Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

# КВАЛІФІКАЦІЙНА РОБОТА
# Пояснювальна записка

рівень вищої освіти другий (магістерський)

**ДОСЛІДЖЕННЯ МЕТОДІВ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ З ВИКОРИСТАННЯМ ПАРАМЕТРІВ КЛАСИФІКАЦІЙНОЇ ВАГОМОСТІ ДАНИХ**
(тема)

Виконав:
студент 2 курсу, групи ІНФм-22-1
Кулик О.В.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник проф. Гороховатський В.О.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2024 р.

# ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Кулику Олександру Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження методів класифікації зображень з використанням параметрів класифікаційної вагомості даних

затверджена наказом по університету від 3 листопада 2023 року № 1280Ст

2. Термін подання студентом роботи до екзаменаційної комісії 23 грудня 2023 р.

3. Вихідні дані до роботи теорія формування і використання для класифікації ключових точок зображення, дескриптор ORB, способи визначення інформативності ключових точок, апарат відстаней на множині багатовимірних векторів, теоретичні основи структурних методів класифікації зображень, способи скорочення і стиснення описів у формі множини векторів, програмне забезпечення для аналізу зображень, середовище програмування Python.

4. Перелік питань, що потрібно опрацювати в роботі

1. Аналіз існуючих структурних методів класифікації зображень.
2. Використання математичної моделі для обчислення значущості ключових ознак зображення.
3. Застосування редукції до множини дескрипторів на основі критеріїв значущості.
4. Здійснення програмної реалізації на тестовому наборі зображень з використанням групи геометричних перетворень.
5. Аналіз результатів працездатності і точності класифікаторів з редукованим описом.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) постановка задачі, база еталонних зображень, зображення з виділеними дескрипторами ORB, математична модель оцінювання інформативності, результати аналізу тестування і оцінювання точності розробленого класифікатора зображень.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

| Найменування розділу | Консультант (посада, прізвище, ім'я, по батькові) | Позначка консультанта про виконання розділу | |
|---|---|---|---|
| | | підпис | дата |
| | | | |

**КАЛЕНДАРНИЙ ПЛАН**

| № з/п | Назва етапів роботи | Терміни виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Отримання завдання на кваліфікаційну роботу | 03.11.2023 | виконано |
| 2 | Аналіз завдання, підбір літератури | 03.11.23-05.11.23 | виконано |
| 3 | Аналіз літератури з досліджуваної проблеми | 06.11.23-11.11.23 | виконано |
| 4 | Аналіз технічних засобів | 12.11.23-13.11.23 | виконано |
| 5 | Розробка методу | 14.11.23-01.12.23 | виконано |
| 6 | Програмна реалізація | 02.12.23-20.12.23 | виконано |
| 7 | Оформлення пояснювальної записки | 20.12.23-25.12.23 | виконано |
| 8 | Перевірка на плагіат | 26.12.2023 | виконано |
| 9 | Рецензування | 27.12.2023 | виконано |
| 10 | Підготовка презентації та доповіді | 28.12.2023 | виконано |
| 11 | Занесення роботи в електронний архів | 01.01.2024 | виконано |
| 12 | Попередній захист кваліфікаційної роботи | 02.01.2024 | виконано |

Дата видачі завдання 3 листопада 2023 р.

Студент _____
(підпис)

Керівник роботи _____ проф.Гороховатський В.О.
(підпис) (посада, прізвище, ініціали)

# РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 86 с., 8 табл., 20 рис., 50 джерел.

КОМП'ЮТЕРНИЙ ЗІР, РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ, СТРУКТУРНІ МЕТОДИ КЛАСИФІКАЦІЇ, ДЕСКРИПТОР ORB, РЕДУКЦІЯ ОЗНАК, ІНФОРМАТИВНІСТЬ.

Об'єктом дослідження є структурні методи класифікації зображень.

Метою дослідження є удосконалення результативності структурних методів класифікації зображень шляхом впровадження редукції описів даних з використанням параметрів класифікаційної значущості.

Використано дескриптори ORB, апарат теорії множин і векторного простору, метричні моделі для визначення релевантності множин багатовимірних векторів, елементи теорії ймовірності та статистичного аналізу. Досліджено математичні моделі для визначення значущості ознак у наборах даних. Розроблено апарат для редукції описів даних еталонних зображень на основі критерію найбільш інформативних дескрипторів.

У результаті дослідження програмно реалізовано модель класифікації зображень та оцінено її ефективність.

COMPUTER VISION, IMAGE RECOGNITION, STRUCTURAL CLASSIFICATION METHODS, ORB DESCRIPTOR, FEATURE REDUCTION, INFORMATIVENESS.

The object of the research is a structural image classification methods.

The aim of the research is to improve performance of a structural image classification methods by introducing image dataset reduction based on image feature importance criteria.

The ORB descriptors, the apparatus of set theory and vector space, metric models for determining the relevance of sets of multidimensional vectors, elements of probability theory and statistical analysis were used. Mathematical models for evaluating dataset features importance were researched. Apparatus for image description reduction based on most informative descriptors was implemented.

As a result, software implementation of the image classification system was implemented and its effectiveness was evaluated.

# ЗМІСТ

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

SIFT – Scale-Invariant Feature Transform (масштабонезалежне перетворення ознак)

SURF – Speeded Up Robust Features (прискорені надійні ознаки)

ORB – Oriented FAST and Rotated BRIEF (орієнтований FAST і обернутий BRIEF)

КТ – ключова точка

FAST – Features from Accelerated Segment Test (ознаки прискореного сегментного тесту)

MSER – Maximally Stable Extremal Regions (максимально стабільні екстремумні області)

BRISK – Binary Robust Invariant Scalable Keypoints (двійкові надійні інваріантні масштабовані ключові точки)

CENSURE – Scale-Invariant Center-Surround Detector (масштабонезалежний центрально-об'ємний детектор)

JPEG – Joint Photographic Experts Group (об’єднана експертна група з фотографій)

FREAK – Fast Retina Keypoints (швидкі ключові точки сітківки)

CNN – Convolutional Neural Network (згорткова нейронна мережа)

PCA – Principal Component Analysis (метод головних компонент)

CPU – Central Processing Unit (центральний процесор)

GPU – Graphics Processing Unit (графічний процесор)

TPU – Tensor Processing Unit (тензорний блок обробки)

# ВСТУП

Наявність цифрових зображень у нашому повсякденному житті в поєднанні з експоненціальним зростанням обсягу візуальних даних в Інтернеті підкреслює критичну важливість систем оброблення та класифікації зображень. Від автономних транспортних засобів, які розпізнають дорожні знаки, до систем охорони здоров'я, які інтерпретують медичні сканування, здатність автоматично класифікувати зображення на основі їх вмісту є одним з основних завдань сучасних технологій [1-6]. Класифікація зображень лежить в основі цієї можливості трансформації.

Класифікація зображень є фундаментальним завданням комп'ютерного зору, що дозволяє розпізнавати та класифікувати зображення на основі їх вмісту. Одним з найбільш поширених сучасних підходів до класифікації є структурні методи [1-6]. Ці підходи використовують безпосередньо вимірювані представлення зображень, відомі як дескриптори, для визначення подібності між зображеннями та попередньо визначеними категоріями чи класами [7].

Основна ідея класифікації зображень в структурних методах полягає в тому, щоб представити зображення як набір відмінних візуальних особливостей або ключових точок. Ці ключові точки виявляються на зображенні за певним алгоритмом і описуються за допомогою математичних векторів (дескрипторів). До найбільш поширених алгоритмів для виявлення ключових точок на зображенні можна віднести SIFT (Scale-Invariant Feature Transform), SURF (Speeded-Up Robust Features) і ORB (Oriented FAST і Rotated BRIEF) [7-10]. Дескриптори містять важливу інформацію про локальну структуру та зовнішній вигляд об'єктів на зображеннях. Далі ці обчислені дескриптори формують певну базу, по якій відбувається пошук та класифікація вхідних зображень, використовуючи міру релевантності.

Однак ці дескриптори можуть бути багатовимірними та обчислювально дорогими, що може створити проблеми з точки зору зберігання, швидкодії

оброблення та використання об’ємів пам’яті. Редукція описів зображень є перспективною технікою, спрямованою на вирішення цих проблем, зберігаючи або навіть покращуючи продуктивність систем комп’ютерного зору [11-13].

Основною метою редукції описів зображень є зменшення кількості дескрипторів без суттєвої втрати у класифікаційної інформативності даних опису. Іншими словами, вона спрямована на те, щоб зберегти найбільш характерну та цінну інформацію, відкидаючи зайві або менш важливі дані. Цей підхід має кілька ключових переваг [11], такі як:

– швидкодія (зменшення обчислювального об’єму роботи призведе до швидшого часу виконання таких завдань, як класифікація зображень, виявлення об’єктів і пошук);

– використання пам’яті (редуковані описи зображень потребують менше пам’яті, що робить їх більш придатними для середовищ з обмеженими ресурсами, таких як мобільні пристрої чи вбудовані системи);

– стійкість до шумів (зменшення розмірності може допомогти відфільтрувати шум і нерелевантні деталі з дескрипторів, потенційно покращуючи стійкість систем комп’ютерного зору до варіацій і шуму в даних).

Актуальність дослідження кваліфікаційної роботи полягає в удосконаленні структурних методів класифікації зображень шляхом впровадження редукції описів даних з використанням параметрів класифікаційної вагомості. Важливим інструментом оцінювання розроблених модифікацій методів є імітаційне комп’ютерне моделювання на реальних зображеннях.

# 1 ОСНОВИ СТРУКТУРНИХ МЕТОДІВ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ

## 1.1 Формалізація методів класифікації за множиною дескрипторів

Структурні методи класифікації зображень зарекомендували себе як швидкий та надійний спосіб розпізнавання візуальних об’єктів у сучасних системах комп’ютерного зору. Дані методи базуються на таких поняттях, як ключові точки зображень (КТ) та дескриптори ключових точок [4-7, 14-16]. Перевага цих методів полягає в їхній здатності фіксувати та представляти візуальну інформацію подібно до того, як люди сприймають і розпізнають об’єкти та шаблони. Вони є потужним інструментом у комп’ютерному зорі для завдань, які вимагають ідентифікації та зіставлення відмінних рис зображень, імітуючи певні аспекти людського зорового сприйняття.

Ключові точки зображення також відомі як ознаки зображення або точки інтересу – це конкретні місця або елементи на зображенні, які виділяються своєю унікальністю та відмінністю (рис. 1.1). Вони служать опорними точками для порівняння та вирівнювання різних зображень.

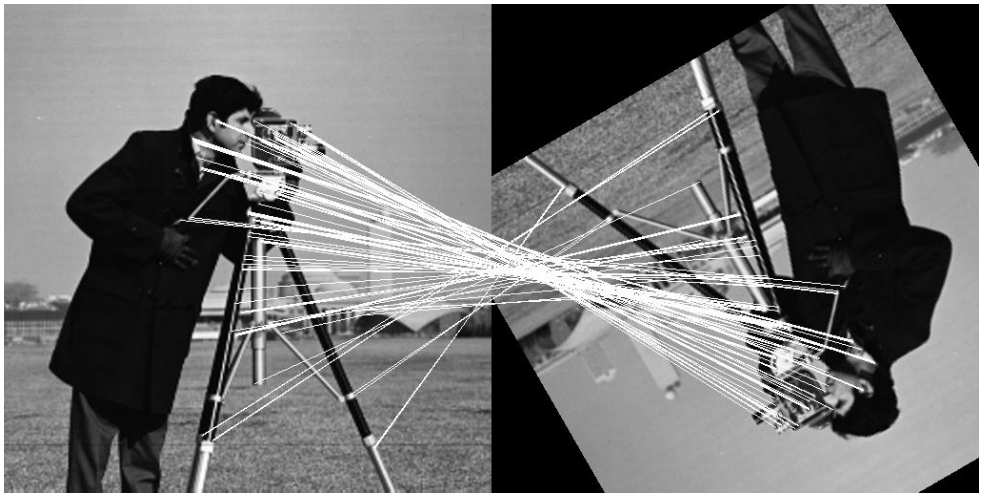

Рисунок 1.1 – Виділені ключові точки та їх відповідність

Для виявлення ключових точок використовуються так звані детектори КТ. Існує багато різних алгоритмів, але загалом вони працюють за схожим принципом [7, 10].

Крок 1. Проводиться попередня обробка зображення для покращення його якості та зменшення шуму. Це може включати такі операції, як зміна розміру, перетворення простору кольорів та розмиття Гауса для покращення процесу виявлення.

Крок 2. Виявляються точки інтересу для визначення потенційних ключових точок на зображенні. Існує безліч алгоритмів, які вирішують дану задачу. До найпоширеніших можна віднести наступні [17-19]:

– Гаррісів виявляч кутів (Harris Corner Detection): визначає кути або з’єднання на зображенні на основі локальних змін інтенсивності;

– FAST (Features from Accelerated Segment Test): виявляє ключові точки, враховуючи кругові шаблони інтенсивності пікселів;

– DoG (Difference of Gaussians): виявляє плями або області зі значними коливаннями інтенсивності в різних масштабах;

– MSER (Maximally Stable Extremal Regions): визначає області зі стабільною інтенсивності при різних порогах;

– виявлення плям (Blob Detection): виявляє області зі специфічними характеристиками, часто використовуючи такі методи, як лапласіан Гауса (LoG) або детектори плям на основі Гессе.

Крок 3. Обирається масштаб: багато алгоритмів виявлення точок інтересу створюють ключові точки в кількох масштабах, щоб зробити їх масштабно-інваріантними. Це може включати аналіз піраміди зображення або застосування методів простору масштабів.

Крок 4. Призначається напрямок кожній КТ, щоб зробити їх інваріантними щодо обертання. Зазвичай це робиться шляхом розгляду інформації про локальний градієнт навколо ключової точки. КТ часто призначається орієнтація на основі домінуючого напрямку градієнта.

Крок 5. Подавляються не-максимуми (Non-Maximum Suppression, NMS), щоб зменшити надмірність і вибрати лише найбільш помітні ключові точки. Цей крок гарантує збереження лише найважливіших КТ.

Крок 6. Встановлюється поріг та проводиться фільтрація на основі таких критеріїв, як сила відгуку або інші відповідні характеристики, щоб додатково покращити набір виявлених ключових точок.

Крок 7. Локалізуються ключові точки, як правило, шляхом субпіксельного уточнення для підвищення їх точності.

Результатом цих кроків є набір ключових точок, які представляють характерні та повторювані особливості зображення. Для подальшого використання в різних задачах комп'ютерного зору, таких як зіставлення зображень та розпізнавання об'єктів, КТ треба представити у вигляді дескрипторів. Дескриптор зображення – це числове представлення локальної області або об'єкта в зображенні. Дескриптори фіксують візуальні характеристики області зображення, дозволяючи ефективно порівнювати та зіставляти їх між собою.

Для того, щоб успішно використовувати дескриптори у системах комп'ютерного зору, вони мають відповідати ряду властивостей [1-6]:

– відмінність: хороший дескриптор зображення має бути характерним, тобто фіксувати унікальну та відмітну інформацію про область, яку він представляє. Ця відмінність допомагає гарантувати, що для схожих областей зображення буде згенеровано подібні дескриптори, тоді як для несхожих – різні;

– стійкість: дескриптори зображення мають бути стійкими до різних трансформацій і варіацій зображення, таких як зміни умов освітлення та точки огляду. Надійний дескриптор повинен забезпечувати подібні представлення для того самого об'єкта або сцени, навіть якщо ці варіації присутні;

– інваріантність: це властивість дескриптора залишатися узгодженим за певних перетворень. Наприклад, масштабно-інваріантні дескриптори

нечутливі до змін масштабу, тоді як поворотно-інваріантні дескриптори залишаються послідовними, незважаючи на зміни в орієнтації. Властивості інваріантності необхідні для зіставлення та розпізнавання об’єктів на зображеннях реального світу;

– компактність: дескриптори зображень мають бути компактними, тобто вони мають бути представлені відносно невеликою кількістю значень або функцій. Компактні дескриптори більш ефективні для зберігання та обчислень;

– ефективність: ефективне обчислення дескрипторів має вирішальне значення при роботи з великими обсягами даних та в програмах реального часу;

– унікальність: дескриптори мають бути унікальними для різних областей зображення. Ця властивість допомагає гарантувати, що дві різні області на зображенні створюють різні дескриптори, полегшуючи їх розрізнення;

– сумісність: дескриптори зображення мають бути сумісні з алгоритмами зіставлення або розпізнавання, що використовуються в певній програмі. Для різних завдань можуть знадобитися різні типи дескрипторів, і сумісність із вибраним алгоритмом є важливою для успішних результатів;

– масштабованість: дескриптори повинні бути масштабованими для роботи з зображеннями різних розмірів і роздільної здатності. Вони мають забезпечувати узгоджене представлення незалежно від розмірів зображення;

– стійкість до шуму: дескриптори зображення мають бути стійкими до шуму та змін, які можуть виникнути під час отримання зображення. Зашумлені області на зображенні не повинні створювати дескриптори, що значно відрізняються від чистих аналогів.

Засновуючись на теоретичній інформації, можна побудувати математичну модель класифікатора зображень структурним методом. Основою структурного розпізнавання зображень є формування бази даних еталонів у вигляді структурних описів:

$$E = \{E^i\}_{i=1}^{I}, \tag{1.1}$$

де $i$ – номер еталону;

$I$ – загальна кількість еталонів;

$E^i$ – структурний опис і-го еталону.

Кожен опис еталону фактично являє собою матрицю $N \times M$, де $N$ – це кількість дескрипторів у кожному описі, а $M$ – це кількість елементів у векторі дескриптора. Для задачі розпізнавання об'єкту $O$ необхідно множину описів даного об'єкта $O = \{o^j\}$ за певним правилом відобразити у скінченній множині номерів еталонів $\{1, 2, \ldots, I\}$ шляхом віднесення до одного з елементів бази даних еталонів $E = \{E^i\}_{i=1}^{I}$. Це правило можна відобразити у вигляді наступної формули:

$$g: O \to M, M = \{1,2, \ldots, I\}. \tag{1.2}$$

Головною задачею структурного розпізнавання зображень є побудова ефективного правила $g$, ефективність якого може залежати від багатьох факторів, таких як кількість виділених КТ на еталонних зображеннях, вибір дескриптора тощо. Критерієм ефективності класифікації можна вважати точність класифікатора, тобто кількість правильно класифікованих об'єктів по відношенню до їх загальної кількості.

Базовий підхід до структурної класифікації відображений в наступних кроках [2, 3].

Крок 1. Побудова структурного опису $O$ для вхідного об'єкту.

Крок 2. Обчислення значень релевантності $d_i(E^i, O)$ для опису $O$ на множині еталонів $E^1, E^2, \ldots, E^I$.

Крок 3. Оптимізація на множині значень $d_i(E^i, O)$, $i$ = 1, 2, …, $I$, і визначення індексу класу еталона $v$ по формулам:

$$v = \arg\min_{i=1,\dots,I} d_i(E^i, O), \tag{1.3}$$

$$v = \arg\max_{i=1,\dots,I} d_i(E^i, O). \tag{1.4}$$

Формула (1.3) застосовується, якщо критерієм релевантності є відстань між описами, (1.4) – коли критерієм релевантності є подібність.

У роботі [20] був проведений аналіз великої кількості популярних детекторів ключових точок на зображенні, розроблених за останні 30 років: BRISK, CENSURE, FAST, MSER, ORB, SIFT та SURF (рис. 1.2). Окрім звичайних умов детектори також перевірялись в умовах зміни освітлення, розмиття, обертання, масштабування, зміни точки огляду, експозиції, стиснення JPEG, комбінованого масштабування та обертання, комбінованих зміни точки огляду, масштабування та обертання на великому наборі тестових даних. За результатами тестування дескриптор ORB за комбінованими показниками швидкодії та якості виявлення КТ отримав високі бали від авторів.

