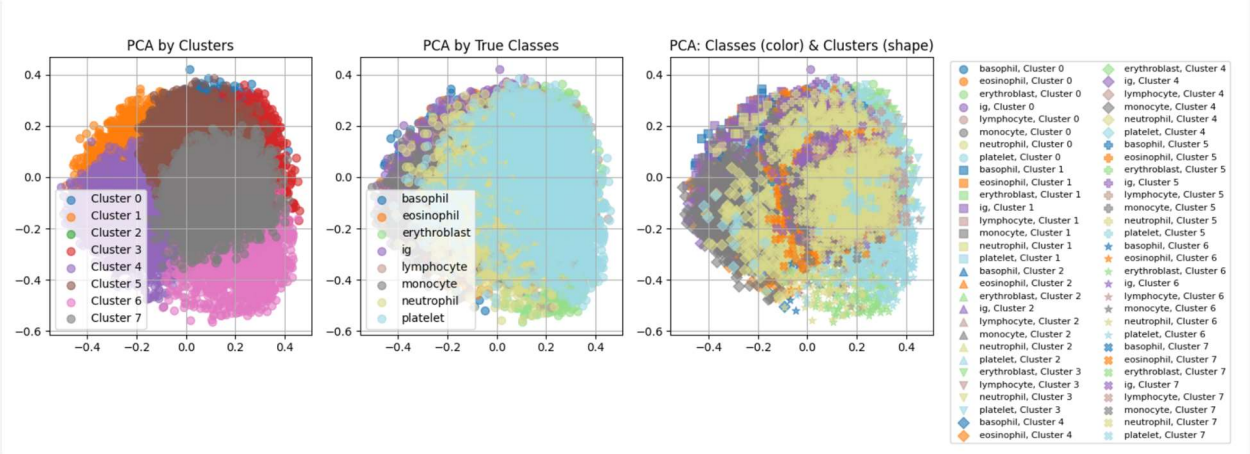


Рисунок 3.11 – Порівняння кластерів з реальними цільовими мітками

Клас IG та lymphocyte переважно зосереджені в кластерах 4 і 5, тоді як erythroblast і monocyte розташовані більш дифузно, що вказує на відносно високу внутрішньокласову варіативність.

На рис. 3.12 видно, що деякі класи переважно групуються в одних і тих же кластерах, як, наприклад, класи ig та eosinophil. Втім, більшість кластерів містить змішані зображення з кількох класів. Це пояснюється високою схожістю між кров'яними клітинами: їхня форма та основні ознаки практично однакові, відмінності полягають переважно у розмірі, розташуванні або дрібних деталях. Проте навіть у таких умовах можна помітити, які клітини мають подібні ознаки та схожі морфологічні характеристики, що дозволяє оцінити структуру та закономірності розподілу клітинних типів.

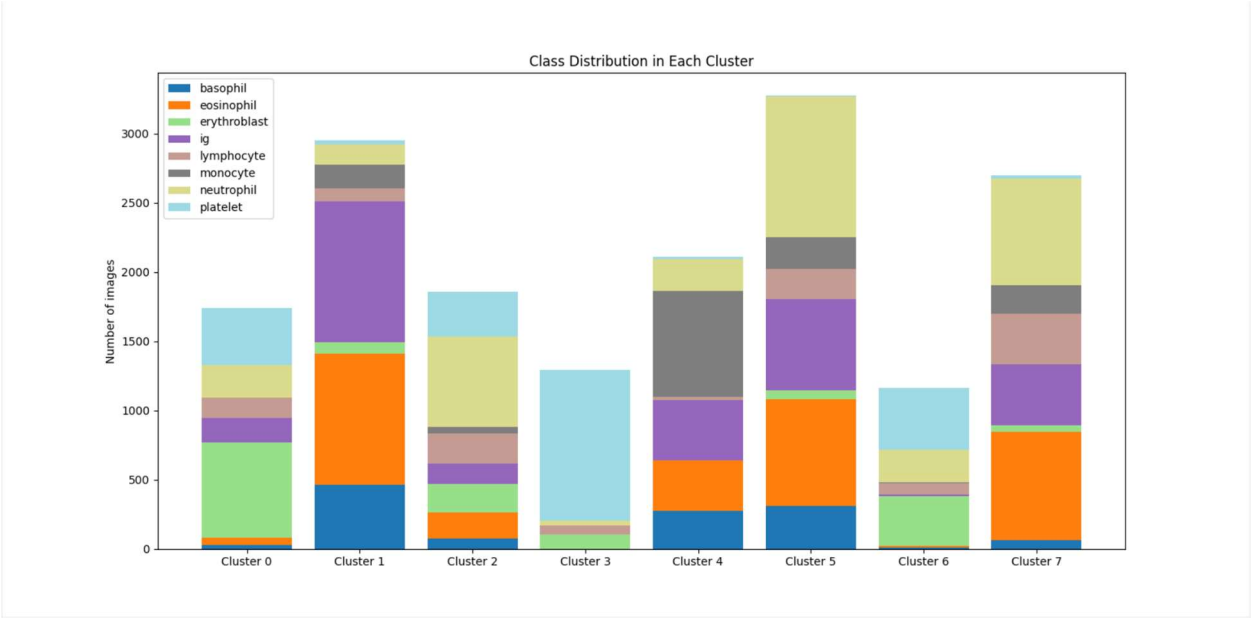


Рисунок 3.12 – Розподіл класів клітин крові в кожному кластері

Найбільш вираженими є Cluster 3 та Cluster 6, які майже повністю складаються з platelet (блакитний колір) та erythroblast (зелений колір) відповідно, що каже про високу однорідність цих груп. Також помітним є Cluster 1, де домінує eosinophil (помаранчевий) та значна частка ig (фіолетовий), утворюючи специфічний профіль розподілу, відмінний від інших.

З іншого боку, Cluster 5 та Cluster 7 виділяються найбільшою загальною кількістю зображень, де переважають neutrophil (пісочний колір) та eosinophil. Зокрема, Cluster 5 містить найвищу концентрацію нейтрофілів серед усіх груп. Водночас Cluster 4 демонструє унікально високу частку monocyte (сірий колір) порівняно з рештою кластерів. Такий розподіл підтверджує, що модель успішно згрупувала клітини зі схожими ознаками, виокремивши як домінуючі класи, так і змішані групи з характерними пропорціями.

Тим не менш, отримані метрики кластеризації виглядають доволі низькими: Silhouette – 0.073, Calinski-Harabasz – 1053.089, Davies-Bouldin – 2.576. Це пояснюється тим, що кров'яні клітини дуже схожі одна на одну: основні ознаки майже однакові, а відмінності спостерігаються лише в розташуванні та розмірі окремих фрагментів клітин [40,41,42].

Для порівняння, на датасеті з овочами (рис. 3.13) кластери виходять більш чіткими, оскільки овочі значно різноманітніші за форму, текстуру та колір. У випадку клітин такі низькі показники є очікуваними та відображають їхню природну схожість.

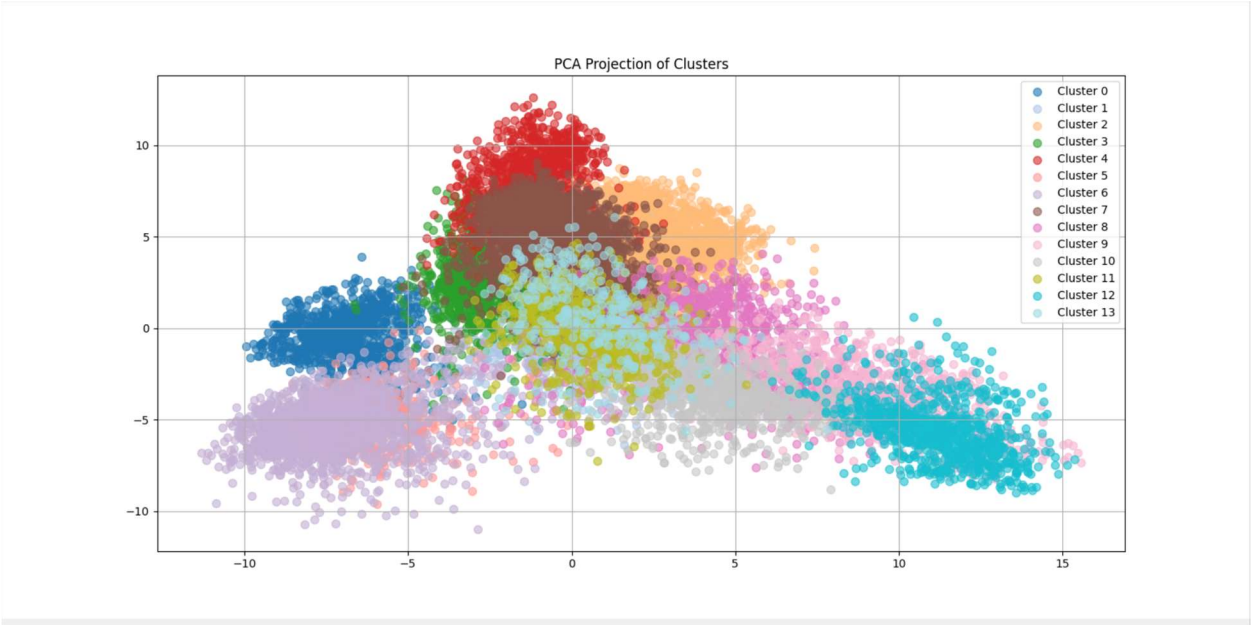


Рисунок 3.13 - PCA проекція кластерів овочей

На рис. 3.14 представлено візуальне порівняння кластеризації овочів та кров'яних клітин (кількість кластерів дорівнює кількості унікалних міток).

