Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

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

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

ВИВЧЕННЯ МЕТОДУ РЕДУКЦІЇ СИСТЕМИ СТРУКТУРНИХ ОЗНАК ЗОБРАЖЕННЯ НА ОСНОВІ МЕТРИЧНОГО КРИТЕРІЮ ІНФОРМАТИВНОСТІ
(тема)

Виконав:
студент 2 курсу, групи ІНФм-20-1
Метелев В.В.
(прізвище, ініціали)

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

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

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

Допускається до захисту

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

2021 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти другий (магістерський)

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

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

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

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Метелеву Валерію Володимировичу
(прізвище, ім'я, по батькові)

1. Тема роботи Вивчення методу редукції системи структурних ознак зображення на основі метричного критерію інформативності
затверджена наказом по університету від « 22 » жовтня 2021 року № 1574Ст.
2. Термін подання студентом роботи до екзаменаційної комісії 26 листопада 2021 р.
3. Вихідні дані до роботи науково-методична та науково-технічна література література з розпізнавання зображень, матеріали наукових конференцій, дані інтернет-мережі, мова програмування Python, теоретичні відомості про особливості структурних ознак, теоретичні відомості про методи редукції множини ознак зображень.

4. Перелік питань, що потрібно опрацювати в роботі
1. Огляд особливостей систем ознак зображень.
2. Аналіз та вивчення алгоритмів для редукції систем ознак.
3. Огляд використання метричних критеріїв для оцінки інформативності ознак.
4. Побудова моделі редукції системи структурних ознак зображення за метричним критерієм інформативності.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність дослідження, об'єкт та мета дослідження, постановка задачі дослідження, вихідні дані дослідження, етапи розроблення, апробація результатів роботи.

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
Консультант з дотримання діючих стандарт та норм	Доцент Бєлова Н.В.		

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	22.10.2021	
2	Аналіз завдання, підбір літератури	22.10.21–27.10.21	
3	Аналіз літератури з досліджуваної проблеми	27.10.21–30.10.21	
4	Аналіз технічних засобів	30.10.21–01.11.21	
5	Розробка методу редукції ознак	01.10.21–02.10.21	
6	Програмна реалізація	02.11.21–11.11.21	
7	Оформлення пояснювальної записки	11.11.21–17.11.21	
8	Перевірка на плагіат	19.11.2021	
9	Рецензування	19.11.2021	
10	Підготовка презентації та доповіді	25.11.2021	
11	Занесення роботи в електронний архів	04.12.2021	
12	Попередній захист кваліфікаційної роботи	06.12.2021	

Дата видачі завдання 22 жовтня 2021 р.

Студент _____
(підпис)

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

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 70 с., 21 рис., 50 джерел.

КЛАСИФІКАЦІЯ ЗОБРАЖЕНЬ, КЛЮЧОВІ ТОЧКИ, ДЕСКРИПТОР, РЕДУКЦІЯ СИСТЕМИ ОЗНАК, ІНФОРМАТИВНІСТЬ, МЕТРИКА, PYTHON.

Об’єктом дослідження є методи зменшення розмірності систем ознак для представлення даних у задачі класифікації зображень.

Метою є розроблення методів, що дозволяють проводити редукцію системи структурних ознак на основі метричного критерію інформативності.

Використано методи метричного аналізу даних, програмного моделювання. Проведено експериментальне дослідження методів редукції на прикладі множин ознак у вигляді наборів бінарних векторів. Критерієм результативності класифікації вибрані метрики Хаусдорфа та Танімото.

Здійснена програмна реалізація для моделі редукції множини ознак у вигляді дескрипторів ключових точок зображення. Обсяг опису можна скоротити в два рази, не зменшуючи ефективність розпізнавання.

IMAGE CLASSIFICATION, KEY POINTS, DESCRIPTOR, FEATURES SYSTEM REDUCTION, IMPORTANCE, METRIC, PYTHON.

Research objects are dimensionality reduction methods of feature systems for data representing in image classification

The aim of research is development of methods, which allow performing structural features system reduction based on importance criteria metrics.

Methods for data metric analisys and programming modeling methods are used. Performed experimental research of reduction methods using features sets represented as groups of binary descriptors. Hausdorff and Tanimoto metrics are chosen as calssification efficiency criteria.

As a result of implemented image keypoints expressed features reduction model. Total number of description members can be reduced in a half.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

КТ − ключові точки

Кб − кілобайти

Мб − мегабайти

PCA − Primary Components Analysis (аналіз головних компонент)

RGB − Red Green Blue

HSV − Hue Saturation Value

SIFT − Scale-Invariant Feature Transform

SURF − Speeded Up Robust Features

ORB − Oriented FAST and Rotated BRIEF

FAST − Features from Accelerated Segment Test

BRIEF − Binary Robust Independent Elementary Features

ВСТУП

У задачах оброблення чи аналізу зображень ознаки − це властивості зображення, що містять інформацію про об’єкти на зображенні. Вони дозволяють дослідити зображення на предмет його відповідності деяким якостям залежно від предмету дослідження. Інколи ознаки, незалежно від предмету дослідження, можуть бути представлені у надлишку або заважають ефективному процесу обробки через свою кількість. Ця проблема потребує вирішення [1-9].

Для виключення зайвих ознак або пошуку найбільш інформативних у системах комп’ютерного зору існують методи редукції ознак, або як їх ще називають методи зменшення розмірності. Методи редукції ознак поділяють на дві групи: методи виділення ознак і методи відбору ознак [10-12].

Основна ідея у методах виділення ознак полягає у тому, щоб віднайти нові ознаки, представлені у меншій кількості, на базі існуючих, при цьому зберігаючи максимальну кількість інформації, що містять оригінальні ознаки.

Методи відбору ознак покладаються на пошук найбільш інформативних ознак із існуючих та їх відбір, або на пошук найменш інформативних та позбавлення від них.

Актуальним на сьогодні є вивчення методів відбору ознак за критеріями інформативності для дослідження їх ефективності та пошук оптимальних рішень щодо використання цих методів у задачі класифікації зображень.

1 АНАЛІЗ СИСТЕМ СТРУКТУРНИХ ОЗНАК ЗОБРАЖЕННЯ

1.1 Ознаки в системах аналізу зображень

В машинному навчанні та розпізнаванні образів ознаки − це властивості або характеристики спостережуваного явища, які можливо виміряти [1]. Ознаки − це те, яким чином представлені ваші дані. Наприклад, для набору даних про людей, ознаками можуть бути зріст або вага людини, її вік, стать тощо. Ознаки дозволяють нам аналізувати дані та будувати моделі на їх основі. Ознаки також називають атрибутами, властивостями або характеристиками. Колекція ознак разом з їх значеннями формують набір даних, що зазвичай представлений у вигляді матриці або набору векторів, у якому кожен вектор описує деякий об’єкт, їх також називають спостереженням, кортежем, випадком, тощо. Кожна з ознак може приймати безперервні, дискретні або комплексні значення [2].

Безперервні ознаки можуть виступати дійсні числа, для яких можлива кількість різних варіантів нескінченна та не може бути обчислена. Для роботи з такими числами частіше за все використовують методи математично аналізу. Для опису таких значень вдаються до теоретичної статистики, а саме до функції густини ймовірності.

З іншого боку, дискретні ознаки мають значення, що можуть бути обчислені та представлені обмеженим набором варіантів. Найбільш розповсюдженими прикладами таких ознак є цілі числа, бінарні числа. Дискретні ознаки також можна поділити на впорядковані та номінальні. Номінальними можуть бути такі показники як колір, ім’я, тощо. Впорядкованими є ті ознаки, що представлені у вигляді деяких чисел, наприклад позиція або оцінка.

Впорядковані ознаки можуть бути відсортовані як у одному порядку так і обох. Різниця полягає у тому, чи є початок відліку ознаки фіксованим за її

змістом, або його зміна покладена на наш розсуд. Наприклад, якщо треба вказати позицію, можна самому вирішити, 1 − це початкова позиція або остання, у той час коли для деяких ознак, як наприклад вік або вага,зміна впорядкування руйнує їх зміст. Номінальні ознаки в свою чергу не мають жодної послідовності у своїх значеннях. Різні значення таких ознак не мають ніякого порядкового взаємозв'язку між собою. Окремо можна виділити саме бінарні значення. Зазвичай нуль та одиниця представляють собою значення «ні» і «так» відповідно.

Безпреривні ознаки можна заміняти дискретними. Для цього значення безперервних ознак ділиться на діапазони, що в свою чергу все є дискретними значеннями. Нове значення діапазону дістається тому об'єкту, який має непреривну ознаку, що попадає до діапазону. Для прикладу, задається один із діапазонів дат у вигляді «від 01.01.2000 до 31.12.2009», тоді усі об'єкти, що містять дату як непреривну ознаку із значенням що підпадає у цей діапазон отримує це дискретне значення, замість безперервного.

У обробці зображень ознаки відрізняються від звичних ознак у інших задач комп'ютерного аналізу. Ознаки зображень являють собою шматочки інформації про наповнення зображення. Зазвичай це зводиться до якогось певного сегмента на зображенні, та чи присутні у цьому сегменті певні властивості. Ознаками можуть виступати специфічні структури присутні на зображенні, такі як точки, кути або об'єкти що на ньому присутні.

Для зображення у сирому вигляді не має ніяких ознак, тому для того щоб їх отримати використовують спеціальни алгоритми пошуку ознак. Загалом ознаки зображення можна поділити на дві основні категорії:

– ознаки в певних місцях зображень, такі як гірські вершини, кути будівель, дверні прорізи або снігові ділянки цікавої форми. Ці типи локалізованих ознак часто називають ключовими ознаками (або навіть кутами) і часто описуються появою ділянок пікселів, що оточують розташування точки;

– ознаки, які можна зіставити на основі їх орієнтації та локального вигляду (профілі країв), називаються краями, і вони також можуть бути хорошими індикаторами меж об’єкта та подій оклюзії в послідовності зображень.

Можливою ознакою може виступати площа − це кількість пікселів, що присутні в області. Областью може виступати як зображення в цілому, так і якась його частина. Розрахунок площі відбувається за формулою:

$$S = \sum_{i=1}^{Y} \sum_{j=1}^{X} 1, \tag{1.1}$$

де X − ширина області на зображенні;

Y − висота області на зображенні.

Вагові центри, або центроїди − інший тип ознак, що представлений середніми координатами усіх пікселів, що присутні у області [3]. Середнє значення по кожній із координат рахується окремо для висоти (1.2) і для ширини (1.3):

$$\bar{y} = \sum_{j=1}^{Y} y_j, \tag{1.2}$$

$$\bar{x} = \sum_{i=1}^{X} x_i, \tag{1.3}$$

де x_i − координати ширини i-тої точки області XY;

y_j − координати ширини j-тої точки області XY.

Периметр області представляє із себе множину граничних пікселів області. Це визначення передбачає відсутність внутрішніх отворів у областях. Піксель області є граничним, якщо один з його сусідів не належить цій області. Для чотиризв’язкових областей граничні пікселі утворюють восьмизв’язкові області, і, навпаки, у восьмизв’язкових областей граничні пікселі чотиризв’язкові. Багато пікселів периметра P_8 для 4- і P_4 для 8- зв’язкових областей визначаються наступним чином:

$$P_4 = \{(x,y) \in XY | N_8(x,y) - XY \neq \varnothing\}, \quad (1.4)$$

$$P_8 = \{(x,y) \in XY | N_4(x,y) - XY \neq \varnothing\}. \quad (1.5)$$

Довжина периметра $|P|$ обчислюється по впорядкованому ланцюжку сусідніх пікселів периметра $P = <(x_0, y_0), (x_1, y_1), \ldots (x_{k-1}, y_{k-1})>$ наступного виразу:

$$|P| = |\{k \mid (x_{k+1}, y_{k+1}) \in N_4(x_k, y_k)\}| + \\ + \sqrt{2}|\{k \mid (x_{k+1}, y_{k+1}) \in N_8(x_k, y_k\} - N_4(x_k, y_k\}|, \quad (1.6)$$

де $k \in \{1, \ldots K\}$, K – довжина впорядкованого ряду пікселів.

Відповідно до цього визначення довжини периметра два сусідніх чотиризв'язкових пікселів збільшують довжину периметра на одиницю, а восьмизв'язкових – на величину корінь з 2.

З урахуванням визначень площі S та периметра P можна ввести властивість округлості (circularity) області як квадрата довжини периметра, поділеного на площу області

$$C_1 = \frac{|P|^2}{S}. \quad (1.7)$$

Зважаючи на те, що значення округлості має різні значення для однієї і тієї ж області в залежності від того, представлена вона у вигляді 4- або 8-зв'язковий, а також від того, що вона набуває мінімального значення для ромбів і восьмикутників, а не для кола, що характерно для безперервних плоских фігур, було запропоновано інше визначення округлості:

$$C_2 = \frac{\mu_{XY}}{\sigma_{XY}}, \quad (1.8)$$

де μ_{XY} − середнє значення відстані від граничних пікселів області до центроїда;

σ_{XY} − середньоквадратичне відхилення відстаней граничних пікселів області до центроїда.

Використані значення відхилення та середнього розраховуються за (1.9) та (1.10). Тут множина пікселів (x_k, y_k), $k = k \in \{1, \dots K\}$ належить периметру P аналізованої області. Обчислення округлості другим способом монотонно збільшується і має поведінку, схожу на ту, що є у цифрових та безперервних образів фігур.

$$\mu_{XY} = \frac{1}{K}\sum_{k=1}^{K} ||(x_k, y_k) - (\bar{x}, \bar{y})||, \quad (1.9)$$

$$\sigma_{XY} = (\frac{1}{K}\sum_{k=1}^{K}[||(x_k, y_k) - (\bar{x}, \bar{y})|| - \mu_{XY}]^2)^{1/2}. \quad (1.10)$$

Моменти кольору − це характеристики, що надають інформацію про розподіл кольору на певній області зображення [4]. До таких характеристик відносяться середнє, квадратичне відхилення та дисперсія. Ще одним показником є асиметрія, вона вказує яку форму набуває розподіл кольору та має формулу:

$$s_I = (\frac{1}{N}\sum_{n=1}^{N}(c_n - \mu_I]^3)^{1/3}, \quad (1.11)$$

де c − вектор значень кольору за усіма каналами;

μ_I − середнє значення кольору в області;

N − кількість пікселів у області.

