Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

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

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

МОДЕЛЮВАННЯ НЕЙРОМЕРЕЖІ ХЕМІНГА ДЛЯ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ
(тема)

Виконав:
здобувач 4 року навчання,
групи ІТІНФ-21-3
Чиж А. О.
(прізвище, ініціали)

Спеціальність 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

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

2025 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту

Кафедра Інформатики

Рівень вищої освіти перший (бакалаврський)

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

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Чижу Артему Олеговичу
(прізвище, ім'я, по батькові)

1. Тема роботи Моделювання нейромережі Хемінга для класифікації зображень

затверджена наказом університету від 19 травня 2025 року № 381Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 29 травня 2025 р.
3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, бібліотека комп'ютерного зору з відкритим кодом OpenCV.

4. Перелік питань, що потрібно опрацювати в роботі
1. Огляд традиційних методів класифікації зображень та вивчення методів з використанням апарату ключових точок та дескрипторів.
2. Розробка методу класифікації на основі нейронної мережі Гемінга.
3. Програмне моделювання та тестування методу.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) актуальність проблеми обробки зображень, постановка задачі кваліфікаційної роботи, еталонні зображення, трансформовані зображення, аналіз результатів програмного моделювання.

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

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

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	07.04.2025	
2	Аналіз завдання, підбір літератури	09.04.25-12.04.25	
3	Аналіз літератури з досліджуваної проблеми	13.04.25-15.04.25	
4	Аналіз технічних засобів	16.04.25-22.04.25	
5	Розробка методу	23.04.25-27.04.25	
6	Програмна реалізація	28.04.25-11.05.25	
7	Оформлення пояснювальної записки	12.05.25-01.06.25	
8	Перевірка на нормоконтроль	21.05.25-01.06.25	
9	Перевірка на плагіат	21.05.25-01.06.25	
10	Рецензування	21.05.25-01.06.25	
11	Підготовка презентації та доповіді	21.05.25-18.06.25	
12	Занесення роботи в електронний архів	02.06.25-18.06.25	
13	Попередній захист кваліфікаційної роботи	02.06.25-18.06.25	

Дата видачі завдання 7 квітня 2025 р.

Здобувач _____
(підпис)

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

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 67 с., 4 табл., 14 рис., 59 джерел.

КЛАСИФІКАЦІЯ ЗОБРАЖЕНЬ, ДЕСКРИПТОРИ, МЕРЕЖА ГЕМІНГА, MAXNET, МЕДОЇД, ВІДСТАНЬ ГЕМІНГА.

Об’єктом роботи є процес розпізнавання зображень на підставі ознак у формі множини дескрипторів ключових точок.

Метою роботи є створення методу класифікації зображень, що поєднує дескриптори BRISK та мережу Гемінга з метою забезпечення високої точності при низьких обчислювальних витратах.

Для кожного еталону формується узагальнений вектор ознак шляхом побітового аналізу множини дескрипторів. Для класифікації реалізовано мережу Гемінга, що забезпечує вибір найбільш схожого прототипу на основі відстані Гемінга.

Проведено тестування на прикладі задачі розпізнавання порід собак із застосуванням геометричних трансформацій зображень.

Розроблений метод забезпечує високу швидкодію та помірну стійкість до геометричних перетворень, що підтверджено результатами експериментів.

IMAGE CLASSIFICATION, DESCRIPTORS, HAMING NETWORK, MAXNET, MEDOID, HAMING DISTANCE.

The object of the work is the process of image recognition based on features in the form of a set of keypoint descriptors.

The aim of the work is to create an image classification method that combines BRISK descriptors and a Hamming network to ensure high accuracy at low computational costs.

For each standard, a generalized feature vector is formed by a bit-wise analysis of the set of descriptors. A Hamming network is implemented for classification, which provides the selection of the most similar prototype based on the Hamming distance.

Testing was carried out on the example of the dog breed recognition problem using geometric image transformations.

The developed method provides high performance and moderate resistance to geometric transformations, which is confirmed by the results of experiments.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

мс – мілісекунди

с – секунди

XOR – Exclusive or (виключне або)

BRISK – Binary Robust Invariant Scalable Keypoints (бінарні робустні інваріантні масштабовані ключові точки) – алгоритм екстракції бінарних дескрипторів локальних ознак, інваріантний до масштабу та повороту

Дескриптор – числовий вектор, що описує околицю ключової точки зображення

Медоїд – узагальнений бінарний вектор, сформований за правилом більшості бітів із множини дескрипторів класу

Відстань Гемінга – кількість позицій, у яких два бінарні вектори відрізняються

Мережа Гемінга – двошаровий нейронний класифікатор, що використовує скалярний добуток і латеральне гальмування для визначення найбільш схожого класу

MAXNET – алгоритм латерального гальмування, який реалізує змагальний вибір єдиного нейрона-переможця

Латеральне гальмування – механізм взаємного пригнічення нейронів, що дозволяє залишити лише найбільш активований

Ключова точка – піксель зображення з високою інформативністю, обраний детектором ознак

Зображення-еталон – приклад об’єкта певного класу, використаний для побудови прототипу

ВСТУП

Сучасні системи комп’ютерного зору вимагають високої точності розпізнавання об’єктів з мінімальними обчислювальними витратами. Зі швидким зростанням обсягів візуальних даних виникає потреба в класифікаторах, які можуть ефективно обробляти зображення в режимі реального часу. Традиційні підходи, засновані на глибоких нейронних мережах, досягають високої точності, але їх навчання та виконання є ресурсомісткими. Тому терміново потрібно розробити методи класифікації, які поєднують швидкість з достатньою точністю завдяки використанню оптимізованих ознак та компактних моделей.

Метою роботи є розробка інтегрованого підходу до класифікації зображень, який забезпечує швидке та надійне розпізнавання об’єктів. Для досягнення цієї мети пропонується поєднати такі компоненти: вилучення інформативних бінарних ознак із зображень за допомогою алгоритму BRISK; агрегування набору дескрипторів кожного класу в єдиний прототип (медоїд) для компактного представлення класу; класифікація векторів ознак за допомогою мережі Гемінга з механізмом латерального гальмування (maxnet) для вибору найбільш схожого прототипу. Цей підхід розроблений для поєднання переваг швидкості бінарних дескрипторів зображень та стійкості нейронних мереж до шуму та неповних даних.

Для досягнення цієї мети було вирішено такі завдання: проаналізувати сучасні методи класифікації зображень та вибрати оптимальні для поставленого завдання; розробити математичну модель класифікатора, яка поєднує вибрані компоненти; реалізувати програмну систему класифікації з використанням бібліотек комп’ютерного зору; провести серію комп’ютерних експериментів для оцінки точності та швидкості розробленого методу, зокрема, дослідити його роботу на прикладі класифікації порід собак за різних

геометричних перетворень зображень; проаналізувати отримані результати та зробити висновки щодо ефективності підходу.

Запропонована методологія спрямована на підвищення стабільності розпізнавання об’єктів за наявності шуму, часткової відсутності даних або варіацій зображення (повороти, масштабування тощо) шляхом використання агрегації дескрипторів (голосування за ознаками) та конкурентного вибору рішень у мережі Гемінга.

1 КЛАСИФІКАЦІЯ ЗОБРАЖЕНЬ ЗА МНОЖИНОЮ ДЕСКРИПТОРІВ

1.1 Дескриптори ключових точок зображення

Комп'ютерний зір – міждисциплінарна галузь, яка охоплює інформатику, штучний інтелект, обробку зображень та статистику, й має на меті надати комп'ютерам здатність «бачити», тобто сприймати, аналізувати й інтерпретувати візуальні дані (зображення або відео) аналогічно до того, як це робить людина. На відміну від простої обробки зображень, де фокус зосереджено на трансформації чи фільтрації даних, комп'ютерний зір прагне до розуміння змісту зображення [1, 2].

Основними завданнями комп'ютерного зору [3, 4] є виявлення та ідентифікація об'єктів, сегментація зображення, оцінка глибини сцени, реконструкція тривимірних моделей, розпізнавання текстів, відстеження рухомих об'єктів та ін. В останні роки розвиток глибокого навчання, зокрема згорткових нейронних мереж [2], суттєво підвищив точність і універсальність методів комп'ютерного зору у таких задачах, як розпізнавання облич, автономне водіння, медична діагностика, тощо.

Перші спроби створити системи, здатні інтерпретувати візуальні дані, з'явилися у 1960-х роках (рис. 1.1). Тоді дослідники намагались розпізнавати прості об'єкти, наприклад геометричні фігури, у контрольованому середовищі. У 1970-х з'являються перші підходи до сегментації та опису зображень. У 1980-х починають активно розроблятися методи для реконструкції тривімірної сцени з двовимірного зображення [1-7].

У 1990–2000-х роках набуває популярності концепція виявлення ключових точок і побудова дескрипторів [8]. Нову еру відкрив прорив глибокого навчання: так званий алгоритм AlexNet [2], розроблений у 2012 році показав вражаючі результати, задавши тренд на використання нейронних

мереж у комп’ютерному зорі. З цього моменту більшість задач вирішуються із залученням навчання на великих наборах даних, а класичні методи відійшли на другий план, залишаючись актуальними у реальному часі та обмежених середовищах

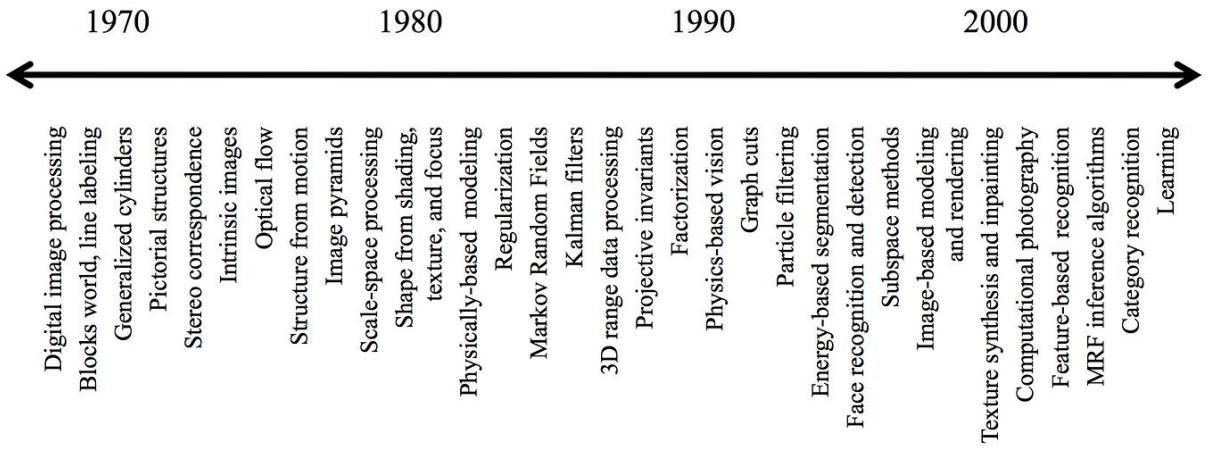


Рисунок 1.1 – Історія розвитку комп’ютерного зору

Сучасний комп’ютерний зір охоплює низку спеціалізованих напрямків, серед яких [1, 9, 10]:

– класифікація зображення як цілого до одного з наявних класів;

– виявлення об’єктів – не лише ідентифікація об’єктів, а й локалізація їх у просторі;

– класифікація кожного пікселя зображення до певного класу;

– визначення положення частин тіла або ключових точок об’єкта;

– тривимірна реконструкція – побудова тривимірної сцени з двовимірних проєкцій;

– SLAM – Simultaneous Localization and Mapping (одночасна локалізація та картографування) – використовується у робототехніці, а також технологіях AR – Augmented Reality (доповнена реальність) та VR – Virtual Reality (віртуальна реальність) для орієнтації у просторі;

– самонавчання – навчання моделей без ручної анотації на великих масивах візуальних даних.

У рамках кваліфікаційної роботи особливу увагу приділено саме проблемам класифікації зі спробами вирішити проблеми швидкодії для вдалого запуску і роботи на пристроях малої потужності. Одним із ключових етапів побудови системи класифікації зображень є виділення ознак за допомогою детектора, на основі яких здійснюється розпізнавання. У контексті комп'ютерного зору, детектором називають алгоритм або модель, що виявляє на зображенні стійкі, помітні області (кути, контури, контрастні ділянки), які можуть служити основою для зіставлення з іншими зображеннями.

Ключові точки – це пікселі або малі області на зображенні, що мають унікальні локальні характеристики, які дозволяють ідентифікувати їх під час зіставлення з іншими зображеннями. Зазвичай вони відповідають кутам, перехрестям текстур або іншим різким змінам у яскравості [11].

Ключові точки використовуються у багатьох задачах: реєстрація зображень, відстеження, реконструкція сцен, класифікація. Їхня ефективність визначає загальну точність та стабільність систем комп'ютерного зору в складних умовах освітлення чи трансформацій [11, 12].

Особливо цінними для детекторів ключових точок є властивості інваріантності (до масштабу, освітлення, повороту) та повторюваності (можливість виявити ті самі точки на подібних зображеннях) [13, 14].

Широко застосовуються дескриптори ключових точок – вектори, що описують околиці особливих точок зображення (кутів, контурів та ін.). Класичними підходами є дескриптори SIFT – Scale-Invariant Feature Transform (масштабоване ознакове перетворення) та SURF – Speed Up Robust Features (прискорені стійкі ознаки), які забезпечують високу інформативність, але є відносно повільними через використання багатовимірних дійсних векторів ознак. Натомість більш сучасні алгоритми, такі як BRISK – Binary Robust Invariant Scalable Keypoints (бінарні стійкі масштабовані ключові точки), генерують бінарні дескриптори, що складаються лише з 0 та 1 [14-16]. Бінарні

ознаки суттєво знижують обчислювальну складність, оскільки порівняння таких векторів можна виконувати швидкими побітовими операціями, як XOR – Exclusive Or (виключне «або»).

У комп’ютерному зорі ознаки поділяють на локальні та глобальні, залежно від області зображення, яку вони описують.

Локальні ознаки витягуються з невеликої ділянки навколо ключової точки. Вони стійкі до змін масштабу, повороту, освітлення і тому широко застосовуються у задачах зіставлення, відстеження та побудови панорам [14, 15]. Завдяки своїй «місцевості», локальні ознаки особливо корисні у випадках, коли сцена частково закрита або містить багато об’єктів.

Глобальні ознаки, навпаки, описують усе зображення як одне ціле. Прикладами є гістограма кольорів, текстурні характеристики, або активації останніх шарів згорткової нейронної мережі [1, 17]. Вони забезпечують ширше охоплення, але менш стійкі до змін масштабу чи фону. Тому їх частіше застосовують у задачах класифікації зображень, де важливий загальний контекст, а не точне положення об’єктів. У практичних системах обидва типи ознак можуть використовуватись спільно – локальні для точного виявлення й відповідності, глобальні – для попередньої фільтрації чи класифікації.