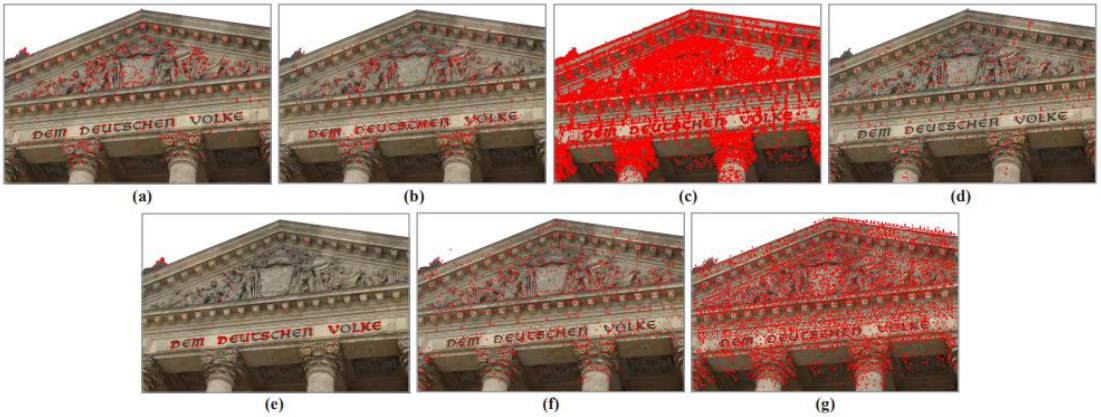


Рисунок 1.2 – Порівняння виділених КТ на зображенні різними детекторами

У роботах [21, 22] автори дослідили продуктивність дескриптора ORB у порівнянні з іншими популярними алгоритмами пошуку ключових точок на зображенні для задач розпізнавання зображень у реальному часі: SIFT, SUFT,

FREAK. Алгоритм ORB виявився найшвидшим алгоритмом серед усіх протестованих, при цьому в задачах класифікації по згенерованим дескрипторам він також проявив себе на високому рівні, поступаючись лише алгоритму SIFT. Результати дослідження наведені на рисунку 1.3.

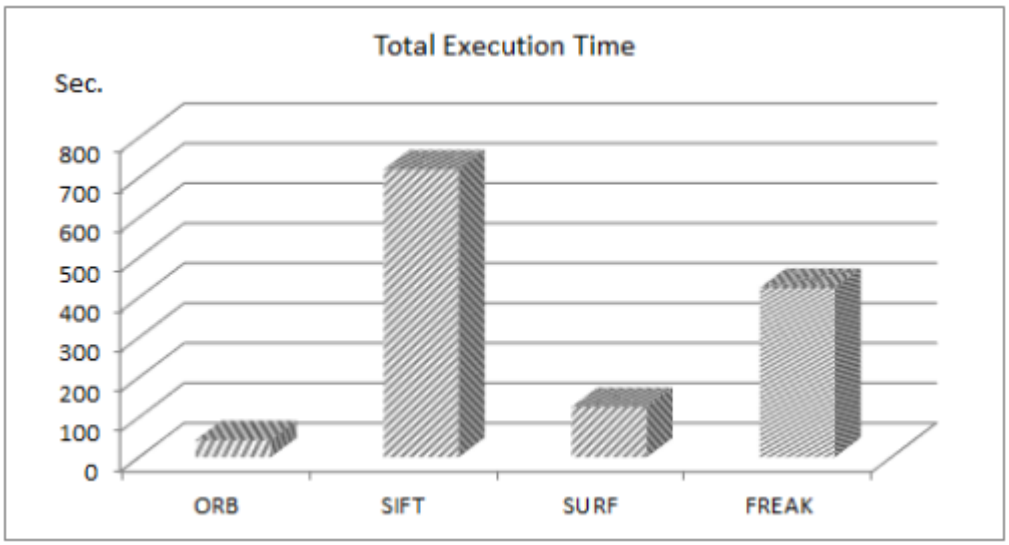


Рисунок 1.3 – Порівняння швидкодії алгоритмів для пошуку КТ

Спираючись на результати досліджень, в даній роботі для подальших експериментів буде використано бінарний детектор ORB.

## 1.2 Аналіз детектору ORB

ORB (Oriented FAST and Rotated BRIEF) належить до класу бінарних дескрипторів. На відміну від традиційних дескрипторів, які використовують безперервні значення для зберігання ознак, бінарні дескриптори представляють ключові точки зображення за допомогою бінарних векторів, які складаються з 0 і 1. Цей спосіб представлення має декілька переваг [10, 13]:

– компактність: бінарні дескриптори використовують значно менше місця для зберігання порівняно з дескрипторами з плаваючою точкою. Це

особливо вигідно для програм з обмеженим обсягом пам’яті та при роботі з великими наборами даних;

– ефективність: бінарні дескриптори є більш ефективними для обчислення, оскільки для порівняння дескрипторів можна використовувати побітові операції, які швидко виконуються на сучасних процесорах. Ця ефективність має вирішальне значення для застосунків, що працюють у режимі реального часу та мають обмежені ресурси;

– стійкість до шуму: двійкові дескриптори можуть бути більш стійкими до шуму зображення порівняно з дескрипторами з плаваючою точкою, оскільки вони зосереджуються на наявності чи відсутності певних шаблонів зображення, а не на безперервних значеннях інтенсивності.

ORB було розроблено в лабораторіях OpenCV в 2011 році як ефективну та життєздатну альтернативу SIFT [23] і SURF [24]. Основна причина створення алгоритму ORB полягала в тому, що SIFT і SURF є запатентованими алгоритмами, однак ORB можна використовувати безкоштовно.

ORB виконує задачу виявлення ключових точок так само добре, як SIFT (і краще, ніж SURF), але майже на два порядки швидше. ORB базується на добре відомому детекторі ключових точок FAST (Features from Accelerated Segment Test) [25] і дескрипторі BRIEF [26]. Обидва ці підходи привабливі своєю високою швидкодією і високою точністю. Основні зміни, які були внесені у алгоритмі ORB, наступні [27]:

– до FAST додано компонент швидкого та точного розрахунку напрямку;

– більш ефективне обчислення дескрипторів BRIEF;

– аналіз дисперсії і кореляції дескрипторів BRIEF;

– можливість декореляції дескрипторів BRIEF, дотримуючись інваріантності повороту, що показує кращу продуктивність при застосуванні методу найближчого сусіда.

На рисунку 1.4 зображено загальну схему роботи алгоритму ORB.

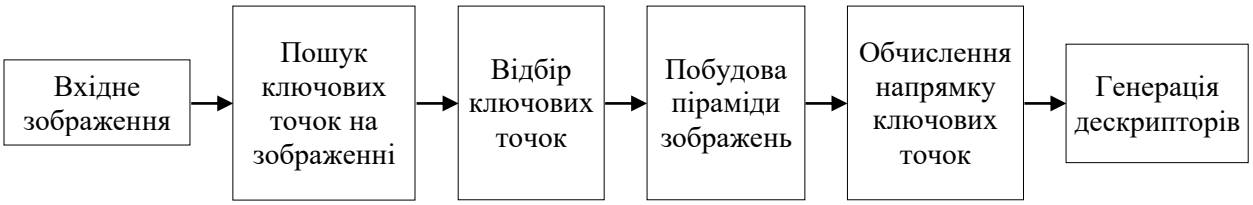


Рисунок 1.4 – Схема алгоритму ORB

Алгоритм ORB за логікою можна розбити на 2 кроки: знаходження ключових точок та генерація дескрипторів. Для виявлення ключових точок алгоритм ORB використовує вдосконалений алгоритм FAST. Ідея полягає в тому, що якщо піксель значно відрізняється від сусідніх пікселів то, швидше за все, це кутова точка. Процес виявлення наступний [27]:

Крок 1. Пошук ключових точок. Для кожного пікселю $p$ на зображенні обчислюється його яскравість $I_p$ та встановлюється поріг яскравості $T$. Після цього піксель $p$ приймається за центр умовного кола з радіусом 3, обираються 16 пікселів на даному колі і порівнюються значення сірого між пікселем $p$ та іншими пікселями в цьому колі. Якщо яскравість послідовних $N$ точок у вибраному колі більше ніж $I_p + T$ або менше ніж $I_p - T$, тоді піксель $p$ може розглядати як потенційну ключову точку. Якщо більше ніж 8 пікселів темніші або яскравіші за $p$, то цей піксель обирається як ключова точка. Таким чином, ключові точки, знайдені за алгоритмом FAST, дають інформацію про розташування визначальних контурів на зображенні. Цей процес зображено на рисунку 1.5.

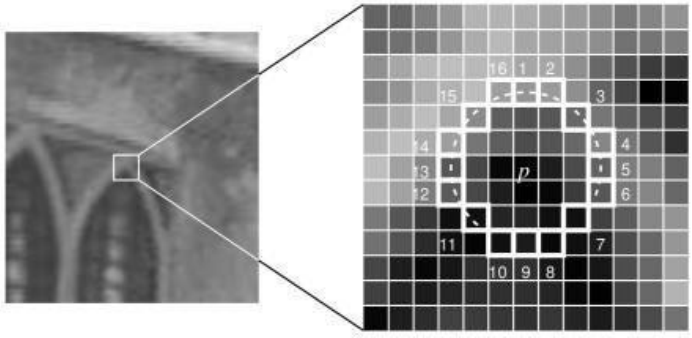


Рисунок 1.5 – Процес пошуку ключових точок за порогом яскравості

Крок 2. Відбір ключових точок. Оскільки процес розрахунку кутової точки FAST полягає лише в порівнянні різниць яскравості між пікселями, їх кількість завелика та не містить інформації про напрямок [25]. Алгоритм ORB покращує оригінальний алгоритм FAST шляхом обчислення значення відгуку Харріса для оригінальних кутових точок, сортування їх відповідно до значення сірого та відбору лише перших $N$ точок. Формула розрахунку значення відгуку Харріса показана в наступних рівняннях:

$$R = \det(M) - k(trace(M))^2, \tag{1.5}$$

$$M = \sum w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}, \tag{1.6}$$

де $R$ – значення відгуку Харріса;

$M$ – матриця 2×2;

$k$ – емпірично підібраний коефіцієнт, $k \in [0{,}04\ 0{,}06]$ [27];

$w(x, y)$ – вікно фрагменту зображення;

$I_x$ – варіація КТ в горизонтальному напрямку;

$I_y$ – варіація КТ у вертикальному напрямку.

Крок 3. Після цього генеруються піраміди зображення. На кожному шарі піраміди обираються ключові точки FAST та до них додається інваріантність масштабу. Приклад такої піраміди зображено на рисунку 1.6.

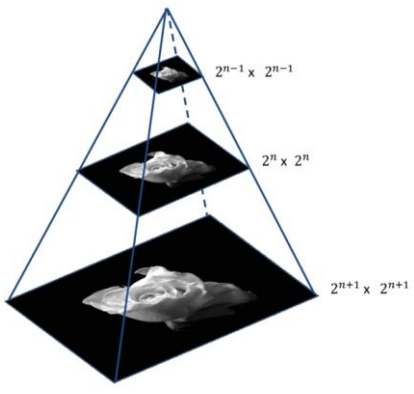


Рисунок 1.6 – Процес генерації піраміди зображень

Крок 4. Останнім кроком визначається напрямок ключової точки. Для досягнення інваріантності щодо обертання знайденими ключовими точками, їх напрямок обчислюється за допомогою методу центроїда інтенсивності [27]. Спочатку розраховується момент фрагмента зображення за формулою:

$$m_{pq} = \sum_{x,y \in B} x^p \, y^q I(x,y), \tag{1.7}$$

де $x$ і $y$ – це координати пікселів;

$I(x, y)$ – це значення сірого відповідного пікселя;

$p, q = \{0,1\}$.

Потім обчислюється центроїд фрагменту зображення за моментом:

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}\right), \tag{1.8}$$

де $(m_{00})$ – маса фрагменту зображення (0-й момент);

$(m_{10}, m_{01})$ – центроїд фрагменту зображення (1-й момент).

Таким чином, геометричний центр $O$ та центроїд $C$ фрагменту зображення формують вектор напрямку $\overrightarrow{OC}$ (рис. 1.7). Напрямок ключової точки обчислюється за формулою:

$$\theta = \arctan\left(\frac{m_{01}}{m_{10}}\right). \tag{1.9}$$

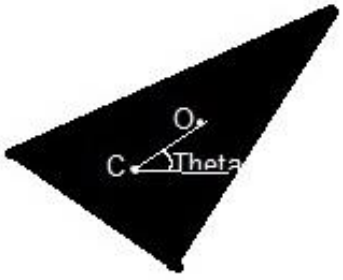


Рисунок 1.7 – Розрахунок напрямку ключової точки

Після наведених вище кроків, ключові точки FAST набувають інваріантність до масштабу та обертання, що значно покращує їх надійність на різних зображеннях.

Наступним кроком після формування остаточного списку ключових точок алгоритм ORB застосовує вдосконалений алгоритм BRIEF [26] для обчислення дескриптора кожної точки. BRIEF – це бінарний векторний дескриптор, вектор якого формується за наступними правилами:

$$\tau(p;x,y) = \begin{cases} 1, p(x) < p(y) \\ 0, p(x) \geq p(y) \end{cases}, \quad (1.10)$$

де $p(x)$ – це інтенсивність пікселю $p$ в точці $x$.

Щоб зменшити вплив шуму, спочатку до зображення застосовується фільтрація Гауса. Прийнявши ключову точку $p$ як центральну точку, на зображенні виділяється вікно розміром $S \times S$ і випадковим чином обираються $N$ ($N$ зазвичай дорівнює 256) пар пікселів у отриманому вікні [29]. Процес вибору випадкових пікселів наведено на рисунку 1.8. Потім значення яскравості кожної пари точок порівнюються відповідно до рівняння 1.10 і виконується бінарне присвоєння. Результатом даної операції буде $N$-вимірний вектор, що складається з $N$ бінарних рядків:

$$f_N(p) = \sum_{1 \leq i \leq N} 2^{i-1} \tau(p; x_i y_i). \quad (1.11)$$

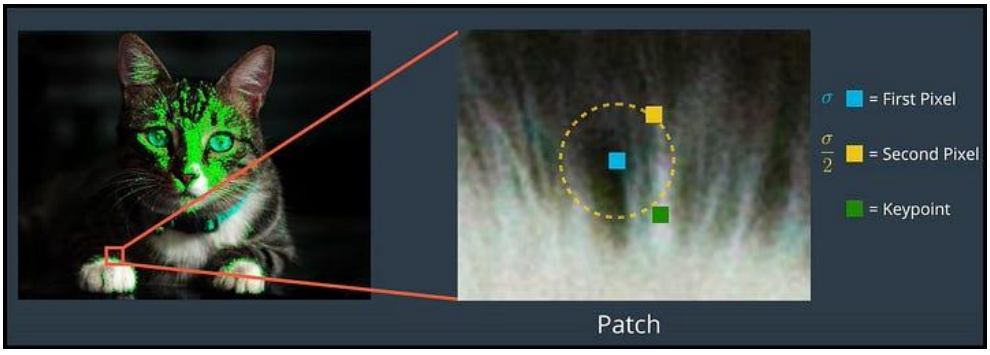


Рисунок 1.8 – Процес випадкового вибору пікселів під час формування бінарного дескриптора

Оскільки оригінальний дескриптор BRIEF не має інваріантності щодо обертання, дані зображення можуть загубитись при повороті. Тому алгоритм ORB використовує алгоритм Steer BRIEF для обчислення основного напрямку кожної ключової точки, щоб дескриптор містив інформацію про напрямок. Матриця обертання $R_\theta$ розраховується за наступною формулою:

$$R_\theta = \begin{pmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{pmatrix}. \quad (1.12)$$

Матриця $R_\theta$ і $N$ пар пікселів разом утворюють матрицю $Q$:

$$Q = \begin{bmatrix} x_1, x_2, \ldots, x_N \\ y_1, y_2, \ldots, y_N \end{bmatrix}. \quad (1.13)$$

Потім виконується корекція повороту, щоб отримати $Q_\theta$:

$$Q_\theta = R_\theta Q. \quad (1.14)$$

Останнім кроком розраховується дескриптор з урахуванням напрямку:

$$g_{N(p,\theta)} = f_N(p) | (x_i, y_i) \in Q_\theta. \quad (1.15)$$

## 1.3 Постановка задачі дослідження

Удосконалення результативності структурних методів класифікації зображень є актуальним на сьогодні завданням. Тому ставиться задача дослідження та розробки удосконаленого методу класифікації зображень шляхом скорочення обчислювальних витрат на підставі здійснення редукції складу описів.

Об’єктом досліджень є структурні методи класифікації зображень.

Метою дослідження є удосконалення результативності структурних методів класифікації зображень шляхом впровадження редукції описів даних з використанням параметрів класифікаційної значущості.

Для досягнення мети необхідно вирішити такі завдання:

– провести аналіз існуючих структурних методів класифікації зображень, а також математичних моделей по обчисленню інформативності дескрипторів та редукції описів даних;

– розробити апарат для редукції описів даних еталонних зображень на основі критерію найбільш інформативних дескрипторів;

– провести комп’ютерне тестування програмної моделі для методу класифікації на прикладній базі зображень з порівнянням результатів традиційних та удосконалених методів класифікації;

– зробити висновки та узагальнення щодо результативності чи напрямків впровадження щодо розроблених модифікацій класифікаторів.

# 2 КЛАСИФІКАЦІЯ ЗОБРАЖЕНЬ НА ОСНОВІ РЕДУКЦІЇ

## 2.1 Класифікація методом найближчого сусіда

Класифікація зображень методом найближчого сусіда є фундаментальною технікою у системах комп'ютерного зору та обробки зображень. Цей підхід особливо популярний для таких завдань, як розпізнавання об'єктів, пошук зображень і класифікація зображень на основі вмісту. Він найкраще підходить для малих і середніх наборів даних.

До переваг даного методу можна віднести [6, 30, 31]:

– простота: даний метод доволі простий та зрозумілий в реалізації, при цьому демонструючи достатньо високі точність та швидкодію;

– відсутність необхідності навчання: на відміну від багатьох методів машинного навчання, класифікація методом найближчого сусіда не вимагає фази навчання, в ньому просто порівнюються нові дані з існуючим еталонним набором;

– можливість інтерпретації: забезпечує прозорість у процесі прийняття рішень, оскільки можна безпосередньо відслідкувати, яке еталонне зображення в базі даних є найближчим сусідом до вхідного та на основі яких метрик було прийняте дане рішення;

– гнучкість: можна використовувати різні типи дескрипторів і метрики відстані, що робить його адаптованим до різних типів даних і вимог.

При цьому можна виділити наступні недоліки [6, 30, 31]:

– відсутність узагальнення: даний метод погано узагальнює нові, невідомі дані, оскільки покладається на існуючий набір посилань. Будь-які нові категорії чи шаблони можуть не розпізнаватися;

– неефективно для великих наборів даних: для великих баз даних час, необхідний для пошуку найближчих сусідів, може стати непрактичним, оскільки він не використовує узагальнюючі структури даних;

– обмежене вивчення ознак: методи найближчого сусіда не враховують ієрархічні або абстрактні характеристики з даних, на відміну від методів глибокого навчання, таких як згорткові нейронні мережі (CNN) [32, 33];

– відсутність розрізняльної сили: у випадках, коли дескриптори недостатньо відрізняються, класифікація методом найближчого сусіда може працювати неефективно.

Побудуємо математичну модель класифікатора на основі методу найближчого сусіда.

Нехай базу даних еталонних зображень подано у вигляді множини $E = \{E_1, E_2, \dots, E_N\}$, де $N$ – кількість еталонних класів, а кожен елемент множини задає собою простір ознак зображення відповідного класу. Кожен еталон представлений у багатовимірному просторі бінарних векторів $B^n$, де $n$ – розмірність бінарного вектору (наприклад, для дескрипторів ORB $n$ дорівнює 256 [27]).

$$E_i = \{e_v(i)\}_{v=1}^{s}, \tag{2.1}$$

де $i$ – номер класу;

$s = cardE_i$ – кількість виділених дескрипторів на кожному еталоні;

$e_v(i)$ – бінарний вектор $v$-го дескриптора еталону.

Для простоти обчислень $s$ приймається однаковою для всіх еталонних зображень. Тоді всю базу даних можна фактично представити у вигляді матриці $D$ розміром $S \times n$ та $S$–мірного вектору $L$, де $S = s \times N$ – кількість усіх обчислених дескрипторів:

$$D = \begin{bmatrix} e_1(1) \\ e_2(1) \\ \dots \\ e_s(1) \\ e_1(2) \\ \dots \\ e_s(N) \end{bmatrix}, L = \begin{bmatrix} 1 \\ 1 \\ \dots \\ 1 \\ 2 \\ \dots \\ N \end{bmatrix}. \tag{2.2}$$

Матриця $D$ формально є описом усіх візуальних ознак на еталонних зображеннях, а вектор $L$ використовується для знаходження відповідності між дескриптором та класом, якому він належить.