Найчастіше за все кожен піксель може бути представлений у форматі RGB, що має три канали: червоний, зелений та синій, або одному каналі яркості, тоді ці характеристики будуть описувати розподіл яскравості. Але існують і інші формати, наприклад: HSV [5].

Іншою ознакою властивостей кольору може бути його гістограма, що описує розподіл кількості пікселів в області. Кількість елементів в гістограмі залежить від кількості біт у кожному її пікселі. Наприклад, якщо розглядається n біт, значення будуть варіюватись від 0 до 2^{n-1} і гістограма буде мати 2^n елементів.

Для великих наборів даних, можна скористатися гістограмою кольорів для розрахунку моментів кольору. Для цього замість значень пікселів просто використовуються значення із гісторами.

Усі ознаки вказані вище, дозволяють описувати деяку область на зображенні за рахунок опрацювання пікселів що належать цій зоні. Як було зазначено раніше, у якості зони можуть виступати такі об’єкти як ключові точки.

1.2 Ознаки ключових точок та їх дескрипторів

Ключові точки − це точки на зображенні у яких є виражена текстура. Це точки у яких межі об’єкту дуже різко змінюються або точки між двома краями сегменту зображення. Приклад того яким чином ключові точки можуть бути представлені на зображенні продемонстрований на рисунку 1.1.

Для того щоб знаходити ці точки існують спеціальні алгоритми, що приймають на вхід зображення та видають дескриптори ознак. Дескриптори зберігають важливу інформацію у вигляді набору чисел, це так би мовити числовий відбиток, що дозволяє відділяти одні ознаки від інших. У ідеалі ця інформація буде незмінна відносно перетворень зображення, а отже можна віднайти ці ознаки, навіть якщо зображення було піддане змінам якимось чином.

Рисунок 1.1 − Зображення ключових точок на зображенні

Дескриптори розраховуються на базі знайдених на зображенні ключових точок і можуть поділені на два класи:

– локальні дескриптори, що є представлення локального оточення точки. Ці дескриптори намагаються збирати інформацію про форму та зовнішні ознаки тільки у локальному оточенні навколо точки тому дуже підходять для представлення його в термінах відповідності;

– глобальні дескриптори навпаки описують усе зображення одразу. Вони загалом не дуже стійки, адже зміну у частині зображення можуть призвести до невдачі, так як це вплине на весь результуючий дескриптор.

Для пошуку ключових точок та виділення дескрипторів існують різні алгоритми. Один з таких алгоритмів – SIFT (Scale Invariant Feature Transform). Алгоритм SIFT включає до себе чотири основних етапи [6]:

– відбір вершин у маштабованому просторі;
– локалізацію ключових точок;
– призначення орієнтації у просторі;
– виділення дескриптору ключової точки.

Спочатку, потенційні зони інтересу визначаються під час сканування простору і масштабу. Цей досягається за рахунок пошуку у максимумі і мінімумі різниці функції Гауса, що застосовується у маштабованому просторі. Таким чином досягається інваріантність до масштабування, зміни положення у просторі, зміни освітлення, а також часткова стійкість до Афінних перетворень [7]. На другому етапі, ключові точки що розглядаються, локалізуються до піксельної точності та виключаються у випадку нестабільності. Потім визнаються орієнтації у просторі, що видаються найбільш переважними. Отримане направлення, масштаб та локалізації дають можливість сформувати канонічне відображення ключової точки, що є стійким до перетворень. У самому кінці формуються локальні дескриптори для кожної з ключових точок. На цій стадії формується представлення, що засноване на групі пікселів по сусідству до точки. Перед виділенням ключової точки, зона інтересу центрується навколо неї, повертається на основі, визначеної на другому етапі, орієнтації у просторі та масштабується до підходящого розміру.

Інший алгоритм для пошуку ключових точок − SURF (Speeded-Up Robust Features). Він був частично натхненний попереднім алгоритмом, але є у декілька разів швидший за свого попередника [8]. Ключові точки що отримуються із використанням алгоритму SIFT дуже довго показували значуще гарні результати у великій кількості систем, що використовують виділені ознаки зображень, включаючи розпізнавання об’єктів, візуальне відображення, зшивання зображень та інші. Але, цей алгоритм вимагає великих обчислювальних затрат, особливо для систем, що працюють, у реальному часі, або для не дуже продуктивних апаратних ресурсів, як смартфони. Це призвело до пошуків алгоритму із потребою у менших обчислювальних витратах, одним з яких і став SURF [9].

SURF базується на трьох етапах:

– виявлення Гессіана, що проходить фільтром по оригінальному зображенню та визначає вісімнадцять різних детермінантів матриці Гессіана, маскимуми якої і будуть ключовими точками;

– визначення орієнтації у просторі;

– порівняння ключових точок для їх виключення.

Для точки $p = (x, y)$ на зображенні матриця Гессіана H, у масштабі σ визначається як:

$$H(p,\sigma) = \begin{bmatrix} L_{xx}(p,\sigma) & L_{yx}(p,\sigma) \\ L_{xy}(p,\sigma) & L_{yy}(p,\sigma) \end{bmatrix}, \quad (1.12)$$

де $L_{xx}(p, \sigma)$ − згортка Гауса другого порядку у точці p.

Гаусіани є оптимальними для аналізу масштабованого простору, але на практиці вони повинні бути дискретизовані та відсічені. Це призводить до втрат при поворотах. Це стосується всіх детекторів на базі Гессіана [10].

Це один алгоритмом для виділення ознак є алгоритм ORB − Oriented FAST and Rotated BRIEF. Цей алгоритм складається із двох основних етапів: виділення ключових точок та формування дескрипторів ключових точок [11]. Виділення ключових точок відбувається за рахунок алгоритму FAST − Features from Accelerated Segment Test, а формування дескрипторів виконується за допомогою алгоритму BRIEF − Binary Robust Independent Elementary Features. Обидві ці методики привабливі завдяки своїй гарній продуктивності та невисокій вартості обчислення.

Ідея алгоритму FAST базується на тому, що пікселі кутів значно відрізняються від сусідніх до них пікселів. Перед початком пошуку виконується масштабування зображення за рахунок білінійної інтерполяції.

Далі беруться деякі пікселі p, у якості центрів і 16 пікселів навколо них, що потрапляють на край кола із радіусом 3, як зображено на рисунку 1.2.

Отримані пікселі порівнюються за показниками яскравості: сусіди відносно центрів. Якщо сума різниць яскравості N послідовних сусідів відносно центра більша за модулем за деякий поріг T − центр розцінюється як потенційна ключова точка. Це обчислення виконується як для оригінального, так і для масштабованого зображення.

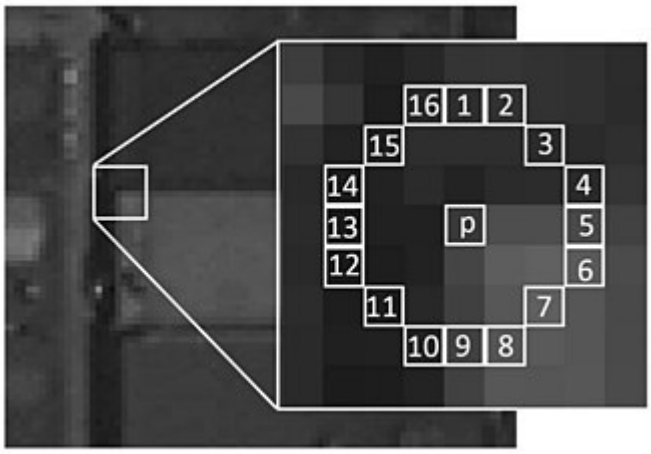

Рисунок 1.2 − Виділення центру та 16 точок навколо

Так як FAST використовує для обчислення кутів лише різницю у яскравості між пікселями, об’єм результатів може бути дуже великим і неточним. Саме тому ORB поліпшує алгоритм FAST, використовуючи відгу Харріса для сортування та фільтрації отриманих ключових точок із FAST.

Далі формуються масштабовані піраміди на кожному рівні яких зменшується дискретизація зображення. Це робиться для того щоб додати ознакам стійкість до перетворень.

До ключових точок додається орієнтація у просторі, для того щоб надати точкам стійкість до поворотів. Орієнтація отримується за рахунок використання інтенсивності центроїда. Для цього розраховуються моменти невеликого блоку навколо ключової точки за формулою [12]:

$$m_{pq} = \sum_{x,y \in XY} x^p y^q B, \tag{1.13}$$

де p, q = {0, 1};

x − координата ширини пікселя у блоці;

y − координата висоти пікселя у блоці;

B − значення яскравості відповідного пікселя.

Далі, із використанням отриманих показників моментів, для комбінацій p, q, визначаються центроїди:

$$C = (\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}). \quad (1.14)$$

Орієнтація ключової точки може бути отримана наступним чином:

$$\theta = arctan(\frac{m_{01}}{m_{10}}). \quad (1.15)$$

Після отримання ключових точок, зорієнтованих у просторі, алгоритм ORB використовує алгоритм BRIEF для знаходження дескрипторів цих точок. BRIEF надає вектор бінарних дескрипторів, значення елементів якого можуть приймати 0 або 1 [13]. Для того, щоб позбутися від можливих шумів спочатку використовується фільтрація Гауса. Далі відносно ключової точки беруться N випадкових пар пікселів по сусідству до неї. Для кожної із пар виконується порівняння їх яскравості. Якщо яскравість першої точки у парі переважає за другу, або їй дорівнює, до дескриптора буде записано значення, що дорівнює нулю, у протилежному випадку до дескриптора буде записано значення яке дорівнює одиниці.

1.3 Постановка задачі дослідження

У багатьох алгоритмах виділення ключових точок та формування їх дескрипторів присутні ти чі інши методи додаткової фільтрації отриманих ознак. Таким чином можна сказати про те, що процес редукції отриманих у ході виділення ключових точок є актуальним.

Об’єктом дослідження є методи зменшення розмірності систем ознак для представлення даних у задачі класифікації зображень.

Метою є розроблення методів, що дозволяють проводити редукцію системи структурних ознак на основі метричного критерію інформативності.

Для досягнення мети пропонується допустити, що у є деяка база еталонних класів $E = \{E_1, E_2, \dots E_n\}$ розмірністю n. Кожен клас E_k являє собою множину ознак розмірністю s, де кожен елемент множини – це деякий дескриптор бінарних значень, що описує ключову точку на зображені.

Кожна із ознак несе у собі деяку кількість інформативності відносно кожного з еталонних класів. Визначивши цю інформативність, можна буде на її основі знайти найбільш значущі для еталонного класу ознаки, і на основі них здійснювати розпізнавання.

Треба обрати метод, за допомогою якого можна редукувати множину ознак E_k, залишивши для кожної множини j ознак, де $j \ll s$. Реалізувати обраний метод у вигляді програмної моделі та використати для класифікації редуковані системи ознак еталонних зображень. Наприклад, можна скористатися деякою метрикою для визначення ступеню подібності значень ознак між собою за критерієм інформативності та виділити лише ті, що є визначними для кожного з еталонних класів.

Із використанням метрики Хаусдорфа та метрики Танімото треба буде оцінити подібність еталонних класів між собою у їх початковому становищі, а потім, після проведення редукції, знов скористатися метриками для отримання нових відстаней. Таким чином можна буде зрозуміти, як проведення редукції вплине на метричне відношення між множинами ознак.

2 МЕТОДИ РЕДУКЦІЇ СИСТЕМИ ОЗНАК

2.1 Поняття та призначення редукції для множини ознак

Редукція ознак − це процес зменшення об’єму простору, де представлені дані задля більш значущого або точного представлення [14]. У ідеалі, редуковане представлення даних повинно містити ознаки у такій кількості, скільки необхідно для того, щоб можна було віднайти характерні властивості даних, що були редуковані [15].

Редукція ознак є важливою у багатьох галузях, так як вона дозволяє позбутися такої проблеми як прокляття розмірності та інших небажаних властивостей роботи з даними у багатовимірних просторах. Зменшення кількості ознак дозволяє нам знизити необхідну кількість обчислювальні ресурсів та значно покращити швидкість обробки наших даних моделлю.

Також редукція ознак має ще одну перевагу: вона дозволяє позбутися від шумів та рудиментарних ознак, що можуть негативно вплинути на результати дослідження даних. Наприклад, дуже часто можна зіштовхнутися із таким перенавчання − це явище, коли навчена модель замість описує взаємозв’язок на основі шумів або похибок, і у результаті не може давати гарні результати з передбачення. До зайвих ознак можна віднести ті ознаки, що мають високу кореляцію з іншими ознаками, або і взагалі їх перекривають. Наявність таких ознак зовсім не обов’язкова у даних, тому і є рудиментарною. Процес відсіювання зайвих, менш значущих ознак або шумів лежить в основі самого процесу редукції ознак.

Ще одна перевага використання редукції ознак полягає у тому, що отримані у результаті дані стає легше візуалізувати і обробляти людині. Відобразити у просторі дані у двох вимірах значно легше ніж у трьох, що є доволі очевидним, не кажучи вже про дані, у яких присутні чотири і більше ознак. Факт того, що людині стає легше обробляти ці дані випливає сам собою. Звісно, сприймати дані відображені у площині із двома вимірами значно легше

ніж у площині із трьома, а сприймати дані у площинах із більшою розмірністю взагалі майже не можливо. Звісно не завжди вхідні дані візуалізуються для сприйняття, але це стається дуже часто. Інколи за рахунок редукції можна досягти ефекту, коли кількість ознак стає настільки малою, порівняно із початковою розмірністю, що людина і сама може досить легко аналітичним шляхом обробляти ті дані, що до цього задавалися ій складними. Навіть якщо такий ефект не може бути досягнутий, все одно сприймати дані у меншому кількості просторів стає легше.

Останнім, але не менш важливим фактором є те, що зменшення розмірності даних зменшує як часову так і просторову складність обробки даних. Це значить, що швидкість обробки і необхідний простір для зберігання даних і роботи з ними зменшується у процесі редукції ознак. Зрозуміло, що якщо позбуватися деяких ознак, пам’ять, що була необхідна для зберігання цієї частини даних, звільняється, а отже можена використати її для чогось ще, або навпаки можна вже мати пам’ять що занята чимось ще. Можна уявити ситуацію, коли необхідно виконати обробку великого об’єму даних, але так стається, що оперативної пам’яті недостатньо для одночасної обробки такої кількості даних. У такому випадку можна виконати редукцію даних, для того щоб звільнити пам’ять та працювати вже з даними меншого об’єму використовуючи ті ж апаратні можливості що і раніше. До такого способу оптимізації апартного простору прибігають досить часто, коли не має можливості вкласти кошти у придбання кращих інструментів обчислення. Інше поліпшення стосується швидкості обчислення. Із зменшення кількості ознак і відповідно кількості даних для обробки, випливає що час, необхідний для обробки отриманих даних, буде меншим за той час, який знадобився би для обробки початкового набору даних. Тут складається та ж ситуація, що і випадку із відсутністю достатньої кількості пам’яті у апаратних ресурсах − коли час, необхідний для виконання обробки даних, є незадовільним дуже часто вдаються до редукції ознак.