В даній роботі обрано алгоритм BRISK для екстракції локальних ознак. Алгоритм BRISK належить до детекторів та дескрипторів ключових точок, що забезпечують інваріантність до масштабу і повороту [14] (рис. 1.2).

Рисунок 1.2 – Візуалізація пошуку ключових точок алгоритмом BRISK

Він складається з швидкого детектора ключових точок, базованого частково на методі FAST – Features From Accelerated Segment Test (ознаки з прискореної сегментної перевірки) та методу побудови бінарного дескриптора шляхом вибіркового порівняння інтенсивностей сусідніх пікселів навколо ключової точки (рис. 1.3). Завдяки цьому BRISK генерує для кожної ключової точки бінарний вектор фіксованої довжини (звичайно 512 біт) – унікальний «відбиток» околу точки. Перевагою BRISK є його швидкодія: він значно швидший за, приміром, SURF при співставній повторюваності ключових точок. Зокрема, відзначено, що BRISK у середньому витрачає в десятки разів менше часу на знаходження однієї ключової точки, ніж SURF, при незначному зниженні точності локалізації. Таким чином, використання BRISK у системі класифікації забезпечує високу швидкість отримання ознак і компактність їх подання (бінарні вектори) [14, 16, 18].

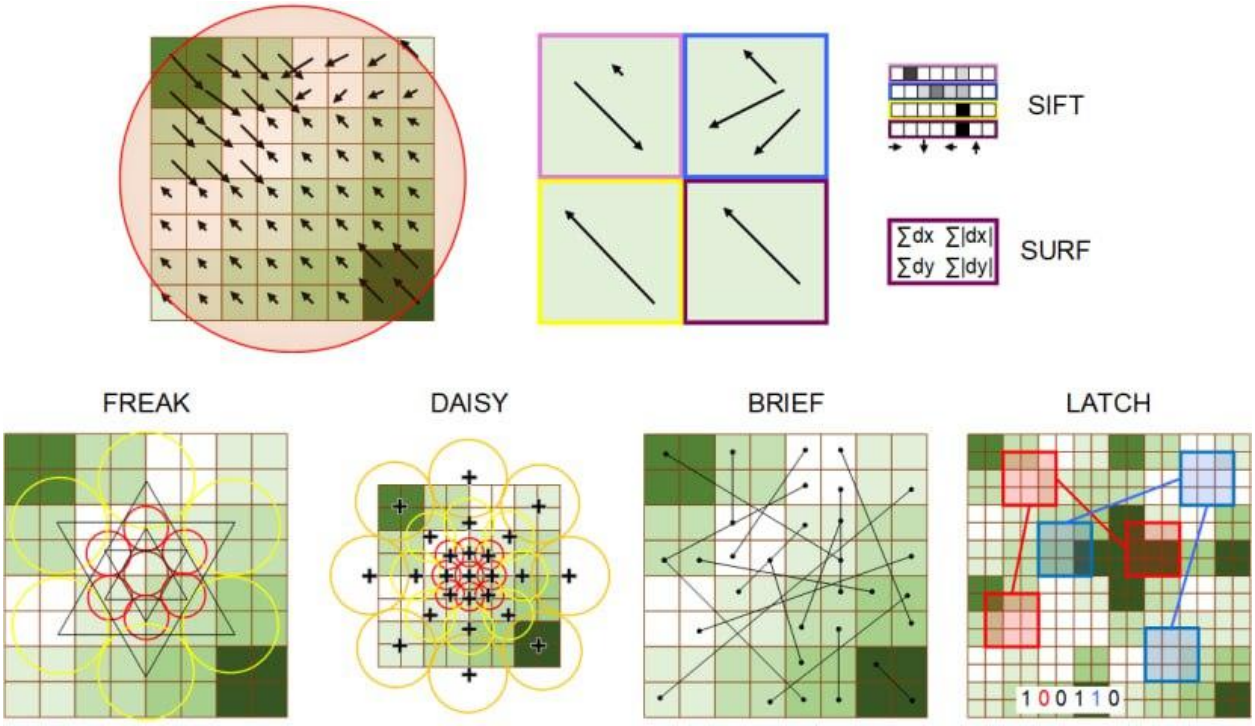


Рисунок 1.3 – Порівняння роботи детекторів ключових точок

Для ефективної класифікації образів за допомогою локальних дескрипторів необхідно вирішити проблему різної кількості та позицій ключових точок на зображеннях одного класу. Різні зображення (навіть одного об’єкта) можуть давати різну кількість дескрипторів. Тому важливо нормалізувати набір ознак до фіксованого розміру та узагальнити їх, щоб мати можливість порівнювати цілі зображення між собою. В теорії розпізнавання образів існують підходи, що ґрунтуються на побудові «візуальних слів» або кодових книг ознак, коли множина локальних дескрипторів кластеризується, а кожне зображення подається гістограмою частот належності дескрипторів до певних кластерів [1, 17]. Наш підхід спрощує цю ідею до побудови усередненого бінарного прототипу класу на основі множини дескрипторів, що фактично відповідає поняттю медоїда – представника кластера, найтиповішого для даного набору [19, 20].

1.2 Агрегація ознак з використанням медоїда

Хоча терміни медоїд і центроїд часто використовуються у контексті кластеризації, між ними існує принципова різниця як з математичної, так і з практичної точки зору [20] (рис. 1.4).

Медоїд – це елемент вибірки, який може слугувати репрезентативним центром кластера, мінімізуючи відстань до інших елементів у кластері. На відміну від центроїда (середнього значення), медоїд завжди належить початковому набору даних.

Центроїд – це середній вектор усіх елементів у кластері, обчислений як середнє арифметичне по кожній координаті . У випадку числових (небінарних) даних це дає змогу точно врахувати варіативність вибірки, але призводить до вектора, який не обов’язково належить початковому набору зразків. Медоїд, навпаки, завжди відповідає реальному зразку або, у випадку бінарних векторів, є наближеним варіантом, побудованим за правилом більшості. У

просторі {0,1} центроїд зазвичай переходить у дробове представлення (наприклад, [0.3, 0.6, 0.1,...]), яке не можна безпосередньо використовувати у порівнянні за бітовою відстанню [19-21].

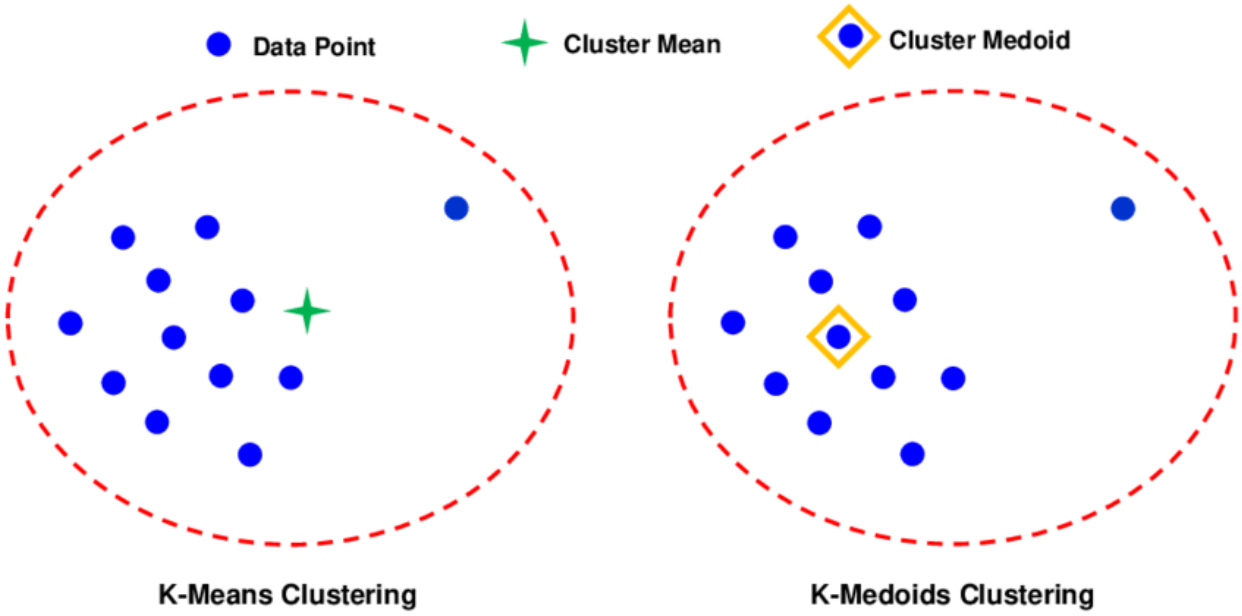


Рисунок 1.4 – Порівняння кластеризації методом середнього (центроїд) та методом медоїда.

Тоді як медоїд, будучи бінарним, зберігає формат, сумісний з швидкими метриками, такими як відстань Гемінга [22]. З цього випливає ще одна перевага: медоїд є інтерпретованим і придатним до практичного застосування у системах з жорсткими обмеженнями на тип даних, на відміну від центроїда, який вимагає додаткових кроків округлення або нормалізації [19, 21]. Таким чином, хоча центроїд є ефективним у безперервному просторі, саме медоїд краще підходить для обробки бінарних ознак, забезпечуючи баланс між узагальненням і сумісністю з алгоритмами класифікації.

Такий підхід еквівалентний обчисленню медіани по координатам в просторі {0,1} – якщо більше половини дескрипторів мають на певній позиції значення 1, то і в прототипі цей біт ставиться рівним 1, інакше 0. В результаті отримуємо узагальнений дескриптор класу, який враховує інформацію від

великої кількості зразків, але має той самий формат, що й окремі дескриптори [19, 21].

Такий спосіб формування медоїда є особливо ефективним у задачах класифікації на основі бінарних ознак, оскільки зменшує чутливість до випадкових або шумових бітів, які можуть з’являтися лише у кількох зразках. На відміну від більш складних кластеризаційних підходів, що потребують ітеративного навчання, медоїдне агрегування можна виконати за один прохід по даних, що робить його привабливим для задач реального часу. Крім того, використання єдиного узагальненого вектора зменшує обсяг пам’яті, необхідної для зберігання описів класів [19, 21].

У поєднанні з бінарними дескрипторами (наприклад, BRISK або ORB), такий прототип легко інтегрується у системи, які використовують швидкі метрики, як-от відстань Гемінга [14, 16, 22]. Завдяки цьому, класифікація нових об’єктів зводиться до порівняння їхніх дескрипторів з уже підготовленими медоїдами, що дає змогу швидко оцінити подібність і прийняти рішення. Цей підхід зберігає масштабованість навіть у випадку великої кількості класів, оскільки замість порівняння з багатьма зразками достатньо оцінити відстань до одного вектора на клас [19].

Агрегування дескрипторів у прототипний вектор має декілька переваг. По-перше, значно зменшується обсяг даних: замість зберігання сотень локальних дескрипторів для класу, ми оперуємо одним вектором фіксованої довжини. По-друге, прототип, отриманий за правилом більшості, є стійким до шумів – випадкові або несистематичні ознаки, притаманні лише окремим зображенням, будуть усереднені й не вплинуть на кінцевий результат. По-третє, такий прототип можна використовувати як еталонний зразок для швидкого порівняння з новими об’єктами [22]. Звичайно, певна доля інформації втрачається при такому узагальненні – зокрема, медоїд може згладити унікальні риси окремих представників класу. Однак у задачах, де кожен клас має достатню внутрішню однорідність, використання медоїда є виправданим компромісом між точністю та швидкістю [19, 21].

У межах класифікації зображень підходом з використанням медоїда можна вважати побудову одного «зведеного» дескриптора для кожного класу. Цей дескриптор фактично виконує роль єдиного прототипу класу. Порівняння вхідного зображення з усіма прототипами класів далі дозволяє визначити найбільш подібний клас. В якості міри подібності природно використовувати відстань Гемінга між бінарними векторами – тобто кількість бітових позицій, в яких два вектори відрізняються. Чим менша відстань Гемінга між вектором ознак об’єкта і прототипом класу, тим більша їх схожість. Таким чином, задача класифікації зводиться до пошуку прототипу з мінімальною відстанню Гемінга до опису вхідного зображення [19, 21, 22].

1.3 Огляд основних метрик для дескрипторів

Метрика – це формалізований спосіб вимірювання «відстані» або «несхожості» між об’єктами. У задачах класифікації зображень метрику зазвичай розуміють як функцію, яка чисельно оцінює ступінь подібності між векторами ознак або дескрипторами. Вона повинна задовольняти певні математичні властивості: невід’ємність, симетрія, тотожність нерозрізнених та нерівність трикутників [20, 23].

Евклідова метрика – це найпростіший та найбільш інтуїтивний спосіб вимірювання відстані між двома точками в просторі [23]. Вона базується на геометричному зображенні прямої лінії між цими точками: якщо уявити дві точки в системі координат, то евклідова відстань між ними — це довжина відрізка, що їх з’єднує. Формально вона визначається як квадратний корінь із суми квадратів різниць відповідних координат:

$$d(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2}\,, \qquad (1.1)$$

де $d(x, y)$ – відстань Евкліда між двома точками x та y у n-вимірному просторі;

$x_1, x_2, x_3, \dots, x_n$ – координати першої точки;

$y_1, y_2, y_3, \dots, y_n$ – координати другої точки.

Ця метрика лежить в основі багатьох класичних алгоритмів у математиці, фізиці, інформатиці та машинному навчанні. Вона використовується для визначення «найближчих» точок, побудови кластерів, у навігації роботів, у комп'ютерній графіці та інших галузях. Завдяки своїй простоті та геометричній чіткості евклідова відстань є першою метрикою, яка вводиться в шкільні курси, і вона часто служить базовою моделлю для порівняння з іншими, складнішими метриками [23].

Однією з найвідоміших метрик у двійковому просторі є відстань Гемінга, яка обчислюється як кількість позицій, в яких відповідні біти двох векторів відрізняються. Простота обчислення (завдяки побітовій операції XOR) робить метрику Гемінга ідеальним вибором для роботи з двійковими дескрипторами [22].

Ще однією поширеною метрикою є Мангетенська відстань, яка обчислюється як сума модулів різниць координат. Вона активно використовується в задачах, де важливо враховувати усереднену різницю між усіма елементами векторів. Зокрема, у модифікованих підходах класифікації з еталонною кластеризацією, ця метрика використовується для порівняння векторів голосів за центрами кластерів. Перевагою Мангетенської метрики є її стабільність щодо окремих змін шуму в дескрипторах [20].

Метрики простору множин, такі як метрики Хаусдорфа або Танімото, також використовуються для порівняння множин або мультимножин дескрипторів. Метрика Хаусдорфа визначає найгіршу з найближчих відстаней між елементами двох множин і добре підходить для виявлення екстремальних випадків, але чутлива до викидів [24]. Метрика Танімото (подібна до коефіцієнта Жаккара) враховує розмір перетину та об'єднання множин і

зручно нормується в [0;1], що дозволяє інтерпретувати результат як міру подібності [25].

Таким чином, вибір метрики є не лише технічним рішенням, але й важливим кроком, що визначає баланс між точністю, швидкістю та стабільністю системи. В умовах роботи з бінарними дескрипторами метрики типу Гемінга залишаються найкращим компромісом, хоча в задачах кластеризації або роботі з мультимножинами доцільно доповнювати їх альтернативними підходами.

