Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

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

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

ДОСЛІДЖЕННЯ РЕЗУЛЬТАТИВНОСТІ ЗАСТОСУВАНЬ АПАРАТУ НЕЙРОМЕРЕЖ У МЕТОДАХ СТРУКТУРНОГО РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ
(тема)

Виконав:
студент 2 курсу, групи ІНФм-18-2

Пупченко Д. В.
(прізвище, ініціали)

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

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Інформатика
(повна назва освітньої програми)

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

Допускається до захисту

Зав. кафедри _____ Путятін Є.П.
(підпис) (прізвище, ініціали)

2019 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
(освітньо-професійна або освітньо-наукова)

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 20 ___ р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

студентові Пупченку Дмитру Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження результативності застосувань апарату нейромереж у методах структурного розпізнавання зображень

затверджена наказом по університету від «21» жовтня 2019 року № 1506Ст.

2. Термін подання студентом роботи до екзаменаційної комісії 23 листопада 2019 р.

3. Вихідні дані до роботи *Моделі отримання структурних описів зображення; математичні моделі методів ORB, FAST, BRIEF, Harris; моделі методів машинного навчання: активаційні функції, SOM Кохонена, GAP, пошук центрів; об'єктно орієнтована мова програмування C++; відкрита бібліотека комп'ютерного зору OpenCV; коллекція тестових зображень Leeds Butterfly Dataset.*

4. Перелік питань, що потрібно опрацювати в роботі
1. Огляд методів обробки та отримання особливих точок зображення.
2. Аналіз моделей отримання дескрипторів з особливих точок зображення.
3. Моделювання та оптимізація методів машинного навчання для структурних описів зображення.
4. Програмна реалізація розроблених методів та аналіз результатів їх застосування.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) *Схема дескрипторів з інваріантністю до обертання, гістограма для дескрипторів ORB, приклад вектору p^i ймовірностей, приклад зображення з координатами ОТ, тестова вибірка зображень, розподіл ОТ за класами, порівняння розподілів отриманих різними способами.*

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

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

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на атестаційну роботу	21.10.2019	
2	Аналіз завдання, підбір літератури	22.10.19-27.10.19	
3	Аналіз літератури з досліджуваної проблеми	28.10.19-02.11.19	
4	Аналіз технічних засобів	03.11.19-05.11.19	
5	Розробка методу	06.11.19-09.11.19	
6	Програмна реалізація	10.11.19-15.11.19	
7	Оформлення пояснювальної записки	16.11.19-19.11.19	
8	Перевірка на плагіат	20.11.19	
9	Рецензування	24.11.19	
10	Підготовка презентації та доповіді	25.11.19-29.11.19	
11	Занесення роботи в електронний архів	30.11.19	
12	Попередній захист атестаційної роботи	2.12.2019	

Дата видачі завдання __21_ ___жовтня____2019 р.

Студент _____
(підпис)

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

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до атестаційної роботи: 75 с., 8 табл., 26 рис., 2 дод., 43 джерела.

КЛЮЧОВІ ТОЧКИ ЗОБРАЖЕННЯ, МЕТОД ORB, ДЕСКРИПТОР КЛЮЧОВИХ ТОЧОК, СТРУКТУРНЕ РОЗПІЗНАВАННЯ, БІНАРНИЙ АНАЛІЗ, МЕРЕЖА КОХОНЕНА, НЕЙРОН, КЛАСТЕРИЗАЦІЯ.

Метою роботи є розробка методів для вирішення проблеми інваріантного розпізнавання візуальних об’єктів з використанням апарату класифікації дескрипторів ключових точок зображення та засобів нейронної мережі Кохонена.

Об’єктом дослідження є розпізнавання різноракурсних зображень.

Використано методи математичного моделювання. Проведено дослідження методів двійкового аналізу дескрипторів для знаходження центрів кластерів та методів згортання дескрипторів зображень за рахунок знаходження відсоткової ймовірності ознак. Досліджені методи формування мережі Кохонена з різними варіантами обробки і знаходження центру класів.

Здійснена програмна реалізація системи класифікації та проведено її дослідження на базах зображень.

IMAGE KEY POINTS, ORB METHOD, KEY POINT DESCRIPTOR, STRUCTURAL RECOGNITION, BINARY ANALYSIS, KOHONEN NETWORK, NEURON, CLUSTERIZATION.

The purpose of this work is to develop methods for solving the problem of invariant recognition of visual objects using the apparatus of classification of key point descriptors of the image and means of the Kohonen neural network.

The object of the study is to recognize multifaceted images.

Methods of mathematical modeling were used. The methods of binary analysis of descriptors for finding cluster centers and methods of minimizing image descriptors by finding the percentage likelihood of features have been investigated. Methods of forming Kohonen network with different variants of processing and computation for the center of classes are investigated.

The classification software was implemented and tested on image databases.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ОТ – особлива точка

ORB – Oriented FAST and Rotated BRIEF

FAST – Features from Accelerated Segment Test

BRIEF – Binary Robust Independent Elementary Features

SIFT – Scale-invariant Feature Transform

SURF – Speeded-Up Robust Features

SOM – Self-organizing map

BMU – Best Matching Unit

SSD – Sum of Squared Differences

RCT – Randomized Classification Trees

LVQ – Learning Vector Quantization

МК – мережа Кохонена

ЛФН – лінійний формальний нейрон

SUSAN – Small Univalue Segment Assimilating Nucleus

PCA – Principal Component Analysis

LDA – Linear Discriminant Analysis

GAP – Global Average Pooling

ВСТУП

Розпізнавання образів є однією з найбільш досліджуваних проблем теорії сучасного комп'ютерного зору, адже воно має велике практичне значення. Замість «розпізнавання» часто використовується термін «класифікація». Ці два терміни у багатьох випадках розглядаються як синоніми, але не повністю взаємозамінні. Кожен з цих термінів має свою сферу застосування, і інтерпретація обох термінів часто залежить від специфіки конкретної задачі.

У даний час набули поширення ряд методів розпізнавання візуальних об'єктів, які засновані на локальних особливостях зображення [1]. Проблеми розпізнавання легко вирішуються людьми, і це зазвичай робиться підсвідомо. Однак спроби створення систем штучного розпізнавання менш переконливі. Основна проблема полягає в тому, що в більшості випадків неможливо адекватно визначити ознаки, за якими має бути зроблено розпізнавання. Для задач, для яких можна виділити такі особливості, системи штучного розпізнавання стали широко поширеними і широко використовуваними.

Можливості сучасних бібліотек програмного забезпечення для комп'ютерного зору, таких як OpenCV, забезпечують вирішення важливих практичних завдань в області комп'ютерного зору: аналіз контенту, пошук і розпізнавання об'єктів, виявлення тексту, відстеження руху об'єкта, виявлення загальних елементів, порівняння зображень, впровадження методів навчання для конкретних відеобаз [2].

Проблема розпізнавання в застосуванні цих методів полягає в тому, щоб обчислити релевантність об'єкта і стандартних описів, представлених у вигляді наборів дескрипторів. Цей підхід дозволяє комп'ютерній програмі працювати з візуальними зображеннями, подібно до людини, чий зір також засновано на локальних особливостях зображення.

Ефективність і час обробки важливі для розпізнавання візуальних баз даних 2D-об'єктів в системах комп'ютерного зору. У зв'язку з цим, при

реалізації прикладних задач, набули широкого поширення структурні методи, засновані на використанні нейронних мереж з метою виявлення закономірностей за багатьма ознаками структурних описів вибіркової бази.

Нейронні мережі, в свою чергу, є одним з напрямків розвитку систем штучного інтелекту. Ідея полягає в тому, щоб максимально близько імітувати роботу нервової системи людини, а саме її здатність вчитися і виправляти помилки. Це головна особливість будь-якої нейронної мережі – вона здатна самостійно вчитися і діяти на основі попереднього досвіду, і кожен раз робити менше помилок.

Завдання, які найчастіше віддають на вирішення нейронним мережам, пов’язані з навчанням на базі якихось елементів. Серед основних додатків нейронних мереж – прогнозування, прийняття рішень, розпізнавання образів, оптимізація, аналіз даних.

Для таких завдань, в яких опис безлічі об’єктів відомо (є описана навчальна вибірка), і необхідно виявити внутрішні зв’язки, залежності, шаблони, які їх об’єднують, найбільше підходять методи навчання нейронних мереж без вчителя. Ці методи використовуються, коли відсутній зовнішній сигнал для інформування нейронної мережі про те, наскільки правильно реагувати на вхідний сигнал. Отже, для формування цих методів дуже важливо знати мету навчання, яка безпосередньо пов’язана з нейронною мережею.

Нейронна мережа імітує не тільки активність, але і структуру нервової системи людини. Мережа складається з великої кількості обчислювальних елементів («нейронів»). Кожен «нейрон» належить до певного рівня мережі. При роботі нейрон одночасно отримує багато вхідних сигналів. Кожен вхід має свою власну синоптичну вагу, яка дає вплив, необхідний для функції суматора елемента обробки.

Нейронні ваги є мірою сили вхідних з’єднань і моделюють різні синоптичні сили біологічних нейронів. Вага значимого входу збільшується, і, навпаки, вага незначного входу примусово зменшується, що визначає

інтенсивність вхідного сигналу. Шкали можуть варіюватися в залежності від прикладів навчання, топології мережі та правил навчання. Параметри кожного «нейрона» можуть змінюватися в залежності від результатів, отриманих в попередніх наборах даних, що призводить до зміни порядку роботи всієї системи [3].

Цікавим є вивчення властивостей штучних нейронних мереж при їх застосуванні в задачах класифікації зображень, де на практиці спостерігається практично необмежену різноманітність даних, а також вивчення ефективності схем навчання мереж, які враховують ступінь близькості елементи класу в побудованому просторі ознак.

1 ОГЛЯД МЕТОДІВ СТРУКТУРНОЇ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ

1.1 Методи виділення ключових точок зображення

Вибір ключових точок є основою багатьох проблем в задачах комп'ютерного зору, таких як розпізнавання об'єктів або структур на зображенні. Завдання розпізнавання об'єктів у мають складні питання, такі як освітлення, варіант обертання, варіант масштабу та зони зіткнення. Ці виклики вимагають використання методів вилучення особливостей об'єкта, на які не може вплинути шум або артефакти.

Для будь-якого об'єкта на зображенні можна отримати точки інтересу на об'єкті, щоб створити «опис особливостей» об'єкта. Ці особливості, отримані з зображення, можуть вживатися для ідентифікації об'єкта, для його виявлення на іншому тестовому зображенні, яке містить багато інших об'єктів. Для надійного розпізнавання важливо, щоб характеристики, отримані на оригінальному зображенні чи документі, можна було розрізнити навіть при зміні розміру, освітленості і шуму. Такі точки зазвичай розташовані на високо контрастних областях зображення, наприклад, на кордонах об'єкта.

Особливі точки – це просторові розташування або точки на зображенні, які визначають, що цікаво, чи що виділяється на зображенні, з точки зори комп'ютеру то обраного алгоритму. Причина, по якій ці точки є особливим, полягає в тому, що незалежно від того, як зображення змінюється, які афінні або проектні перетворення виконуються на зображенні, ви все одно зможете знайти той же ключові точки в цьому зміненому зображенні по порівнянні з вхідним зображенням.

Останнім часом було впроваджено багато підходів та досягнень, таких як методи, засновані на моделях, на основі форми та зовнішнього вигляду. Сучасні методи вживають різні детектори кута для виявлення особливих точок та їх порівняння. Для зіставлення зображень, які еволюціонували в результаті деяких перетворень і спотворення, такі як масштаб, обертання,

освітлення, шум і стиснення, дослідники запропонували багато надійних функцій. Щоб проілюструвати відомі детектори і дескриптори, згадаємо алгоритми, які найчастіше зустрічаються в наукових роботах, це ORB, SIFT, SURF, BRISK, BRIEF, HARRIS, FAST і MSER.

Незважаючи на переваги існуючих методів, все ще існує великий попит на такі алгоритми, щоб наблизитися до виключення недоліків пропонованих методів. Оскільки існує компроміс між надійним виявленням ознак і часом виконання, найшвидший алгоритм, який дає кращі результати в будь-яких умовах, ще не був розроблений.

Отже вибір таких алгоритмів схожий на вибір алгоритмів безпеки. Оскільки, неможливо отримати комбінацію найкращою точності, а також найкращої безпеки і мінімального часу обчислень одночасно. Тому ми повинні піти на поступки заради вибору оптимального методу виявлення ознак по відношенню до виконуваної задачі.

У цій роботі використовується високоефективний метод обчислення двійкових дескрипторів, який називається ORB. Алгоритм був введений Ітаном Рублі, Вінсентом Рабо, Куром Коноліге та Гарі Р. Брадські у своїй роботі у 2011 році, як альтернатива методам SIFT та SURF з покращеними показниками обчислювальної вартості та продуктивності. ORB – це безкоштовний алгоритм з відкритим кодом, доступний як частина бібліотеки комп’ютерного зору OpenCV.

Метод ORB заснований на комбінації алгоритмів, таких як детектор FAST (Features Accelerated from Segment Test) і дескриптор BRIEF (binary robust independent elementary features), з деякими поліпшеннями в їх функціональності [4, 5].

Модульність побудови методу ORB дозволяє вибрати довільні метод побудови детектора в поєднанні з будь-яким методом визначення дескриптора і навпаки, оптимізуючи бажану продуктивність вирішуваних завдань.

Тестування алгоритму ORB показує, що він менш вимогливий до обчислювальних ресурсів, ніж інші алгоритми які працюють з локальними прикметами зображення. Збільшення швидкості обчислень пояснюється простою процедурою побудови дескрипторів. ORB дає приріст продуктивності з порівняльною або кращою точністю, ніж в аналогів SIFT і SURF [6, 7].

ORB має можливість контролювати кількість ОТ. На процес виявлення впливають такі фактори як:

– зазначена максимальна кількість збережених ОТ;

– коефіцієнт поділу масштабу піраміди, яких визначається в межах 1 ... 2 і впливає на деталізацію обробки зображення;

– обмежувач кількості рівнів піраміди;

– розмір кордону навколо вже певного ОТ, в межах якого нові ОТ не утворюються;

– граничне значення для детектора Харріса, який використовується для класифікації і зберігання кращих ОТ.

Отримання ОТ і дескрипторів в ORB займає кілька етапів. На першому етапі будується масштабна гауссова піраміда зображення. Потім на кожному рівні шкали піраміди визначаються екстремуми яскравості. Для цього використовується алгоритм FAST.

Існують і інші алгоритми виявлення ОТ, такі як: алгоритм Моравека, алгоритм визначення кута Харріса і Стівенса, SUSAN. Причиною створення алгоритму FAST стала необхідність створення ОТ-детектора для використання в додатках реального часу на пристроях з обмеженими обчислювальними ресурсами.

Алгоритм FAST – це алгоритм виявлення функцій, запропонований Ростен та Драммонд [8], що є вдосконаленням алгоритму вилучення кута SUSAN. Він зберігає алгоритм SUSAN для виявлення характеристик різних точок функцій [9, 10], та має переваги швидкості виявлення та високої точності виявлення точок функції.

Згідно FAST, для деякої точки зображення формується коло деякого радіуса, і обчислюється кількість пікселів, що лежать на ньому і мають значення, менше або більше яскравості його центру. Якщо таких точок більше 75%, то центр кола вважається кандидатом на роль ОТ.

ORB використовує алгоритм FAST-9 (передбачається, що радіус кола для обчислення дорівнює 9), оскільки він виявився найбільш ефективним з точки зору продуктивності в дослідженнях проведених розробниками [11].

При обчисленні кутових точок спочатку розглядаються околиці в 16 пікселів навколо обранного пікселя p. Точка p вважається особливою, якщо існує N пікселів в її окружності довжиною 16 пікселів та якщо для трьох пікселів з чотирьох виконується умова $I_i < I_p - t$ або $I_i > I_p + t$, $i = 1 \dots 4$, p вважається особливою точкою.

При виконанні цієї умови досліджується значення яскравості на окружності під номерами 1, 5, 9, 13 (рис. 1.1).

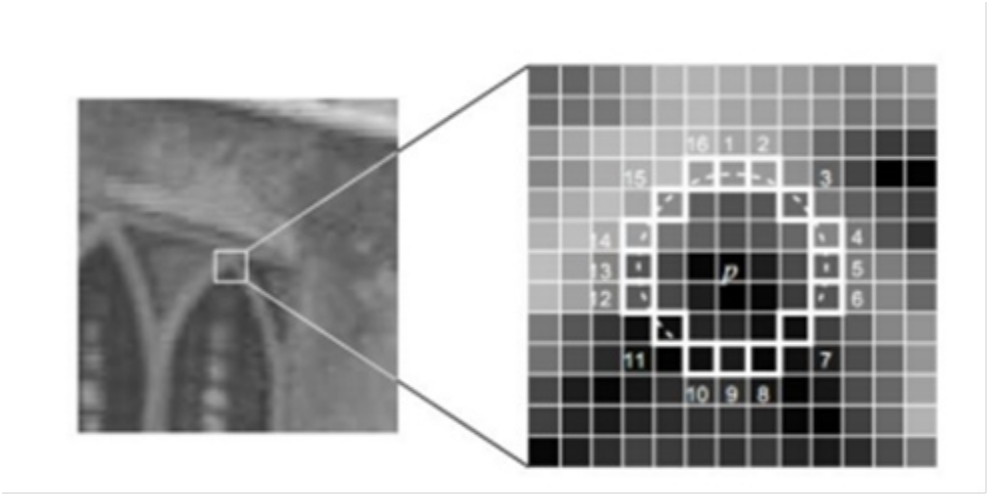

Рисунок 1.1 – Околиця точки p FAST детектора

Вибір тільки 4 пікселів дозволяє обрізати точки, які не підходять, але в деяких випадках можливо визначити різні елементи в одному колі. В алгоритмі ORB максимальна кількість спеціальних точок за замовчуванням не перевищує задане значення, якщо воно використовується, детектор кута

Харріса використовується для виключення найменш значущих особливих точок.