Резюмуючи, можна сказати, що у редукції ознак є такі переваги:

– зменшення часу необхідного для обчислення операцій над даними;

– зменшення простору необхідного для зберігання даних;

– покращення сприйняття даних для людини та полегшення процесу візуалізації даних;

– відсіювання ознак, що є шумами, похибками або мають високу кореляцію з іншими ознаками.

Задачу редукції ознак можна сформулювати наступним чином. Візьмемо набір даних, що представлений у вигляді матриці X, розмірністю n x F, що складається з n векторів x_i, де $i \in \{1, 2, ... , n\}$ і кожен з векторів представлений із F ознак. Для того щоб повноцінно представити характеристики даних, нам потрібно f ознак, назвимо f − внутрішньою розмірністю, де $f < F$, а часто навіть $f << F$. Внутрішня розмірність означає, що елементи набору даних X лежать біля або у многовиді із розмірністю f, що знаходиться у середині простору із розмірністю F. Техніки редукції ознак перетворюють набір даних X із розмірністю F на набір даних Y із розмірністю f [14]. Загалом, заздалегідь нам невідомі які ознаки формують внутрішню розмірність f. Визначити які саме ознаки потрібні можна лише у ході визначення властивостей набору даних та його ознак.

Існує досить велика кількість методів редукції ознак. Загалом існуючі методи можна поділити на дві групи. Перша група методів називається виділення ознак, а друга група методів − відбір ознак. Кожна із груп різниця за підходом, який дозволяє отримати новий набір ознак для набору даних. Кожен із підходів та методи що їх реалізують потребує більш детального огляду.

2.2 Методи виділення ознак

Виділення ознак − це процес здобуття набору нових ознак на основі оригінальних використовуючи функціональну трансформацію [16].

Припускаючи що у є набір ознак $F = \{F_1, F_2, \ldots F_n\}$, після виділення ознак на виході можна отримати новий набір ознак $f = \{f_1, f_2, \ldots f_m\}$, де $m < n$, а $f_i = Fun_i(F_1, F_2, \ldots F_n)$, і Fun_i − це та сама функція трансформації. Дуже важливо провести дослідження для того щоб обрати доцільні перетворення. Ціллю виділення ознак є пошук мінімально можливого набору нових ознак, користуючись деякими перетвореннями у відповідності до деяких мір продуктивності. Для того щоб обрати доцільні методи виділення ознак потрібно дослідити наступні питання [2].

Міра продуктивності. Тут потрібно оцінити що є найбільш підходящим для оцінки виділених ознак. Для задачі класифікації, дані мають маркери класів і точність передбачень можна використати для того, щоб визначити чи були виділені ознаки вдалими у даному випадку. Коли мова йде про кластеризацію, дані не мають жодних маркерів, тому не можна бути певним щодо якості отриманих ознак, а отже необхідно вдаватися до використання інших метрик, таких як дисперсія, між-кластерна подібність та інші.

Перетворення. Потрібно визначити яким чином, використовуючи початкові ознаки, можливо буде отримувати нові. Різні функції перетворення можна використати для виділення ознак. Загалом, методи трансформації можна поділити на дві категорії: лінійні та нелінійні.

Кількість нових ознак. Розглядаючи це питання, треба визначитись із тим, яку мінімальну кількість нових ознак потрібно виділити, щоб зберегти усі характеристики і властивості даних, що потрібні для дослідження, після перетворення цих самих даних.

Розглянемо деякі алгоритми, що формують групу цих методів.

Метод головних компонент є одним із найбільш популярних методів зменшення розмірності представлення даних. Цей метод застосовується у багатьох галузях, серед яких комп’ютерний зір, розпізнавання образів, стиснення даних та інші. Основною ідеєю методу головних компонент є зменшення розмірності набору даних, що складається з великої кількості взаємопов’язаних ознак, водночас концентруючи у новому наборі даних

настільки багато дисперсії початкової вибірки наскільки це можливо [17]. Це досягається за рахунок перетворення до нового набору ознак, головних компонент, які не корелюють між собою, і впорядковані таким чином, що перші за порядком зберігають у собі більшість дисперсії оригінального набору ознак.

Припустимо, що існує x − це вектор з n випадкових ознак, і що інтерес представляють дисперсії n випадкових ознак, а також структура коваріацій та кореляції між n окнаками. До тих пір поки n має маленьке значення, або структура дуже проста, часто, буде не дуже ефективно просто оцінити дисперсії n ознак та усі коваріації або кореляції. Альтернативним підходом буде відштовхуватись від декількох ($<< n$) диференційованих ознак, які зберігають найбільшу кількість інформації, що наданими цими дисперсіями і кореляціями або коваріаціями.

Те, яким чином головні компоненти представлені для множини вихідних даних відносно первинних ознак, продемонстровано на рисунку 2.1.

Не дивлячись на те, що метод головних компонент не ігнорує такі показники як коваріації та кореляції, він здебільшого концентрується саме на дисперсії. Дисперсія виступає, у даному випадку, і мірою оцінки алгоритму. Кількість ознак, які будуть отримані у ході перетворення визначається тим, яку кількість головних компонент потрібно взяти, щоб захопити бажану кількість дисперсії. PCA, входить до групи методів, що не потребують промарковані класами дані на вхід.

PCA − це дуже популярний та завжди зручний метод для редукції розмірності набору даних, але треба пам’ятати що його використання повністю позбавляє можливості інтерпретувати якимось чином ознаки після його використання.

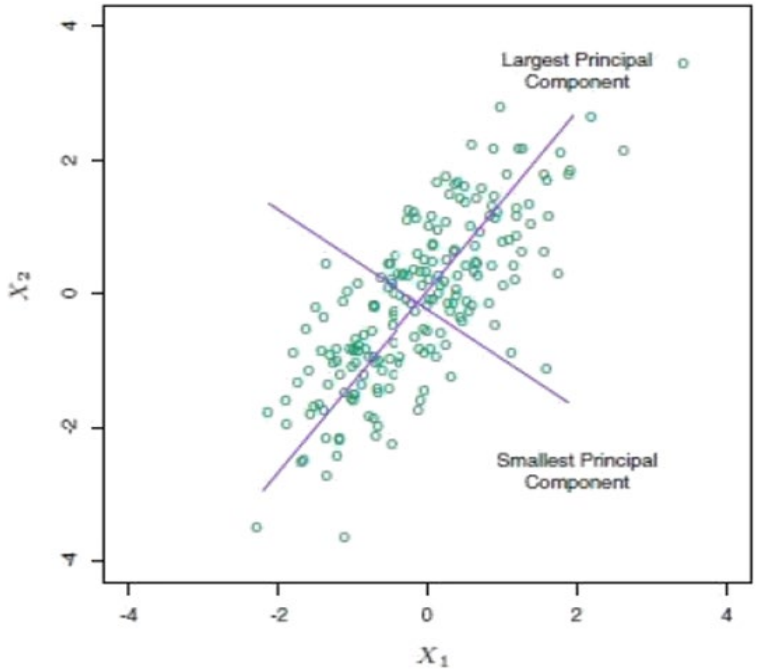

Рисунок 2.1 − Сформовані головні компоненти відносно первинних ознак

Лінійний дискримінантний аналіз дозволяє оцінювати відмінності між двома і більше групами об’єктів за декількома ознаками одночасно. Цей метод інтерпретує міжгрупові відмінності, базуючись на тому наскільки добре набір ознак, що використовується, може сформувати роздільну поверхню для об’єктів навчальної вибірки і які з цих ознак найбільш інформативні. Суть методу полягає у відтворенні додаткової осі, що проходить через набір точок, що представляють дані таким чином, що проекції на цю вісь забезпечують максимальну розподільність на два класи. Її положення визначається лінійною дискримінантною функцією з ваговими коефіцієнтами, що визначають внесок кожної із початкових ознак.

$$z(x) = w_1x_1 + w_2x_2 + \ldots + w_nx_n, \qquad (2.1)$$

де n − початкова кількість ознак;

$w_1, w_2 \ldots, w_n$ − вагові коефіцієнти ознак;

$x_1, x_2 \ldots, x_n$ − значення, що відповідають ознакам.

В основі дискримінантного аналізу лежить припущення про те, що описи об’єктів кожного класу є реалізацією багатовимірної випадкової величини, розподіленої за нормальним законом з коваріаційною матрицею C_k:

$$C_k = \frac{1}{n_k - 1}\sum_{i=1}^{n_k}(x_{ik} - \mu_k)^T(x_{ik} - \mu_k), \quad (2.2)$$

де μ − середнє очікуване.

Окрім цього, у ході лінійного дискримінантного аналізу робиться ще одне припущення, що коваріаційні матриці усіх класів дорівнюють одна одній. У такому випадку, вектор коефіцієнтів w_1, w_2 …, w_n можна визначити по формулі:

$$w = C^{-1}(\mu_1 - \mu_2). \quad (2.3)$$

Отримана у результаті вісь, співпадає з рівнянням прямої, що проходить через центроїди класів. Таким чином, у ході дискримінантного аналізу робиться припущення про нормальність розподілу даних у кожному класі, що на практиці виконується дуже рідко, а також про статистичну рівність класових матриць дисперсій та коваріацій.

Приклад отриманої у результаті алгоритму осі продемонстровано на рисунку 2.2.

Проблемами цього алгоритму є те, що він не дуже добре працює з даними номінальними даними, а також те, що його використання дуже складно інтерпретувати за евклідовим простором ознак.

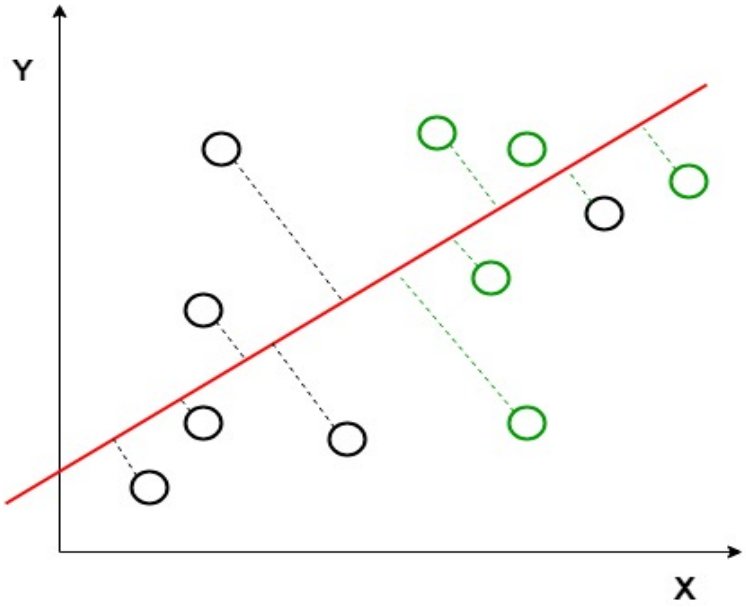

Рисунок 2.2 − Результат роботи лінійного дискримінантного аналізу

Ще одним способом для редукції ознак методом виділення ознак є використання штучних нейронних мереж прямого поширення. Прикладом такої мережі може бути одношаровий перцептрон. Ідея полягає у тому, щоби використовувати дані на прихованих шарах у якості нових виділених ознак. Точність передбачень використовується тут у якості оцінки ефективності, це також зумовлює необхідність у присутності маркерів класів у наборі даних. Кількість нових ознак формується на базі того, скільки буде нейронів присутньо у прихованому шарі.

На вхід до перцептрона подається вектор x розмірністю F, що трансформується у інший проміжний вектор h розмірністю f. Якщо $f < F$, мова йде про редукцію ознак. Для того щоб отримати вектор h, користуються формулою:

$$h = \sigma(Wx + b), \tag{2.4}$$

де σ − функцію активації;

W − матриця вагів відносно початкового набору ознак;

b − це вектор зсувів відносно тих самих ознак.

У ролі функції активації може виступати, наприклад, сигмоїда. Після розрахунку прихованого шару іде процес декодування, або розрахунку вихідних даних. Цей процес стосується безпосередньо класифікації, таким чином поєднуючи одночасно два процеси − редукцію ознак і класифікацію.

2.3 Методи відбору ознак

Відбір ознак − інший метод редукції ознак. Цей метод базується на тому, що із набору початкових ознак F обирається підмножина нових ознак f, де $f < F$, а ознаки оптимальним чином редуковані за певними критеріями [18]. Використання відбору ознак дуже розповсюджене для вирішення проблеми редукції ознак. Відповідно до того, чи промарковані вхідні дані алгоритми для відбору ознак можна поділити на алгоритми із вчителем або алгоритми без вчителя. Відбір ознак з вчителем ще можна поділити на методи фільтрації, методи обгортання та вбудовані методи − на основі того, яким чином вони взаємодіють безпосередньо із моделлю навчання [19].

Методи фільтрації відділяють процес виділення ознак від процесу навчання моделі класифікації, таким чином відділяючи похибку класифікації від похибки відбору ознак. Ці методи не взаємодіють із моделлю ніяким чином, а отже відбір ознак проходить на основі лише загальних ознак, таких як відстань, дисперсія, залежність, постійність та кореляція. Методи цього типу позбуваються найменш важливих ознак. Методи фільтрації є дуже ефективними відносно покращення швидкості обробки даних, а також показують гарні результати у вирішенні проблеми перенавчання.

Методи обгортання використовують точність передбачення визначеного раніше алгоритму навчання для того щоб визначити якість обраних ознак. Ці методи непомірно дорогі для використання на даних із великою кількістю ознак. Також недоліком для цих методів є ризик перенавчання за умови низької кількості спостережень.