1.4 Мережа Гемінга для класифікації образів

Мережа Гемінга є логічним продовженням і спеціалізованим варіантом мережі Кохонена, призначеним для ефективної класифікації бінарних або біполярних векторів на основі принципу найменшої Гемінгової відстані. Незважаючи на структурну схожість, ці дві мережі мають відмінні цілі, математичне формулювання і призначення. Проте історично і концептуально мережу Гемінга слід розглядати як розвиток і вдосконалення ідей, закладених у карти Кохонена із самоорганізацією [22, 26, 27].

Мережа Кохонена – це класична самонавчальна нейронна мережа, яка формує просторову топологію вхідних векторів шляхом навчання без наглядача (рис. 1.5). Основна її особливість полягає у використанні евклідової відстані між вхідним вектором і вагами нейронів для вибору так званого «нейрона-переможця». Надалі відбувається адаптація ваг сусідніх нейронів, що формує карту, відображаючи структуру вхідного простору [27, 28].

$$y_j = w_{i0} + \sum_{i=1}^{m} w_{ij} x_i, \tag{1.2}$$

де w_{ij} – ваговий коефіцієнт i-го входу j-го нейрона;

w_{j0} – пороговий коефіцієнт.

У той час як мережа Кохонена навчається без нагляду і будує топологію ознак, вона не є оптимальною для чіткої, гарантованої класифікації у просторі бінарних чи біполярних даних. Це й стало підґрунтям для створення дискретних мереж із жорстким правилом вибору, таких як мережа Гемінга.

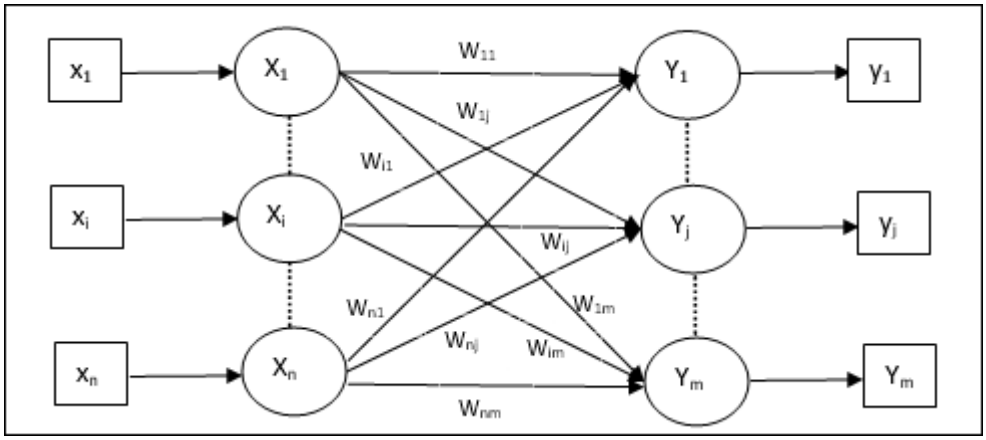


Рисунок 1.5 – Схема роботи мережі Кохонена

Мережа Гемінга формалізує ключову ідею мережі Кохонена – вибір найбільш схожого зразка із наявних, але у фіксованому класовому просторі з використанням метрики Гемінга замість евклідової. Саме завдяки такій заміні мережа Гемінга забезпечує високу ефективність при роботі з бінарними шаблонами, наприклад, дескрипторами зображень у системах комп’ютерного зору [22, 25, 29, 30].

На відміну від Кохонена, де переможець визначається на основі відстані, мережа Гемінга оперує скалярними добутками у біполярному просторі (що легко перетворюється у відстань Гемінга), а для вибору найсильнішого нейрона використовує другий шар латерального гальмування, відомий як макснет.

Завдяки побудові на основі чітких прототипів, мережа Гемінга не потребує навчання в класичному сенсі: усі ваги першого шару – це просто біполярні подання шаблонів. Це робить її ідеальною для задач класифікації, де

вже є еталонна база. Крім того, підвищена швидкодія забезпечується за рахунок простоти обчислень (логічні операції, відсутність ітеративного навчання), що дає змогу реалізувати систему на апаратному рівні або у реальному часі.

Отже, мережа Гемінга – це двошаровий нейронний класифікатор, спеціально призначений для розпізнавання вхідних векторів шляхом знаходження найближчого прототипу із заданого набору. Вона складається з: першого шару нейронів, кожен з яких відповідає за обчислення міри подібності між вхідним вектором та одним із прототипів; другого шару, який реалізує механізм латерального гальмування для придушення всіх нейронів окрім одного – того, який має найбільшу активацію.

Такий механізм дає змогу системі не лише обрати найбільш ймовірний клас, а й приглушити вплив менш релевантних відповідей, що особливо важливо у випадках, коли кілька прототипів мають схожі значення подібності. Латеральне гальмування виконує роль динамічного фільтра, який усуває неоднозначність і забезпечує стійкість класифікації, зменшуючи ймовірність помилкових спрацьовувань [22, 30]. Ця властивість вирізняє мережу Гемінга серед інших нейронних архітектур, які часто потребують додаткових механізмів нормалізації або порогових рішень.

Принцип роботи мережі Гемінга такий. Нехай маємо K прототипів $p_1, p_2, \dots p_K$ кожен з яких є вектором довжини n. Перший шар містить K нейронів, причому і-ий нейрон обчислює скалярний добуток між вектором прототипу p_i та вхідним x, який можна подати формулою:

$$S_i = \sum_{j=1}^{n} p_{ij} \cdot x_j \quad (1.3)$$

,

де S_i – скалярний добуток між вхідним вектором і прототипом;

n – довжина вхідного вектора ознак, тобто кількість бітів у дескрипторі (наприклад, n = 512 для BRISK);

p –прототип класу;

x – вхідний вектор ознак;

i – індекс класу (нейрона) у шарі мережі;

j – індекс елемента вектора, тобто позиція в ознаковому векторі.

Для біполярних векторів, тобто таких, де замість 1 та 0, вектори складаються з -1 або 1, скалярний добуток S_i пропорційний різниці між кількістю збігів та розбіжностей бітів в p_i та x. Зручним масштабуванням є обчислення активації нейрона як

$$a_i = \frac{n + S_i}{2}, \tag{1.4}$$

де a_i – активація нейрона i в першому шарі мережі Гемінга, тобто фінальне значення, яке визначає, наскільки вхідний вектор схожий на прототип класу;

n – довжина вектора ознак (512 для BRISK);

S_i – скалярний добуток між вхідним вектором і прототипом.

Легко побачити, що якщо x повністю збігається з прототипом p_i (усі бітові значення співпали), то $S_i = n$ і активація $a_i = n$, тобто максимальна. Якщо ж x зовсім не схожий (усі біти протилежні), то $S_i = -n$ і $a_i = 0$ – нейрон не збуджується. У випадку часткового збігу a_i набуває проміжного значення, яке лінійно залежить від кількості співпадінь. Таким чином, після першого шару ми отримуємо набір активацій a_i, що характеризують міру схожості вхідного образу до кожного з K прототипів.

Цей підхід дозволяє не лише визначити, наскільки схожий вхідний вектор до кожного класу, але й забезпечує чітку математичну інтерпретацію цієї схожості. Завдяки лінійній залежності між кількістю бітових збігів і рівнем активації, можна зручно контролювати чутливість системи до часткових

відповідностей. Така властивість є особливо корисною в задачах класифікації, де образи можуть бути спотвореними або неповними.

Другий шар мережі Гемінга реалізує рекурентне латеральне гальмування. Механізм латерального гальмування (алгоритм Maxnet) імітує змагальну поведінку нейронів: на кожній ітерації всі нейрони дещо пригальмовують (зменшують активації) один одного пропорційно їх поточним активаціям.

Завдяки такій організації мережа Гемінга виконує роль змагального класифікатора, де лише найсильніший нейрон «виживає», пригнічуючи решту. Це дозволяє уникнути неоднозначностей у випадках, коли кілька класів мають схожу активність на виході першого шару. Алгоритм латерального гальмування поступово зменшує активації менш релевантних нейронів, поки не залишиться один – із найбільшою початковою активацією. Таким чином, мережа реалізує стійкий механізм вибору єдиного рішення навіть у складних або шумних умовах, і це особливо важливо при роботі з реальними зображеннями, де повна відповідність – рідкісне явище.

Формально, якщо $y_i(t)$ – вихід і-го нейрона на t-тій ітерації, то на наступній ітерації обчислюється:

$$y_i(t+1) = \max\left\{y_i(t) - \epsilon\left[\sum_{j=1}^{K} y_j(t) - y_i(t)\right], 0\right\}, \qquad (1.5)$$

де $y_i(t)$ – активація нейрона i на поточній ітерації t;

$y_i(t+1)$ – активація нейрона i на наступній ітерації t+1;

K – загальна кількість нейронів у другому (MAXNET) шарі, тобто кількість класів;

ϵ – коефіцієнт гальмування, мале позитивне число, яке визначає інтенсивність придушення;

$\sum_{j=1}^{K} y_j(t)$ – сума активацій усіх нейронів на ітерації t;

$\sum_{j=1}^{K} y_j(t) - y_i(t)$ – гальмуючий вплив усіх інших нейронів на нейрон i;

Цей процес повторюється і призводить до того, що найбільша активація з часом зменшується найповільніше, тоді як менші активації дедалі більше придушуються сумою більших. Після достатньою кількості ітерацій (або коли змінами можна знехтувати) виходи мережі збігаються до стану, в якому лише один нейрон має ненульову активацію – той, чий прототип найкраще відповідає вхідному вектору. В результаті відбувається вибір переможця: індекс цього нейрона є номером знайденого найбільш схожого класу.

Такий підхід, що поєднує простоту обчислень із біологічно натхненою логікою конкуренції, забезпечує високу інтерпретованість результатів класифікації. На відміну від більш складних моделей, де важко простежити логіку прийняття рішення, мережа Гемінга дозволяє чітко побачити, як саме відбувається відбір: через пряме порівняння, змагальність та пригнічення. Це робить її привабливою в задачах, де важлива пояснюваність або обмежені обчислювальні ресурси, наприклад, вбудовані системи або освітні демонстраційні проєкти.

Мережа Гемінга гарантує знаходження найближчого прототипу принаймні для випадку, коли вхідний вектор достатньо близький до одного з еталонів. Однак у задачах класифікації важливо також визначати випадки, коли жоден з прототипів не відповідає вхідним даним [32] (наприклад, об’єкт належить невідомому класу або зображення надто спотворене). Для цього вводять критерій порогу впевненості.

Один зі способів – реалізувати після мережі Гемінга так званого голосувальника, що підраховує «голоси» за кожен клас і порівнює відносну кількість голосів переможця з заданим порогом. Якщо переможний нейрон набрав менше голосів, ніж визначено порогом, система утримується від відповіді (вирішує, що образ не розпізнано). В нашому випадку, оскільки кожне зображення описується одним узагальненим вектором, логічно визначати поріг на основі значення активації або відстані Гемінга: наприклад, задати максимальну допустиму відстань до прототипу або мінімальну частку збіжних біт. Якщо жоден клас не задовольняє цю умову, результат

класифікації вважається негативним, тому об’єкт не віднесено до жодного класу.

Отже, теоретичною основою роботи є поєднання сучасних методів аналізу зображень (детектори ключових точок та бінарні дескриптори) із класичним нейромережевим підходом до розпізнавання (мережею Гемінга). Наступний розділ присвячено побудові математичної моделі такої інтегрованої системи класифікації.

1.5 Постановка задачі

Таким чином, побудова ефективних класифікаторів зображень є актуальним завданням у галузі комп’ютерного зору. Тому ставиться завдання розробки методу, що дозволяє виконувати класифікацію об’єктів за візуальними ознаками з високою швидкодією та стійкістю до спотворень. У межах цієї роботи запропоновано поєднати бінарні локальні дескриптори BRISK із класифікацією за допомогою мережі Гемінга [34].

Об’єктом роботи є процес розпізнавання зображень за допомогою ознак ключових точок.

Метою роботи є створення інтегрованого методу класифікації зображень, що поєднує локальні бінарні дескриптори та механізм конкурентного відбору в мережі Гемінга, з метою досягнення високої точності при знижених обчислювальних витратах.

Для досягнення поставленої мети необхідно розв’язати такі завдання:

– провести огляд сучасних методів виділення ознак і класифікації зображень;

– реалізувати алгоритм екстракції бінарних дескрипторів BRISK для ключових точок;

– розробити підхід до формування узагальнених векторів ознак (медоїдів) для кожного класу;

– побудувати класифікатор на основі мережі Гемінга з латеральним гальмуванням (макснет);

– здійснити програмну реалізацію системи класифікації та провести експериментальне дослідження її ефективності на прикладі розпізнавання порід собак за зображеннями, включаючи їх трансформовані версії.

2 МАТЕМАТИЧНІ АСПЕКТИ АНАЛІЗУ ДЕСКРИПТОРІВ

2.1 Особливості екстракції дескрипторів

Вхідними даними є зображення об’єкта (фотографія собаки певної породи). Для підвищення стійкості до змін освітлення та спрощення обчислень зображення переводиться у відтінки сірого. Далі застосовується детектор ключових точок BRISK. На виході детектора отримуємо множину N ключових точок $\{k_1, k_2, \ldots, k_N\}$, для кожної з яких обчислено бінарний дескриптор довжини m біт (рис. 2.1):

$$D = \{ d_1, d_2, \ldots, d_N \mid d_i \in \{0,1\}^m \}. \quad (2.1)$$

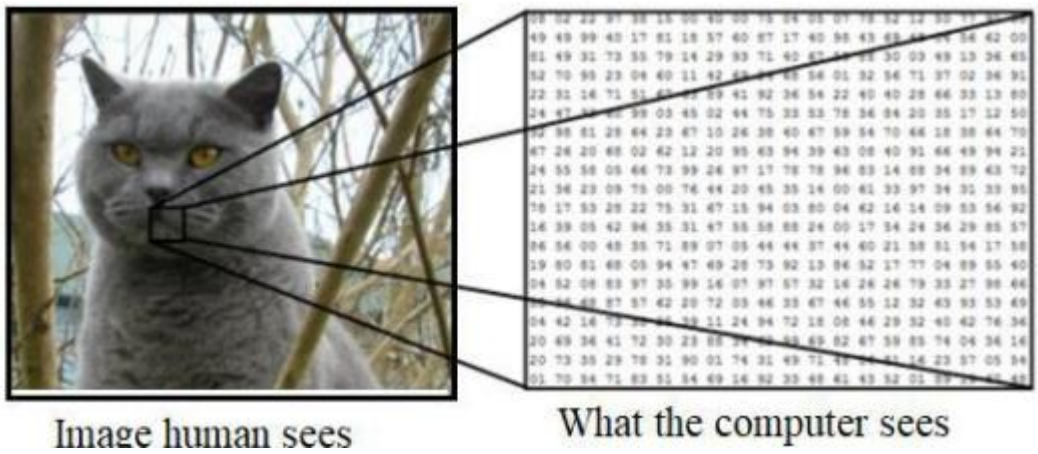


Рисунок 2.1 – Різниця у сприйнятті об’єктів людиною і машиною

Ключові точки сортуються за показником значущості (response). Для подальшої обробки беруться лише перші N_{sel} дескрипторів із відсортованої множини, де N_{sel} – заздалегідь заданий параметр. Якщо реальна кількість знайдених дескрипторів N менша за N_{sel}, виконується доповнення відсутніх дескрипторів. Доповнення можна реалізувати, наприклад, додаванням нульових векторів (усі біти 0) або дублюванням середнього дескриптора.

