Міністерство освіти і науки України
Донбаська державна машинобудівна академія (ДДМА)

Л. М. Богданова, Л. В. Васильєва

МЕТОДИ ОБРОБКИ ЗОБРАЖЕНЬ І КОМП'ЮТЕРНИЙ ЗІР

Навчальний посібник

для здобувачів другого (магістерського) рівня вищої освіти

Затверджено
на засіданні вченої ради ДДМА
Протокол № 4 від 27.11.2025

Краматорськ
ДДМА
2026

УДК 004.9
Б 73

Рецензенти:

Сагайда П. І., д-р техн. наук, доц., проф. кафедри цифрових технологій та проєктно-аналітичних рішень ТОВ «Технічний університет ”МЕТІНВЕСТ ПОЛІТЕХНІКА”»;

Гончаров О. А., д-р фіз.-мат. наук, професор кафедри комп’ютерних та інформаційних технологій і систем Державного податкового університету.

Богданова, Л. М.

Б 73 Методи обробки зображень і комп’ютерний зір : навч. посіб. : для здобувачів другого (магістерського) рівня вищої освіти / Л. М. Богданова, Л. В. Васильєва. – Краматорськ : ДДМА, 2026. – 116 с.
ISBN 978-617-7893-10-2

У посібнику розглянуто теоретичні та практичні аспекти використання методів цифрової обробки зображень та комп’ютерного зору. Висвітлено способи математичного опису зображень, методи лінійної просторово-інваріантної фільтрації, методи відновлення та морфологічної обробки бінарних зображень. Детально викладено основні поняття цифрової обробки зображень, характеристику технічних засобів, параметричні та непараметричні методи класифікації. Наведено використання згорточних нейронних мереж у розпізнаванні образів. Описано методи комп’ютерного зору для виявлення та відстеження об’єктів. Рекомендовано для здобувачів першого та другого рівнів вищої освіти усіх форм навчання за спеціальністю «Комп’ютерні науки», а також для дослідників, які працюють у галузі «Інформаційні технології».

УДК 004.9

ISBN 978-617-7893-10-2

ЗМІСТ

ВСТУП

Багато галузей техніки, що мають відношення до отримання, обробки, зберігання, передачі інформації, у значній мірі орієнтуються на розвиток систем, у яких інформація має характер зображень. Зображення, яке можна розглядати як двовимірний сигнал, є значно більш ємним носієм інформації, ніж звичайний одновимірний (часовий) сигнал.

Цифрова обробка і розпізнавання зображень – один із напрямків досліджень, що інтенсивно розвиваються [1].

Комп'ютерна обробка зображень передбачає обробку цифрових зображень за допомогою комп'ютерів або спеціалізованих пристроїв, побудованих на цифрових сигнальних процесорах. При цьому під обробкою зображень розуміється не тільки поліпшення зорового сприйняття зображень, але і класифікація об'єктів, виконувана при аналізі зображень.

1 Типи задач

Три класичні задачі в застосуванні до графіки формулюються наступним чином. Завдання синтезу зображень передбачає отримання зображення за його описом. Завдання аналізу полягає в отриманні опису заданого зображення. Завдання обробки зображень полягає в отриманні нового зображення на підставі вихідного з використанням потрібного алгоритму. Залежно від виду розв'язуваної задачі конкретизується послідовність дій та задіяних методів.

Завдання пошуку зображень на основі змісту. Є база даних зображень. Такими зображеннями можуть бути особи людей, різні об'єкти тривимірного світу, зображення автомобілів, товарів, різних виробів, будівель і т. п. Ставиться завдання пошуку потрібного зображення в базі даних. Причому завдання може формулюватися як для пошуку точного зображення, так і максимально наближеного до заданого. Можливе також

виділення класу об'єктів, до якого належить заданий об'єкт. У цьому випадку у якості сцени реального світу виступають відповідні об'єкти і для отримання бази даних зображень використовуються необхідні технічні засоби.