Детектор Харріса [12] побудований на основі детектора Моравця і є удосконаленням. У порівнянні зі своїм попередником, він інваріантний щодо повороту, кількість помилок визначення кутів поменшено через введення згортки з гауссовими вагами.

Для зображення I розглядається вікно W з центром (x, y) і його зсувами, і обчислюється зважена сума квадратів різниць (SSD) між зміщенням і вихідним вікном.

Алгоритм детектора кутів Харріса заснований на центральному принципі: в залежності від кута яскравість зображення буде різною в різних напрямках. В якості альтернативи, це може бути сформульовано шляхом вивчення змін інтенсивності через зсуви в локальному вікні. Навколо кутової точки інтенсивність зображення буде сильно мінятися при переміщенні вікна у випадковому напрямку. Слідкуючи цій формулі, детектор Харріса використовує другу матрицю перетворення в якості основи своїх кутових рішень.

Матриця A, також звана матрицею автокореляції, має значення, тісно пов’язані з похідними інтенсивності зображення.

$$A(x) = \sum_{p,q} \omega(p,q) \begin{bmatrix} I_x^2(p,q) & I_x I_y(p,q) \\ I_x I_y(p,q) & I_y^2(p,q) \end{bmatrix},$$

де I_x та I_y – відповідні похідні (інтенсивності пікселів) у напрямку x і y у точці (p, q); p та q – параметри положення вагової функції ω.

Ця матриця описує форму вимірювання автокореляції, яка пов’язана зі змінами в розмітці вікна. Таким чином, ми отримаємо кількісний опис того, як міра автокореляції змінюється в просторі: її основні кривини.

Детектор Харріса залишається дуже стабільним при розмитті зображення. Оскільки детектор не залежить від сегментації зображення,

повторюваність і відповідні оцінки залишаються незмінними. Недоліком цього методу є те, що він слабо стійкий до зміни кута зору [13].

Детектор Харріса частково інваріантний до змін освітленості, зміщення і масштабування. Для інваріантності до збільшення або зменшення застосовується алгоритм на піраміді Гаусса. Введення параметра кутової орієнтації дозволяє домогтися стабільності виявлення при обертанні об’єкта. Він заснований на напрямках градієнта яскравості навколо центру точки, напрямок з найбільшою інтенсивністю присвоюється орієнтації особливої точки θ.

Детектор Харріса вимагає більше обчислень, ніж конкуруючі детектори через необхідність побудови згортки ядра Гауса. Тим не менш, він досить чутливий до шуму. Пониження шумів може збільшити розмір вікна Гауса, але це призводить до значних обчислювальних витрат, тому необхідно знайти компроміс між якістю алгоритму і кількістю виконуваних операцій.

Детектор Харріса має властивість анізотропії по горизонтальному і вертикальному напрямках, оскільки матриця автокореляції містить перші похідні тільки за цими напрямками. У порівнянні зі своїм попередником цей детектор інваріантний щодо повороту, кількість помилок виявлення кутів невелика через введення згортки з гаусовими ваговими коефіцієнтами. Результати виявлення значно змінюються при масштабуванні зображення.

1.2 Класифікація зображень за множиною дескрипторів ключових точок

Результатом роботи детекторів є набір спеціальних точок, для яких необхідно побудувати математичний опис. На вхід при обчисленні дескриптора подається зображення і набір спеціальних точок, виділених на даному зображенні. Результатом роботи дескриптора є набір векторів ознак для вихідного набору особливих точок. Слід зазначити, що деякі

дескриптори вирішують дві проблеми одночасно – знаходження конкретних точок і побудова описів цих точок.

Знаки (опису) засновані на інформації про інтенсивність, кольори і текстури особливої точки. Але спеціальні точки можуть бути представлені кутами, ребрами або навіть контуром об’єкта, тому, як правило, обчислення виконуються для деякої околиці. В ідеалі хороші ознаки повинні володіти рядом властивостей.

1. Частота. На зображеннях одного і того ж об’єкта або сцени, знятих з різних точок зору і при різних умовах освітлення, необхідно проектувати як найбільше ОТ;

2. Місцезнаходження. ОТ повинні бути якомога більш локальними, щоб зменшити ймовірність перекриття;

3. Репрезентативність. Кількість ознак повинно бути достатнім для виявлення, тобто необхідно досягнути розумної кількості ознак навіть на невеликому зображенні об’єкта;

4. Точність. Ознаки повинні бути точно визначені в залежності від масштабу і форми об’єкта;

5. Ефективність. Для додатків реального часу дуже важливо, щоб процес розрахунку не вимагав значних обчислювальних витрат;

Щоб побудувати дескриптор отриманих точок, у роботі використовується модифікація BRIEF [14], інваріантна до повороту через використання додаткових перетворень.

Дескриптори характерних точок в даний час лежать в основі багатьох технологій Computer Vision, таких як розпізнавання об’єктів, 3D-реконструкція, пошук зображень і локалізація камери. Оскільки додатки цих технологій повинні обробляти все більше даних або працювати на мобільних пристроях з обмеженими обчислювальними ресурсами, існує зростаюча потреба в локальних дескрипторах, які швидко обчислюються, швидко зіставляються і ефективно використовують пам’ять.

Одним із способів прискорення зіставлення і скорочення споживання пам’яті є робота з короткими дескрипторами. Вони можуть бути отримані шляхом застосування зменшення розмірності, такого як PCA [15] або LDA [16], до оригінального дескриптору, такому як SIFT [17] або SURF [18]. Наприклад, в [19, 20, 21] було показано, що значення вектора дескриптора з плаваючою комою можна квантувати, використовуючи дуже мало бітів на значення без втрати продуктивності розпізнавання.

Ще більш значне зменшення розмірності може бути досягнуто за допомогою хеш-функцій, які призводять дескриптори SIFT до бінарних рядків, як це зроблено в [22]. Ці рядки представляють бінарні дескриптори, схожість яких може бути виміряна відстанню Хеммінга. Але незважаючи на це, ці підходи до зменшення розмірності вимагають спочатку обчислення повного дескриптора, перш ніж буде мати місце подальша обробка.

Автори показують, що все це обчислення може бути скорочено шляхом безпосереднього обчислення довічних рядків з патчів зображення. [4] Окремі біти виходять шляхом порівняння інтенсивностей пар точок по тим самим лініях, що і в [23], але без необхідності в навчальній фазі. Саме такий метод автори назвали BRIEF. Проведені експерименти показують, що тільки 256 біт або навіть 128 біт часто досить для отримання дуже хороших результатів зіставлення. Таким чином, BRIEF дуже ефективний як для обчислення, так і для зберігання в пам’яті.

Крім того, порівняння рядків може бути виконано шляхом обчислення відстані Хеммінга, яке може бути виконане з надзвичайною швидкістю на сучасних процесорах, які часто надають спеціальну інструкцію для виконання операції XOR або підрахунку бітів, як це має місце в останньому наборі команд SSE [24]. Це означає, що BRIEF легко перевершує інші швидкі дескриптори, такі як SURF і U-SURF, з точки зору швидкості [25].

Використання BRIEF забезпечує розпізнавання ідентичних областей зображення, знятих з різних кутів. Кількість виконаних розрахунків зведено до мінімуму. Алгоритм розпізнавання зводиться до побудови масиву

випадкових дерев (RCT) або наївного байєсівського класифікатора на деякому навчальному наборі зображень і подальшої класифікації розділів тестових зображень [4].

У спрощеному варіанті здійснення метод найближчого сусіда може використовуватися для знаходження найбільш схожого графіка в навчальній вибірці. Невелика кількість операцій забезпечується поданням вектора ознак у формі двійкових векторів і, як наслідок, використанням в якості міри схожості відстані Хеммінга.

Початковий варіант BRIEF-алгоритму не має інваріантності до обертання, і, отже, алгоритм зазнає невдачі, коли зображення обертається. Це пов’язано з тим, що BRIEF не вираховує напрямок кутів.

Щоб отримати дескриптор, повинно створити квадратне вікно з центром в ключовій точці і відповідно до його орієнтації. У цьому вікні по заданому правилу вибирається набір пар точок, значення яскравості яких порівнюються один з одним. Кожен з чотирьох кутів квадрата на рисунку 1.2 може бути чітко описаний за допомогою рядка двійкових символів.

Двійкові рядки символів чотирьох дескрипторів шаблону – це 00011, 10100, 01000 і 00000, а чотири рядки двійкових символів мети – 00011, 10100, 01000 і 00000. Через різницю, яка існує між ними, чотири дескриптора для тому шаблон і мета можуть бути чітко розділені.

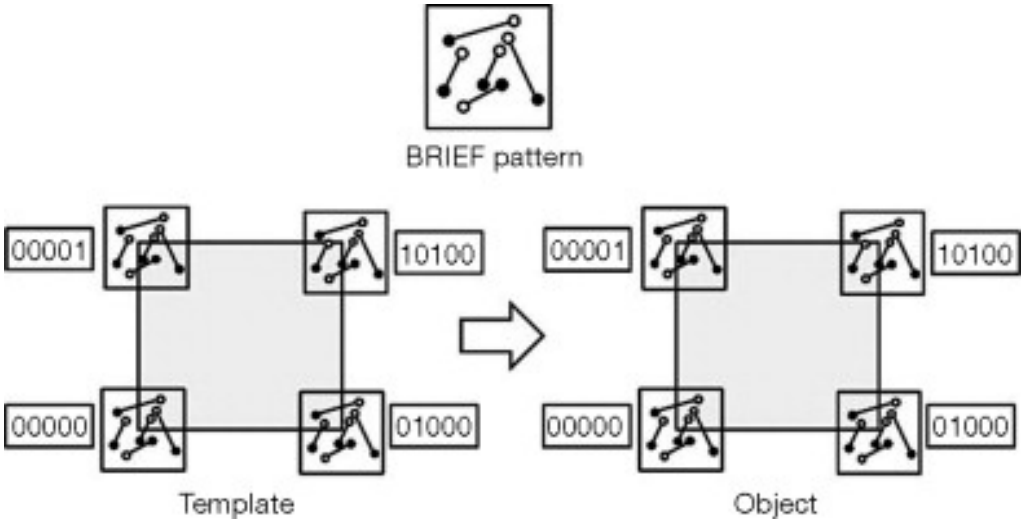

Рисунок 1.2 – Схема дескрипторів без інваріантності до обертання

Для розрахунку дескриптора в околиці ОТ, отриманого при роботі FAST, за поточним масштабом виділяється область, для якої знаходиться центр мас. Для досягнення інваріантності до повороту область обчислення дескриптора орієнтується по орієнтації особливої точки θ, тобто вектор, спрямований з певної точки до центру мас, буде визначати орієнтацію конкретної точки.

Все n = 256 наборів x_i, y_i формують матрицю S розмірністю 2 × n. Далі S із застосуванням матриці повороту R_θ орієнтується відповідно кута θ:

$$S_\theta = R_\theta S.$$

Вектор дескриптора отримуємо як:

$$g_n(I,\theta) := f(I) \mid (x_i, y_i) \in S_{\theta,}$$
$$f_n(I) := \sum_{1 \le i \le n} 2^{i-1} \tau(I; x_i, y_i).$$

Якщо яскравість першої точки вище, відповідний елемент дескриптора записується в 1, в іншому випадку записується значення 0. Після виконання обчислення напрямків по кутах шаблону і цілі модель отримує інваріантність до обертання (рис. 1.3).

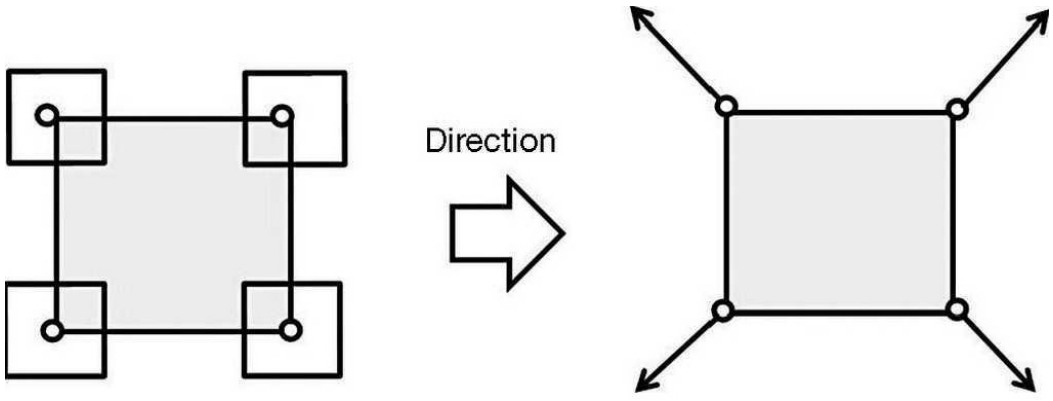

Рисунок 1.3 – Схема отримання інваріантності до обертання

Підходи, прийняті в останні роки для обчислення дескрипторів з обертальний інваріантністю, такі як BRISK і FREAK, більш-менш збігаються з ORB. Побудовані таким чином дескриптори можуть порівнюватися один з одним відповідно до норми Хеммінга.

Результат методу представлений у вигляді вектора довжиною 256, який складається з результатів випробувань по ОТ. В околиці 31 × 31 пікселів порівнюються значення яскравості між x і y, де x, y це області 5 × 5 пікселів:

$$\tau(I_{x,y}) := \begin{cases} 1 : I_x < I_y \\ 0 : I_x \geq I_y \end{cases};$$

де I – середня яскравість обраної області.

На рисунку 1.4 показано, що дескриптор з інваріантністю до обертання має те ж самий рядок двійкових символів, який використовується в якості дескриптора шаблону при обертанні.

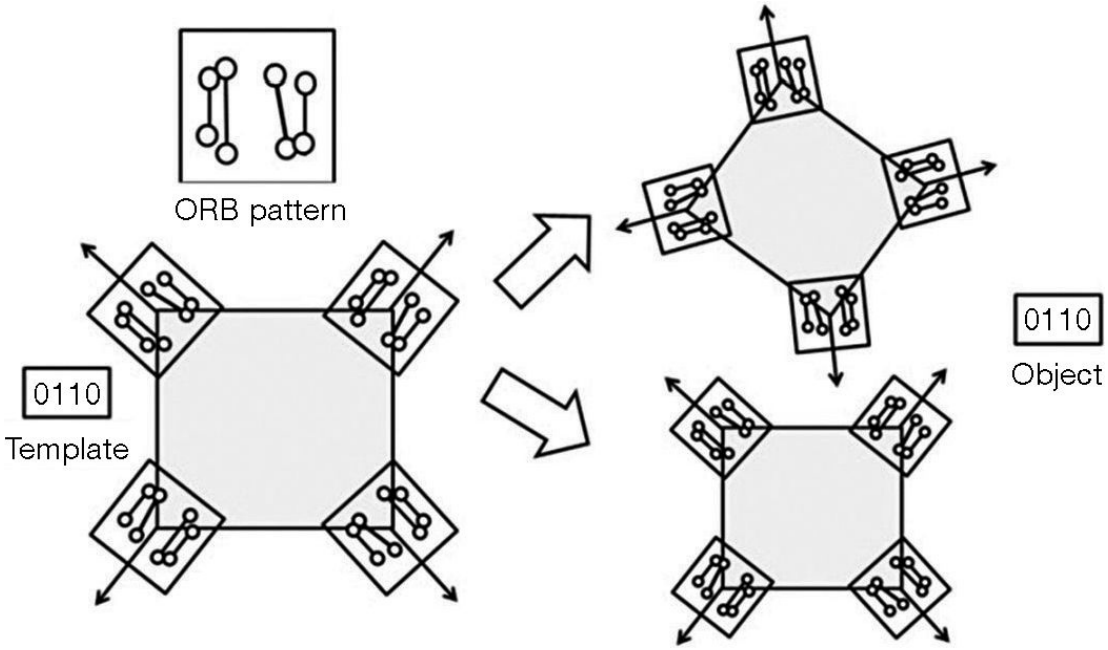

Рисунок 1.4 – Схема дескрипторів з інваріантністю до обертання

Сам дескриптор ORB, у векторному вигляді, показаний на рисунку 1.5, де одиниця позначена чорним кольором, а нуль – білим.

Рисунок 1.5 – Гістограма для дескрипторів ORB

Автори представляють результати експериментів (розпізнавання) при виборі пар точок по закону рівномірного розподілу, а також нормального розподілу з різними значеннями математичного очікування і стандартного відхилення. Результатом є те, що при одних і тих же умовах експериментів точність виявлення за допомогою BRIEF майже в 1,5 рази вище, ніж при використанні аналогічних дескрипторів [25, 4].

1.3 Активаційні функції штучних нейронів

Публікацією, що заклала основи для створення поняття нейронних мереж і штучних нейронів, які є їх невід’ємною частиною, вважалися роботи Уоррена С. Маккаллока і Уолтера Питтса [26]. У цій роботі була введена теорія, заснована на тому факті, що всі аспекти нервової діяльності можуть моделюватися мережею елементів, які мають два стабільних стани.