Структурний опис для вхідного зображення подається у вигляді моделі, аналогічній опису еталонних зображень:

$$O = \{o_v\}_{v=1}^{s}. \tag{2.3}$$

Для усіх введених множин справедливі наступні правила: $e_v \in B^n, E_i \subset B^n, o_v \in B^n, O \subset B^n$.

Наступним кроком треба обрати метрику для обчислення відстані між двома дескрипторами. В загальному вигляді її можна представити так: $d(f_a, f_b)$, де $f_a, f_b$ – бінарні дескриптори однакової розмірності, $f_a \in B^n, f_b \in B^n$.

Граничні значення даної метрики будуть залежати від обраної функції для обчислення відстані. Головне припущення, на яке спирається метод найближчого сусіда, полягає в тому, що чим менше відстань між двома дескрипторами, тим більше вони подібні між собою [30]. Найчастіше в якості функції відстаней для дескрипторів ключових точок використовуються Мангеттенська відстань (2.4), Евклідова відстань (2.5) та відстань Геммінга (2.6):

$$d(f_a, f_b) = \sum_{i=1}^{n} |f_a(i) - f_b(i)|. \tag{2.4}$$

$$d(f_a, f_b) = \sum_{i=1}^{n} (f_a(i) - f_b(i))^2. \tag{2.5}$$

$$d(f_a, f_b) = \sum_{i=1}^{n} (f_a(i) \oplus f_b(i)). \tag{2.6}$$

Оскільки у дослідженні використовуються бінарні дескриптори, то найоптимальнішою метрикою для таких даних буде відстань Геммінга, оскільки її результатом буде ціле число. Крім того, для її обчислення використовуються лише побітові операції, які є надзвичайно швидкими для обчислення на сучасних комп'ютерних архітектурах. На рисунку 2.1 наведено приклад обчислення відстані Геммінга для двох дескрипторів розмірністю 10 біт.

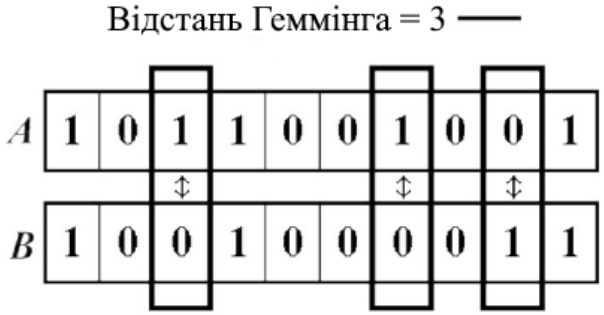


Рисунок 2.1 – Приклад розрахунку відстані Геммінга

Наступним кроком алгоритму є обчислення матриці попарних відстаней між еталонними дескрипторами множини $E$ та дескрипторами об'єкту, що досліджується:

$$P = d(O, D) = \begin{bmatrix} d(o_1, e_1(1)) & d(o_1, e_2(1)) & \cdots & d(o_1, e_s(1)) \\ d(o_2, e_1(1)) & d(o_2, e_2(1)) & \dots & d(o_2, e_s(1)) \\ \vdots & \vdots & \ddots & \vdots \\ d(o_v, e_1(N)) & d(o_v, e_2(N)) & \cdots & d(o_v, e_s(N)) \end{bmatrix}. \quad (2.7)$$

В отриманій матриці $P$ кожен рядок являє собою вектор відстаней між одним дескриптором розпізнаваного об'єкта та кожним дескриптором в базі даних, і такий вектор обчислюється для кожної ключової точки у вхідному зображенні. Далі, використовуючи цю матрицю, необхідно для кожного вхідного дескриптора визначити найбільш подібний йому еталонний

дескриптор в базі, тобто знайти в кожному рядку матриці найменшу відстань, а потім визначити його клас, використовуючи вектор відповідності $L$:

$$h(i) = L[arg \min_{i=1..v} P[i]], \tag{2.8}$$

де $P[i]$ – $i$-ий рядок у матриці відстаней;

$h(i)$ – номер найбільш подібного класу з бази даних еталонів до складової опису $o_i$, що тестується.

Номер класу, до якого належить об'єкт, що досліджується, обчислюється шляхом підрахунку кількості дескрипторів для кожного класу окремо (так зване голосування). Результатом даної операції буде вектор розподілу голосів $V = \{v_1, v_2, \dots, v_N\}$, де $N$ – загальна кількість класів у базі даних, $v_i$ – підсумована кількість голосів для $i$-го еталонного класу.

Для отримання фінального класу $\omega$ розпізнаваного об'єкту необхідно знайти номер еталону у векторі розподілу $V$ з максимальною кількістю отриманих голосів:

$$\omega = arg \max_{i=1..N} v_i. \tag{2.9}$$

Крім цього, для покращення результатів класифікації можна ввести поріг голосів, який визначає мінімальну кількість голосів, необхідну для віднесення вхідного зображення до певного класу. Якщо в результаті голосування кількість голосів не перевищила встановлене значення порогу, то такий об'єкт не можна з впевненістю віднести до жодного з класів, й скоріш за все, дане зображення відсутнє в еталонній базі даних.

$$\acute{\omega} = \begin{cases} \omega, \max\limits_{i=1..N} v_i \geq \varphi \\ -1, \max\limits_{i=1..N} v_i < \varphi \end{cases}, \tag{2.10}$$

де $\acute{\omega}$ – індекс знайденого класу або –1 у разі, якщо зображення не вдалось ідентифікувати;

$\varphi$ – поріг голосів, підібраний емпіричним шляхом.

Отримане значення $\acute{\omega}$ також необхідне для перевірки якості класифікатора на навчальній вибірці [15]. Якщо навчальна вибірка має вигляд $O = \{O_1, O_2, \dots, O_k\}$, де $k$ – загальна кількість зображень у вибірці, а $O_i$ – це структурний опис $i$-го зображення, $u = \{u_1, u_2, \dots, u_k\}$ – це множина істинних номерів класів для цих описів, $w = \{w_1, w_2, \dots, w_k\}$ – це множина номерів класів, отримана в результаті класифікації за формулою (2.10), то критерій якості класифікатора можна обчислити за наступною формулою:

$$\alpha = \frac{\sum_{i=1}^{k}(w_i \equiv u_i)}{k}, \alpha \in \{0,1\}. \quad (2.11)$$

Іншими словами, обчислюється відношення числа зображень, які за результатами оброблення віднесені до «правильного» класу, до загальної кількості зображень у навчальній вибірці. Ідеальним класифікатором можна вважати такий, для якого критерій $\alpha$ дорівнює або близький до одиниці.

## 2.2 Метод найближчого сусіда з подвійною перевіркою

У разі використання методу грубого підбору для зіставлення ключових точок між двома зображеннями завжди буде збіг для ключової точки на першому зображенні, навіть якщо її фактично немає на другому зображенні. Це може призвести до кількох помилкових збігів.

Розглянемо ситуацію на рисунку 2.2 [34]. Для простоти відображення ключові точки зображень тут наведені у вигляді кольорових двовимірних координат, де зелені точки описують дескриптори першого зображення, а червоні – іншого зображення, і ці зображення порівнюються між собою.

Зліва і по центру рисунку 2.2 можна побачити результат пошуку стандартним методом найближчого сусіда для кожного елементу опису зображення, зліва перше зображення порівнюється з другим, а по центру – в зворотному порядку. Можна відмітити, що найближчі сусіди для деяких ключових точок відрізняються. Це свідчить про те, що не завжди найближчий за відстанню дескриптор можна вважати збігом. Щоб вирішити цю проблему, можна використати пошук із подвійною перевіркою (Cross Checking) [34]. Це передбачає застосування процедури зіставлення в обох напрямках і збереження лише тих пар, у яких найкращий збіг в одному напрямку збігається з найкращим збігом в іншому напрямку. Результат даної операції можна побачити справа на рисунку 2.2. Дескриптори, обведені пунктиром, знайшли собі пару, решта залишились без співпадінь.

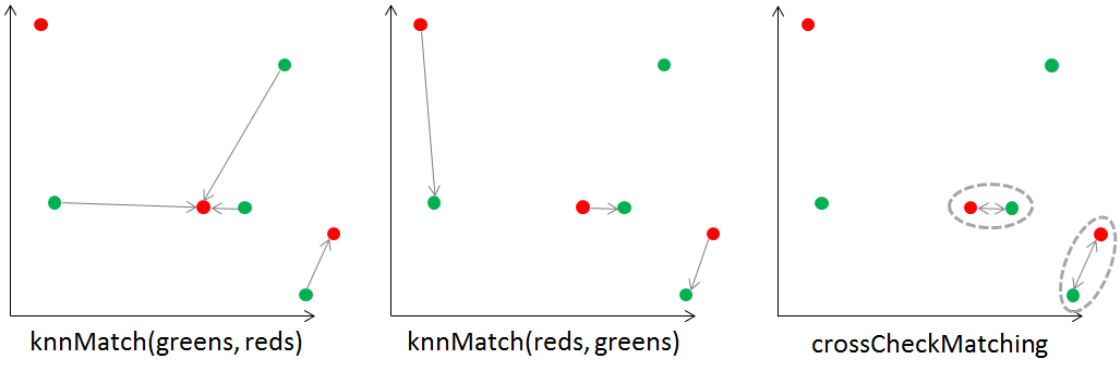


Рисунок 2.2 – Принцип роботи подвійної перевірки під час зіставлення ключових точок

Побудуємо математичну модель для даної операції. Вона виконується після розрахунку матриці відстаней (2.7) в стандартному методі найближчого сусіда. Для цього треба знайти ще одну матрицю відстаней, але в зворотному напрямку. Заново обчислювати повну матрицю нема потреби, оскільки достатньо просто транспонувати оригінальну матрицю $P$.

$$\acute{P} = P^T = \begin{bmatrix} d(e_1(1), o_1) & d(e_1(1), o_2) & \cdots & d(e_1(1), o_v) \\ d(e_2(1), o_1) & d(e_2(1), o_2) & \dots & d(e_s(1), o_v) \\ \vdots & \vdots & \ddots & \vdots \\ d(e_s(N), o_1) & d(e_s(N), o_2) & \cdots & d(e_s(N), o_v) \end{bmatrix}. \quad (2.12)$$

Після транспонування кожен рядок матриці $\acute{P}$ містить в собі вектор відстаней між одним еталонним дескриптором та кожним дескриптором опису вхідного зображення. Розмір даного вектору дорівнює загальній кількості ключових точок у описі розпізнаваного зображення, а висота матриці залежить від розміру бази даних. Далі, по аналогії з (2.8) треба знайти номери найближчих по відстані дескрипторів, але треба зробити це одночасно для обох матриць $P$ та $\acute{P}$:

$$I_1(i) = arg \min_{i=1..v} P[i], \quad (2.13)$$

$$I_2(i) = arg \min_{i=1..S} \acute{P}[i], \quad (2.14)$$

де $I_1$ – це вектор найбільш подібних ключових точок для опису розпізнаваного зображення;

$I_2$ – це вектор найбільш подібних ключових точок для еталонних описів.

Наступним кроком виконується співставлення отриманих векторів $I_1$ та $I_2$ та відбувається відсіювання дескрипторів, які не знаходяться у однакових позиціях по відношенню один до одного у цих векторах. Результатом буде новий вектор індексів $I, cardI = r, r \leq v$. Він буде містити в собі лише ті номери ключових точок опису, які знайшли собі пару після подвійної перевірки. Цей вектор далі використовується для отримання номеру класу по аналогії із стандартним методом найближчого сусіда (2.8).

$$h(i) = L[I[i]], i = 1,2, \dots, r. \quad (2.15)$$

Крім цього, для подальшого підвищення якості класифікації та зниження ймовірності помилкових спрацьовувань можна ввести порогове значення для відстані, при перевищенні якого ключові точки вважаються надто різними й, тому вважати їх еквівалентними буде не надійно [6]. Цей поріг обирається емпіричним шляхом в залежності вид вимог до класифікатора та виду аналізованих зображень. Менший поріг прийматиме лише дуже близькі збіги, тоді як вищий поріг може включати збіги з трохи вищими відстанями. Ця гнучкість важлива, оскільки ідеальний поріг може змінюватися залежно від програми та змісту даних.

Відсіювання збігів по максимальній відстані є необов'язковим кроком алгоритму. Найкраще застосовувати її до операції подвійної перевірки (2.12) для зменшення часу виконання, тому що подвійна перевірка вимагає сортування елементів, і чим менше в підсумку буде кількість елементів у матриці відстаней, тим швидше буде працювати алгоритм.

Для ще кращої якості та гнучкості у задачах класифікації можна ввести додаткову метрику співвідношення найближчих сусідив (Nearest Neighbor Ratio, NNR) або коефіцієнт впевненості [35]. Він обчислюється як співвідношення кількості голосів за клас, який переміг у голосуванні, до кількості голосів за клас, який посів друге місце. Щоб обчислити цей коефіцієнт, після знаходження розподілу голосів $V$ та виведення номеру класу-переможця $\omega$ після кроку (2.10) треба видалити його з множини $V$:

$$\acute{V} = V - \{\omega\}. \tag{2.16}$$

Далі треба ще раз знайти максимальне значення у отриманій множині:

$$\acute{\omega} = arg \max_{i=1..N-1} \acute{v}_i, \tag{2.17}$$

де $\acute{v}_i$ – $i$-ий елемент множини розподілу $\acute{V}, card\acute{V} = N - 1$.

Після цього коефіцієнт впевненості можна розрахувати за наступною формулою:

$$\rho = \frac{\acute{V}[\acute{\omega}]}{V[\omega]}, \rho \in \{0,1\}. \quad (2.18)$$

Чим ближче отриманий коефіцієнт $\rho$ до нуля, тим більше впевненість в тому, що отримана мітка класу є вірною. У випадку, коли цей коефіцієнт близький до одиниці, це буде означати що класифікатор не може однозначно визначити, до якого класу треба віднести вхідне зображення.

Введення коефіцієнта впевненості для класифікатора підвищує його надійність, додає стійкість до неоднозначності, а також створює додаткову гнучкість, що надає змогу налаштовувати класифікатор під конкретні задачі та набори даних.

## 2.3 Редукція даних на основі інформаційної вагомості

Одним з основних недоліків методу найближчого сусіда є низька швидкість при роботі з великими наборами даних. Це пов’язано з тим, що для пошуку подібних ключових точок в базі даних еталонних описів треба перевірити кожний наявний там дескриптор. Це дуже дорога операція з точку зору швидкодії, оскільки розмір бази даних може сягати тисяч або навіть десятків тисяч елементів. Для вирішення даної проблеми пропонується застосувати редукцію даних [4, 5, 11]. Але слід бути дуже обережним, інакше є ризик втрати важливої інформації, що негативно вплине на якість класифікації.

Деякі підходи, наприклад метод головних компонент (Principal Component Analysis, PCA) [36], використовується для зменшення розмірностей описів ключових точок, тобто знижується кількість елементів у

векторах ознак. Але цей метод вимагає високої кореляції між усіма ключовими точками для всіх еталонних зображень, що не завжди є справедливим при роботи з різними зображеннями, бо кореляція буде сильно залежати вид еталонної вибірки. Крім того, результатом даної операції будуть вектори з плаваючою зап’ятою, що ускладнює подальшу роботу з ними. Тому можна застосувати інший підхід – спробувати зменшити кількість самих дескрипторів, які представляють опис одного зображення. Основною задачею є вибір надійної техніки редукції, яка б дозволяла провести фільтрацію еталонних дескрипторів, що позитивно позначиться на швидкості роботи класифікатора, при цьому не сильно впливаючи на якість класифікації.

Для цього можна використати певний критерій для оцінювання значущості ключових точок, на підставі якого можна було б робити рішення наскільки дана ознака важлива у прийнятті рішень класифікації. Даний підхід базується на принципі роботи людського зору: якісь фрагменти зображення одразу кидаються в око, інші ж не сильно впливають на формування загальної картинки. Якщо застосувати цей підхід до систем комп’ютерного зору, то він дозволить стиснути описи зображень, залишивши лише найцінніші з точки зору класифікації елементи [37-40].

Критерії для оцінювання значущості ключових точок на зображенні зазвичай базуються на статистичному дослідженні описів [37-40]: обчислюється їх відмінність всередині власного опису і в межах бази еталонних описів, перевіряється ступінь стійкості до геометричних перетворень і дії завад, знаходяться унікальність та цінність присутності на зображенні і т. д. Основною вимогою є доступ до повної бази даних еталонів, з якою можна проводити ці статистичні дослідження. Тобто цей метод не підходить для задач, в яких еталонний набір часто змінюється, або нема доступу до еталонних зображень до процесу класифікації.

Нехай $D$ – матриця з еталонними описами об’єктів. Тоді для кожного рядка матриці, який являє собою багатомірний вектор з описом певної

ключової точки еталонного зображення, можна ввести таке поняття, як значущість. Вона розраховується за певним критерієм та дозволяє для кожного дескриптора опису отримати інформацію, наскільки він є важливим в даному конкретному описі. Маючи цю інформацію, можна побудувати класифікатор, який використовує значення значущості як параметр для фільтрації найбільш важливих ознак та їх подальшу редукцію. На рисунку 2.3 наведено принцип роботи такого класифікатора.


Рисунок 2.3 – Схема класифікації з редукцією опису

Найбільш популярні метрики по визначенню значущості ознак базуються на понятті еквівалентності дескрипторів [4, 5]. Якщо $o_1 \in B^n, o_2 \in B^n$ – це деякі бінарні вектори, що описують певну ключову точку на зображенні, то їх еквівалентність можна визначити за наступною формулою:

$$o_1 \equiv o_2 : d(o_1, o_2) \leq \delta. \tag{2.19}$$

де $\delta$ – деякий поріг, який визначає еквівалентність між двома ознаками.

Інтервал значень для даного порогу залежить від обраної функції відстані. Наприклад, для метрики Геммінга значення порогу повинне знаходитись в інтервалі $\{0, n\}$, де $n$ – це розмірність бінарного дескриптора, який використовується для обчислень. Для простоти використання можна обчислювати нормалізовану відстань замість абсолютної, що дає змогу використовувати значення порогу в інтервалі $[0, 1]$. Це буде зручним при використанні різних детекторів ключових точок, тому що $n$ може мінятись в залежності від обраного алгоритму, при цьому поріг буде працювати для всіх випадків однаково без необхідності зміни налаштувань алгоритму.

Існує декілька метрик, на основі яких можна робити припущення про значущість ознаки [4, 5]. Для початку введемо поняття числа повторювань $c_k$ для елемента опису $o_k$:

$$c_k = \sum_{v=1}^{s} 1(d(o_k, o_v) \leq \delta), \tag{2.20}$$

де $1(d(o_k, o_v) \leq \delta) = \begin{cases} 1, d(o_k, o_v) \leq \delta, \\ 0, \text{інакше.} \end{cases}$

Дана метрика відображає кількість елементів опису $O = \{o_v\}_{v=1}^{s}$, які еквівалентні елементу ознаки $o_k$, результатом операції буде ціле невід'ємне число.

Базуючись на числі повторювань, можна обчислити індекс унікальності елемента в своєму еталоні. Маючи множину еталонів $E = \{E_1, E_2, \dots, E_N\}$, де $N$ – загальна кількість еталонів в базі, індекс унікальності в своєму еталоні для дескриптора $e_v(i)$ можна розрахувати за наступною формулою:

$$\alpha_v(i) = \frac{c_k}{s}; a_v(i) \in [0,1], \tag{2.21}$$

де $c_k$ – число повторювань для елемента опису $e_v(i)$ в $i$-му еталоні, обчислений за формулою (2.20);

$s$ – загальна кількість ознак для одного еталона.

Індекс унікальності $\alpha_v(i)$ відображає ступінь повторюваності *v*-го елементу для *i*-го еталону. Низькі значення даної метрики свідчать про те, що ознаки зображення всередині одного еталону мають дуже високу розрізненість.

Оскільки для одного зображення ознаки можуть мату певну схожість, то високе значення індексу унікальності $\alpha_v(i)$ не є критичною проблемою при класифікації. Більш важливою метрикою для оцінювання унікальності елементу можна вважати індекс його унікальності в решти еталонів. Обчислити його можна за наступною формулою:

$$\beta_v(i) = \frac{c_k'}{\acute{s}}; \beta(i) \in [0,1], \tag{2.22}$$

де $c_k'$ – кількість подібних до $e_v(i)$ елементів серед бази даних, але за виключенням опису *i*-го еталону;

$\acute{s}$ – загальна кількість дескрипторів у базі даних за виключенням дескрипторів *i*-го еталону.