Важливим завданням є підбір методів і алгоритмів обробки зображень, які можуть дати якісне вирішення завдання. Технології обробки в цьому випадку залежать від багатьох параметрів: об'єму бази даних, розмірів на зображенні, якості зображень, параметрів яскравості і контрастності зображень, наявності фону, ракурсів розташування об'єктів і т. ін.

Залежно від вимог завдання можливий широкий спектр необхідних висновків. У найпростіших випадках можлива двозначна відповідь типу «так/ні», у більш складних – вибір потрібного об'єкту на зображенні або класу об'єктів.

Завдання обробки медичних зображень. Як зображення використовуються зображення внутрішніх органів людини. Для їх отримання використовують ефект ядерного магнітного резонансу для голови, рентгенографію для зображень легенів і скелета, комп'ютерну томографію для різних органів, ультрасонографію для серця, нирок та інших органів. Діагностика станів внутрішніх органів базується на залученні сучасних медичних знань і виробленні якісних методів аналізу зображень.

Завдання обробки текстових документів. Тут об'єктом реального світу виступає текстовий документ, за яким отримують цифрову форму його зображення за допомогою сканера. Завдання комп'ютерного зору може формулюватися як розпізнавання окремих символів, слів чи переведення цифрового зображення в текстовий файл. Крім того, може ставитися завдання і семантичного розпізнавання тексту для індексації його у великій базі даних.

Завдання сортування деталей на конвеєрі. У таких завданнях передбачається обмежений набір деталей, що дозволяє сформувати для них

набір характерних параметрів. Отримані зображення деталей за допомогою камери (або декількох камер) необхідно обробити в реальному масштабі часу і виконати сортування деталей. При розпізнаванні деталі може використовуватися тривимірна модель, попередньо сформована за допомогою їх характерних властивостей.

Завдання обробки відеопотоку для охоронних систем. Як правило, такі завдання пов'язані з розпізнаванням руху об'єктів на тлі нерухомої сцени. Тут також задіяні камери, налаштовані на передачу зображення деякої сцени. Апріорно передбачається, що сцена в процесі часу залишається незмінною або майже незмінною, тобто динаміка руху можлива в обмежених межах. Завдання полягає у виявленні рухомих об'єктів, можливо, без розпізнавання самих об'єктів. Схоже завдання може виконуватися і для різних систем, що стежать.

Аналіз транспортної магістралі. Це завдання близьке до попереднього і відрізняється від нього деякими деталями по швидкості руху об'єктів і можливими елементами розпізнавання типу об'єкту і номера автомобіля.

Завдання обробки супутникових зображень. Обробка супутникових зображень може мати різні аспекти. Можуть вирішуватися завдання сільськогосподарського призначення, аналізу і стану лісових масивів, виявлення пожеж, оцінки стану сніжного покриву перед весняними паводками, завдання розпізнавання військового призначення та ін. На вирішення цих завдань істотно впливають якість апаратури для зйомок, погодні умови і швидкість обробки інформації. У багатьох випадках для ефективного вирішення завдань може бути корисним використання мультиспектральних зображень.

2 Категорії (групи) методів обробки зображень

Коригування околу пікселів. Призначення цих методів полягає в придушенні шумів і спрощенні зображення за рахунок видалення дрібних деталей зображення. Іноді застосовується коригування граничних пікселів суцільних областей зображення. Ця операція в деяких випадках дозволяє чіткіше розділити окремі елементи зображення. Коригування граничних пікселів полягає в заміні сусідніх пікселів на фонові значення.

Поліпшення якості зображення. Операція виконується для всього зображення за одним алгоритмом. Поліпшення досягається за рахунок зміни яскравості (підвищення або зниження), а також за рахунок видалення шумів. Отримання потрібного результату домагаються підбором відповідних фільтрів. Нерідко використовуються фільтри, що дозволяють змінити контрастність зображення або виділити контури об'єктів.