У більш інтуїтивних термінах нейрони можуть розумітися як субодиниці нейронної мережі в біологічному мозку (рис. 1.6). Тут сигнали змінних величин досягають дендритів.

Ці вхідні сигнали потім накопичуються в тілі клітини нейрона, і якщо накопичений сигнал перевищує певний поріг, генерується вихідний сигнал, який буде передаватися аксоном.

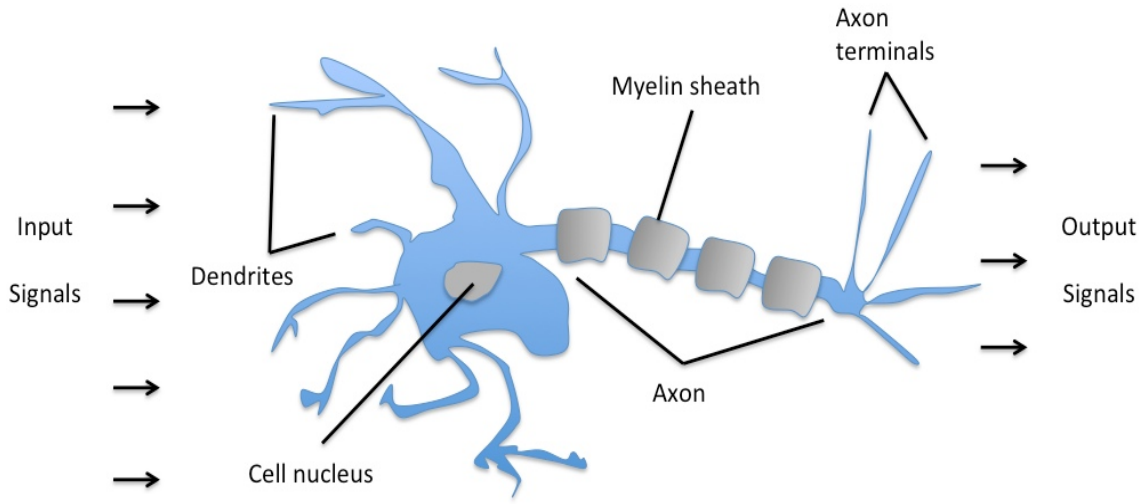

Рисунок 1.6 – Схема біологічного нейрону

Через кілька років після Маккаллока і Піттса, Френк Розенблат опублікував першу концепцію правила навчання перцептроном [27]. Основна ідея полягала в тому, щоб визначити алгоритм, щоб дізнатися значення ваг w, які потім множаться на вхідні характеристики, щоб прийняти рішення, активований нейрон чи ні.

В контексті класифікації патернів такий алгоритм може бути корисний для визначення того, чи належить вибірка одного класу або іншому. Такий елемент був названий формальним нейроном (рис. 1.7).

Ідея цього «порогового» перцептрону полягала в тому, щоб імітувати, як працює один нейрон в мозку: він або активований, або ні. Отже персептрон отримує кілька вхідних сигналів, і якщо сума вхідних сигналів перевищує певний поріг, він або повертає сигнал, або залишається не активованим в іншому випадку.

Те, що зробило цей алгоритм «машинного навчання» справжні перспективним, було ідеєю Френка Розенблатта про правило навчання персептрона: алгоритм перцептрону призначений для вивчення ваг для вхідних сигналів, щоб намалювати лінійний кордон рішення, який дозволяє нам розрізняти два лінійно розділимих класа + 1 і -1.

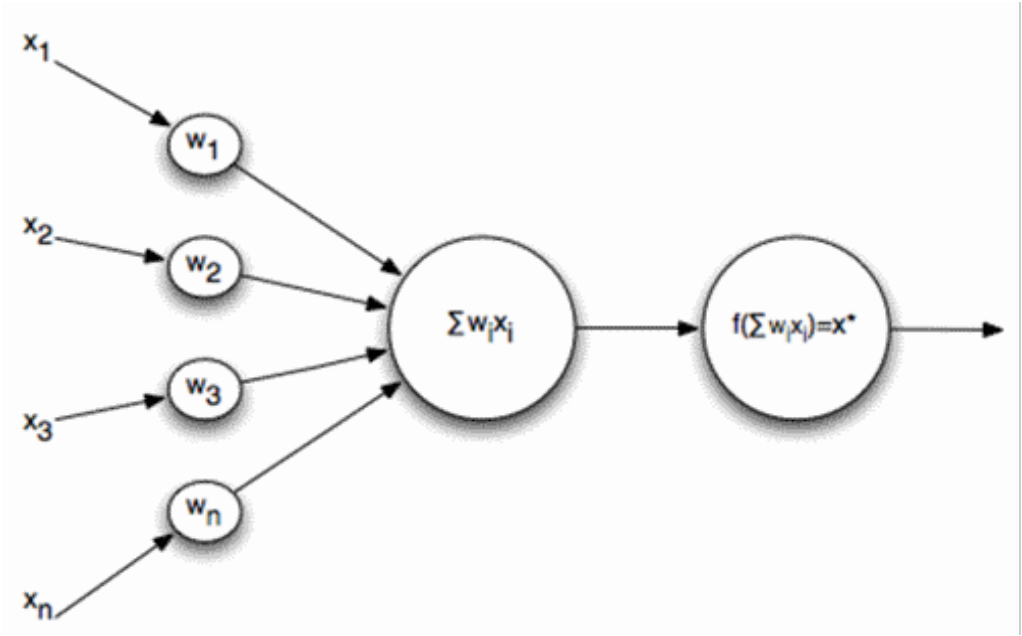

Рисунок 1.7 – Схема формального нейрону

Моделювання функцій синапсів відбувається шляхом масштабування вхідних сигналів $(x_0, x_1, ..., x_n)$ за допомогою вагових коефіцієнтів $(w_0, w_1, ..., w_n)$.

Одержані сигнали подаються на вхід суматора, який виконує їх обробку за формулою:

$$v = \sum_{i=0}^{n} w_i x_i .$$

Змінна v є аргументом порогової активаційної функції:

$$y = f(v).$$

У сучасній літературі існує велика кількість парадигм штучних нейронних мереж, елементи яких реалізують різні функції активації.

Метою функцій активації є введення нелінійностей в мережу. Лінійні функції активації здійснюють лінійні рішення незалежно від вхідного розподілу. Нелінійності дозволяють краще наближати довільно складні

функції. Активаційна функція, що вперше була запропонована в [26], має вигляд:

$$y = \begin{cases} 1, v \le 0, \\ 0, v > 0. \end{cases}$$

Найбільш поширеною є сигмоїдальна функція, яка може бути представлена в дискретному і аналоговому варіантах. Це логічна функція з параметрами b, c і d, яка визначається виразом:

$$y = \frac{b}{c + e^{dv}}.$$

При одиничних значеннях параметрів b = 1, c = 1, d = −1 одержуємо:

$$y = \frac{1}{1 + e^{-v}}.$$

Рисунок 1.8 показує, що крутизна S-образної функції залишається високою тільки в деякому певному діапазоні.

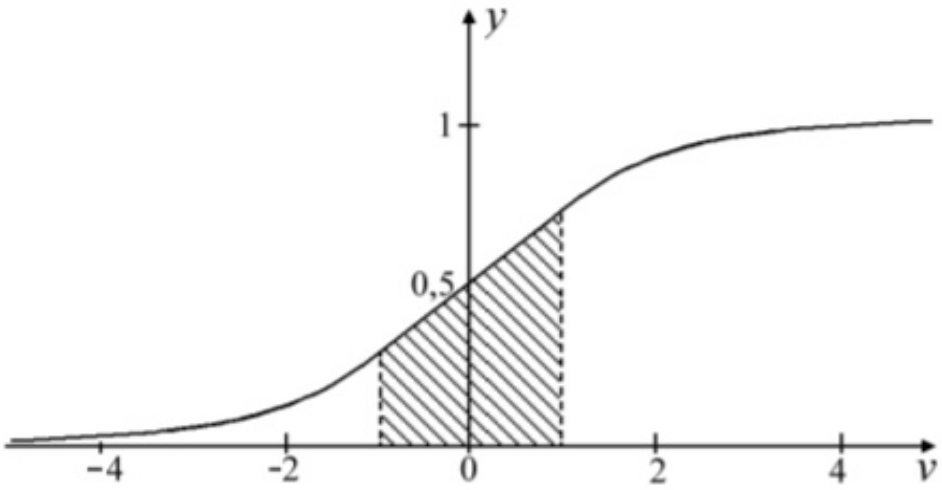

Рисунок 1.8 – Сигмоїдальна логічна функція

Отримана амплітуда з цією функцією активації залежить від амплітуди вхідних сигналів. У залежності від нелінійного посилення нейрон може надійно працювати в широкому діапазоні вхідних рівнів. Іншим корисним аспектом сигмоідальної функції активації є диференціація по всій абсцисі і просте вираження для похідної.

При застосуванні формальних нейронів з сигмоїдальною або іншою безперервною функцією активації в штучних нейронних мережах доцільно вирішити проблеми регресії. Такий поділ проведено певною мірою умовно, оскільки в задачах класифікації також можуть використовуватися функції безперервної активації, значення яких розглядаються як ймовірності належністю до відповідних класів.

Прикладом більш простої функції активації, яка використовується частіше, є дискретна сигмоїда з параметром, званим функцією Хевісайда або тета-функцією:

$$y = \begin{cases} 1, v \le a, \\ 0, v > a. \end{cases}$$

Якщо значення v нейрона з функцією активації Хевісайда (рис. 1.9) не перевищує значення параметра a, то нейрон залишається пасивним, а коли поріг перевищено, дає значення функції, прийнятої за логічну одиницю.

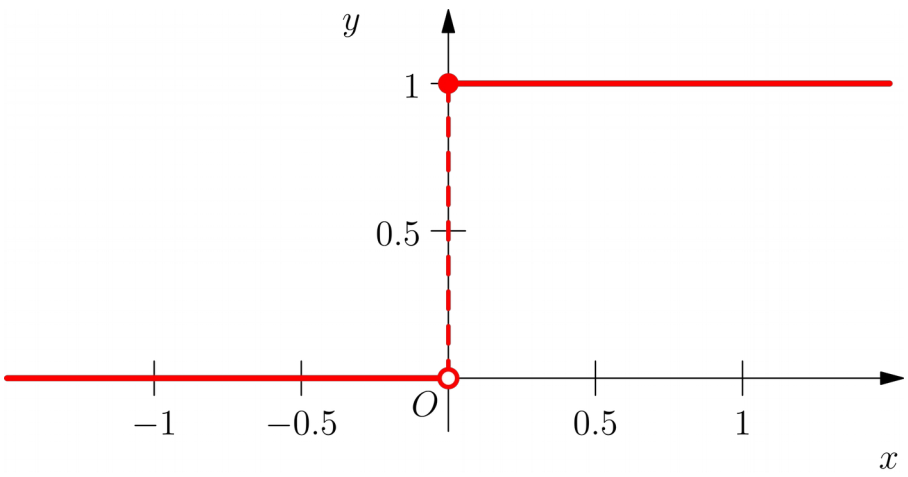

Рисунок 1.9 – Логічна функція Хевісайда

Використання дискретної функції активації призводить до поділу вихідного простору на кілька класів. Завдання нейронної структури в цьому випадку полягає в тому, щоб встановити чітку відповідність між векторами вхідних сигналів і цими класами. Тип завдань для встановлення такої відповідності називається завданнями класифікації або кластеризації.

1.4 Застосування штучних нейронних мереж для класифікації

Формально штучну нейронну мережу називають структурою, яка складається з великої кількості елементів, кожен з яких має локальну пам’ять і може взаємодіяти з іншими елементами через канали зв’язку з метою передачі даних, які можуть бути довільно інтерпретовані. Ці елементи обробляють локальні дані, які вони отримують через вхідні канали. Зміна параметрів алгоритмів обробки залежить тільки від характеристик даних.

Поняття класу не завжди чітко визначено і визначається в контексті вирішуваних проблем. Але в більшості випадків термін «клас» означає набір об’єктів, згрупованих за деякими атрибутам і правилам. Фактично завдання класифікації полягає в застосуванні до векторів вхідних сигналів встановлених правил для визначення набору ознак і прийняття рішення про призначення конкретного вектора класу.

Кластеризація – це завдання поділу сукупності чи точок даних на ряд груп, так що точки даних у одних і тих же групах більше схожі на інші точки даних цієї ж групи, ніж на інші групи. Простими словами, мета – відокремити групи з подібними ознаками та розподілити їх у кластери.

Якщо проблема кластеризації дозволяє динамічно змінювати набір ознак, то правила формування цих ознак залишаються незмінними і визначаються моделлю нейрона.

Тому питання полягає в тому, чи може конкретна нейронна структура бути універсальним класифікатором. Іншими словами, чи може весь дійсний

набір векторів вхідних сигналів бути унікально розподілений серед заданих класів. Відповідь на це питання полягає у вирішенні проблеми лінійного розділу.

Наприклад хоча персептрон відмінно працює для двох сильно різних класів, конвергенція є однією з найбільших проблем персептрона. Френк Розенблат математично довів, що правило навчання персептрона сходиться, якщо два класи можна розділити лінійної гіперплощиною, але виникають проблеми, якщо класи не можуть бути повністю розділені лінійним класифікатором.

Щоб логічна функція була реалізована на формальному нейроні, вона повинна бути лінійно розділена. Визначення лінійного розділу логічної функції не є тривіальним в разі її багатовимірності, коли число можливих варіантів дуже велике. Отже, проблема лінійного дозволу повинна бути подолана шляхом використання відповідних архітектур нейронних мереж.

Сьогодні відомо велика кількість нейронних структур і їх модифікацій, які орієнтовані на рішення певного типу проблем. Найбільш відомі типи таких структур показані на рисунку 1.10.

Основна мета при розробці системи машинного навчання полягає в тому, щоб гарантувати, що алгоритм навчання буде узагальнювати або точно виконуватися на нових прикладах після навчання на кінцевому числі з них.

Стабільність, також відома як алгоритмічна стабільність, є поняттям в обчислювальної теорії навчання про те, як алгоритм машинного навчання порушується невеликими змінами його вхідних даних. Стабільний алгоритм навчання – це алгоритм, для якого пророкування не сильно змінюється, коли дані навчання трохи змінені [28].

Наприклад, для алгоритму машинного навчання, який навчається розпізнавати рукописні букви алфавіту, використовуючи 1000 прикладів рукописних букв і їх міток (від «A» до «Z») в якості навчального набору. Один із способів змінити цей навчальний набір – це пропустити приклад, щоб було доступно тільки 999 прикладів рукописних букв і їх міток.

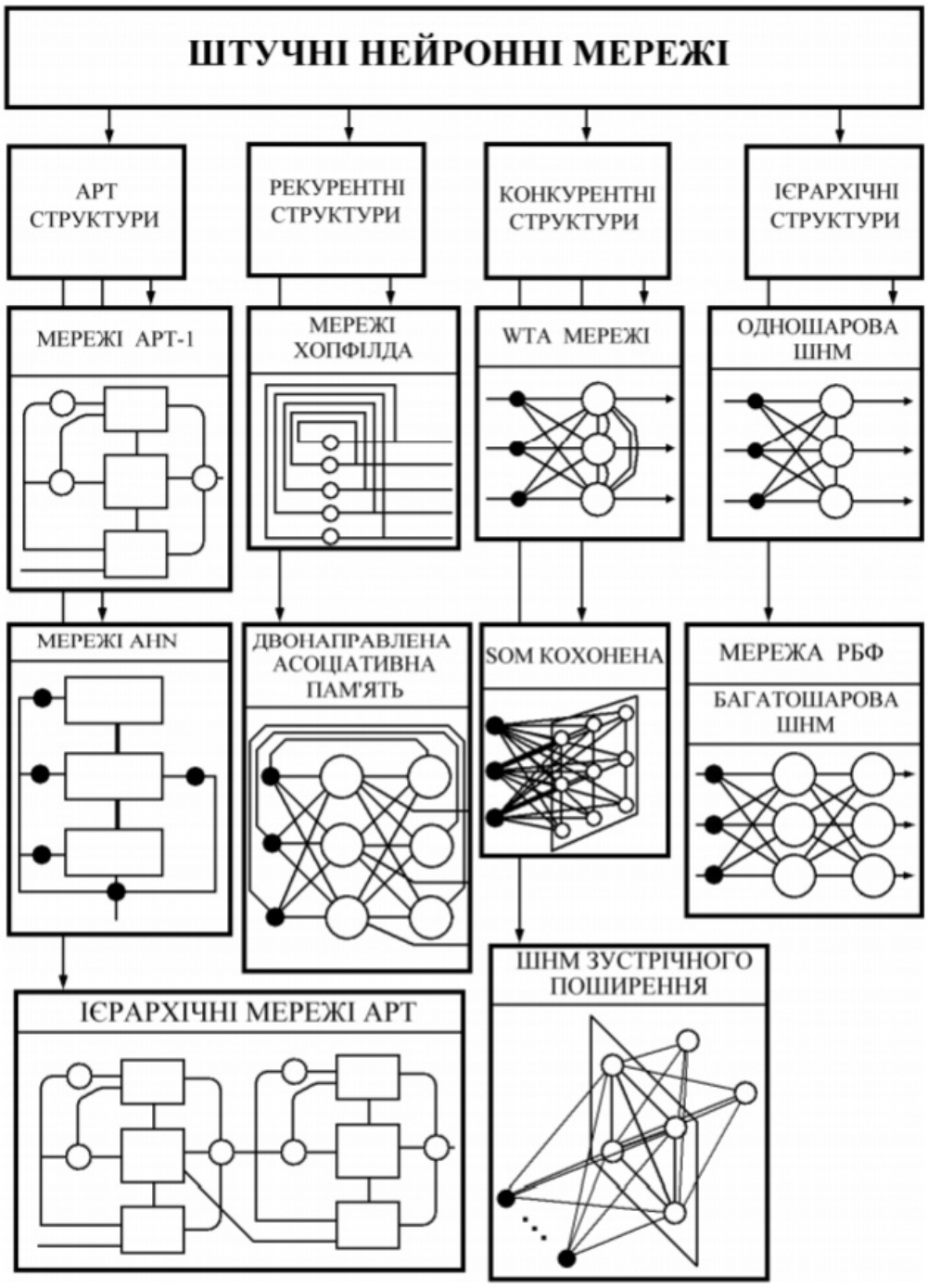

Рисунок 1.10 – Нейронні структури

Стабільний алгоритм навчання дасть схожий класифікатор з навчальними наборами, що складаються як з 1000 елементів, так і з 999 елементів. Стійкість можна вивчати для багатьох типів проблем навчання, від вивчення мови до обернених задач у фізиці і техніці, оскільки це властивість процесу навчання, а не тип інформації, що вивчається.