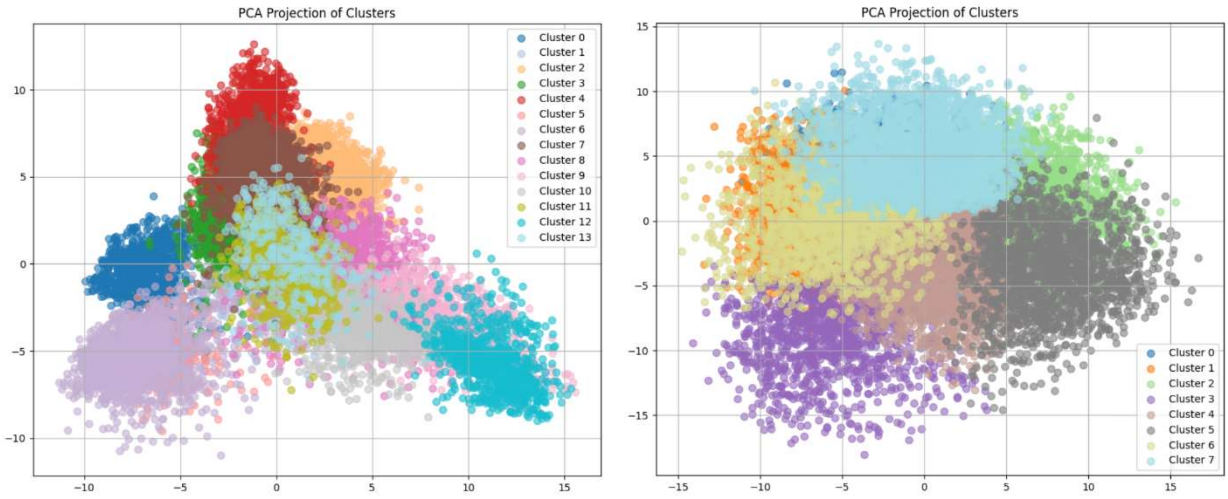


Рисунок 3.14 – Порівняння кластеризації овочів та кров'яних клітин

3.3 Інтерпритація результатів кластеризації кров’яних клітин

Процес кластеризації продемонстрував високу здатність моделі виявляти закономірності в морфології клітин крові (рис. 3.12). Зокрема, Cluster 1, Cluster 4, Cluster 5, та Cluster 7 успішно об’єднали класи monocyte та ig. Це підтверджує ефективність екстрактора ознак: як видно з порівняння на рис. 3.5, ці клітини мають споріднену структуру ядра та цитоплазми, тому їх групування в межах одного кластера є логічним результатом роботи алгоритму.

Цікаво, що у розділі 3.1, під час розробки моделі класифікації, було виявлено, що модель плутала як раз ці ж класи: monocyte та ig. Це підтверджується матрицею помилок (рис. 3.4), де спостерігаються взаємні помилкові передбачення між цими двома класами. Така поведінка моделі пояснюється високою візуальною схожістю даних класів, що ускладнює їх коректне розділення на етапі класифікації. Водночас результати кластеризації показують, що ці об’єкти дійсно мають спільні морфологічні ознаки, оскільки часто потрапляють до одних і тих самих кластерів. Це підтверджує, що модель коректно виділяє приховану подібність між класами на рівні ознак, навіть якщо на етапі класифікації вони трактуються як окремі категорії.

У Cluster 1, Cluster 4 та Cluster 5 спостерігається чітке виділення групи гранулоцитів (basophil та eosinophil). Модель успішно ідентифікувала їхні спільні характеристики — сегментовані ядра та специфічну зернистість, що детально представлено на порівняльному аналізі рис. 3.15. Такий розподіл свідчить про те, що нейронна мережа орієнтується на ключові біологічні маркери, об’єднуючи схожі за текстурою об'єкти в стійкі сегменти.

Отриманий результат також підтверджує, що basophil та eosinophil мають високий рівень візуальної подібності, через що під час кластеризації вони часто формують близько розташовані або частково спільні групи в просторі ознак. Це вказує на здатність моделі виділяти загальні закономірності структури клітин навіть при наявності тонких морфологічних відмінностей.

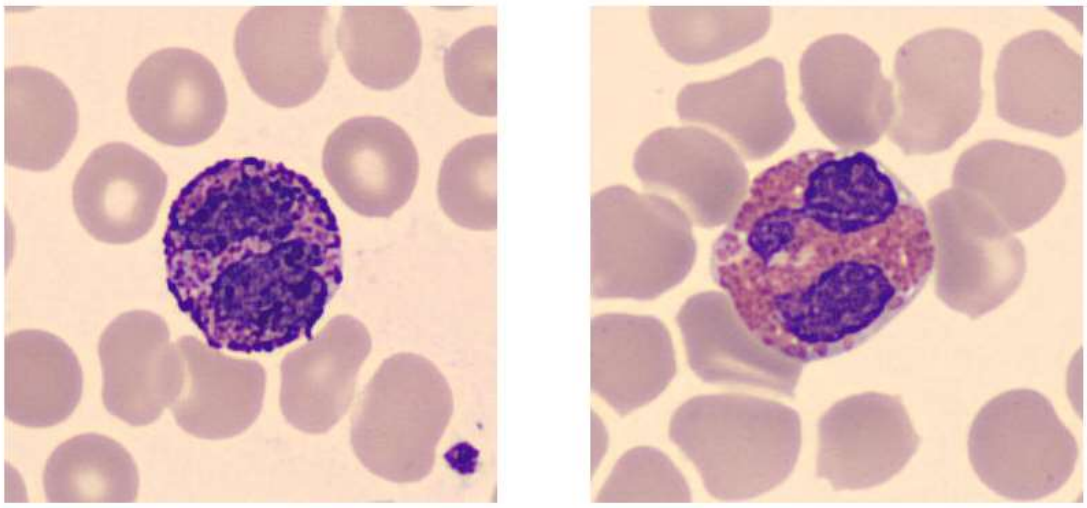


Рисунок 3.15 – Порівняння клітин basophil та eosinophil

Кластери Cluster 0 та Cluster 6 демонструють якісне групування клітин з високим ядерно-цитоплазматичним співвідношенням, таких як erythroblast та lymphocyte. Їхня візуальна близькість, зафіксована на рис. 3.16, дозволяє моделі формувати єдині вектори ознак для об'єктів з масивними круглими ядрами. Це вказує на стабільність роботи алгоритму при обробці різних типів мононуклеарних клітин.

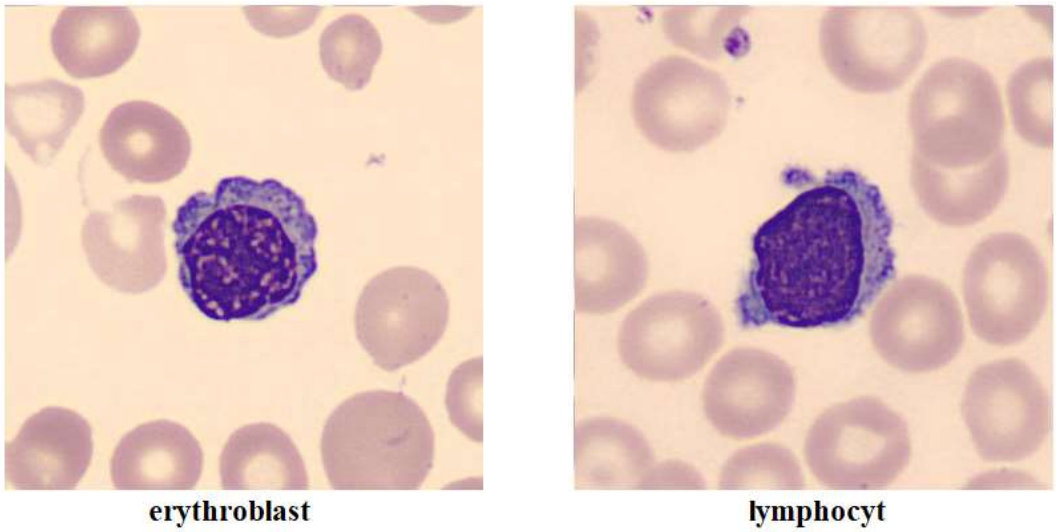


Рисунок 3.16 – Порівняння клітин erythroblast та lymphocyte

Найбільш характерний змішаний результат спостерігається у Cluster 0 та Cluster 6, де переважають зображення erythroblast та platelet. Така кластеризація пояснюється подібністю їхніх візуальних характеристик, зокрема відносно невеликими розмірами об'єктів та схожими геометричними параметрами.