Чим ближче значення $\beta_v(i)$ до нуля, тим більш унікальним є елемент $e_v(i)$ серед решти еталонних описів. Високі значення даної метрики свідчать про те, що унікальність ознаки достатньо низька, що може негативно впливати на якість класифікації, якщо при прийнятті рішень враховувати цю ознаку.

Аналіз індексів унікальності $\alpha_v(i)$ та $\beta_v(i)$ дає можливість оцінити вплив тих чи інших ознак при їх застосуванні у структурних підходах класифікації зображень. Ця унікальність має вирішальне значення для уникнення неоднозначностей та неправильної класифікації, що особливо важливо в таких завданнях, як розпізнавання об’єктів, де системі потрібно точно ідентифікувати та класифікувати об’єкти на зображеннях. Аналіз інтегральних показників метрик $\sum \alpha_v(i)$ та $\sum \beta_v(i)$ додатково дає змогу оцінити спільну унікальність елементів еталону в межах бази. Це важливо

для аналізу побудованих структурних описів зображень для розуміння наскільки ефективно вони можуть використовуватись у задачах класифікації. Ці показники можуть бути використані для вибору детектора ключових точок та підбору параметрів класифікатора під конкретний набір даних.

Індекси унікальності $\alpha_v(i)$ та $\beta_v(i)$ можна застосувати як критерій для подальшої редукції ознак як всередині одного конкретного опису, так і в межах всієї еталонної бази [4, 5]. Ефективність класифікатора після редукції буде залежати від обраного порогу $\delta$, а також від кількості ознак, які залишаться після стиснення описів.

Ще одною важливою метрикою для аналізу значущості ознак є значення інформативності кожного еталонного дескриптора [4, 5]. Маючи множину еталонних описів $E = \{E_1, E_2, \dots, E_N\}$ та бінарний вектор $e_i$ як складову $i$-го еталонного опису $e_i \in E$, можна впровадити поняття інформативності дескриптора $V(e_i, E)$ у складі еталонної множини $E$:

$$V(e_i, E) = d(e_i, \bar{E}_k) - d(e_i, E_k), \tag{2.23}$$

де $d(e_i, \bar{E}_k)$ – це відстань від ознаки $e_i$ до найближчого дескриптора з множини описів еталонів, які не належать до класу $E_k$;

$d(e_i, E_k)$ – це мінімальна відстань від ознаки $e_i$ до найбільш подібного у цьому еталоні дескриптора, виключаючи відстань $e_i$ до самого себе, оскільки така відстань завжди буде найменшою та дорівнювати нулю.

Значення даної метрики буде знаходитись в інтервалі [$-n, n$], де $n$ – розмірність бінарних векторів, які використовуються для кодування простору ознак. Для зручності обчислень можна ввести нормалізацію відстані, після чого значення метрики інформативності буде трансформовано в інтервал [-1, 1]. Показники інформативності близькі до одиниці означають, що елемент опису має найбільшу відстань до множини елементів інших еталонів та найменшу відстань серед ознак для свого еталону. Такі елементи можна вважати найбільш значимими для класифікації. При низьких значеннях

інформативності ознаки можна спробувати відсіяти як такі, що не мають високого впливу на результат класифікації.

Критерій інформативності будемо використовувати як основу для здійснення редукції ознак. Для цього виконання редукції необхідно обчислити матриці попарних відстаней $M[E_k, E_k]$ та $M\left[E_k, E_j\right], \forall j \neq k$, та обчислити критерій (2.23) для кожної ознаки, яка зберігається в еталонній базі даних $E$.

Одним з можливих варіантів застосування метрики інформативності для редукції даних буде фільтрація еталонних дескрипторів за певним пороговим значенням:

$$\acute{E} = \{V(ei, E) \geq \varepsilon\}, \tag{2.24}$$

де $\acute{E}$ – нова обмежена множина опису, $\acute{E} \subset E$;

$\varepsilon$ – певний поріг, підібраний користувачем, на підставі якого виконується відсіювання.

Якщо інформативність ознаки не перевищує заданий поріг, то її дескриптор буде видалений з опису як не важливий. При такому підході є суттєвий недолік, який полягає в тому, що після застосування (2.24) кількість елементів в описах еталонів залишиться неоднорідною через неможливість гарантування однакового розподілу інформативності для всіх ознак у еталонах, зважаючи на принципи роботи алгоритмів для пошуку ключових точок на зображенні. Це ускладнить подальшу роботу з описами, а також може здійснити негативний вплив на результати класифікації через різну кількість ознак у описах.

Кращим варіантом буде відбір $k$ найбільш інформативних дескрипторів в одному описі, $k < s$. Якщо $V = \{v_1, v_2, \dots, v_s\}$ – це множина обчислених інформативностей для кожного дескриптора еталону $E_i$, то необхідно спочатку відсортувати отриману множину $V$ в порядку зменшення:

$$I = argsort\, V, \tag{2.25}$$

де $I$ – множина відсортованих дескрипторів у вигляді їх індексів.

Після цього для редукції опису необхідно відкинути з початкової множини ключових точок $E$ зайві дескриптори, використовуючи відсортовані індекси $I$:

$$\acute{E} = E_{i=1}^{k}[I[i]]. \tag{2.26}$$

Після стиснення класифікація зображень проводиться по стандартному методу найближчого сусіда, але використовуючи для обчислень редуковані множини описів еталонів.

Редукція ознак зображення має як переваги, так і недоліки, залежно від контексту та цілей обробки зображень або завдання комп'ютерного зору. До переваг редукції можна віднести наступні пункти [11, 40]:

– обчислювальна ефективність: обробка редукованих ознак вимагає менше обчислювальних ресурсів, що робить алгоритми швидшими та ефективнішими, що має вирішальне значення для обробки зображень у реальному часі або при роботі з великими наборами даних;

– покращене узагальнення: зменшення розмірності даних може допомогти запобігти перенавчанню, оскільки менша кількість дескрипторів робить модель менш схильною до збереження шумів або викидів у даних;

– спрощення моделі: редуковані дані утворюють простіші моделі, які краще інтерпретуються, що може бути цінним для вивчення базових закономірностей у наборах даних;

– збереження та економія пам'яті: менші набори дескрипторів займають менше місця для зберігання, що полегшує роботу з великими наборами еталонних зображень;

– зменшення шуму: зосереджуючись на найбільш інформативних дескрипторах, в якості бонусу редукції відфільтровується шумна або менш

релевантна інформація, що може призвести до кращої продуктивності класифікатора.

Однак редукція може мати й деякі негативні наслідки. Серед основних можна виділити наступні [11, 40]:

– втрата інформації: найсуттєвішим недоліком редукції дескрипторів зображень є потенційна втрата важливої інформації. Якщо відсіяти важливі деталі, продуктивність класифікатора може погіршитися. Але ця проблема вирішується правильним вибором критерію значущості;

– знижена розподільна здатність: менша кількість дескрипторів може призвести до меншої розподільної здатності, що призводить до підвищеного ризику неправильної класифікації, особливо при роботі зі складними або різноманітними даними зображення;

– чутливість до вибору ознак: вибір того, які ключові точки залишити або відкинути, може значно вплинути на результати. Неточний вибір ознак або зменшення розмірності може погіршити якість класифікації;

– компроміс зі складністю: занадто агресивне скорочення може надто спростити модель, унеможливлюючи розпізнавання нею складних моделей і деталей у наборах даних.

Слід зауважити, що в разі використання редукції для стиснення ознак зображення треба проводити тестування, щоб дослідити її вплив на якість класифікатора. При належному використанні стиснення даних точність класифікації повинна залишатись на тому ж рівні, що й до редукції або знизитись несуттєво. Редукція ознак є компромісом між обчислювальною ефективністю та збереженням інформації. Головне – знайти правильний баланс, вибравши відповідні методи зменшення та оцінивши вплив на продуктивність моделі. У деяких випадках це може значно підвищити ефективність (на порядки), але це слід робити з розумом, щоб уникнути критичної втрати інформації.

Використання критеріїв інформативності при редукції даних хоча й максимально знижує ризики втрати важливих даних, але редукція все одно

призводить до певних втрат інформації у описах ознак. При великих ступенях стиснення це може мати негативний вплив на якість класифікації. Якщо навіть для кількох дескрипторів не було знайдено підходящої пари, це може внести суттєвий вклад до результатів класифікації.

Пропонується використати значущість дескрипторів не тільки в якості критерію для редукції, а й як додатковий класифікаційний параметр. Для цього можна спробувати ввести ваговий коефіцієнт для кожного дескриптора, який буде визначати його важливість у еталонній базі даних. Тоді весь процес класифікації буде виглядати наступним чином:

Крок 1. Обчислюються критерії значущості для ознак. Можна використовувати ті ж самі критерії, що й для редукції, або можна спробувати інші, в залежності від задач та набору даних.

Крок 2. Після обчислення значущостей ознак для кожного дескриптора у базі назначається відповідна йому вага. Це може бути саме значення значущості, або якесь його трансформоване значення. Ознаки, важливіші для задачі класифікації, отримують вищі ваги, тоді як менш важливі ознаки отримують менші або нульові ваги.

Крок 3. Під час визначення релевантності між двома дескрипторами, ваги використовуються як мультиплікаційний фактор. Це дає більше впливу на важливі дескриптори та менше на менш важливі.

До переваг використання вагових коефіцієнтів можна віднести:

– покращену точність – зважування дескрипторів на основі значущості може підвищити точність моделі класифікації, зосереджуючись на найбільш інформативних характеристиках і применшуючи менш релевантні;

– стійкість – даний підхід може зробити модель більш надійною, зменшивши вплив шумних або нерелевантних ознак, що сприяє кращому узагальненню;

– інтерпретованість – призначаючи ваги дескрипторам, процес прийняття рішень у моделі стає більш зручним для інтерпретації. Легше зрозуміти, чому певні дескриптори були вирішальними при класифікації.

При цьому слід відзначити, що такий підхід має й свої недоліки:

– складність – обчислення та налаштування вагових коефіцієнтів для ознак може ускладнити модель і може потребувати додаткових етапів попередньої обробки;

– при невірно вибраному алгоритмі розрахунку вагових коефіцієнтів це може призвести навпаки до погіршення якості класифікації в деяких випадках.

Побудуємо математичну модель класифікації з використанням вагових коефіцієнтів. Після редукції даних (2.26) зробимо перерахунок показників значущостей для стисненої множини ознак, використовуючи критерій інформативності (2.23). При редукції до $k$ ознак, множина значень інформативності буде мати наступний вигляд: $V = \{v_1, v_2, \dots, v_k\}$. Можна спробувати використати ці інформативності в якості вагового коефіцієнта. Тоді при знаходженні найближчого дескриптора у (2.10) необхідно також додатково використати ваговий коефіцієнт:

$$h(i) = L[arg \min_{i=1..k} (v_i P[i])]. \quad (2.27)$$

Слід зазначити, що у разі використання фільтрації дескрипторів з обмеженням на максимальну відстань, цей поріг необхідно застосовувати саме до оригінальної відстані без урахування вагового коефіцієнта, щоб не відсіяти потенційно подібні дескриптори.

Але використання значень інформативності напряму в якості вагових коефіцієнтів може призвести до непередбачуваних та неконтрольованих результатів. Більш доречно буде розбити усі значення інформативності на рівномірні інтервалі фіксованого розміру та призначити дескриптору ваговий коефіцієнт в залежності від інтервалу, в який попадає його інформативність.

Наприклад, якщо взяти пул з 4 вагових коефіцієнтів, то інтервал значень інформативності розбивається на 4 рівні частини. Якщо використовуються нормалізовані значення відстаней, то отримаємо наступні

інтервали: [–1:–0,5), [–0,5:0), [0:0,5), [0,5:1]. Якщо значення інформативності для дескриптора, що розглядається, дорівнює –0,75, то воно попадає в перший інтервал і відповідно отримує ваговий коефіцієнт 1. При попаданні в другий інтервал дескриптору буде назначено коефіцієнт 2 і т. д. Таку операцію необхідно зробити для кожного еталонного дескриптора після застосування редукції. Після цього для класифікації слід застосувати (2.27), використовуючи замість значення інформативності отриманий ваговий коефіцієнт.

Використання інформативності дескрипторів як вагових коефіцієнтів може бути корисною стратегією, коли необхідно точно налаштувати продуктивність моделі класифікатора, надавши більшого значення найбільш релевантним ознакам. Але необхідно ретельно вивчити вплив використання вагових коефіцієнтів під набір даних, що використовується, щоб не знизити загальну якість класифікації.

# 3 АНАЛІЗ РЕЗУЛЬТАТІВ ТЕСТУВАННЯ КОМП'ЮТЕРНОЇ МОДЕЛІ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ

## 3.1 Обґрунтування вибору середовища програмної реалізації

У рамках кваліфікаційної роботи був розроблений алгоритм для класифікації зображень з редукцією описів на підставі критерію інформативності ознак. Для реалізації обрано мову програмування Python. Це мова програмування високого рівня загального призначення. Python має динамічну типізацію та автоматичну модель очищення пам'яті, що робить розробку простішою та більш зручною. Він підтримує кілька парадигм програмування, включаючи структуроване (зокрема процедурне), об'єктно-орієнтоване та функціональне [41]. Python вважається однією з найпопулярніших мов програмування, особливо в таких сферах, як комп'ютерне бачення.

Це зумовлено наступними перевагами [41]:

– багато бібліотек: Python може похвалитися багатьма бібліотеками обробки зображень, таких як OpenCV, Pillow, scikit-image тощо, надаючи широкий спектр інструментів і функцій для різноманітних завдань обробки зображень;

– простота та читабельність: синтаксис і читабельність Python полегшують розробникам розуміння, написання та підтримку коду, скорочуючи час на розробку та знижуючи ймовірність допустити помилку при програмуванні;

– підтримка спільноти: Python має велику та активну спільноту. Результатом цього є велика кількість онлайн-ресурсів, навчальних посібників та форумів, де розробники можуть шукати допомогу, ділитися знаннями та знаходити рішення, зокрема для систем обробки зображень;

– сумісність: Python добре взаємодіє з іншими мовами програмування та системами, полегшуючи інтеграцію з різними програмними та апаратними

компонентами, які зазвичай використовуються в програмах обробки зображень;

– швидке створення прототипів і розробка: простота використання Python і можливості швидкого створення прототипів дозволяють розробникам швидше експериментувати та виконувати ітерації, забезпечуючи ефективну розробку програм;

– кросплатформна сумісність: Python сумісний з основними операційними системами, забезпечуючи переносимість розроблених застосунків на різні платформи;

– машинне навчання та інтеграція штучного інтелекту: Python легко інтегрується з різними бібліотеками машинного навчання та штучного інтелекту, такими як TensorFlow, Keras і PyTorch, дозволяючи застосовувати у системах комп'ютерного зору більш просунуті рішення, такі як виявлення об'єктів, розпізнавання зображень і сегментація.

Оскільки в структурних методах класифікації зображень значна частина операцій – це робота з багатовимірними даними, то для цих цілей було використано бібліотеку NumPy. NumPy – це розширення мови Python, що додає підтримку великих багатовимірних масивів і матриць, разом з великою бібліотекою високорівневих математичних функцій для операцій з цими масивами розширення мови [42].

У контексті систем комп'ютерного зору NumPy пропонує кілька суттєвих переваг:

– ефективні операції з масивами: NumPy надає високопродуктивні багатовимірні масиви як новий тип даних та забезпечує інструментами для роботи з ними. Ці можливості мають вирішальне значення для обробки даних зображень, які по суті є набором пікселів, розташованих у сітці;

– подання зображення: зображення можуть бути представлені у вигляді масивів, де кожен піксель має певні значення, що позначають колір або інтенсивність. Маніпуляції з масивами NumPy дозволяють легко отримати

доступ, модифікувати та аналізувати ці значення пікселів, дозволяючи виконувати різноманітні завдання обробки зображень;

– математичні та статистичні операції: функції NumPy дозволяють виконувати різні математичні та статистичні операції над даними зображень. Ці операції можуть включати арифметичні операції, статистичний аналіз, фільтрацію, перетворення тощо;

– інтеграція з іншими бібліотеками: NumPy легко інтегрується з іншими потужними бібліотеками, які зазвичай використовуються для обробки зображень, такими як OpenCV і scikit-image. Разом вони утворюють комплексну екосистему для різноманітних завдань, пов’язаних із зображеннями;

– продуктивність і оптимізація: операції NumPy оптимізовані та реалізовані на C, що робить їх значно швидшими за стандартні операції Python. Ця ефективність життєво важлива для обробки великих наборів даних і виконання складних алгоритмів обробки зображень;

– трансляція масиву та нарізка: можливості трансляції та нарізки NumPy дозволяють стислу та ефективну маніпуляцію масивами. Це полегшує обробку різних форм даних зображення та застосування операцій у масивах без явного циклу.

У програмних застосунках, побудованих у екосистемі Python, які потребують великої кількості операцій з багатовимірними даними, включаючи фільтрацію, перетворення, маніпуляції та аналіз, NumPy є невід’ємним інструментом.

Для операцій над зображеннями було використано одну з найбільш популярних бібліотек для мови програмування Python – OpenCV (Open Source Computer Vision Library). Це безкоштовна бібліотека функцій та алгоритмів комп’ютерного зору, обробки зображень і чисельних алгоритмів загального призначення з відкритим кодом, розроблена компанією Intel. Її використання дозволяє отримати ряд переваг [43]:

– розширені функціональні можливості: OpenCV пропонує широкий набір функцій і алгоритмів, розроблених спеціально для аналізу зображень і відео. Дана бібліотека охоплює широкий спектр завдань, включаючи фільтрацію зображень, виявлення об’єктів, виділення ознак, зшивання зображень тощо;

– багатоплатформова підтримка: OpenCV має сумісність з різними операційними системами (Windows, Linux, macOS тощо), що робить його універсальним вибором для будь-яких задач;

– обробка в реальному часі: OpenCV оптимізовано для обробки зображень у реальному часі, що робить його ідеальним для застосунків, які потребують швидкої обробки великих масивів даних;

– висока продуктивність: OpenCV написано на C і C++ з програмними інтерфейсами, доступними в тому числі й для Python. Це забезпечує високу швидкодію і ефективне виконання алгоритмів обробки зображень, що робить його придатним для обробки великих обсягів даних;

– спільнота та підтримка: велика та активна спільнота робить внесок у його розвиток. Це призводить до постійного вдосконалення, виправлення помилок і наявності достатньої кількості документації, навчальних посібників і ресурсів для розробників;

– інтеграція з апаратним забезпеченням: OpenCV може використовувати можливості спеціалізованого обладнання, наприклад GPU, щоб прискорити певні завдання обробки зображень, забезпечуючи покращену продуктивність програм;

– зручний інтерфейс: OpenCV пропонує прості у використанні функції та API, які спрощують розробку складних систем комп’ютерного зору, роблячи їх доступними як для початківців, так і для досвідчених розробників;

– інтеграція з екосистемою Scientific Python: будучи частиною екосистеми Scientific Python, OpenCV легко інтегрується з іншими бібліотеками, такими як NumPy, SciPy і Matplotlib. Ця інтеграція дозволяє легко маніпулювати даними, науковими обчисленнями та візуалізацією;

– модульність і розширюваність: модульна конструкція OpenCV дозволяє легко розширювати та змінювати налаштування, спрощуючи додавання нових функцій або адаптацію існуючих для задоволення конкретних потреб при обробці зображень.

В якості програмного середовища для розробки застосунку було обрано Google Colab. На рисунку 3.1 зображено його інтерфейс.