Отримання зображення за кількома вихідними. У цій категорії методів використовуються операції віднімання або додавання зображень. Операція віднімання дозволяє виявляти рухомі об'єкти, а складання – отримати нове зображення із сумарними елементами вихідних зображень. Як правило, у якості вихідних використовуються два зображення, але можливі й інші варіанти.

Обчислення характеристик зображення. Залежно від розв'язуваної задачі можна підібрати характерні ознаки розпізнаваних об'єктів: довжина, ширина, площа, колір, характерні контури, орієнтація та інші. За складеним набором характеристик проводиться підрахунок їх на зображенні. Установивши деякі допустимі діапазони значень заданих параметрів, можна проводити аналіз результатів обчислень і робити підсумкові висновки.

Отримання описового результату. Ці методи відносяться здебільшого саме до завдань комп'ютерного зору. Результати розв’язання задачі можуть видаватися у вигляді підрахунку деяких об'єктів на зображенні, наявності або відсутності якихось ознак, у вигляді текстового файлу, отриманого за його графічним зображенням. Можливе отримання опису сцени із зазначенням переліку виявлених об'єктів, їхніми характеристиками і орієнтацією на сцені.

1 ОСНОВИ АНАЛІЗУ ТА ОБРОБКИ ЗОБРАЖЕНЬ

1.1 Колірні моделі

Колірна модель – абстрактна модель опису представлення кольорів у вигляді кортежів (наборів) чисел, зазвичай із трьох або чотирьох значень, званих колірними компонентами або колірними координатами. Разом із методом інтерпретації цих даних (наприклад, визначення умов відтворення та/або перегляду – тобто завдання способу реалізації), множина кольорів колірної моделі визначає колірний простір.

Також під колірною моделлю необхідно розуміти спосіб відображення колірної гами в дискретному вигляді, для представлення її в обчислювальних, цифрових системах.

Адитивна модель. Вона є досить штучним прийомом, оскільки продиктована технологією виготовлення електронно-променевих трубок. Це апаратно-орієнтована модель, у якій кольори описуються за допомогою складання трьох базових кольорів – червоного, зеленого, синього – у різних пропорціях. Тому модель RGB називають адитивною (від англ. «add» – складати, додавати). Кольори також називають колірними каналами моделі RGB. Модель названа за першими буквами англійських слів:

R (RED) – червоний;
G (GREEN) – зелений;
B (BLUE) – синій.

Кожен із базових кольорів може набувати інтенсивності (насиченості) у діапазоні від 0 до 255. Повна кількість кольорів, які представляються цією моделлю, дорівнює 256 х 256 х 256 = 16 777 216. За допомогою моделі RGB описуються кольори, що отримуються змішуванням світлових променів. Цю модель використовують монітори,

телевізори, сканери, слайд-проєктори, кольорові лампи реклами та інші пристрої, у яких колір отримують шляхом змішування світлових пучків. Вона також використовується для опису кольорів на сторінках інтернету в спеціальному шістнадцятирічному вигляді (#RRGGBB). Змінюючи інтенсивність свічення кольорових точок, можна створити велике різноманіття відтінків. Якщо інтенсивність кожного з них максимальна (255), то виходить білий колір. Відсутність усіх трьох кольорів дає чорний колір. Якщо змішуються всі кольори з однаковою інтенсивністю (але не максимальною і не мінімальною), отримуємо сірий колір (рис.1.1).

