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 перший (бакалаврський)

РОЗРОБЛЕННЯ МЕТОДУ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ ІЗ ВИКОРИСТАННЯМ КЛАСТЕРНИХ СТРУКТУР ДЛЯ ШВИДКОГО ПОШУКУ ДАНИХ
(тема)

Виконав:
студент 4 курсу, групи ІТІНФ-17-1
Чмутов Ю. В.
(прізвище, ініціали)

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

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)

2021 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти перший (бакалаврський)

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

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2021 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Чмутову Юрію Вадимовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення методу класифікації зображень із використанням кластерних структур для швидкого пошуку даних

затверджена наказом університету від « 20 » травня 2021 року № 663Ст.

2. Термін подання студентом роботи до екзаменаційної комісії 24 травня 2021 р.

3. Вихідні дані до роботи *Детектування ключових точок на зображенні, методи класифікації зображень за допомогою детекторів ключових точок, кластеризація даних, вхідні бази зображень*

4. Перелік питань, що потрібно опрацювати в роботі.
1. Моделювання методу швидкісного пошуку даних.
2. Бінарна обробка дескрипторів ключових точок.
3. Моделі обробки ключових точок зображення.
4. Програмна реалізація та аналіз результатів.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) *Дескриптор BRISK - схема вибірки BRISK, Результати роботи методів класифікації зображень, Матриця розподілу даних про кластерам, Масив центрів кластерів, Матриця відстаней між центрами кластерів та дескрипторами ключових точок, Розподіл КТ еталонів по класах.*

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	12.04.2021	виконано
2	Аналіз завдання, підбір літератури	12.04.21-14.04.21	виконано
3	Аналіз літератури з досліджуваної проблеми	15.04.21-17.04.21	виконано
4	Аналіз технічних засобів	18.04.21-19.04.21	виконано
5	Розробка методу	20.04.21-28.04.21	виконано
6	Програмна реалізація	29.04.21-03.05.21	виконано
7	Оформлення пояснювальної записки	04.05.21-13.05.21	виконано
8	Перевірка на плагіат	24.05.21	
9	Рецензування	25.06.21	
10	Підготовка презентації та доповіді	26.05.21	
11	Занесення роботи в електронний архів	28.05.21	
12	Попередній захист кваліфікаційної роботи	02.06.21	

Дата видачі завдання 12 квітня 2021 р.

Студент _____
(підпис)

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

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 66 с., 29 рис., 1 дод., 36 джерел.

КОМП'ЮТЕРНИЙ ЗІР, КЛАСИФІКАЦІЯ ОБ'ЄКТІВ, КЛАСТЕРНИЙ АНАЛІЗ, КЛЮЧОВА ТОЧКА, ДЕСКРИПТОР, ВІДСТАНЬ ГЕММІНГА, ШВИДКИЙ ПОШУК, БАЗА ЕТАЛОНІВ, КЛАС.

Об'єктом роботи є методи класифікації зображень у системах комп'ютерного зору.

Метою роботи є розроблення методу класифікації зображень з використанням засобів швидкісного пошуку на підставі кластерних структур.

Розроблено метод класифікації зображень із використанням кластерних структур для швидкісного пошуку на основі множини дескрипторів ключових точок. Ключові точки побудовані за допомогою детектору BRISK. Розбиття множини дескрипторів ключових точок по кластерах надає можливість суттєво прискорити обчислення, в результаті чого швидкодія класифікації значно зростає. Проведено вивчення властивостей засобів швидкісного пошуку.

Результати роботи апробовано у вигляді тез доповідей на Міжнародному молодіжному форумі «РАДІОЕЛЕКТРОНІКА І МОЛОДЬ У XXI СТОЛІТТІ».

COMPUTER VISION, OBJECT CLASSIFICATION, CLUSTER ANALYSIS, KEY POINT, DESCRIPTOR, HEMMING DISTANCE, QUICK SEARCH, STANDARD BASE, CLASS.

The object of the work is the process of developing a method for image classification.

The aim of the work is to develop a method of image classification using high-speed search tools based on cluster structures.

A method of image classification using cluster structures for high-speed search based on a set of key point descriptors has been developed. Key points are constructed using a BRISK detector. Partitioning the set of key point descriptors into clusters makes it possible to significantly speed up the calculation, resulting in a significant increase in the speed of classification. A study of the properties of high-speed search tools.

The results of the work were tested in the form of abstracts during the International Youth Forum «RADIO ELECTRONICS AND YOUTH IN THE XXI CENTURY».

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

КТ – ключові точки

FRiS – Function of Rival Similarity

CART – Classification And Regression Tree

FAST – Features from Accelerated Segment Test

BRIEF – Binary Robust Independent Elementary Features

МН – машинне навчання

BRISK – Binary Robust Invariant Scalable Keypoints

OpenCV – Open Source Computer Vision Library

SWIFT – Society for Worldwide Interbank Financial Telecommunications

SSE – Streaming SIMD Extensions

SIMD – Single Instruction, Multiple Data

ВСТУП

Задачу розпізнавання образів кожна людина у своєму житті розв’язує повсякденно. Читаючи текст, переходячи дорогу, ми розпізнаємо букви, цифри, колір сигналу світлофора, і цей процес здається настільки простим і природним, що ми навіть не замислюємося, що розв’язуємо якусь задачу. Взагалі, людина має дивовижну властивість досить легко розв’язувати задачі розпізнавання образів. Але з технічним прогресом і комп’ютеризацією суспільства виникла потреба автоматизації процесу розпізнавання. У результаті стали розробляти методи розпізнавання образів. Вони – предмет розгляду у даній кваліфікаційній роботі.

Під розпізнаванням образів розуміють процес віднесення деякого вхідного об’єкта до певного образу із множини еталонів [1].

У задачі розпізнавання в об’ємних базах візуальних об’єктів у системах комп’ютерного зору важливими показниками є результативність і час оброблення. Саме тому останнім часом набули прикладного застосування бінарні детектори ключових точок, такі як BRISK чи ORB [2], які отримують опис зображення як множину дескрипторів ключових точок у вигляді бінарного вектору, розмір якого є кратним ступені двійки. Бінарне подання даних значно прискорює процес порівняння дескрипторів за рахунок можливості застосування двійкових операцій та відповідно спрощує апаратну реалізацію системи розпізнавання. Крім того, бінарна арифметика дає можливість застосувати ефективний апарат оброблення бінарних даних та синтезувати нові підходи для визначення подібності дескрипторів КТ при побудові правил класифікації [3].

Метою кваліфікаційної роботи є розроблення методу класифікації зображень з використанням засобів швидкісного пошуку на підставі кластерних структур

1 АНАЛІЗ МЕТОДІВ КЛАСИФІКАЦІЇ

1.1 Формалізація задачі класифікації образів

Розпізнавання образів (pattern recognition) – це розділ теорії штучного інтелекту (artificial intelligence), що вивчає методи класифікації об'єктів. За традицією об'єкт, що піддається класифікації, називається образом (pattern). Образом може бути цифрова фотографія (розпізнавання зображень), буква або цифра (розпізнавання символів), запис мови (розпізнавання мови) тощо. У межах теорії штучного інтелекту розпізнавання образів включається в більш широку наукову дисципліну – теорію машинного навчання (machine learning), метою якої є розробка методів чи алгоритмів, що здатні навчатися [2].

Однією з найфундаментальніших проблем теорії інтелектуальних систем є розпізнавання образів. Але, треба також відмітити, що задача розпізнавання образів має велике практичне значення.

Зазвичай, використовується термін «класифікація» замість звичного терміну «розпізнавання». Дані терміни прийнято сприймати як синонімічні, але вони не повністю можуть замінювати один одного. Кожний з них застосовується у конкретних та індивідуальних сферах, і розуміння та вживання даних термінів часто залежить від специфіки конкретної задачі.

Розпізнавання образів, як і кожна математична дисципліна, має власний математичний апарат, який включає в себе дискретну математику, методи оптимізації, геометрію та алгебру. Класифікація образів має широко застосовується та використовується при створенні усіх комп'ютерних систем, на які покладаються інтелектуальні функції, тобто саме ті функції, які пов'язані з прийняттям рішень замість людини: пошук інформації, медична діагностика, інтелектуальний аналіз даних й криміналістична експертиза і т.д.

Введемо позначення і сформулюємо математичну постановку задачі класифікації [4].

Нехай Ω – простір образів; $\omega \in \Omega$ – образ; $M = \{1,2,...,m\}$ – номери класів $\Omega_1, \Omega_2, ..., \Omega_m$, таких що $\Omega_i \cap \Omega_j = \varnothing$, $\varnothing$ – пуста множина, та $\Omega = \bigcup_{i=1}^{m} \Omega_i$, $g : \Omega \to M$ – індикаторна функція, що є невідомою; X – простір ознак, тобто векторний простір, точками якого є вектори ознак образів; $x : \Omega \to X$ – функція, що ставить у відповідність образу ω його вектор ознак $X(\varpi\}$, $K_1, K_2, ..., K_m$ – підмножини простору X, такі що $K_i \cap K_j = \varnothing$ та $X = \bigcup_{i=1}^{m} K_i$, $g : X \to M$ – вирішальне правило, яке ставить у відповідність вектору ознак образа номер класу, якому він належить. Метою розпізнавання є побудова вирішального правила.

Ідея задачі класифікації з учителем полягає у тому, щоб на підставі навчальної вибірки, яка представлена у вигляді множини прецедентів (g_j, x_j), $j = 1,..., N$, мати можливість створити вирішальне правило g. За допомогою цього правила можна мінімізувати кількість помилок. У свою чергу, вчителем називається сама навчальна вибірка, або той, хто указав значення g_j.

У свою чергу, ідея задачі класифікації без учителя часто називається кластеризацією, або, власне кажучи, кластерним аналізом. У даній задачі вибірку ознак образів x_j, $j = 1,..., N$, необхідно розбити на підмножини, які не перетинаються, тобто кластери. Кластери складаються зі схожих один на одного об’єктів, але вкрай необхідно, щоб об’єкти, що належать до різних кластерів мали вагому відмінність один від одного.

Продемонструємо частину з типових постановок задач класифікації [5]:

– необхідність у тому, щоб віднести об’єкта до конкретного класу. Це може бути задача розпізнавання символів у тексті або, наприклад, ситуація, у якій необхідно прийняти рішення щодо наявності якогось дефекту у технічній деталі. Віднесення об’єкта до певного класу є відображенням найтиповішої

проблему класифікації. Зазвичай, говорячи про розпізнавання образів, найчастіше мають на увазі дану проблему;

– задача ідентифікації, мета якої, безсумнівно, полягає в тому, щоб виокремити конкретний об’єкт серед об’єктів, що подібні до нього. (наприклад, ситуація коли необхідно впізнати серед великої кількості людей свого родича);

– кластерний аналіз, мета якого полягає у тому, щоб виконати поділ заданого набору об’єктів (наприклад, даних про погодні умови у разних регіонах) на класи. Клас – це група об’єктів, які мають деяку схожість між собою за різними критеріями чи показниками. Через те, що класи апріорно не задані, дану задачу зазвичай називають класифікацією без учителя.

Як правило, проблеми розпізнавання легко вирішуються людьми підсвідомо. У свою чергу, спроби побудуви штучної системи розпізнавання не настільки переконливі. Зрозуміло, що основною проблемою є той факт, що неможливо адекватно визначити ознаки, за допомогою яких, власне, здійснюється розпізнавання. Найбільшої ефективності досягли штучні системи, які створюються для задач у яких можна виокремти такі ознаки.

Широку сферу використання на практиці отримали методи розпізнавання образів і технічні системи, за допомогою яких реалізуються дані методи. Далі наведемо деякі з них [5-6]:

– технічна діагностика. На виробництві часто виникає потреба у автоматизації контролю якості деталей. Зазвичай, задача полягає у з’ясуванні, чи деталь дефективна, чи ні. Якщо, у результаті визначається, що деталь містить деякий дефект, постає необхідність у визначенні типу цього дефекту;

– розпізнавання символів у тексті, наприклад, літер. Зазвичай використовується у сфері комп’ютерних технологій. Системи розпізнавання літер працюють разом зі сканерами. Сканери – це такі пристрої, за допомогою яких здійснюється введення друкованих зображень або текстів до комп’ютера. При введенні друкованого тексту сканер формує лише графічне

зображення. У свою чергу, спочатку необхідно розпізнати на зображенні окремі літери. Це робиться для того щоб була можливість створення текстового документу, з яким, власне кажучи, матиме можливість працювати текстовий редактор. Не менш важливою опцією розпізнавання літер є необхідність підтримки пристроїв, які здійснюють рукописне введення. Такі пристрої, зазвичай, схожі на звичайну авторучку. Вони часто комплектуються на персональні комп'ютери. Основною метою таких пристроїв є опція заміни класичного способу вводу інформаціх за допомогою клавіатури;

– робототехніка. Застосування методів розпізнавання в робототехніці є безсумнівно необхідним явищем. Роботи повинні безпосередньо сприймати зовнішній світ. Для цього їм необхідно мати пристрої машинного зору;

– розпізнавання мови. Сьогодні інтенсивно розвиваються та використовуються технології, які пов'язані з такими поняттями як голосе керуванням комп'ютером та введенням текстів із використанням голосу;

– охоронні системи. Методи розпізнавання в охоронних системах застосовуються для ідентифікації. Допустимо, необхідно ідентифікувати певну особу, з метою визначення того, чи має вона можливість входити на приватну територію. Також активно розвиваються такі системи, які можуть вирішувати задачі ідентифікації відбитків пальців.

1.2 Класи та їх властивості

Ключова парадигма теорії розпізнавання має наступний вигляд: будь-який об'єкт у природі є унікальним, всі об'єкти є типізованими [6].

Зміст цієї парадигми наступний. Кожний об'єкт характеризується рядом властивостей. Існує такий термін, як ознаки об'єкта. Саме наявність чи відсутність властивостей, а також якісні та кількісні характеристики цих властивостей і складають ознаки об'єкта. Будь-який об'єкт є унікальним, у

тому випадку, якщо в природі не існує двох різних об’єктів, для яких збігаються абсолютно всі ознаки. Теоретично, саме цей фактор дозволяє відрізнити один об’єкт від іншого. Але і бувають випадки, коли деякі ознаки різних об’єктів можуть збігатися. Це дає можливість міркувати про те, що ці об’єкти належать до одного типу або класу.