Залишилися вбудовані методи − це методи які намагаються компенсувати недоліки методів обгортання та методів фільтрації. Спочатку, ці методи включають статистичні показники, як це роблять моделі фільтрації, для того щоб вибрати підмножину ознак із заданою кардинальністю. Потім, вони обирають ті підмножини ознак, що мають найвищі показники точності класифікації. Таким чином вбудовані моделі можуть досягати як порівнянної точності із методами обгортання, так і порівнянної ефективності із методами фільтрації. Вбудовані моделі виконують відбір ознак безпосередньо під час навчання. Іншими словами, використовуючи вбудовані моделі можна отримати навчену модель і відібрані ознаки одночасно.

Існує декілька методів відбору ознак, які поділяють алгоритми на групи, які описані тими підходами до відбору ознак що ці алгоритми використовують. Одні з них використовують вичерпний або повний підхід, що фокусується на мірі невідповідності і вичерпно оцінює підмножину ознак починаючи із множини, яка складається лише з однієї ознаки. Інший підхід називається евристичним, він не дає гарантовані результати, але дозволяє отримати перевагу у швидкості роботи алгоритму. Два останні підходи − це недетермінований підхід та підхід що базується на об’єктах. Розглянемо деякі з цих підходів на прикладі алгоритмів що їх використовують.

Алгоритм послідовного прямого відбору − це жадібний алгоритм, що був розроблений як напів оптимальне рішення для часто обчислювально нездійсненного вичерпного пошуку. Як можна зрозуміти, цей алгоритм використовує саме вичерпний підхід. Також алгоритм послідовного прямого відбору використовує метод обгортання для роботи з даними.

Цей алгоритм, якщо описувати коротко, видаляє або додає одну ознаку на певному кроці, покладаючись на оцінку ефективності класифікатора, до тих пір поки підмножина ознак не досягне бажаного розміру.

Візьмемо початковий набір ознак $F = \{F_1, F_2, \ldots F_n\}$. Алгоритм послідовного прямого відбору приймає на вхід весь набір цих ознак у якості вхідних даних. Набір бажаних ознак ініціалізується у якості пустої множини

$f = \emptyset$. На кожному кроці алгоритму додаться ознака f_+ до підмножини відібраних ознак. Для того щоб визначити яка саме із ознак додається до підмножини використовують формулу:

$$f_+ = arg\ max\ C(F\ +\ f), \tag{2.5}$$

де C − це функція оцінки.

Результати функції оцінки максимізуються для того щоб обрати найбільш оптимальну ознаку, а набір ознак $f \in F - f$. Ця операція повторюється до тих пір поки не буде отримано розмір підмножини ознак f бажаного розміру.

Критерієм оцінки може виступати індекс подібності або будь яка інша функція що надає інформацію про якість класифікації. У випадку якщо буде використана метрика можна мінімізовувати критерій.

Основний недолік, як і перевага, цього алгоритму використання його одночасно із процесом класифікації. Отже, не можливо побудувати модель незалежно від процесу класифікації.

2.4 Метод відбору ознак за критерієм інформативності

Пропонується розглянути деяку базу еталонів у вигляді множини E описів еталонних зображень розмірністю n. $E = \{E_1, E_2, \ldots E_n\}$ − це набір вхідних даних, тобто навчальна вибірка. Кожний еталонний опис E_i репрезентує окремий клас.

Опис окремого еталону $E_i = \{e_i(i)\}$ − це скінченна множина дескрипторів КТ у просторі B_n, де e_i (i) ∈ B_n, індекс $i \in \{1, 2, \ldots, s\}$, а $s = card\ E_i$ − це число дескрипторів у множині [20]. Кожний дескриптор $e_i(i)$ бази E має параметр i номеру класу, а загальне число ознак – дескрипторів у базовій множині E складає $card\ E = sN$ [21].

У просторі B_n дескрипторів вводиться відстань $p(z_1, z_2)$, для всіх $– z_1, z_2 \in B_n$, $B_n \times B_n \rightarrow R(B_n)$, $R(B_n)$ простір значень метрики, що встановлює метричне відношення відмінності між векторами із простору Bn. Метрика для визначення відстані між ознаками обирається відносно типу даних, якими представлена навчальна вибірка. Загалом це може бути як і Евклідова метрика для дійсних чисел або метрика Хемінга для категоріальних.

Для довільного представника $z \in E$ у системі класів як складового елемента $z \in E$ еталонного опису E_k введемо поняття інформативності $V(z, E)$ у складі бази E. Для розрахунку інформативності користуються формулою:

$$V(z, E) = p_m(z, \bar{E}_k) - p_m(z, E_k), \tag{2.6}$$

де $p_m(z, \bar{E}_k)$ – мінімальна відстань від z до елементу бази, що не належить класу E_k;

$p_m(z, E_k)$ − мінімальна відстань від z до найближчого елементу із класу E_k, до якого належить сам елемент z.

$p_m(z, \bar{E}_k)$ розраховується як:

$$p_m(z, \bar{E}_k) = min\, p(z, \bar{e}_v(i)), \tag{2.7}$$

де p – метрика оцінки відстані між дескрипторами;

$\bar{e}_i(i)$ – i-тий дескриптор множини $\bar{E}_k$.

$p_m(z, E_k)$, в свою чергу розраховується наступним чином:

$$p_m(z, E_k) = min\, p(z, e_v(i)), \tag{2.8}$$

де p – та сама метрика оцінки відстані між дескрипторами;

$e_i(i)$ – i-тий дескриптор множини до якої належить елемент z.

Треба зауважити, що відстані до самого себе, що дорівнюють нулю не враховуються − $p(z, z) = 0$. У випадку якщо впроваджуються нормовані

відстані $0 \le p \le 1$, оцінювані значення критерію інформативності знаходяться в інтервалі $-1 \le V \le 1$.

Використання формули інформативності для визначення індивідуальної інформативності V для елементу опису засновано на припущенні про те, що інформативність ознаки є тим вищою, чим краще вона розділяє екземпляри навчальної вибірки на класи. Виходячи з цього, чим далі від екземпляра класу знаходиться найближчий до нього екземпляр іншого класу, тим вищою є індивідуальна інформативність. У той же час, чим далі від екземпляру у просторі ознак знаходиться найближчий до нього елемент цього ж класу, тим нижчою є індивідуальна інформативність. Тобто впроваджується принцип: «ближче до своїх, подалі від інших».

Також треба зауважити, що критерій (2.6) продуктивно буде враховувати відмінності між елементами свого класу тільки у випадку аналізу множини ознак, що не є значуще метрично близькими між собою всередині класу, так як для таких ознак для другої складової (2.6) виконується $p_m(z, E_k) \approx 0$. . Задля зниження впливу цього фактору множини цього фактору множини E_k ознак еталонів на попередньому етапі треба додатково обробити з виключенням повторень та близьких елементів. Наприклад, попередньо можна скоротити описи Ei застосуванням логічної фільтрації з порогом δ_p: $p(z, e_i(i)) \le \delta_p$. З іншого боку, можна обчислити параметри ознак за виразом (2.6), а потім провести додатковий аналіз значень інформативності.

Таким чином, ознаки (2.6) з високими значеннями індивідуальної інформативності вважаються значущими й інформативними стосовно результативної класифікації, ознаки зі значеннями вважаються малозначущими, тобто є претендентами на виключення із системи ознак.

У випадку розпізнавання на множині класів (більше двох класів) з метою більш поглибленого аналізу можна у виразі (2.6) враховувати відстань не тільки до найближчого елементу з протилежним значенням класу, а, наприклад, як середню відстань до множини найближчих екземплярів кожного з можливих класів. В прикладному застосуванні таке усереднення іноді

зважують ваговими коефіцієнтами класів у конкретній вибірці [22]. Тоді відстань обчислюється з урахуванням зважування на частоту класів вибірки.

Якщо класи еталонів E_i задано параметрами центрів $c(E_i)$ дани, то оцінку інформативності елемента $z \in E$ у відповідності до (2.6) можна отримати за виразом:

$$V(z,E) = p_m(z,c(E_i)) - p_m(z,c(E_k)), \forall \mathrm{i} \neq \mathrm{k}. \qquad (2.9)$$

Цей вираз описує різницю між відстанями до центрів власного та найближчого серед решти класів, де $p_m(z, c(E_i)) = min\ p(z, c(E_i))$. Зважаючи на те, що центри $c(E_i)$ взагалі можуть не належати простору дескрипторів, тут потрібно застосовувати більш універсальний тип метрики.

Пропонується взяти описаний метод у якості основного для реалізації моделі. Тепер, коли були розглянуті деякі інші методи редукції ознак, можна зрозуміти які рішення пропонують різні методи і які можна знайти переваги саме для використання саме методу на основі метричного критерію інформативності.

2.5 Метричний критерій оцінювання подібності множин

Для того, щоб оцінити подібність між двома об’єктами, використовують метрики − функції, що розраховують віддаленість між двома точками у метричному просторі. Точками можуть виступати набори характеристик об’єкта. Наприклад, якщо потрібно оцінити подібність людей можна розмістити у метричному просторі точку, що відповідає показникам людини і розрахувати відстань до іншої точки, що належить тим же показникам іншої людини. Характеристиками можуть виступати будь які показники: ріст, вік, вага та будь який інший показник, що можна виміряти. Якщо провести паралелі із ознаками, можна зрозуміти, що у випадку коли робота ведеться з

якимись даними, характеристиками, що описують один із об’єктів вибірки – це і є її ознаки.

Для того, щоб називатися метрикою функція повинна задовольняти наступним умовам:

– відстань повинна бути більше нуля;

– відстань дорівнює нулю тільки тоді, коли відстань вимірюється до точки у просторі що займає ті ж самі координати, тобто об’єкти що її представляють є ідентичними;

– відстань від однієї точки a до точки b дорівнює відстані від точки b до точки a;

– нерівність трикутника. У математиці воно визначає, що сума довжини двох сторін повинна бути рівною або більшою за довжину останньої сторони [23]. У метричному просторі вона означає що сума відстаней від однієї точки до двох інших повинна бути більше або рівно за відстань між цими двома точками.

Останнє правило є найбільш складним для дотримання. За рахунок його дотримання можна бути певним, що проходячи відстань від точки *a* до точки *b* із використанням відстаней, отриманих метриками, неможливо отримати ніякої вигоди проходячи через якусь проміжну точку *c*. Правило нерівності трикутника дозволяє описувати усі відстані так, наче вони описують довжину найкоротшого шляху від однієї точки у просторі до іншої. Загалом усі ці правила дозволяють досить точно відділяти ті функції що можуть називатися метриками від інших функцій. У всякому разі, існує досить багато різних метрик, що мають відмінні від одна одної способи розрахунку.

Найбільш відома та використовувана метрика − евклідова метрика у *n*-мірному просторі, де точки представлені векторами, що складаються з *n* дійсних чисел. Вона розраховується як норма другого порядку та визначається як [24]:

$$d_E(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}. \quad (2.10)$$

Для отримання відстані зводиться у квадрат різниця між показниками у кожному просторі із n, сумуються отримані різниці та береться квадратний корінь із суми. Евклідова метрика є найбільш поширеної і досить інтуїтивно зрозумілою для роботи з нею.

Іншою, досить поширеною, мірою відстані є норма другого порядку, або Манхеттенською метрикою. Це відстань між двома точками у вигляді суми модулей різниць в кожному просторі. Свою назву ця метрика отримала посилаючись на те, які відстань треба пройти, щоб досягнути однієї точки міста з іншою, перетинаючи міські квартали Манхеттена. У вигляді формули ця метрика виглядає наступним чином:

$$d_{He}(x, y) = \sum_{i=1}^{n} |x_i - y_i|. \quad (2.11)$$

З (2.11) ведеться робота з дескрипторами, що представлені у якості бінарних векторів, Евклідова метрика не підходить для цього випадку. Тому потрібно звернутися до іншої метрики, що заточена саме під роботу з бінарними векторами − метрики Хемінга. Ця метрика визначається як число відповідних позицій у двух векторах, де значення не співпадають. Насправді, якщо повернутися до Манхеттенської відстані, можна побачити, що цю формулу можна використати для розрахунку і відстані Хемінга для бінарних векторів. Таким чином випливає, що метрика Хемінга є окремим випадком Манхетенської метрики.

Інтуїтивно зрозуміло, що використати наведені вище метрики для множин дескрипторів неможливо. Отже, необхідно якимось чином знайти спосіб використати метрику для визначення подібності набору дескрипторів відносно іншого набору. Тут на допомогу приходить інша метрика − метрика Хаусдорфа. Метрика Хаусдорфа визначається формулою [25]:

$$d_H(X, Y) = max\{sup_{x \in X} d(x, Y), sup_{y \in Y} d(y, X)\}, \quad (2.12)$$

де $d(x, Y)$ визначається як:

$$d(x, Y) = inf_{y \in Y}\, d(x, y), \tag{2.13}$$

де d − означає відстань між векторами x та y;

sup − супремум, тобто найменший елемент підмножини що більший або дорівнює усім елементам множини;

inf − означає інфімум, тобто найбільший елемент підмножини що менший або дорівнює усім елементам множини.

Так як у цьому випадку векторами x та y є бінарні дескриптори, можна скористатися метрикою Хемінга для отримання «внутрішньої» відстані між елементами описів. Як можна зрозуміти із визначення, метрика Хаусдорфа надає змогу знайти відстань між двома множинами, оперуючи відстанями між елементами цих множин, як продемонстровано на рисунку 2.3. Користуючись цією метрикою, можна оцінити подібність двох множин між собою .

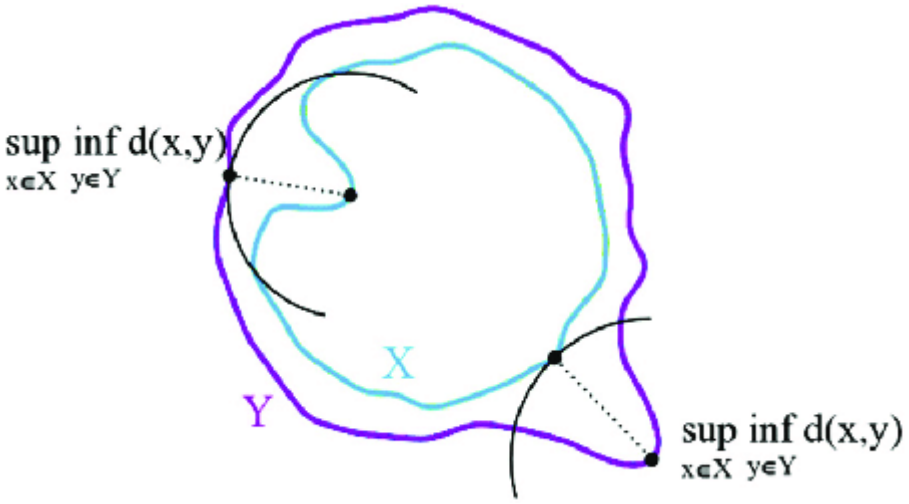

Рисунок 2.3 − Графічне представлення обчислення метрики Хаусдорфа

Метрика Хаусдорфа заточена під те, щоб оцінювати множини окремо беручи елементи з них, але існує і інша метрика, що дозволяє оперувати з множиною як з єдиним цілим − це метрика Танімото.