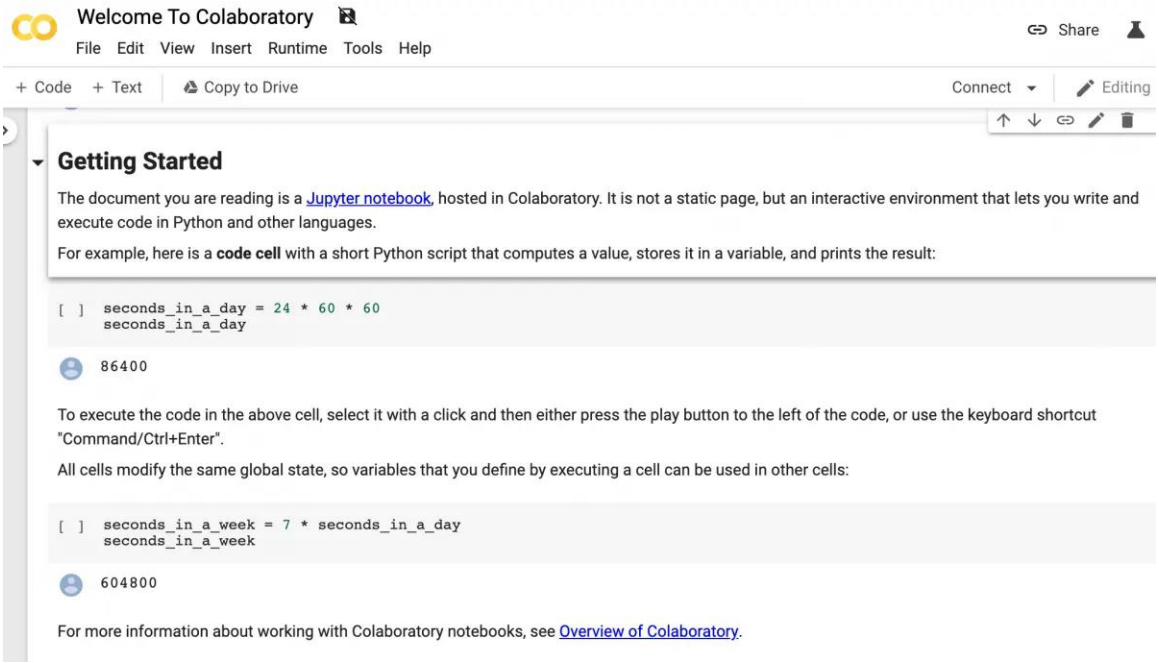


Рисунок 3.1 – Інтерфейс Google Colab

Цей вибір обумовлено рядом переваг, які надає ця платформа [44]:

– безкоштовний доступ і розміщення в хмарі: Google Colab є безкоштовним у використанні та повністю працює в хмарі, усуваючи потребу в локальних налаштуваннях. Користувачі можуть отримати доступ до нього з будь-якого пристрою з підключенням до Інтернету, що робить його дуже зручним і доступним при роботі з декількох пристроїв або при роботі командою над одним проєктом;

– інтеграція Jupyter Notebooks: Google Colab забезпечує інтеграцію Jupyter Notebooks, надаючи інтерактивне середовище, яке дозволяє

створювати та обмінюватися документами, що містять живий код, формули, візуалізації та пояснювальний текст в одному місці;

– підтримка GPU та TPU: Google Colab пропонує доступ до графічних процесорів (GPU) і тензорних процесорів (TPU), забезпечуючи значну обчислювальну потужність для таких завдань, як машинне навчання, глибоке навчання та аналіз даних. Ця можливість дозволяє написаним застосункам працювати набагато швидше, ніж це було б при використанні звичайного апаратного забезпечення користувача;

– попередньо встановлені бібліотеки: Google Colab постачається з попередньо встановленими різними бібліотеками, які зазвичай використовуються в наукових проєктах і машинному навчанні, включаючи NumPy, OpenCV, scikit-learn тощо, що економить час користувачів на встановлення та налаштування, а також виключає проблему з конфліктами версій;

– проста інтеграція з Диском Google: користувачі можуть безперешкодно отримувати доступ до даних, що зберігаються на Диску Google, що дозволяє легко обмінюватися файлами, наборами даних і моделями між Google Colab і Google Диском;

– контроль версій і співпраця: Google Colab інтегрується з такими службами, як GitHub, що дозволяє контролювати версії та співпрацювати над проєктами з колегами по команді, полегшуючи спільну розробку та підвищуючи продуктивність;

– гнучкість і масштабованість: забезпечує гнучкість щодо ресурсів і можливість масштабування. Користувачі можуть перемикатися між CPU, GPU або TPU залежно від своїх вимог, що робить його придатним для проєктів різної складності.

## 3.2 Особливості програмної реалізації

Перед розробкою програмного застосунку для проведення експериментального дослідження було сформовано базу еталонних зображень. Дана база складається з 5 фото музейних прикрас з Національного музею історії України [45]. Кожне зображення має однотипний світлий фон для чіткого розділення між об’єктом і заднім планом. Усі еталонні об’єкти були приведені до однакового розміру 512×512, що повинно сприяти більш ефективним і послідовним порівнянням. Це зумовлено тим, що в основі алгоритму класифікації експериментального дослідження лежить метод найближчого сусіда, який ґрунтується на вимірюванні відстаней та подібностей між зображеннями, а наявність однакового розміру зображення допомагає вирівняти характеристики для точного порівняння. Базу еталонних зображень наведено на рисунку 3.2.

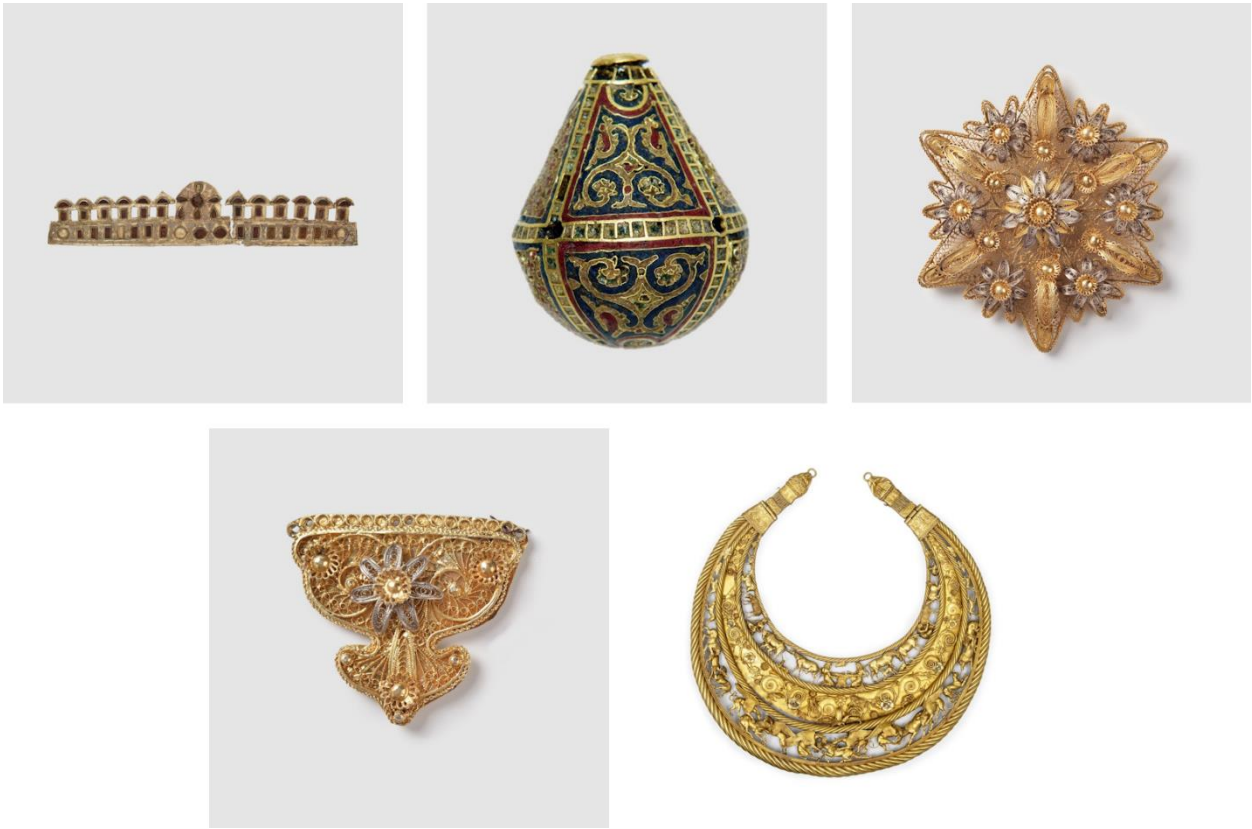

Рисунок 3.2 – Еталонна база даних

За допомогою бібліотеки OpenCV кожне еталонне зображення трансформується в чорно-біле, як цього вимагає детектор ORB, після чого на ньому виділяються ключові точки у кількості, заданій користувачем. Для даного експериментального дослідження кількість виділених ознак становить 500. Детектор повертає 2 множини – з двомірними координатами знайдених ключових точок та їх відповідними дескрипторами. Координати ознак в розрахунках не використовуються і застосовуються лише для візуалізації, а дескриптори зберігаються у один загальний масив, який фактично являє собою еталону базу даних. Слід зазначити, що OpenCV зберігає дескриптори не у масиві окремих бітів, а в закодованому вигляді, який складається з 8 цілочисельних значень. Для простоти розробки, а також для сумісності з алгоритмами комп'ютерного зору з інших бібліотек, перед тим як помістити їх в базу даних, дані дескриптори додатково конвертуються у стандартне бітове представлення за допомогою бібліотеки NumPy. Також, поряд зі списком еталонних дескрипторів додатково формується список з номерами класів, до яких належать дескриптори. Хоча це не є необхідністю для цього конкретного дослідження, оскільки кожний еталон містить однакову кількість дескрипторів, але наявність цього списку спрощує роботу при використанні стандартних бібліотечних функцій, а також при фільтрації даних.

Цей процес виконується для кожного еталонного зображення і відображений у лістингу 3.1.

Лістинг 3.1 Генерація еталонної бази даних:

```
descriptors = []
labels = []

# Initialize the ORB detector
orb = cv.ORB_create(nfeatures=num_descriptors)
```

```
for i in range(class_count):
  img_path = img_base_path + images[i]
  img = cv.imread(img_path)
  img = cv.resize(img, image_size)

  # Convert it to grayscale
  img_bw = cv.cvtColor(img, cv.COLOR_BGR2GRAY)

  # Find ORB descriptors on the original image
  kp, ds = orb.detectAndCompute(img_bw, None)
  ds = np.unpackbits(ds, axis=1)

  descriptors.extend(ds)
  labels.extend(np.full(shape=ds.shape[0], fill_value=i))

descriptors = np.array(descriptors).astype(float)
labels = np.array(labels)
```

На рисунку 3.3 наведено результат конвертації закодованого дескриптора ORB у стандартне бінарне подання.

```
1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1,
0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0,
0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0,
1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0,
0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1,
0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0,
0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1,
1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0,
1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1,
1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1,
1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1
```

Рисунок 3.3 – Приклад бінарного ORB-дескриптора

На рисунку 3.4 зображено результат роботи детектора ORB. Координати знайдених ключових ознак зображення позначені синіми точками.

Рисунок 3.4 – Виділені 500 КТ на одному з еталонних зображень

Для тестування продуктивності класифікатора було також сформовано базу тестових зображень. Вона складається з еталонних зображень, до яких було застосовано наступні афінні перетворення: повороти 45°, 30° та 60° і масштаби 120% та 80%. Комбінація різних масштабів та поворотів створює загалом 30 різних тестових зображень. На рисунку 3.5 наведено приклад декількох таких об’єктів.

Рисунок 3.5 – Перетворені еталонні зображення

Крім того, для перевірки наскільки якісно класифікатор вміє розрізняти зображення, до тестової вибірки були введені 3 нові зображення з прикрасами з того самого музею, але яких не було серед еталонних. Цей підхід також відомий як zero-shot training [46]. До кожного з цих зображень було застосовано ті ж самі трансформації, що й до еталонних, а також додатково було збережено оригінальне зображення до перетворень. На рисунку 3.6 можна побачити ці об’єкти.

Рисунок 3.6 – Тестові зображення, не включені у навчальну вибірку

Слід зазначити, що при геометричних трансформаціях оригінальний фон зображення може втрачатись. Неоднорідний фон може негативно вплинути на виділення ознак у тестовій вибірці та знизити якість класифікації. Тому, окрім трансформацій, при генерації тестових зображень

необхідно вирівнювати задній фон, щоб забезпечити його однорідність. Далі наведено код, який виконує формування тестової вибірки.

Лістинг 3.2 Формування тестової вибірки:

```
test_samples = []

# Initialize the ORB detector
orb = cv.ORB_create(nfeatures=num_descriptors)

for sample_index in range(class_count):
  img_path = img_base_path + images[sample_index]

  img_color = cv.imread(img_path)
  img_color = cv.resize(img_color, image_size)

  # Convert it to grayscale
  img_gray = cv.cvtColor(img_color, cv.COLOR_BGR2GRAY)

  for sample_transform in test_sample_transforms:

    r = cv.getRotationMatrix2D((img_gray.shape[0]/2,
img_gray.shape[1]/2), angle=sample_transform[0], scale=sample_transform[1])
    img_gray_transformed = cv.warpAffine(img_gray, r, image_size
borderMode=cv.BORDER_CONSTANT, borderValue=tuple( img_color[0,
0].tolist() ))

    # Find ORB descriptors on the original image
    kp, ds = orb.detectAndCompute(img_gray_transformed, None)
    ds = np.unpackbits(ds, axis=1)
```

*test_sample = TestSample(true_label=sample_index, transform=sample_transform, descriptors=ds)*

*test_samples.append(test_sample)*

Результатом виконання наведених вище скриптів буде сформована еталонна база даних на 2500 дескрипторів і тестова вибірка, до якої входить 51 зображення. З цими вхідними даними буде проводитись подальше тестування експериментальної моделі класифікатора.

## 3.3 Аналіз результатів тестування

В першу чергу необхідно реалізувати класифікатор класичним методом найближчого сусіда з подвійною перевіркою. Це дасть змогу порівнювати зміни у точності та швидкодії при реалізації експериментальної моделі. Далі наведено код з реалізацією класифікатора.

Лістинг 3.3 Функція для класифікації зображення методом найближчого сусіда з подвійною перевіркою:

*def predict_class(sample, descriptors, labels, min_votes, max_nearest_coef=0, max_distance=64):*

*indices = match_descriptors(sample, descriptors, metric='hamming', max_distance=(max_distance+1)/256., cross_check=True)[:,1]*

*# count number of indices for each class*

*class_indices, match_counts = np.unique(labels[indices], return_counts=True)*

*# fill vote table*

```
vote_table = np.zeros(class_count)
vote_table[class_indices] = match_counts

# calculate predicted label by sorting vote table and taking 1st element
sorted_votes = np.argsort(-vote_table)
predicted_label = sorted_votes[0]

# calculate nearest neighboor metric
nearest_max = vote_table[sorted_votes[1]]
nearest_max_coef = nearest_max / vote_table[predicted_label]

if vote_table[predicted_label] < min_votes:
  predicted_label = -1

if max_nearest_coef > 0 and nearest_max_coef >= max_nearest_coef:
  predicted_label = -1

return predicted_label, vote_table, nearest_max_coef
```

Функція для класифікації отримує на вхід еталонну базу даних дескрипторів, дескриптори об’єкта, що тестується, а також параметри, необхідні для прийняття рішення щодо класу об’єкта: мінімальна кількість голосів, максимальний коефіцієнт впевненості та максимальна відстань між дескрипторами, за якою можна визначити еквівалентність двох ключових точок. У таблиці 3.1 наведені результати класифікації з такими вхідними параметрами: мінімальна кількість голосів – 220, максимальний коефіцієнт впевненості – 0,62, максимальна відстань – 64.

Таблиця 3.1 – Результат класифікації методом найближчого сусіда

| № | Розподіл голосів | Коефіцієнт впевненості | Час класифікації | Результат |
|---|---|---|---|---|
| **1** | **2** | **3** | **4** | **5** |
| 1 | 270 – 5 – 0 – 8 – 6 | 0,02963 | 0,31154 | + |
| 2 | 270 – 15 – 5 – 6 – 3 | 0,05556 | 0,29767 | + |
| 3 | 284 – 6 – 5 – 5 – 3 | 0,02113 | 0,28841 | + |
| 4 | 273 – 12 – 2 – 4 – 6 | 0,04396 | 0,29050 | + |
| 5 | 286 – 8 – 3 – 7 – 9 | 0,03147 | 0,28802 | + |
| 6 | 276 – 11 – 2 – 7 – 10 | 0,03986 | 0,28079 | + |
| 7 | 18 – 238 – 7 – 4 – 1 | 0,07563 | 0,29314 | + |
| 8 | 4 – 230 – 9 – 6 – 6 | 0,03913 | 0,29399 | + |
| 9 | 19 – 241 – 12 – 8 – 7 | 0,07884 | 0,29098 | + |
| 10 | 8 – 224 – 5 – 5 – 8 | 0,03571 | 0,29746 | + |
| 11 | 12 – 223 – 11 – 7 – 8 | 0,05381 | 0,29286 | + |
| 12 | 8 – 227 – 7 – 8 – 7 | 0,03524 | 0,29205 | + |
| 13 | 13 – 8 – 287 – 5 – 2 | 0,04530 | 0,29379 | + |
| 14 | 12 – 11 – 257 – 3 – 13 | 0,05058 | 0,28634 | + |
| 15 | 7 – 11 – 284 – 6 – 1 | 0,03873 | 0,28831 | + |
| 16 | 10 – 7 – 270 – 4 – 7 | 0,03704 | 0,23030 | + |
| 17 | 11 – 10 – 291 – 6 – 5 | 0,03780 | 0,17731 | + |
| 18 | 7 – 8 – 258 – 8 – 8 | 0,03101 | 0,17837 | + |
| 19 | 7 – 7 – 7 – 266 – 3 | 0,02632 | 0,16291 | + |
| 20 | 17 – 7 – 5 – 267 – 10 | 0,06367 | 0,16968 | + |
| 21 | 8 – 6 – 6 – 276 – 6 | 0,02899 | 0,17675 | + |
| 22 | 10 – 9 – 6 – 281 – 11 | 0,03915 | 0,16418 | + |
| 23 | 8 – 7 – 9 – 275 – 4 | 0,03273 | 0,16687 | + |
| 24 | 16 – 7 – 5 – 263 – 10 | 0,06084 | 0,16970 | + |
| 25 | 15 – 11 – 7 – 14 – 244 | 0,06148 | 0,16486 | + |
| 26 | 11 – 7 – 7 – 7 – 255 | 0,04314 | 0,17510 | + |
| 27 | 14 – 5 – 6 – 7 – 261 | 0,05364 | 0,18177 | + |
| 28 | 14 – 7 – 4 – 5 – 264 | 0,05303 | 0,17336 | + |
| 29 | 12 – 10 – 6 – 7 – 249 | 0,04819 | 0,16910 | + |
| 30 | 10 – 8 – 6 – 8 – 253 | 0,03953 | 0,17182 | + |
| 31 | 56 – 34 – 23 – 14 – 33 | 0,60714 | 0,16900 | + |
| 32 | 51 – 38 – 28 – 14 – 23 | 0,74510 | 0,17112 | + |
| 33 | 56 – 33 – 20 – 24 – 30 | 0,58929 | 0,17932 | + |
| 34 | 56 – 37 – 20 – 20 – 29 | 0,66071 | 0,16533 | + |
| 35 | 62 – 42 – 25 – 26 – 26 | 0,67742 | 0,16504 | + |
| 36 | 49 – 30 – 23 – 16 – 22 | 0,61224 | 0,16636 | + |
| 37 | 51 – 43 – 25 – 18 – 23 | 0,84314 | 0,17363 | + |
| 38 | 51 – 31 – 18 – 24 – 19 | 0,60784 | 0,16972 | + |

Продовження таблиці 3.1

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 39 | 57 – 22 – 15 – 23 – 28 | 0,49123 | 0,17824 | + |
| 40 | 58 – 18 – 17 – 23 – 14 | 0,39655 | 0,16983 | + |
| 41 | 55 – 24 – 13 – 23 – 22 | 0,43636 | 0,16335 | + |
| 42 | 47 – 18 – 17 – 22 – 21 | 0,46809 | 0,16495 | + |
| 43 | 52 – 28 – 20 – 25 – 18 | 0,53846 | 0,16325 | + |
| 44 | 47 – 19 – 13 – 22 – 22 | 0,46809 | 0,16552 | + |
| 45 | 54 – 36 – 16 – 26 – 34 | 0,66667 | 0,17499 | + |
| 46 | 60 – 42 – 20 – 32 – 32 | 0,7 | 0,16774 | + |
| 47 | 54 – 40 – 20 – 34 – 35 | 0,74074 | 0,16678 | + |
| 48 | 59 – 42 – 30 – 25 – 37 | 0,71186 | 0,17018 | + |
| 49 | 51 – 33 – 16 – 20 – 44 | 0,86275 | 0,16944 | + |
| 50 | 59 – 47 – 24 – 32 – 25 | 0,79661 | 0,17234 | + |
| 51 | 60 – 27 – 21 – 29 – 33 | 0,55 | 0,17761 | + |

В останньому стовпці таблиці можна побачити, чи правильно було класифіковано тестований об’єкт. Перші 30 зображень – це еталони після трансформації, решта 21 – це зображення, яких не було при формуванні еталонної вибірки. Такий порядок буде зберігатись для усіх проведених експериментів. Загалом, точність класифікації склала 100%, середній час класифікації становив 0,2 с. Середнє значення коефіцієнта впевненості для об’єктів, які входять до еталонної вибірки, складає 0,044, для об’єктів, які відсутні в еталонній вибірці – 0,627.