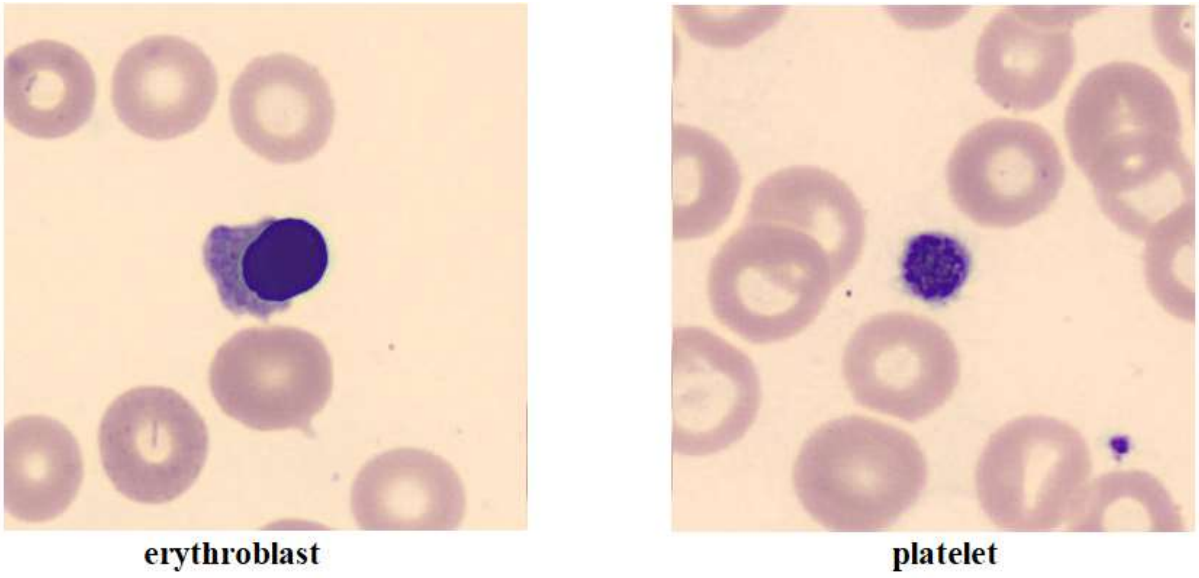


Рисунок 3.16 – Порівняння клітин erythroblast та platelet

Таким чином, результати на рис. 3.12 повністю узгоджуються з візуальними даними та підтверджують адекватність обраного підходу для автоматичного групування медичних зображень.

Висновки до розділу 3

У даному розділі було реалізовано та досліджено процес навчання моделей для задач класифікації та кластеризації зображень кров'яних клітин. На етапі класифікації було побудовано згорткову нейронну мережу (CNN), яка продемонструвала високу точність як на навчальній, так і на тестовій вибірках. Отримані результати підтвердили здатність моделі ефективно виділяти ключові ознаки зображень та узагальнювати отримані закономірності, що забезпечило стабільну якість передбачень.

Аналіз матриці помилок показав, що найбільші труднощі виникають при розмежуванні візуально подібних класів клітин, таких як monocyte та ig, а також окремих гранулоцитів. Це пояснюється високою морфологічною схожістю між певними типами клітин, зокрема за формою ядра, текстурою цитоплазми та загальними геометричними характеристиками. Водночас це також підтверджує, що модель коректно виявляє складні для розрізнення патерни у даних.

На етапі кластеризації, з використанням попередньо навченої моделі MobileNetV2, було отримано векторні представлення зображень та виконано їх групування алгоритмом KMeans. Результати показали, що модель здатна виділяти природні групи клітин на основі їх візуальної схожості без використання міток класів. Зокрема, спостерігається формування кластерів, що відповідають певним морфологічним типам клітин, таким як platelet, erythroblast, а також змішані групи гранулоцитів.

Порівняльний аналіз класифікації та кластеризації показав узгодженість отриманих результатів: у випадках, де класи є візуально подібними, вони також мають тенденцію об'єднуватися в одні й ті самі кластери. Це підтверджує, що модель ефективно навчається на рівні ознак і коректно відображає приховану структуру даних.

Таким чином, у розділі було продемонстровано ефективність використання глибоких нейронних мереж для задач аналізу медичних зображень, а також доведено доцільність комбінованого підходу, що включає як класифікацію, так і кластеризацію для глибшого розуміння структури даних.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСТОСУНКА

4.1 Структура проєкту

Проєкт має чітку модульну архітектуру, організовану за принципом розділення обов'язків. Основний пакет app містить ядро системи, де логіка розподілена між файлами маршрутизації (routes), сервісним шаром для обробки даних (dataset_service.py) та описом моделей бази даних (dataset.py). Для збереження результатів аналізу використовується директорія static/plots, де генеруються графіки розподілу та результати кластеризації, а взаємодія з користувачем реалізована через шаблон home.html.

Технічна частина проєкту доповнена директорією instance, що містить базу даних SQLite (datasets.db), та текою uploaded_datasets для зберігання вхідних файлів. Директорія experiments відокремлює дослідницький код від стабільної версії додатка, що забезпечує чистоту основної кодової бази та зручність супроводу розробки (рис. 4.1).

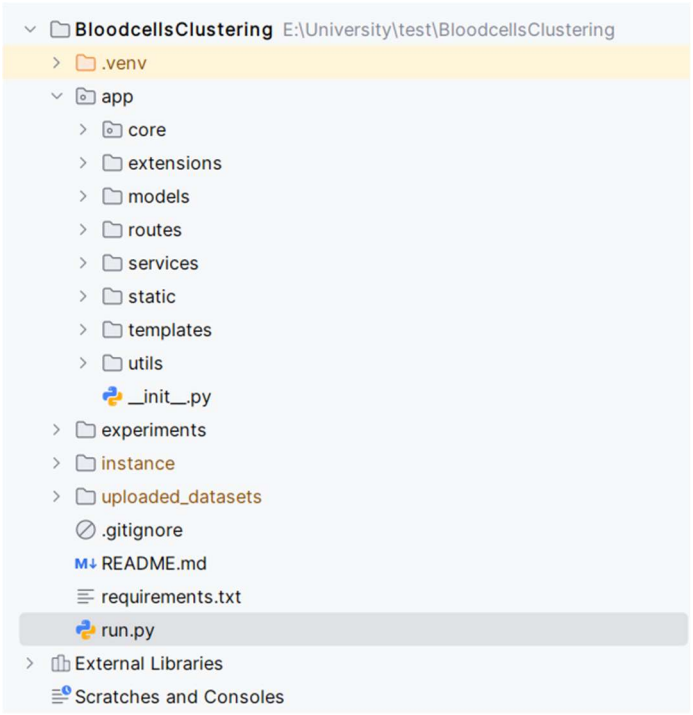


Рисунок 4.1 – Структура проекту

Програмний комплекс побудований на базі мікрофреймворку Flask та використовує архітектуру REST для взаємодії між клієнтською та серверною частинами [43]. Всі функції системи розподілені за трьома основними модулями (Blueprints): модулем загального доступу, модулем керування даними та модулем інтелектуального аналізу [44].

Нижче наведено специфікацію кінцевих точок (endpoints) кожного модуля (табл. 4.1 та табл. 4.2).

Таблиця 4.1 – Модуль керування даними

Метод	Маршрут	Опис
GET	/datasets	Отримання списку всіх завантажених датасетів із бази даних.
POST	/datasets	Ручне створення запису про датасет (передача імені та шляху в JSON).
POST	/datasets/upload	Завантаження ZIP-архіву: зберігає файл, розпаковує його та реєструє в системі.
DELETE	/datasets/<id>	Видалення запису про датасет із бази даних.
GET	/datasets/<id>/preview	Повертає випадкове зображення з датасету для попереднього перегляду.

Таблиця 4.2 – Модуль інтелектуального аналізу

Метод	Маршрут	Опис
POST	/run	Запуск Pipeline: завантаження фото, екстракція ознак (MobileNetV2) та кластеризація (K-Means).
GET	/metrics	Отримання математичних оцінок якості (Silhouette, Calinski-Harabasz, Davies-Bouldin).
GET	/pca	Генерація та отримання 2D-графіка головних компонент (PCA).
GET	/clusters	Отримання візуалізації розподілу даних за знайденими кластерами.
GET	/distribution	Побудова гістограми розподілу вихідних класів по отриманих кластерах.
GET	/compare	Порівняльний аналіз: відображення реальних міток класів проти результатів кластеризації.

Головний модуль (main) містить єдиний маршрут GET /, що є точкою входу в систему. Він відповідає за рендеринг базового інтерфейсу (home.html) та завантаження необхідних статичних ресурсів для ініціалізації вебдодатка.

Програмна реалізація системи базується на принципах об'єктно-орієнтованого програмування та модульної архітектури, що забезпечує гнучкість налаштування алгоритмів кластеризації [45]. Основна логіка розподілена між спеціалізованими класами, де кожен відповідає за конкретний етап обробки даних: від попередньої підготовки зображень та вилучення ознак за допомогою глибокого навчання до математичного аналізу отриманих груп.

Для забезпечення чистоти коду та зручності тестування, аналітичні модулі відокремлені від інфраструктурних компонентів. Зокрема, використання сервісного шару для роботи з базою даних та окремого конвеєра (pipeline) для машинного навчання дозволяє легко масштабувати систему або замінювати окремі моделі без втручання в загальну структуру додатка. Детальний перелік та призначення основних класів системи наведено в табл. 4.3.