Дуже часто можна зустріти згадування метрики Жаккара, коли мова іде про метрику Танімото. Загалом, можна сказати, що це одне і те ж саме. Відстань між множинами за метрикою Танімото обчислюється наступним чином:

$$d_T(X,Y) = \frac{|X \cup Y| - |X \cap Y|}{|X \cup Y|}. \quad (2.14)$$

Відповідно до наведеної вище формули, знаходження відстані між двома множинами описується як відношення кардинальності різниці об’єднання та перетину множин до їх об’єднання. Користуючись цією метрикою не потрібно обчислювати проміжні відстані між елементами множин що є дуже зручним у випадку роботи з ними.

3 РЕЗУЛЬТАТИ ЕКСПЕРИМЕНТАЛЬНОГО ДОСЛІДЖЕННЯ МЕТОДУ РЕДУКЦІЇ ОЗНАК

3.1 Середовищє програмної реалізації

У якості основного інструмента програмної реалізації було вирішено вибрати мову програмування Python. Це мова програмування високого рівня з динамічною типізацією, виконання коду у якій виконується на базі інтерпретатора. У Python можно реалізовувати програмний код із використанням таких парадигм, як: об’єктно-орієнтоване програмування, функціональне, декларативне та інше [26]. Python розроблявся з ціллю бути дружелюбною та простою для розуміння мовою програмування, що буде давати розробникам більше часу для роздумів над тим як вирішити задачу з точки зору логіки, а не витрачати увесь час на роздуми над написанням програмного коду. Зараз підтримкою мови займаються безліч людей по всьому світу формуючи спільноту. Усі вони намагаються вносити свій вклад у розвиток Python, розвиваючи синтаксис та додаючи нові функціональні можливості. Також, багато людей із спільноти займаються розробкою бібліотек, які дозволяють вирішувати різні задачі у межах Python. Наприклад, існують бібліотеки для обробки даних, для роботи з мережею, для формування систем штучного інтелекту, для створення веб-серверів та безліч інших. Треба зауважити, що абсолютна більшість бібліотек доступна повністю безкоштовно. Усе це зробило Python однею із найбільш популярних мов програмування на сьогоднішній день.

Так, популярність мови робить її однією із перших варіантів для розгляду на позицію мови для реалізації, але які саме її переваги дозволять нам якісно виконати поставлену перед нами задачу − розглянемо далі.

Python − інтерпретована мова програмування. Доволі велика кількість мов програмування, таких як Java, C++, C# використовують так звані компілятори, що перед виконанням програми перетворюють написаний на мові програмування код на більш низькорівневий. Такий підхід має свої переваги, серед яких перевірка коду на помилки під час компіляції та пришвидшення роботи програми за рахунок виконання низькорівневих команд. На відміну від компілятора, інтерпретатор Python виконує код як він є, порядково. Тобто, він читає кожен рядок коду починаючи з початку файлу і виконує його один за одним. Це дозволяє реалізувати пряму роботу з інтерпретатором, де можливо поступово відправляти йому свої команди, зберігаючи стан між ними. Це дозволяє динамічно працювати з мовою, аналізуючи стан програми та даних з якими ведеться робота [27]. Мови що використовують коплітори не таке не здатні. Приклад прямої роботи з інтерпретатором можна побачити на рисунку 3.1.

Рисунок 3.1 − Виконання коду у інтерпретаторі IPython

Синтаксис Python заточений на те, щоб зробити читання та написання коду простішим. На відміну від інших мов програмування, Python використовує відступи для того щоб відділяти блоки коду лише за рахунок відступів. Наприклад для того, щоб помістити код під умовний оператор if,

треба додати усім строкам коду, однакові відступи, коли у інших мовах часто можна зустріти сегментацію коду за рахунок фігурних дужок. Також, Python не використовує круглі дужки для своїх операторів: якщо у деякі інші мови програмування зобов'язують помістити вираз для умовних операторів або операторів циклу під дужки, у Python достатньо просто написати ваш вираз після оператора.

Крім простих базових типів, у Python існують і базові типи, що представляють структури даних. Так, наприклад, у Python є такі типи даних як: list − список, що може зберігати у собі впорядковані набір даних різного типу, tuple − кортеж, що як список може зберігати у собі набір даних, але на відміну від списку має фіксовану довжину та не може її змінювати, dict − словник, цей тип можна порівняти з так званою хеш-мапою у інших мовах програмування, цей тип даних дозволяє зберігати попарні відповідності ключ-значення, та set і frozenset − множина і незмінна множина, відповідно, це тип даних дозволяє зберігати невпорядковані набори даних довільної і фіксованої довжини відповідно. Всі типи даних мають відповідний інтерфейс роботи з ними. Наприклад, множини мають методи для виконання операцій алгебри множин, серед яких операції об'єднання, перетину, симетричної різниці, тощо. Таким чином, Python дозволяє дуже легко працювати у межах теорії множин. Існування базових типів, що представляють структури даних робить Python дуже зручним для роботи з наборами даних для обробки. Коли інші мови інколи зобов'язують встановлювати бібліотеки, у Python все присутнє одразу і готове для використання. Таке наповнення мови дозволяє виділяти її у окремий клас мов програмування − мов ультрависокого рівня.

Не дивлячись на те, що у Python одразу присутні типи даних для роботи з структурами даних, інколи їх можливостей може бути недостатньо, або для вирішення деяких задач потрібно самостійно виконувати реалізацію великої кількості допоміжного інтерфейсу. Тут на допомогу приходить згадана раніше величезна спільнота мови програмування, що працює над створенням та підтримкою бібліотек для вирішення різноманітних задач, однією з яких є

робота з даними. Із величезної кількості бібліотек можна знайти ті, що будуть корисні для нас. Наприклад, найпопулярніша бібліотека для математичної обробки даних − NumPy. Ця бібліотека дозволяє створювати вектори та багатовимірні масиви та виконувати операціями над ними. За допомогою NumPy можна зручно застосовувати однотипні операції до всіх елементів вектору, наприклад перемножуючи вектор на одне число або навпаки застосовуючи ті ж операції попарно між двома векторами [28]. На рисунку 3.2 привидений приклад того, як робота з векторами NumPy виглядає на практиці.

```
In [1]: import numpy as np

In [2]: vector = np.array([1, 2, 3, 4])

In [3]: vector
Out[3]: array([1, 2, 3, 4])

In [4]: vector * 2
Out[4]: array([2, 4, 6, 8])

In [5]: vector * vector
Out[5]: array([ 1,  4,  9, 16])

In [6]: vector ** 2
Out[6]: array([ 1,  4,  9, 16], dtype=int32)
```

Рисунок 3.2 − Приклад роботи з векторами за допомогою бібліотеки NumPy

Окрім цього, об’єкти масивів бібліотеки мають інтерфейс, що дозволяє обчислювати статистичні показники масивів, такі як: середнє, дисперсія, середньоквадратичне відхилення, медіана та багато інших корисних речей. Також у самої бібліотеки присутні функції для розрахунку норми, функція побудови нормального розподілу, тощо. Усе це робить NumPy привабливим для того щоб використовувати його для обробки даних.

3.2 Огляд вхідних даних

На вхід приходять множини бінарних дескрипторів, кожен з яких представляє ключову точку зображення. Для прикладу можна взяти дві множини дескрипторів і розглянути перші десять з кожної. Приклад дескрипторів можна побачити нижче на рисунку 3.3 та рисунку 3.4. Усі дескриптори мають однакову довжину та містять по тридцять два бінарних значення. Саме з такими множинами і буде вестись робота надалі. Кожна з множин містить у собі сто дескрипторів ключових точок [29-34].

0	0	0	1	0	0	1	1	1	1	1	1	0	1	0	0	1	0	0	1	1	0	1	0	0	0	1	0	0	0	0	0
0	0	1	1	1	1	1	1	0	0	0	1	1	0	1	0	1	0	1	0	0	1	0	0	1	0	1	0	1	0	0	1
0	0	1	1	1	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	1	1	1	0	1
0	0	1	1	0	1	0	1	1	0	0	0	1	1	0	0	1	0	0	0	0	0	1	1	0	0	1	0	1	0	0	1
1	1	0	0	1	0	0	0	0	0	0	1	0	1	1	0	1	1	0	0	0	1	1	1	0	1	1	0	0	1	1	1
1	0	1	1	0	0	1	0	1	0	1	0	1	1	1	0	1	0	1	0	1	0	0	0	0	0	0	1	0	1	1	0
1	1	0	0	1	1	0	0	1	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1
1	0	0	1	0	1	0	1	0	1	0	1	1	1	0	1	0	1	1	0	1	0	1	0	0	0	1	0	1	0	1	0
0	0	1	0	1	1	0	0	1	1	0	0	0	0	0	1	1	0	0	1	1	0	1	1	0	1	0	0	0	1	1	0
0	0	1	1	0	0	0	0	1	0	1	0	1	1	1	1	1	0	0	0	0	1	0	1	0	0	0	1	1	1	0	0

Рисунок 3.3 − Значення перших десяти дескрипторів першої множини

0	1	0	1	1	0	1	0	1	0	1	0	0	1	1	1	1	1	0	1	1	1	1	0	1	1	0	1	1	1	1	1
1	0	1	1	1	0	1	0	0	0	1	1	0	1	0	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	0
0	0	0	1	0	1	1	1	1	1	0	1	1	0	1	0	1	0	0	0	0	1	0	0	1	1	0	0	1	0	0	1
0	1	1	1	0	0	0	1	0	1	0	1	0	1	0	1	1	1	1	0	1	1	1	1	1	0	1	0	1	1	1	1
0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	0	1	1	1	0	1	0	1	0	1
0	1	0	1	0	1	1	0	0	0	0	0	0	1	1	1	0	0	0	0	0	1	0	1	1	1	0	1	1	0	1	1
0	1	1	1	1	1	0	1	1	1	1	0	1	0	0	1	1	0	0	1	0	0	0	0	0	1	1	0	0	1	0	0
0	1	1	1	1	0	0	0	1	1	0	1	0	0	1	1	1	0	1	0	1	1	1	1	1	0	1	1	1	0	1	0
0	1	0	0	0	1	0	1	1	0	1	0	1	1	0	1	1	1	0	0	1	1	1	1	0	1	1	0	0	1	0	0
0	0	1	1	1	1	1	1	0	0	1	1	1	1	1	1	0	1	1	0	0	1	0	0	1	1	0	1	1	1	0	1

Рисунок 3.4 − Значення перших десяти дескрипторів другої множини

Так як представлення дескрипторів мало що може сказати у такому вигляді, пропонується проаналізувати кожну із множин та отримати статистичні показники, що нададуть більше інформації про те, що із себе представляє кожна з множин. Для того щоб оцінити різницю між отриманими на вхід множинами дескрипторів, можна оцінимо наскільки часто у них представлені значення 1 та 0. Для цього береться середня кількість одиниць у дескрипторі на множину. Для бінарних дескрипторів можна представити у вигляді формули:

$$p(Z) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} x_{ij}}{n * m}, \tag{3.1}$$

де n − довжина дескрипторів;

m − потужність множини ознак.

Підрахунок цієї характеристики програмно реалізується наступним чином, приведеним на лістингу 3.1.

Лістинг 3.1 Реалізація обчислення характеристики вірогідності появи 1:

```
def estimate_binary_probability(keypoints: np.ndarray):
_, inner_dimension_size = keypoints.shape
one_counts = np.array([
        np.count_nonzero(descriptor)
        for descriptor in keypoints
])
return np.mean(one_counts) / inner_dimension_size
```

Створюється функція, що приймає на вхід множину дескрипторів ключових точок у вигляді масиву numpy. Для безпосереднього підрахунку обчислюється кількість одиниць для кожного з дескрипторів, береться середнє значення і ділиться на розмір дескрипторів у множині.

Для того щоб краще зрозуміти, що із себе представляють множини, тепер можна скористатися розробленою функцією. На рисунку 3.5 зображено використання функції для обліку характеристики для обох множин дескрипторів ключових точок.

```
In [10]: keypoints_report(vset1) # Звіт за першою множиною десткрипторів
Властивості множини дестрипторів:
Очікувана середня вірогідність появи 1: 0.505625

In [11]: keypoints_report(vset2) # Звіт за другою множиною десткрипторів
Властивості множини дестрипторів:
Очікувана середня вірогідність появи 1: 0.569375
```

Рисунок 3.5 − Характеристики першої та другої множини дескрипторів

Як можна побачити на рисунку, вірогідність появи одинці для першої множини становить 0,505625, у той час як для другої − 0,569375. Це надає розуміння про те, що за наповненням з статистичної точки зору вони дуже схожі. Отже, можна зробити попередні висновки про те, що обидві із множин можуть бути подібні одна одній, але на даному етапі це лише припущення, для того щоб робити якісь заключення треба скористатися іншими способами оцінки подібності.

3.3 Програмна реалізація метрик

Тепер, коли відомо які метрики можна використовувати для оцінки подібності множин, потрібно реалізувати їх у вигляді програмного коду, щоб скористатися ними для порівняння множин, отриманих на вході.

Для подальшої реалізації метрик, пропонується реалізувати функцію, що дасть змогу перевірити коректність роботи отриманих метрик. У цій функції, буде виконуватися перевірка на дотримання чотирьом умовам, що дозволяють

метриці називатися метрикою. Створюється наступна функцію Python, зображена на лістингу 3.2.