Таким чином, на виході першого етапу маємо фіксований набір бінарних векторів розміру $N_{sel} \times m$ для кожного зображення [35].

Зважаючи на складність розпізнавання зображень у комп'ютерному зорі, на етапі попередньої обробки важливим кроком є стандартизація вхідного зображення, зокрема його переведення в градації сірого. Це не лише знижує обчислювальне навантаження, але й нівелює вплив різких змін кольору чи освітлення, які можуть призводити до спотворення ознак. Дослідження показують, що для детекторів типу BRISK така уніфікація дозволяє підвищити стабільність ключових точок, особливо в умовах природного освітлення або при варіаціях сцени [7].

Застосування детектора BRISK є ефективним завдяки його здатності формувати інваріантні до масштабів і поворотів бінарні дескриптори. Такі дескриптори, що отримані з BRISK, добре відображають локальні особливості зображення та зберігають свої властивості навіть після геометричних деформацій. У рамках дослідження структурного аналізу, бітова природа таких дескрипторів дозволяє не тільки ефективно зберігати інформацію, але й адаптувати їх до побудови узагальнених ознак, зокрема у вигляді фрагментів чи статистичних розподілів.

Впорядкування дескрипторів за показником значущості (response) дозволяє зосередитися на найбільш інформативних точках, що суттєво підвищує результативність подальшого аналізу. Роботи з цієї теми [19, 35] демонструють, що концентрований опис – тобто обмеження аналізу лише до найсильніших дескрипторів – дозволяє зменшити кількість зайвої інформації та уникнути впливу шумів. Особливо це важливо при класифікації об'єктів у випадках, коли дані неповні або спотворені.

Формування фіксованого набору дескрипторів є ключовим для уніфікації подання зображення та забезпечення подальшої порівнюваності [3, 14]. У випадках, коли кількість знайдених ключових точок не досягає порогового значення N_{sel}, доцільним є доповнення структури [19]. Збереження фіксованої розмірності дозволяє уникнути складнощів при побудові

векторних моделей, що використовуються у класифікаторах або нейронних мережах, і забезпечує узгодженість розміру ознак у навчальній та тестовій вибірках [35].

2.2 Нормалізація та формування прототипів класів

Щоб представити кожен клас уніфікованим способом, множина дескрипторів перетворюється у єдиний вектор-прототип – репрезентативний опис класу. Це досягається шляхом нормалізації за правилом більшості: для кожної позиції у бінарному векторі підраховується кількість одиниць серед усіх дескрипторів класу, і якщо їх більшість – у відповідну позицію вносять 1, інакше – 0 [3, 19]. У результаті отримуємо вектор, який зберігає статистичні характеристики класу та дозволяє значно зменшити обчислювальну складність під час класифікації [37].

Цей етап виконується на основі навчальної вибірки (еталонних зображень) для кожного класу. Припустимо, ми маємо K класів (категорій) об’єктів і для кожного класу підготовлено еталонні зображення. Застосувавши перший етап до всіх еталонних зображень одного класу, ми отримаємо множину бінарних дескрипторів $D^c = d_i^c$, де індекс c позначає клас, а i пробігає всі дескриптори зображень цього класу.

Об’єднаємо дескриптори класу c у матрицю:

$$B^{(c)} = \begin{pmatrix} d_1^{(c)} \\ d_2^{(c)} \\ d_{n_c}^c \end{pmatrix}, \tag{2.2}$$

де $B^{(c)}$ – це матриця, яка містить усі бінарні дескриптори, отримані з еталонних зображень класу c;

$d_i^{(c)}$ – це бінарний дескриптор (вектор ознак) для і-го зображення (чи ключової точки) класу c;

n_c – це кількість таких дескрипторів, тобто кількість рядків у матриці $B^{(c)}$.

Звідси будується прототип класу $p_c \in \{0,1\}\hat{}m$ за правилом більшості: для кожної колонки матриці B^c обчислюється сума значень по рядках (кількість одиниць у даному стовпчику).

Якщо кількість одиниць перевищує половину кількості рядків > $\frac{n_c}{2}$, то відповідний біт прототипу встановлюється в 1, інакше – в 0. Формально, позначивши через $B_j^{(c)}$ j-тий стовпець матриці $B^{(c)}$, маємо:

$$p_j^c = \begin{cases} 1, \text{якщо} \sum_{i=1}^{n_c} B_{ij}^{(c)} > \frac{n_c}{2}, \\ 0, \text{інакше} \end{cases} \quad (2.3)$$

де $p_j^{(c)}$ – j-та компонента (біт) прототипу класу c;

$B_{ij}^{(c)}$ – значення j-го біта дескриптора, тобто елемент у і-го рядка j-го стовпчика матриці $B^{(c)}$;

n_c – кількість дескрипторів у класі с.

Отриманий вектор p_c є медоїдом бінарних описів усіх еталонних зображень класу с [37-40]. Аналогічно обчислюються прототипи $p_1, p_2, \ldots, p_K$ для кожного з К класів. Для подальшого використання в мережі Гемінга доцільно конвертувати бінарні прототипи у біполярну форму, замінивши 0 на -1, залишивши 1 без змін. Таким чином, кожен прототипний вектор представляємо як $p_c \in \{-1,1\}^m$.

2.3 Аналіз даних у мережі Гемінга

Після проведення попередніх двох етапів (навчання прототипів на еталонах) система готова класифікувати нові зображення.

Нехай подано невідоме зображення Z, для якого потрібно визначити клас. Спершу із зображення Z виділяється фіксований набір дескрипторів D_Z розміру $N_{sel} \times m$. Далі за тим самим алгоритмом більшості обчислюється узагальнений вектор опису об'єкта $v_Z \in \{0,1\}^m$ (тобто береться медіанний за бітами дескриптор з D_Z).

Після цього v_Z конвертується у біполярний вектор і подається на вхід мережі Гемінга (рис. 2.2).

Перший шар мережі Гемінга обчислює активації нейронів [25] для кожного класу $c = 1..K$ за формулою 1.3. Оскільки обидва вектори біполярні, скалярний добуток можна виразити через відстань Гемінга [31] d_c між відповідними бінарними векторами (до конвертації): виконується співвідношення $p_c \cdot x = m - 2d_c$.

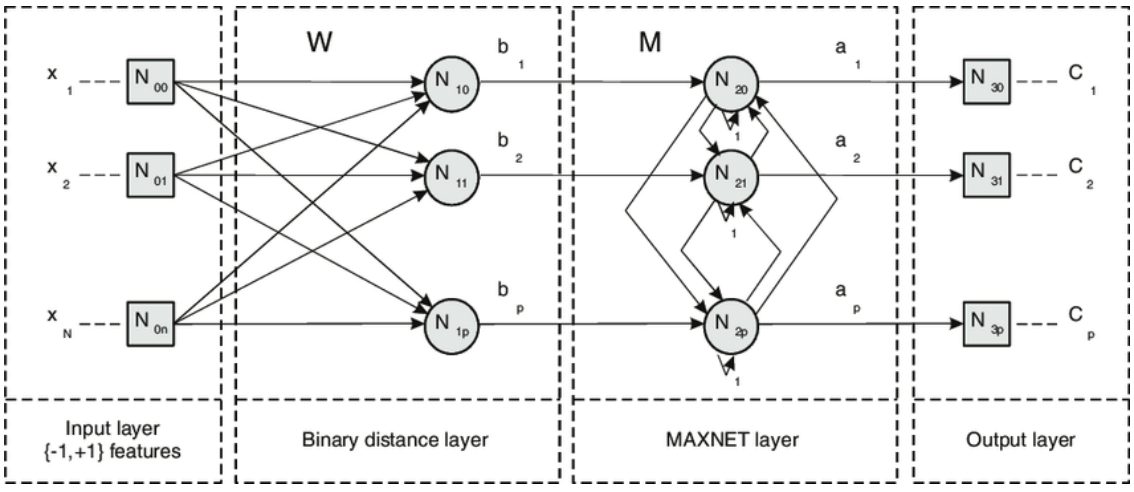


Рисунок 2.2 – Схема роботи двошарової мережі Гемінга

Тому активацію можна переписати як $a_c = m - d_c$. Таким чином, значення a_c чисельно рівне кількості біт, в яких опис Z збігається з

прототипом класу c (що еквівалентно $m - d_c$). Чим більша ця величина, тим ближче об'єкт до даного класу.

Другий шар, який за своєю суттю реалізує алгоритм латерального гальмування макснет отримує на вході вектор активацій $A = (a_1, a_2, \dots a_K)$. З початкового вектора $A^0 = A$ мережа ітеративно обчислює оновлені активації $A^{t+1} = f(A^t)$ за правилом, що описується формулою

$$a_i^{t+1} = \max\left\{a_i^t - \epsilon \sum_{j \neq i} a_j^t, 0\right\}, i = 1..K, \quad (2.4)$$

де a_i^t – активація i-го нейрона на t-ій ітерації;

a_i^{t+1} – активація i-го нейрона на t-ій ітерації;

ϵ – коефіцієнт гальмування (невелике додатне число, зазвичай від 0 до 1);

$\sum_{j \neq i} a_j^t$ – сума активацій усіх нейронів, крім i-го, на поточній ітерації;

K – загальна кількість нейронів (класів).

Ця формула моделює конкуренцію між нейронами. На кожній ітерації кожен нейрон зменшує свою активацію пропорційно до суми активацій усіх інших. Таким чином, нейрони з меншими активаціями швидше «гальмуються» й поступово згасають, залишаючи найсильніший нейрон (той, який відповідає найбільш релевантному класу) активним.

Алгоритм триває поки активації не збіжаться до стаціонарних значень [22].

У результаті вихідний вектор має єдиний ненульовий елемент [31] – припустимо, це a_w, відповідний класу w. Це означає, що прототип p_w виявився найбільш схожим до вхідного образу Z серед усіх. Отже, базове рішення класифікатора – віднести зображення Z до класу w.

2.4 Прийняття класифікаційного рішення

Для підвищення надійності класифікації вводимо перевірку на мінімальну прийнятну схожість. У рамках нашого підходу застосовано пороговий критерій на основі голосування дескрипторів. Ідея полягає в тому, щоб врахувати усі вихідні дескриптори D_Z при винесенні остаточного рішення. Зокрема, якщо дозволити кожному дескриптору з D_Z «проголосувати» за найближчий клас шляхом визначення класу з мінімальною Гемінговою відстанню до даного дескриптора, то отримаємо розподіл голосів між класами. Для нашого випадку, коли кожен клас описується одним прототипом, голосування зводиться до підрахунку, скільком дескрипторам об'єкта Z найближчим є прототип p_1, скільком – p_2 і т.д.

Нехай $votes(c)$ – кількість дескрипторів із Z, найбільш схожих на прототип класу c. Тоді можна обчислити коефіцієнт впевненості класифікації як відношення кількості голосів за переможця до загальної кількості голосів. В ідеалі, якщо всі дескриптори об'єкта узгоджено «голосують» за один клас, то коефіцієнт впевненості = 1. Якщо ж голоси розпорошені між класами, він буде нижчим. Вводимо порогове значення – мінімально допустиме значення коефіцієнта для прийняття рішення. Якщо значення коефіцієнта нижче порогового, результат класифікації вважається невпевненим і система повертає відповідь «не розпізнано».

Введення порогового коефіцієнта впевненості дозволяє не лише формалізувати процедуру прийняття рішення, а й зменшити кількість хибно позитивних спрацювань, що особливо важливо при роботі з неоднозначними або низькоякісними зображеннями. Такий підхід наближає класифікаційну систему до поведінки людини, яка, не будучи впевненою в упізнаванні, воліє утриматися від остаточного вердикту. Як зазначається в дослідженнях, системи з елементами відмови на основі статистичних критеріїв демонструють вищу загальну точність при розпізнаванні на основі багатьох класів [38].

Перевага підрахунку голосів за допомогою Гемінгової відстані полягає в тому, що дана метрика дуже ефективно працює з бінарними дескрипторами, оскільки дозволяє використовувати логічні операції для обчислень. Це забезпечує високу швидкодію класифікатора, що особливо важливо для застосування у реальному часі або вбудованих системах. До того ж, врахування всіх дескрипторів дозволяє побудувати більш стійку оцінку, знижуючи вплив випадкових чи шумових компонентів опису зображення.

Також розподіл голосів між класами часто має чітку локалізацію навколо одного чи двох найближчих прототипів, особливо при чіткому візуальному поданні об’єкта. У випадку, коли спостерігається розпорошення голосів, це може свідчити про низьку якість вхідного зображення, неоднозначність у межах класів або необхідність розширити навчальну вибірку для покриття нових варіантів. У таких випадках доцільним є повернення відповіді «не розпізнано» та потенційне перенаправлення зображення для повторного аналізу або до іншого класифікаційного модуля.

Варто зазначити, що запровадження механізму відмови не лише знижує помилки, а й дозволяє будувати багаторівневі системи класифікації. Наприклад, при низькому коефіцієнті впевненості можна запускати альтернативний метод – структурний аналіз на основі центрів, кластеризації або навіть глибоких нейронних мереж. Така модульна архітектура дозволяє обирати найдоцільніший алгоритм залежно від складності конкретного випадку.

Загалом запропонована система класифікації демонструє збалансоване поєднання ефективності, обчислювальної простоти та адаптивності. Вона здатна працювати із широким спектром зображень, при цьому забезпечуючи високу достовірність рішень та здатність до масштабування. Подальші експериментальні результати підтвердять практичну доцільність такого підходу в умовах варіативних вхідних даних.

Таким чином, математична модель класифікатора визначена. На основі цієї моделі реалізовано програмну систему, опис якої та експериментальні результати наведено у наступному розділі.

3 РЕЗУЛЬТАТИ КОМП'ЮТЕРНОГО МОДЕЛЮВАННЯ

Для перевірки працездатності розробленого підходу була створена програмна реалізація системи та проведено низку експериментів на тестових даних. У цьому розділі описано середовище реалізації, представлено тестову вибірку та сценарії експериментів, наведено отримані результати класифікації та їх аналіз.

3.1 Вибір середовища реалізації

Програмну реалізацію виконано мовою Python 3 (рис. 3.1) із використанням бібліотеки OpenCV для комп'ютерного бачення. Бібліотека OpenCV надає готову імплементацію алгоритму BRISK для детектування ключових точок і обчислення бінарних дескрипторів [53].