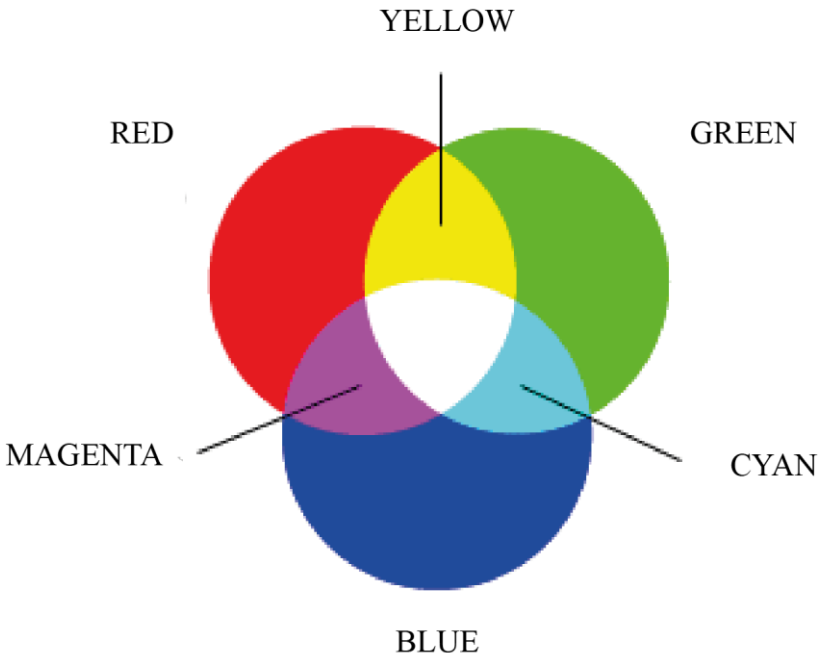


Рисунок 1.1 – Комбінації базових кольорів моделі RGB

Для зображення адитивної моделі найчастіше застосовують одиничний куб із розподілом кольорів уздовж одиничних векторів (рис. 1.2). Початок відліку (0,0,0) відповідає чорному кольору. Максимальне значення RGB (1,1,1) відповідає білому кольору, (1;0;0) – червоному, (0;1;0) – зеленому, (0;0;1) – синьому.

На сьогодні система RGB є офіційним стандартом. Рішенням Міжнародної комісії з освітлення (МКО) у 1931 р. були стандартизовані

основні кольори. Комісія рекомендувала використовувати як *R*, *G*, *B* такі монохроматичні кольори – випромінювання хвиль довжиною: для *R* – 700 нм, для *G* – 546,1 нм, для *B* – 435,8 нм.

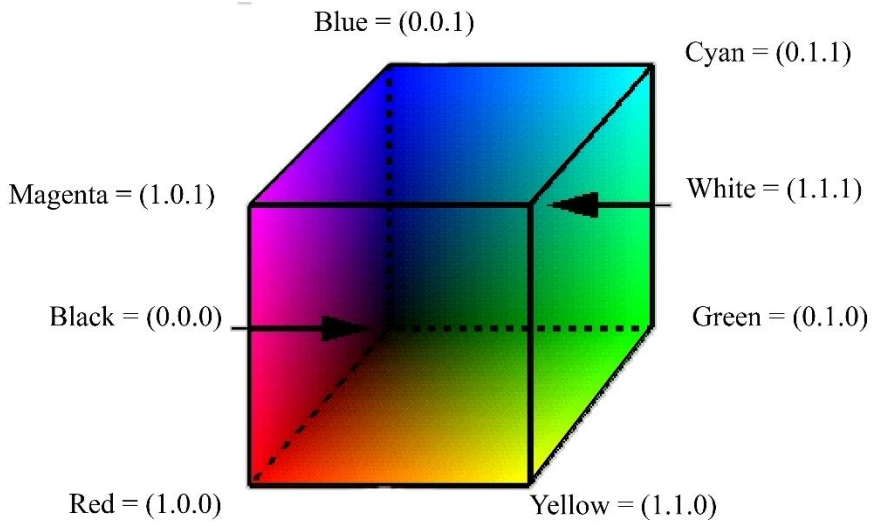


Рисунок 1.2 – Колірний куб моделі RGB

Недолік моделі RGB полягає в тому, що не всі кольори, утворені в ній, можна вивести на друк. Проте більше 16 млн кольорів, що представляються в RGB, виявляються цілком достатніми для практичних потреб. Іншими словами, кольори на екрані вашого монітора можуть виглядати інакше при їхньому виведенні на друк, причому ця відмінність може виявитися принциповою, а не обумовленою низькою якістю принтера або монітора.