Фундаментальні поняття «клас» та «об’єкт» неможливо повністю формалізувати. Спробуємо навести їх неформальні визначення.

У теорії розпізнавання, об’єктом, зазвичай, прийнято називати будь-яку сутність, яка вже існує або може існувати в реальному світі. Також це може бути будь-яке явище або процес.

Це дуже широке визначення. Подальші уточнення можуть бути пов’язані з тим чи іншим звуженням нашого розуміння про те, що саме слід вважати об’єктом. Так, реально існуюча Ейфелева башта може вважатися об’єктом практично у будь-якій інтерпретації, а «розвиток наукових досліджень про етнокультурний стан Ефіопії та Антарктиди за період з липня 1898 року по січень 1927 року» – можливо, ні. Тут усе залежить від специфіки конкретної задачі і від мети, з якою вона вирішується.

У свою чергу, класом у теорії розпізнавання образів зазвичай прийнято називати сукупність об’єктів. Ці об’єкти мають якісь спільні ознаки.

Клас може об’єднувати фізично існуючі сутності – людина та яблуко, а може бути абстрактним поняттям – горе та економічна криза.

Інформативні ознаки – це ознаки, які дають можливість відрізнити представників одного класу від іншого.

Класом називається сукупність об’єктів, пов’язаних між собою деяким відношенням еквівалентності [7]. Відношенням еквівалентності – це відношення, яке є симетричним, рефлексивним а також транзитивним (наприклад, таке відношення, як «дорівнювати»).

Набори інформативних та інваріантних ознак можуть збігатися, але це зовсім необов’язково.

Іноді, для того, щоб виключити непорозуміння, називатимемо об’єкти і класи реального світу відповідно *P*-об’єктами і *P*-класами. Зрозуміло, що *P*-об’єкт може належати будь-якій кількості *P*-класів. *P*-об’єкти часто називаються реалізаціями, або зразками *P*-класів.

Класам характерні такі основні властивості:

– усі представники класу мають певний набір спільних ознак;

– змінюваність реалізацій класів. По-перше, бувають ситуації, коли різні об’єкти, які належать до одного класу, не схожі між собою. Необхідно, щоб вони мали спільні інваріантні ознаки. При цьому, всі інші ознаки можуть варіювати як завгодно. По-друге, з часом, один і той самий об’єкт може змінюватися. Цей об’єкт може навіть поступово переходити від одного класу до іншого, наприклад, перетворення пуголовка на жабу. Усе це свідчить про те, що розпізнати чіткі межі класу зазвичай буває фактично неможливо;

– ознайомлення з деякою сформованою кількістю представників одного класу дає можливість впізнавати інших представників цього класу. Власне кажучи, дана властивість означає можливість навчання на прикладах, тобто на основі спостереження певної кількості прикладів (можливо, разом з контрприкладами). Такоє надається можливість сформулювати правило розпізнавання, за допомогою якого можна відрізняти представників даного класу від представників іншого, але, скоріш за все, з певним відсотком помилок. У деяких випадках правилом розпізнавання може бути предикат, який залежить від інформативних або інваріантних ознак. Інколи правило розпізнавання реалізується у вигляді складної процедури.

У тих випадках, коли навчання на прикладах неможливе або неефективне, інколи, правило розпізнавання можна задати явно. Якщо це не можливо, єдиною можливістю для надійного розпізнавання залишається запам’ятовування всіх можливих представників даного класу. Цей випадок не становить інтересу з теоретичної точки зору. Його можна реалізувати лише у тому випадку, коли кількість можливих представників не є дуже великою.

У багатьох випадках, створення пристроїв, які можуть виконувати функції розпізнавання різних об’єктів, відкриває можливість заміни людини, як елемента складної системи, спеціалізованим автоматом. Дана заміна надає можливість значно розширити можливості різних систем, які здатні виконувати складні інформаційно-логічні задачі. Якість робіт, виконуваних людиною на будь-якому робочому місці, залежить від кваліфікації та досвіду. У той же час, автомат, який заміняє люлину, діє одноманітно і завжди забезпечує однакову якість.

Причиною створення і пошуку шляхів створення ряду систем розпізнавання є не тільки зазначена заміна і звільнення людини від виконання рутинних операцій. Існують ситуації, коли людина не в змозі вирішити поставлену задачу зі швидкістю, яка задається обставинами. Наприклад, протиракетний маневр літака в складних метеоумовах. У свою чергу, автоматизована система з задачами подібного характеру може справлятися без проблем.

Отже, основними причинами заміни людини в задачах розпізнавання є наступні фактори:

– звільнення людини від одноманітних рутинних операцій для того, аби мати можливість вирішити інші важливі задачі;

– підвищення якості виконуваних робіт;

– підвищення швидкості вирішення задач.

1.3 Методи класифікації даних

Задача розпізнавання з навчанням, або класифікації, з формального огляду полягає у наступному [7].

Мають місце K класів, які позначатимемо S_1, S_2, … S_K. Задано множину прецедентів, тобто об’єктів, для кожного з яких відомо, до якого з цих класів він належить:

$$X=\{X_i,y_i;\ i=\overline{1,N}\}=\{(x_{i,1}\ x_{i,2}\ \dots\ x_{i,p}),y_i;i=\overline{1,N}\}=\begin{pmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,p} & y_1 \\ x_{2,1} & x_{2,2} & \dots & x_{2,p} & y_2 \\ \dots & \dots & \dots & \dots & \dots \\ x_{N,1} & x_{N,2} & \dots & x_{N,p} & y_N \end{pmatrix}, \quad (1.1)$$

де $X_i = (x_{i,1}\ x_{i,2}\ \dots\ x_{i,p})$ – i-й об’єкт, описаний p ознаками;

$x_{i,j}$ – значення j-ї ознаки для i-го об’єкта;

y_i – назва класу, до якого належить i-й об’єкт;

N – кількість об’єктів;

p – кількість ознак.

Цю множину називають навчальною вибіркою.

На її основі потрібно побудувати правило, яке б дозволяло будь-який новий об’єкт $X_0 = (x_{0,1}\ x_{0,2}\ \dots\ x_{0,p})$ відносити до одного з класів $S_1, S_2, \dots S_K$.

Розв’язання задачі здійснюють двома етапами:

Етап 1. Навчання – на цьому етапі будується вирішальне правило. Цей етап, як правило, складніший і більш трудомісткий. Геометрично на етапі навчання будується поверхня, яка розділяє класи, хоча аналітичний вигляд цієї поверхні можна визначити не всіма методами;

Етап 2. Класифікація – власне класифікація нового об’єкта на основі побудованого правила.

Серед методів розв’язання задачі класифікації можна виділити: метричні (ближнього сусіда та його модифікації, на основі порівняння з еталоном, потенціальних функцій), опорних векторів, дерева рішень, статистичні (байєсівський класифікатор, логістичну регресію), нейронні мережі, колективи методів (бустінг, беггінг) [8, 9].

Гіпотеза компактності, яка стверджує, що класам відповідають компактні множини у просторі ознак, покладена в основу багатьох методів класифікації. Це означає, що реалізації одного і того самого класу відображуються у просторі ознак у близькі точки, що, в результаті, утворюють «компактні» згустки. Цю гіпотезу не завжди можна підтвердити експериментально. Але найголовніше, що ті задачі, в межах яких вона

виконується, усі без винятку мають простий розв'язок. І навпаки, задачі, для яких гіпотезу компактності не можна підтвердити, або зовсім не мають розв'язку, або їх розв'язують зі значними труднощами, із залученням додаткових прийомів. Тому гіпотезу компактності можна розглядати як ознаку можливості успішного розв'язання задачі.

Метод найближчого сусіда (Nearest Neighbor – NN) – це найпростіший метод класифікації. Процес навчання в ньому полягає в запам'ятовуванні всіх об'єктів навчальної вибірки. Класифікують новий об'єкт X_0 за таким правилом (рисунок 1.1, *а*): класифікований об'єкт відносять до того класу, представник якого найближче розташований до X_0, тобто

$$X_0 \in S_i : \qquad X_q \in S_i, \qquad d(X_0, X_q) = \min_{i=\overline{1,N}} d(X_0, X_i). \tag{1.2}$$

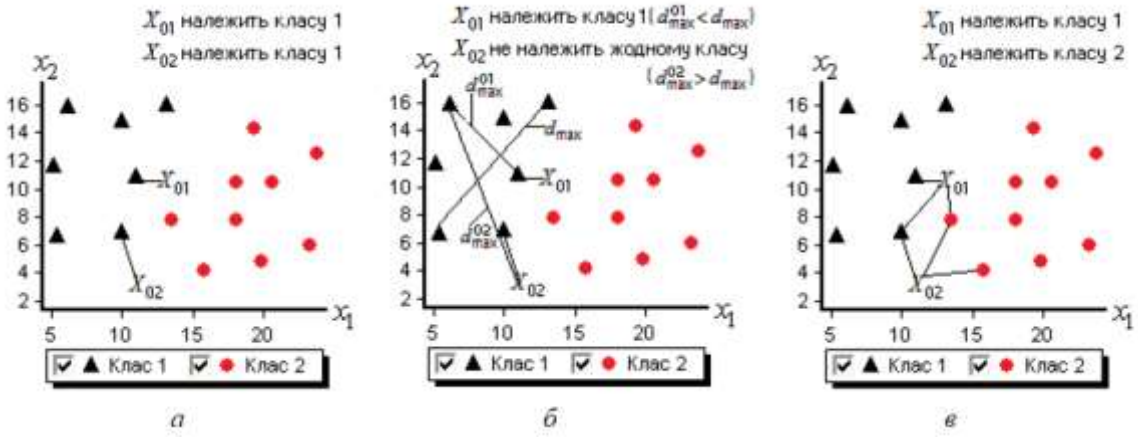

Рисунок 1.1 – Пояснення роботи методів:
а – ближнього сусіда; *б* – модифікованого ближнього сусіда;
в – *k*-ближніх сусідів (*k* = 3)

Перевагами методу є простота реалізації та наочна інтерпретація результатів.

Наведемо недоліки даного методу [10]:

– метод потребує зберігання усієї навчальної вибірки, а це може призвести до неефективної витрати пам’яті;

– нестійкість до шуму. Якщо серед навчальних об’єктів є викид (об’єкт, який розташований серед представників чужого класу), то всі об’єкти, для яких він виявиться найближчим, будуть класифіковані неправильно;

– пошук найближчого сусіда передбачає порівняння класифікованого об’єкта з усіма об’єктами навчальної вибірки, здійснюваного за O(N) операцій, тобто потребує багато часу для задач із великими вибірками або високою частотою запитів. Для вирішення цієї проблеми слід застосовувати ефективні алгоритми пошуку найближчих сусідів, наприклад, застосовувати *k*-вимірні дерева (*k-d* tree) для зберігання навчальної вибірки;

– відсутні параметри для налаштування.

Замість того, щоб шукати одне схоже зображення, ми будемо шукати k найближчих сусідів відповідно до метрики відстані. Після цього проведемо голосування по кожному сусідові і спрогнозуємо мітку відповідно до більшості голосів. Набір точок з числом сусідів $k = 1$, $k = 3$ і $k = 5$ наведено на рисунку 1.2.

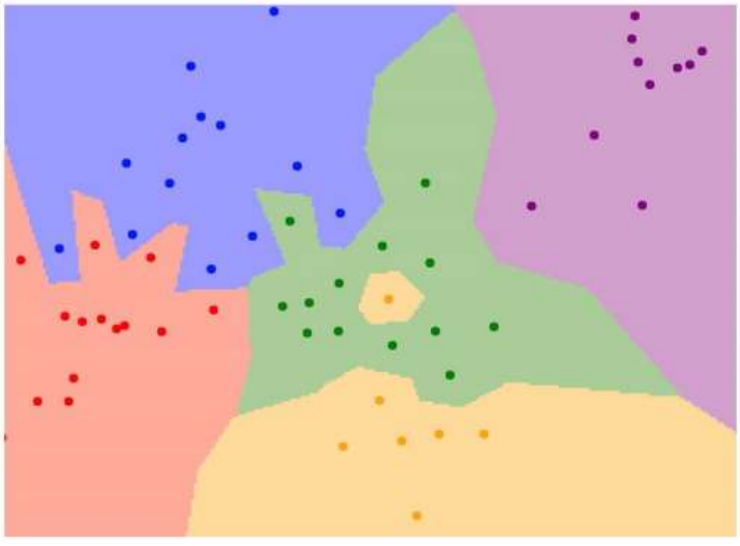

Рисунок 1.2 – Набір точок з числом сусідів $k = 1$, $k = 3$ і $k = 5$

Можна побачити, що жовта точка всередині зеленого кластера більше не створює область навколо себе, а межа між червоним і синім класами стає більш гладкою. Як правило, для класифікації використовується значення $k > 1$. Білі регіони тут означають області, в яких не знайдено жодного найближчого сусіда.

Еталоном називають узагальнене описання класу.

Основна суть методів на основі порівняння з еталоном полягає у побудові еталонів кожного класу на етапі навчання з подальшою класифікацією нових об'єктів за правилом найближчого або k найближчих еталонів [11-13].

Залежно від способу описання еталонів класів виділяють декілька варіантів методів.

Спосіб 1. Еталонами обирають усереднені об'єкти класу, які визначають за формулою:

$$\overline{X}^{(l)} = \frac{1}{N_l} \sum_{X_i \in S_l} X_i, \; l = \overline{1,K}, \tag{1.3}$$

де N_l – кількість об'єктів у класі S_l.

Тоді новий об'єкт X_0 класифікують за правилом найближчого сусіда:

$$X_0 \in S_l : \quad d(X_0, \overline{X}^{(l)}) = \min_{h=1,K} d(X_0, \overline{X}^{(h)}). \tag{1.4}$$

Недоліком методу можна вважати те, що його застосування призводить до побудови у просторі ознак лінійних роздільних поверхонь.

Спосіб 2. Еталонами можуть бути один або декілька навчальних об'єктів, що забезпечують якість класифікації не гіршу, ніж за усією навчальною вибіркою. Задача навчання передбачає вибір еталонних об'єктів, наприклад, методом STOLP або FRiS-STOLP [14]. Таким чином, навчальна вибірка значно скорочується за рахунок видалення з неї неінформативних та

шумових об’єктів. Класифікацію нових об’єктів здійснюють за методом найближчого або *k* найближчих сусідів.