Таблиця 4.3 – Основні класи та модулі програмного комплексу

Назва класу / модуля	Призначення	Опис функціоналу
ImageLoader	Завантаження даних	Зчитування зображень з диску, зміна розміру (224x224) та нормалізація для нейромережі.
FeatureExtractor	Аналіз ознак	Використання нейромережі MobileNetV2 для перетворення зображень у вектори ознак (embeddings).
Clusterer	Кластеризація	Реалізація алгоритму K-Means для автоматичного групування векторів ознак у k кластерів.
ClusterMetrics	Оцінка якості	Обчислення математичних метрик: Silhouette, Calinski-Harabasz та Davies-Bouldin.
Plotter	Візуалізація	Зниження розмірності через PCA та побудова графіків розподілу і порівняння результатів.
ClusteringPipeline	Оркестрація	Об'єднання процесів завантаження, екстракції та кластеризації в єдиний конвеєр.
Dataset	Модель даних	Об'єктне представлення таблиці бази даних (SQLAlchemy) для зберігання інформації про датасети.
DatasetService	Сервісний шар	Ізольована логіка для роботи з базою даних (створення, пошук та видалення записів).

Лістинг коду головних модулів IIS наведено в Додатку А, де подано реалізацію основних компонентів системи та пояснення їх функціонування.

На рис. 4.2 представлена діаграма потоків даних (DFD), яка детально описує взаємодію користувача з інтелектуальною системою кластеризації зображень клітин крові [46]. Процес розпочинається з управління датасетами: користувач може завантажувати нові дані, видаляти існуючі або переглядати доступні набори, при цьому всі зміни синхронізуються з базою даних.

Після вибору конкретного датасету та введення параметрів (кількості кластерів), запускається основний конвеєр обробки, що включає етапи підготовки даних, вилучення ознак за допомогою згорткової нейронної мережі (CNN) та безпосередньо кластеризацію алгоритмом K-means.

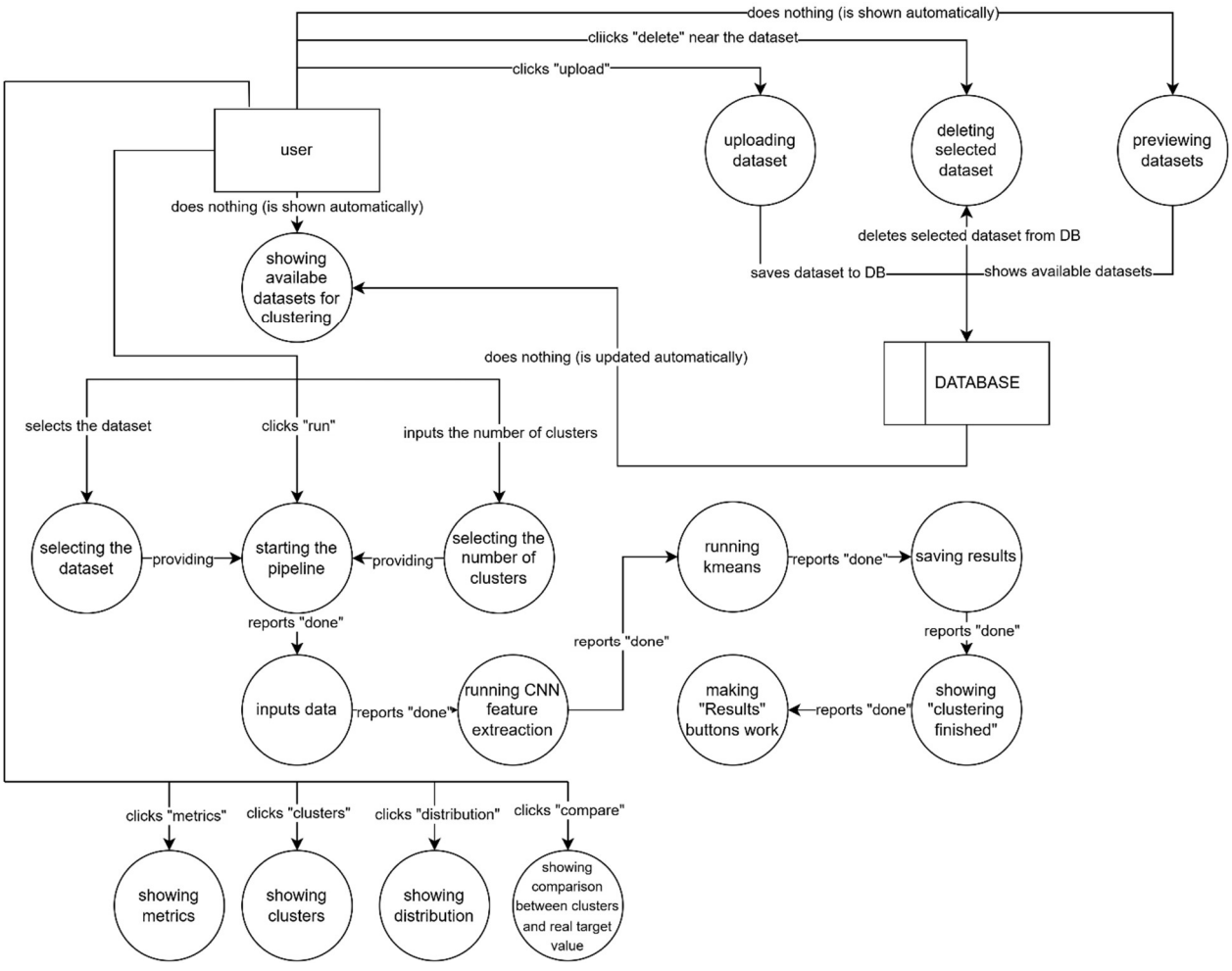


Рисунок 4.2 – Data flow diagram розробленної IIS

Результати обробки автоматично зберігаються в базі даних, після чого система активує інтерфейс для аналізу отриманих значень. Користувачу надається можливість візуалізувати результати через декілька функціональних вікон: перегляд метрик якості, графічне відображення розподілу кластерів та порівняльний аналіз отриманих груп з реальними мітками класів (ground truth) [47]. Така архітектура забезпечує замкнений цикл роботи з даними — від сирих зображень до інтерпретованих результатів кластеризації.

Окремою перевагою архітектури є масштабованість: розроблений конвеєр (pipeline) дозволяє легко адаптувати систему під нові типи клітин або інші медичні датасети без зміни ядра програмного забезпечення.

Збереження результатів у базі даних забезпечує можливість ретроспективного аналізу, що дозволяє порівнювати різні сесії кластеризації та відстежувати динаміку зміни точності моделі при зміні параметрів нейронної мережі.

Реалізований підхід до обробки даних у реальному часі, що відображений у логах сервера, мінімізує часові витрати на рутинні операції, дозволяючи зосередитися на інтерпретації результатів.

Таким чином, програмний продукт являє собою повноцінне робоче середовище для дослідника, яке поєднує в собі потужність методів глибокого навчання, надійність реляційних баз даних та гнучкість інструментів інтерактивної візуалізації. Це створює надійну базу для подальшого впровадження інтелектуальних систем підтримки прийняття рішень у клінічну практику.

4.2 Клієнтська частина

Користувацький інтерфейс реалізовано як динамічну веб-сторінку (Dashboard) на базі HTML5, CSS3 та JS [48]. Дизайн побудований за принципом адаптивної сітки (Grid Layout), що забезпечує коректне відображення компонентів на різних роздільних здатностях.

Візуальна частина включає інтерактивні панелі для керування наборами даних, форми для налаштування параметрів кластеризації (зокрема вибору кількості кластерів k) та блок графічної візуалізації результатів з використанням анімаційних ефектів для покращення користувацького досвіду.

Взаємодія з сервером відбувається асинхронно за допомогою технології AJAX (Fetch API), що дозволяє завантажувати архіви, запускати обчислювальні процеси та отримувати метрики без перезавантаження сторінки.

Логіка клієнтської частини автоматично оновлює список доступних датасетів, підтягує випадкові зображення для попереднього перегляду (Preview) та динамічно формує посилання на згенеровані моделлю графіки й аналітичні звіти (рис. 4.3).