Лістинг 3.2 Реалізація валідації метрики:

```
def validate_metric(metric, a, b, c):
self_distance_zero = not bool(metric(a, a))
negative_inequality = metric(a, b) >= 0
inverse_equality = metric(a, b) == metric(b, a)
triangle_inequality = metric(a, b) + metric(a, c) >= metric(b, c)

print((
    f'Відстать до самого себе дорінює 0: {self_distance_zero}\n'
    f'Відстать білше або дорінює 0: {negative_inequality}\n'
    f'Відстать від a до b дорінює відстані від b до a: {inverse_equality}\n'
    f'Нерісність трикутника виконується: {triangle_inequality}\n'
))
```

Наведена вище функція перевіряє усі правила, яких повинна дотримуватися метрика. На вхід, першим параметром, вона приймає функцію що відповідає метриці, а далі три об'єкта, на яких і буде перевірятися відповідність заданим умовам. Спочатку у функціїї обчислюється відстань об'єкта до самого себе, яка перетворюється у булеве значення та виконується операція заперечення − у результаті повинно бути отримане значення True, якщо відстань дорівнює нулю. Потім перевіряється, що відстань між двома поданими на вхід об'єктами більше або дорівнює нулю. Далі виконується перевірка, що відстань між об'єктом *a* та *b* дорівнює відстані від *b* до *a*, і, у кінці, перевіряється нерівність трикутника. Результат виконання обчислень виводиться у вигляді звіту до консолі.

Тепер, коли реалізовний механізм перевірки коректності роботи метрики, можна переходити до реалізації метрик. Для початку, пропонується реалізувати метрику Хемінга. Вона буде використовуватися для розрахунку відстані між елементами множин у метриці Хаусдорфа. У вигляді функції Python метрика реалізується наступним наступним чином, наведеним на лістингу 3.3.

Лістинг 3.3 Реалізація обчислення метрики Хемінга:

```
def hamming_distance(
descriptor1: np.ndarray,
descriptor2: np.ndarray
):
return np.count_nonzero(descriptor1 != descriptor2)
```

Для обчислення відстані, на вхід до функції подаються два дескриптори у вигляді масивів із бібліотеки NumPy. Потім, користуючись можливостями бібліотеки, лише за допомогою одного оператора нерівності отримуються булеві значення, що свідчать про те відмінні від один одного відповідні елементи дескрипторів чи ні. Далі, користуючись функцією count_nonzero, обчислюється кількість елементів відмінних від нуля − результатом і є отримана відстань між двома дескрипторами. Така форма обчислення відрізняється від формули Манхеттенської відстані, але якщо поглянути на ті операції, що відбуваються у функції, можна зрозуміти, що у випадку з бінарними векторами отриманий результат буде ідентичний.

Для того, щоб перевірити коректність роботи метрики, можна скористатися функцією валідації, що була реалізована раніше. На вхід до функції валідації будуть подані три дескриптора із першої множини. Звіт за результатом функції наведений нижче, на рисунку 3.6.

```
In [4]: validate_metric(hamming_distance, vset1[0], vset1[1], vset1[2])
Відстать до самого себе дорінює 0: True
Відстать більше або дорінює 0: True
Відстать від a до b дорінює відстані від b до a: True
Нерісність трикутника виконується: True
```

Рисунок 3.6 – Звіт за валідацією метрики Хемінга

Як можна побачити на рисунку, усі умови яким повинна відповідати метрика виконуються. Отже, можна бути певним, що реалізація метрики Хемінга працює задовільно. Далі, пропонується реалізувати саме метрику Хаусдорфа, що, як було зазначено раніше, буде використовувати тільки но реалізовану метрику Хемінга.

Для реалізації було вирішено створити дві функції Python: першу − яка буде розраховувати супремум однієї множини відносно іншої, другу − що буде шукати максимум із супремумів відносно обох множин. Отже, перша функція має наступний вигляд, наведений на лістингу 3.4.

Лістинг 3.4 Реалізація розрахунку супремума відстаней:

```
def distance_supremum(keypoints1, keypoints2, *, distance):
return max(
min(distance(xi, xj) for xj in keypoints2)
for xi in keypoints1
)
```

Для розрахунку супремума на вхід подаються дві множини дескрипторів. У першу чергу виконується пошук мінімуму відстаней від кожного з елементів першої множини до елементів другої. Далі знаходиться максимум із отриманих відстаней. Метрика, для розрахунку відстаней, також подається на вхід до функції. Далі можна реалізувати наступну функцію, що визначає відстань, таким чином як на лістингу 3.5.

Лістинг 3.5 Реалізація обчислення метрики Хаусдорфа:

```
def hausdorff_distance(
keypoints1, keypoints2,
distance=euclidean_dist
):
_distance_supremum = partial(
distance_supremum,
distance=distance
)
return max(
_distance_supremum(keypoints1, keypoints2),
_distance_supremum(keypoints2, keypoints1)
)
```

Друга функція також приймає на вхід метрику для розрахунку відстані для супрему і дві множини дескрипторів. Спочатку запам’ятовується метрика для функції супрему. Потім розраховуються супремуми для першої множини відносно другої і супремум другої відносно першої. Отримавши два супремума, обирається максимум із них, таким чином і отримується відстань між двома множинами.

Для того, щоб провести валідацію, на вхід до відповідної функції подаються дві множини що розглядаються, а також необхідно взяти деяку третю множину, для того, щоб перевірити правило трикутника. Як можна побачити на рисунку 3.7, усі правила виконуються, а отже і метрика Хаусдорфа працює коректно.

```
In [9]: validate_metric(hausdorff_distance, vset1, vset2, vset3)
Відстать до самого себе дорінює 0: True
Відстать більше або дорінює 0: True
Відстать від a до b дорінює відстані від b до a: True
Нерісність трикутника виконується: True
```

Рисунок 3.7 – Звіт за валідацією метрики Хаусдорфа

Врешті решт можна перейти до реалізації метрики Танімото. Як вже було згадано раніше, для роботи з множинами у Python присутній досить зручний інтерфейс, отже, реалізується функцію розрахунку відстані Танімото можливостями Python, перетворивши масиви NumPy у множини Python. Реалізація приведена на лістингу 3.6.

Лістинг 3.6 Реалізація обчислення метрики Танімото:

```
def tanimoto_distance(keypoints1, keypoints2):
keypoints1 = ensure_2d_set(keypoints1)
keypoints2 = ensure_2d_set(keypoints2)
return len(set1 ^ set2) / len(set1 | set2)
```

Отже, у функції розрахунку відстані Танімото спочатку буде виконана трансформація набору дескрипторів у множин Python, що зберігають всередині кортежі. Отримавши дані у вигляді множин та користуючись інтерфейсом цих об’єктів, розраховується кардинальність симетричної різниці між цими множинами та кардинальність об’єднання двох множин і перше ділиться на друге. Тепер треба виконати валідацію останньої метрики, для цього знову на вхід подаються три множини. Звіт за валідацією метрики Танімото можна побачити на на рисунку 3.8.

```
In [11]: validate_metric(tanimoto_distance, vset1, vset2, vset3)
Відстать до самого себе дорінює 0: True
Відстать більше або дорінює 0: True
Відстать від a до b дорінює відстані від b до a: True
Нерісність трикутника виконується: True
```

Рисунок 3.8 – Звіт за валідацією метрики Танімото

Отже, відповідно до звіту, можна бути певним, що метрика Танімото також працює коректно. Таким чином усі реалізовані метрики можна використовувати для оцінки подібності наших множин.

Тепер, коли для використання доступні усі реалізовані метрики, в коректності роботи яких можна бути впевненим, можливо оцінити подібність множин між собою. Для цього необхідно подати на вхід до цих метрик множини що розглядаються. Результати обчислень наведені на рисунку 3.9.

```
In [15]: hausdorff_distance(vset1, vset2, hamming_distance)
Out[15]: 12

In [16]: tanimoto_distance(vset1, vset2)
Out[16]: 0.8571428571428571

In [17]: hausdorff_distance(vset2, vset1, hamming_distance)
Out[17]: 12

In [18]: tanimoto_distance(vset2, vset1)
Out[18]: 0.8571428571428571
```

Рисунок 3.9 – Оцінка подібності двох множин дескрипторів за метриками Хаусдорфа і Танімото

Відповідно до результатів виконання, за метрикою Хаусдорфа був отриманий результат 12, що означає, що відстань до крайніх дескрипторів між множинами приблизно у дванадцять біт. За метрикою Танімото результат оцінки дорівнює 0,8571428571428571, це значить, що множини подібні між собою приблизно на 86 відсотків [35-39]. Також зайвий раз можна

пересвідчитися у тому, що виконується умова симетричності, за рахунок повторного використання метрик, подавши на вхід множини у зворотньому порядку.

Маючи ці оцінки, можна зрозуміти наскільки множини подібні одна одній. Потім, треба буде повернутися до цих метрик, щоб виміряти подібність множин після процесу редукції ознак. Це дасть змогу зрозуміти, яким редукція вплинула на вхідні дані.

3.4 Програмна модель редукції

Тепер можливо перейти до реалізації моделі редукції ознак, представлених у вигляді множин. Модель буде оформлена у вигляді класу Python, що буде приймати на вхід для ініціалізації метрику для обчислення показників інформативності ознак та кількість ознак *k*, яку треба залишити після відбору. Об’єкти класу будуть зберігати у собі показники інформативності для кожного екземпляру, які згодом можна буде використати для відбору ознак. Отже, створюється клас з конструктором, який ініціалізує початковий стан моделі.

Тепер необхідно реалізувати алгоритм підрахунку інформативності ознак між екземплярами. Для цього використовується формула (2.6), що була розглянута раніше. Спочатку потрібно реалізувати пошук найменшої відстані від елементу бази до елементів класу. Щось схоже, уже було зроблено, коли виконувалась реалізація метрику Хаусдорфа. Для пошуку створюється функція, що буде приймати на вхід вектор, множину ознак іншого класу та метрику яку потрібно використовувати для обчислення відстаней між дескрипторами. Для пошуку мінімуму все так же використовується вбудована функцією min. Результатом є наступна функція, приведена на лістингу 3.7.

Лістинг 3.7 Реалізація знаходження мінімуму відстаней:

```
def find_closest(target, options, distance):
return min(distance(target, option) for option in options)
```

Ця функція буде використана для пошуку мінімальної відстані від дескриптора оцінюваної множини до дескрипторів іншої. Для пошуку мінімальної відстані до елементів того ж класу створюється іншу функцію, яка дозволить позбутися відстаней до самого себе та подібних елементів, за рахунок фільтрації нульових відстаней. Для початку реалізується функція фільтрації, яка просто буде виключати нулі, а потім буде використана у функції подібній тій що була розроблена раніше, перед пошуком мінімальної відстані. У результаті отримується функції, наведені на лістингу 3.8.

Лістинг 3.8 Реалізація обчислення мінімуму відстаней за виключенням нульових:

```
def exclude_zeros(iterable: Iterable[numeric]) -> Generator:
    return (item for item in iterable if item > 0)

def find_closest_nonzero(target, options, distance):
    return min(exclude_zeros(
        distance(target, option)
        for option in options
    ))
```

Де функція exclude_zeros дозволяє фільтрувати нульові відстані, а find_closest_nonzero використовує її для пошуку мінімальної відстані у множини класу дескрипторів.

Тепер, коли є реалізовані функції пошуку мінімальної відстані, можна скористатися ними для обчислення оцінок інформативності у реалізації відповідного методу нашої моделі. У цьому методі просто обчислюється мінімальна відстань $p_m(z, \bar{E}_k) = \min p(z, e_v(i))$ до елементу іншого класу та

мінімальна відстань $p_m(z, E_k) = \min p(z, e_v(i))$ до елементу того ж класу для кожного елементу z класу E_k, а потім віднімається перше від другого, отримуючи у результаті вектор оцінок інформативності [40-44]. Реалізація має наступний вигляд, наведений на лістингу 3.9.

Лістинг 3.9 Реалізація обчислення оцінок інформативності:

```
def _get_reduction_difference(self, keypoints1, keypoints2):
    return np.array([
    sub(
            self._find_closest(descriptor, keypoints2),
            self._find_closest_nonzero(descriptor, keypoints1)
    ) for descriptor in keypoints1
    ])
```

Де аргументи keypoints1 та keypoints2 відповідають за першу і другу множину дескрипторів відповідно, а функція sub використовується для операції віднімання. Можна побачити, що тут замість прямого використання функцій для розрахунку мінімальної відстані, реалізованих раніше використовуються вбудовані у клас методи. Ці методи приймають на вхід лише перші два параметри тих функцій. Функції find_closest і find_closest_nonzero викликаються у середині всередині цих методів, користуючись тою метрикою, що була передана на вхід до конструктору класу, методи доповнюють виклик функцій пошуку відстані.

Тепер, коли реалізований метод розрахунку оцінок інформативності можна реалізувати метод, що приймає на вхід обидва набори дескрипторів та розраховує для обидвох оцінки інформативності. Це робиться наступним чином, наведеним на лістингу 3.10.

Лістинг 3.10 Реалізація обчислення оцінок для еталонних класів:

```
def fit(self, *keypoints):
```

```
        keypoints1, keypoints2 = keypoints
        self._first_rates = self._get_reduction_difference(
keypoints1, keypoints2)
        self._second_rates = self._get_reduction_difference(
keypoints2, keypoints1)
```

У цьому методі просто викликаються методи розрахунку інформативності, спочатку для першої множини відносно другої, а потім для другої відносно першої, запамятовуючи результат всередині об’єкта класу моделі. Таким чином можна буде у будь який час повернутися до оцінок інформативності отриманих під час навчання, та використати їх повторно.

Врешті решт, можна перейти до процесу фільтрації ознак за оцінками інформативності. Цей етап також представлений у якості методу, що буде обирати ту кількість ознак, яку нам потрібно, за переданим на ініціалізації об’єкту моделі значенням. Реалізація наведена на лістингу 3.11.

Лістинг 3.11 Реалізація визначення індексів найбільш інформативних ознак:

```
def _get_reduction_indexes(self, reduction_rates):
        sorted_indexes = sorted(
enumerate(reduction_rates),
key=lambda pair: pair[1],
reverse=True
)
        return [
index for index, rate in sorted_indexes[:self.filter_count]
]
```

Метод _get_reduction_indexes приймає на вхід оцінки інформативності, потім сортує їх і додає до них індекси за допомогою вбудованого методу

enumerate. Далі застосовується сортування у порядку зменшення інформативності. Отже, маючи відсортовані за інформативністю індески можна брати ту кількість найбільш значущих, що потрібна. Використовуючи цей метод реалізується останній, що буде приймати на вхід обидві множини дескрипторів та обирати для кожної лише найбільш інформативні ознаки. Цей метод матиме наступний вигляд, описаний на лістингу 3.12.