Метод STOLP для випадку двох класів працює таким чином. Спочатку знаходять найбільш «напружені» примежові об’єкти. Для цього для кожного об’єкта обчислюють відстань до найближчого об’єкта свого класу (d_{in}) та найближчого об’єкта чужого класу (d_{out}). Відношення $W=d_{in}/d_{out}$ характеризує величину ризику для даного об’єкта бути розпізнаним як об’єкт чужого класу. Серед об’єктів кожного класу обирають по одному з максимальним значенням величини *W*, які заносять до списку еталонів. Далі виконують пробне розпізнавання всіх об’єктів навчальної вибірки за правилом найближчого сусіда за допомогою еталонів. Серед об’єктів, класифікованих неправильно, обирають один із максимальним значенням *W*, і ним поповнюють список еталонів. Після цього пробне розпізнавання всіх об’єктів навчальної вибірки повторюють. Процедуру продовжують, поки всі навчальні об’єкти не будуть розпізнаватися правильно.

Якщо кількість класів більша двох, описану процедуру застосовують для кожної пари класів. Для того, щоб здійснити прискорення роботи, рекомендується розпочинати застосовувати метод для пари двох найближчих класів. Після розв’язання задачі, для них обирають наступну найближчу пару класів. Вибір пари виконується до тих пір, поки всі пари не будуть розглянуті. Якщо під час розгляду поточної пари виявиться, що для одного з класів, який входить до неї, на попередніх кроках вже було сформовано список еталонів, цей список включають із самого початку і за необхідності поповнюють. Як відстань між класами доцільно застосовувати відстань між двома найближчими об’єктами двох різних класів (відстань Хаусдорфа).

1.4 Застосування дерев рішень

Дерево рішень (decision tree) – це правило класифікації у вигляді дерева (рис. 1.3). Внутрішні вузли дерева називають вузлами перевірки, оскільки в них перевіряють умову щодо значення деякої ознаки. Листи дерева містять рішення у вигляді назв класів [15].

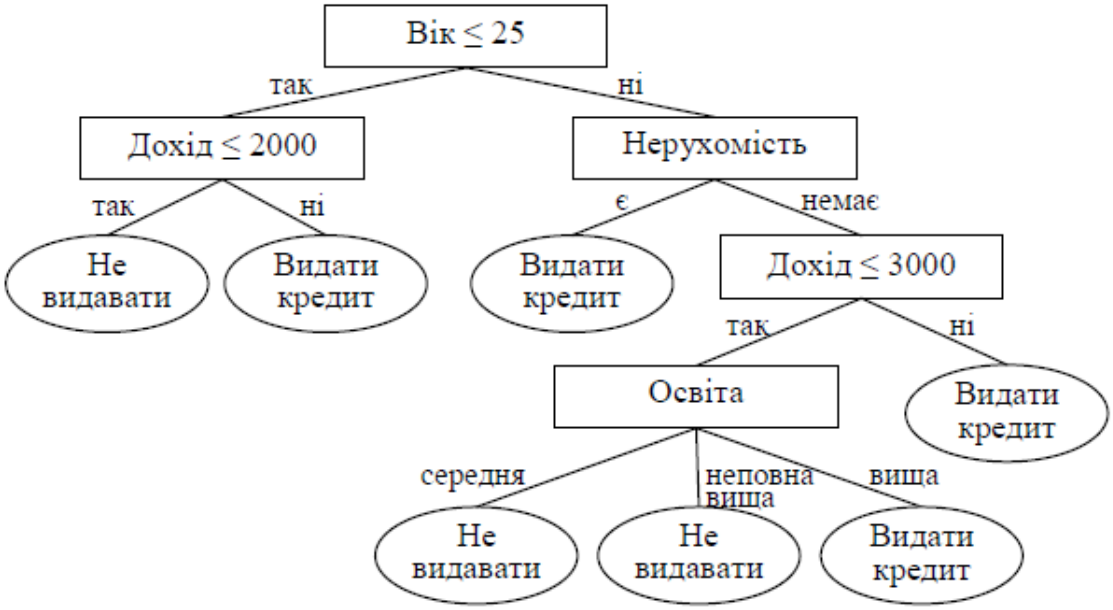

Рисунок 1.3 – Приклад дерева рішешь

Побудова дерева рішень за навчальною вибіркою зазвичай відбувається на етапі навчання [16]. Для того, щоб здійснити класифікацію нового об’єкту на основі побудованого дерева, відбувається подолання шляху від самого кореня й до листа. Необхідно відзначити, що перевірка значення певної ознаки об’єкта відбувається на кожному внутрішньому вузлі. У залежності від того, яка була отримана відповідь, здійснюється пошук відповідного розгалуження, за допомогою якого рухають до вузла, який безпосередньо знаходиться рівнем нижче. Ця процедура може продовжуватись доти, поки не буде виявлено лист, який, власне кажучи, містить назву класу об’єкта.

Зазвичай, для побудови дерева рішень використовують такі алгоритми, як ID3, C4.5, CART тощо [16].

Нагадуємо, що процес побудови дерева відбувається зверху вниз. У першу чергу створюють корінь дерева. Для виконання цієї задачі, серед усіх ознак, використовуючи метод перебору, обирають одну. Дана ознака є найкращою за рядом критеріїв. Далі, узявши за основу обрану ознаку, формують умову для кореня. Ця умова розбиває множину об’єктів, що з ним асоціюється, на декілька підмножин. Відповідно, це породжує нові вузли дерева (рис. 1.4). У результаті, аналогічна процедура рекурсивно застосовується до кожного з породжених вузлів рекурсивно. Якщо з вузлом асоціюється множина об’єктів, що належать одному класу, то вузол перетворюють на лист. Побудова завершується. Рішенням листа стає клас, об’єкти якого з ним асоціюються. У ситуації, коли з вузлом асоціюється пуста множина об’єктів, він перетворюється на лист. У такому випадку, клас стає рішенням вузла, представників якого більше у вузла-предка.

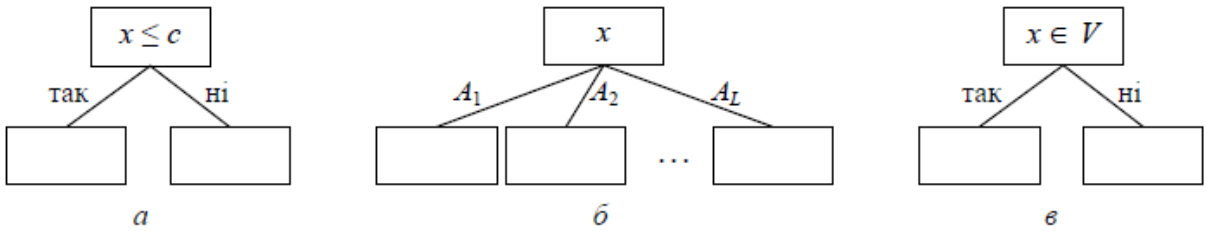

Рисунок 1.4 – Приклади умов перевірки у вузлі за ознакою x:
а – кількісною; *б-в* – якісною

Існує ряд відмінностей у лозіці побудови дерева рішень. Головним чином, причинами відмінності можуть бути критерій вибору ознаки, можливе врахування пропущених значень ознак, умови перевірки у вузлі або, навіть, наявність механізму, який здійснює відсікання гілок.

Буває так, що алгоритми будують досить складні дерева. Побудовані дерева, у результаті, мають достатньо велику кількість вузлів та гілок. Їх

називають гіллястими. Такі дерева є небажаними. По-перше, такі дерева мають вершини, які розташовані далеко від кореня. Вони асоціюються з незначною кількістю об’єктів навчальної вибірки. У результаті, правила в цих вершинах ненадійні. По-друге, подібні дерева достатньо складно інтерпретувати.

Для того, щоб мати можливість отримати дерево, яке має меншу кількістю вузлів, можна застосувати одну з наведених далі технік [17, 18]:

– рання зупинка (prepruning). Створюється критерій, за допомогою якого визначається доцільність побудови вузла. Якщо критерій виконується, тоді вузол перетворюють на лист і побудова закінчується. Рішенням листа стає клас, у якого кількість представників у вузлі найбільше. Існує ряд критеріїв зупинки. Наприклад, це може бути досягнення заданої глибини дерева або кількості об’єктів у вузлі. Рання зупинка надає можливість скоротити час побудови дерева. Але, у свою чергу, така зупинка знижує якість класифікації;

– відсікання гілок (pruning). Ця техніка більш ефективна, ніж попередня. Відсікання гілок виконується тоді, коли дерево повністю побудовано. Деякі алгоритми передбачають власний механізм відсікання гілок. Якщо механізм відсутній, тоді можна використати наступну процедуру. Кожен вузол необхідно перетворити на лист, починаючи з вузлів найнижчого рівня. Звичайно, таку процедуру необхідно використовувати у випадку, коли через це не відбувається суттєве збільшення помилки класифікації. Зазвичай, наведену процедуру виконують, поки існують вузли, які мають можливість бути перетвореними на листя.

За рахунок використання дерев рішень можна:

– будувати правила класифікації на природній мові, використовуючи терміни, які є зрозумілими для спеціалістів предметної галузі;

– одночасно враховувати кількісні та якісні ознаки в правилі;

– відбирати інформативні ознаки в ході побудови дерева, що дозволяє не застосовувати спеціальні методи;

– будувати правила за даними з пропусками.

У свою чергу, існує ряд недоліків дерев рішень:

– усі алгоритми, які здійснюють побудову дерев рішень, завжди дуже жадібні, а це означає, що дерева, отримані у результаті побудови, зазвичай, неоптимальні щодо якості класифікації та розміру;

– алгоритми, які здійснюють побудову дерев рішень, можуть ускладнювати структуру дерева і, як результат, будувати досить гіллясті дерева. Через це, дерева стають менш надійними. Також це може призвести до ефекту перенавчання. Також, варто позначити, що нівелюється така перевага, як наочність і простота інтерпретації.

1.5 Постановка задачі

Класифікація зображень із використанням методів швидкісного пошуку даних за допомогою кластеризації є актуальним завданням в задачах обробки і розпізнавання візуальних об’єктів. Ставиться завдання розробити алгоритм швидкої класифікації зображень, який використовує кластерні структури даних на основі інформації, отриманої з бази зображень, з використанням спільного методу детектування КТ та обрахування дескрипторів BRISK та їх подальшого розбиття на кластери за допомогою методу k-means.

Об’єктом роботи є методи швидкісної класифікації зображень.

Метою роботи є розроблення методу швидкісного пошуку, який використовує кластерні структури. Необхідно реалізувати програмний застосунок, який буде використовувати даний метод для розпізнавання зображень на базі еталонів. Формування для кожного з еталонів дескриптора класу вважаємо засобом онлайн-навчання в межах прикладної бази зображень. Виконання наведеної задачі необхідне для подальшого порівняння методу швидкісного пошуку із класичним лінійним методом у

задачі класифікації зображень з оцінюванням швидкодії та точності класифікації.

Враховуючи мету роботи необхідно вирішити такі завдання:

– провести аналіз існуючих методів класифікації зображень;

– реалізувати методи лінійного пошуку та швидкісного пошуку у базі дескрипторів із використанням кластерних структур даних;

– реалізувати комп’ютерну модель детектування КТ та обчислювання множини дескрипторів на базі зображень;

– провести комп’ютерне тестування алгоритму класифікації на реальній базі зображень з порівнянням результатів впровадження швидкісного та лінійного пошуку, оцінюванням працездатності та конкурентоспроможності методів;

– зробити висновки щодо виконаної роботи.

2 КЛАСИФІКАЦІЯ ЗОБРАЖЕНЬ НА ПІДСТАВІ МЕТОДІВ ШВИДКОГО ПОШУКУ

2.1 Класифікація з використанням швидкого пошуку

Пропонується розглянути простір B^n, що складається з багатовимірних бінарних векторів розмірністю n. Саме у цьому просторі будуть створюватися образи розпізнаваного об’єкту разом з еталонами. Необхідно зафіксувати деяку мультимножину векторів $E_i \subset B^n$ у вигляді деякого опису еталону $E_i = \{e_v(i)\}_{v=1}^{s}$, що знаходиться у просторі множин дескрипторів КТ ($s = card\ E$ – це число дескрипторів у множині [19, 20]). Ознаки – це вектори $e_v \in B^n$, скінченна множина яких створює опис об’єкту. Класифікація, власне, передбачає наявність деякої бази E описів еталонних зображень розмірністю $N : E = \bigcup_{i=1}^{N} E_i$. Кожен еталонний опис E_i має можливість репрезентувати окремий клас. Цей клас представляється у вигляді скінченної множини дескрипторів КТ із потужністю S.

Пропонується розглянути довільний опис $Z \subset B^n$, $Z = \{z_w\}_{w=1}^{s}$ розпізнаваного об’єкту. Ставиться задача побудови класифікатора K як відображення $K : Z \to [1,2,..., N]$ на основі попереднього конструювання деякої індексованої структури на множині E. Класифікацію K необхідно представити у вигляді процесу, що складається з двох етапів $K = K_2 K_1$. На першому етапі $K_1 : B^n \to [1,2,..., N]$ здійснюється визначення класу d_w для кожного дескриптора $z_w \in Z$. У свою чергу, на другому етапі $K_2 : D \to [1,2,..., N]$ із множини $D = \{d_w\}_{w=1}^{s}$ локальних рішень формується результуюче рішення щодо класу об’єкту Z. Власне кажучи, K_1 можна розглядати як багатозначну характеристичну функцію, за допомогою якої можна визначити еталонний клас E_i.

Етапи K_1, K_2 будуються за допомогою створення на етапі K_1 деякого випадкового розподілу за класами еталонів. Не варто забувати про логічне оброблення таких розподілів [20, 21].

Об'єктивно реалізація K здійснюється за допомогою апріорних даних, що належать існуючій базі E. Це відбуваться через те, що належність усіх $e_v(i)$ до відповідного образу E_i всередині бази уже відома на початку класифікації.