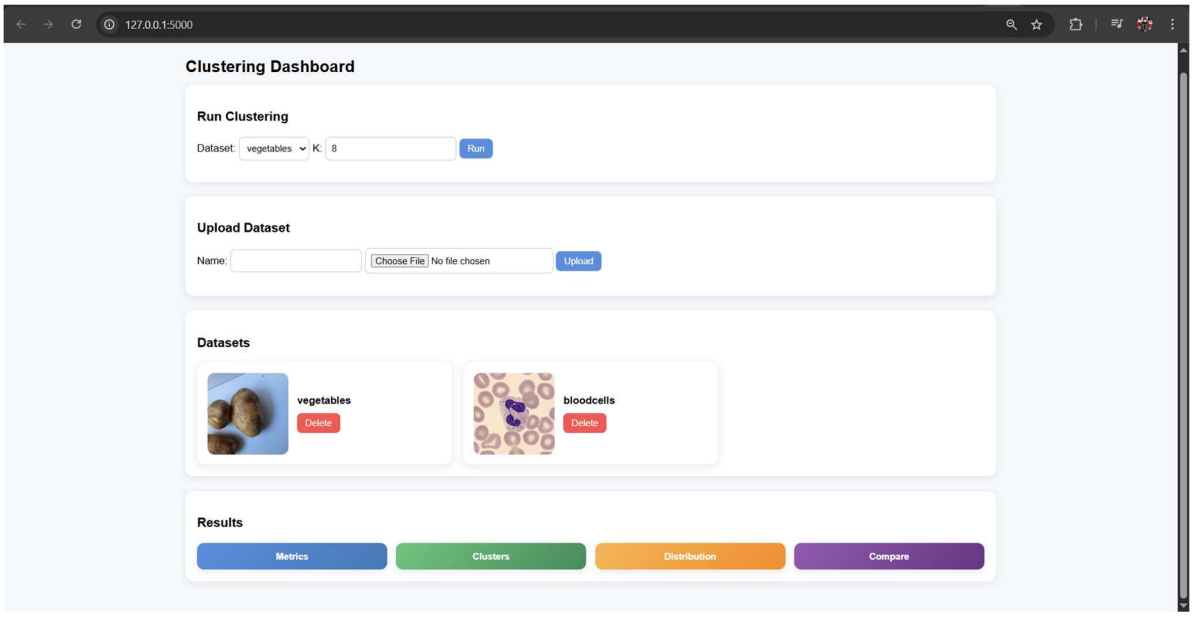


Рисунок 4.3 – Клієнтська частина застосунку

Для покращення користувацького досвіду в інтерфейсі застосовано комплекс графічних ефектів на базі CSS3. Процес взаємодії починається з плавної появи контенту через анімацію, що поступово збільшує прозорість та зміщує блоки вгору. Динамічність елементів керування забезпечується через трансформації, що

дозволяють карткам датасетів реагувати на курсор підйомом та одночасним масштабуванням зображення (рис. 4.4).

```
@keyframes pageFade {
    from { opacity: 0; transform: translateY(20px); }
    to { opacity: 1; transform: translateY(0); }
}

.card:hover {
    transform: translateY(-6px) scale(1.02);
    box-shadow: 0 10px 25px rgba(0,0,0,0.15);
}
```

Рисунок 4.4 – Реалізації анімації та hover-ефектів

Кнопки аналітичних звітів мають складний ефект «світлового бліку», реалізований за допомогою зміщення прихованого псевдоелемента [49]. Це візуально підкреслює активність елемента та створює відгук на дії користувача (рис. 4.5).

```
.result-btn::after {
    content: "";
    position: absolute;
    top: 0;
    left: -120%;
    width: 100%;
    height: 100%;
    background: rgba(255,255,255,0.25);
    transform: skewX(-25deg);
    transition: 0.5s;
}

.result-btn:hover::after {
    left: 120%;
}
```

Рисунок 4.5 – Код реалізації візуального ефекту кнопок

Програмна інтерактивність базується на асинхронній обробці подій за допомогою JS. Замість традиційного перезавантаження сторінок система

використовує обробники подій для форм завантаження та запуску обчислень, що дозволяє динамічно оновлювати дані в інтерфейсі (рис. 4.6).

```
document.getElementById("runForm").onsubmit = async (e) => {
    e.preventDefault();
    const res = await fetch("/run", {
        method: "POST",
        body: new FormData(e.target)
    });
    const data = await res.json();
    document.getElementById("runMsg").innerText = data.message || data.error;
};
```

Рисунок 4.6 – Асинхронний запуск Pipeline

Такий підхід забезпечує безперервність робочого процесу, де користувач може миттєво бачити результат роботи алгоритмів або зміни у списку доступних даних без втрати контексту поточної сторінки [50].

Важливою особливістю реалізованого інтерфейсу є повна відсутність статичних елементів у списку доступних даних. Весь контент генерується "на льоту" після отримання відповіді від сервера. Це реалізовано за допомогою ітерації по масиву об'єктів JSON та маніпуляції з методами createElement, що дозволяє автоматично створювати структуру карток, додавати до них зображення-прев'ю та призначати обробники подій для кнопок видалення.

Такий підхід забезпечує синхронізацію стану інтерфейсу з базою даних у реальному часі. Окрім візуальних карток, цей же скрипт паралельно оновлює випадаючий список (<select>) у формі запуску кластеризації. Це гарантує, що користувач завжди бачить актуальний перелік наборів даних, доступних для інтелектуального аналізу, без потреби в ручному оновленні сторінки браузера.

4.3 Тестування роботи систем

Для перевірки стабільності системи було проведено тестування основних функцій API через Flask-сервер. Логи підтверджують успішне завантаження

датасетів (статус 201), коректну роботу ендпоінтів для отримання метрик і візуалізації результатів (статус 200), а також справну функцію видалення даних. Відсутність помилок під час запитів свідчить про надійну взаємодію між клієнтською та серверною частинами.

Окрему увагу приділено модулю обробки даних на базі TensorFlow. Консоль фіксує успішне розпізнавання 34 184 зображень, розподілених за 9 класами, та стабільний прогрес виконання обчислень. Система ефективно використовує ресурси процесора, забезпечуючи стабільну обробку великих масивів даних без збоїв у роботі алгоритмів (рис. 4.7).

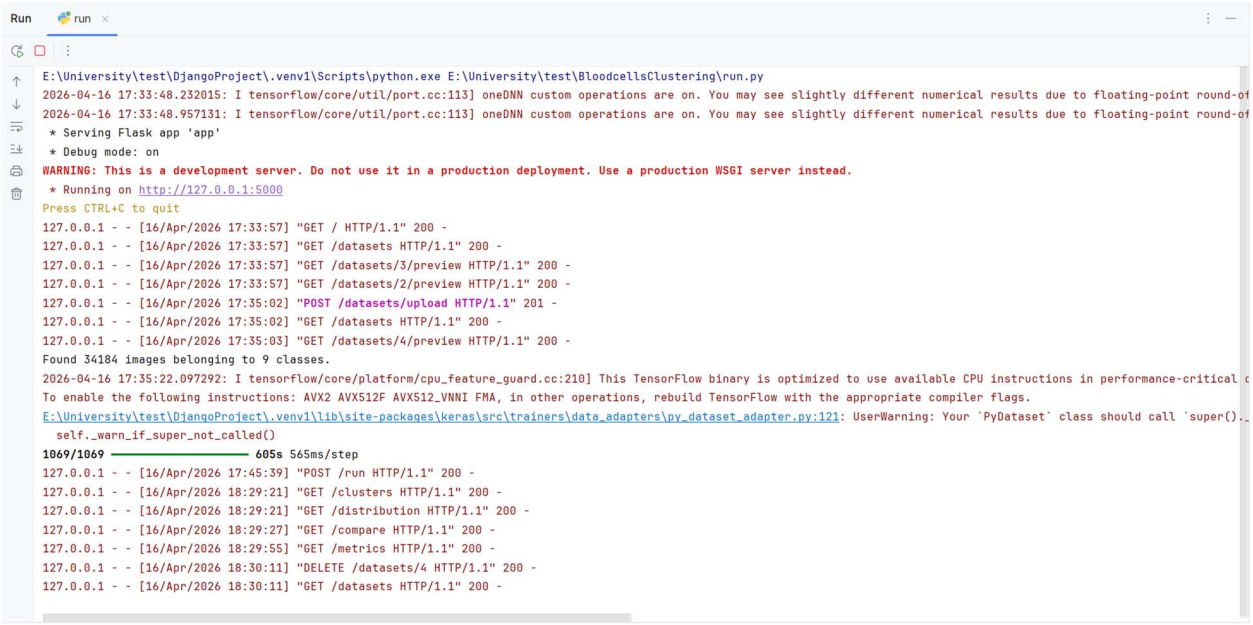


Рисунок 4.7 – Тестування енд поїнтів

Додатково було проведено комплексне тестування працездатності системи, що включало перевірку коректності обробки запитів, стійкості до некоректних вхідних даних та стабільності роботи при багаторазових зверненнях до API. Усі основні сценарії використання (створення, завантаження, перегляд та видалення датасетів) пройшли тестування без критичних помилок, що підтверджує готовність системи до базового експлуатаційного навантаження.

Під час тестування застосовувалися різні типи перевірок, зокрема функціональне тестування (перевірка відповідності заявленій логіці роботи), інтеграційне тестування (взаємодія між API, базою даних та сервісними модулями), навантажувальне тестування (оцінка стабільності при великій кількості запитів), а також тестування обробки помилок і некоректних вхідних даних. Такий підхід дозволив комплексно оцінити надійність і стійкість системи в різних умовах використання [51-54].

Під час проєктування та реалізації системи було враховано вимоги до кібербезпеки та захисту інформації, оскільки оброблювані дані можуть містити чутливу або структурно важливу інформацію. У зв'язку з цим було розроблено архітектуру, орієнтовану на мінімізацію ризиків несанкціонованого доступу та виключення можливості виконання шкідливих запитів на рівні бази даних.

Зокрема, система реалізована з використанням ORM-рівня (SQLAlchemy), що забезпечує параметризовану обробку запитів і запобігає SQL-ін'єкціям [55,56]. Додатково передбачено перевірку вхідних даних, обмеження типів файлів під час завантаження та контроль коректності HTTP-запитів. Такий підхід дозволяє підвищити загальний рівень безпеки системи та забезпечити її стійкість до базових кіберзагроз (рис. 4.8).