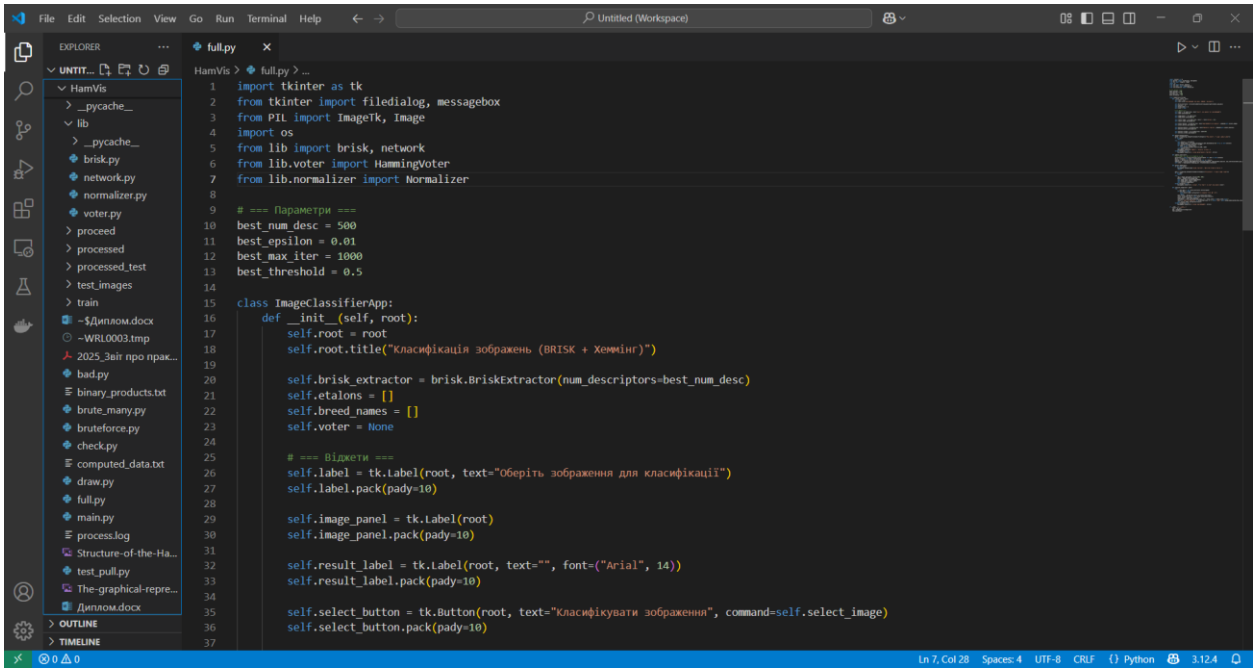


Рисунок 3.1 – Реалізація інтерфейсу користувача у середовищі VSCode

OpenCV – бібліотека з відкритим кодом, розроблена компанією Intel, що містить велику кількість функцій та алгоритмів комп'ютерного зору [53]. OpenCV налічує більше 2500 оптимізованих класичних та більш нових алгоритмів, включаючи детектор ключових точок BRISK. Функціонал OpenCV може бути використаний для завдань виявлення об'єктів, сегментації зображень, пошуку зображень на основі вмісту, аналізу руху та стеження за об'єктом, створення тривимірних моделей об'єктів, виконання перетворень зображень та відео [36].

Для роботи з масивами використано бібліотеку NumPy. NumPy – бібліотека з відкритим кодом для мови програмування Python, призначена для ефективної роботи з багатовимірними масивами та виконання чисельних обчислень [54]. Вона була створена шляхом об'єднання можливостей бібліотек Numeric і Numarray [53-54], а її розробку ініціював Тревіс Оліфант у 2005 році. NumPy забезпечує базову функціональність для обробки великих обсягів числових даних, включаючи засоби для маніпулювання масивами, математичних операцій і векторизації обчислень.

Функціонал NumPy може бути використаний для вирішення завдань лінійної алгебри, статистичного аналізу, обробки сигналів, інтерполяції та перетворення Фур'є. Бібліотека є основою для широкого кола інших інструментів у Python-екосистемі, таких як SciPy, Pandas, Matplotlib, TensorFlow та PyTorch, і залишається критично важливою для сучасних наукових досліджень, машинного навчання та обробки даних [54].

Графічний інтерфейс користувача було реалізовано з використанням модуля Tkinter – стандартної бібліотеки для створення віконних застосунків у Python. Tkinter надає набір віджетів (кнопок, міток, полів введення, меню тощо), які дозволяють швидко створювати інтуїтивно зрозумілі інтерфейси для взаємодії користувача з програмою. Він є обгорткою над бібліотекою Tcl/Tk і забезпечує кросплатформну сумісність, завдяки чому програми, написані з використанням Tkinter, однаково працюють на Windows, macOS і

Linux. Простота використання та інтеграція з основними модулями Python роблять його популярним вибором для прикладних проєктів.

3.2 Структура програми

Розроблений код структуровано у вигляді класів, що відповідають основним компонентам моделі (рис. 3.2).

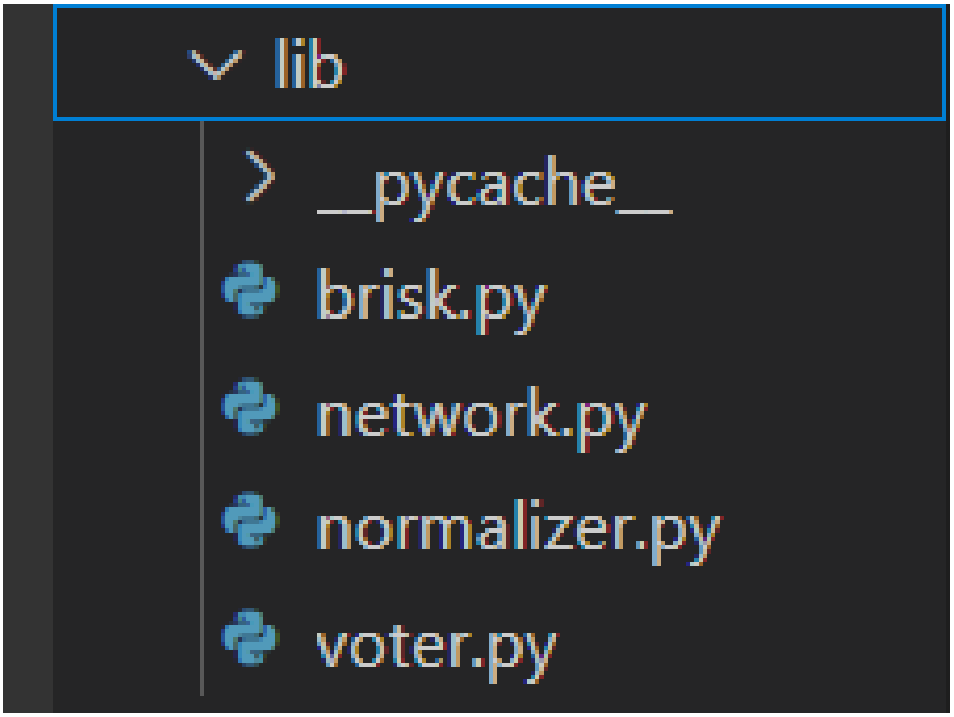


Рисунок 3.2 – Файли, що реалізують центральні компоненти моделі класифікатора, відображені у середовищі VSCode

Програмна реалізація організована за принципом модульності [42]: кожен етап обробки даних винесено у відповідний клас, що забезпечує високу гнучкість і полегшує підтримку коду. Основна логіка класифікації побудована як послідовність незалежних операцій: екстракція дескрипторів, їх агрегування, формування прототипів, розпізнавання через мережу Гемінга та механізм голосування з оцінкою впевненості.

3.2.1 Детектор ключових точок

BriskExtractor – клас для екстракції дескрипторів BRISK. При ініціалізації задаються параметри, такі як максимальна кількість дескрипторів num_descriptors (це відповідає N_{sel} у моделі) та режим доповнення pad_mode («zeros» для доповнення нульовими векторами або «average» для доповнення середнім дескриптором).

Метод extract(image) цього класу завантажує зображення, приводить його до відтінків сірого та виконує пошук ключових точок і обчислення дескрипторів. Далі сортує ключові точки сортуються за значущістю, після чого алгоритм вибирає перші num_descriptors дескрипторів і конвертує їх з формату OpenCV (масив байтів) у бінарний масив 0/1 шляхом операції np.unpackbits. Якщо дескрипторів менше, ніж треба, реалізовано доповнення (в режимі «zeros» додаються рядки з нулями, в режимі «average» – дублюється середній вектор, обчислений як побітове середнє). Повертається масив розміром $num\_descriptors \times m$, де $m = 512$ – розмір бінарного дескриптора BRISK.

3.2.2 Нормалізатор

Normalizer – клас, що реалізує статичні методи для агрегації множини дескрипторів у компактне та уніфіковане представлення. Основна функція класу полягає у формуванні узагальненого дескриптора (прототипу класу) за бінарними ознаками на основі правила більшості.

Метод matrix_to_vector(binary_matrix) приймає на вхід двовимірний масив, де кожен рядок є бінарним дескриптором, і повертає один вектор – медоїд [38-40], сформований шляхом побітового голосування: для кожної позиції бітів обирається те значення (0 або 1), яке зустрічається частіше. Такий підхід забезпечує зниження чутливості до шуму або незначних варіацій у

вхідних зображеннях. Отриманий вектор виступає репрезентативним описом [41, 42] усього класу, зберігаючи при цьому бітову структуру, сумісну з ефективними метриками, як-от відстань Гемінга.

Метод vector_to_polarized(*binary_vector*) здійснює перетворення бінарного вектора у біполярний формат [43], який необхідний для обчислень у мережі Гемінга. Зокрема, значення 0 замінюється на -1, а 1 залишається незмінним. Це дає змогу зручно застосовувати скалярні добутки для оцінки подібності, що прямо відповідає математичному опису активацій першого шару нейронної мережі [44, 45].

3.2.3 Мережа-класифікатор

HammingNetwork – клас, що реалізує нейронну мережу Гемінга. При створенні йому передається список прототипів [46] (вектори ±1 для кожного класу), а також параметри ϵ (коефіцієнт гальмування) та max_iterations (максимальна кількість ітерацій макснету).

У даному класі метод _first_layer(input_vector) обчислює активації першого шару за формулою $a = \frac{n + p \cdot x}{2}$ (в коді: обчислення скалярного добутку між матрицею прототипів та вектором).

Лістинг 3.1. Алгоритм обчислення активацій

```
x = np.array(input_vector)
dot_products = np.dot(self.P, x)
activations = self.n / 2 + 0.5 * dot_products
```

Метод _maxnet(activations) реалізує процес латерального гальмування: у циклі оновлює активації за формулою 1.4. Цикл переривається, якщо зміни активацій стали меншими за поріг або перевищено кількість ітерацій.

Лістинг 3.2. Алгоритм рекурентного латерального гальмування (maxnet).

```
y = activations.copy()
for iteration in range(self.max_iterations):
    total = np.sum(y)
    new_y = y - self.epsilon * (total - y)
    new_y = np.maximum(new_y, 0)
    if np.allclose(new_y, y, atol=1e-6):
        break
y = new_y
return y
```

Нарешті, метод recognize(input_vector) комбінує ці кроки: він приймає біполярний вектор ознак, отримує активації першого шару, пропускає їх через _maxnet() і визначає індекс переможця як np.argmax() фінальних активацій. Повертається індекс найбільш схожого прототипу [54] та вектор активацій після завершення роботи мережі.

3.2.4 Система голосування

HammingVoter – клас-обгортка для реалізації механізму голосування. Він приймає об’єкт HammingNetwork та параметр порогу *threshold*. Метод vote(*binary_vectors*) дозволяє подати список декількох бінарних векторів на вхід мережі Гемінга. Кожен з них послідовно класифікується методом recognize(), і підраховується кількість голосів (кількість випадків, коли даний клас отримав перемогу).

У результаті метод повертає словник з лічильниками голосів по класах та визначається переможець (клас з максимумом голосів). Додатково

обчислюється коефіцієнт впевненості як частка голосів переможця [47, 48]. Якщо ця частка менша за порогове значення *threshold*, то рішенням класифікатора буде відмова від розпізнавання (присвоєння спеціального індексу, -1).

3.2.5 Графічний інтерфейс

Історично концепція GUI – Graphic User Interface (графічний інтерфейс користувача) виникла як спроба зробити взаємодію з комп'ютерами доступнішою для нефахівців, що не володіли навичками командного рядка. Починаючи з 1980-х років, коли компанії Xerox, Apple та пізніше Microsoft представили перші ОС з віконним інтерфейсом, GUI став стандартом у взаємодії людини з машиною. З того часу ідея керування програмами за допомогою візуальних елементів – кнопок, меню, вікон — набула масового поширення. Сучасні GUI-фреймворки, як-от Tkinter, є спадкоємцями цієї філософії, що поєднує зручність, інтуїтивність та швидкість взаємодії. У контексті наукових або освітніх проєктів це дозволяє зосередитися на логіці роботи алгоритму, водночас надаючи користувачу зручний спосіб запуску та перегляду результатів без потреби взаємодіяти з кодом напряму.

Інтерфейс було реалізовано з використанням модуля Tkinter – стандартної бібліотеки для створення віконних застосунків у Python. Tkinter надає набір віджетів (кнопок, міток, полів введення, меню тощо), які дозволяють швидко створювати інтуїтивно зрозумілі інтерфейси для взаємодії користувача з програмою. Він є обгорткою над бібліотекою Tcl/Tk і забезпечує кросплатформну сумісність, завдяки чому програми, написані з використанням Tkinter, однаково працюють на Windows, macOS і Linux. Простота використання та інтеграція з основними модулями Python роблять його популярним вибором для освітніх та прикладних проєктів, де потрібно

забезпечити базову візуалізацію процесів, відображення результатів класифікації або запуск алгоритмів із кнопок інтерфейсу.

3.3 Інструкція користувача

Дана програма є прикладом інтелектуальної системи обробки та аналізу зображень, яка реалізує структурний метод класифікації [49-51] на основі бінарних дескрипторів ключових точок. Основна ідея методу полягає у тому, що кожне зображення описується набором ключових точок, які детектор BRISK виділяє за певними критеріями [49]. Далі, дескриптори цих точок використовуються для побудови узагальненого представлення класу зображень (еталонів), а мережа Гемінга з латеральним гальмуванням виконує класифікацію нових зразків за цими представленнями.

Інтерфейс користувача реалізовано засобами бібліотеки Tkinter, що забезпечує простоту у використанні та достатню гнучкість для інтеграції з основним обчислювальним модулем. Програма створена мовою програмування Python, що забезпечує широку сумісність з операційними системами Windows, Linux і macOS.

3.3.1 Вимоги до середовища

Для повноцінної роботи програми необхідно встановити Python версії 3.7 або новішої. Крім того, використовуючи пакетний менеджер pip – Pip Install Python (pip встановлює python) слід попередньо інсталювати такі бібліотеки:

- OpenCV;
- Numpy;
- Pillow [57].