Здійснюючи класифікацію K_1 шляхом традиційного лінійного пошуку (метод повного перебору), послідовно переглядаючи кожен елемент набору E, то, зазвичай, використовується конкурентне правило [22]:

$$d_w = \arg \min_{i,v} \rho(z_w, e_v(i)), \tag{2.1}$$

де d_w – це номер еталону E_i, до якого буде віднесено дескриптор z_w об'єкту $d_w \in \{1,..., N\}$;

$\rho(z_w, e_v(i))$ – метрика у векторному просторі.

У свою чергу, необхідну кількість Q обчислених значень метрики в (2.1) лінійним пошуком можна оцінити параметром $Q = N \cdot s^2$. При цьому, обсяги описів еталонів та об'єкта вважаються рівноцінними. Для векторів простору B^n в (2.1) може бути застосована проста в обчислювальному сенсі метрика Геммінга. Ця метрика здійснює підрахунок кількості бітів, які не співпадають.

Далі, на етапі класифікації K_2, усе зводиться до того, що на кожному кроці аналізу опису Z за правилом (2.1) при значенні d_w відбувається інкрементування числа r_i голосів елементів, які віднесені до i-го класу

$$r_i = \begin{cases} r_i + 1, & d_w = i \\ r_i, & d_w \neq i \end{cases} \tag{2.2}$$

а клас i_0 образу Z об'єкта визначається за найбільшої кількості голосів:

$$i_0 = \arg \max_{i=1,\dots,N} r_i . \qquad (2.3)$$

Значення $\{r_i\}_{i=1}^{N}$ відображають гістограму класів за числом голосів елементів із Z. Вирази (2.2), (2.3) надають можливість конкретизувати етап K_2. Конкретизація полягає у обробці голосів для компонентів опису Z. Розглянута процедура класифікації базується на принципі інтелектуального аналізу даних. Принцип полягає у підрахунку числа позитивних рішень на аналізованій множині даних [23].

На етапі попередньої обробки, необхідно створити у еталонній множині образів E спеціалізовану структуру з метою забезпечення швидкісної результативної класифікації.

Здійснюється розбиття T на множині E дескрипторів бази зображень. У результаті розбиття з'являється множина з M непересічних груп $T_k(E)$:

$$E = T(E) = \bigcup_{k=1}^{M} T_k(E), T_k(E) \cap T_j(E) = \varnothing . \qquad (2.4)$$

При цьому, необхідно мати на увазі один з найбільш поширених способів розбиття – кластеризацію [23]. Впровадження кластеризації пов'язують із наближеними засобами самонавчання для вирішення прикладних задач. Трансформація (2.4) здійснює попередню класифікацію, при цьому повністю зберігає усю сукупність аналізованих даних, що тепер розподіляється між групами $T_k(E)$. У результаті перетворення (2.4) кожний дескриптор $e_v \in E$ бази еталонів отримує параметр k номеру групи (кластера).

Зважаючи на уже існуюче розбиття $E = \bigcup_{i=1}^{N} E_i$ на окремі еталонні образи, можна визначити величину

$$t_{i,k} = card\{e_v \mid e_v \in E_i \ \& \ e_v \in T_k\} \tag{2.5}$$

як число дескрипторів i-го класу, що потрапили до кластеру T_k. На підставі $t_{i,k}$ отримаємо вагову кількісну характеристику $b_i(T_k)$ для кожного кластеру як відношення

$$b_i(T_k) = \frac{t_{i,k}}{card\, T_k}. \tag{2.6}$$

Вираз (2.6) визначає розподіл елементів кожного сегменту даних за класами еталонів у вигляді вагових коефіцієнтів класів, причому $\sum_{i=1}^{N} b_i = 1$. Тепер кожний елемент $e_v \in E$ бази можна вважати функцією $e_v = f(k, i, b_i(T_k))$ від параметрів номерів кластеру, еталону та розподілу даних за класами всередині кластеру. Розподіл b є спільною характеристикою усіх елементів кластеру, отриману за результатами аналізу (навчання) для бази E. На етапі навчання ефективним для підвищення результативності класифікації може бути запровадження логічного оброблення вектору b задля спрощення аналізу та підсилення впливу вагових коефіцієнтів b_i для найбільш значущих класів у кластері [24, 25].

Для кожного кластеру визначимо параметр центру $\alpha_k = \varphi(T_k)$, який далі будемо застосовувати як характеристику для організації швидкісного доступу до даних. Центр може визначатися за різноманіттям моделей φ в залежності від змісту даних, але найчастіше застосовують середнє чи медіану [26].

Ще одним важливим параметром, що впливає на швидкодію пошуку, є число кластерів M. Воно не обов'язково пов'язане з числом класів. Чим менше число кластерів, тим вища швидкодія переходу до кластера. Чим більше число кластерів, тим менше об'єм даних для аналізу всередині кластерів. Граничними рівноцінними ситуаціями є один кластер $(M=1)$ та взагалі відсутність кластерування $(M=N\cdot s)$, що відповідають лінійному пошуку. За думкою дослідників тут можна формулювати задачу оптимізації числа M кластерів, де критерієм виступає число Q обчислень метрики для компонентів опису [27, 28].

Якщо в одному кластері в середньому число елементів є $(N\cdot s)/M$, то кількість обчислень (2.1) всередині індексної структури з використанням центрів кластерів або значень хеш-коду пропорційна $Q_1 = s\cdot M + (N\cdot s)/M$, що значно менше, ніж $Q_2 = N\cdot s^2$ для традиційного лінійного пошуку. Для конкретних значень, $s=500$, $N=10$, $M=10$ виграш Q_2/Q_1 складає приблизно 450 разів і зростає зі збільшенням обсягів N, s даних.

Задля подальшого прискорення процедури оброблення можна застосувати додаткове упорядкування отриманої кластерної структури даних. Це відбувається за допомогою впровадження, наприклад, сортування даних всередині кластерів T_k за деяким ключем $g(e_v)$. У якості $g(e_v)$ можна застосувати, наприклад, число одиничних бітів або відстань від центру кластера [28]. У результаті, функціонально, елемент даних приймає вигляд $e_v = f(k,i,b,g)$.

Тепер опишемо загальний вигляд класифікації на основі індексованої структури: застосовуємо детектор КТ, формуємо опис $Z=\{z_w\}_{w=1}^{s}$ розпізнаваного об'єкту як множину дескрипторів. Побудуємо класифікатор з використанням швидкісного пошуку у базі E.

Спосіб класифікації із впровадженням структури кластерів полягає в наступному:

Етап 1. Для дескриптора запиту $z_w \in Z$ конкурентним шляхом лінійного пошуку визначаємо кластер, який має найменшу відстань у множині центрів

$$k_0 = \arg \min_{k=1,\dots M} \rho(z_w, \alpha_k); \tag{2.7}$$

Етап 2. Всередині кластера з номером k_0 для дескриптора запиту, з метою встановлення його класу, можна застосувати різновиди пошуку: лінійний, підмножини найближчих сусідів, визначення на підставі розподілів еталонних даних всередині кластеру та інші;

Етап 3. За результатом пошуку на етапі 2 для z_w визначаємо клас d_w;

Етап 4. На підставі аналізу усієї множини Z дескрипторів об'єкту накопичуємо лінійку $\{r_i\}_{i=1}^{N}$ значень гол осів за кожний еталонний клас;

Етап 5. За виразом (2.3) класифікуємо об'єкт до класу, що набрав найбільшу кількість голосів.

Розподіли за класами еталонів для створених груп даних теж несуть важливу інформацію для класифікації, на підставі чого можна видалити із аналізу окремі кластери (якщо класи розподілені суто рівномірно), або за розподілом приймати однозначне класифікаційне рішення (якщо спостерігаються суттєві переваги для окремих класів).

Метрика Геммінга застосовується для порівняння двох двійкових рядків даних [26]. При порівнянні двох двійкових рядків однакової довжини, відстань Геммінга – це кількість бітових позицій, в яких два біти відрізняються.

$$d_{ij} = \sum_{k=1}^{p} \left| x_{ik} - x_{jk} \right|. \tag{2.8}$$

На рисунку 2.1 можна побачити два двійкових ряди, у яких відрізняються 6 бітів, отже, відстань Геммінга у даному випадку дорівнює трьом.

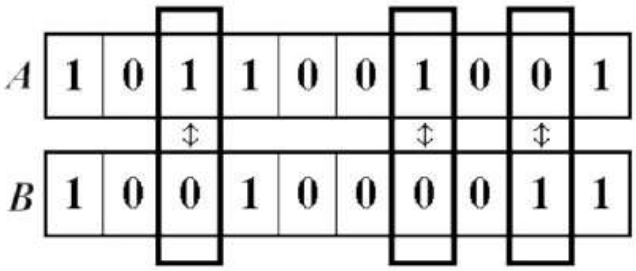

Рисунок 2.1 – Два бітових ряди, чия відстань Геммінга дорівнює 3

Якщо розглядати відстань Геммінга у більш загальному випадку, то дана відстань застосовується для рядків однакової довжини будь-яких q-ічних алфавітів. Вона служить метрикою відмінності (функцією, яка може визначати відстань в метричному просторі) об’єктів однакової розмірності. Далі наведений приклад відстаней в двійковому Тессеракті на рисунку 2.2.

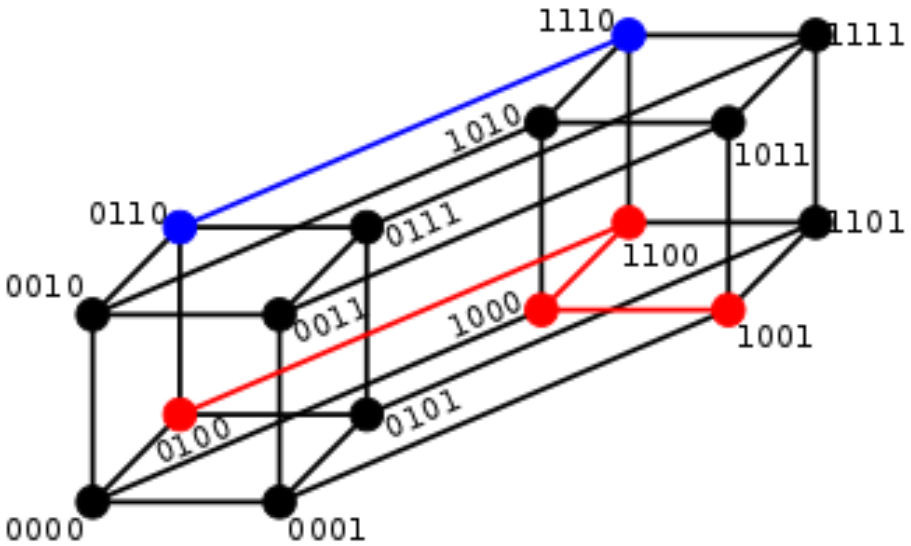

Рисунок 2.2 – Приклади відстаней в двійковому Тессеракті: відстань 1(синя) 0110 → 1110, відстань 3(червона) 0100 → 1001

2.2 Аналіз методів кластеризації даних

Кластерний аналіз даних – це поділ множини об’єктів на кластери або класи, групи. Для задач кластеризації характерна відсутність відмінностей об’єктів по атрибутам. Вперше, такий термін як *кластерний аналіз*, ввів Тріоном ще в 1939 році. При проведенні кластерного аналізу не будуються апріорні припущення щодо заданого набору даних. Також не вводяться обмеження на подання об’єктів аналізу і типів даних. Кластерний аналіз, зазвичай, також використовується для того, щоб скоротити розмірність. Зараз кластерний аналіз розвивається у різних напрямках. Ці напрямки пов’язані з такими сферами діяльності, як комерційна, технічні науки, біологія і психологія [28, 29].

Формальна постановка задачі кластеризації здійснюється наступним чином. Визначається множина об’єктів даних $Q = \{q_1, q_2, ..., q_n\}$. Кожен об’єкт q_j характеризується набором атрибутів:

$$\tilde{x}_j = \langle x_{j,1}, x_{j,2}, ..., x_{j,m} \rangle . \qquad (2.9)$$

Прикладом такої множини об’єктів може бути колектив викладачів вищого навчального закладу, кожен з яких характеризується набором показників (атрибутів) про кваліфікацію, навчально-методичну та наукову діяльність.

Кожна змінна з набору x_j приймає значення з множини дійсних чисел $x_j = \{v_j^1, v_j^2, ...\}$. Рішенням задачі кластеризації є множина сформованих кластерів

$$C = \{c_1, c_2, ..., c_g\}, \qquad (2.10)$$

де $c_k = \left\{ q_i, q_j \middle| q_i \in Q,\ q_j \in Q \ u\, d(q_i, q_j) < \sigma \right\}$ – це кластер, який має схожі об'єкти із множини Q;

$d(q_i, q_j)$ – міра близкості між об'єктами;

σ – величина, яка визначає міру близкості між об'єктами.

Міра близкості повинна відповідати наступним умовам [15, 16]:

– $d(q_i, q_j) \geq 0$, *для усіх* q_i *та* q_j;

– $d(q_i, q_j) = 0$, *тоді і тільки тоді, коли* $q_i = q_j$;

– $d(q_i, q_j) = d(q_j, q_i)$;

– $d(q_i, q_j) \leq d(q_i, q_k) + d(q_k, q_j)$.

При виконанні нерівності $d(q_i, q_j) < \sigma$ об'єкти з множини Q розглядаються як близькі та розміщуються до одного кластера. Інакше об'єкти розміщуються до різних кластерів.

У задачах кластеризації вибір міри близкості передбачає представлення об'єктів у вигляді точок m-мірного простору R^m. При цьому міри близкості визначають відстань між двома точками простору R^m. Найбільше застосування знаходять такі міри: евклідова відстань, відстань по Геммінгу, відстань Чебишева, відстань Махаланобіса.

До теперішнього часу відомо більше 100 алгоритмів кластерного аналізу. Всі алгоритми поділяють на ієрархічні і неієрархічні алгоритми.

Ієрархічні алгоритми кластерного аналізу в свою чергу поділяють на агломіративні і дівизимні.