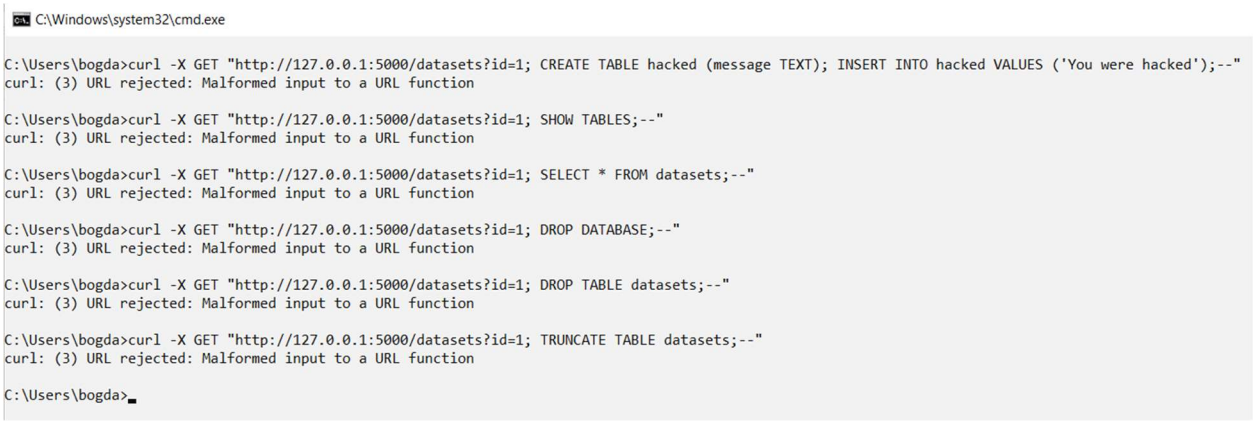


Рисунок 4.8 – Спроба SQL injection

Крім перевірки SQL-ін'єкцій, у межах роботи було проведено додаткове тестування кібербезпекових аспектів системи. Зокрема, здійснювалась перевірка на path traversal-атаки, які пов'язані зі спробами доступу до заборонених директорій через модифікацію шляхів до файлів [57,58]. Також було протестовано стійкість до некоректних та спеціально сформованих HTTP-запитів, включаючи передачу нестандартних символів та некоректних структур JSON. Усі сценарії були коректно оброблені системою без виникнення критичних помилок.

Додатково було виконано тестування на XSS-подібні вставки (Cross-Site Scripting) у контексті обробки та повернення даних API, що дозволило переконатися у відсутності інтерпретації шкідливих скрипт-конструкцій у відповідях сервера [59,60]. Також перевірено стійкість до перевантаження сервера (DoS-рівня навантаження) шляхом багаторазових послідовних запитів до основних ендпоінтів, при цьому система зберігала стабільну роботу без збоїв [61-64].

Окрему увагу приділено тестуванню безпечної обробки файлів, зокрема завантаженню ZIP-архівів. Перевірялися сценарії некоректних архівів, змінених розширень та потенційно шкідливої структури файлів. Система коректно відхиляла або безпечно обробляла такі випадки, забезпечуючи ізольоване збереження даних у визначеній директорії (рис. 4.9).

Рисунок 4.9 – Спроба File spoofing

У результаті проведених перевірок встановлено, що система демонструє базову стійкість до основних класів веб-атак, зокрема SQL-ін'єкцій, path traversal, XSS-подібних вставок та перевантажувальних запитів.

Висновки до розділу 4

У даному розділі було детально описано структуру та принципи реалізації програмного комплексу. Розглянуто архітектуру проєкту, яка побудована за модульним принципом із чітким розділенням серверної та клієнтської частин. Окремі компоненти системи (маршрутизація, сервісний шар, моделі даних та модулі машинного навчання) організовані таким чином, щоб забезпечити простоту супроводу, масштабованість та можливість подальшого розширення функціоналу.

Також у розділі наведено опис REST API, що реалізує основні функції системи, включаючи керування датасетами, запуск конвеєра обробки даних, отримання метрик якості кластеризації та візуалізацію результатів. Описані кінцеві точки демонструють логічне групування функціональності за окремими модулями, що підвищує структурованість та зрозумілість системи.

Окремо розглянуто клієнтську частину застосунку, яка реалізує інтерактивний вебінтерфейс із використанням сучасних вебтехнологій. Взаємодія з сервером відбувається асинхронно, що забезпечує динамічне оновлення даних без перезавантаження сторінки та покращує користувацький досвід. Використання графічних ефектів і динамічного формування елементів інтерфейсу дозволило реалізувати зручну та інтуїтивну систему керування даними.

У межах розділу також описано проведене тестування системи, яке включало перевірку основних функцій API, стабільності роботи алгоритмів машинного навчання, а також оцінку кібербезпекових аспектів. Проведені випробування підтвердили коректність роботи всіх основних компонентів системи, її стабільність під час обробки даних та базовий рівень захищеності від типових вебзагроз.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи було розроблено програмне забезпечення для аналізу та кластеризації медичних зображень клітин крові з використанням методів глибокого навчання та алгоритмів машинного навчання. Основний акцент у системі зроблено на автоматичному виділенні ознак зображень та їх подальшому групуванні без використання розмічених класів, що дозволяє досліджувати внутрішню структуру даних і виявляти приховані закономірності між різними типами клітин.

Мета роботи досягнута. Поставлені завдання виконані, а саме:

– досліджено предметну область аналізу медичних зображень клітин крові, розглянуто сучасні підходи до їх обробки, кластеризації та візуального аналізу, а також проаналізовано існуючі рішення та їх обмеження;

– сформовано вимоги до програмного забезпечення, включаючи можливість роботи з великими наборами даних, гнучке налаштування параметрів кластеризації та підтримку візуального аналізу результатів;

– розроблено архітектуру системи кластеризації на основі попередньо навченої моделі MobileNetV2, яка використовується для автоматичного виділення глибоких ознак зображень та формування їх компактних векторних представлень;

– реалізовано алгоритм K-Means для групування отриманих ознак з можливістю вибору кількості кластерів користувачем;

– організовано процес оцінювання результатів кластеризації з використанням метрик якості (Silhouette, Calinski-Harabasz, Davies-Bouldin), а також проведено візуалізацію результатів у просторі зниженої розмірності за допомогою PCA;

– розроблено серверну частину системи на базі Flask, яка реалізує REST API для завантаження датасетів, запуску процесу кластеризації, отримання метрик та генерації візуалізацій;

– реалізовано клієнтську частину вебзастосунку, що забезпечує інтерактивну взаємодію користувача з системою, включаючи завантаження даних, запуск обчислень та перегляд результатів у вигляді графіків і таблиць;

– інтегровано базу даних SQLite для збереження інформації про завантажені датасети;

– реалізовано базові механізми захисту системи, зокрема контроль вхідних даних, захист від SQL-ін’єкцій, та перевірку файлів при завантаженні;

– проведено тестування програмного забезпечення, включаючи функціональне, інтеграційне та навантажувальне тестування, що підтвердило стабільну роботу системи та коректність виконання основних сценаріїв використання.

У ході виконання роботи було досліджено предметну область морфологічного аналізу клітин крові, яка є важливою задачею медичної діагностики. Особливістю даної задачі є висока візуальна подібність між окремими типами клітин, що ускладнює їх автоматичне групування. Саме тому використання глибоких нейронних мереж для виділення ознак дозволяє суттєво покращити якість кластеризації.

Запропонована архітектура системи дозволяє ефективно перетворювати зображення у векторний простір ознак та виконувати їх подальше групування, що забезпечує виявлення природних структур у даних без попереднього розмічування. Додатково система підтримує візуальний аналіз результатів, що дозволяє інтерпретувати отримані кластери та оцінювати їх відповідність морфологічним особливостям клітин.

Таким чином, розроблена система є комплексним інструментом для дослідження структури медичних зображень, який поєднує методи глибокого навчання, кластеризації, візуалізації та збереження результатів, забезпечуючи повний цикл аналізу даних від завантаження до інтерпретації.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. K-Means vs K-Means++ Clustering Algorithm. URL: https://www.geeksforgeeks.org/machine-learning/k-means-vs-k-means-clustering-algorithm/ (date of request: 05.06.2026).

2. Principal Component Analysis (PCA) Explained Visually with Zero Math. URL: https://towardsdatascience.com/principal-component-analysis-pca-explained-visually-with-zero-math-1cbf392b9e7d/ (date of request: 05.06.2026).

3. Convolutional Neural Networks (CNN) — Architecture Explained. URL: https://medium.com/@draj0718/convolutional-neural-networks-cnn-architectures-explained-716fb197b243 (date of request: 05.06.2026).

4. Frontiers in Plant Science. URL: https://www.researchgate.net/figure/It-shows-the-overall-architecture-of-MobileNet-V3-Small-It-includes-a-lightweight-neural_fig4_376364618 (date of request: 05.06.2026).

5. MedTransCluster: Transfer learning for deep medical image clustering. URL: https://www.sciencedirect.com/science/article/pii/S2666521224000061 (date of request: 05.06.2026).