Це показники, на які необхідно спиратися при реалізації експериментальної моделі. Спробуємо покращити їх за допомогою редукції ознак. Першим етапом необхідно визначити інформативність кожного дескриптора у еталонній базі даних. Далі йде код, який визначає інформативність ознаки у складі еталонної множини за формулою (2.23).

Лістинг 3.4 Функція для розрахунку інформативності еталонних дескрипторів:

```
def calculate_importance(descriptors, labels):
```

```
    distance_matrix = pairwise_distances(descriptors, descriptors,
metric='l1')

    importance = np.empty(descriptors.shape[0])

    for c_index in range(class_count):
      mask = labels==c_index
      self_distances = distance_matrix[mask]

      # find closest descriptor in class by searching for 2nd smallest distance
      self_class_nearest_min = np.partition(self_distances[:,mask], 1,
axis=1)[:,1]

      # find closest descriptor in all other classes
      other_class_nearest_min = np.min(self_distances[:,~mask], axis=1)

      # calculate descriptor importance
      importance[mask] = other_class_nearest_min - self_class_nearest_min

    return importance
```

Результатом роботи функції буде масив інформативностей, розмір якого еквівалентний кількості дескрипторів, переданих вхідним параметром. Також в якості параметру необхідно передати масив міток класів, щоб можна було згрупувати дескриптори по своїм еталонам. У таблиці 3.2 наведено дані, отримані в результаті розрахунків для еталонної вибірки зображень.

Таблиця 3.2 – Результати обчислення інформативності для еталонного набору даних

| № еталону | 50 найвищих інформативностей еталону | Середнє значення | Медіана |
|---|---|---|---|
| 1 | 34, 34, 34, 34, 34, 34, 34, 35, 35, 35, 35, 35, 35, 35, 35, 36, 36, 36, 36, 36, 36, 36, 36, 36, 37, 37, 37, 37, 37, 37, 37, 38, 38, 38, 39, 39, 39, 39, 39, 41, 41, 42, 43, 43, 44, 44, 44, 45, 48, 50 | 15,65 | 16,0 |
| 2 | 24, 24, 24, 24, 24, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 26, 26, 26, 26, 26, 26, 27, 27, 27, 27, 27, 27, 27, 28, 29, 29, 29, 29, 29, 30, 30, 31, 31, 31, 31, 31, 31, 32, 32, 33, 34, 35, 36, 36, 37 | 4,286 | 3,0 |
| 3 | 25, 25, 25, 25, 25, 25, 26, 26, 26, 26, 26, 26, 26, 27, 27, 27, 27, 27, 27, 28, 29, 30, 30, 30, 31, 31, 31, 32, 32, 32, 33, 33, 33, 33, 33, 33, 34, 34, 34, 34, 34, 35, 35, 36, 37, 37, 37, 38, 38, 41 | 6,774 | 6,0 |
| 4 | 25, 25, 25, 25, 25, 25, 26, 26, 26, 26, 26, 26, 26, 27, 27, 27, 27, 27, 27, 28, 29, 30, 30, 30, 31, 31, 31, 32, 32, 32, 33, 33, 33, 33, 33, 33, 34, 34, 34, 34, 34, 35, 35, 36, 37, 37, 37, 38, 38, 41 | 7,602 | 8,0 |
| 5 | 24, 24, 25, 25, 25, 25, 25, 25, 26, 26, 26, 26, 26, 26, 26, 27, 27, 27, 27, 27, 27, 27, 28, 28, 28, 28, 29, 29, 29, 29, 29, 29, 30, 30, 30, 31, 31, 31, 31, 31, 31, 31, 32, 33, 34, 34, 36, 36, 38, 40 | 6,02 | 5,0 |

При аналізі отриманої таблиці можна побачити, що статистичні дані інформативностей відрізняються в залежності від еталону. Можна припустити, що редукція матиме найкращі результати для першого еталону, в якому показники інформативності його дескрипторів найвищі по відношенню до решти еталонів.

Маючи значення інформативностей для кожного дескриптора, можна провести редукцію даних. Далі наведено програмний код для редукції множини ознак, спираючись на їх інформативність.

Лістинг 3.5 Функція для редукції ознак зображення:

```
def reduce_by_importance(descriptors, labels, importance, after_reduce_count, return_indices=False):
    descriptors_reduced = np.empty((class_count*after_reduce_count, descriptors.shape[1]))
    labels_reduced = np.empty(class_count*after_reduce_count, dtype=int)

    indices_reduced = None
    if return_indices:
      indices_reduced = np.empty((class_count,after_reduce_count), dtype=int)

    for c_index in range(class_count):
      mask = labels==c_index
      self_descriptors = descriptors[mask]

      # take N most important descriptors
      indices = np.argsort(importance[mask])[-after_reduce_count:]

      # put it into `reduced` arrays
      mask_reduced = np.arange(after_reduce_count*c_index, after_reduce_count*(c_index+1), dtype=int)
      descriptors_reduced[mask_reduced,:] = self_descriptors[indices,:]
      labels_reduced[mask_reduced] = np.full(after_reduce_count, c_index)

      if return_indices:
        indices_reduced[c_index] = indices

    result = [descriptors_reduced, labels_reduced]
```

```
    if return_indices:
      result.append(indices_reduced)

    return result
```

На вхід у функцію необхідно подати масив дескрипторів, мітки їх класів, обчислену раніше інформативність, а також кількість ознак, яку необхідно залишити після редукції. На рисунку 3.7 виділені найбільш значущі ключові точки на еталонних зображеннях, які залишились після редукції до 50 дескрипторів.

Рисунок 3.7 – Ключові точки на еталонних зображеннях після застосування редукції даних

Наступним кроком проведемо класифікацію методом найближчого сусіда, застосувавши написаний раніше класифікатор, але подавши на вхід редуковані масиви дескрипторів та мітки класів. Оскільки кількість ознак в еталонах зменшилась в 10 разів, то мінімальна кількість голосів для успішної класифікації буде складати 25. Решта параметрів залишились без змін. У таблиці 3.3 наведено результати класифікації редукованих даних.

Таблиця 3.3 – Результат класифікації редукованих даних (50 дескрипторів) методом найближчого сусіда

| № | Розподіл голосів | Коефіцієнт впевненості | Час класифікації | Результат |
|---|---|---|---|---|
| **1** | **2** | **3** | **4** | **5** |
| 1 | 40 – 9 – 2 – 0 – 7 | 0,225 | 0,02104 | + |
| 2 | 39 – 16 – 4 – 2 – 8 | 0,41026 | 0,01648 | + |
| 3 | 43 – 13 – 1 – 2 – 7 | 0,30233 | 0,01645 | + |
| 4 | 38 – 13 – 2 – 0 – 7 | 0,34211 | 0,01617 | + |
| 5 | 41 – 13 – 3 – 4 – 6 | 0,31707 | 0,01766 | + |
| 6 | 40 – 15 – 3 – 2 – 9 | 0,375 | 0,01632 | + |
| 7 | 4 – 29 – 0 – 1 – 1 | 0,13793 | 0,01655 | + |
| 8 | 2 – 31 – 2 – 2 – 0 | 0,06452 | 0,01736 | + |
| 9 | 7 – 35 – 1 – 3 – 1 | 0,2 | 0,01611 | + |
| 10 | 3 – 33 – 0 – 0 – 2 | 0,09091 | 0,01867 | + |
| 11 | 3 – 38 – 0 – 1 – 3 | 0,07895 | 0,01631 | + |
| 12 | 0 – 35 – 2 – 3 – 2 | 0,08571 | 0,01593 | + |
| 13 | 5 – 12 – 37 – 2 – 3 | 0,32432 | 0,01688 | + |
| 14 | 6 – 12 – 38 – 3 – 4 | 0,31579 | 0,01830 | + |
| 15 | 6 – 10 – 37 – 1 – 4 | 0,27027 | 0,01567 | + |
| 16 | 8 – 7 – 39 – 2 – 6 | 0,20513 | 0,01562 | + |
| 17 | 8 – 12 – 35 – 2 – 4 | 0,34286 | 0,01569 | + |
| 18 | 7 – 8 – 38 – 3 – 7 | 0,21053 | 0,01609 | + |
| 19 | 3 – 3 – 0 – 32 – 2 | 0,09375 | 0,01616 | + |
| 20 | 13 – 5 – 2 – 38 – 6 | 0,34211 | 0,01618 | + |
| 21 | 3 – 7 – 3 – 32 – 1 | 0,21875 | 0,01718 | + |
| 22 | 11 – 5 – 2 – 41 – 4 | 0,26829 | 0,02870 | + |
| 23 | 3 – 4 – 1 – 33 – 0 | 0,12121 | 0,02828 | + |
| 24 | 11 – 7 – 5 – 35 – 2 | 0,31429 | 0,03349 | + |
| 25 | 9 – 5 – 7 – 1 – 26 | 0,34615 | 0,02745 | + |
| 26 | 10 – 12 – 3 – 6 – 30 | 0,4 | 0,02118 | + |
| 27 | 7 – 7 – 5 – 2 – 29 | 0,24138 | 0,01581 | + |
| 28 | 7 – 10 – 5 – 6 – 33 | 0,30303 | 0,01644 | + |
| 29 | 10 – 10 – 6 – 2 – 31 | 0,32258 | 0,01573 | + |
| 30 | 6 – 9 – 5 – 6 – 33 | 0,27273 | 0,01597 | + |
| 31 | 9 – 16 – 5 – 4 – 9 | 0,5625 | 0,01562 | + |
| 32 | 16 – 16 – 3 – 3 – 4 | 1 | 0,01563 | + |
| 33 | 10 – 17 – 6 – 7 – 8 | 0,58824 | 0,01558 | + |
| 34 | 12 – 16 – 4 – 3 – 7 | 0,75 | 0,02184 | + |
| 35 | 15 – 16 – 5 – 7 – 8 | 0,9375 | 0,01731 | + |
| 36 | 13 – 17 – 4 – 4 – 8 | 0,76471 | 0,01802 | + |

Продовження таблиці 3.3

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 37 | 11 – 21 – 5 – 7 – 8 | 0,52381 | 0,02654 | + |
| 38 | 7 – 11 – 5 – 6 – 11 | 1 | 0,01943 | + |
| 39 | 14 – 7 – 5 – 3 – 3 | 0,5 | 0,01816 | + |
| 40 | 9 – 9 – 7 – 6 – 6 | 1 | 0,01565 | + |
| 41 | 8 – 10 – 5 – 2 – 6 | 0,8 | 0,01860 | + |
| 42 | 10 – 10 – 5 – 5 – 7 | 1 | 0,01596 | + |
| 43 | 14 – 8 – 7 – 7 – 8 | 0,57143 | 0,01624 | + |
| 44 | 10 – 11 – 6 – 4 – 8 | 0,90909 | 0,01673 | + |
| 45 | 8 – 16 – 5 – 7 – 9 | 0,56250 | 0,01624 | + |
| 46 | 15 – 17 – 8 – 7 – 13 | 0,88235 | 0,01642 | + |
| 47 | 12 – 19 – 7 – 6 – 10 | 0,63158 | 0,01935 | + |
| 48 | 11 – 16 – 7 – 5 – 10 | 0,6875 | 0,01641 | + |
| 49 | 11 – 16 – 5 – 6 – 11 | 0,6875 | 0,01835 | + |
| 50 | 13 – 23 – 6 – 8 – 9 | 0,56522 | 0,01612 | + |
| 51 | 13 – 15 – 5 – 7 – 11 | 0,86667 | 0,01611 | + |

Точність класифікації у порівнянні з класичним методом найближчого сусіда залишилась на рівні 100%. При цьому кількість дескрипторів необхідних для зберігання опису одного об’єкта знизилась в 10 разів – з 500 до 50. Середній час класифікації одного об’єкта знизився приблизно в 11 разів і склав 0,018 с, що є суттєвим покращенням швидкодії. Середнє значення коефіцієнта впевненості для об’єктів, які входять до еталонної вибірки, складає 0,25, а для об’єктів, які відсутні в еталонній вибірці – 0,752. Тобто через стиснення описів трохи погіршились коефіцієнти впевненості, але при цьому вони залишились на достатньому рівні для успішної класифікації вхідних зображень.

Для подальшого аналізу побудуємо гістограми для результатів класифікації першого протестованого зображення з числа еталонних до редукції та після, а також першого протестованого зображення з числа не еталонних (рис. 3.8).

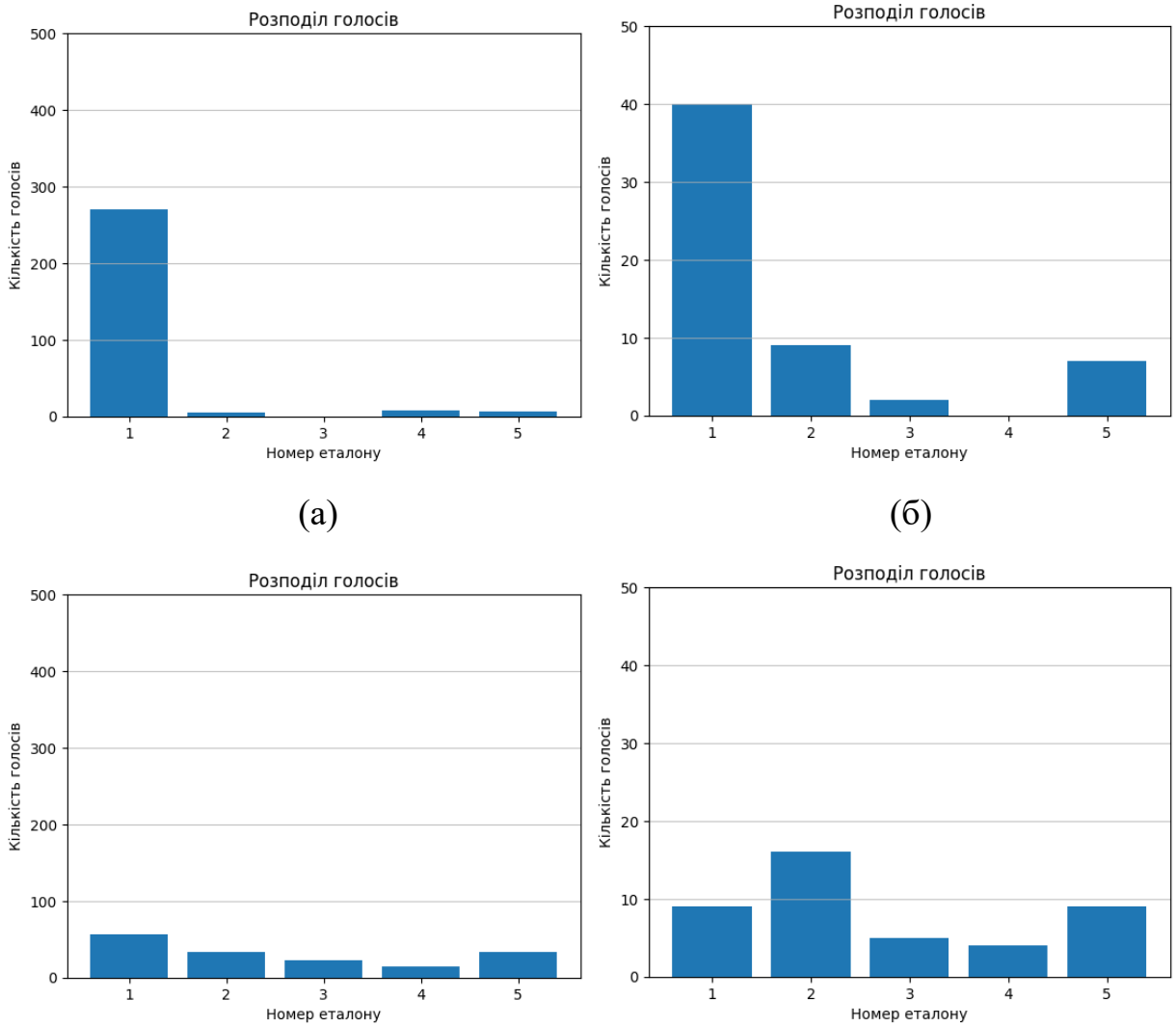


Рисунок 3.8 – Гістограми розподілу голосів:

(а) зображення з множини еталонних, до редукції;

(б) зображення з множини еталонних, після редукції;

(в) зображення не з множини еталонних, до редукції;

(г) зображення не з множини еталонних, після редукції

На гістограмах добре видно, що в процентному відношенні кількість голосів за класи, до якого об’єкт не відноситься, трохи зросла. Але й відносно зросла кількість голосів за правильний клас. В підсумку це дає можливість отримати правильний результат класифікації, скоротивши при цьому витрати на зберігання та швидкодію на порядок.

Спробуємо покращити ці результати ще більше, збільшивши ступінь стиснення. Здійснимо редукцію до 25 дескрипторів. Це можна зробити двома шляхами: провести повторну редукцію вже редукованої раніше множини на 500 дескрипторів, перерахувавши значення інформативностей, або виконати редукцію оригінальної множини на 2500 дескрипторів. Проведемо експеримент, щоб визначити, який варіант дає кращі результати.

В таблиці 3.4 відображені результати класифікації після подвійної редукції. Мінімально необхідна кількість голосів – 13.