В ієрархічних агломіративних алгоритмах кластеризації вихідна множина об'єктів Q представляється як множина кластерів C. Таким чином, на першому кроці алгоритму маємо:

$$c_1 = \{q_1\}, c_2 = \{q_2\}, ..., c_g = \{q_n\} \text{ та } g = n. \qquad (2.11)$$

На другому кроці алгоритму, використовують обрану міру близкості $d()$. Далі знаходять кластери з найменшими віддаленнями один від одного та здійснюють злиття кластерів c_i, c_j до одного загального кластеру $c_k = \{c_i, c_j\}$. Процес пошуку кластерів з найменшими віддаленням та їх злиттям повторюють. Як результат, формуються множини кластерів із потужностями $g-1$, $g-2$, $g-3$.

Перерахунок відстані між кластером c_k та кластером $c_l, l = 1,2,...$ здійснюється за формулою:

$$d_{k,l} = \alpha_i d_{i,l} + \alpha_j d_{j,l} + \beta d_{i,j} + \gamma \left| d_{i,l} - d_{j,l} \right|, \quad (2.12)$$

де $d_{i,j}$ – відстань між кластерами c_i, c_j;

$d_{i,l}$ – відстань між кластерами c_i, c_l;

$d_{j,l}$ – відстань між кластерами c_j, c_l;

$\alpha_i, \alpha_j, \beta, \gamma$ – вагові коефіцієнти.

У методі медіан використовуються наступні значення коефіцієнтів: $\alpha_i = 0{,}5$, $\alpha_j = 0{,}5$, $\beta = 0{,}25$, $\gamma = 0$ [28].

У дівизимних алгоритмах вихідна множина представляється як єдиний кластер. Таким чином, на першому кроці маємо:

$$c_1 = \{q_1, q_2, ..., q_{n_1}\}, n_1 = n. \quad (2.13)$$

На другому кроці алгоритму обирається об’єкт q_r. Необхідно, щоб цей об’єкт був найбільш віддалений від інших об’єктів у цьому кластері. Віддалення об’єкту q_r визначається як найбільша середня відстань до інших об’єктів кластера і розраховується за формулою:

$$d_r = 1/n_1 \times \sum d(q_r, q_k) \forall q_k \in c_1 . \quad (2.14)$$

Після цього формується новий кластер c_2. Обраний об'єкт q_r видаляється з кластеру c_1 та поміщається до кластеру c_2 ($n_2 = 1$). На наступних кроках алгоритму об'єкти з кластера c_1, у яких різниця значень між середньою відстанню до об'єктів у c_1 та середньою відстанню до об'єктів у c_2 є найбільшою, переносяться до c_2. Перенесення об'єктів з c_1 до c_2 продовжується доти поки різниці середніх відстаней не стануть негативними. У результаті виконання послідовності кроків формуються два кластери.

На наступному кроці, застосовується щойнорозглянута процедура поділу до одного з сформованих кластерів. На основі оцінки діаметрів кластерів здійснюється вибір кластера для поділу. У свою чергу, оцінка діаметру кластерів виконується із застосуванням формули:

$$D_k = \max d(q_i, q_j) \ \forall q_i, q_j \in c_k, \ k = 1, 2, ..., g . \quad (2.15)$$

Поки всі члени одного кластера не відповідатимуть вимозі близькості, поділ кластерів буде продовжуватись. Також, поділ кластерів може бути зупинен, якщо кластери будуть містити лише по одному об'єкту.

При заданій цільовій функції, поділ об'єктів забезпечують неієрархічні алгоритми. Таким чином, при поділі об'єктів, основною метою є досягнення максимуму або мінімуму цільової функції.

Точки даних на графіку, наведеному на рисунку 2.3, які згруповані разом, можна класифікувати в одну групу. Дійсно, можна розрізнити скупчення. Відзначаємо, що на зображенні нижче є 3 скупчення.

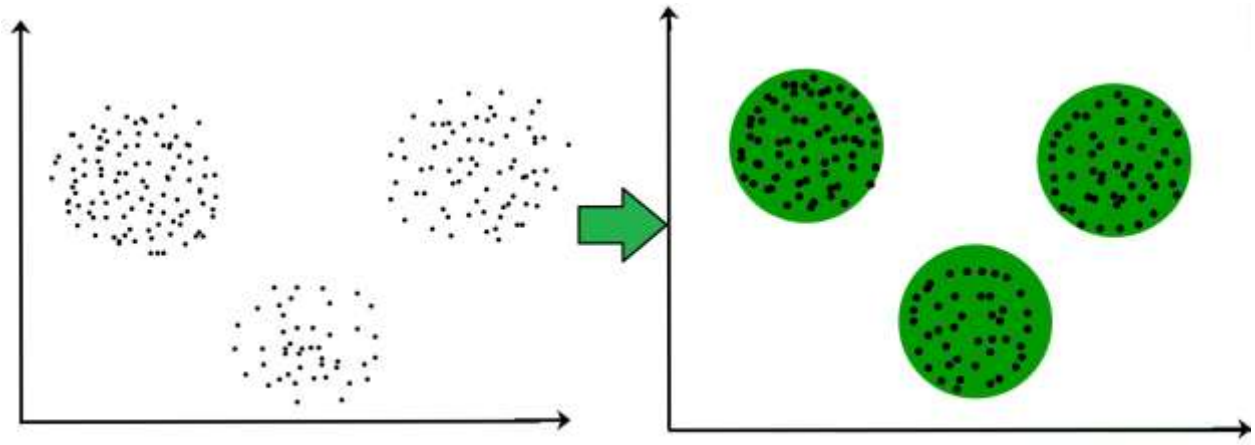

Рисунок 2.3 – Кластеризація точок

Ці точки даних кластеризовані за допомогою базової концепції, згідно з якою точка даних знаходиться в межах даного обмеження від центру кластера. Для розрахунку викидів використовуються різні методи та прийоми відстані.

Кластеризація дуже важлива, оскільки вона визначає внутрішнє групування серед наявних немаркованих даних. Немає критеріїв хорошої кластеризації. Це залежить від користувача, якими критеріями можна скористатися так, щоб задовільнити потреби. Наприклад, можна бути зацікавленим у пошуку представників однорідних груп (скорочення даних), у пошуку «природних кластерів» та описі їх невідомих властивостей («природні» типи даних), у пошуку корисних та придатних груп («корисні» класи даних) або у пошуку незвичних об’єктів даних (виявлення сторонніх даних). Алгоритм повинен робити деякі припущення, які становлять подібність точок, і кожне припущення робить різні і однаково справедливі кластери.

Векторне квантування (VQ – Vector Quantization) відносять до сучасних методів апроксимації в просторах багатовимірних сигналів [30, 31]. Метод VQ здійснює дискретну апроксимацію вхідних даних з множини N безперервних векторів $x \in R^n, \mathrm{N} \subseteq R^n$ за допомогою фіксованого числа k

кодуючих векторів $m_i \in R^n$, $i = 1,2,...,k$. Для сформованого набору кодуючих векторів $M = \{m_i\}_{i=1}^k$ апроксимація довільного вектора $x \in N$ конкурентним способом означає визначення найближчого до нього вектора $m_\upsilon \in M$ у просторі кодуючих векторів:

$$\upsilon = \arg \min_{i=1,...,k} \rho(\mathrm{x},\mathrm{m}_i). \qquad (2.16)$$

Один з варіантів оптимальної побудови набору $M = \{m_i\}_{i=1}^k$ може бути зведений до мінімізації середньоквадратичної помилки дискретизації. До теперішнього часу, рішення задачі оптимального розміщення множини $M = \{m_i\}_{i=1}^k$ у вхідній множині N, в явному вигляді, аналітично, ніхто не знайшов. Зазвичай, дану задачу прийнято вирішувати ітеративним способом з достатньо швидкою збіжністю [31].

2.3 Пакетні і онлайн методи кластеризації

Варіант пакетних обчислень на практиці реалізується у вигляді алгоритму, який представлений у вигляді обчислювальної схеми. Ця схема застосовується у ситуаціях, коли множина N доступна у повному обсязі вже на початку навчання. При цьому усі $x \in N$ вважають рівноцінними для навчання.

Один з найпоширеніших варіантів пакетної обробки отримав назву *k*-середніх (*k*-means). Пропонується застосувати цей алгоритм для довільних метрик при порівнянні елементів. Алгоритм розміщує центри кластерів (центроїди) так, щоб середні значення для списків елементів, які утворилися всередині побудованих кластерів, максимально можливо відрізнялись між собою.

Кластеризація даним методом є одним з найпоширеніших алгоритмів машинного навчання та роботи із даними. Як кажуть експерти з області машинного навчання та аналізу даних – мета *K*-means проста: згрупувати подібні точки даних разом і виявити основні закономірності. Для досягнення цієї мети *K*-means шукає фіксовану кількість (*k*) кластерів у наборі даних.

Кластер відноситься до сукупності точок даних, об'єднаних разом через певну схожість.

Визначається цільове число *k*, яке відноситься до кількості центроїдів, яка потрібна в наборі даних. Центроїд – це уявне або реальне місце, що представляє собою центр скупчення. Кожна точка даних призначається кожному з кластерів за рахунок зменшення суми квадратів у кластері. Кажучи іншими словами, алгоритм *K*-means визначає (*k*) кількість центроїдів. Після цього розподіляє кожну точку даних до найближчого кластера. При цьому, центроїди зберігаються якомога меншими. «Means» в *K*-means означає середнє значення даних; тобто знаходження центроїда [30, 31].

Далі наведено етапи обчислень алгоритму.

Етап 1. У якості k початкових центроїдів беруть довільні k векторів з N:

$$M = \{m_i\}_{i=1}^{k},\ m_i \in N. \tag{2.17}$$

Етап 2. Для кожного $i = \overline{1,k}$ шляхом навчання конкурентного типу (1) формує список $N_i \subseteq N$ тих елементів, для яких найближчим кодуючим вектором є m_i. Це означає, що складаються подмножини $N_i = \{x \in N \mid \arg\min_{\upsilon} \rho(x, m_\upsilon) = i\}$; іншим варіантом побудови списку N_i – є включення в нього зразків, що знаходяться в межах області сусідства для m_i: $N_i = \{x \in N \mid \rho(x, m_i) \le \varepsilon_\rho\}$. Мається на увазі, що елементи, які розташовані всередині кулі радіусом ε_ρ з центром m_i; при цьому $N = \cup N_i$, $N_i \cap N_j = \varnothing$.

Етап 3. У якості чергового значення m_i обчислюютьється середнє значення в списку N_i, який було отримано на попередньому етапі:

$$m_i = \sum_{\upsilon=1}^{s(i)} \frac{x_\upsilon}{s(i)}, \tag{2.18}$$

де $s(i)$ – потужність N_i.

Етап 4. Повторюються етапи 2-3 кілька разів до досягнення збіжності.

Алгоритм особливо ефективний у тих випадках, коли початкові значення векторів попередньо, деяким чином, були узгоджені з навчальною множиною N. Алгоритм не містить параметра швидкості навчання, тому управляти їм або оцінювати його в ході навчання не потрібно, і проблем зі збіжністю немає. Зупинку на етапі 4 здійснюють у тому випадку, якщо на черговій ітерації зміни центрів кластерів стають незначними. Це відповідає застосуванню критерію виду

$$\Delta(M[h+1], M[h]) \le \varepsilon_M, \tag{2.19}$$

де Δ – деяка міра відмінності двох списків центроїдів;

$M[h]$– значення списку на кроці h ітерації;

ε_M – апріорно задана похибка.

Прикладом критерію Δ в (2) може бути сумарна відстань між списками центрів:

$$\Delta = \sum_{i=1}^{k} \rho(m_i(h+1), m_i(h)), \tag{2.20}$$

де $M[h] = \{m_i(h)\}_{i=1}^{k}$, а відстань на черговій ітерації може бути перерахована тільки для змінених центроїдів.

Аналіз змінних схеми k-середніх показує, що, зазвичай, достатньо зробити невелику кількість ітерацій. Алгоритм k-середніх апроксимує функцію щільності розподілу множини вхідних зразків за критерієм мінімуму помилки – суми їх квадратів відхилень від центрів сформованих кластерів [32].

$$E = \sum_{i=1}^{k} \sum_{\upsilon=1}^{s(i)} \rho^2(x_\upsilon, m_i).. \tag{2.21}$$

Як результат, ітераційний алгоритм сходиться до локального мінімуму помилки E.

За рахунок процесу навчання, традиційний метод *k*-середніх виробляє центри кластерів. Ці центри можна інтерпретувати як базиси, в яких можна розкласти довільну множину вхідних сигналів. Недоліком обчислень, які засновані на усередненні, безсумнівно, є чутливість до аномальних викидів, які, нажаль, викривлюють середнє. Але існує рішення цієї проблеми для методу k-середніх, яке полягає у використанні його модифікації – метода *k*-медіани, де на етапі 3, замість середнього обчислюється медіана $m_i = med\,\{x_\upsilon\}_{\upsilon=1}^{s(i)}$. Медіана множини – це елемент, сумарна відстань якої до інших елементів є мінімальною [33].

Векторне квантування, що навчається, найбільш часто застосовується для вирішення завдань статистичного розпізнавання образів, які представлені у вигляді зашумлених випадкових даних високої розмірності. Головною перевагою, в порівнянні з традиційними статистичними методами, є зменшення обчислювальної трудомісткості вирішення завдань. Відомі численні застосування і модифікації методу, наприклад, в мережах векторного квантування, для оптимального вибору початкових центроїдів, в алгоритмах глибокого навчання для згортальних нейронних мереж і ін. [34].

2.4 Локальні дескриптори зображення

Одним з самих вдалих локальних дескрипторів зображення, які були відкриті за останні десять років, – метод маштабно-інваріантного перетворення ознак (Scale-Invariant Feature Transform, SIFT), який був сформульований Девідом Лоуі (David Lowe). Пізніше метод SIFT був вдосконалений та детально описан у статті [32] і у такому вигляді витримав іспит часом. SIFT включає детектор особливих точок та дескриптор. Дескриптор дуже стійкий, й саме завдяки ньому SIFT завоював таку популярність. З моменту його появи було запропоновано багато альтернатив, у яких використовується майже той самий дескриптор. Сьогодні він поєднується з найрізноманітнішими детекторами ключових точок (та областей), а іноді застосовується навіть до всього зображення. Ознаки, які виділяються методом SIFT, інваріантні відносно маштабування, повороту та зміни яскравості. Вони надійно зіставляються навіть для проекцій трьохмірних зображень при різному положенні камери та при наявності шуму.