Існує проблема перенавчання в машинному навчанні. Перенавчання означає модель, яка занадто добре моделює тренувальні дані. Перенавчання відбувається, коли модель вивчає деталі і шум в навчальних даних в тій мірі, в якій це негативно впливає на продуктивність моделі на нових даних. Це означає, що шум або випадкові флуктуації в навчальних даних підібрані і вивчені моделлю як важливі характеристики.

Проблема в тому, що ці поняття не можливо застосувати до нових даних і вони негативно впливають на здатність моделей до узагальнення. Перенавчання більш імовірно при використанні непараметричних і нелінійних моделей, які мають більшу гнучкість при вивченні цільової функції. Таким чином, багато непараметричних алгоритмів машинного навчання також включають параметри або методи, щоб обмежувати деталізацію, яку вивчає модель.

Наприклад, дерева рішень є непараметричим алгоритмом машинного навчання, який є дуже гнучким і часто стикається з перенавчанням навчальних даних. Цю проблему можна вирішити, обрізавши дерево після того, як воно навчилося, щоб видалити деякі деталі, які воно вивчило.

Протилежний варіант – недостатнє навчання в машинному навчанні. Недостатнє навчання відноситься до моделі, яка не може ні моделювати дані навчання, ні узагальнювати на нових даних. Модель машинного навчання з недостатнім навчанням не є практичною моделлю і це буде очевидно, оскільки вона буде мати низьку продуктивність на тренувальних даних. Недостатнє навчання часто не обговорюється, тому що його легко виявити, з огляду на наявність хорошого показника продуктивності.

Рішення проблеми часто вимагає переходу на альтернативні алгоритми машинного навчання. Проте, це забезпечує хороший контраст з проблемою перенавчання. В ідеалі, необхідно вибрати модель в найкращому місці між недостатнім навчанням і перенавчанням.

Це мета, але на практиці це дуже складно зробити. Щоб зрозуміти цю мету, ми можемо подивитися на продуктивність алгоритму машинного

навчання з плином часу, поки він вивчає дані навчання. Ми можемо відобразити коректне розпізнавання як на тренувальних даних, так і в тестовому наборі даних, який видається алгоритму окремо від тренувального процесу.

З часом, поки алгоритм вчитися, помилка моделі в навчальних даних зменшується, як і помилка в наборі тестових даних. Якщо алгоритм тренується занадто довго, продуктивність в наборі навчальних даних може почати знижуватися, тому що модель перенавчається і вивчає нерелевантні деталі і шум в наборі навчальних даних. У той же час помилка для тестового набору теж починає рости, оскільки здатність моделі до узагальнення зменшується.

Саме правильне рішення – це знаходження точки безпосередньо перед тим, як помилка в наборі тестових даних починає збільшуватися, коли модель має гарні результатами як для навчального набору даних, так і для поки невидимого тестового набору даних.

Однак на практиці цей метод підходить не завжди, оскільки вибір точки для зупинки з використанням набору тестових даних означає, що набір тестів більше не «невидимий» або не є окремою об’єктивною мірою. Деякі характеристики цих даних просочилися в процедуру навчання.

Як перенавчання, так і недостатнє навчання можуть привести до зниження продуктивності моделі. Але, безумовно, найпоширеніша проблема в прикладному машинному навчанні – це перенавчання.

Перенавчання є проблемою, тому що оцінка алгоритмів машинного навчання на даних навчання відрізняється від оцінки, яка нас цікавить найбільше, а саме, наскільки добре алгоритм працює з невидимими даними.

Існує два важливих метода, які можна використовувати при оцінці алгоритмів машинного навчання для обмеження перенавчання:

1. Метод передискретизації для оцінки точності моделі;
2. Утримання набору даних перевірки.

Найпопулярнішим методом передискретизації є перехресна перевірка в k-кратному порядку. Це дозволяє навчати і тестувати модель k-раз на різних підмножинах навчальних даних і будувати оцінку продуктивності моделі машинного навчання на невидимих даних. Набір даних перевірки – це просто підмножина навчальних даних, які не подаються в алгоритм навчання.

Після того, як були обрані і налаштовані алгоритми машинного навчання на наборі навчальних даних, перевірочні дані використовуються, щоб отримати остаточне об’єктивне уявлення про те, як моделі можуть працювати з невидимими даними. Використання перехресної перевірки є золотим стандартом в прикладному машинному навчанні для оцінки точності моделі по невидимим даним.

Регуляризація – це метод, який обмежує оптимізацію мережі, щоб перешкоджати появі складних моделей (тобто уникати запам’ятовування нерелевантних даних). Зазвичай метод регуляризації це Dropout, який являє собою процес випадкового відкидання деякої частки прихованих нейронних елементів в кожній ітерації під час фази навчання (тобто установка відповідних ваг в 0) і / або рання зупинка, яка полягає в зупинці тренування, перш ніж з’являється шанс перенавчитися.

Для цього вираховуються втрати на стадії навчання і тестування щодо кількості ітерацій навчання. Навчання припиняється, коли помилка під час тестування починає збільшуватися.

Цікаво вивчити можливості і властивості такого навчального процесу з точки зору найбільш ефективного використання довідкової інформації, доступної в описах, а також вивчити схеми поглибленого навчання, що враховують ступінь близькості елементів різні класи в просторі функцій.

1.5 Постановка задачі дослідження

Класифікація в базах даних зображень з використанням нейронних мереж є актуальним завданням в задачах обробки і розпізнавання. Тому ставиться завдання, яке полягає в тому, щоб розробити алгоритм кластеризації на основі даної бази даних зображень, використовуючи метод виявлення ОТ і обчислення дескрипторів ORB і класифікації з використанням нейронних мереж.

Для цього необхідно вирішити такі завдання:

– проаналізувати існуючи методи структурної класифікації зображень;

– розробити методи попередньої обробки отриманих дескрипторів з метою їх згортки або аналізу з метою оптимізації нейронної мережі;

– розробити алгоритм для кластеризації отриманих дескрипторів;

– реалізувати та дослідити результативність комбінованої комп’ютерної моделі для визначення ОТ для бази зображень, розрахунку їх дескрипторів та подальшої обробки в нейронній мережі з різними варіантами попередньої обробки.

2 ЗАСТОСУВАННЯ МЕРЕЖІ КОХОНЕНА У МОДЕЛІ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ

2.1 Аналіз та формалізація мережі Кохонена для задачі класифікації

Беручи до уваги особливості завдання, в роботі була використана нейронна мережа Кохонена. Нейронні мережі Кохонена – це клас мереж, основним елементом яких є шар Кохонена. Шар Кохонена, в свою чергу, складається з лінійних формальних нейронів.

Використання нейронних мереж Кохонена в класифікації баз даних великих зображень вимагає деякої формалізації. Кожен об’єкт представляється як деякий вектор, що надходить на вхід нейронної мережі (рис. 2.1). Кількість нейронів у вхідному шарі визначається кількістю компонентів цього вхідного вектора. Кількість виходів визначається кількістю класів, тобто, якщо є тільки M класів, то число нейронів у вихідному шарі також дорівнюватиме M. Таким чином, кожен нейрон у вихідному шарі «відповідає» за свої клас. Значення, взяті нейронами у вихідному шарі, відображають, наскільки близький вхідний вектор об’єкта, класифікованого на вході, згідно нейронної мережі Кохонена, того чи іншого класу. Чим більше «впевненість» в тому, що об’єкт належить класу, тим більше значення нейрона відповідного класу. Іноді використовується спеціальна функція активації, яка прирівнює суму виходів всіх нейронів одиниці [29]. У цьому випадку кожен вихід можна інтерпретувати як імовірність того, що об’єкт належить даному класу.

Ідея мережі вперше була запропонована фінським вченим Т. Кохоненом. Підвищена живучість досягається за рахунок розподіленої пам’яті, що пов’язано з тим, що кластер нейронів відповідає за класифікацію вхідного вектора. Ця структура дозволяє уникнути катастрофічної деградації, якщо один з нейронів виходить з ладу.

Багато типів мереж Кохонена відрізняються методами коригування вхідних ваг суматорів і вирішуваних завдань [3]. Найбільш відомі:

– мережі векторного квантування сигналів [3], тісно пов’язані з найпростішим базовим алгоритмом кластерного аналізу (метод динамічних ядер або K-середніх);

– самоорганізовані карти Кохонена (SOM) [1, 2, 4, 6];

– мережі векторного квантування за допомогою вчителя (LVQ) [1].

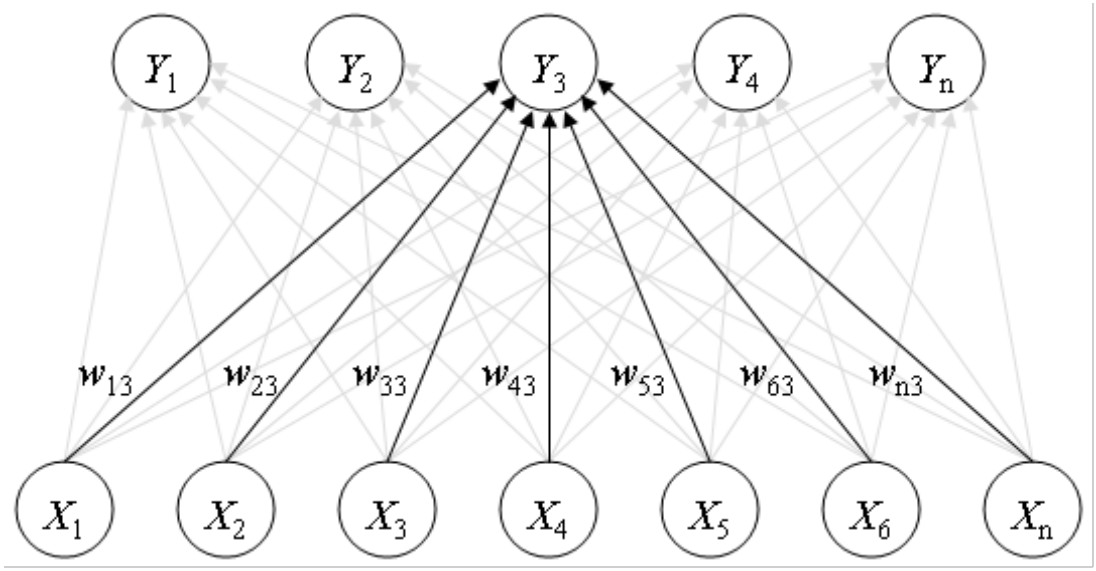

Рисунок 2.1 – Приклад мережі з п’ятьма кластерними одиницями Y і семи вхідними одиницями X

Варто відзначити, що існує більш проста реалізація нейронної мережі Кохонена, звана «переможець отримує все». У цьому випадку кожен нейрон вихідного рівня може приймати значення або нуль, або одиницю. В цьому випадку для одного вхідного вектора одиниця може дорівнювати одному і тільки одному нейрона вихідного шару, тобто один об’єкт не може належати двом класам одночасно.

Самоорганізовані карти є одним з видів алгоритмів нейронної мережі. Основна відмінність між цією технологією і нейронними мережами, яких навчають за алгоритмом зворотного поширення, полягає в тому, що метод навчання використовується без учителя, тобто результат навчання залежить

тільки від структури вхідних даних. Нейронні мережі цього типу часто використовуються для вирішення різних завдань.

Самоорганізована карта Кохонена – це конкурентоспроможна нейронна мережа з навчанням без вчителів, яке виконує завдання візуалізації і кластеризації. Це метод проектування багатовимірного простору в простір з меншим виміром (найчастіше, двовимірним), він також використовується для вирішення задач моделювання і прогнозування. Важливою особливістю карт Кохонена є те, що об’єкти з однаковими характеристиками (за всіма компонентами) розміщуються поруч, утворюючи кластери.

Самоорганізована карта складається з компонентів, які називаються вузлами або нейронами. Їх кількість вказує аналітик. Мережева взаємодія починається з ініціалізації карти, тобто початкового завдання векторів ваги для вузлів. Набір ваг структурований як матриця

$$W = \begin{pmatrix} W^{11}, W^{12}, ..., W^{1m} \\ W^{21}, W^{22}, ..., W^{2m} \\ ..... \\ W^{k1}, W^{k2}, ..., W^{km} \end{pmatrix},$$

елементами якої є вектори вагових коефіцієнтів $W^{ij} = (w_1^{ij}, w_2^{ij}, ..., w_n^{ij})$, що масштабують відповідні елементи вхідного вектора.

На відміну від інших типів нейронних мереж, карта самонавчання Кохонена не вимагає вихідного вектора. Навчання здійснюється шляхом зміни компонентів вектора кожного з вузлів мережі відповідно до значень компонентів вхідного вектора.

Відомо структуру вхідних даних, так чи інакше побудована початкова версія карти. В процесі навчання вектори ваг вузлів наближаються до вхідних даних. Для кожного спостереження вибирається найбільш схожий за вагою вузол, і значення його вагового вектора наближається до спостереження. Крім того, вектори ваг декількох вузлів, розташованих

поблизу, наближаються до спостереження, тому, що якщо два набору спостережень схожі в наборі вхідних даних, близькі вузли будуть відповідати їм на карті. Циклічний процес навчання мережі, яка сортує вхідні дані, закінчується після того, як карта досягла допустимої помилки (попередньо визначеної аналітиком), або після виконання попередньо визначеного числа ітерацій.

Правильно обраний метод ініціалізації ваг карти може значно прискорити навчання і привести до кращих результатів. Є три способи ініціалізації початкових ваг:

– ініціалізація випадковими значеннями, коли всім ваг задані невеликі випадкові величини;

– ініціалізація прикладами, де значення випадково вибраних прикладів навчальної вибірки наведені в якості початкових значень;

– лінійна ініціалізація. В цьому випадку шкали наповнюються значеннями векторів, лінійно впорядкованих вздовж лінійного підпростору, що проходить між двома основними власними векторами вихідного набору даних.

У нашому випадку на початку навчання компоненти вектора ваги кожного з вузлів задаються випадковим чином. Під час ітерацій значення ваг стабілізуються, утворюючи зони, близькі за значенням до кожного набору векторів з навчального набору.

Розглянемо класифікаційну модель, коли для кожного класу, заданого еталоном, він попередньо синтезується «центром», а класифікація здійснюється шляхом посилання на клас з найближчим центром [6].

Нехай $W = \{x \mid x \in R^n\}$, $W \subseteq R^n$ – простір структурних ознак (дескрипторів) зображень. База описів зразків (еталонів) $Z \subset W$ задана у вигляді множини $Z = \{Z^j\}_{j=1}^{J}$, $s_j = \text{card } Z^j$, $s = \text{card } Z$, $s = \sum_j s_j$. При цьому кожна ознака $x_i^j \in Z^j$ асоціюється з еталоном Z^j в плані включення до його складу, бо база зразків задана.

Розглянемо процедуру навчання мережі в наступному вигляді:

1. Ініціюємо матрицю $M = \{m_j\}_{j=1}^{J}$ ваг, рядками якої є формовані вектора нейронів m_j – центрів класів так, що $m_j = x_i^j$, $x_i^j \in Z^j$, де i – номер довільного вектора з класу зразків Z^j;

2. Сформуємо елементи навчальної множини у вигляді сукупності $Z = \{Z^j\}_{j=1}^{J}$ всіх s елементів бази еталонів;

3. Виберемо поточний елемент $z \in Z$, для кожного $j \in [1,2,...,J]$ обчислимо відстань $q_j = \rho(z, m_j)$ і визначимо клас d нейрона-переможця: $d = \arg \min_j q_j$. Вид метрики визначається властивостями дескрипторів;

4. Обчислимо зміни ваг для нейронів вихідного шару мережі

$$\Delta m_j = h(j,d,t) \cdot \eta \cdot (z - m_j), \qquad (2.1)$$

де η – коефіцієнт швидкості навчання що задається; $h(j,d,t)$ – значення функції околиці для нейрона j в момент часу навчання t; зазвичай $h(j,d,t) = \exp[-\rho(j,d)/\sigma(t)]$ визначають у вигляді гаусової функції, а радіус $\sigma(t) = 1/\exp(t^{-2})$ околиці зменшують зі збільшенням параметра t часу навчання. $t = 1,...,s$;

5. Коректуємо матрицю ваг $M = M + \Delta M$ на кроці t;

6. Продовжуємо навчання (п. 3-5) до завершення списку Z;

7. Перевіряємо виконання критерію припинення навчання. Це може бути величина помилки кластеризації. При необхідності (невиконання умови зупинки) ми продовжуємо навчання за пунктом 2. Це може бути дублювання набору вхідних даних, вибір даних з набору в установленому або випадковому порядку.