Таблиця 3.4 – Результат класифікації редукованих даних (25 дескрипторів, подвійна редукція) методом найближчого сусіда

| № | Розподіл голосів | Коефіцієнт впевненості | Час класифікації | Результат |
|---|---|---|---|---|
| **1** | **2** | **3** | **4** | **5** |
| 1 | 19 – 3 – 0 – 0 – 7 | 0,36842 | 0,01405 | + |
| 2 | 20 – 9 – 0 – 1 – 4 | 0,45 | 0,00814 | + |
| 3 | 19 – 6 – 1 – 3 – 5 | 0,31579 | 0,00896 | + |
| 4 | 20 – 5 – 0 – 0 – 5 | 0,25 | 0,01541 | + |
| 5 | 20 – 7 – 2 – 1 – 8 | 0,4 | 0,01248 | + |
| 6 | 19 – 7 – 1 – 2 – 7 | 0,36842 | 0,00801 | + |
| 7 | 3 – 16 – 0 – 2 – 2 | 0,1875 | 0,00791 | + |
| 8 | 3 – 13 – 1 – 1 – 1 | 0,23077 | 0,00792 | + |
| 9 | 6 – 19 – 1 – 3 – 0 | 0,31579 | 0,00854 | + |
| 10 | 2 – 16 – 0 – 0 – 1 | 0,125 | 0,00842 | + |
| 11 | 4 – 18 – 0 – 2 – 2 | 0,22222 | 0,00840 | + |
| 12 | 1 – 16 – 2 – 3 – 1 | 0,1875 | 0,00847 | + |
| 13 | 3 – 6 – 19 – 2 – 2 | 0,31579 | 0,00848 | + |
| 14 | 4 – 7 – 17 – 1 – 4 | 0,41176 | 0,00840 | + |
| 15 | 5 – 7 – 18 – 0 – 3 | 0,38889 | 0,00847 | + |
| 16 | 6 – 4 – 18 – 1 – 3 | 0,33333 | 0,00848 | + |
| 17 | 6 – 7 – 16 – 2 – 4 | 0,4375 | 0,00851 | + |
| 18 | 6 – 7 – 20 – 2 – 6 | 0,35 | 0,00897 | + |
| 19 | 1 – 1 – 0 – 18 – 1 | 0,05556 | 0,00888 | + |
| 20 | 9 – 3 – 0 – 19 – 5 | 0,47368 | 0,00850 | + |
| 21 | 2 – 3 – 1 – 19 – 1 | 0,15789 | 0,00912 | + |
| 22 | 8 – 4 – 0 – 20 – 3 | 0,4 | 0,00861 | + |
| 23 | 2 – 2 – 0 – 19 – 1 | 0,10526 | 0,01094 | + |

Продовження таблиці 3.4

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 24 | 9 – 4 – 1 – 20 – 4 | 0,45 | 0,00901 | + |
| 25 | 5 – 2 – 3 – 0 – 15 | 0,33333 | 0,00857 | + |
| 26 | 6 – 6 – 2 – 3 – 14 | 0,42857 | 0,00865 | + |
| 27 | 4 – 3 – 2 – 2 – 13 | 0,30769 | 0,00881 | + |
| 28 | 3 – 8 – 3 – 4 – 16 | 0,5 | 0,00873 | + |
| 29 | 7 – 6 – 3 – 2 – 16 | 0,4375 | 0,00845 | + |
| 30 | 4 – 5 – 2 – 5 – 16 | 0,3125 | 0,00848 | + |
| 31 | 6 – 9 – 2 – 1 – 6 | 0,66667 | 0,00845 | + |
| 32 | 9 – 5 – 2 – 2 – 3 | 0,55556 | 0,00910 | + |
| 33 | 7 – 11 – 4 – 3 – 5 | 0,63636 | 0,00839 | + |
| 34 | 8 – 10 – 1 – 1 – 5 | 0,8 | 0,00861 | + |
| 35 | 9 – 11 – 2 – 4 – 5 | 0,81818 | 0,00841 | + |
| 36 | 7 – 7 – 3 – 4 – 6 | 1 | 0,00845 | + |
| 37 | 8 – 10 – 3 – 3 – 5 | 0,8 | 0,00830 | + |
| 38 | 4 – 5 – 2 – 4 – 6 | 0,83333 | 0,00926 | + |
| 39 | 7 – 3 – 3 – 2 – 1 | 0,42857 | 0,00848 | + |
| 40 | 6 – 4 – 3 – 4 – 5 | 0,83333 | 0,00839 | + |
| 41 | 5 – 5 – 3 – 1 – 4 | 1 | 0,01046 | + |
| 42 | 7 – 6 – 1 – 4 – 4 | 0,85714 | 0,00846 | + |
| 43 | 8 – 3 – 4 – 5 – 3 | 0,625 | 0,00842 | + |
| 44 | 7 – 7 – 2 – 3 – 5 | 1 | 0,00851 | + |
| 45 | 8 – 9 – 2 – 4 – 6 | 0,88889 | 0,00835 | + |
| 46 | 9 – 7 – 5 – 3 – 7 | 0,77778 | 0,00832 | + |
| 47 | 8 – 12 – 3 – 4 – 5 | 0,66667 | 0,01199 | + |
| 48 | 7 – 12 – 5 – 3 – 6 | 0,58333 | 0,00777 | + |
| 49 | 9 – 9 – 3 – 5 – 6 | 1 | 0,00830 | + |
| 50 | 5 – 10 – 5 – 4 – 6 | 0,6 | 0,00834 | + |
| 51 | 10 – 8 – 1 – 5 – 8 | 0,8 | 0,00859 | + |

При зниженні кількості дескрипторів до 25 на опис одного зображення точність склала все ті ж 100%. При цьому середній час класифікації одного об’єкта склав вже 0,009 с, що є покращенням в 20 разів відносно класичного методу. Середнє значення коефіцієнта впевненості для об’єктів, які входять до еталонної вибірки, складає 0,32, що є незначним погіршенням відносно редукованої до 50 ознак множини, а для об’єктів, які відсутні в еталонній вибірці – 0,77.

В таблиці 3.5 наведені результати класифікації з тією ж кількістю дескрипторів, але редукція проводилась одноразово з 500 ознак до 25. Параметри класифікатора залишились без змін.

Таблиця 3.5 – Результат класифікації редукованих даних (25 дескрипторів, первинна редукція) методом найближчого сусіда

| № | Розподіл голосів | Коефіцієнт впевненості | Час класифікації | Результат |
|---|---|---|---|---|
| **1** | **2** | **3** | **4** | **5** |
| 1 | 20 – 2 – 0 – 1 – 4 | 0,2 | 0,01396 | + |
| 2 | 21 – 6 – 1 – 1 – 6 | 0,28571 | 0,00828 | + |
| 3 | 21 – 4 – 0 – 1 – 4 | 0,19048 | 0,00798 | + |
| 4 | 20 – 4 – 0 – 0 – 4 | 0,2 | 0,00811 | + |
| 5 | 20 – 4 – 2 – 2 – 5 | 0,25 | 0,00797 | + |
| 6 | 23 – 3 – 0 – 1 – 5 | 0,21739 | 0,00796 | + |
| 7 | 3 – 16 – 0 – 1 – 3 | 0,1875 | 0,00785 | + |
| 8 | 0 – 17 – 2 – 1 – 0 | 0,11765 | 0,01303 | + |
| 9 | 6 – 20 – 1 – 1 – 1 | 0,3 | 0,01558 | + |
| 10 | 2 – 16 – 0 – 0 – 2 | 0,125 | 0,00806 | + |
| 11 | 4 – 20 – 0 – 1 – 2 | 0,2 | 0,00822 | + |
| 12 | 0 – 16 – 2 – 1 – 2 | 0,125 | 0,00840 | + |
| 13 | 3 – 3 – 20 – 1 – 1 | 0,15 | 0,00849 | + |
| 14 | 3 – 5 – 20 – 1 – 4 | 0,25 | 0,00852 | + |
| 15 | 4 – 5 – 20 – 0 – 1 | 0,25 | 0,00836 | + |
| 16 | 4 – 4 – 20 – 0 – 3 | 0,2 | 0,00856 | + |
| 17 | 5 – 3 – 18 – 0 – 4 | 0,27778 | 0,00873 | + |
| 18 | 4 – 4 – 23 – 2 – 8 | 0,34783 | 0,00865 | + |
| 19 | 1 – 2 – 0 – 19 – 1 | 0,10526 | 0,01105 | + |
| 20 | 9 – 4 – 0 – 19 – 6 | 0,47368 | 0,00846 | + |
| 21 | 3 – 2 – 1 – 17 – 1 | 0,17647 | 0,00865 | + |
| 22 | 10 – 4 – 0 – 23 – 4 | 0,43478 | 0,00909 | + |
| 23 | 2 – 1 – 0 – 17 – 0 | 0,11765 | 0,01198 | + |
| 24 | 7 – 6 – 1 – 21 – 4 | 0,33333 | 0,01102 | + |
| 25 | 6 – 2 – 3 – 0 – 15 | 0,4 | 0,00809 | + |
| 26 | 7 – 7 – 2 – 3 – 16 | 0,4375 | 0,00787 | + |
| 27 | 6 – 3 – 2 – 0 – 15 | 0,4 | 0,00825 | + |
| 28 | 6 – 5 – 3 – 2 – 17 | 0,35294 | 0,00798 | + |
| 29 | 5 – 5 – 3 – 1 – 16 | 0,3125 | 0,00797 | + |
| 30 | 3 – 5 – 3 – 6 – 16 | 0,375 | 0,00785 | + |
| 31 | 7 – 8 – 1 – 3 – 5 | 0,875 | 0,00861 | + |

Продовження таблиці 3.5

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 32 | 12 – 10 – 2 – 2 – 3 | 0,83333 | 0,00943 | + |
| 33 | 10 – 10 – 5 – 5 – 5 | 1 | 0,00821 | + |
| 34 | 8 – 8 – 2 – 3 – 5 | 1 | 0,00818 | + |
| 35 | 10 – 9 – 3 – 5 – 4 | 0,9 | 0,00852 | + |
| 36 | 10 – 7 – 2 – 3 – 5 | 0,7 | 0,00860 | + |
| 37 | 9 – 9 – 3 – 5 – 5 | 1 | 0,00837 | + |
| 38 | 2 – 4 – 2 – 4 – 6 | 0,66667 | 0,00849 | + |
| 39 | 9 – 3 – 3 – 2 – 2 | 0,33333 | 0,01036 | + |
| 40 | 5 – 4 – 4 – 4 – 6 | 0,83333 | 0,00887 | + |
| 41 | 6 – 5 – 1 – 1 – 4 | 0,83333 | 0,00853 | + |
| 42 | 8 – 4 – 2 – 3 – 5 | 0,625 | 0,00842 | + |
| 43 | 7 – 3 – 4 – 2 – 5 | 0,71429 | 0,00867 | + |
| 44 | 6 – 5 – 2 – 4 – 7 | 0,85714 | 0,00871 | + |
| 45 | 9 – 6 – 2 – 5 – 6 | 0,66667 | 0,00855 | + |
| 46 | 7 – 5 – 4 – 6 – 8 | 0,875 | 0,01188 | + |
| 47 | 9 – 8 – 3 – 6 – 5 | 0,88889 | 0,00871 | + |
| 48 | 6 – 9 – 4 – 5 – 7 | 0,77778 | 0,00837 | + |
| 49 | 9 – 7 – 3 – 6 – 7 | 0,77778 | 0,00850 | + |
| 50 | 7 – 8 – 4 – 7 – 8 | 1 | 0,00908 | + |
| 51 | 9 – 8 – 1 – 7 – 5 | 0,88889 | 0,01059 | + |

При даному підході точність становить ті ж самі 100%. Час класифікації, очевидно, залишився на тому ж рівні, оскільки кількість даних не змінилась. Але дещо покращились середні значення коефіцієнтів впевненості: для об’єктів, які входять до еталонної вибірки, він склав 0,26, що фактично на тому ж рівні, що й при редукції множини до 50 ознак, для об’єктів, які відсутні в еталонній вибірці він склав 0,81. Тому можна зробити висновок, що редукція ознак дає кращі результати, якщо проводити її в один етап одразу до потрібного ступеня стиснення.

Спробуємо стиснути дані ще більше до 10 ознак. В таблиці 3.6 наведено результати роботи класифікатору при редукції еталонних ознак до 10 з мінімальним порогом голосів – 6.

Таблиця 3.6 – Результат класифікації редукованих даних (10 дескрипторів) методом найближчого сусіда

| № | Розподіл голосів | Коефіцієнт впевненості | Час класифікації | Результат |
|---|---|---|---|---|
| **1** | **2** | **3** | **4** | **5** |
| 1 | 8 – 1 – 0 – 0 – 3 | 0,375 | 0,00598 | + |
| 2 | 9 – 4 – 0 – 0 – 3 | 0,44444 | 0,00780 | + |
| 3 | 9 – 3 – 1 – 1 – 1 | 0,33333 | 0,00461 | + |
| 4 | 9 – 3 – 0 – 0 – 2 | 0,33333 | 0,00410 | + |
| 5 | 9 – 3 – 2 – 0 – 2 | 0,33333 | 0,00408 | + |
| 6 | 9 – 4 – 0 – 1 – 3 | 0,44444 | 0,00394 | + |
| 7 | 0 – 8 – 0 – 0 – 0 | 0 | 0,00405 | + |
| 8 | 1 – 7 – 1 – 0 – 0 | 0,14286 | 0,00397 | + |
| 9 | 3 – 8 – 0 – 0 – 0 | 0,375 | 0,00397 | + |
| 10 | 1 – 8 – 0 – 0 – 0 | 0,125 | 0,00408 | + |
| 11 | 1 – 8 – 0 – 0 – 0 | 0,125 | 0,00361 | + |
| 12 | 0 – 6 – 1 – 0 – 1 | 0,16667 | 0,00406 | + |
| 13 | 1 – 0 – 8 – 0 – 0 | 0,125 | 0,00400 | + |
| 14 | 1 – 1 – 9 – 0 – 1 | 0,11111 | 0,00404 | + |
| 15 | 2 – 2 – 8 – 0 – 1 | 0,25 | 0,00396 | + |
| 16 | 1 – 1 – 8 – 0 – 1 | 0,125 | 0,00400 | + |
| 17 | 3 – 1 – 5 – 0 – 2 | 0,6 | 0,00402 | – |
| 18 | 1 – 1 – 9 – 1 – 4 | 0,44444 | 0,00404 | + |
| 19 | 0 – 0 – 0 – 8 – 0 | 0 | 0,00402 | + |
| 20 | 4 – 2 – 0 – 7 – 3 | 0,57143 | 0,00404 | + |
| 21 | 2 – 1 – 1 – 7 – 0 | 0,28571 | 0,00405 | + |
| 22 | 7 – 1 – 0 – 10 – 2 | 0,7 | 0,00407 | – |
| 23 | 1 – 0 – 0 – 7 – 0 | 0,14286 | 0,00394 | + |
| 24 | 4 – 2 – 1 – 10 – 3 | 0,4 | 0,00404 | + |
| 25 | 4 – 2 – 3 – 0 – 4 | 1 | 0,00399 | – |
| 26 | 4 – 2 – 1 – 0 – 6 | 0,66667 | 0,00411 | – |
| 27 | 3 – 2 – 1 – 0 – 3 | 1 | 0,00404 | – |
| 28 | 3 – 1 – 2 – 0 – 8 | 0,375 | 0,00408 | + |
| 29 | 3 – 2 – 2 – 0 – 5 | 0,6 | 0,00412 | – |
| 30 | 2 – 2 – 3 – 1 – 6 | 0,5 | 0,00410 | + |
| 31 | 4 – 3 – 1 – 1 – 1 | 0,75 | 0,00400 | + |
| 32 | 5 – 3 – 2 – 1 – 2 | 0,6 | 0,00396 | + |
| 33 | 4 – 5 – 1 – 2 – 1 | 0,8 | 0,00398 | + |
| 34 | 4 – 3 – 1 – 1 – 3 | 0,75 | 0,00403 | + |
| 35 | 5 – 5 – 1 – 2 – 2 | 1 | 0,00416 | + |
| 36 | 6 – 2 – 2 – 2 – 1 | 0,33333 | 0,00400 | – |

Продовження таблиці 3.6

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 37 | 4 – 3 – 1 – 2 – 3 | 0,75 | 0,00438 | + |
| 38 | 2 – 2 – 0 – 0 – 3 | 0,66667 | 0,00412 | + |
| 39 | 4 – 2 – 2 – 0 – 1 | 0,5 | 0,00431 | + |
| 40 | 4 – 2 – 3 – 1 – 3 | 0,75 | 0,00405 | + |
| 41 | 2 – 3 – 1 – 0 – 2 | 0,66667 | 0,00407 | + |
| 42 | 4 – 3 – 0 – 1 – 3 | 0,75 | 0,00440 | + |
| 43 | 3 – 4 – 2 – 1 – 2 | 0,75 | 0,00422 | + |
| 44 | 2 – 4 – 1 – 2 – 3 | 0,75 | 0,00425 | + |
| 45 | 4 – 4 – 1 – 2 – 3 | 1 | 0,00430 | + |
| 46 | 4 – 3 – 2 – 3 – 2 | 0,75 | 0,00420 | + |
| 47 | 5 – 5 – 2 – 2 – 3 | 1 | 0,00448 | + |
| 48 | 3 – 5 – 2 – 1 – 3 | 0,6 | 0,00541 | + |
| 49 | 4 – 4 – 2 – 3 – 3 | 1 | 0,00636 | + |
| 50 | 4 – 4 – 3 – 4 – 3 | 1 | 0,00362 | + |
| 51 | 5 – 5 – 0 – 3 – 2 | 1 | 0,00373 | + |

При редукції до 10 ознак класифікатор почав відчувати проблеми с визначенням класів. З 51 тестового зображення правильно було класифіковано 44, що дає точність на рівні 86%, що все ще можна вважати високим результатом, особливо беручи до уваги, що уся еталонна база тепер налічує 50 елементів, що складає лише 2% від розміру оригінальної бази, при цьому приріст швидкодії по відношенню до класичного методу зріс у неймовірних 50 разів та склав 0,004 с в середньому на один об’єкт. В залежності від вимог до класифікатора, між точністю та швидкодією можна балансувати за рахунок зниження або збільшення ступеню редукції. Також варто відзначити, що від самого початку до класифікатору ставилось вимога необхідності розпізнавання об’єктів, які не входять до еталонної бази. Якщо це не є вимогою конкретної задачі, то точність класифікатора можна покращити ще більше за рахунок налаштування вищого коефіцієнта впевненості, при цьому швидкість класифікації залишиться на тому ж високому рівні.

Спробуємо довести ефективність редукції на основі параметрів значущості за допомогою наступного експерименту: порівняємо статистичні

показники класифікації редукованих даних з оригінальними даними, з яких випадковим чином було вибрано деяку кількість дескрипторів. Результати порівняння занесені у таблицю 3.7.

Таблиця 3.7 – Результат порівняння класифікації редукції на основі інформативності та випадкової редукції

| Назва параметру | Редукція на основі інформативності | | | Випадкова редукція | | |
|---|---|---|---|---|---|---|
| | 50 | 25 | 10 | 50 | 25 | 10 |
| Точність | 100% | 100% | 86% | 86% | 68% | 46% |
| Середній коефіцієнт впевненості для об’єктів з еталонної вибірки | 0,25 | 0,26 | 0,33 | 0,38 | 0,44 | 0,58 |
| Середній коефіцієнт впевненості для об’єктів не з бази | 0,75 | 0,81 | 0,8 | 0,61 | 0,55 | 0,55 |

Можна побачити, що точність при редукції на основі інформативності вище, при цьому чим більший ступінь стиснення, тим більша різниця у точності. Також, що досить важливо, коефіцієнти впевненості для об’єктів з еталонної вибірки мають нижчі значення. Це означає, що класифікатор більш впевнений у правильності рішення при присвоєнні мітки класу об’єкту при класифікації множини, редукованої з використанням параметрів значущості. Це може мати вирішальну роль при класифікації наборів даних, об’єкти яких мають високу подібність між собою.

Проведений експеримент з редукцією до 10 ознак показав, що цієї кількості вже недостатньо для безпомилкової роботи класифікатора. Спробуємо покращити цю ситуацію за рахунок введення вагових коефіцієнтів, заснованих на інформативності дескрипторів, що підвищить вплив більш значущих ознак на результати голосування. При високих ступенях стиснення це може мати суттєвий внесок у точність класифікатора.

Після проведення редукції перерахуємо значення інформативностей для усіх дескрипторів, що залишились. Визначимо мінімальне та

максимальне значення інформативності для усіх еталонних ознак, та поділимо їх на 4 однакові інтервали. Далі кожному дескриптору буде призначене певне значення ваги в залежності від інтервалу, в який входить його інформативність. Значення ваг – це цілі числа у діапазоні від 1 до 4. Далі йде код, який реалізує описаний вище підхід.

Лістинг 3.6 Функція для визначення вагомості ознак:

```
def make_even_intervals(start, end, num_intervals):
  return np.linspace(start, end+1, num=num_intervals+1, dtype=int)


def calculate_weights(descriptors, labels, num_intervals=4, desc_order=True):
  importance = calculate_importance(descriptors, labels)

  bin_edges = make_even_intervals(np.min(importance), np.max(importance), num_intervals)

  if desc_order:
    bin_weights = np.arange(num_intervals, 0, -1) # desc
  else:
    bin_weights = np.arange(1, num_intervals+1) # asc order (higher importance -> higher weight)

  weights = np.empty(descriptors.shape[0])

  for i in range(len(bin_edges)-1):
    weights[(importance >= bin_edges[i]) & (importance < bin_edges[i+1])] = bin_weights[i]
```

*return weights, importance*

Дана функція розраховує значення інформативностей та відповідних їм вагових коефіцієнтів та повертає їх в якості результату. На рисунку 3.9 зображено результат застосування цієї функції до наявної еталонної вибірки після редукції до 10 дескрипторів.

```
Номер класу: 0
Інформативність дескрипторів:
[ -3.   3.  75.  -8.  80.  47. -19.  26.  60. -15.]
Вагові коефіцієнти:
[3. 3. 1. 3. 1. 2. 4. 2. 1. 3.]

Номер класу: 1
Інформативність дескрипторів:
[-25.  10.  17.  -8. -18. -24.  -7.  49.   5.  46.]
Вагові коефіцієнти:
[4. 3. 2. 3. 4. 4. 3. 1. 3. 2.]

Номер класу: 2
Інформативність дескрипторів:
[46. 37. 45. 38. 37. 47. 71. 46. 54. 61.]
Вагові коефіцієнти:
[2. 2. 2. 2. 2. 2. 1. 2. 1. 1.]

Номер класу: 3
Інформативність дескрипторів:
[-12.  31. -16.  -1.  -6. -25.  53.  57.  60.  60.]
Вагові коефіцієнти:
[3. 2. 4. 3. 3. 4. 1. 1. 1. 1.]

Номер класу: 4
Інформативність дескрипторів:
[ 40. -47.  47.  57.  45.  -1.  72. -26. -33.  79.]
Вагові коефіцієнти:
[2. 4. 2. 1. 2. 3. 1. 4. 4. 1.]
```

Рисунок 3.9 – Результат обчислення вагових коефіцієнтів

Щоб можна було застосувати ці коефіцієнти при класифікації, треба модифікувати вже наявний класифікатор методом найближчого сусіда. Далі наведено оновлений код функції класифікації, яка враховує вагові коефіцієнти при визначенні міток класів, розраховані по формули (2.27).