Для цього необхідно скористатися командним рядком та введіть команду pip install opencv-python-headless numpy pillow

Бібліотека opencv-python-headless використовується замість стандартної opencv-python, оскільки вона не потребує графічного інтерфейсу операційної системи і є більш стабільною у середовищах без дисплея.

3.3.2 Запуск програми

Щоб розпочати роботу із застосунком, необхідно відкрити термінал (Command Prompt, PowerShell, Terminal тощо) та перейти до директорії, де збережено файл програми hamming_gui.py. Запуск здійснюється наступною командою python hamming_gui.py

Після виконання команди відкриється графічне вікно, у якому користувач зможе взаємодіяти з функціоналом класифікації.

3.3.3 Опис інтерфейсу користувача

Після запуску програми користувач бачить просте, інтуїтивно зрозуміле вікно, яке містить низку елементів управління (рис. 3.3):

– кнопка «Встановити еталони» – дозволяє завантажити зображення, які будуть використовуватись як зразки для побудови моделі, які повинні належати до певних класів (наприклад, «Beagle.jpg», «Bulldog.jpg» тощо), причому назва файлу без розширення вважається назвою класу;

– список еталонів – після додавання еталонів, у вікні з’являється перелік їхніх назв, що дозволяє переконатися, що файли завантажені правильно;

– кнопка «Класифікувати зображення» – відкриває діалог вибору файлу, після чого система виконує класифікацію на основі побудованої моделі.

Результат класифікації відображається текстом під зображенням і містить як назву передбаченого класу, так і рівень впевненості у відсотках.

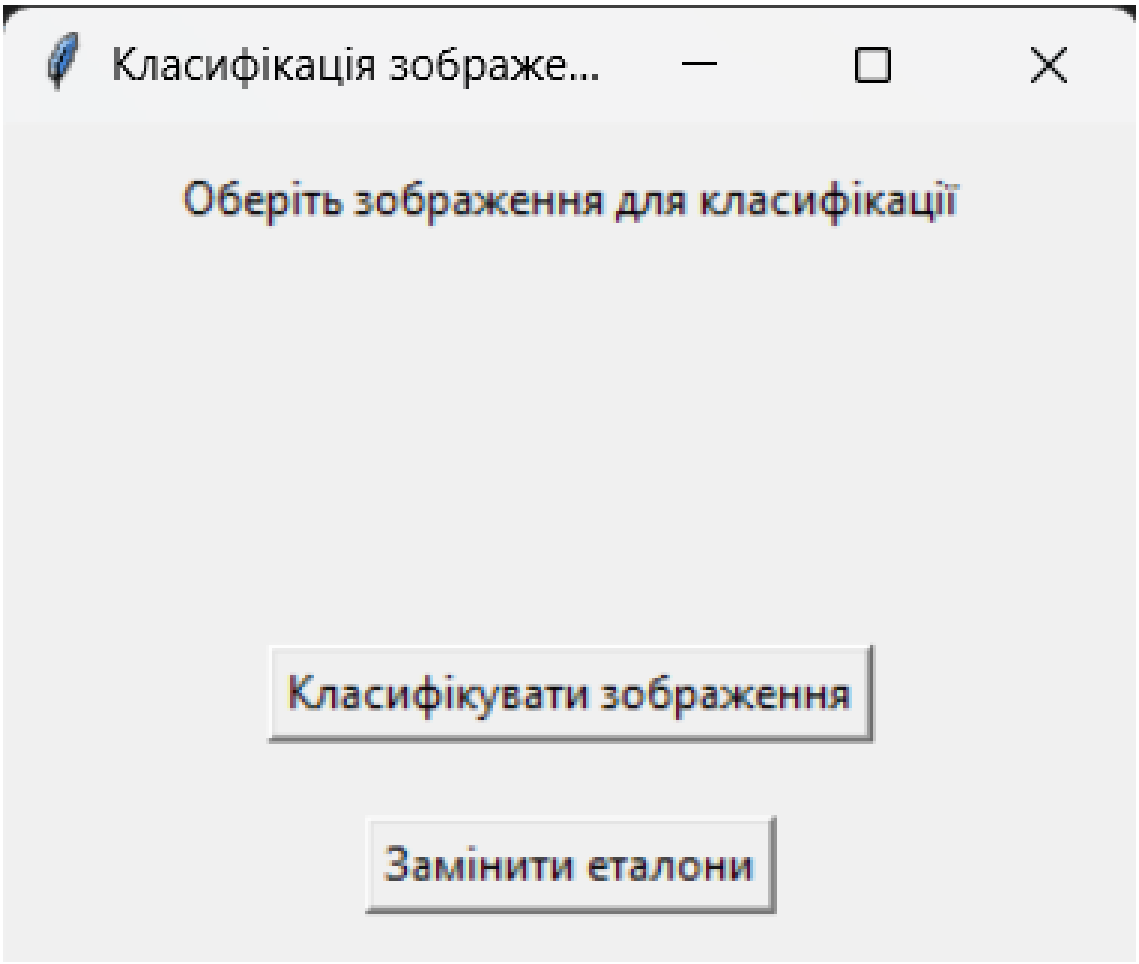


Рисунок 3.3 – Інтерфейс користувача

3.3.4 Завантаження еталонів

Однією з важливих особливостей програми є можливість самостійного налаштування еталонної бази. Це дозволяє адаптувати систему під конкретні потреби користувача. Для цього слід натиснути кнопку «Замінити еталони» та

обрати одне або кілька зображень (рис. 3.4), які репрезентують класи, з якими буде проводитись подальше порівняння.

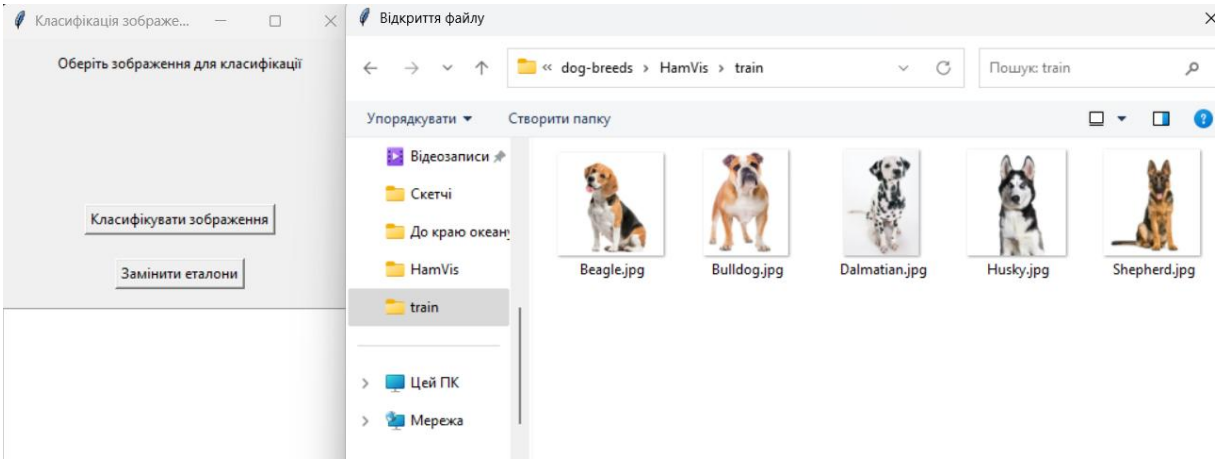


Рисунок 3.4 – Діалогове вікно завантаження еталонів

Програма автоматично обробляє зображення, виділяє ключові точки та будує дескриптори. Ці дескриптори агрегуються у єдиний прототип кожного класу – так званий медоїд, який виступає еталонним представником.

Такий механізм формування еталонів надає користувачу гнучкість у створенні та оновленні бази знань класифікатора. Це особливо корисно в умовах змінного набору класів або при переході до нової предметної області. Наприклад, у навчальному процесі можна швидко замінити еталони для демонстрації іншої категорії зображень без переписування коду. Водночас, оскільки еталонні прототипи формуються автоматично з реальних зображень, важливо дотримуватись рекомендацій щодо якості, контрастності та розміру вхідних файлів – це позитивно впливає на точність подальшої класифікації.

Особливу увагу необхідно звернути на розширення файлів. Усі файли повинні мати однаковий формат (наприклад, .jpg, .png) і бажано, щоб назви не містили спеціальних символів, окрім літер латиниці, цифр, підкреслення або дефісів.

3.3.5 Класифікація зображення

Після завантаження еталонів користувач може перейти до основної функції програми – класифікації зображення. Натискаючи кнопку «Класифікувати зображення», користувач обирає файл, який підлягає аналізу. Після цього програма:

– завантажує зображення та конвертує його у відтінки сірого;

– виділяє ключові точки BRISK та будує бінарні дескриптори;

– формує бінарний вектор більшості (медоїд) та переводить його у біполярну форму;

– передає отриманий вектор у мережу Гемінга, яка розраховує активації та визначає найбільш релевантний клас;

– повертає результат у текстовому вигляді (назва класу та відсоток впевненості) (рис. 3.5).

У випадку, якщо жоден з класів не набрав достатньої кількості голосів або впевненість нижча за порогове значення, буде виведено повідомлення «Клас не визначено».

Описана функціональність забезпечує користувачу повний цикл взаємодії з програмою – від завантаження зображення до отримання результату класифікації у зручному, зрозумілому вигляді. Такий підхід реалізує принцип інтерактивного аналізу, де ключові етапи обробки приховані за простим натисканням кнопки. Уся складна внутрішня логіка (обчислення дескрипторів, побудова векторів, активація мережі) відбувається автоматично, без необхідності технічного втручання з боку користувача.

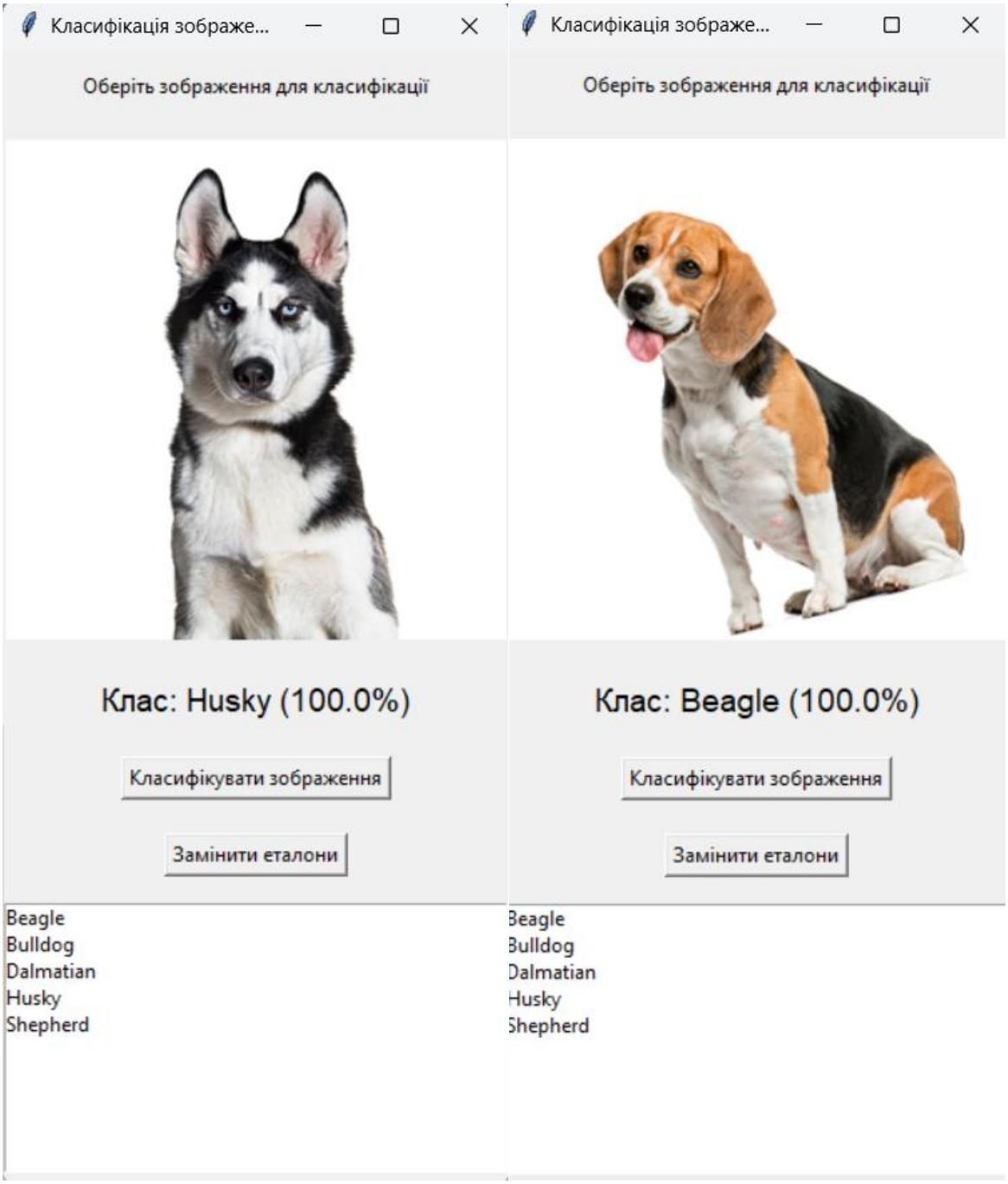


Рисунок 3.5 – Приклади класифікації зображень

3.3.6 Рекомендації щодо використання

Для кращої точності рекомендовано обрати чіткі зображення з фокусом на об’єкті (наприклад, на морді собаки).

Бажано використати зображення приблизно одного масштабу (300–600 пікселів шириною).

Необхідно уникати зображень із великою кількістю фону або сторонніх об’єктів.

Якщо планується багаторазове використання програми, бажано зберегти еталонний набір зображень у спеціальну директорію з латинською назвою.