Якість класифікації безпосередньо залежить від результатів навчання системи розпізнавання, від наявного набору характеристик структурних описів бази вибірки (навчальної вибірки).

Дослідники відзначають, що нормалізація даних (приведення векторної норми до одиниці) при використанні мереж Кохонена прискорює збіжність мережі до локального мінімуму помилки [31, 32]. Для мереж з невеликою кількістю нейронів (2-3) нормалізація вважається обов'язковою. При обробці нормалізованих даних мережеві нейрони нормалізуються під час навчання. У цій ситуації ми використовуємо ненормалізовані дані, щоб підтримувати достатню відстань між нейронами, щоб забезпечити ефективне розділення класів.

Критерієм для оцінювання якості класифікації та отримання помилки може бути значення, яке підраховує частку елементів навчальної вибірки Z, які потрапили «не в свої» класи. Воно обчислюється як

$$\beta = \sum_{j=1}^{J} (s_j - a_j) / s, \qquad (2.2)$$

де a_j – число ознак із загального їх числа s_j в описі еталона Z^j, віднесених в процесі класифікації до класу j. Значення β відображають рівень помилкових рішень при класифікації ознак множини Z. Чим ближче β до нуля, тим вище досягнуто якість класифікації на навчальній вибірці.

2.2 Прикладні аспекти оптимізації мережі Кохонена для структурних описів

Оптимізація нейронної мережі – це не тільки налагодження її коефіцієнтів для досягнення бажаного відгуку на вхідні дані, а й оптимізація всієї мережевої архітектури. Цей підхід включає в себе оптимізацію топології штучної нейронної мережі і її узагальнення для забезпечення ефективної

роботи в набагато більшому просторі, ніж це використовується при її навчанні.

Отже, в контексті оптимізації, вивчимо порівняльним чином два практичних варіанти побудови класифікатора, що розрізняються за кількістю нейронів, які модифікуються в процесі навчання:

– налаштування тільки нейрона-переможця;

– три нейрона, найближчих до нейрона-переможця, модифіковані;

Обговорювані варіанти навчання класифікатора принципово відрізняються лише п. 4, де по-різному обчислюється функція $h(j,d,t)$. У той же час, для досягнення збіжності повинна бути виконана умова $h(j,d,t) \in [0,1]$.

У той же час у версії 1 реалізований грубе розпізнавання (коригується тільки переможець, принцип «переможець бере все»). Версія 2 приписує елемент трьом уайближчим центрам (м'які обчислення, «переможець отримує більше»), а версія 3 реалізує найточнішу обробку, аналогічну невизначеному підходу, а саме відсилання аналізованого дескриптора до всіх кластерам одночасно, але з різними вагами, пропорційно відстані. Версії 2 і 3 дозволяють контролювати нейрони сусіди переможця і фактично реалізовувати принцип «нейронного газу», згідно з яким проводиться ранжування мережевих елементів [3, 6].

Деякі джерела рекомендують використовувати максимальну кількість нейронів на карті. У той же час початковий радіус навчання (neighborhood в літературі англійською мовою) значно впливає на здатність узагальнювати за допомогою отриманої карти. Коли кількість вузлів карти перевищує кількість прикладів в навчальній вибірці, успіх алгоритму в значній мірі залежить від відповідного вибору початкового радіуса навчання. Однак, коли розмір карти становить десятки тисяч нейронів, час, необхідний для вивчення карти, зазвичай дуже великий для вирішення практичних завдань, тому необхідно вибирати прийнятний компроміс при виборі кількості вузлів.

Тривіальний підхід до розширення сфери ефективного функціонування нейронної мережі – пряме застосування знань про все наборі вхідних даних

при формуванні архітектури та навчанні. Але цей підхід зазвичай недосяжний в основному з двох причин:

– брак знань про всі можливі варіації даних;

– великий обсяг роботи, який необхідно виконати в разі великих масивів вхідних даних.

Але в розглянутих випадках ці причини не заважають такому підходу до оптимізації. Тому, знаючи тип введених даних і витягуючи вигоду з того факту, що продуктивність мережі Кохонена в значній мірі визначається початковим умовами функціювання, які визначаються значеннями початкових центрів m_j в пункті 1 процедури навчання, ми вивчаємо такі варіанти центрів:

– довільно по одному дескриптор для кожного зображення еталону;

– за допомогою застосування спеціальної процедури формування центрів.

З огляду на бінарне представлення дескрипторів, для Z^i ми визначаємо вектор центру класу на основі логічного правила яке порівнює загальну кількість нулів і одиниць для кожного з 256 бітів в пошуку значення, яке є найбільш загальними для вектора в цілому

$$m_i(b)=\begin{cases}1, & \sum_{d=1}^{s(i)} x_d(b)^3 s(i)/2,\\ 0, & \sum_{d=1}^{s(i)} x_d(b)<s(i)/2,\end{cases}, \; x_d \in Z^i, \; b=1,...,256, \qquad (2.3)$$

де $x_d(b)$ – біт з номером b дескриптора с номером d в описі еталону.

Відповідно до (2.3) значення кожного з бітів центру m_i визначається значеннями відповідних бітів всіх дескрипторів, що належать еталону з номером i. Центр (2.3) відображає властивості еталона (класу).

Однією з очікуваних проблем для описаних варіантів побудови мережі є час обробки у відносинах до якості класифікації. Потенційна необхідність збільшувати кількість ОТ, через те що ORB інколи вибирає ОТ які не

релевантні класифікації зображення до класу і збивають результат при класифікації вносить необхідність розглянути альтернативи методу (2.3). Однією з цих альтернатив є збільшення кількості ключових точок для подальшої їх обробки та вилучення основних рис класу з кожного єталону.

В останні кілька років експерти працюючі зі згортковими нейронними мережами звернулися до методу глобального пуллінга за середнім значенням (GAP), щоб мінімізувати перенавчання за рахунок скорочення загальної кількості параметрів в моделі [33].

Основна ідея всіх методів згорткових нейромереж це узагальнення наявності ознак у вхідному зображенні. Одним з підходів до вирішення цієї чутливості є зменшення вибірки карт характеристик. Шари пуллінга (інакше підвибірки, суб-дискретизації) це підхід до зниження вибірки карт об’єктів шляхом підсумовування присутності характеристик на ділянках карти характеристик.

Два найпоширеніші методи пуллінга – це пуллінг за середнім значенням і пуллінг за максимальним значенням, які підсумовують середню присутність рис і максимальну присутність рис відповідно. Але замість зниження вибірки вхідної карти об’єктів фільтр глобального пуллінга стискає всю карту характеристик до одного значення.

GAP використовується для зменшення просторових розмірів тривимірного тензора та виконує екстремальний тип зменшення розмірності, коли тензор з розмірами h × w × d зменшується в розмірі, щоб мати розміри 1 × 1 × d. GAP зменшують кожну карту об’єктів h × w до одного числа, просто беручи середнє значення всіх значень hw.

Глобальний пуллінг може використовуватися в моделі для агресивного підсумовування наявності характеристики на зображенні. Він також іноді використовується в моделях в якості альтернативи використанню безлічі пов’язаних шарів для переходу від карт характеристик до вихідного прогнозу для моделі.

Таким чином, карти характеристик можуть бути легко інтерпретовані як карти впевненості категорій. Такий підхід підсумовує просторову інформацію, таким чином, отримуючи модель більш стійку до просторовим перетворенням вхідних даних. Крім того, це допомагає уникнути перенавчання.

З огляду на отримані з ОТ бінарні дескриптори, та переносячи принципи екстремального типу зменшення розмірності GAP, визначемо вектор p^i на основі логічного правила.

$$p^i = \sum_{j=1}^{k} Z^{ij} / K \ . \tag{2.4}$$

Відповідно до (2.4) значення кожного з елементів p^i визначається нормованими за K значеннями відповідних бітів всіх дескрипторів в кожному стовпці з номером j. Отриманий за (2.4) вектор згорнутий для досягнення максимальної виразності властивостей кожного еталона, а отже і класу.

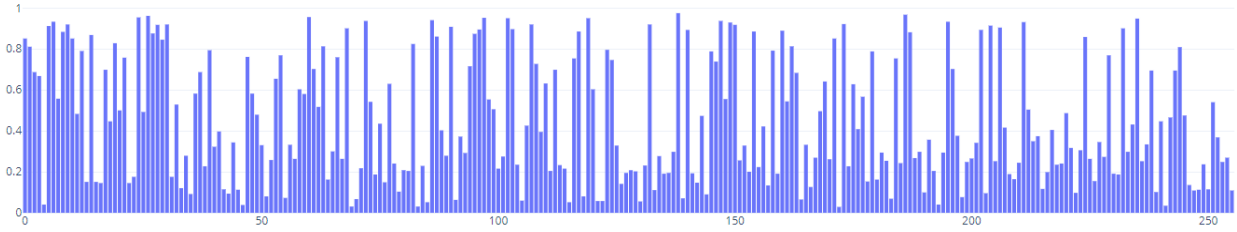

Рисунок 2.3 – Приклад вектору p^i з ймовірністю отримання одиниці

Вектор p^i фактично є одновимірним вектором з узагальненою версією характеристик зображення, через подання відсоткової ймовірності появи одиниці в кожному з 256 бітів (рис. 2.3).

Отримані вектори можна легко використовувати в будь-яких типах і варіантах нейронних мереж. Це новий метод, заснований на принципах GAP, що дозволяє отримати значне зниження часу навчання за рахунок обробки одного вектору, там де раніше оброблялися десятки або сотні векторів .

2.3 Аналіз метрик для визначення подібності ознак

Відзначимо, що в розд. 3 процедури навчання мережі Кохонена, необхідно визначити метрику для розрахунку різниці між двома дескрипторами. У методі Кохонена природно використовувати евклідову метрику або її квадрат.

Ця метрика (рис. 2.2) може бути використана для розрахунку відстані між об’єктами, що описується кількісними, якісними і дихотомічними ознаками. Її використання доцільне, коли ознаки однорідні за смисловим навантаженням і однаково важливі для вирішуваної проблеми.

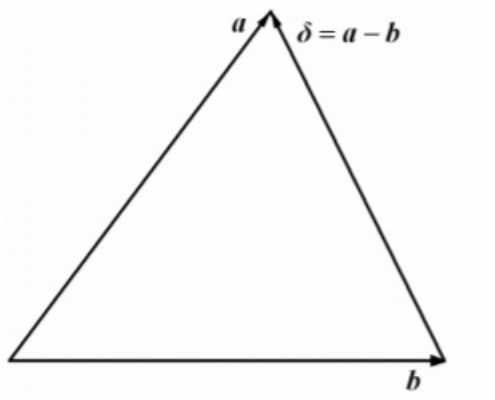

Рисунок 2.2 – Евклідова відстань

Евклідова метрика є найбільш часто використовуваної мірою відстані. Це геометрична відстань в багатовимірному просторі і розраховується вона наступним чином:

$$L(S_i, X_j) = \sqrt{\sum_{k=1}^{n} (S_{ik} - X_{jk})^2}\ .$$

Операції зведення в ступінь і добування кореня не завжди зручно використовувати при визначенні відстаней, оскільки вони є нелінійними операціями. Тому для визначення відстаней в просторі зображення часто

використовується сума модулів різниць між відповідними компонентами n-мірних векторів:

$$L(S_i, X_j) = \sum_{k=1}^{n} |S_{ik} - X_{jk}|.$$

Природно, з геометричної точки зору евклідова відстань може бути не кращим варіантом, якщо характеристики вимірюються в різних одиницях. Щоб виправити ситуацію, використовується нормалізація кожного еталону. Використання цієї метрики виправдано в наступних випадках:

– властивості (ознаки) об’єкта однорідні по фізичному змісту і однаково важливі для класифікації;

– простір ознак збігається з геометричним простором.

Квадрат евклідової відстані використовується, коли ви хочете надати більше значення більш віддаленим об’єктам. Це відстань розраховується наступним чином:

$$\rho(x,m_i)=\sum_{b=1}^{256}(x(b)-m_i(b)|)^2. \qquad (2.5)$$

Саме ця метрика найбільш підходить для обробки даних дескриптора, оскільки обробляються реальні дані (де 256 – це розмір детектора ORB).

У той же час при роботі з бітовими даними, більш ефективно використовувати метрику відстані Хеммінга (відстань) з точки зору обчислень.

В теорії інформації відстань Хеммінга, це відстань між двома рядками однакової довжини, а саме – кількість позицій, на яких відповідні символи різні. Іншими словами, вона вимірює мінімальну кількість підстановок, необхідних для зміни одного рядка в інший, або мінімальну кількість помилок, які могли перетворити один рядок в інший.

У більш загальному контексті відстань Хеммінга є однією з декількох рядкових метрик для вимірювання відстані редагування між двома послідовностями [3].

$$\rho(x,m_i)=\sum_{b=1}^{256}|x(b)-m_i(b)|. \qquad (2.6)$$

Ця міра найчастіше використовується для виявлення відмінностей між об’єктами, заданими дихотомічними ознаками, і інтерпретується як число відмінностей в значеннях ознак в розглянутих об’єктах. Для дихотомічних ознак це відповідає квадрату євклидової відстані.

У результаті обробки кожного зразка відповідно до процедури Кохонена отримуємо кластерне уявлення $h[Z^i]=(h^i_1,...,h^i_J)$ еталона, де h^i_a – цілі числа. Воно відповідає розподілу елементів множини Z^i за класами еталонів.

3 РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ НА ПІДСТАВІ КОМП'ЮТЕРНОГО МОДЕЛЮВАННЯ

3.1 Обґрунтування вибору середовища програмної реалізації

В рамках магістерської роботи був розроблений алгоритм класифікації зображень з використанням мережі Кохонена. Для реалізації було обрано середовище Visual Studio Community 2019. Це обумовлено тим, що Visual Studio дозволяє використовувати високопродуктивну мова C ++ [34].

C ++ це одна з найбільш широко використовуваних мов програмування в світі. Ця мова більш гнучка, ніж інші мови, тому що її можна використовувати для створення самих різних додатків.

C ++ підтримує різні стилі і технології програмування, включаючи традиційне директивне програмування, ООП, метапрограмування (шаблони, макроси) та інші.

Передбачуване виконання програм, що використовуються на мові, є важливою перевагою для побудови систем реального часу. Весь код неявно генерується компілятором для реалізації мовних можливостей (наприклад, при перетворенні змінної в інший тип), зазначених в стандарті. Крім того, місця, де виконується код, строго визначені. Це дозволяє виміряти або розрахувати час реакції програми на зовнішні події.

Мова підтримує поняття фізичної (постійної) і логічної (мінливої) сталості. Це робить програму більш надійною, оскільки дозволяє компілятору діагностувати помилкові спроби змінити значення змінної. Оголошення сталості дає програмісту, який читає текст програми, додаткове розуміння правильного використання класів і функцій, а також може бути підказкою для оптимізації. Перевантаження функцій-членів на основі сталості дозволяє визначити зсередини об'єкта об'єкти виклику методу (постійний для читання, і не постійний для зміни).

Використовуючи шаблони, можна створювати універсальні контейнери і алгоритми для різних типів даних, а також спеціалізуватися і обчислювати їх на етапі компіляції.

C++ призначений для того, щоб дати програмісту максимальний контроль над усіма аспектами структури і виконання програми. Жодна з мовних можливостей, що призводять до додаткових накладних витрат, не є додатковою – ви можете використовувати мову для максимізації ефективності програми, коли це необхідно.

Програмна реалізація детектора ORB була відображена в багатьох сучасних бібліотеках комп'ютерного зору, таких як OpenCV [1, 2, 6].

Бібліотека OpenCV написана на C++ і популярна завдяки своїй відкритості і можливості безкоштовного використання в освітніх і комерційних цілях. Вона також підтримує можливість використання бібліотеки на мовах програмування Python, Java, Ruby, Matlab, Lua.

OpenCV в основному орієнтований на додатки реального часу і використовує інструкції MMX і SSE, коли це можливо. В даний час активно розробляються повнофункціональні інтерфейси CUDA і OpenCL. Існує більше 500 алгоритмів і приблизно в 10 разів більше функцій, які становлять або підтримують ці алгоритми.

Бібліотека має більше 2500 оптимізованих алгоритмів, які включають в себе повний набір як класичних, так і сучасних алгоритмів комп'ютерного зору і машинного навчання. Ці алгоритми можуть використовуватися для виявлення і розпізнавання осіб, ідентифікації об'єктів, класифікації дій людини в відео, переміщення камер, відстеження рухомих об'єктів, вилучення тривимірних моделей об'єктів, створення тривимірних хмар точок з стереокамер, об'єднання зображень для отримання високого розрізняння або зображення всієї сцени, знаходження схожих зображень в базі даних зображень, відстеження руху очей, розпізнавання пейзажів і встановлення маркерів ОТ. Бібліотека широко використовується компаніями, дослідницькими групами та державними органами.