Особливі точки SIFT знаходить за допомогою різниці гаусіанів:

$$D(x,\sigma) = [G_{k\sigma}(x) - G_{\sigma}(x)] * I(x) = [G_{k\sigma} - G_{\sigma}] * I = I_{k\sigma} - I_{\sigma}, \quad (2.22)$$

де G_σ – це двумірне гаусове ядро зі стандартним відхиленням σ;

I_σ – це результат G_σ – розмиття напівтонованого зображення;

k – постійний множник, який визначає розділення при маштабуванні. Особливі точки шукаються у вигляді мінімумів і максимумів функції $D(x, \sigma)$ від точки зображення та маштабу. З найдених таким чином кандидатів виключаються нестійкі точки. Для виключення використовуються декілька критеріїв, а саме низька контрасність та потрапляння точки за межу.

Описаний вище детектор особливих (ключових) точок дає положення та маштаб. Для того, щоб домогтися інваріантності відносно повороту, обирається опорний напрямок, виходячи направлення та модуля градієнту зображення в околиці точки. Опорним вважається опорний напрямок, який визначається за допомогою гістограми орієнтацій (яка була зважена за модулем градієнту).

Наступний крок – обчислення дескриптора на основі положення, маштабу та напряму. Для того, щоб домогтися незалежності від яскравості, у дескрипторі SIFT використовуються градієнти зображення. Ми беремо сітку підобластей в окрузі точки та для кожної підобласті вираховуємо гістограму орієнтацій градієнту зображення. Ці гістограми об’єднуються, й у результаті виходить вектор дескриптору. Стандартно беруть підобласті розмірами 4 × 4 та гістограми орієнтацій з 8 інтервалами, так що в результаті виходить гістограма з 128 інтервалами (4 * 4 * 8 = 128). На рисунку 2.4 показана побудова дескриптора.

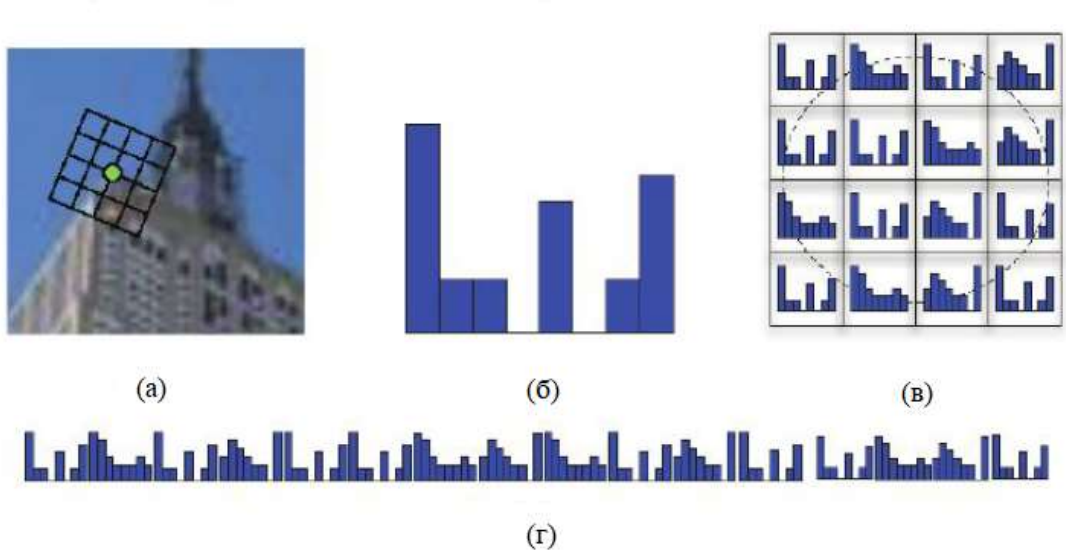

Рисунок 2.4 – Побудова вектора ознак для SIFT-дескриптора:
(а) сітка навколо особливої точки, яка орієнтована уздовж домінуючого напрямку градієнту; (б) 8-інтервальна гістограма орієнтацій градієнту у частині сітки; (в) гістограми, які побудовані у кожній комірці сітки;
(г) гістограми об’єднуються в один довгий вектор ознак

3 РЕЗУЛЬТАТИ КОМП'ЮТЕРНОГО МОДЕЛЮВАННЯ

3.1 Обґрунтування вибору середовища програмної реалізації

У рамках атестаційної роботи розроблений алгоритм для класифікації зображень із використанням кластерних структур для швидкого пошуку даних. Для реалізації роботи обрана бібліотека з відкритим доступом OpenCV. Це обумовлено можливістю безкоштовного використання даної бібліотеки та можливості вільного продажу програмного забезпечення написаного з її використанням, а також висока швидкість, що обумовлюється високопродуктивною мовою C++, й на кінець великий обсяг готових рішень зв'язаних з обробленням зображень [35].

OpenCV (Open Source Computer Vision Library) – це бібліотека програмного забезпечення з відкритим кодом для комп'ютера та машинного навчання. Бібліотека написана об'єктно орієнтованою мовою програмування C++ і має модулі, які охоплюють різноманітні області комп'ютерного зору. Також існує і постійно розвивається підтримка інтерфейсу із мовою програмування Python, яка є більш простою та зручнішою у використанні мовою. Цей інтерфейс продовжує розроблюватись – ще не всі частини OpenCV охоплені. Але у майбутньому положення, обов'язково, зміниться, тому що за розробкою інтерфейсу стоїть активна спільнота.

Далі, у якості прикладу на рисунку 3.1 наведено зображення, на якому було побудовано 500 дескрипторів за допомгою бібліотеки OpenCV.

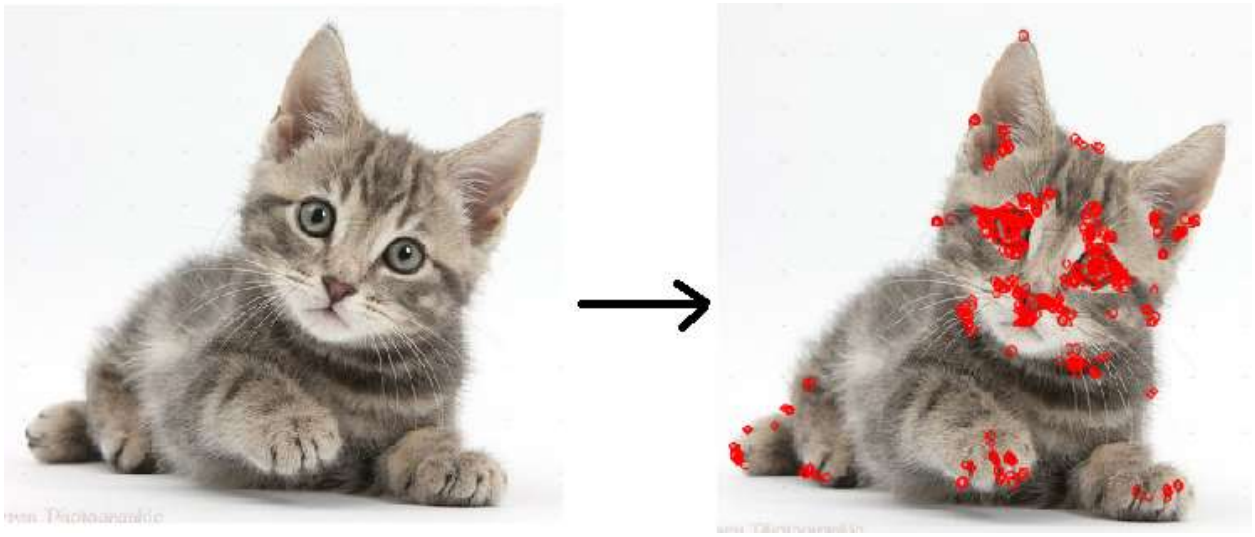

Рисунок 3.1 – Результат побудови ключових точок

Причини вибору мови програмування Python:

– мова Python проста у використанні та вивченні для початківців та новачків. Мова python є однією з найдоступніших для розуміння серед існуючих мов програмування, оскільки вона спрощує синтаксис і не ускладнює, що робить більший акцент на природній мові. Завдяки простоті навчання та використання, скрипти python можна легко писати та виконувати набагато швидше, ніж інші мови програмування. Передбачуване виконання програм є важливою перевагою для побудови систем реального часу. Весь код, неявно генерується компілятором для реалізації мовних можливостей (наприклад, при перетворенні змінної до іншого типу), визначений в стандарті. Також строго визначені місця програми, в яких цей код виконується. Це дає можливість заміряти або розраховувати час реакції програми на зовнішні події;

– Python був створений більше 30 років тому, тобто багато часу для того, щоб будь-яке співтовариство мови програмування адекватно зростало та дозрівало для підтримки розробників, починаючи від початківців і закінчуючи рівнями експертів. Існує велика кількість документації, посібників та відео-посібників для мови Python, за допомогою яких навчаються. Розробники будь-якого рівня кваліфікації та віку можуть

використовувати та отримувати підтримку, необхідну для підвищення своїх знань з мови програмування python. Використовуючи шаблони, можливо створювати узагальнені контейнери і алгоритми для різних типів даних, а також спеціалізувати і обчислювати на етапі компіляції [36];

– завдяки своїй корпоративній спонсорській підтримці та великій підтримці спільноти, python має чудові бібліотеки, якими можна заощадити свій час та зусилля на початковому циклі розробки. Існує також багато хмарних медіа-сервісів, які пропонують підтримку між платформами за допомогою бібліотечних інструментів, що може бути надзвичайно корисним;

– універсальність мови Python – це те, що вона може використовуватися в багатьох різновидах середовищ, таких як мобільні додатки, настільні програми, веб-розробка, апаратне програмування та багато іншого. Універсальність python робить її більш привабливою для використання завдяки великій кількості додатків;

– хмарні обчислення, машинне навчання та великі дані – одні з найгарячіших тенденцій у світі комп'ютерних наук зараз, що допомагає багатьом організаціям трансформувати та вдосконалювати свої процеси та робочі процеси. Сотні бібліотек python щодня використовуються у тисячах проектів машинного навчання, таких як TensorFlow для нейронних мереж та OpenCV для комп'ютерного зору тощо;

– мова Python може якісно допомогти в автоматизації завдань, оскільки існує безліч інструментів та модулів, що робить речі набагато комфортнішими. Неймовірно знати, що можна легко досягти просунутого рівня автоматизації, просто використовуючи необхідні коди python. Python – найкращий прискорювач продуктивності в автоматизації тестування програмного забезпечення. Можна вразитись тим, наскільки менше часу та кількох рядків потрібно для написання кодів засобів автоматизації.

3.2 Особливості програмної реалізації

Для роботи алгоритму та програмного забезпечення створено базу вхідних зображень. Ця база являє собою два еталони. Кожен еталон – це схоже за тематикою зображення. Також, для виконання класифікації, на вхід подається зображення. Вхідне зображення подається у поверненому стані для того, щоб перевірити якість та час виконання класифікації зображень за допомогою класичного лінійного методу та швидкісного методу класифікації зображень із використання кластерних структур даних.

Далі наведено вхідні зображення та еталони.

Вхідні зображення та еталони наведені у Додатку А (рисунки А.1, А.2).

За допомогою бібліотеки OpenCV визначаються дескриптори КТ для кожного із зображень. На кожному з них будується чітко фіксована кількість дескрипторів, а саме для 500 ключових точок. Це означає, що кількість дескрипторів, які належать вхідному зображенню, першому та другому еталону, дорівнює 500 ключових точок.

Тестові зображення, на яких демонструється результат роботи знаходження особливих точок за допомогою бібліотеки OpenCV, наведені у Додатку А (рисунки А.3, А.4).

Для побудови дескрипторів ключових точок використовується бібліотека мови програмування Python, чий метод повертає колекцію дескрипторів, кожен з яких складається з тридцяти двох десяткових чисел, але для подальших розрахунків та досліджень необхідно конвертувати значення дескрипторів у бітовий вигляд. Для цього було створено наступну програмну реалізацію

Лістинг 3.1. Скрипт для конвертування числа у бітовий вигляд:

```
def bitfield(n):
    result = [int(digit) for digit in bin(n)[2:]]
    len_result = len(result)
    new_result = []
    if len_result < 8:
        amount_of_zeros = 8 - len_result
```

```
        for i in range(amount_of_zeros):
            new_result.append(0)
        for i in range(len(result)):
            new_result.append(result[i])
        return new_result
    return result
```

Результат роботи методу bitfield(), який конвертує десятичне число у бітовий вигляд наведено на рисунку 3.2.

```
Enter a number to convert it -> 7
[0, 0, 0, 0, 0, 1, 1, 1]

Enter a number to convert it -> 180
[1, 0, 1, 1, 0, 1, 0, 0]
```

Рисунок 3.2 – Результат роботи методу bitfield, який конвертує десятичне число у бітовий вигляд

Лістинг 3.2. Скрипт для перетворення дескриптора у бітовий вигляд:

```
def return_array_of_256_bits(point):
    result_array = []
    for byte in range(len(point)):
        bit_point = bitfield(point[byte])
        for x in range(len(bit_point)):
            result_array.append(bit_point[x])
    return result_array
```

Лістинг 3.3. Скрипт для перетворення усі дескрипторів еталону у бітовий вигляд:

```
def convert_32_descriptors_to_256_bit(descriptors):
    result_matrix = []
    for keypoint in range(len(descriptors)):
        a = return_array_of_256_bits(descriptors[keypoint])
        result_matrix.append(a)
    return result_matrix
```

У результаті виконання скриптів, що наведено вище, маємо масив дескрипторів зображення, який представлено у бітовому вигляді.

Тепер, маючи усі необхідні дані, можна на практиці порівняти два методи класифікації зображень – класичний лінійний метод та швидкий

метод із використанням кластеризації масиву дескрипторів. Дослідження полягає у вимірюванні швидкості виконання та точності класифікації за результатами моделювання кожного з наведених методів.

Необхідно програмно реалізувати наступну логіку: виконати пошук найменшої відстані Геммінга між кожним дескриптором вхідного зображення та дескрипторами першого та другого еталонів, тобто виконати повний перебір масиву еталонних дескрипторів; проаналізувати та визначити до якого з класів необхідно віднести кожен дескриптор вхідного зображення – якщо *i*-дескриптор зображення має найменшу відстань за метрикою Геммінга до дескриптора, який належить до першого еталону, тоді дескриптор зображення відноситься до першого класу, якщо найближчим є дескриптор другого еталону, тоді – до другого класу.