Окрім моделі RGB існують також багато інших моделей.

Модель HSV створена Елві Смітом у 1978 році. ЇЇ зручно представляти у вигляді світлової шестигранної піраміди. При цьому по вертикальній осі відкладається значення *V*, а відстань від осі до бічної грані в горизонтальному перетині відповідає параметру *S* (за діапазон зміни цих величин приймається інтервал від нуля до одиниці, рис. 1.3). Шестикутник, що лежить в основі піраміди, являє собою проекцію колірного куба в напрямку його головної діагоналі. Тон кольору *H* задається кутом, відкладеним навколо вертикальної осі, починаючи від червоного. Точки на

самій окружності відповідають чистим (максимально насиченим) кольорам. Точка в центрі відповідає нейтральному кольору мінімальної насиченості (білий, сірий, чорний – це залежить від яскравості).

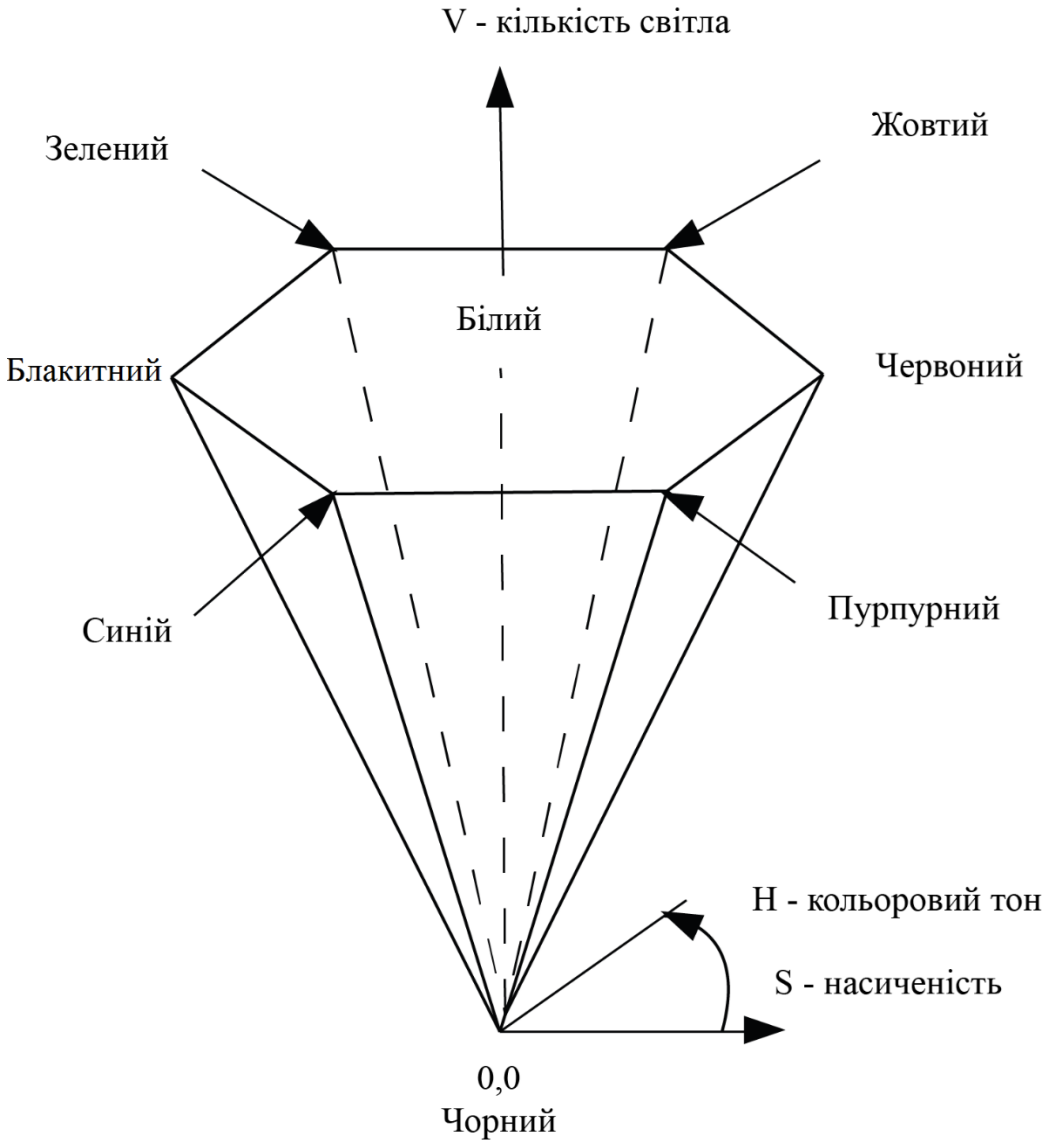


Рисунок 1.3 – Графічне представлення моделі HSV