Як правило, бібліотека OpenCV реалізує тільки основні операції, які використовуються в задачах комп'ютерного зору. Таким чином, її можна розглядати як низько рівневу бібліотеку комп'ютерного зору. Для вирішення серйозних проблем ви повинні створювати свої власні складні додатки на основі методів, що надаються бібліотекою.

Основні модулі бібліотеки можна віднести до 4 груп (розділів):

– модулі core, highgui, що реалізують базову функціональність (базові структури, математичні функції, генератори випадкових чисел, лінійна алгебра, швидке перетворення Фур'є, введення / виведення зображень і відео, введення / виведення в форматах XML, YAML і ін.);

– модулі imgproc, features2d для обробки зображень (фільтрація, геометричні перетворення, перетворення колірних просторів, сегментація, виявлення особливих точок і ребер, контурний аналіз і ін);

– модулі video, objdetect, calib3d (калібрування камери, аналіз руху і відстеження об'єктів, обчислювати координати в просторі, побудова карти глибини, детектування об'єктів, оптичний потік);

– модуль ml, який реалізує алгоритми машинного навчання (метод найближчих сусідів, наївний байесовский класифікатор, дерева рішень, бустінг, градієнтний бустінг дерев рішень, випадковий ліс, машина опорних векторів, нейронні мережі).

Використання бібліотеки OpenCV дозволило в короткі терміни створити дослідницьку програму і повністю виправдало себе. Набір процедур бібліотеки досить великий, і в переважній більшості випадків, коли є необхідність в будь-яких діях з зображеннями, існують відповідні процедури для підтримки цих дій.

3.2 Програмна реалізація

У C ++ в операційній системі Windows реалізована модель класифікації зображень, заснована на присвоєння отриманих з об'єкту ОТ одному з центрів класів з використанням нейронної мережі Кохонена. Обчислення ОТ, отримання і конвертація дескрипторів ОТ реалізовані у вигляді програмного забезпечення на основі модуля features2d для багатоплатформової бібліотеки OpenCV.

Функціональні можливості цього модуля полягають в розробці проекту зберігання даних зображення, визначенні ОТ на зображенні або серії зображень, збереженні ключових точок і дескрипторів, знаходженні найкращої відповідності для кожного дескриптора в наборі запитів і виведення зображень і отриманих даних після їх обробки.

Для роботи алгоритму і самої програми використана база зображень з еталонними зображеннями метеликів. Набір даних, що використовується в цьому проекті, називається набором даних Leeds Butterfly Dataset. У цьому наборі даних є десять видів метеликів із зображеннями та текстовими описами.

Приклад різних категорій показаний на рисунку 3.1. Походження цих зображень було зібрано з Google Images відповідно до наукової назви виду.

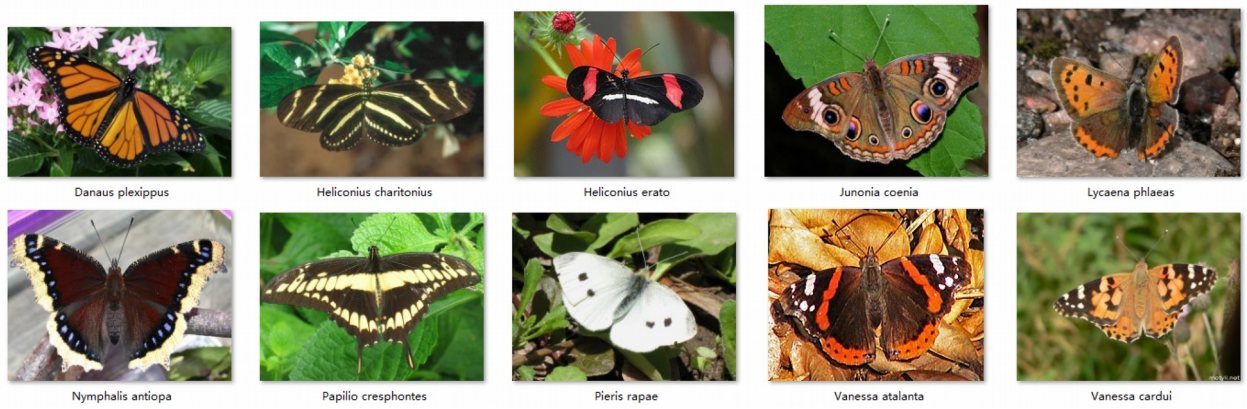

Рисунок 3.1 – Категорії метеликів в Leeds Butterfly Dataset

Всього є 832 зображення, а розподіл зображень різних видів відносно збалансований, як показано на рисунку 3.2. Це вигідно для алгоритмів машинного навчання для більш рівномірного вивчення особливостей кожної категорії.

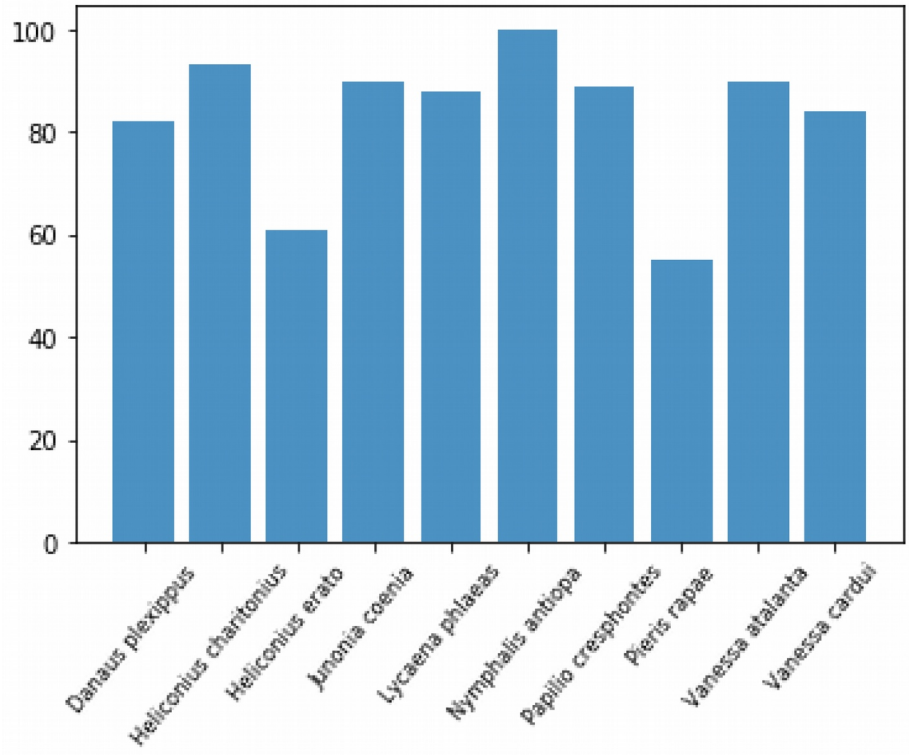

Рисунок 3.2 – Розподілення категорій метеликів в Leeds Butterfly Dataset

Варто зазначити, що ОТ, отримані методом ORB, природно підходять більше для складних зображень, як обрані зображення метеликів (рис. 3.3). Для описаних експериментів було обрано 4 категорії. Приклади категорій зображень, використовуваних для визначення ОТ у роботі, можуть бути знайдені в Додатку А.

Метелики є цікавою областю для дослідження, оскільки візуально вони мають як відмінні, так і загальні властивості. У порівнянні з категоріями «верхнього рівня» (людина, автомобіль, велосипед і т. д.) у метеликів є лиш декілька «глобальних» рис, як наприклад, конфігурація деталей у авто, яку можна використовувати для розмежування між видами.

Рисунок 3.3 – Приклад зображення з координатами виділених ОТ

Дескриптори мають вигляд бінарного вектора розміром 256, кодуються в Open CV у вигляді типу uchar (unsigned char – тип даних в C++, використовують для зберігання символів, обсяг 8 біт, значення 0 ... 255), а не в бітах. Дані зберігаються в матриці, де кількість рядків дорівнює числу виявлених дескрипторів, а число стовпців дорівнює 32 (256 біт дескриптора трансформуються в 32 uchar).

Також були програмно реалізовані додаткові функції, такі як побудова графіків і гістограм дескрипторів, конверторів даних для внутрішнього використання в алгоритмах і для відображення інформації для користувача, щоб оцінити продуктивність алгоритмів і порівняти їх.

3.3 Інструкція користувача

До запуску програми потрібно спочатку підключити бібліотеку OpenCV, додатком до якої є дана робота. Для цього потрібно виконати описану послідовність дій.

1. Завантажте Visual Studio Community 2019 (рис. 3.4).

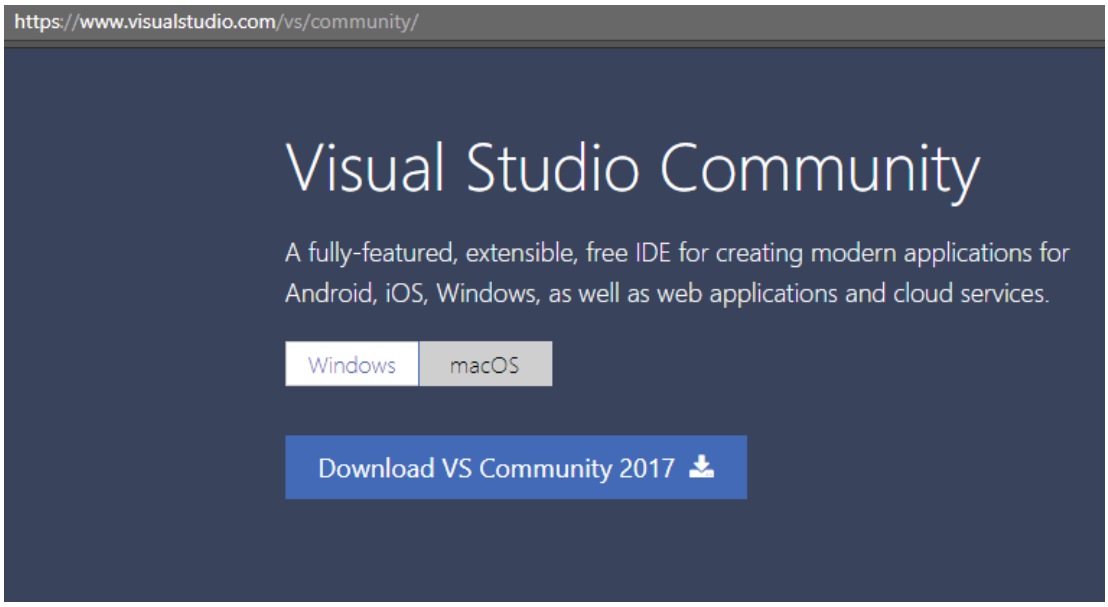

Рисунок 3.4 – Сторінка завантаження Visual Studio Community

2. При запуску інсталятор завантажуватиме, як правило, 3 Гб файлів. Інсталяція добре описана (рис. 3.5). Просто натисніть «Далі» та «Далі» з експресними налаштуваннями.

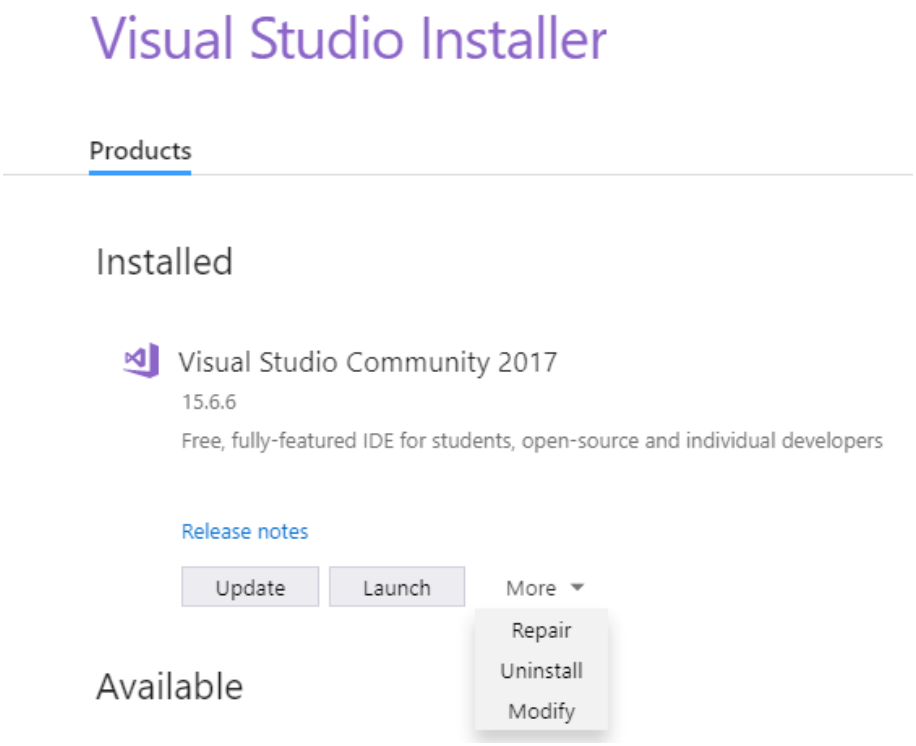

Рисунок 3.5 – ІнсталяторVisual Studio Community

3. Завантажте версію 3.4.1 OpenCV із офіційної сторінки (рис. 3.6) та встановіть файли бібліотеки.

Рисунок 3.6 – Сторінка завантаження OpenCV

4. Включіть OpenCV до системного шляху (рис. 3.7). Рекомендується розпакувати OpenCV архів у кореневому каталозі, наприклад C:\.

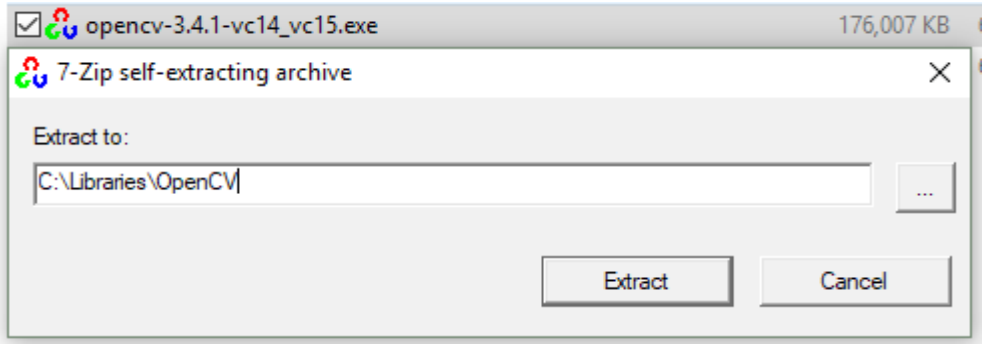

Рисунок 3.7 – Розпаковка архіву OpenCV

5. Відкрийте вікно Environment Variables та натисніть ‘New’ щоб додати нову змінну середовища у меню System Variables (рис. 3.8).

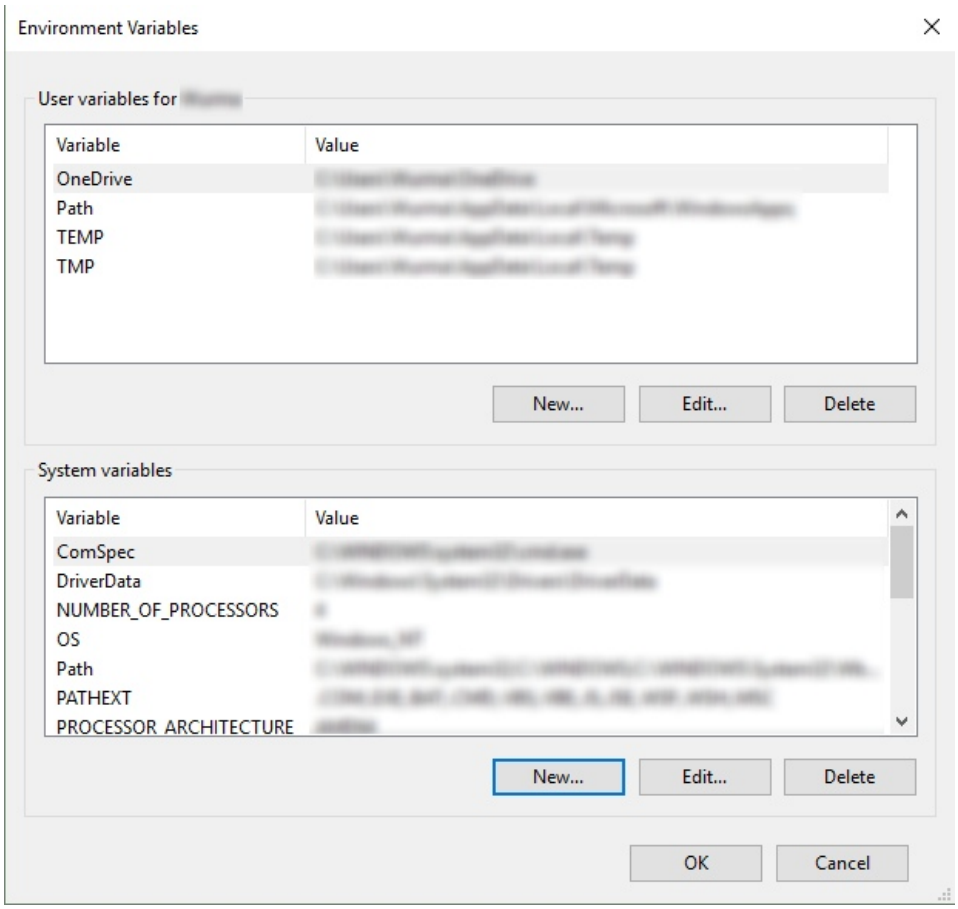

Рисунок 3.8 – Вікно Environment Variables

6. Скопіюйте та вставте шлях до папки bin всередині розпакованого пакета OpenCV (рис. 3.9). Шлях буде виглядати аналогічно C:\opencv\build\x64\vc14\bin.