Лістинг 3.7 Функція класифікації з врахуванням вагових коефіцієнтів ознак:

```
def predict_class_weighted(sample, descriptors, labels, weights, min_votes, max_nearest_coef=0, max_distance=np.inf, cross_check=True):
    distances = pairwise_distances(sample, descriptors, metric='l1')
    weighted_distances = distances * weights

    indices1 = np.arange(sample.shape[0])
    indices2 = np.argmin(weighted_distances, axis=1)

    if cross_check:
        matches1 = np.argmin(weighted_distances, axis=0)
        mask = indices1 == matches1[indices2]
        indices1 = indices1[mask]
        indices2 = indices2[mask]

    if max_distance < np.inf:
        mask = distances[indices1, indices2] <= max_distance
        indices1 = indices1[mask]
        indices2 = indices2[mask]

    # count number of indices for each class
    filtered_labels = labels[indices2]
    class_indices, match_counts = np.unique(filtered_labels, return_counts=True)

    # fill vote table
    vote_table = np.zeros(class_count)
    vote_table[class_indices] = match_counts
```

```
# calculate predicted label by sorting vote table and taking 1st element
sorted_votes = np.argsort(-vote_table)
predicted_label = sorted_votes[0]

# calculate nearest neighboor metric
nearest_max = vote_table[sorted_votes[1]]
nearest_max_coef = nearest_max / vote_table[predicted_label]

if vote_table[predicted_label] < min_votes:
  predicted_label = -1

if max_nearest_coef > 0 and nearest_max_coef >= max_nearest_coef:
  predicted_label = -1

return predicted_label, vote_table, nearest_max_coef
```

Результат класифікації редукованих даних з використанням вагових коефіцієнтів за допомогою оновленої функції класифікації наведений у таблиці 3.8. Оскільки з введенням ваг середня кількість голосів у розподілах знизилась, то мінімальний поріг голосив було також знижено до 3.

Таблиця 3.8 – Результат класифікації редукованих даних (10 дескрипторів) з використанням вагових коефіцієнтів.

| № | Розподіл голосів | Коефіцієнт впевненості | Час класифікації | Результат |
|---|---|---|---|---|
| **1** | **2** | **3** | **4** | **5** |
| 1 | 7 – 1 – 0 – 0 – 0 | 0,14286 | 0,00786 | + |
| 2 | 8 – 1 – 0 – 0 – 0 | 0,125 | 0,00855 | + |
| 3 | 7 – 1 – 1 – 1 – 0 | 0,14286 | 0,00496 | + |
| 4 | 8 – 1 – 0 – 0 – 0 | 0,125 | 0,00504 | + |

Продовження таблиці 3.8

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 5 | 7 – 1 – 1 – 0 – 0 | 0,14286 | 0,00517 | + |
| 6 | 7 – 1 – 0 – 0 – 0 | 0,14286 | 0,00519 | + |
| 7 | 0 – 4 – 0 – 0 – 0 | 0 | 0,00485 | + |
| 8 | 0 – 5 – 0 – 0 – 0 | 0 | 0,00507 | + |
| 9 | 0 – 5 – 0 – 0 – 0 | 0 | 0,00499 | + |
| 10 | 0 – 5 – 0 – 0 – 0 | 0 | 0,00494 | + |
| 11 | 0 – 6 – 0 – 0 – 0 | 0 | 0,00481 | + |
| 12 | 0 – 3 – 0 – 0 – 0 | 0 | 0,00508 | + |
| 13 | 0 – 0 – 7 – 0 – 0 | 0 | 0,00479 | + |
| 14 | 1 – 0 – 9 – 0 – 0 | 0,11111 | 0,00483 | + |
| 15 | 1 – 1 – 7 – 0 – 0 | 0,14286 | 0,00485 | + |
| 16 | 1 – 0 – 8 – 0 – 0 | 0,125 | 0,00482 | + |
| 17 | 1 – 0 – 5 – 0 – 1 | 0,2 | 0,00496 | + |
| 18 | 2 – 1 – 9 – 1 – 1 | 0,22222 | 0,00486 | + |
| 19 | 0 – 0 – 0 – 6 – 0 | 0 | 0,00488 | + |
| 20 | 1 – 0 – 0 – 3 – 0 | 0,33333 | 0,00484 | + |
| 21 | 0 – 0 – 1 – 5 – 0 | 0,2 | 0,00521 | + |
| 22 | 2 – 0 – 0 – 9 – 1 | 0,22222 | 0,00629 | + |
| 23 | 0 – 0 – 0 – 5 – 0 | 0 | 0,00751 | + |
| 24 | 1 – 0 – 0 – 6 – 1 | 0,16667 | 0,00694 | + |
| 25 | 1 – 1 – 1 – 0 – 3 | 0,33333 | 0,00441 | + |
| 26 | 1 – 1 – 0 – 0 – 5 | 0,2 | 0,00482 | + |
| 27 | 1 – 1 – 1 – 0 – 2 | 0,5 | 0,00503 | – |
| 28 | 2 – 0 – 2 – 0 – 6 | 0,33333 | 0,00510 | + |
| 29 | 1 – 1 – 0 – 0 – 3 | 0,33333 | 0,00497 | + |
| 30 | 1 – 1 – 1 – 0 – 4 | 0,25 | 0,00651 | + |
| 31 | 2 – 1 – 0 – 0 – 1 | 0,5 | 0,00512 | + |
| 32 | 2 – 1 – 0 – 0 – 1 | 0,5 | 0,00500 | + |
| 33 | 0 – 1 – 1 – 1 – 0 | 1 | 0,00503 | + |
| 34 | 1 – 1 – 0 – 0 – 1 | 1 | 0,00512 | + |
| 35 | 3 – 1 – 1 – 0 – 0 | 0,33333 | 0,00564 | – |
| 36 | 2 – 1 – 0 – 1 – 0 | 0,5 | 0,00441 | + |
| 37 | 1 – 1 – 1 – 1 – 2 | 0,5 | 0,00441 | + |
| 38 | 0 – 1 – 0 – 0 – 0 | 0 | 0,00458 | + |
| 39 | 0 – 1 – 1 – 0 – 0 | 1 | 0,00691 | + |
| 40 | 0 – 1 – 1 – 1 – 0 | 1 | 0,00599 | + |
| 41 | 0 – 1 – 0 – 0 – 0 | 0 | 0,00451 | + |
| 42 | 1 – 1 – 0 – 1 – 1 | 1 | 0,00533 | + |
| 43 | 0 – 1 – 1 – 1 – 0 | 1 | 0,00511 | + |
| 44 | 1 – 1 – 0 – 1 – 1 | 1 | 0,00688 | + |

Продовження таблиці 3.8

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 45 | 0 – 1 – 0 – 1 – 0 | 1 | 0,00636 | + |
| 46 | 1 – 1 – 0 – 1 – 0 | 1 | 0,00717 | + |
| 47 | 0 – 1 – 0 – 1 – 0 | 1 | 0,00714 | + |
| 48 | 1 – 1 – 0 – 0 – 1 | 1 | 0,00511 | + |
| 49 | 0 – 1 – 0 – 1 – 1 | 1 | 0,00517 | + |
| 50 | 0 – 1 – 0 – 1 – 0 | 1 | 0,00496 | + |
| 51 | 1 – 1 – 0 – 1 – 0 | 1 | 0,00486 | + |

З використанням вагових коефіцієнтів вдалося досягти точності класифікації 96%: неправильно було класифіковано лише 2 об’єкта з 51 – один з числа еталонних і один не з еталонного набору. Середній час класифікації трохи виріс у порівнянні з методом без використання ваг – з 0,004 с до 0,005 с. Середні коефіцієнти впевненості для об’єктів з числа еталонних також зазнали покращення – 0,15 при використанні ваг проти 0,33 без їх використання. Можна зробити висновок, що застосування вагових коефіцієнтів при класифікації дає серйозний приріст точності за рахунок незначного погіршення швидкодії.

# ВИСНОВКИ

У рамках кваліфікаційної роботи здійснено удосконалення структурних методів класифікації зображень шляхом впровадження редукції описів даних з використанням параметрів класифікаційної вагомості.

Застосування редукції для трансформації множини дескрипторів опису зображень дає можливість значно прискорити оброблення без втрати точності класифікації. Швидкодія оброблення збільшується пропорційно числу редукованих даних.

Побудований класифікатор дозволяє обирати баланс між точністю та швидкодією. Найбільш доцільним є використання 10% найбільш інформативних дескрипторів, що забезпечує приріст швидкодії в 10 разів при повному збереженні точності. Якщо швидкодія є ключовим критерієм, то допустимо використовувати навіть 5% від оригінальної кількості дескрипторів, що забезпечує приріст швидкодії майже у 20 разів. Використання додаткових вагових коефіцієнтів, основаних на критерії інформативності ознак, навіть при стисненні описів до такого ступеня дозволяє досягти показника точності у 96%.

Розроблений класифікатор дозволяє проводити тонке налаштування класифікаційних параметрів, що робить його придатним для використання з різноманітними еталонними наборами даних з різними характеристиками. Вибір правильних класифікаційних параметрів є критично важливим для досягнення високих показників продуктивності.

Результати дослідження апробовано у вигляді тез доповідей міжнародної науково-практичної конференції «Modern scientific technologies and solutions of scientists to create the latest ideas», 22-25 серпня 2023 року, Лондон, Велика Британія [47].

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Гороховатський В.О., Гадецька С.В. (2020). Статистичне оброблення та аналіз даних у структурних методах класифікації зображень: монографія. Харків, ФОП Панов А.Н., 128 с.

2. Гадецька С.В., Гороховатський В.О. (2018). Методи структурної класифікації зображень на засадах баєсовської теорії прийняття рішень. *Радіоелектроніка, інформатика, управління*, №2 (45), С. 90–97.

3. Gorokhovatskyi V.A. (2018). Image Classification Methods in the Space of Descriptions in the Form of a Set of the Key Point Descriptors, *Telecommunications and Radio Engineering*, 77 (9), pp. 787-797.

4. Гороховатський В.О., Творошенко І.С. (2022). Аналіз багатовимірних даних за описом у формі множини компонент: монографія. Харків, ХНУРЕ, 124 с.

5. Гороховатський В.О. (2014). Структурний аналіз та інтелектуальне оброблення даних в комп'ютерному зорі: монографія. Комп. СМИТ, 316с.

6. Гороховатський, В.О., Путятін, Е.П. (2008). Структурне розпізнавання зображень на основі моделей голосування ознак характерних точок, *Реєстрація, зберігання і обробка даних*, Т.10. №4, С. 75–85.

7. Tymchyshyn R. and al. (2018). Modern Approaches to Computer Vision, *Control systems and computers*, № 6, pp. 46-73.

8. Гороховатський В.О., Пупченко Д.В., Солодченко К.Г. (2018). Аналіз властивостей, характеристик та результатів застосування новітніх детекторів для визначення особливих точок зображення, Системи управління, навігації та зв'язку, С. 93–98.

9. Gorokhovatsky V.A. (2016). Efficient Estimation of Visual Object Relevance during Recognition through their Vector Descriptions, *Telecommunications and Radio Engineering*, Vol. 75, №14. pp. 1271–1283.

10. Yakovleva O., & Nikolaieva K. (2020). Research of descriptor based image normalization and comparative analysis of SURF, SIFT, BRISK, ORB, KAZE, AKAZE descriptors, *Advanced Information Systems*, 4 (4), pp. 89-101.

11. Gorokhovatskyi, V., Vlasenko, N. (2021). Редукція опису зображення у складі множини дескрипторів на основі метричного критерію інформативності, *Advanced Information Systems*, 5(4), pp. 10-16.

12. Gorokhovatskyi, V., Peredrii, O., Tvoroshenko, I., Markov, T. (2023). Матриця відстаней для множини компонентів структурного опису як інструмент для створення класифікатора зображень, *Advanced Information Systems*, 7 (1), С. 5–13.

13. Gadetska, S.V., Gorokhovatskyi, V. O., Stiahlyk, N. I., Vlasenko, N.V. (2021). Statistical data analysis tools in image classification methods based on the description as a set of binary descriptors of key points, *Radio Electronics, Computer Science, Control*, №4, pp. 58-68.

14. Gorokhovatsky V.A. & Putyatin Y. P. (2009). Image Likelihood Measures of the Basis of the Set of Conformities, *Telecommunications and Radio Engineering*, 68 (9), p. 763–778.

15. Гороховатский, В.А., Путятин, Е.П., Столяров В.С. (2017). Дослідження результативності структурних методів класифікації зображень, *Радіоелектроніка, інформатика, управління*, №3 (42), С. 78–85.

16. Kuchuk, H., Podorozhniak, A., Liubchenko, N., & Onishchenko, D. (2021). System of license plate recognition considering large camera shooting angles, *Radioelectronic and Computer Systems*, 4(100), pp. 82 –91.

17. Daradkeh, Y.I., Gorokhovatskyi, V., Tvoroshenko, I., Zeghid, M. (2022). Tools for Fast Metric Data Search in Structural Methods for Image Classification, *IEEE Access*, 10, pp. 124738-124746.

18. Gorokhovatskyi V., Tvoroshenko I., Kobylin O., Vlasenko N. (2023). Search for Visual Objects by Request in the Form of a Cluster Representation for the Structural Image Description, *Advances in Electrical and Electronic Engineering*, 21 (1), pp. 19–27.

19. Гороховатський В., Творошенко І., Сидоренко Д. (2021) Класифікація зображень із використанням кластерного подання, Міжн. наук. симпозіум Інтелектуальні рішення-С. Обчислювальний інтелект. Теорія прийняття рішень: праці міжн. наук. симп. (Вересень 29, 2021). Київ – Ужгород, 44-45.

20. Gonzalez-Ruiz, M., Diaz-Ramirez, V. & Juarez-Salazar, R. (2019). A comparative study of image feature detection and description methods for robot vision., Proc. SPIE 11136, Optics and Photonics for Information Processing XIII, 111360P.

21. Tosun, U. (2019). Comparative Analysis of the Feature Extraction Performance of Augmented Reality Algorithms, *Cumhuriyet Science Journal*, 40. pp. 958-966.

22. Tareen, S.A.K. & Saleem, Z. (2018). A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK, International Conference on Computing, Mathematics and Engineering Technologies (iCoMET).

23. Lowe, D. G. (1999). Object recognition from local scale-invariant features. Proceedings of the International Conference on Computer Vision. Vol. 2. pp. 1150–1157.

24. Xingteng, J., Xuan, W. & Zhe, D. (2016). Image matching method based on improved SURF algorithm, IEEE International Conference on Computer and Communications (ICCC), pp. 142-145.

25. Rosten, E., Porter, R. & Drummond, T. (2008). Faster and better: a machine learning approach to corner detection, IEEE Trans Pattern Anal Mach Intell, 32(1) pp. 105-119.

26. Calonder, M., Lepetit, V., Strecha, C., et al. (2010). BRIEF: Binary Robust Independent Elementary Features, Computer Vision-Eccv 2010, Pt Iv, 6314, pp. 778-792.

27. Rublee, E., Rabaud, V., Konolige, K. & Bradski, G. (2011). ORB: an efficient alternative to SIFT or SURF, *2011 International Conference on Computer Vision*, pp. 2564-2571.

28. Rosin, P. (1999). Measuring Corner Properties, *Computer Vision and Image Understanding*, №73, pp. 291-307.

29. Introduction to ORB (Oriented FAST and Rotated BRIEF). URL: https://medium.com/@deepanshut041/introduction-to-orb-oriented-fast-and-rotated-brief-4220e8ec40cf (дата звернення 14.10.2023).

30. Гороховатський В.О., Передрий Е.О. (2009) Кореляційні методи розпізнавання зображень шляхом голосування систем фрагментів. *Радіоелектроніка, інформатика, управління*, №1 (20), с.74–81.

31. Gadetska, S.V., Gorokhovatsky, V.O. (2018). Statistical Measures for Computation of the Image Relevance of Visual Objects in the Structural Image Classification Methods, *Telecommunications and Radio Engineering*, Vol. 77 (12), pp. 1041–1053.

32. Pomazan V., Tvoroshenko I., and Gorokhovatskyi V. (2023) Development of an application for recognizing emotions using convolutional neural networks, *International Journal of Academic Information Systems Research*, 7(7), 25–36.

33. Gorokhovatskyi, O., Peredrii, O., Gorokhovatskyi, V., Vlasenko, N. (2023). Explanation of CNN Image Classifiers with Hiding Parts. In: J. Benois-Pineau, R. Bourqui, D. Petkovic, G. Quenot (eds), *Explainable Deep Learning Artificial Intelligence*, pp. 125-146.

34. 2D Feature Tracking — Part 3: Feature Matching. URL: https://medium.com/@nikhilnair8490/2d-feature-tracking-part-3-feature-matching-72a1bc8dc58a (дата звернення 14.10.2023).

35. Local Feature Matching. URL: https://sites.cc.gatech.edu/classes/AY2016/cs4476_fall/results/proj2/html/alieberman3/ (дата звернення 14.10.2023).

36. Jolliffe, I. T., & Cadima, J. (2016). Principal component analysis: a review and recent developments. *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, 374(2065), 20150202.

37. Gadetska S., Gorokhovatskyi V., Stiahlyk N., Vlasenko N. (2022) Aggregate Parametric Representation of Image Structural Description in Statistical Classification Methods. In CEUR Workshop Proceedings: Computer Modeling and Intelligent Systems (CMIS-2022), 3137, pp. 68-77.

38. Gorokhovatskyi V., Gadetska S., Ponomarenko R. (2020) Recognition of Visual Objects Based on Statistical Distributions for Blocks of Structural Description of Image. Proc. of the XV Int. Scientific Conference «Intellectual Systems of Decision Making and Problems of Computational Intelligence» (ISDMCI'2019), Ukraine, May 21–25, 2019, pp. 501-512.

39. Gorokhovatskyi V., Gadetska S., Stiahlyk N. (2020) Image structural classification technologies based on statistical analysis of descriptions in the form of bit descriptor set. In CEUR Workshop Proceedings: Computer Modeling and Intelligent Systems (CMIS-2020). 2608. pp. 1027-1039.

40. Гороховатський В.О., Гадецька С.В., Стяглик Н.І., Власенко Н.В. (2020) Класифікація зображень на підставі ансамблю статистичних розподілів за класами еталонів для компонентів структурного опису. *Радіоелектроніка, інформатика, управління*, №4 , с. 85–94.

41. Luciano Ramalho (2022). Fluent Python: Clear, Concise, and Effective Programming 2nd Edition: study guide. *O'Reilly Media*.

42. Charles R Harris, K. Jarrod Millman, Stéfan J. van der Walt, et al. (2020). Array programming with NumPy, *Nature*, 585 (7825), pp. 357–362.

43. Adrian Kaehler, Gary Bradski (2018). Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library: study guide. *O'Reilly Media*.

44. Getting Started With Google Colab. URL: https://wiki.rit.edu/display/JupyterForTeaching/Getting+Started+With+Google+Colab (дата звернення 14.10.2023).

45. Національний музей історії України. URL: https://artsandculture.google.com/partner/national-museum-of-the-history-of-ukraine (дата звернення: 16.08.2023).

46. Xian, Y., Lampert, C. H., Schiele, B., & Akata, Z. (2018). Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly. *IEEE transactions on pattern analysis and machine intelligence*, *41*(9), 2251-2265.

47. Кулик О. Стиснення опису зображення на підставі параметру вагомості ознак/ О. Кулик // Proc. of XXXIII International Scientific and Practical Conference «Modern scientific technologies and solutions of scientists to create the latest ideas» (August 22-25, 2023), London, Great Britain, 295-299.

48. Гороховатський В.О., Гадецька С.В., Стяглик Н.І. (2019) Вивчення статистичних властивостей моделі блочного подання для множини дескрипторів ключових точок зображень. Радіоелектроніка, інформатика,управління, No. 2, с. 100–107.

49. Gorokhovatskiy, V.A. (2011) Compression of Descriptions in the Structural Image Recognition, Telecommunications and Radio Engineering, Vol. 70, No 15, pp. 1363-1371.

50. Gorokhovatskyi O., Gorokhovatskyi V., and Peredrii O. (2018) Analysis of Application of Cluster Descriptions in Space of Characteristic Image Features. Data, 3 (4), pp. 52.