6. DeepMCAT: Large-Scale Deep Clustering for Medical Image Categorization. URL: https://arxiv.org/abs/2110.00109 (date of request: 05.06.2026).

7. A deep dictionary clustering approach for unsupervised image retrieval using convolutional sparse coding. URL: https://www.nature.com/articles/s41598-025-32982-z (date of request: 05.06.2026).

8. Semi-Supervised Medical Image Classification Combined with Unsupervised Deep Clustering. URL: https://www.mdpi.com/2076-3417/13/9/5520 (date of request: 05.06.2026).

9. A comprehensive survey of image clustering based on deep learning. URL: https://www.sciencedirect.com/science/article/abs/pii/S0031320325012531 (date of request: 05.06.2026).

10. Blood Cells Image Dataset. URL: https://www.kaggle.com/datasets/unclesamulus/blood-cells-image-dataset (date of request: 05.06.2026).

11. Section 5 (Week 5) TensorFlow and PyTorch. URL: https://cs230.stanford.edu/section/5/ (date of request: 05.06.2026).

12. KMEANS AND KERNEL KMEANS A comparative study of classical and kernel kmeans for data clustering. URL: https://www.researchgate.net/figure/Classic-and-kernel-kmeans-on-circle-moons-classification-and-iris-datasets-Moon_fig1_337932249 (date of request: 05.06.2026).

13. Python Data Visualization with Matplotlib – Part 1. URL: https://towardsdatascience.com/visualizations-with-matplotlib-part-1-c9651008b6b8/ (date of request: 05.06.2026).

14. Array programming with NumPy. URL: https://www.nature.com/articles/s41586-020-2649-2/figures/1 (date of request: 05.06.2026).

15. Implementation of database using python flask framework: college database management system. URL: https://www.researchgate.net/publication/339091903_Implementation_of_database_using_python_flask_framework_college_database_management_system (date of request: 05.06.2026).

16. DEVELOPING RESEARCH MANAGEMENT TOOLS USING PYTHON AND FLASK. URL: https://ijerst.org/index.php/ijerst/article/view/1505 (date of request: 05.06.2026).

17. A Performance Evaluation of Flask (WSGI) and Fastapi (ASGI) Under High-Concurrency Workloads. URL: https://www.researchgate.net/publication/404177352_A_Performance_Evaluation_of_Flask_WSGI_and_Fastapi_ASGI_Under_High-Concurrency_Workloads (date of request: 05.06.2026).

18. Activity 19: Research on Python SQLAlchemy. URL: https://joshuanato.hashnode.dev/activity-19-research-on-python-sqlalchemy (date of request: 05.06.2026).

19. Object-Role Modeling (ORM/NIAM). URL: https://www.academia.edu/50609722/Object_Role_Modeling_ORM_NIAM_ (date of request: 05.06.2026).

20. SQL for Data Science: Data Cleaning, Wrangling and Analytics with Relational Databases URL: https://www.researchgate.net/publication/346806515_SQL_for_Data_Science_Data_Cleaning_Wrangling_and_Analytics_with_Relational_Databases (date of request: 05.06.2026).

21. An Observational Study on Flask Web Framework Questions on Stack Overflow (SO). URL: https://www.researchgate.net/publication/387258232_An_Observational_Study_on_Flask_Web_Framework_Questions_on_Stack_Overflow_SO (date of request: 05.06.2026).

22. The R Language: An Engine for Bioinformatics and Data Science. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC9148156/ (date of request: 05.06.2026).

23. Scientific Data Analysis using Jython Scripting and Java. URL: https://link.springer.com/book/10.1007/978-1-84996-287-2 (date of request: 05.06.2026).

24. AHP Method to Support Decision Making for Sustainability. URL: https://www.researchgate.net/publication/365578042_AHP_Method_to_Support_Decision_Making_for_Sustainability (date of request: 05.06.2026).

25. Analytic Hierarchy Process. URL: https://www.sciencedirect.com/topics/computer-science/analytic-hierarchy-process (date of request: 05.06.2026).

26. State-of-the-art on analytic hierarchy process in the last 40 years: Literature review based on Latent Dirichlet Allocation topic modelling. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC9140269/ (date of request: 05.06.2026).

27. The Theory of Ratio Scale Estimation: Saaty's Analytic Hierarchy Proces. URL: https://www.researchgate.net/publication/227445821_The_Theory_of_Ratio_Scale_Estimation_Saaty's_Analytic_Hierarchy_Process (date of request: 05.06.2026).

28. Role of Consistency and Random Index in Analytic Hierarchy Process—A New Measure. https://www.researchgate.net/publication/340481466_Role_of_Consistency_and_Random_Index_in_Analytic_Hierarchy_Process-A_New_Measure (date of request: 05.06.2026).

29. Data exploration and analysis with Jupyter notebooks URL: https://www.researchgate.net/publication/346022216_Data_exploration_and_analysis_with_Jupyter_notebooks (date of request: 05.06.2026).

30. Google Colab: The Free Cloud Platform Powering Machine Learning. URL: https://www.researchgate.net/publication/379000829_Google_Colab_The_Free_Cloud_Platform_Powering_Machine_Learning (date of request: 05.06.2026).

31. Management of scientific information with Google Drive. URL: https://www.researchgate.net/publication/256929567_Management_of_scientific_information_with_Google_Drive (date of request: 05.06.2026).

32. Data Augmentation: A Multi-Perspective Survey on Data, Methods, and Applications. URL: https://www.sciencedirect.com/org/science/article/pii/S1546221825009580 (date of request: 05.06.2026).

33. 2D Convolutional neural network with four Conv2D layers followed by Global Average Pooling and Softmax as activation function. URL: https://www.researchgate.net/figure/D-Convolutional-neural-network-with-four-Conv2D-layers-followed-by-Global-Average-Pooling_fig2_365460898 (date of request: 05.06.2026).

34. Batch Normalization. URL: https://www.sciencedirect.com/topics/computer-science/batch-normalization (date of request: 05.06.2026).

35. Feature Extraction Layers with their various Conv2D -Maxpooling2D stage configurations. URL: https://www.researchgate.net/figure/Feature-Extraction-Layers-with-their-various-Conv2D-Maxpooling2D-stage-configurations_fig5_362301361 (date of request: 05.06.2026).

36. Flattening Layer Pruning in Convolutional Neural Networks. URL: https://www.semanticscholar.org/paper/Flattening-Layer-Pruning-in-Convolutional-Neural-Jeczmionek-Kowalski/474b97314d5e145085aef8b573026c6fc3d678a1 (date of request: 05.06.2026).

37. Analysis on the Dropout Effect in Convolutional Neural Networks. URL: https://www.researchgate.net/publication/314521313_Analysis_on_the_Dropout_Effect_in_Convolutional_Neural_Networks (date of request: 05.06.2026).

38. Implementation of Approximate Softmax Function for Neural Network. URL: https://www.researchgate.net/publication/398266837_Implementation_of_Approximate_Softmax_Function_for_Neural_Network (date of request: 05.06.2026).

39. Confusion Matrices: A Unified Theory. URL: https://www.researchgate.net/publication/386206668_Confusion_Matrices_A_Unified_Theory (date of request: 05.06.2026).

40. When Does the Silhouette Score Work? A Comprehensive Study in Network Clustering. URL: https://www.researchgate.net/publication/399276072_When_Does_the_Silhouette_Score_Work_A_Comprehensive_Study_in_Network_Clustering (date of request: 05.06.2026).

41. Calinski-Harabasz Index for K-Means Clustering Evaluation in Python. URL: https://towardsdatascience.com/calinski-harabasz-index-for-k-means-clustering-evaluation-using-python-4fefeeb2988e/ (date of request: 05.06.2026).

42. Davies-Bouldin Index for K-Means Clustering Evaluation in Python. URL: https://towardsdatascience.com/davies-bouldin-index-for-k-means-clustering-evaluation-in-python-57f66da15cd/ (date of request: 05.06.2026).

43. REST API: the basics. URL: https://towardsdatascience.com/rest-api-the-basics-d91859537c9d/ (date of request: 05.06.2026).

44. Understanding HTTP Methods: GET, POST, DELETE, PUT, PATCH, and When to Use Them. URL: https://medium.com/@vaibhavtiwari.945/understanding-http-methods-get-post-delete-put-patch-and-when-to-use-them-c1ef6949c671 (date of request: 05.06.2026).

45. Comparative Performance Analysis of Object-Oriented Programming and Data-Oriented Programming in TensorFlow. https://www.researchgate.net/publication/402352527_Comparative_Performance_Analysis_of_Object-Oriented_Programming_and_Data-Oriented_Programming_in_TensorFlow?_tp=eyJjb250ZXh0Ijp7ImZpcnN0UGFnZSI6InB1YmxpY2F0aW9uIiwicGFnZSI6InNjaWVuY2VUb3BpYyJ9fQ (date of request: 05.06.2026).

46. Data Flow Diagram. https://www.sciencedirect.com/topics/computer-science/data-flow-diagram (date of request: 05.06.2026).

47. Ground Truth Information. https://www.sciencedirect.com/topics/computer-science/ground-truth-information (date of request: 05.06.2026).