Тобто можна сказати, що кут нахилу вектору визначає відтінок, довжина вектору – насиченість кольору. Величина S змінюється від нуля на осі конуса до одиниці на його гранях. Значенню $V = 0$ відповідає вершина піраміди (чорний колір), значенню $V = 1$ – основа піраміди; кольори при цьому найбільш інтенсивні. Точка з координатами $V = 1, S = 0$ – центр основи піраміди – відповідає білому кольору. Проміжні значення

координати V при $S = 0$ (тобто на осі піраміди) відповідають сірим кольорам; якщо $S = 0$, то значення відтінку H вважається невизначеним. Якщо точка лежить на бічній грані піраміди, то $S = 1$ (рис. 1.4).

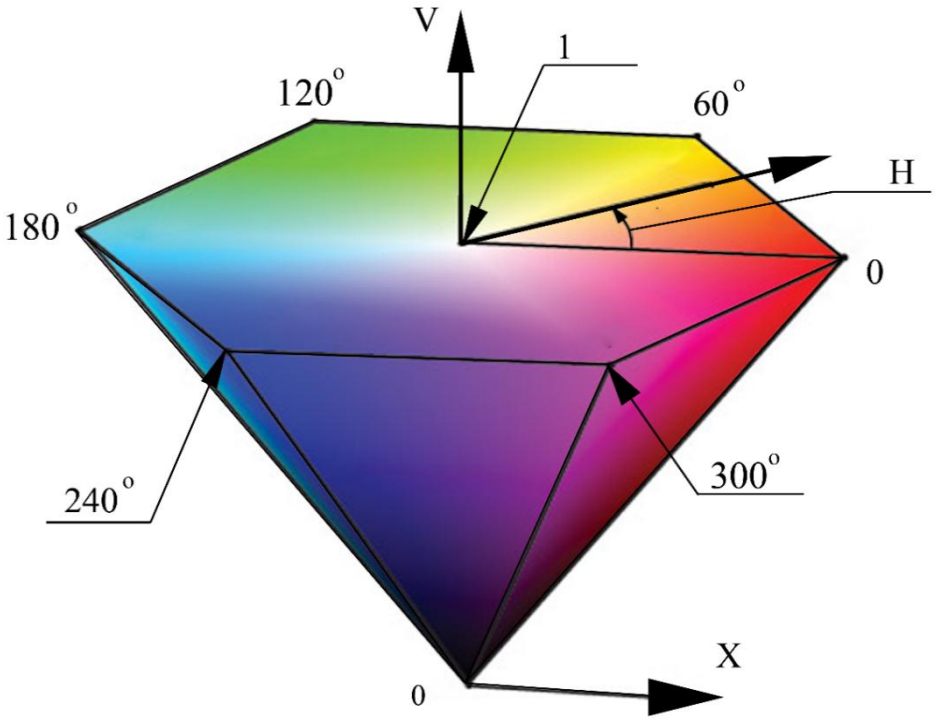


Рисунок 1.4 – Графічне представлення моделі HSV із кольоровим градієнтом

Для перетворення RGB до HSV використовуються така формула. Нехай MAX – максимальне значення з R, G, B, а MIN – мінімальне:

$$H = \begin{cases} 0, & \text{if } MAX = MIN \\ 60 \times \dfrac{G-B}{MAX-MIN} + 0, & \text{if } MAX = R \text{ и } G \geq B \\ 60 \times \dfrac{G-B}{MAX-MIN} + 360, & \text{if } MAX = R \text{ и } G < B \\ 60 \times \dfrac{B-R}{MAX-MIN} + 120, & \text{if } MAX = G \\ 60 \times \dfrac{R-G}{MAX-MIN} + 240, & \text{else } MAX = B \end{cases}$$

$$S = \begin{cases} 0, & \text{if} \quad MAX = 0; \\ 1 - \dfrac{MIN}{MAX}, & \text{else}; \end{cases}$$

$$V = MAX.$$

Модель HSL. Розширенням колірної моделі HSV/HSB є HLS, параметрами якої є Hue, Lightness, Saturation, відповідно: кольоровий тон, кількість світла (освітленість), насиченість (рис. 1.5). Різниця між моделями HSV/HSB і HSL полягає в заміні нелінійного компонента «яскравість» на лінійний компонент «освітлення».

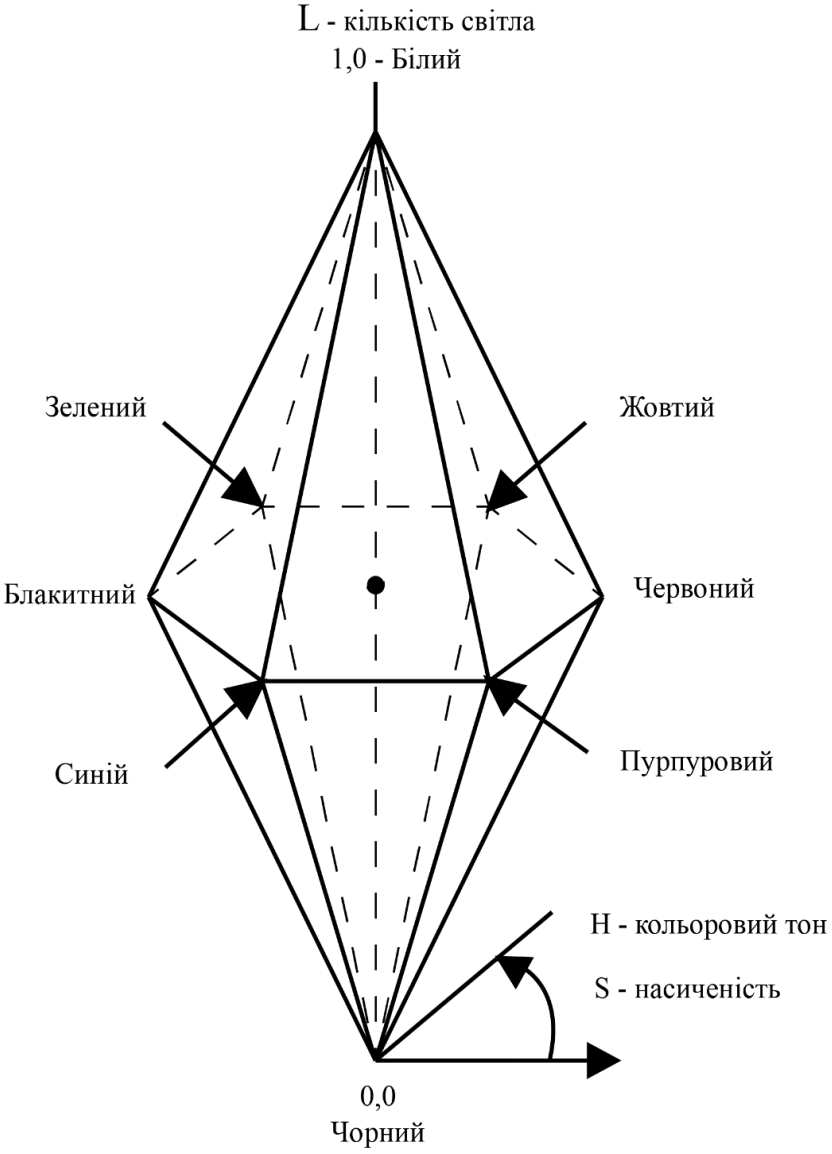


Рисунок 1.5 – Графічне представлення моделі HSL

В основі колірної моделі HSL лежить система Освальда. Ця модель утворює простір у формі подвійного конуса. Колірний тон задається кутом повороту навколо вертикальної осі конусів. За початок відліку прийнятий синій колір. Кольори йдуть у спектральному порядку й замикаються пурпуровим. Отже, по вертикальній осі відкладається L (освітленість), а інші два параметри задаються як і в HSV/HSB. Ця модель утворює підпростір, що являє собою подвійний конус, у якому чорний колір задається вершиною нижнього конуса й відповідає значенню $L = 0$, білий колір максимальної інтенсивності задається вершиною верхнього конуса й відповідає значенню $L = 1$. Максимально інтенсивні кольорові тони відповідають основі конуса з $L = 0{,}5$, що не зовсім зручно. Тон кольору H, аналогічно системі HSV/HSB, задається кутом повороту. Насиченість S змінюється в межах від 0 до 1 і задається відстанню від вертикальної осі L до бічної поверхні конуса. Тобто максимально насичені кольори розташовуються при $L = 0{,}5$ і $S = 1$.