Лістинг 3.12 Реалізація редукції ознак для множин ключових точок:

```
def transform(self, keypoints1, keypoints2):
    first_reduction_indexes = self._get_reduction_indexes(
                                                self._first_rates)
    second_reduction_indexes = self._get_reduction_indexes(
                                                self._second_rates
                                            )
    return keypoints1[first_reduction_indexes],
                                    keypoints2[second_reduction_indexes]
```

Таким чином була отримана повна модель відбору ознак за критерієм інформативності, яку можна використовувати далі.

3.5 Аналіз результатів моделювання

Для того, щоб дослідити роботу моделі, спочатку треба зробити звіт за оцінками інформативності та подивитися на такі показники як максимальне та мінімальне значення оцінок, їх розкид, сумарну та середню інформативність, а також що вони являють собою взагалі. Для того щоб отримати оцінки інформативності, створюється об’єкт моделі та ініціалізується з метрикою Хемінга та бажаною кількістю відібраних ознак у кількості п’ятидесяти. На вхід подаються дві множини дескрипторів, що розглядаються, а на виході буде

отриманий об’єкт, який містить оцінки інформативності. Для того щоб розглянути оцінки інформативності обидвох множин, пропонується створити звіти для них. На рисунку 3.10 продемонстрований звіт за оцінками інформативності ознак першої множини.

```
Оцінки інформативності першої множини:
Оцінки: [-10, -3, 0, -1, 0, 1, 0, 0, 0, 1, -11, 0, -1, 1, 0, 0, 0, -2, -10, 0, 0, -8, 0, 1,
Максимум: 3
Мінімум: -11
Розкид: 14
Сумарна: -228
Середня: -2.28
```

Рисунок 3.10 – Звіт за оцінками інформативності ознак першої множини

Як можна побачити на рисунку, ознаки першої множини мають оцінки інформативності від -11 до 3 із розкидом оцінок у 14. По-перше, мінімальна оцінка є негативною, це свідчить що присутні ознаки які значно більше відносяться до інших множин і мають дуже малу інформативність. Також можна побачити, що за модулем мінімальна оцінка значно більша за максимальну, це в свою чергу свідчить що присутні ознаки, які можуть дуже погано впливати на процес навчання за цими ознаками. Такі показники як сумарна і середня інформативність говорять про те, що таких ознак дуже багато, або що ті що є викликають тенденцію до зменшення інформативності усього класу. Гарним результатом роботи моделі буде збільшення цих показників і зменшення розкиду оцінки. Подивимося на те яким чином фільтрація вплинула на оцінки ефективності. Також, пропонується скористатися можливостями бібліотеки matplotlib та порівняти такі показники початкових та редукованих ознак як максимальна оцінка, мінімальна оцінка, середня та сума оцінок, вивівши їх для обох множин на гістормах, щоб візуально порівняти динаміку.

Як можна побачити на рисунку 3.11 максимальна оцінка не змінилася, це свідчить що алгоритм ніде не дав збою. Мінімальна оцінка піднялась до 0.

Такий приріст є дуже гарним, вдалося позбутися всіх негативних оцінок. Це дозволяє бути певними про те, що жодна із виділених ознак не бути негативно впливати на процес подальшого навчання за ними. Найкращим результатом було б, щоб усі ознаки, присутні після редукції, були б позитивною оцінкою. У такому разі можна було б говорити про те що всі ознаки, що залишилися, повністю інформативні. Як можна вирахувати, розкид оцінок зменшився до 3, це означає, що концентрація оцінок однієї і тієї ж інформативності збільшилась. Середня оцінка тепер позитивна, як і було бажано. Це, в свою чергу свідчить про те, що у процесі навчання були отримані деякі якісні результати. Також позитивною стала і сума оцінок.

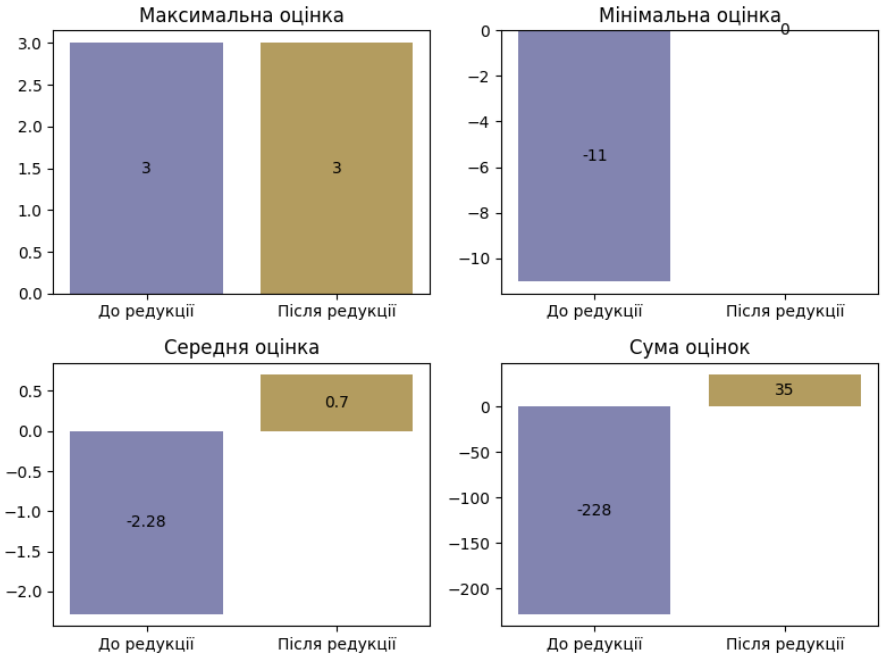

Рисунок 3.11 − Порівняння статистичних показників оцінок інформативності першого класу до і після редукції

Можна зробити попередні висновки про те, що процес редукції ознак для першої множини дескрипторів пройшов задовільно. Тепер можна подивитися на те яким чином редукція ознак вплинула на другу множину. Виведемо звіт для оцінок інформативності ознак другої множини.

На рисунку 3.12, наведеному нижче, можна побачити більшість тих самих проблем, що спостерігалися і для ознак першої множини.

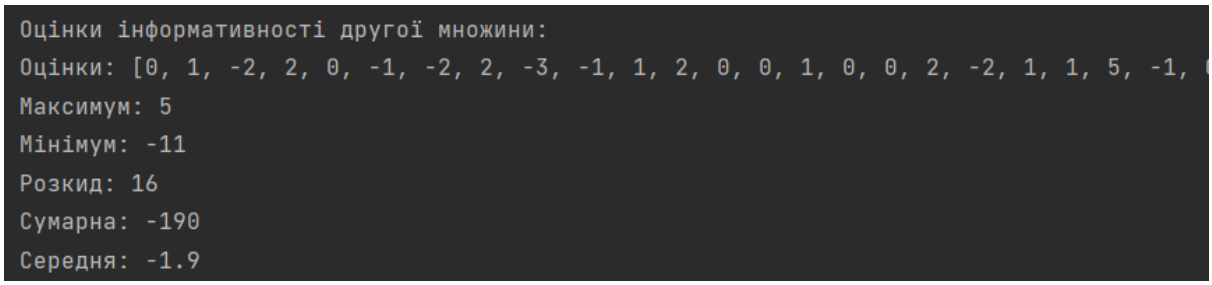

Рисунок 3.12 – Звіт за оцінками інформативності ознак другої множини

Негативні показники сумарної оцінки, середньої та мінімальної – це те, чого бажано позбутися для редукованої множини. Із примітних спостережень, можна зауважити, що для другої множини середня і сумарна оцінки більші за статистичні показники оцінок інформативності першої множини. Після редукції ознак другої множини можна буде очікувати на кращі результати. Подивимось на те, що вийшло у результаті на рисунку 3.13. Там можна побачити, що, як і у випадку із першим класом, маються позитивні тенденції росту показників сумарної, мінімальної та середньої оцінок. Мінімальна оцінка також дорівнює нулю, а ось показники середньої і сумарної оцінок більші за ті що були у випадку із першим класом, як і очікувалось. Максимальна оцінка залишилась незмінною.

Проаналізувавши попередні оцінки інформативності та оцінки інформативності для редукованої системи ознак, можна сказати про те, що процес редукції мав позитивний вплив на якість множини ознак. Серед негативного, можна відмітити лише те, що мінімальна оцінка інформативності як для першої редукованої системи ознак, так і для другої дорівнює нулю, а отже і у кожній із множин присутні ознаки, що можуть ніяким чином не впливати на процес навчання за цими системами ознак.

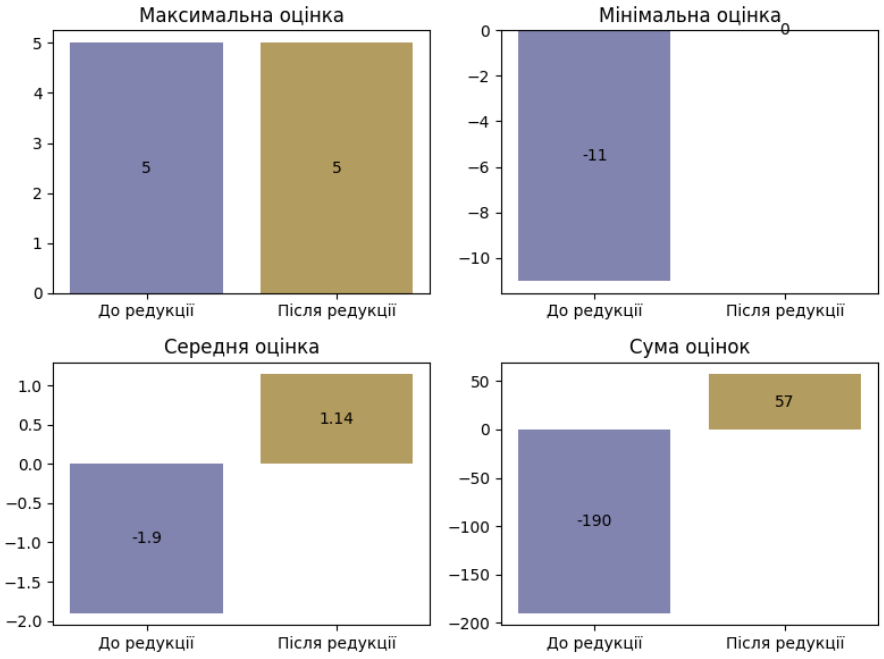

Рисунок 3.13 − Порівняння статистичних показників оцінок інформативності другого класу до і після редукції

3.6 Аналіз впливу редукції ознак

Маючи редуковану систему ознак пропонується дослідити те, яким чином це вплине на роботу із нашими множинами дескрипторів. Для цього можна знову оцінити подібність множин між собою за допомогою метрик Хаусдорфа і Танімото і проаналізувати результати. Отже, викликаються реалізовані методи знов, але тепер для редукованих множин. Результат виклику метрик для редукованих множин наведений на рисунку 3.14.

Як можна побачити на рисунку, після редукції відстані між класами збільшилися. Нагадаємо, що до цього відстань за метрикою Хаусдорфа відстань дорівнювала 12, а за метрикою Танімото − 0.8571428571428571. Тепер отримані оцінки у 13 та 1 за відповідними метриками. Такий результат закономірний, вбачаючи те, що були виключені ті ознаки які несли негативну інформативність для еталонних класів. а отже, залишились лише ті дескриптори, що описують виключно класи яким вони належать [45-49].

```
In [15]: hausdorff_distance(nvset1, nvset2, hamming_distance)
Out[15]: 13

In [16]: tanimoto_distance(nvset1, nvset2)
Out[16]: 1.0
```

Рисунок 3.14 – Оцінка подібності двох редукованих множин дескрипторів за метриками Хаусдорфа і Танімото

Також можна дослідити те яким чином зміниться концентрація бінарних показників у дескрипторах. Для цього пропонується скористатися реалізовано раніше функцією keypoints_report, результати виконання якої наведені на рисунку 3.15.

```
In [20]: keypoints_report(nvset1)
Властивості множини дестрипторів:
Очікувана середня вірогідність появи 1: 0.489375

In [21]: keypoints_report(nvset2)
Властивості множини дестрипторів:
Очікувана середня вірогідність появи 1: 0.6175
```

Рисунок 3.15 – Характеристики першої та другої редукованої множини дескрипторів

На рисунку 3.15 знову можна побачити, що класи віддалилися один від одного і за розподілом значень дескрипторів. Це зайвий раз доводить, що ознаки наших множини дескрипторів тепер мають більше індивідуальності. Тепер можна затвердити, що реалізований метричний метод відбору ознак дозволяє вирішити проблему позбавлення від неякісних та неінформативних досить добре.

Іншою задачею редукції ознак є поліпшення просторової та часової складності виконання обчислень з даними. Для того щоб перевірити яким

чином редукція ознак вплинула на ці показники пропонується скористатися можливостями стандартних бібліотек Python та інтерпретатора IPython. Для оцінки того, наскільки менше пам’яті займають редуковані дані можна скористатися функцією Python getsizeof із модуля sys. За отриманими вимірами були отримані результати у 12,06 Кб пам’яті необхідного для зберігання об’єкту для редукованого опису у той час коли повного опису у той час, коли для повного об’єм пам’яті складає 25,11 Кб. Як можна побачити, кількість пам’яті що необхідна для зберігання редукованої множини значно менша за ту, яка була потрібна на початку, а саме десь на 50%. Це свідчить про значне поліпшення стосовно просторової складності роботи з даними після редукції ознак реалізованим досліджуваним методом.

Для того щоб оцінити поліпшення з боку часу виконання обчислення пропонується скористатися утилітою інтерпретатора IPython, що називається %timeit. Вона дозволяє виміряти час, що необхідний для виконання поданої на вхід команди.

Час виконання для метрики Хаусдорфа склав 26 мілісекунд для початкових множин дескрипторів та 5,04 мілісекунд для редукованих множин. Час необхідний для обчислення цієї метрики зменшився більше ніж у п’ять разів, що є досить вагомим поліпшенням. Що стосується метрики Танімото, тут час необхідний для обчислення відстані між початковими множинами склав 1,45 мілісекунд, а час виконання операції для редукованих даних − 294 мікросекунди, що знов приблизно у п’ять разів швидше. Отже, можна із впевненістю сказати, що обчислення метрик для редкуваних множин у п’ять разів швидше за обчислення першочергових даних.