Лістинг 3.4. Скрипт для пошуку найменшої відстані Геммінга для дескриптора вхідного зображення та набору дескрипоторів першого та другого еталонів:

```
def findMinimalHammingDistanceForDescriptor(descriptor, etalons):
    min_distance: float = 257
    index = 1
    index_of_min_distance = 0
    while index < len(etalons):
        current_distance = my_hamming_distance(descriptor, etalons[index])
        # print(f'{index}) Current distance: {current_distance}')
        if min_distance >= current_distance > 0.0:
            index_of_min_distance = index
            min_distance = current_distance
        index += 1
    return index_of_min_distance
```

Лістинг 3.5. Скрипт для класифікації вхідного зображення на основі відстані Геммінга для кожного з дескрипоторів:

```
first_class: int = 0
    second_class: int = 0
    for i in range(0, 500):

      index_from_etalon = \

      findMinimalHammingDistanceForDescriptor(descriptors_liverpool_side_bit_fm[i],
      etalon_liverpool_plus_lester_bit_format)

        if index_from_etalon < 500:
            first_class += 1
```

```
        else:
            second_class += 1

    print(f'First class: {first_class}')
    print(f'Second class: {second_class}')
```

Скрипт, що наведено вище, аналізує та визначає до якого з класів необхідно віднести кожен дескриптор вхідного зображення. Оскільки кількість дескрипторів у кожному з еталонів дорівнює 500, дескриптори вхідного зображення класифікуються наступним чином: якщо індекс дескриптора з набору двох еталонів менший за 500 та його відстань Геммінга до поточного дескриптора вхідного зображення була найменшою, то даний дескриптор потрапляє до першого класу; якщо індекс дескриптора з набору еталонів більший або дорівнює 500 та його відстань Геммінга до поточного дескриптора вхідного зображення була найменшою, то даний дескриптор потрапляє до другого класу.

Для класифікації кожного дескриптора вхідного зображення необхідно виконати 1000 порівнянь, це означає, що даний метод виконує повний перебір, а складність алгоритму складає $O(n^2)$.

Користуючись класичним лінійним методом класифікації зображень, було отримано наступний результат: час виконання дорівнює 30,11 секунд, дескриптори розподілилися по класам наступним чином: до першого класу було віднесено 472 дескрипторів, до другого – 28. Отже, точність виконання даного методу складає 0,94.

Теоретично метод швидкої класифікації за допомогою впровадження кластерних структур має зменшити час виконання класифікації зображень у порівнянні із лінійним методом. Це досягається шляхом розбиття множини дескрипторів по кластерам. За рахунок цього, замість того, щоб працювати одразу з усією колекцією дескрипторів, моємо можливість працювати з окремою групою дескрипторів із схожими показниками.

Отже, отримавши колекцію бітових дескрипторів КТ, тепер необхідно розподілити дескриптори першого та другого еталонів по кластерам, тобто їх

кластеризувати. Для виконання кластеризації дескрипторів у рамках атестаційної роботи було обрано метод *K*-means.

Для виконання кластеризації дескрипоторів методом *K*-means використовується бібліотека мови програмування Python, яка має назву sklearn. Далі наведено скрипт, за допомогою якого дескриптори другого еталону було розподілено по кластерам.

Лістинг 3.6. Скрипт, що виконує кластеризацію дескрипторів еталонів:

```
kmeans = KMeans(n_clusters=8,random_state=0).fit(second_etalon_bit_format)
array_after_clustering = kmeans.labels_
centers_of_clusters = kmeans.cluster_centers_
```

Як можна побачити із наведеного скрипта, функція KMeans надає можливість обрати яку саме кількість кластерів користувач бажає отримати на виході. Дана опція робить використовувану функцію більш адаптивною та гнучкою, і, саме ця опція активно використовується у рамках даної атестаційної роботи і відіграє важливу роль. Також дана функція надає можливість отримати на виході центри усіх кластерів.

У даному випадку кількість кластерів дорівнює 8. Далі, на рисунку 3.3, можна побачити результати кластеризації еталонів.

Кожен з дескрипторів еталонів отримує індекс кластера (від 0 до 7), до якого він потрапив. Перший елемент матриці, що наведена на рисунку вище, – це індекс кластера до якого було віднесено перший дескриптор першого еталону, тобто цей дескриптор, у даному випадку, потрапив до шостого кластера.

Рисунок 3.3 – Результат кластеризації еталонів

Далі використовується метод, який підраховує загальну кількість дескрипторів еталонів у кожному із кластерів.

Лістинг 3.7. Скрипт, який підраховує кількість дескрипторів еталонів у кожному із кластерів:

```
amount_of_elements_in_cluster = \
        calculate_amount_elements_at_cluster(array_after_clustering, 0,
len(array_after_clustering))

    amount_of_elements_in_cluster_first_class = \
        calculate_amount_elements_at_cluster(array_after_clustering, 0,
(int(len(array_after_clustering) / 2)))

    amount_of_elements_in_cluster_second_class = \
        calculate_amount_elements_at_cluster(array_after_clustering,
                                             (int(len(array_after_clustering) / 2)),
len(array_after_clustering))
```

На рисунку 3.4 можна побачити, як було розподілено дескриптори еталонів. Перший рядок демонструє загальну кількість дескипторів двох еталонів. Далі, у другому та третьому рядках, продемонстровано скільки дескрипторів, першого та другого еталонів відповідно, потрапило до кожного кластера.

```
Amount of elements at each cluster

[113, 76, 193, 140, 113, 128, 126, 111]
First etalon: [61, 25, 141, 71, 69, 59, 40, 34]
[52, 51, 52, 69, 44, 69, 86, 77]
```

Рисунок 3.4 – Кількість дескрипторів еталонів у кожному кластері

Наступним кроком є визначення того, до якого з центрів кластерів знаходиться ближче кожен з дескрипторів зображення, що подається на вхід. Для цього використовується відстань Геммінга. Але існує проблема – центри кластерів, які було отримано у результаті роботи методу KMeans, не знаходяться у бітовому форматі, а для того, щоб працювати надалі з метрикою Геммінга в рамках даної атестаційної роботи, необхідно, щоб центри кластерів мали саме цей формат. Тому було розроблено спеціальній метод, який виконує конвертацію кожного центру кластера до бітового виду. Нижче, наведені скрипт та зображення (рисунки 3.5, 3.6), на яких можна побачити вигляд центру першого кластеру до і після роботи методу round_array_of_centers(), який виконує конвертацію центрів кластерів до бітового вигляду.

Лістинг 3.8. Скрипт, за допомогою якого виконується конвертація формату центру кластера до бітового:

```
def round_array_of_centers(input_array):
    rounded_array = input_array
    for i in range(len(input_array)):
        for j in range(len(input_array[i])):
            if input_array[i, j] < 0.5:
```

```
            rounded_array[i, j] = 0
        else:
            rounded_array[i, j] = 1
return rounded_array
```

```
Not converted first cluster`s center:
[0.39823009 0.60176991 0.47787611 0.52212389 0.38053097 0.68141593
 0.97345133 0.25663717 0.5840708  0.46902655 0.68141593 0.61946903
 0.53097345 0.38938053 0.38053097 0.53097345 0.69026549 0.46017699
 0.85840708 0.31858407 0.46902655 0.50442478 0.61061947 0.26548673
 0.78761062 0.2920354  0.36283186 0.56637168 0.54867257 0.33628319
 0.40707965 0.18584071 0.66371681 0.6460177  0.42477876 0.40707965
 0.79646018 0.51327434 0.7079646  0.79646018 0.56637168 0.75221239
 0.50442478 0.69911504 0.19469027 0.3539823  0.28318584 0.49557522
 0.75221239 0.14159292 0.46902655 0.89380531 0.74336283 0.47787611
 0.45132743 0.28318584 0.7079646  0.84070796 0.36283186 0.15929204
```

Рисунок 3.5 – Фрагмент центру першого кластеру

```
Converted first cluster`s center:
[0. 1. 0. 1. 0. 1. 1. 0. 1. 0. 1. 1. 1. 0. 0. 1. 1. 0. 1. 0. 0. 1. 1. 0.
 1. 0. 0. 1. 1. 0. 0. 0. 1. 1. 0. 0. 1. 1. 1. 1. 1. 1. 1. 1. 0. 0. 0. 0.
 1. 0. 0. 1. 1. 0. 0. 0. 1. 1. 0. 0. 1. 0. 1. 1. 1. 0. 0. 0. 1. 0. 1. 1.
 1. 1. 0. 1. 1. 1. 1. 1. 1. 0. 0. 1. 1. 1. 0. 1. 1. 0. 1. 0. 1. 1. 1. 1.
 0. 0. 0. 0. 1. 1. 0. 0. 1. 1. 1. 0. 1. 1. 1. 0. 0. 1. 1. 1. 1. 1. 1. 1.
 0. 1. 0. 0. 1. 0. 1. 0. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 0. 1. 0. 0. 1. 1.
 1. 1. 1. 1. 1. 0. 1. 1. 1. 0. 0. 1. 0. 1. 1. 1. 1. 0. 0. 1. 0. 1. 1. 0.
 0. 1. 0. 0. 0. 0. 1. 1. 1. 0. 1. 0. 1. 0. 1. 0. 1. 1. 0. 1. 0. 0. 0. 0.
 0. 0. 0. 1. 1. 0. 1. 1. 0. 1. 0. 1. 0. 0. 0. 0. 1. 1. 0. 0. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 0. 1. 0. 1. 0. 1. 0. 1. 1. 0. 0. 1. 1. 0.
 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 0. 1. 1. 1.]
```

Рисунок 3.6 – Центр першого кластеру після приведення до бітового вигляду

Конвертувавши кожний з центрів кластерів, було вирішено проблему і тепер існує можливість подальшої роботи з метрикою Геммінга.

Виконується пошук найменшої відстані Геммінга між *i*-дескриптором вхідного зображення та центром кожного із кластерів. У даному випадку, центрів кластерів 8, значить, для кожного з дескрипторів обирається найменша відстань серед восьми варіантів.

Код, що наведено нижче, генерує масив даних, який фіксує індекс кластера, що знаходиться найближче (за метрикою Геммінга) до *i*-дескриптора вхідного зображення.

Лістинг 3.9. Скрипт, який фіксує до якого з центрів кластерів знаходиться найближче *i*-дескриптор вхідного зображення:

```
array_centers_for_each_descriptor = []
    for i in range(0, 500):
        index_from_etalon = \

findMinimalCenterByHammingDistance(descriptors_liverpool_side_bit_format[i],
centers_of_clusters_rounded)
        array_centers_for_each_descriptor.append(index_from_etalon)
```

На рисунку 3.7 видно результат роботи скрипта, який шукає для кожного дескриптора вхідного зображення найближчий центр кластера.

```
Indexes of the closest cluster center for each descriptor:
[3, 2, 5, 3, 1, 5, 6, 4, 0, 2, 7, 6, 5, 2, 0, 4, 1, 2, 3, 2, 6, 3, 4,
```

Рисунок 3.7 – Результат роботи методу findMinimalCenterByHammingDistance

Масив чисел, що наведено вище, означає, що для першого дескриптора вхідного зображення, за метрикою Геммінга, найближчим буде центр третього кластера, для другого – центр другого кластера і так далі.

Метод швидкої класифікації зображення за допомогою кластерних структур даних дозволяє відмовитись від способу повного перебору дескрипторів, коли по черзі виконується пошук найменшої відстані Геммінга між усіма дескрипторами вхідного зображення та усіма дескрипторами першого та другого еталонів. Замість цього, достатньо виконати пошук найменшої відстані Геммінга між поточним дескриптором вхідного зображення та дескрипторами, які належать до кластера, чий центр знаходиться найближче до поточного дескриптора. Теоретично, даний підхід дозволяє значно заощадити на часі виконання класифікації зображень, але

при цьому є ризик зниження точності розрахунків. Даний підрозділ атестаційної роботи пропонує перевірити на практиці як саме впливає на результати роботи алгоритму зменшення або збільшення кількості кластерів, а також порівняти отримані результати із стандартним лінійним методом.

Лістинг 3.10. Скрипт, який шукає усередині кластера найближчий дескриптор до дескриптора вхідного зображення, а потім аналізує до якого класу віднести дескриптор вхідного зображення:

```
def get_in_cluster_class_of_descriptor(descriptor, cluster, indexes):
    index_of_descriptor = findMinimalHammingDistanceForDescriptor(descriptor,
cluster)
    a = indexes[index_of_descriptor]

    if a < 500:
        return 1
    else:
        return 2
```

Лістинг 3.11. Скрипт, що аналізує до якого класу віднести кожен дескриптор вхідного зображення та підраховує кількість дескрипторів у кожному класі:

```
classes_array = [0, 0]
    for i in range(len(descriptors_liverpool_side_bit_format)):
        value = 0
        if array_centers_for_each_descriptor[i] == 0:
            value = get_in_cluster_class_of_descriptor(
                descriptors_liverpool_side_bit_format[i], first_cluster_values,
first_cluster_indexes
            )
        ...
        if value == 1:
            classes_array[0] = classes_array[0] + 1
        if value == 2:
            classes_array[1] = classes_array[1] + 1
```

Скрипти, що наведено вище, виконують прохід спочатку по усім центрам кластерів, а потім по всім дескрипторам усередині одного з кластерів. Як і у випадку із лінійним методом класифікації зображень, для того, щоб класифікувати кожен дескриптор вхідного зображення, необхідно визначити індекс дескрипторів еталонів, чия відстань до поточного дескриптора є найменшою, перевірити чи даний індекс менший чи більший

за 500 і після цього віднести дескриптор вхідного зображення до першого або другого класів відповідно.

Пропонується підрахувати кількість порівнянь, які необхідно здійснити для класифікації першого дескриптора вхідного зображення. Оскільки кількість центрів кластерів дорівнює 8, а кількість кластерів, що належать до 4-го кластера дорівнює 140 (саме до центру 4-го кластера відстань Геммінга є найменшою для першого дескриптора вхідного зображення), то робиться висновок, що для класифікації поточного дескриптора необхідно виконати лише 148 порівнянь.