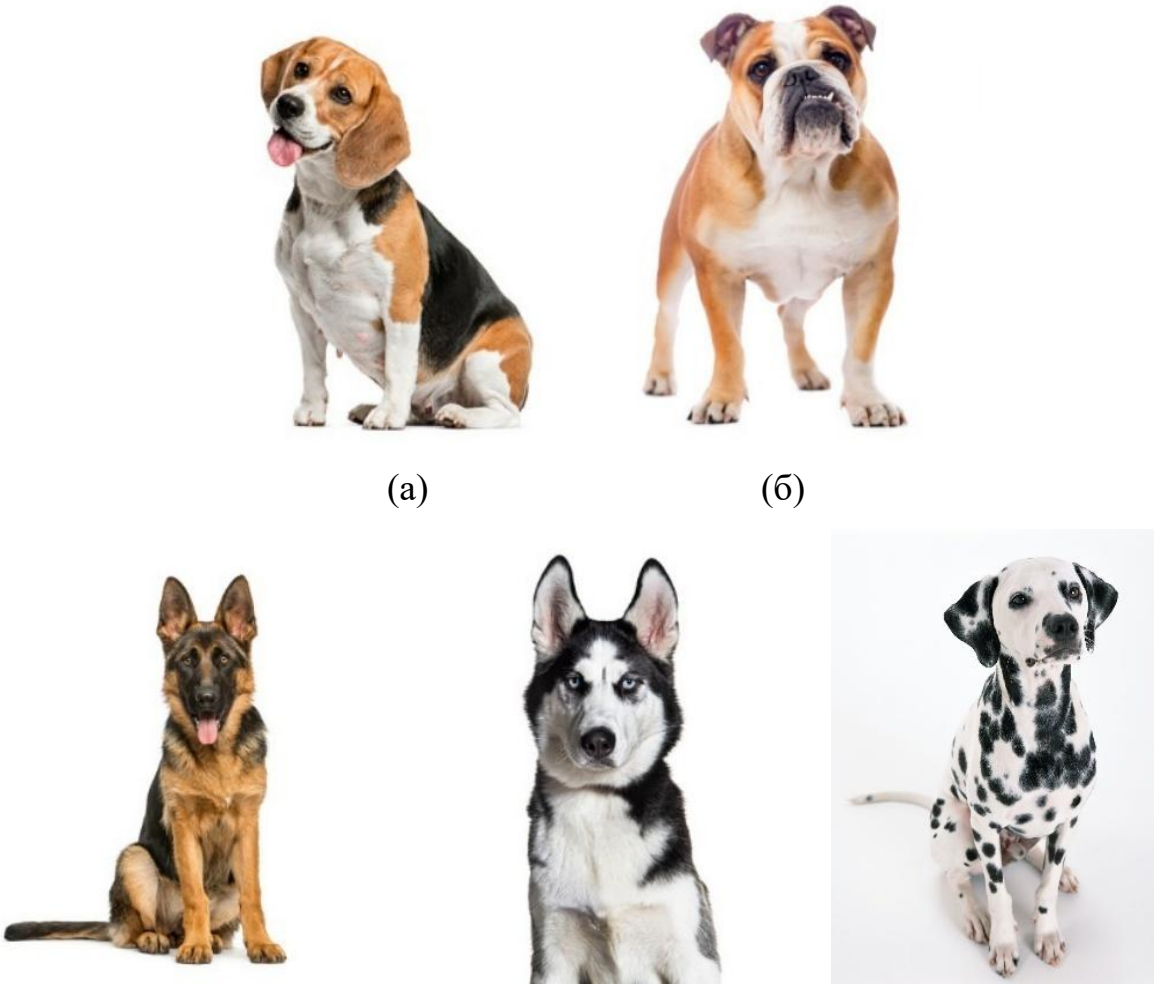
Таблиця 3.1 – Найбільш поширені проблеми та способи їхнього вирішення

Повідомлення	**Можлива причина**	**Спосіб вирішення**
Не знайдено дескрипторів	Порожнє, розмите або одноколірне зображення	Використайте чіткіше або інше зображення
Клас не визначено	Низька впевненість моделі (менше 50%)	Збільште якість зображення, додайте більше еталонів
Не вдалося завантажити зображення	Пошкоджений або не підтримуваний файл	Відкрийте зображення у графічному переглядачі та переконайтесь, що воно справне

Розроблений застосунок є прикладом простої у використанні системи комп’ютерного зору, яка поєднує ефективні алгоритми обробки зображень та нейромережеві підходи для класифікації. Завдяки підтримці гнучкої структури даних, підтримці кирилиці та інтуїтивному інтерфейсу, програма може бути використана як у навчальних цілях, так і для створення базових рішень у задачах візуальної ідентифікації.

3.4 Опис тестових даних

Для експериментальної перевірки було обрано задачу класифікації зображень собак за породами. Розглянуто K=5 класів: бігль (Beagle), бульдог (Bulldog), далматин (Dalmatian), хаскі (Husky) та німецька вівчарка (Shepherd). Для кожного класу було підготовлено еталонне зображення, як прототип (рис. 3.6).



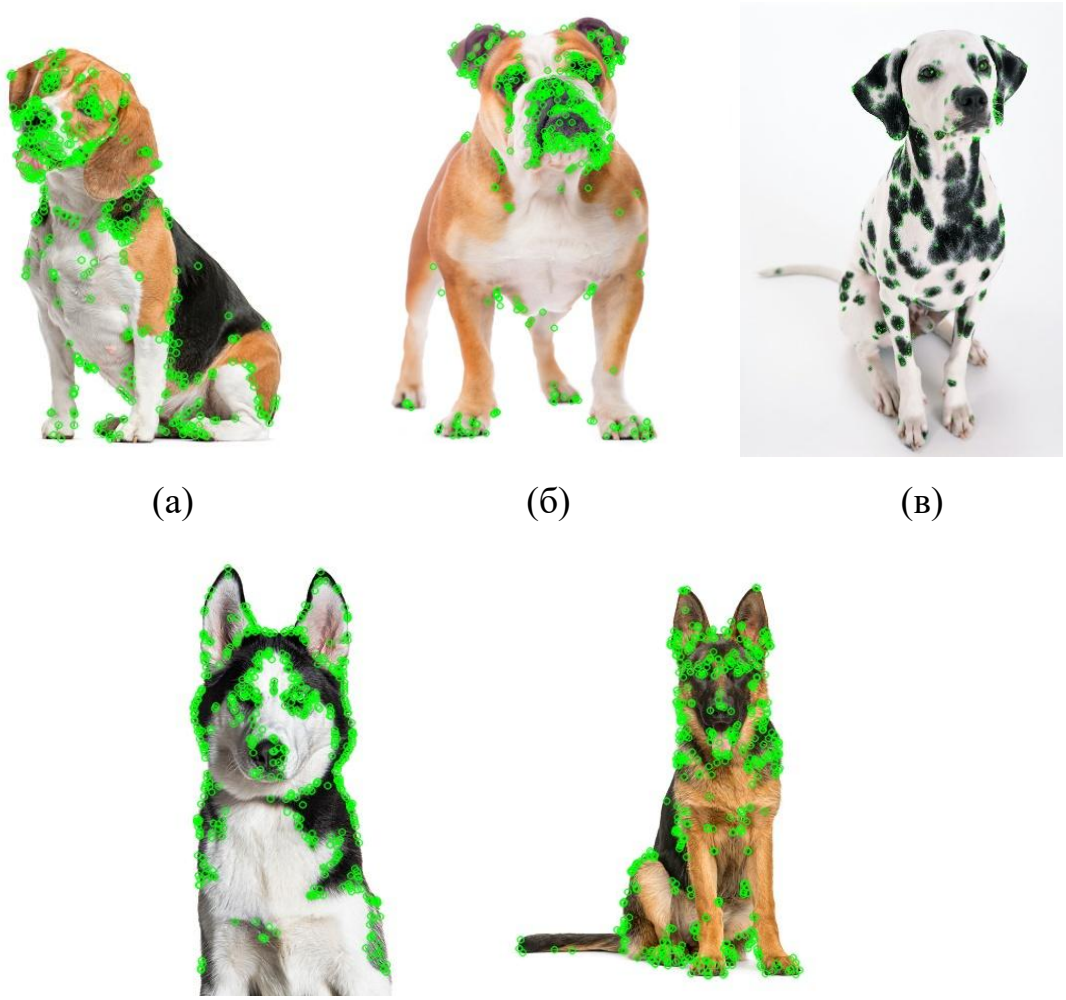
(а) (б)

(в) (г) (д)

Рисунок 3.6 – Еталонні зображення порід собак: (а) бігль; (б) бульдог; (в) вівчарка; (г) хаскі, (д) долматин

Таким чином, мережа Гемінга оперувала п’ятьма окремими прототипними векторами (тими, що відображають еталони заданих класів),

кожен довжиною $m = 512$ бітів (рис. 3.7) після їхнього перетворення із бінарних, тобто таких що представлені у двійковому вигляді у біполярні, тобто такі, що мають у складі лише додатну або від'ємну одиницю (±1).



(а) (б) (в)

Рисунок 3.7 – Еталонні зображення із ключовими точками: (а) бігль; (б) бульдог; (в) вівчарка; (г) хаскі, (д) долматин

Ця конвертація є важливим та необхідним етапом перед подачею даних у мережу Гемінга, оскільки її архітектура побудована на основі обробки сигналів, подібної до нейронів, де обчислення скалярного добутку між вхідним вектором та кожним прототипом (представником класу) є важливим та ключовим механізмом для визначення ступеня відповідності.

У таких обчисленнях використання знакових значень (+1 або -1), тобто біполярних векторів, значно полегшує інтерпретацію результатів: збіг

компонентів додає до суми, тоді як розбіжність – її зменшує. У бінарному представленні (0/1) таке розрізнення менш очевидне, а обчислення менш чутливі до відмінностей.

Завдяки цьому підходу мережа Гемінга може ефективно виконувати масове паралельне обчислення активацій для всіх класів, де величина скалярного добутку фактично відображає ступінь подібності між тестовим зображенням і прототипом класу.

Параметри алгоритму було встановлено експериментально: кількість дескрипторів на зображення N_{sel}=500, коефіцієнт гальмування ϵ=0.01 і максимальна кількість ітерацій макснету 1000, поріг впевненості 0.5 (50%).

Для тестування побудовано набір зображень, що моделюють різні можливі спотворення об’єктів у реальних умовах. Зокрема, для кожного класу взято контрольне зображення – сам еталон. До кожного зображення застосовано ряд геометричних трансформацій, а саме:

– поворот 15˚ навколо центру зображення за годинниковою стрілкою;

– поворот 15˚ навколо центру зображення проти годинникової стрілки;

– масштабування 0.85, тобто зменшення розмірів зображення на 15% по обох осях;

– масштабування 1.15, тобто збільшеня розмірів зображення на 15% по обох осях.

Таким чином для кожного з 5 базових контрольних зображень було згенеровано 4 додаткові модифіковані версії. Всього – 25 тестових випадків.

На кожному з цих зображень виконувалась процедура класифікації, описана в попередньому розділі. Фіксувались отримані відповіді системи – визначений клас або повідомлення про неможливість розпізнавання.

Нижче наведено оцінку точності класифікації (табл. 3.2-3.4).

Таблиця 3.2 – Зведена таблиця точності класифікації за породами

Класи	**Загальна кількість зображень**	**Кількість правильно розпізнаних зображень**	**Точність класифікації**
Beagle	5	5	1.0
Bulldog	5	5	1.0
Dalmatain	5	4	0.8
Husky	5	2	0.4
Shepherd	5	2	0.4
Всього	25	18	0.72

Таблиця 3.3 – Зведена таблиця точності класифікації за типами геометричних перетворень

Тип перетворення	**Загальна кількість зображень**	**Кількість правильно розпізнаних зображень**	**Точність класифікації**
Оригінал	5	5	1.0
Поворот +15	5	4	0.8
Поворот -15	5	4	0.8
Масштаб 0.85	5	2	0.4
Масштаб 1.15	5	3	0.6
Всього	25	18	0.72

До того ж було розроблено порівняльну характеристику на основі зіставлення досліджуваного методу із двома класичними методами розпізнавання зображень. Порівняння проводилося на предмет точності класифікації та швидкодії.

Перший класичний метод – порівняння на основі метрикою Евкліда. Схоже порівняння класів використовується у мережі Кохонена, однак вона менш придадтна для роботи із бінарними векторами. Для кожного зображення було виділено дескриптори, після чого сформовано медоїд за правилом більшості і обчислено Евклідову відстань до кожного еталону за формулою 1.1. Результати відстаней використовувались для визначення найближчого класу. Метод демонструє простоту реалізації та зрозумілу логіку прийняття рішення. Попри це, він є чутливим до невеликих змін у структурі векторів і може давати похибки при схожих значеннях.

Лістинг 3.3 – Алгоритм обчислення мінімальної відстані Евкліда

```
test_avg = majority_descriptor(test_desc)
    distances = [euclidean_distance(test_avg, class_desc) for class_desc in class_avg_descs]
    return np.argmin(distances), distances
```

Другий метод базується на метриці Гемінга, але замість повноцінної двошарової нейронної мережі з алгоритмом латерального гальмування, реалізує повний перебір. Для кожного зображення обчислено бінарні дескриптори. Після цього для кожного з дескрипторів проходить пошук серед еталонів такого дескриптора, який має найменшу відстань Гемінга. Отримані результати використовуються для підрахунку голосів за кожен клас. Метод вирізняється передбачуваністю поведінки та високою стабільністю у разі правильної підготовки еталонів. Його ефективність напряму залежить від кількості дескрипторів і глибини перебору.

Лістинг 3.4 – Алгоритм обчислення мінімальної відстані Гемінга шляхом повного пербору

```
for test_vec in test_desc:
    min_dist = float("inf")
    best_class = -1
    for class_idx, etalon_vectors in enumerate(etalon_descs):
        for etalon_vec in etalon_vectors:
            dist = hamming_distance(test_vec, etalon_vec)
            if dist < min_dist:
                min_dist = dist
                best_class = class_idx
    class_votes[best_class] += 1
predicted_class = np.argmax(class_votes)
return predicted_class, class_votes
```

Таблиця 3.4 – Порівняння методів класифікації

Алгоритм класифікації	**Кількість зображень**	**Точність, %**	**Швидкодія, мс**
Мережа Гемінга	25	72	2517
Метрика Евкліда	25	52	2950
Повний перебір	25	100	59831

3.5 Інтерпретація результатів

Результати експериментального дослідження свідчать про доцільність використання мережі Гемінга у задачах класифікації зображень за бінарними дескрипторами. Запропонована система показала високу точність (100%) на

еталонних зображеннях, що підтверджує коректність побудови моделі та ефективність механізму узагальнення ознак (медоїдів).

Разом з тим, за наявності геометричних трансформацій (поворотів, масштабування), точність класифікації зменшувалася до 40–80%, що вказує на обмежену інваріантність системи до змін масштабу та орієнтації. Найбільш чутливими до перетворень виявилися класи Husky та Shepherd, що може бути пов’язано з недостатньою відмінністю між їхніми дескрипторними представленнями або меншою кількістю ключових точок.

Порівняння з класичними підходами показало, що:

– мережа Гемінга забезпечує оптимальний баланс між точністю (72%) та швидкодією (приблизно 2.5 с);

– метрика Евкліда виявилась менш ефективною (52%), ймовірно, через неадекватність такої метрики для бінарного простору ознак [55, 56];

– повний перебір за відстанню Гемінга дає найвищу точність (100%), але за рахунок значно більшого часу обробки (близько 60 с), що обмежує його практичне застосування у реальному часі.