Для перевірки прикладної вагомості теоретичних викладок нами змодельовані дві скінченні множини бінарних векторів із 100 елементів із компонентами у 32 біти. Елементи множин різнилися заданими ймовірностями появи 1 у наборі із 32 бітів. Для визначення подібності між множинами векторів застосовано метрики Хаусдорфа H та Танімото

(Жаккара) *T*, де у якості внутрішньої метрики між елементами (бінарними векторами) використано метрику Хемінга.

Для двох масивів векторів з ймовірностями p_1=0,5 та p_2=0,6 появи 1 отримано значення метрик H = 12, T = 0,85. Нормована метрика *T* Танімото тут приймає значення із відрізку [0,1], а метрика *H* Хаусдорфа – цілі значення із відрізку [0,32]. Як бачимо, за метрикою Танімото ці множини суттєво відрізняються, а за метрикою Хаусдорфа – різняться приблизно наполовину.

Обчислення за виразом значень інформативності показали, що для першої множини інформативність елементів опису змінюється у межах – *11,..,3*; для другої множини – у межах *-11,..,5*. Інформативності множин продемонстровані на рисунку 3.16.

За отриманими оцінками відібрано по п’ятдесят найбільш інформативних елементів кожної із множин. Для них отримано значення метрик: H = 13 – відстань за метрикою Хаусдорфа та T = 1 – відстань за метрикою Танімото.

Як бачимо, за результатами редукції множин (у два рази зменшуються обчислювальні витрати) значення метрики Хаусдорфа збільшилось від 12 для повного опису до 13 для редукованого (трохи зменшилась подібність), а значення метрики Танімото від 0,85 до 1 відповідно.

[-10, -3, 0, -1, 0, 1, 0, 0, 0, 1, -11, 0, -1, 1, 0, 0, 0, -2, -10, 0, 0, -8, 0, 1, -10, -10, -1, 1, 0, -1, -3, 1, 0, 0, 0, 3, -10, -10, 2, 0, 0, 0, 0, 0, 2, -1, -10, 2, -10, 2, 1, -7, -9, 1, 1, -9, -2, 2, -9, -1, -1, -7, -1, 0, -10, 1, -9, 0, 1, -7, 0, -1, -2, -3, 0, 3, 1, -1, -1, 0, -1, 2, 0, 0, 0, 0, -10, -11, 0, -11, 0, 2, 0, -10, -1, -9, -8, -1, 3, -9]

[0, 1, -2, 2, 0, -1, -2, 2, -3, -1, 1, 2, 0, 0, 1, 0, 0, 2, -2, 1, 1, 5, -1, 0, 3, 1, -1, 1, 3, 3, -1, 0, 0, 1, 0, 4, 0, 2, -1, 0, -2, 0, 0, 0, 0, 0, 1, 1, 2, 1, 0, 3, 1, -1, 1, 1, -1, -1, 0, 2, 0, 1, 0, 1, -2, 0, 1, 0, 1, 1, 1, 0, 2, 0, 0, -7, -9, -7, -10, -10, -7, -10, -9, -9, -10, -6, -9, -11, -6, -10, -9, -10, -8, -9, -10, -10, -10, -10, -9, -10]

Рисунок 3.16 – Значення інформативності (1) для експериментальних множин векторів

Для інших варіантів вхідних даних, отриманих змішуванням векторів із двох розглянутих множин, відстань Танімото за результатом редукування зменшилася від значення 0,70 для повного опису до 1 для скороченого.

Відносно зменшення об’ємів пам’яті, для отриманих редукованих описів, кількість простору, необхідного для зберігання об’єкту, що представляє множину, складає 12,06 Кб. Відносно повного опису, що потребує для зберігання 25,11 Кб, вбачається зменшення простору більше ніж на п’ятдесят відсотків.

У той самий час, швидкість обробки редукованих множин також має покращення. У контексті обчислення відстані між описами, для метрики Хаусдорфа зміни у швидкості обчислення склали 5,04 мілісекунди, для редукованих множин, відносно 26 мілісекунд для повних. Для метрики Танімото отримані результати виміру швидкості обчислення становлять 294 мікросекунди та 1,45 мілісекунди, для редукованого і повного описів відповідно. Після редукування множин, швидкість обчислення відстаней між ними за отриманими ознаками відбувається, як для метрики Танімото, так і для метрики Хаусдорфа, у п’ять разів швидше за обчислення відстаней у випадку роботи із повними описами.

Проведене пробне моделювання підтверджує практичну можливість ефективного впровадження метричного критерію інформативності задля скорочення обчислювальних витрат на класифікацію. Об’єм аналізованих даних скорочується у два рази, а рівень відмінностей між описами збільшується незначно, що сприяє забезпеченню результативної класифікації [47-49].

ВИСНОВКИ

У рамках кваліфікаційної роботи розроблено та досліджено метод редукції системи структурних ознак зображення на основі метричного критерію інформативності.

Для перевірки результативності методу редукції ознак вирішено скористатися метриками Танімото і Хаусдорфа для визначення подібності множин ознак до і після редукції, що надало б розуміння впливу редукції на ці множини.

Експериментальним шляхом досліджено роботу досліджуваного методу. У якості вхідних даних було обрано еталоні класи множин бінарних дескрипторів ключових точок. Реалізована модель проведення редукції за метричними критеріями інформативності, яка була використана для вхідних даних. У якості метрики для обчислення інформативності було вирішено обрати метрику Хемінга.

У результаті редукції із використанням реалізованої моделі було вібірано половину дескрипторів із загальної множини. За отриманими результатами можна побачити, що редуковані множини ознак стали займати в половину менше простору, а також роботи з ними на прикладі обчислення метрик Танімото і Хаусдорфа пришвидшилась у п’ять разів, що надало змогу підвердити факт про цю властивість редукції. За метриками Танімото і Хаусдорфа множини ознак еталонних класів незначно віддалились один від одного, що свідчить про позбавлення від сторонніх ознак. Це дає змогу підтвередити властивість відсіювання ознак із негативним вливом для досліджуваного методу.

За результатами експериментального дослідження можна сказати про те, що метод редукції системи структурних ознак на основі критерію інформативності є досить ефективним як в аспекті зменшення необхідних ресурсів для обробки даних, так і задля виділення найбільш важливих ознак.

Результати роботи апробовано у вигляді тез доповіді під час Міжнародної конференції «SCIENCE, THEORY AND PRACTICE» у Токіо [50].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Bishop, Christopher (2006). Pattern recognition and machine learning. Berlin: Springer. ISBN 0-387-31073-8.

2. Motoda, H., & Liu, H. (2002). Feature selection, extraction and construction. Communication of IICM (Institute of Information and Computing Machinery, Taiwan) Vol, 5(67-72), 2.

3. Шапиро, Л., Стокман, Д., Богуславский, А. А., & Соколов, С. М. (2013). Компьютерное зрение.

4. Kodituwakku, S. R., & Selvarajah, S. (2004). Comparison of color features for image retrieval. Indian Journal of Computer Science and Engineering, 1(3), 207-211.

5. Arivazhagan, S., Shebiah, R. N., Nidhyanandhan, S. S., & Ganesan, L. (2010). Fruit recognition using color and texture features. Journal of Emerging Trends in Computing and Information Sciences, 1(2), 90-94.

6. Ke, Y., & Sukthankar, R. (2004, June). PCA-SIFT: A more distinctive representation for local image descriptors. In Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004. (Vol. 2, pp. II-II). IEEE

7. Lowe, D. G. (1999, September). Object recognition from local scale-invariant features. In Proceedings of the seventh IEEE international conference on computer vision (Vol. 2, pp. 1150-1157). Ieee.

8. Panchal, P. M., Panchal, S. R., & Shah, S. K. (2013). A comparison of SIFT and SURF. International Journal of Innovative Research in Computer and Communication Engineering, 1(2), 323-327.

9. Rublee, E., Rabaud, V., Konolige, K., & Bradski, G. (2011, November). ORB: An efficient alternative to SIFT or SURF. In 2011 International conference on computer vision (pp. 2564-2571). Ieee

10. Battezzati, N., Colazzo, S., Maffione, M., & Senepa, L. (2012, March). SURF algorithm in FPGA: A novel architecture for high demanding industrial applications. In 2012 Design, Automation & Test in Europe Conference & Exhibition (DATE) (pp. 161-162). IEEE.

11. Luo, C., Yang, W., Huang, P., & Zhou, J. (2019, June). Overview of image matching based on ORB algorithm. In Journal of Physics: Conference Series (Vol. 1237, No. 3, p. 032020). IOP Publishing.

12. Fang, W., Zhang, Y., Yu, B., & Liu, S. (2017, December). FPGA-based ORB feature extraction for real-time visual SLAM. In 2017 International Conference on Field Programmable Technology (ICFPT) (pp. 275-278). IEEE.

13. Yao, J., Zhang, P., Wang, Y., Luo, Z., & Ren, X. (2019). An adaptive uniform distribution ORB based on improved quadtree. IEEE Access, 7, 143471-143478.

14. Van Der Maaten, L., Postma, E., & Van den Herik, J. (2009). Dimensionality reduction: a comparative. J Mach Learn Res, 10(66-71), 13.

15. Fukunaga, K. (2013). Introduction to statistical pattern recognition. Elsevier.

16. Wyse, N., Dubes, R., & Jain, A. K. (1980). A critical evaluation of intrinsic dimensionality algorithms. Pattern recognition in practice, 415-425.

17. Jolliffe, I. T. (2002). ISBN 978-0-387-95442-4,“. Principal component analysis, series: Springer series in statistics, 29(487), 28.

18. Blum, A. L., & Langley, P. (1997). Selection of relevant features and examples in machine learning. Artificial intelligence, 97(1-2), 245-271.

19. Stańczyk, U. (2015). Feature evaluation by filter, wrapper, and embedded approaches. In Feature Selection for Data and Pattern Recognition (pp. 29-44). Springer, Berlin, Heidelberg.

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

21. Oliinyk, A., Subbotin, S., Lovkin, V., Blagodariov, O., & Zaiko, T. (2017). The system of criteria for feature informativeness estimation in pattern recognition. Радіоелектроніка, інформатика, управління, (4 (43))

22. Гороховатський, В. О., Гадецька, С. В., Жадан, О. В., & Хвостенко, О. О. (2021). Дослідження результативності класифікаторів зображень за статистичними розподілами для компонентів структурного опису.

23. Moszyńska, M., & Richter, W. D. (2012). Reverse triangle inequality. Antinorms and semi-antinorms. Studia Scientiarum Mathematicarum Hungarica, 49(1), 120-138.

24. Leskovec, J., Rajaraman, A., & Ullman, J. D. (2020). Mining of massive data sets. Cambridge university press.

25. Henrikson, J. (1999). Completeness and total boundedness of the Hausdorff metric. MIT Undergraduate Journal of Mathematics, 1(69-80), 10.

26. Srinath, K. R. (2017). Python–the fastest growing programming language. International Research Journal of Engineering and Technology (IRJET), 4(12), 354-357.

27. Sanner, M. F. (2008). The Python interpreter as a framework for integrating scientific computing software-components. Scripps Res. Inst., 26(1), 1-12.

28. Van Der Walt, S., Colbert, S. C., & Varoquaux, G. (2011). The NumPy array: a structure for efficient numerical computation. Computing in science & engineering, 13(2), 22-30.

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

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

31. Kohonen, T. (1991). Self-organizing maps: ophmization approaches. In Artificial neural networks (pp. 981-990). North-Holland.

32. Лесковец, Ю., Раджараман, А., & Ульман, Д. Д. (2016). Анализ больших наборов данных. М.: ДМК Пресс, 488.

33. Bai, Q., Li, S., Yang, J., Song, Q., Li, Z., & Zhang, X. (2020). Object detection recognition and robot grasping based on machine learning: A survey. IEEE Access, 8, 181855-181879.

34. Flach, P. (2012). Machine learning: the art and science of algorithms that make sense of data. Cambridge University Press.

35. Kira, K., & Rendell, L. A. (1992). A practical approach to feature selection. In Machine learning proceedings 1992 (pp. 249-256). Morgan Kaufmann.

36. Kruse, R., Borgelt, C., Braune, C., Mostaghim, S., Steinbrecher, M., Klawonn, F., & Moewes, C. (2011). Computational intelligence. Vieweg+ Teubner Verlag.

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

38. Гавриленко, С. Ю., Шевердін, І. В., & Гейко, Г. В. (2021). Оцінка інформативності та вибір ознак при ідентифікації стану комп'ютерної системи.

39. Гороховатський, В. О., Гадецька, С. В., & Стяглик, Н. І. (2020). Вивчення критеріїв інформативності даних при впровадженні апарату дерев рішень у методах структурної класифікації зображень.

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

41. 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).

42. Gorokhovatsky, V. A., Gadetska, S. V., & Stiahlyk, N. I. (2019). Вивчення статистичних властивостей моделі блочного подання для множини дескрипторів ключових точок зображень. Радіоелектроніка, інформатика, управління, (2), 100-107.

43. Гороховатський, В. О., & Творошенко, І. С. (2021). Методи інтелектуального аналізу та оброблення даних: навч. посібник.

44. Ayaz, A. M., Gorokhovatskyi, V., Tvoroshenko, I., Vlasenko, N., & Khalid, M. S. (2021). The Research of Image Classification Methods Based on the Introducing Cluster Representation Parameters for the Structural Description.

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

46. Gorokhovatskyi, V. O., & Gadetska, S. V. (2019). Determination of Relevance of Visual Object Images by Application of Statistical Analysis of Regarding Fragment Representation of their Descriptions. Telecommunications and Radio Engineering, 78(3).

47. Gorokhovatskyi, V., Gorokhovatskyi, O., Yevgenyi, P., & Olena, P. (2018, August). Quantization of the space of structural image features as a way to increase recognition performance. In 2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP) (pp. 464-467). IEEE.

48. Gorokhovatskyi, V. A., Rusakova, N., & Tvoroshenko, I. S. (2020). The application of image analysis methods and predicate logic in applied problems of magnetic monitoring. Telecommunications and Radio Engineering, 79(20).

49. Daradkeh, Y. I., Gorokhovatskyi, V., Tvoroshenko, I., Gadetska, S., & Al-Dhaifallah, M. (2021). Methods of Classification of Images on the Basis of the Values of Statistical Distributions for the Composition of Structural Description Components. IEEE Access, 9, 92964-92973.

50. Гороховатський, В., & Метелев, В. (2021). Редукція структурного опису зображення на основі критерію інформативності.