Користуючись методом класифікації зображень із використанням кластерних структур даних, було отримано наступний результат: час виконання складає 5,1 секунди, дескриптори розподілилися по класам наступним чином: до першого класу було віднесено 433 дескрипторів, до другого – 67. Отже, точність виконання даного методу складає 0,87.

У рамках даної кваліфікаційної роботи було вирішено провести додаткові дослідження. У попередніх підрозділах було доведено на практиці, що метод швидкої класифікації зображень із використанням кластерних структур дійсно значно зменшує час виконання класифікації, але тепер пропонується дізнатися, як саме впливає на результат роботи методу збільшення або зменшення числа кластерів. Для тестової перевірки пропонується наступна кількість кластерів: 2, 5, 8, 10, 15 та 30. Також, для об’єктивності досліджень, пропонується подавати на вхід і перше і друге зображення. Обидва вхідних зображення було повернуто вліво під кутом нахилу 30 градусів.

Вхідні зображення з виділеними ключовими точками наведені у Додатку А (рисунки А.5, А6).

Графіки, які демонструють час та точність виконання для різної кількості кластерів, наведені на рисунках 3.8, 3.9.

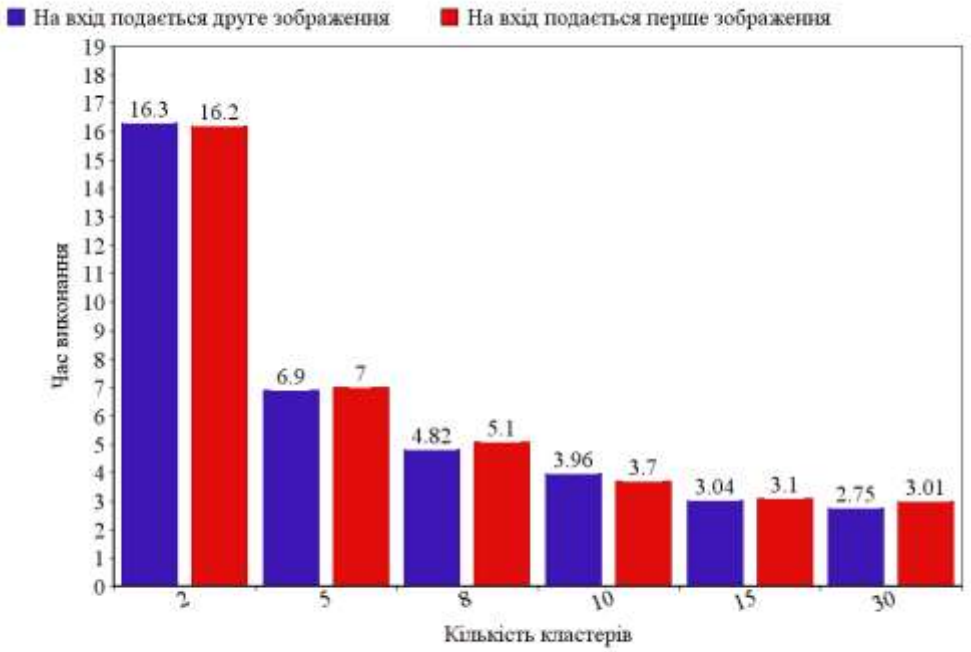

Рисунок 3.8 – Час виконання методу швидкого пошуку для різної кількості кластерів

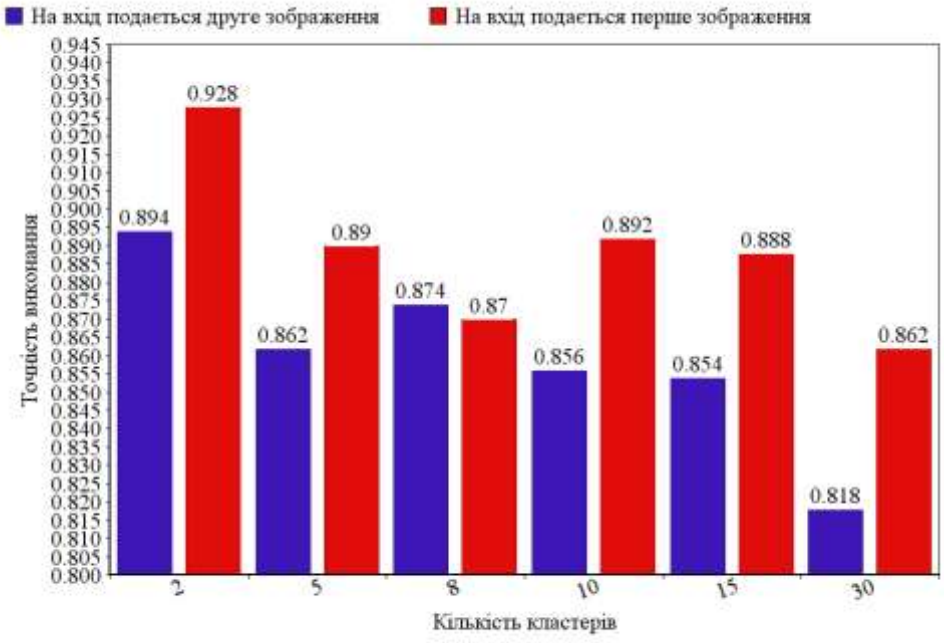

Рисунок 3.9 – Точність виконання методу швидкого пошуку для різної кількості кластерів

На графіках, що неведені вище, моделюються ситуації, коли по черзі на вхід подаються різні зображення для виконання швидкої класифікації із використанням різної кількості кластерів.

Для откримання результатів, які проілюстровано на рисунку 3.8 виконувалось замірювання часу виконання методу швидкого пошуку із використанням кластерних структур даних, при цьому заміри часу виконання проводились для різної кількості кластерів. Графік наглядно дементсрує скільки займає час виконання швидкого пошуку для певної кількості кластерів, наприклад, використовуючи для класифікації 10 кластерів і подаючи на вхід друге зображення, можна отримати результат часу виконання, що приблизно дорівнює 4 секундам.

Для отримання рузультатів, що наведено на рисунку 3.9 розраховувалась точність виконання методу швидкого пошуку для різної кількості кластерів. Стовідсотковий (або 1,0) результат точності класифікації першого або другого зображення відбувається лише у тому випадку, коли усі 500 дескрипторів вхідного зображення було віднесено до першого або другого класів, відповідно. Якщо, класифікуючи перше зображення, було отримано результат, що до першого класу віднесено 485 дескрипторів, а до другого – 15, тоді точність вираховується наступним чином: 485 / 500 = 0,97. Точність класифікації у даному випадку складає 97 відсотків, що є високим паказником точності методу швидкого пошуку.

Орієнтуючись на результати, що наведено на графіках, можна зробити наступні висновки: збільшення числа кластерів допомагає прискорити виконання методу швидкої класифікації, але, при цьому, втрачається точність класифікації. Тож, при виборі числа кластерів для роботи з методом швидкісного пошуку, який використовує кластерні структури даних, необхідно знайти компроміс між швидкістю та точністю.

ВИСНОВКИ

У рамках кваліфікаційної роботи було програмно реалізовано та досліджено методи класифікації зображень із впровадженням засобів швидкісного пошуку, що використовують кластерні структури даних.

Метод класифікації зображення з використанням швидкісного пошуку у порівнянні із лінійним методом суттєво зменшує час обчислення класу об’єкта у 2…6 разів відповідно до кількості кластерів, зменшує час виконання класифікації зображення, але при цьому зменшується точність самої класифікації, приблизно на 5-15 % у залежності від кількості кластерів.

Проведені дослідження з метою порівняння часу та точності класифікації для різного числа кластерів при використанні методу швидкісного пошуку дескрипторів у базі даних. У результаті досліджень стало зрозуміло, що при збільшенні кількості кластерів час виконання зменшується, а точність також зменшується.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Daradkeh Y. I., Tvoroshenko I., Gorokhovatskyi V., Latiff L. A., & Ahmad, N. (2021). Development of Effective Methods for Structural Image Recognition Using the Principles of Data Granulation and Apparatus of Fuzzy Logic. IEEE Access, 9, 13417-13428.

2. Гороховатский, В. А. (2014). Структурный анализ и интеллектуальная обработка данных в компьютерном зрении.

3. Путятин, Е. П. (1990). *Обработка изображений в робототехнике*. Москва.

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

5. Гороховатский, В. А., Путятин, Е. П., & Столяров, В. С. (2017). Исследование результативности структурных методов классификации изображений с применением кластерной модели данных. Радіоелектроніка, інформатика, управління, (3), 78-85.

6. Gorokhovatskiy, V. A. (2011). Compression of descriptions in the structural image recognition. *Telecommunications and Radio Engineering*, *70*(15).

7. Gadetska, S. V., & Gorokhovatskyi, V. O. (2018). Statistical measures for computation of the image relevance of visual objects in the structural image classification methods. Telecommunications and Radio Engineering, 77(12).

8. Гороховатский, В. А. (2008). Иерархия пространственных отношений структурных признаков в задачах сопоставления визуальных объектов. *Системи управління, навігації та зв'язку.—Київ: Центральний наук.-досл. ін-т навігації і управління*, 85-89.

9. Гороховатский, В. А. (2003). Распознавание изображений в условиях неполной информации. *Харьков: ХНУРЭ*.

10. Воронцов К.В. Машинное обучение (курс лекций). URL: http://www.machinelearning.ru/wiki/index.php?title=Машинное_обучение_(курс_лекций%2C_К.В.Воронцов) (дата звернення 26.04.2021).

11. Гороховатський, В. О., Пупченко, Д. В., & Солодченко, К. Г. (2018). Аналіз властивостей, характеристик та результатів застосування новітніх детекторів для визначення особливих точок зображення.

12. Gorokhovatskyi V.O., Tvoroshenko I.S., Vlasenko N.V. (2020) Using Fuzzy Clustering in Structural Methods of Image Classification. Telecommunications and Radio Engineering, 79 (9), pp. 781-791.

13. Gorokhovatskyi V., Gadetska S., Ponomarenko R. Recognition of Visual Objects Based on Statistical Distributions for Blocks of Structural Description of Image. Lecture Notes in Computational Intelligence and Decision Making. Proceedings of the XV International Scientific Conference “Intellectual Systems of Decision Making and Problems of Computational Intelligence” (ISDMCI’2019) (Ukraine, May 21–25, 2019). P. 501-512.

14. Загоруйко, Н. Г. (1999). *Прикладные методы анализа данных и знаний*.

15. Гороховатський, В. О., & Гадецька, С. В. (2020). Статистичне оброблення та аналіз даних у структурних методах класифікації зображень.

16. Горелик, А. Л., Гуревич, И. Б., & Скрипкин, В. А. (1985). *Современное состояние проблемы распознавания: Некоторые аспекты*. Радио и связь.

17. Флах, П. (2019). *Машинное обучение. Наука и искусство построения алгоритмов, которые извлекают знания из данных*. Litres.

18. Gorokhovatskyi, V. O., Tvoroshenko, I. S., & Peredrii, O. O. (2020). Image classification method modification based on model of logic processing of bit description weights vector. Telecommunications and Radio Engineering, 79(1).

19. Gorokhovatskiy, V. A., & Putyatin, Y. P. (2009). Image Likelihood Measures on the Basis of the Set of Conformities. Telecommunications and Radio Engineering, 68(9).

20. 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.

21. Чмутов Ю.В. Методи швидкої класифікації зображень на підставі кластерезації: матеріали конференції «Сучасні методи обробки зображень». (Харків, 20-22 квітня 2021 р.). Харків, 2021. С 12-13.

22. Kohonen, T. Self-Organizing Maps. Springer-Verlag, Berlin, Heidelberg doi:book/10.5555/558021. 2001.

23. Gorokhovatsky V.A., Putyatin Ye. P. (2009). Image Likelihood Measures of the Basis of the Set of Conformities. Telecommunications and Radio Engineering. 68 (9). P. 763–778.

24. Gorokhovatsky, V. (2014). Structural analysis and intellectual data processing in computer vision. SMIT: Kharkiv, Ukraine.

25. Гороховатский, В. А., & Передрий, Е. О. (2009). Корреляционные методы распознавания изображений путем голосования систем фрагментов. Радіоелектроніка, інформатика, управління, (1 (20)).

26. Kinoshenko, D., Mashtalir, V., Yegorova, E., & Vinarsky, V. (2005, July). Hierarchical partitions for content image retrieval from large-scale database. In International Workshop on Machine Learning and Data Mining in Pattern Recognition (pp. 445-455). Springer, Berlin, Heidelberg.

27. Babenko A., Slesarev A., Chigorin A., Lempitsky V. (2014) Neural codes for image retrieval. Conference Paper. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 8689 LNCS(PART 1). P. 584-599.

28. Барсегян, А. (2004). Методы и модели анализа данных: OLAP и Data Mining. БХВ-Петербург.

29. Ларсон Б. Разработка бизнес-аналитики в SQL Server 2005. СПб., 2008. 684 с.

30. Дюк В. Data Mining - интеллектуальный анализ данных : URL: http://www.iteam.ru/publications/it/section_55/ article_1448./ (дата звернення 29.04.2021).

31. Барсегян, А. (2008). Технологии анализа данных: Data Mining, Text Mining, Visual Mining, OLAP. 2 изд. БХВ-Петербург.

32. Aggarwal, C. C., & Reddy, C. K. (2014). Data clustering. Algorithms and applications. Chapman&Hall/CRC Data mining and Knowledge Discovery series, Londra.

33. Gorokhovatskyi V., Putyatin Ye., Gorokhovatskyi O., Peredrii O. Quantization of the Space of Structural Image Features as a Way to Increase Recognition Performance. The Second IEEE International Conference on DataStream Mining & Processing (Lviv, 21-25 August 2018 y.), Lviv, 2018. P. 464 – 467.

34. Ye, N. (2013). Data mining: theories, algorithms, and examples. CRC press.

35. Smeulders, A. W., Worring, M., Santini, S., Gupta, A., & Jain, R. (2000). Content-based image retrieval at the end of the early years. IEEE Transactions on pattern analysis and machine intelligence, 22(12), 1349-1380.

36. Berman, A. P., & Shapiro, L. G. (1999). A flexible image database system for content-based retrieval. Computer Vision and Image Understanding, 75(1-2), 175-195.