Таким чином, мережа Гемінга із системою агрегування дескрипторів на базі медоїда є ефективним компромісом між точністю, швидкістю та складністю реалізації. Вона демонструє хорошу масштабованість і може бути корисною для задач з обмеженими ресурсами. Водночас подальше підвищення точності потребує покращення інваріантності до спотворень, наприклад з використанням адаптивного навчання прототипів чи комбінацією з іншими алгоритмами виділення ключових точок.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було розроблено та досліджено інтелектуальну систему класифікації зображень, яка поєднує в собі ефективні алгоритми комп'ютерного зору (зокрема BRISK-дескриптори) з простим, але інформативним нейромережевим підходом – мережею Гемінга з латеральним гальмуванням. Проведене дослідження дозволяє сформулювати низку ключових висновків щодо теоретичних засад, технічної реалізації, експериментальних результатів і перспектив використання запропонованого методу.

Запропонована методика базується на поєднанні локальних бінарних ознак з механізмами агрегації та конкурентного навчання. Такий підхід не лише забезпечує прийнятну обчислювальну ефективність, а й дозволяє зберегти інтерпретованість процесу прийняття рішень, що є перевагою у порівнянні з глибокими нейронними мережами. Використання BRISK-дескрипторів як джерела ознак дозволяє досягати стійкості до масштабування та обертання, водночас зберігаючи компактність векторів ознак завдяки бінарному представленню. Агрегування дескрипторів у вигляді медоїдів спрощує структуру представлення класів і дозволяє ефективно зменшити обсяг даних без суттєвих втрат точності.

Мережа Гемінга, доповнена латеральним гальмуванням, забезпечує механізм змагального вибору найбільш релевантного класу. Такий двошаровий підхід дозволяє уникати неоднозначних відповідей у класифікації, знижуючи ризик помилкового вибору за умов наближених ознак у кількох класах.

Система була реалізована у вигляді програмного застосунку з графічним інтерфейсом. Було проведено серію експериментів з класифікації зображень порід собак, включаючи модифіковані версії зображень із поворотами та масштабуванням. Результати показали високу точність (100%) на еталонних або контрольних зображеннях, середню точність (72%) по всій вибірці,

включно із трансформованими версіями та зниження точності у випадках сильніших змін масштабу (від 40% до 60%).

Найбільші похибки виявлено в класах з менш стабільними ключовими точками або схожими візуальними рисами (наприклад, Husky і Shepherd). Ці результати дозволяють зробити висновок про чутливість методу до геометричних змін, але також підкреслюють потенціал подальшого підвищення стійкості шляхом розширення бази еталонів.

Запропонований підхід був зіставлений із двома класичними методами.

Класифікація за евклідовою відстанню показала значно нижчу точність (52%) навіть при аналогічній швидкодії, що підтверджує обмеженість застосування метричних підходів у бінарному просторі ознак.

Повний перебір із підрахунком Гемінгової відстані забезпечив 100% точність, але за рахунок суттєвого зростання часу обробки (~60 секунд на зображення), що робить його непридатним для систем реального часу.

Таким чином, мережа Гемінга забезпечує компроміс між точністю та продуктивністю, дозволяючи досягти задовільних результатів з порівняно низькими обчислювальними витратами.

На основі проведеного аналізу можна виділити такі сильні сторони системи:

– простота реалізації — відсутність необхідності у складному навчанні або великому обсязі даних;

– інтерпретованість — чітка логіка класифікації на основі метрик та голосування;

– швидкодія — придатність до роботи у режимі реального часу при невеликій кількості класів;

– гнучкість — можливість оновлення бази еталонів без повного перенавчання системи;

– висока точність на еталонах — доводить потенціал системи у контрольованих умовах.

Однак, попри загальну ефективність, система має певні обмеження, які варто враховувати:

– зниження точності при складних трансформаціях або зашумлених зображеннях;

– неможливість обробки зображень з великою варіативністю всередині класу без розширення кількості еталонів;

– фіксована кількість дескрипторів може призводити до втрати важливої інформації при складних зображеннях;

– відсутність навчання у класичному розумінні (градієнтного чи адаптивного) зменшує здатність до генералізації;

Кваліфікаційна робота продемонструвала, що поєднання бінарних дескрипторів та мережі Гемінга є перспективним напрямом для створення легких та пояснюваних систем класифікації. В умовах сучасної потреби у швидких, автономних і прозорих алгоритмах така архітектура виглядає доцільною, особливо для задач з обмеженими ресурсами. Подальші дослідження можуть бути спрямовані на покращення адаптивності системи та її здатності до навчання на потокових даних.

Результати роботи апробовано у формі тез доповіді під час Міжнародного молодіжного форуму ХНУРЕ [59].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT Press.

2. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems, 25, 1097–1105.

3. Gorokhovatsky, V. O., Stiahlik, N. I., Mazur, Ye. V., & Vechirska, A. D. (2024). Способи метричної грануляції для опису зображень у задачі класифікації [Methods of metric granulation for image description in classification tasks]. Control, Navigation and Communication Systems, (3), 106–112.

4. Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 580–587.

5. Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 3431–3440.

6. Cao, Z., Hidalgo, G., Simon, T., Wei, S.-E., & Sheikh, Y. (2017). Realtime multi-person 2D pose estimation using part affinity fields. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 7291–7299.

7. Szeliski, R. (2010). Computer vision: Algorithms and applications. Springer.

8. Mur-Artal, R., Montiel, J. M. M., & Tardós, J. D. (2015). ORB-SLAM: A versatile and accurate monocular SLAM system. IEEE Transactions on Robotics, 31(5), 1147–1163.

9. Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P. H., Buchatskaya, E., Doersch, C., Pires, B. A., Guo, Z., Azar, M. G., Piot, B., Kavukcuoglu, K., Munos, R., & Valko, M. (2020). Bootstrap your own latent: A

new approach to self-supervised learning. Advances in Neural Information Processing Systems, 33, 21271–21284.

10. Гороховатський, В. О. (2014). Структурний аналіз і інтелектуальна обробка даних у комп’ютерному зорі.

11. Гороховатський В.О., Гадецька С.В., Стяглик Н.І. (2019) Вивчення статистичних властивостей моделі блочного подання для множини дескрипторів ключових точок зображень. Радіоелектроніка, інформатика, управління, №2, 100–107. https://doi.org/10.15588/1607-3274-2019-2-11

12. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., and Zeghid M. (2024) Improving the effectiveness of image classification structural methods by compressing the description according to the information content criterion, Computers, Materials & Continua, vol. 80, no. 2, 3085-3106.

13. Gorokhovatskyi V., Tvoroshenko I., Yakovleva O., and Hudáková M. (2025) Image description compression in classification structural methods, IEEE Access, vol. 13, pp. 43631-43641, doi: 10.1109/ACCESS.2025.3548910.

14. Leutenegger, S., Chli, M., & Siegwart, R. Y. (2011). BRISK: Binary robust invariant scalable keypoints. Proceedings of the IEEE International Conference on Computer Vision, 2548–2555.

15. Calonder, M., Lepetit, V., Strecha, C., & Fua, P. (2010). BRIEF: Binary robust independent elementary features. European Conference on Computer Vision, 778–792.

16. Rublee, R., Rabaud, V., Konolige, K., & Bradski, G. (2011). ORB: An efficient alternative to SIFT or SURF. In Proceedings of the IEEE International Conference on Computer Vision (pp. 2564–2571).

17. Jégou, F., Douze, M., Schmid, C., & Pérez, P. (2010). Aggregating local descriptors into a compact image representation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 3304–3311). https://doi.org/10.1109/CVPR.2010.5540039

18. Rosten, E., & Drummond, T. (2006). Machine learning for high-speed corner detection. In Proceedings of the European Conference on Computer Vision (pp. 430–443).

19. Gorokhovatskyi, O., & Yakovleva, O. (2024). Medoids as a packing of orb image descriptors. Advanced Information Systems, 8(2), 5–11. https://doi.org/10.20998/2522-9052.2024.2.01

20. Jain, A. K., & Dubes, R. C. (1988). Algorithms for clustering data. Prentice Hall.

21. Gorokhovatskyi, V., Chmutov, Y., Tvoroshenko, I., & Kobylin, O. (2025). Reducing computational costs by compressing the structural description in image classification methods. Advanced Information Systems, 9(1), 5–12. https://doi.org/10.20998/2522-9052.2025.1.01

22. Nishikawa, K., & Yajima, H. (1999). Hamming neural networks: Fast pattern classification based on Hamming distance. Neural Networks, 12(8), 1375–1383.

23. Bishop, C. M. (2006). Pattern recognition and machine learning. Springer.

24. Dubuisson, M. P., & Jain, A. K. (1994). A modified Hausdorff distance for object matching. IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(2), 182–187.

25. Abu-Mostafa, Y. S., & St Jacques, J. (1985). Performance of Hamming networks in pattern classification. IEEE Transactions on Neural Networks, 1(2), 286–292.

26. Rogers, D., & Tanimoto, T. T. (1960). A computer program for classification of plants. Science, 132(3434), 1115–1118.

27. Carpenter, G. A., & Grossberg, S. (1987). A massively parallel architecture for a self-organizing neural pattern recognition machine. Computer Vision, Graphics, and Image Processing, 37(1), 54–115.

28. Gorokhovatskyi V., Tvoroshenko I., Yakovleva O., Hudáková M., and Gorokhovatskyi O. (2024) Application a committee of Kohonen neural networks to

training of image classifier based on description of descriptors set, IEEE Access, vol. 12, 73376-73385.

29. Mezher, L. S., & Hasan, A. A. (2021). Hamming neural network application with FPGA device. International Journal of Reconfigurable and Embedded Systems, 10(1), 37–46.

30. Dmitrienko, V., Kalyagin, A., Dmitrienko, A., & Limonova, E. E. (2021). New neural networks for the affinity functions of binary images with bipolar coding. Advances in Science, Technology and Engineering Systems Journal, 6(4), 91–99.

31. Gayler, T. W. (2022). A survey on hyperdimensional computing aka vector symbolic architectures. ACM Computing Surveys, 55(2), Article 39.

32. Taheri, R., Ghahramani, M., Javidan, R., Shojafar, M., Pooranian, Z., & Conti, M. (2019). Similarity-based Android malware detection using Hamming distance of static binary features.

33. Klimo, M., Lukáč, P., & Tarábek, P. (2021). Deep neural networks classification via binary error-detecting output codes. Applied Sciences, 11(8), 3563. https://doi.org/10.3390/app11083563

34. Mikolajczyk, K., & Schmid, C. (2005). A performance evaluation of local descriptors. IEEE Transactions on Pattern Analysis and Machine Intelligence, 27(10), 1615–1630.

35. Gorokhovatskyi V., Tvoroshenko I. (2024) An effective method for transforming an image description into a compact vector for classification. Information Technology and Implementation (Satellite): Conference Proceedings, November 21, 2024, Kyiv, Ukraine / V. Snytyuk (Editor). – Kyiv: Publishing House «Caravela», 25-28. https://openarchive.nure.ua/handle/document/29476

36. Bradski, G., & Kaehler, A. (2008). Learning OpenCV: Computer vision with the OpenCV library. O'Reilly Media.

37. Park, H.-S., & Jun, C.-H. (2009). A simple and fast algorithm for K-medoids clustering. Expert Systems with Applications, 36(2), 3336–3341.

38. Everitt, B. S., Landau, S., Leese, M., & Stahl, D. (2011). Cluster analysis (5th ed.). Wiley.

39. Xu, R., & Wunsch, D. (2005). A survey of clustering algorithms. IEEE Transactions on Neural Networks, 16(3), 645–678.

40. Aggarwal, C. C., & Reddy, C. K. (2014). Concepts of visual and interactive clustering. In Data Clustering: Algorithms and Application (pp. 483–500). CRC Press.

41. Gorokhovatskyi, V., Gadetska, S., Stiahlyk, N. (2024) Classification of images based on distance assessment. Information Technology and Implementation (Satellite): Conference Proceedings, November 21, 2024, Kyiv, Ukraine / V. Snytyuk (Editor). – Kyiv: Publishing House «Caravela», 22-24. https://openarchive.nure.ua/handle/document/29478

42. Pupchenko, D., Gorokhovatskyi, V. (2024) Accelerated filtration of ultrasound images. Information Technology and Implementation (Satellite): Conference Proceedings, November 21, 2024, Kyiv, Ukraine / V. Snytyuk (Editor). – Kyiv: Publishing House «Caravela», 69-72. https://openarchive.nure.ua/handle/document/29492

43. V. Gorokhovatsky, Y. Putyatin and V. Stolyarov (2017) Research of Effectivenss of Structural Image Classification Methods using Cluster Data Model, Radio Electronics Computer Science Control, vol. 3, no. 42, 78-85.

44. Gorokhovatsky, V.A., Putyatin, Y.P. (2008) Structural recognition of images on the basis of voting models of attributes of typical points, Data recording, storage and processing, 10(4), 75-85.

45. Gorokhovatskyi V.A., Zamula A.A. (2016) Employment of Intelligent Technologies in Multiparametric Control Systems. Telecommunications and Radio Engineering. Vol. 75, No 19, 1775–1785.

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

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

48. Gorokhovatskyi, O., Peredrii, O., Gorokhovatskyi, V., Vlasenko, N. (2023) Explanation of CNN Image Classifiers with Hiding Parts. In: J. Benois-Pineau, R. Bourqui, D. Petkovic, G. Quenot (eds), Explainable Deep Learning Artificial Intelligence, pp. 125-146, Academic Press, 346 p.

49. Gorokhovatskyi V., Tvoroshenko I., Yakovleva O. (2024) Transforming image descriptions as a set of descriptors to construct classification features, Indonesian Journal of Electrical Engineering and Computer Science, 33 (1), 113-125.

50. Gorokhovatskyi, V., Gadetska, S., & Stiahlyk, N. (2023). Accelerating Image Classification based on a Model for Estimating Descriptor-to-Class Distance. International Journal of Computing, 22(4), 485-492.

51. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., Gadetska S., and Al-Dhaifallah M. (2023) Statistical data analysis models for determining the relevance of structural image descriptions, IEEE Access, 11, 126938-126949.

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

53. Bradski, G. (2000). The OpenCV Library. Dr. Dobb’s Journal of Software Tools.

54. Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Oliphant, T. E. (2020). Array programming with NumPy. Nature, 585(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

55. Oliphant, T. E. (2006, December 7). Guide to NumPy [PDF]. Retrieved February 2, 2017, from https://archive.org/details/GuideToNumPy

56. SciPy Community. (2015, October 24). History_of_SciPy [Wiki page]. SciPy. Retrieved May 28, 2025, from https://scipy.github.io/old-wiki/pages/History_of_SciPy

57. Pillow Contributors. (2025). Pillow Documentation (вер. 9.5.0) [Комп’ютерне програмне забезпечення]. Отримано 28 травня 2025 із https://pillow.readthedocs.io/en/stable/

58. Gorokhovatskyi, V., & Tvoroshenko, I. (2020, June). Image classification based on the Kohonen network and the data space modification (CMIS-2020 Paper 11). https://doi.org/10.32782/cmis/2608-76

59. Чиж А.О. (2025) Класифікація ознак з використанням мережі Хемінга. Радіоелектроніка і молодь у XXI столітті: тези доповідей 29-го Міжнародного молодіжного форуму (Харків, 16–19 квітня 2025 р.). Харків: ХНУРЕ, 2025. Т. 7. С. 172-174.