48. The Role of HTML5 and CSS3 in Creating Optimized Graphic Prototype Websites and Application Interfaces. URL: https://www.researchgate.net/publication/387238322_The_Role_of_HTML5_and_CSS3_in_Creating_Optimized_Graphic_Prototype_Websites_and_Application_Interfaces (date of request: 05.06.2026).

49. Comparative Analysis of the Performance of CSS Animation Methods (Transition and Animation) under High DOM Load. URL: https://www.researchgate.net/publication/401062345_Comparative_Analysis_of_the_Performance_of_CSS_Animation_Methods_Transition_and_Animation_under_High_DOM_Load (date of request: 05.06.2026).

50. JAVASCRIPT ASYNCHRONOUS PROGRAMMING. URL: https://www.researchgate.net/publication/334510290_JAVASCRIPT_ASYNCHRONOUS_PROGRAMMING (date of request: 05.06.2026).

51. What is functional testing? https://www.ibm.com/think/topics/functional-testing (date of request: 05.06.2026).

52. Integration testing. URL: https://www.atlassian.com/continuous-delivery/software-testing#Integration%20testing (date of request: 05.06.2026).

53. What Is Load Testing and Why It Matters? URL: https://medium.com/@cevherd/what-is-load-testing-and-why-it-matters-ae93057bc08e (date of request: 05.06.2026).

54. Testing for Improper Error Handling. URL: https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/08-Testing_for_Error_Handling/01-Testing_For_Improper_Error_Handling

55. A Survey of SQL Injection Attack Detection and Prevention. URL: https://www.researchgate.net/publication/276496047_A_Survey_of_SQL_Injection_Attack_Detection_and_Prevention (date of request: 05.06.2026).

56. Journal of Cyber Security and Risk Auditing. URL: https://jcsra.thestap.com/archives/volume-2025-4/68bfbc1dd5383abb29265895 (date of request: 05.06.2026).

57. Eradicating the Unseen: Detecting, Exploiting, and Remediating a Path Traversal Vulnerability across GitHub. URL: https://arxiv.org/abs/2505.20186 (date of request: 05.06.2026).

58. Deep Dive into Directory Traversal and File Inclusion Attacks leads to Privilege Escalation. URL: https://www.researchgate.net/publication/352390325_Deep_Dive_into_Directory_Traversal_and_File_Inclusion_Attacks_leads_to_Privilege_Escalation (date of request: 05.06.2026).

59. Cross Site Scripting Attacks (XSS): A Review. URL: https://www.researchgate.net/publication/397096348_Cross_Site_Scripting_Attacks_XSS_A_Review (date of request: 05.06.2026).

60. Twenty-two years since revealing cross-site scripting attacks: a systematic mapping and a comprehensive survey. URL: https://arxiv.org/abs/2205.08425 (date of request: 05.06.2026).

61. DDoS attack detection and defense techniques in software defined networks: A survey. URL: https://www.sciencedirect.com/science/article/abs/pii/S1574013726000304 (date of request: 05.06.2026).

62. DDoS Attack Detection: Strategies, Techniques, and Future Directions. URL: https://www.researchgate.net/publication/382393041_DDoS_Attack_Detection_Strategies_Techniques_and_Future_Directions (date of request: 05.06.2026).

63. Analysing Cloud DDoS Attacks Using Supervised Machine Learning. URL: https://www.deepscienceresearch.com/dsr/catalog/book/71 (date of request: 05.06.2026).

64. A Comprehensive Analysis of DDoS attacks based on DNS. URL: https://iopscience.iop.org/article/10.1088/1742-6596/2024/1/012027 (date of request: 05.06.2026).

ДОДАТОК А

Лістинг програмної реалізації головних модулів IIS

Спочатку імпортуються базові бібліотеки для роботи з файлами, чисельними обчисленнями та візуалізацією. Підключаються моделі глибокого навчання (MobileNetV2), інструменти кластеризації (KMeans), зменшення розмірності (PCA) та метрики оцінки якості кластеризації (рис. А.1).

```
import os
import numpy as np
import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt

from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications.mobilenet_v2 import MobileNetV2, preprocess_input
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.metrics import (
    silhouette_score,
    calinski_harabasz_score,
    davies_bouldin_score
)

```

Рисунок А.1 – Імпорт бібліотек

Клас ImageLoader призначений для завантаження набору зображень із директорій та підготовки їх до подальшої обробки у нейронній мережі. У методі load використовується ImageDataGenerator з попередньою обробкою preprocess_input, після чого функція flow_from_directory автоматично зчитує зображення з заданого шляху, масштабує їх до розміру 224×224, формує пакети розміру batch_size та не перемішує дані (shuffle=False) для збереження порядку. У результаті метод повертає генератор зображень для моделі та список імен файлів, що дозволяє надалі співвідносити отримані ознаки з вихідними зображеннями (рис. А.2).

```
19  class ImageLoader:  1 usage  Bohdan Somriakov
20      def __init__(self, dataset_path):  Bohdan Somriakov
21          self.dataset_path = dataset_path
22
23      def load(self, batch_size=32, target_size=(224, 224)):  1 usage  Bohdan Somriakov
24          datagen = ImageDataGenerator(preprocessing_function=preprocess_input)
25          gen = datagen.flow_from_directory(
26              self.dataset_path,
27              target_size=target_size,
28              batch_size=batch_size,
29              class_mode=None,
30              shuffle=False
31          )
32          return gen, gen.filenames
33
```

Рисунок А.2 – Клас ImageLoader

Клас Clusterer призначений для виконання кластеризації векторів ознак, отриманих із зображень. У конструкторі задається кількість кластерів k, яка визначає, на скільки груп будуть поділені дані. Метод cluster використовує алгоритм KMeans з фіксованим параметром random_state=42 для відтворюваності результатів і виконує розбиття переданих ознак features на задану кількість кластерів, повертаючи мітки кластерів для кожного об'єкта (рис. А.3).

```
36  class FeatureExtractor:  1 usage  Bohdan Somriakov
37      def __init__(self, input_shape=(224, 224, 3)):  Bohdan Somriakov
38          self.model = MobileNetV2(
39              weights="imagenet",
40              include_top=False,
41              pooling="avg",
42              input_shape=input_shape
43          )
44
45      def extract(self, generator):  1 usage  Bohdan Somriakov
46          return self.model.predict(generator, verbose=1)
47
```

Рисунок А.3 – Клас FeatureExtractor

Клас ClusterMetrics призначений для оцінки якості результатів кластеризації на основі отриманих векторів ознак та міток кластерів. Статичний метод compute обчислює три основні метрики: коефіцієнт силуету (Silhouette score), індекс Калінаскі–Харабаша (Calinski-Harabasz) та індекс Девіса–Булдіна (Davies-Bouldin).

При цьому значення силуету розраховується лише у випадку наявності більше ніж одного кластера, інакше повертається NaN. У результаті метод формує та повертає словник, що містить числові значення цих метрик, які використовуються для аналізу щільності, відокремленості та загальної якості кластеризації (рис. А.4).

```
class ClusterMetrics:  2 usages  Bohdan Somriakov
    @staticmethod  1 usage  Bohdan Somriakov
    def compute(features, labels):
        sil = silhouette_score(features, labels) if len(np.unique(labels)) > 1 else float("nan")
        ch = calinski_harabasz_score(features, labels)
        db = davies_bouldin_score(features, labels)

        return {
            "Silhouette": float(sil),
            "Calinski-Harabasz": float(ch),
            "Davies-Bouldin": float(db)
        }
```

Рисунок А.4 – Клас ClusterMetrics

Клас ClusteringPipeline реалізує повний конвеєр обробки даних для задачі кластеризації зображень. У конструкторі задаються шлях до датасету path та кількість кластерів k, які визначають вхідні параметри системи.

Метод run послідовно виконує всі етапи обробки: спочатку завантажує зображення за допомогою ImageLoader, після чого передає їх у FeatureExtractor для отримання векторів ознак за допомогою згорткової нейронної мережі MobileNetV2.

Далі отримані ознаки кластеризуються алгоритмом KMeans через клас Clusterer. У результаті метод повертає вектори ознак, мітки кластерів та список імен файлів, що дозволяє аналізувати та візуалізувати результати.

```
167  class ClusteringPipeline:  2 usages  Bohdan Somriakov
168      def __init__(self, path, k):  Bohdan Somriakov
169          self.path = path
170          self.k = k
171
172      def run(self):  2 usages (1 dynamic)  Bohdan Somriakov
173          loader = ImageLoader(self.path)
174          gen, files = loader.load()
175
176          features = FeatureExtractor().extract(gen)
177          labels = Clusterer(self.k).cluster(features)
178
179          return features, labels, files
```

Рисунок А.5 – Клас ClusteringPipeline

Кожен із реалізованих класів виконує окрему важливу роль у загальній системі кластеризації зображень. ImageLoader забезпечує завантаження та попередню підготовку даних, FeatureExtractor відповідає за виділення інформативних ознак за допомогою згорткової нейронної мережі MobileNetV2, а Clusterer виконує групування отриманих векторів ознак методом KMeans.

Клас ClusterMetrics дозволяє оцінити якість кластеризації за стандартними метриками, а ClusteringPipeline об'єднує всі етапи в єдиний автоматизований процес. Така модульна структура забезпечує гнучкість, масштабованість та зручність використання системи для кластеризації зображень.