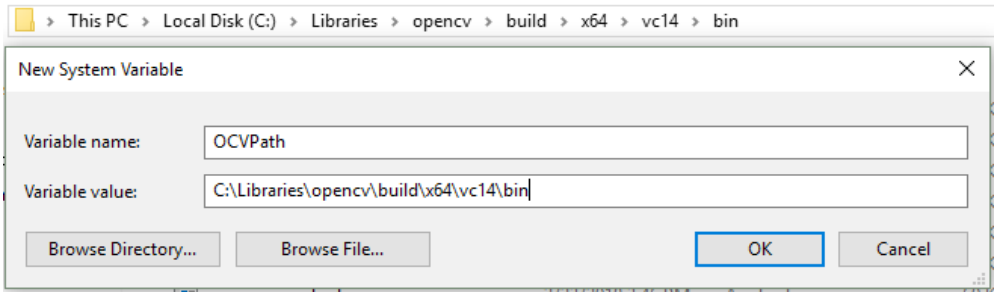

Рисунок 3.9 – Запис змінних середовища

7. Натисніть кнопку OK і закрийте діалогове вікно змінної середовища, знову натиснувши кнопку OK.

8. Відкрийте файл проекту.

9. Перш ніж продовжувати, переконайтеся, що платформа рішення була обрана правильно. В роботі використана 64-розрядня версія. Включіть OpenCV до Visual Studio.

10. Зайдіть до Properties проекту, потім C/C++ > General. Скопіюйте шлях до include папки opencv і вставте її в Additional Include Directories (рис. 3.9). Шлях виглядатиме як C:\opencv\build\include. Потім натисніть кнопку Apply.

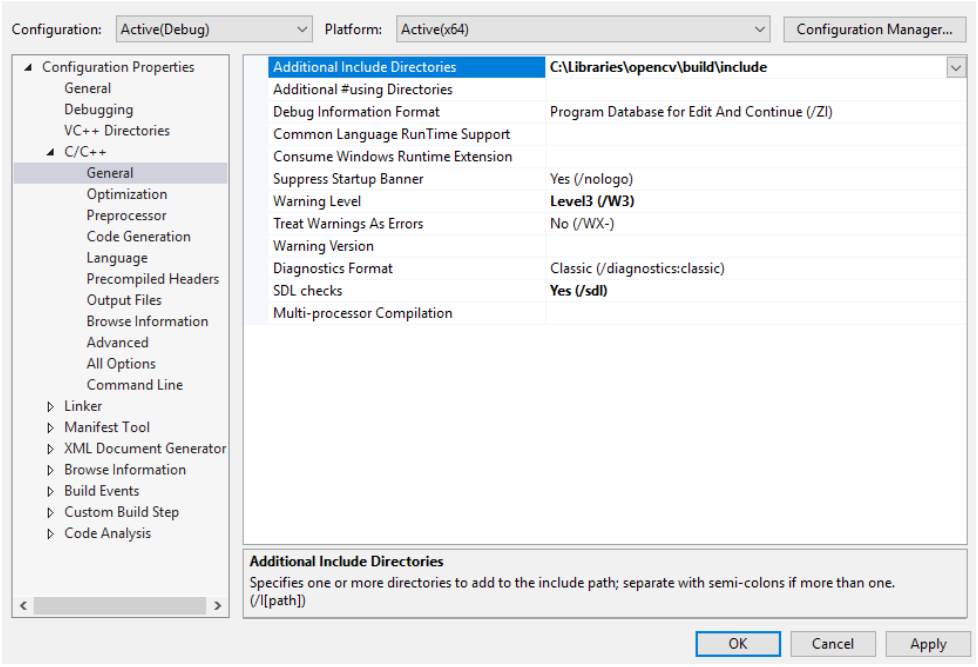

Рисунок 3.9 – Additional include directories

11. Перейдіть до linker > General. Скопіюйте шлях до папки, що містить файли opencv lib, та вставте її в Additional Library Directories (рис. 3.10). Шлях виглядатиме як C:\opencv\build\x64\vc14\lib.

12. Натисніть кнопку Apply для виходу з меню та застосування усіх змінених налаштувань.

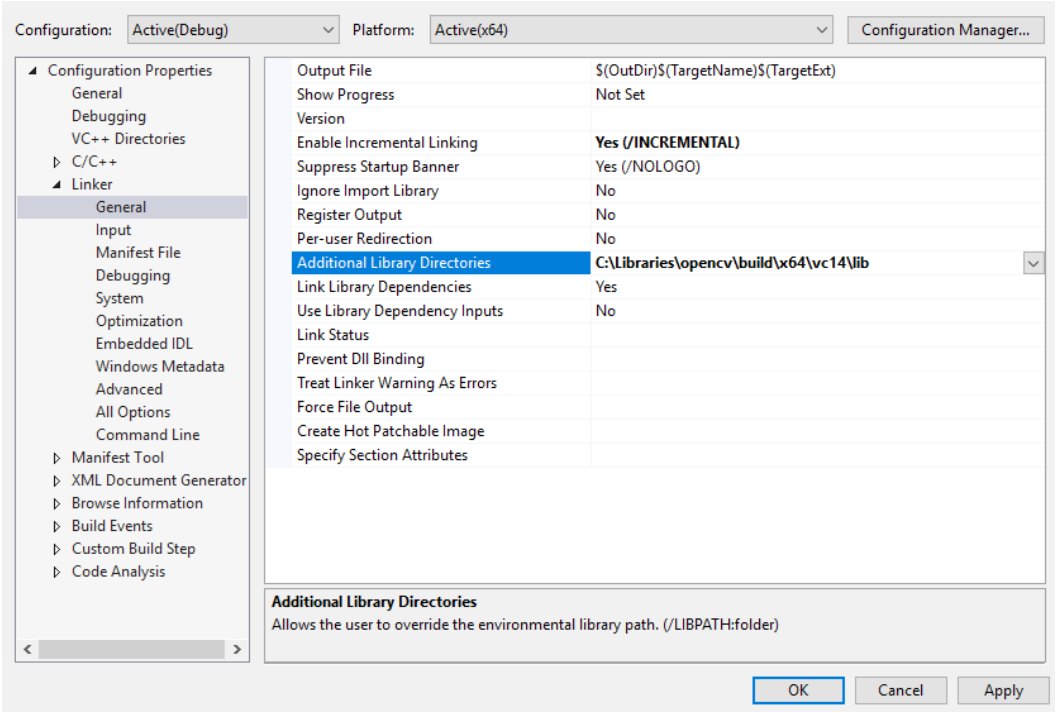

Рисунок 3.10 – Additional library directories

13. Перейдіть до Input (рис. 3.11). Змініть Additional Dependencies та вставте ім’я *.lib файлу. Виберіть файл *.lib відповідно конфігурації. Для режиму Debug файл закінчується на 'd', наприклад: opencv_world341d.lib.

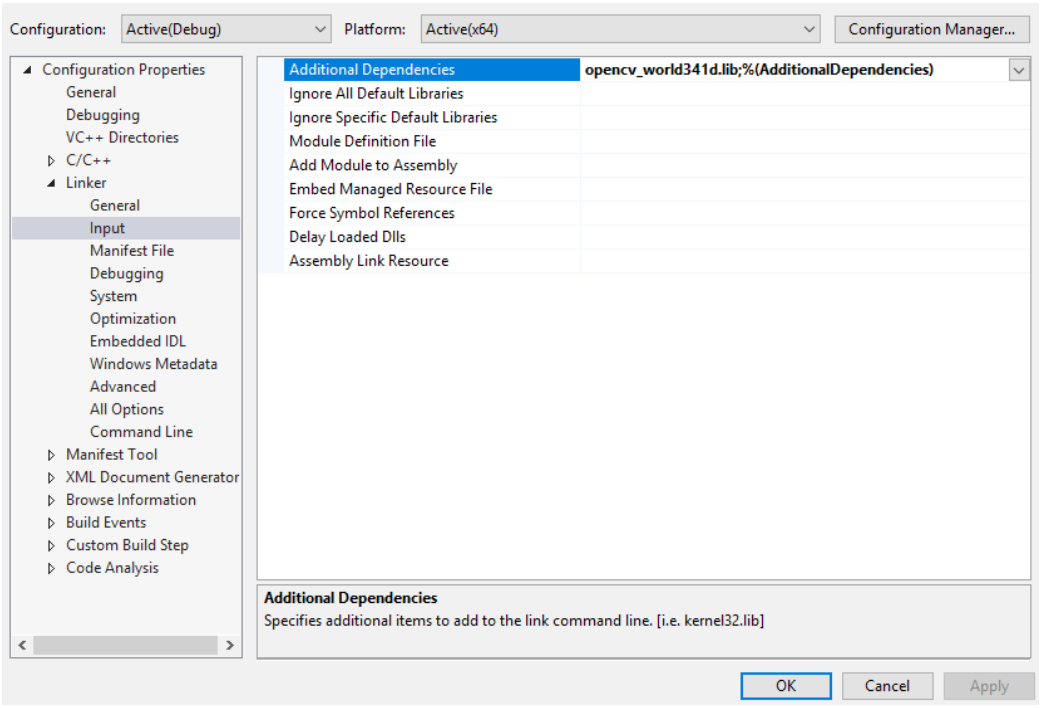

Рисунок 3.11 – Additional dependencies

14. Закрийте меню Properties натиснув OK. Перевірте роботу коду побудувавши його, натиснувши «Build Solution» або просто натиснувши на клавіатурі Ctrl + Shift + B.

Якщо все інструкції були виконані правильно, програма повинна бути побудована. Якщо в теці зображень є дійсні зображення, буде виконано алгоритм кластеризації. У вікні консолі зображається інформація про отримані дескрипторах і виконані процедури.

3.4 Тестування розробленої моделі

Кількість виявлених ОТ на заданій базі зображень коливалося в межах 375 ... 400 для тестування на великій базі дескрипторів та 100 ... 125 для тестування зменшеної кількості дескрипторів на швидкодію.

Оцінимо ефективність розпізнавання, використовуючи як описані опції мережевого навчання, що розрізняються за глибиною аналізу даних так і варіанти побудови класифікатора та обробки початкових даних. Критерієм оптимізації, як і в процесі навчання, є мінімізація функції (2.2), яка використовується як стандартна помилка.

Використовуючи раніше зазначену базу зображень (рис. 3.12), проведемо процедури обробки (2.3) та (2.4) для множин дескрипторів кожного з еталонів.

Кількість епох мережі Кохонена протестована на 100 та 200 ітераціях, для пошуку та досягнення балансу швидкості роботи і точності розпізнавання. Варіанти з підстроюванням одного нейрона переможця, чи навпаки, трьох нейронів, найближчих за відстанню до нейрона переможця, дозволяють виконувати розпізнавання швидше або точніше. Результати тестування приведені у відповідних експериментах.

Рисунок 3.12 – Тестова вибірка зображень

Слід зазначити, що над зображеннями не проводилося ніякої додаткової обробки або приведення до однакового розміру – він коливається у межах від 2000 пікселів до 400 пікселів. Інваріантність ORB до масштабу дозволяє відкинути ресурсомістку операцію конверсії розмірів для кожного зображення в базі даних.

Проведено ряд дослідів. Для першого, на визначеній базі зображень, використовуючи підхід «грубої сили», а не спираючись процедуру (2.3), беремо набір еталонів з кожного з чотирьох класів та навчаємося на базі їх дескрипторів.

Ефективність класифікації візуальних об’єктів по набору особливих точок з використанням мережі Кохонена безпосередньо залежить від наступних ключових взаємопов’язаних факторів: початкова база даних, застосовані методи попередньої обробки, такі як метод формування

дескрипторів чи початковий вибір центрів, метрика для порівняння дескрипторів, розмір кортежу адаптованих нейронів.

Навчання йде як на множині даних класу, так і між класами, що найбільш універсально, але для обраної задачі розпізнавання на невеликій базі дескрипторів зображень цей метод не підходить. Як видно з таблиці 3.1, множина дескрипторів ORB для конкретної бази зображень не дозволяє однозначно класифікувати множину ОТ кожного з еталонів. Причина – число ОТ, віднесених до певного класу, не є максимальним для багатьох еталонів.

Таблиця 3.1 – Розподіл ОТ за класами

Еталон	Кластери			
	0	1	2	3
Z^1	67	37	165	131
Z^2	77	49	157	117
Z^3	166	109	100	25
Z^4	120	127	84	47
Z^5	224	3	23	150
Z^6	213	10	48	83
Z^7	290	14	19	77
Z^8	152	48	117	58
Z^9	202	69	50	34
Z^{10}	211	76	91	22
Z^{11}	144	94	126	36
Z^{12}	108	31	92	133
Z^{13}	138	32	97	90
Z^{14}	156	108	65	47
Z^{15}	229	49	80	32

Залежність кластер – еталон отримана обчисленням максимальної кучності даних кожного класу у кластерах. Отримана помилка класифікації становить $\beta = 0{,}65$ за формулою (2.2) та 10 еталонів були віднесені до не свого класу. Кількість нейронів для кожного еталону коливалася у межах 375...400, а кількість ітерацій мережі дорівнювала 200. Час роботи такого алгоритму склав 14.94 с.

Отримано погану результативність розподілення за кластерами за рахунок поганої роздільності великої кількості даних дескрипторів та перенавчання на нерелевантних структурних описах. Це аргументує застосування методів попередньої обробки вхідних даних з метою покращення результатів.

Для поліпшення роботи алгоритму мережі Кохонена для кожного з еталонів обчислюємо центри за допомогою (2.3), та проведемо навчання і тестування на загальній базі дескрипторів. Спочатку для тестування роботи алгоритму застосуємо його до задачі класифікації у просторі, де кількість класів дорівнює кількості еталонів (табл. 3.2).

Як і раніше кількість нейронів для кожного еталону коливалася у межах 375...400, а кількість ітерацій мережі дорівнювала 200. Час роботи нейронної мережі з урахуванням процедури обробки масиву дескрипторів для пошуку вектора центру для кожного еталону склав 4.98 с.

Таблиця 3.2 – Розподіл ОТ за класами з центрами по (2.3)

Еталон	Класи			
	0	1	2	3
Z^1	279	33	27	61
Z^2	58	288	25	29
Z^3	36	91	228	23
Z^4	88	9	3	300

Значення помилки (2.2) складають $\beta = 0{,}31$. Але не зважаючи на це, спостерігається високий рівень розрізнення еталонів: максимуми значень знаходяться на діагоналях і значно перевищують інші елементи. Всі еталони віднесені до свого класу без помилок.

Використання мережі Кохонена при розрахунку центрів по методу (2.3) значно знижує похибку класифікації. Оцінимо і порівняємо тепер результативність класифікації з застосуванням обговорюваних двох варіантів навчання мережі, різних за глибиною аналізу через різне число настоюваних

нейронів мережі, різну кількість дескрипторів для кожного еталону, та різну кількість ітерацій навчання нейронної мережі.

У таблиці 3.3 представлено результати роботи методу на базі 15 зображень еталонів, вибраних із чотирьох класів.

Таблиця 3.3 – Порівняння розподілу ОТ з центрами по (2.3)

	Час, с	Загальна точність, %	Кількість помилкових ОТ	Кількість помилок у визначенні еталону
1 нейрон 400 д/е 200 ітерацій	15.1 с	67%	1897	2
3 нейрона 100 д/е 100 ітерацій	6.2 с	54%	2644	4
3 нейрона 100 д/е 200 ітерацій	7.93 с	61%	2242	2
3 нейрона 400 д/е 200 ітерацій	16.21 с	77%	1322	0
3 нейрон 400 д/е 100 ітерацій	13.7 с	72%	1609	1

Як бачимо, основна залежність результативності роботи – це кількість ітерацій роботи мережі. Зменшення кількості адаптованих нейронів майже не впливає на показники часу роботи мережі, але впливає на результативність отриманих даних класифікації, що можна бачити як по загальній точності, яка обчислена за формулою (2.2), так і по кількості помилок, які апарат класифікації зробив при віднесенні еталону до класу. Саме через це результати роботи методу при модифікації одного нейрону зведені тільки до великої кількості ітерацій та числа дескрипторів у еталоні.

У таблиці 3.4 приведені наочні результати ідентифікація класу з бінарною класифікацією «один проти всіх», з використанням трьох нейронів, 400 дескрипторів та 100 ітерацій.

У тренувальних даних обрано один еталон, який представляє клас, пошук якого проводиться. За допомогою логічного правила (2.3) отримуємо центр у вигляді вектора таких же розмірів та далі навчаємо мережу на масиві дескрипторів з приведенням їх до визначеного центру.

Таблиця 3.4 – Розподіл ОТ з пошуком 1-го класу з центрами (2.3)

Еталон	Кластери			
	0	1	2	3
Z^1	322	13	20	45
Z^2	348	0	48	4
Z^3	118	56	10	216
Z^4	57	119	96	106
Z^5	32	70	195	103
Z^6	32	154	100	68
Z^7	3	167	214	16
Z^8	134	1	21	219
Z^9	109	49	77	120
Z^{10}	70	225	17	88
Z^{11}	132	195	56	17
Z^{12}	243	45	35	41
Z^{13}	256	8	26	67
Z^{14}	146	103	68	59
Z^{15}	100	42	64	184

Як можна побачити, один клас розпізнається, а іншим бракує даних для коректної класифікації.

Отримана помилка в розпізнаванні навченого еталона (в таблиці 3.4 це Z^1, Z^2, Z^{12}, Z^{13}) $\beta = 0.23$. При цьому всі еталони віднесені до класу коректно. Отримані результати характерні і для випадку навчання на даних кількох класів. Багато також залежить від обраного на навчання класу, тестування було проведене з 4-ма класами зображень. Помилка коливалася до максимального значення в 0.39.

Слід відмітити, що у розглянутих варіантах процеси тренування мережі і розпізнавання займають великий проміжок часу. Цей показник погіршується і далі якщо збільшити кількість ОТ, епох або зображень, і обумовлено це тим, що для кожного зображення отримується та оброблюється велика база дескрипторів ОТ.

Розглянемо можливості скоротити час навчання для застосування методів, заснованих на обробці дескрипторів ORB в мережі Кохонена у великих базах зображень, де час обробки має велике значення, наприклад, у додатках реального часу або при роботі з відеоматеріалами (табл. 3.5). Природно, що маючи на руках центр, отриманий після обробки масиву дескрипторів за процедурою побітової обробки (2.3), його можна використовувати як самостійно для навчання та тестування роботи, не беручи вектори дескрипторів до уваги. Цей метод фактично вважає вектор-центр описом для всього зображення.

Протестуємо розглянуті варіанти побудови класифікаційної мережі .

Таблиця 3.5 – Порівняння розподілу векторів-центрів (2.3) за класами

	Час, с	Точність, %	Загальна кількість помилок
1 нейрон 400 д/е 200 ітерацій	2.27 с	80%	3
3 нейрона 100 д/е 100 ітерацій	2.36 с	66%	5
3 нейрона 100 д/е 200 ітерацій	2.48 с	80%	3
3 нейрона 400 д/е 200 ітерацій	2.77 с	80%	3
3 нейрон 400 д/е 100 ітерацій	2,58 с	66%	5

У таблиці 3.6 також можна побачити варіант застосування такої мережі з для вектору-центру отриманого за процедурою (2.3) з масиву розміром у 400 дескрипторів, настроюванням трьох нейронів та 100 ітерацій.

Таблиця 3.6 – Розподіл векторів-центрів (2.3) за класами

Еталон	Класи			
	0	3	2	1
Z^1 - 0	+			
Z^2 - 0	+			
Z^3 - 3		+		
Z^4 - 2		-		
Z^5 - 1				+
Z^6 - 1				+
Z^7 - 1			-	
Z^8 - 3		+		
Z^9 - 3		+		
Z^{10} - 2		-		
Z^{11} - 2		-		
Z^{12} - 0	+			
Z^{13} - 0	+			
Z^{14} - 2		-		
Z^{15} - 3		+		

Помилка становить $\beta = 0.3$. За результатами цього етапу дослідження можна побачити, що розширення набору адаптованих нейронів в процесі навчання не значно підвищує продуктивність мережі.

Якість класифікації у разі застосування всіх зазначених варіантів методу, заснованого на класифікації тільки отриманих за (2.3) векторів центрів порівняно нижче ніж при застосуванні їх у якості центрів мас для класифікації всього масиву дескрипторів еталону.

Ще однією проблемою є те, що мережа тепер працює з прямим віднесенням до класу, що подалі погіршує результати, отримані при практичному застосуванні, оскільки раніше кількість віднесених

дескрипторів еталону можна було сприймати як вірогідність приналежності еталону класу, тепер такої можливості немає.

Вирішити цю проблему без додавання деякої кількості дескрипторів назад до мережі не є можливим. А це додасть час обробки і поверне проблему часу роботи мережі. Отже становить інтерес вивчення методів попередньої обробки масиву дескрипторів з метою виділення з них ключових ознак.

Для цього скористаємося процедурою бітового аналізу (2.4) і обчислюємо з усіх дескрипторів кожного еталона вектор, з відсотковою ймовірністю появи одиниці в кожному з бітів. Далі, як і у попередніх експериментах, тестуємо мережу тільки на цих векторах (табл. 3.7).

Таблиця 3.7 – Порівняння розподілу векторів (2.4) за класами

	Час, с	Точність, %	Загальна кількість помилок
1 нейрон 400 д/е 200 ітерацій	2.82 с	94%	1
3 нейрона 100 д/е 100 ітерацій	2.48 с	66%	5
3 нейрона 100 д/е 200 ітерацій	2.57 с	66%	5
3 нейрона 400 д/е 200 ітерацій	2.84 с	94%	1
3 нейрон 400 д/е 100 ітерацій	2,54 с	80%	3

Результати обробки показують чітку залежність між коректною класифікацією та кількістю дескрипторів у кожному еталоні. Кількість ітерацій також виявилася важливим параметром, який здатен сильно

посилити класифікаційні якості мережі. Отримано якість класифікації яку можна порівняти з методом обробки на основі процедури (2.3), де масив дескрипторів для кожного еталону класифікується за допомогою обчисленого центру. Приклад класифікації (3 нейрона, 400 дескрипторів, 200 ітерацій) приведено у таблиці 3.8.

Таблиця 3.8 – Розподіл відсоткових векторів (2.4) за класами

Еталон	Класи			
	0	2	1	3
Z^1 - 0	+			
Z^2 - 0	+			
Z^3 - 2			-	
Z^4 - 1			+	
Z^5 - 3				+
Z^6 - 3				+
Z^7 - 3				+
Z^8 - 2		+		
Z^9 - 2		+		
Z^{10} - 1			+	
Z^{11} - 1			+	
Z^{12} - 0	+			
Z^{13} - 0	+			
Z^{14} - 1			+	
Z^{15} - 2		+		

Помилка класифікації становить $\beta = 0.06$.

Метод (2.4) дозволяє скоротити час роботи нейронної мережі в порівнянні з конкурентними методами без погіршення показників її роботи.

При цьому час обробки довелося знизити майже у 8 раз, що дозволяє потенційно використовувати зазначений алгоритм, як для дуже великих баз зображень, так і у додатках реального часу або при обробці відеосигналу.

Зазначений час роботи потенційно дозволяє обробляти кожен десятий кадр відео з частотою оновлення у 60 кадрів в секунду, або частіше, для більш поширеного відео з низькою частотою кадрів.

ВИСНОВКИ

У процесі виконання магістерської роботі проведено ряд досліджень, де з використанням алгоритмів кластеризації вирішена задача класифікації зображень на основі структурного опису у вигляді набору ключових точок, що визначаються детектором ORB.

Для цього аналізуються особливості детектора, пропонуються способи визначення центрів кластерів шляхом бітової логіки замість більш складних метрик та метод згортки масиву вхідних даних на основі GAP. Проведено порівняльний аналіз розроблених методів при різних умовах роботи класифікаційної мережі.

На практиці при побудові набору правил зазначених методів класифікації візуальних об’єктів необхідно звернути увагу на більш ефективний вибір і об’єднання як можливостей детектора ОТ, так і апарата нейронних мереж для досягнення продуктивної обробки на тестових базах зображень.

Дослідження підтвердило здатність успішно адаптувати мережу Кохонена до довільних наборів візуальних даних, особливо в разі успішного формування центрів первинного класу або виконання процесу попередньої обробки дескрипторів для знаходження відсоткової ймовірності одиниці.

Поряд з прямим присвоєнням дескрипторів, результати якого можна вважати не дуже вдалими, розглянутий метод розрізнення зображень за допомогою процедури (2.3) де вектори кожного еталону класифікуються як можна ближче до обрахованих центрів. Цей метод застосований у просторі кластер-еталон де кількість класів дорівнює кількості зображено показав найвищий рівень розрізнення еталонів, у порівнянні зі всіма розглянутими, і може використовуватися для ситуацій у яких потрібна якісна класифікація. Однак для застосування у задачах, де час обробки є дуже важливим, цей метод не підходить.

Метод заснований на тренуванні та тестуванні мережі з використанням тільки центральних векторів, він показав в експериментах досить високий рівень відмінності між протестованими зображеннями, при цьому час обробки значно зменшився. Недоліком цього методу можна вважати інваріантне віднесення до класу, на відміну від попереднього методу, де кількість віднесених ОТ можна вважати показником впевненості нейронної мережі.

Метод, заснований на побудові центрального дескриптора, з віднесенням до нього векторів-дескрипторів ОТ еталонів, був перевершений, як у швидкодії обробки даних на зазначеній базі зображень так і в якості розпізнавання методом заснованим на модифікованих принципах GAP.

Наукова новизна дослідження полягає в побудові методів структурної класифікації з використанням мережі Кохонена на основі бінарної логіки для знаходження центрів кластерів або відсоткової згортки, які забезпечують високу ефективність диференціювання класів при порівняно високій швидкодії що дозволяє застосовувати ці методи при обробці та класифікації в додатках реального часу.

Практична значимість роботи полягає в отриманні моделей прикладного програмного забезпечення для застосування і оцінки ефективності методів структурного розпізнавання, а також для підтвердження їх ефективності в конкретних прикладах баз даних зображень.

Подальше поліпшення розпізнавання може бути досягнуто за допомогою підходів до навчання з учителем, оскільки класи дескрипторів даних відомі. Основною проблемою подальшої модернізації є потенційна втрата простоти вивчених алгоритмів, а з нею і досягнуту швидкодію.

Результати атестаційної роботи апробовано у вигляді 2 статей у журналі «Системи управління, навігації та зв’язку» [39, 40], тез доповіді на міжнародній конференції ISDMCI’18 [41]. А також нагороджено 2 місцем на ІІ турі Всеукраїнського конкурсу студентських наукових робіт 2018/2019 зі спеціальності «Комп’ютерні науки» (Додаток Б).

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Форсайт, Д. А., & Понс, Ж. (2004). Компьютерное зрение. Современный подход. М.: Изд. дом „Вильямс“, 928 с.

2. OpenCV foundation. (2017). OpenCV: Image Thresholding. Retrieved from https://docs.opencv.org/3.3.0/d7/d4d/tutorial_py_thresholding.html

3. Новотарський, М. А., & Нестеренко, Б. Б. (2004). Штучні нейронні мережі: обчислення. К.: Інститут Математики НАН України, 400 с.

4. Calonder, M., Lepetit, V., Strecha, C., & Fua, P. (2010, September). Brief: Binary robust independent elementary features. In European conference on computer vision (pp. 778-792). Springer, Berlin, Heidelberg.

5. Lowe, D. G. (2004). Object Recognition from Local Scale-Invariant Features. Int. Journal of Computer Vision, 60(2), 91–110.

6. Патин, М. В., & Коробов, Д. В. (2016). Сравнительный анализ методов поиска особых точек и дескрипторов при группировке изображений по схожему содержанию. Молодой ученый, (11), 214-221.

7. Rosten, E., & Drummond, T. (2006, May). Machine learning for high-speed corner detection. In European conference on computer vision (pp. 430-443). Springer, Berlin, Heidelberg.

8. Rosten, E., & Drummond, T. (2005, October). Fusing points and lines for high performance tracking. In ICCV (Vol. 2, pp. 1508-1515).

9. Liang, Y. J., Li, Q., Chen, D. P., & Yan, X. J. (2012). Fast and Robust LOG-FAST Corner Algorithm. Computer Science, 39(6), 251-254.

10. Wang, J., Markert, K., & Everingham, M. (2009, September). Learning Models for Object Recognition from Natural Language Descriptions. In BMVC (Vol. 1, No. 2009, p. 2).

11. Calonder, M., Lepetit, V., Strecha, C., & Fua, P. (2010, September). Brief: Binary robust independent elementary features. In European conference on computer vision (pp. 778-792). Springer, Berlin, Heidelberg.

12. Rublee, E., Rabaud, V., Konolige, K., & Bradski, G. R. (2011, November). ORB: An efficient alternative to SIFT or SURF. In ICCV (Vol. 11, No. 1, p. 2).

13. Rodehorst, V., & Koschan, A. (2006, March). Comparison and evaluation of feature point detectors. In 5th International Symposium Turkish-German Joint Geodetic Days.

14. Mikolajczyk, K., & Schmid, C. (2005). A performance evaluation of local descriptors.

15. Hua, G., Brown, M., & Winder, S. (2007, October). Discriminant embedding for local image descriptors. In 2007 IEEE 11th International Conference on Computer Vision (pp. 1-8). IEEE.

16. Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. International journal of computer vision, 60(2), 91-110.

17. Bay, H., Tuytelaars, T., & Van Gool, L. (2006, May). Surf: Speeded up robust features. In European conference on computer vision (pp. 404-417). Springer, Berlin, Heidelberg.

18. Tuytelaars, T., & Schmid, C. (2007, October). Vector quantizing feature space with a regular lattice. In 2007 IEEE 11th International Conference on Computer Vision (pp. 1-8). IEEE.

19. Winder, S., Hua, G., & Brown, M. (2009, June). Picking the best daisy. In 2009 IEEE conference on computer vision and pattern recognition (pp. 178-185). IEEE.

20. Calonder, M., Lepetit, V., Fua, P., Konolige, K., Bowman, J., & Mihelich, P. (2009, September). Compact signatures for high-speed interest point description and matching. In 2009 IEEE 12th International Conference on Computer Vision (pp. 357-364). IEEE.

21. Shakhnarovich, G. (2005). Learning task-specific similarity (Doctoral dissertation, Massachusetts Institute of Technology).

22. Ozuysal, M., Calonder, M., Lepetit, V., & Fua, P. (2009). Fast keypoint recognition using random ferns. IEEE transactions on pattern analysis and machine intelligence, 32(3), 448-461.

23. Intel. Intel® SSE4 Programming Reference, D91561-103, 2007.

24. Rosenblatt, F. (1957). The perceptron, a perceiving and recognizing automaton Project Para. Cornell Aeronautical Laboratory.

25. Leutenegger, S., Chli, M., & Siegwart, R. (2011). BRISK: Binary robust invariant scalable keypoints. In 2011 IEEE international conference on computer vision (ICCV) (pp. 2548-2555).

26. McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics, 5(4), 115.

27. Kearns, M., & Ron, D. (1999). Algorithmic stability and sanity-check bounds for leave-one-out cross-validation. Neural computation, 11(6), 1427-1453.

28. Lin, M., Chen, Q., & Yan, S. (2013). Network in network. arXiv preprint arXiv:1312.4400.

29. Кохонен, Т. (2008). Самоорганизующиеся карты. М.: Бином. Лаборатория знаний.

30. Борисов, Е. С. (2014). Кластеризатор на основе нейронной сети Кохонена. Retrieved from http://mechanoid.kiev.ua/neural-net-kohonen-clusterization.html

31. Осовский, С. (2004). Нейронные сети для обработки информации. Финансы и статистика.

32. Горбаченко, В. И. (2010). Сети и карты Кохонена. Научно-исследовательский центр самоорганизации и развития систем.

33. Gorokhovatskyi, V. A. (2018). Image classification methods in the space of descriptions in the form of a set of the key point descriptors. Telecommunications and Radio Engineering, 77(9).

34. Gorokhovatskiy, V. A. (2016). Efficient Estimation of Visual Object Relevance during Recognition through their Vector Descriptions. Telecommunications and Radio Engineering, 75(14).

35. Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1985). Learning internal representations by error propagation (No. ICS-8506). California Univ San Diego La Jolla Inst for Cognitive Science.

36. Гороховатский, В. А. (2014). Структурный анализ и интеллектуальная обработка данных в компьютерном зрении. М.: Компания СМИТ, 316 с.

37. Гороховатский, В. А., & Полякова, Т. В. (2018). Применение пространственных структур признаков для классификации изображений в компьютерном зрении.

38. Гороховатский, В. А., Гороховатский, А. В., & Берестовский, А. Е. (2016). Структурное распознавание изображений с применением моделей интеллектуальной обработки и самоорганизации признаков. Радіоелектроніка, інформатика, управління, (3 (38)).

39. Пупченко, Д. В., Солодченко, К. Г. & Гороховатський, В. О. (2018). Аналіз властивостей, характеристик та результатів застосування новітніх детекторів для визначення особливих точок зображення. Системи управління, навігації та зв'язку, 1(47) , 93–98.

40. Пупченко, Д. В. & Гороховатский, В. А., (2018). Классификация изображений визуальных объектов по множеству дескрипторов особенных точек на основе нейронной сети Кохонена. Системи управління, навігації та зв'язку, 1 (47), 68–72.

41. Пупченко, Д. В., Гороховатский, В. А., & Путятин, Е. П. (2018). Изучение адаптационных свойств сети Кохонена в задаче классификации изображений. In 2018 ISDMCI international conference (pp. 274–276).

42. Пупченко Д. В. (2018) Применение нейронной сети Кохонена для структурной классификации изображений. In XXII Міжнародний молодіжний форум Радіоелектроніка та молодь у XXI столітті. (pp. 23-24)

43. Пупченко Д. В. (2019) Применение нейронной сети Кохонена для структурной классификации изображений. In XXIII Міжнародний молодіжний форум Радіоелектроніка та молодь у XXI столітті. (pp. 65-66)