HSL добре пристосована для опису кольорів таким чином, як це властиво людині. Дивлячись на пофарбований об'єкт, людині простіше його описати за допомогою кольору, освітленості й насиченості, що й роблять ці колірні моделі. Безперечною перевагою цих моделей при побудові алгоритмів обробки зображень є простота розуміння, оскільки в їхній основі лежить природний і інтуїтивно близький людині опис кольору, адже саме людина в кінцевому рахунку і є розробником та користувачем цих алгоритмів (рис. 1.6).

Для перетворення RGB у HSL використовуються така формула:

$$H = \begin{cases} \text{undefined} & \text{if } MAX = MIN \\ 60^\circ \times \frac{G-B}{MAX-MIN} + 0^\circ, & \text{if } MAX = R \\ & \text{and } G \geq B \\ 60^\circ \times \frac{G-B}{MAX-MIN} + 360^\circ, & \text{if } MAX = R \\ & \text{and } G < B \\ 60^\circ \times \frac{B-R}{MAX-MIN} + 120^\circ, & \text{if } MAX = G \\ 60^\circ \times \frac{R-G}{MAX-MIN} + 240^\circ, & \text{if } MAX = B \end{cases}$$

$$S = \begin{cases} 0 & \text{if } L = 0 \text{ or } MAX = MIN \\ \frac{MAX-MIN}{MAX+MIN} = \frac{MAX-MIN}{2L}, & \text{if } 0 < L \leq \frac{1}{2} \\ \frac{MAX-MIN}{2-(MAX+MIN)} = \frac{MAX-MIN}{2-2L}, & \text{if } \frac{1}{2} < L < 1 \end{cases}$$

Або в загальному випадку:

$$S = \frac{MAX - MIN}{1 - |1 - (MAX + MIN)|},$$
$$L = \frac{1}{2}(MAX + MIN),$$

де R, G, B — значення кольору в колірній моделі RGB, значення в діапазоні [0; 1] (R — червоний, G — зелений, B — синій), MAX — максимум із трьох значень (R, G, B), MIN — мінімум із трьох значень (R, G, B), H — тон [0; 360], S — насиченість [0; 1], L — яскравість [0; 1].

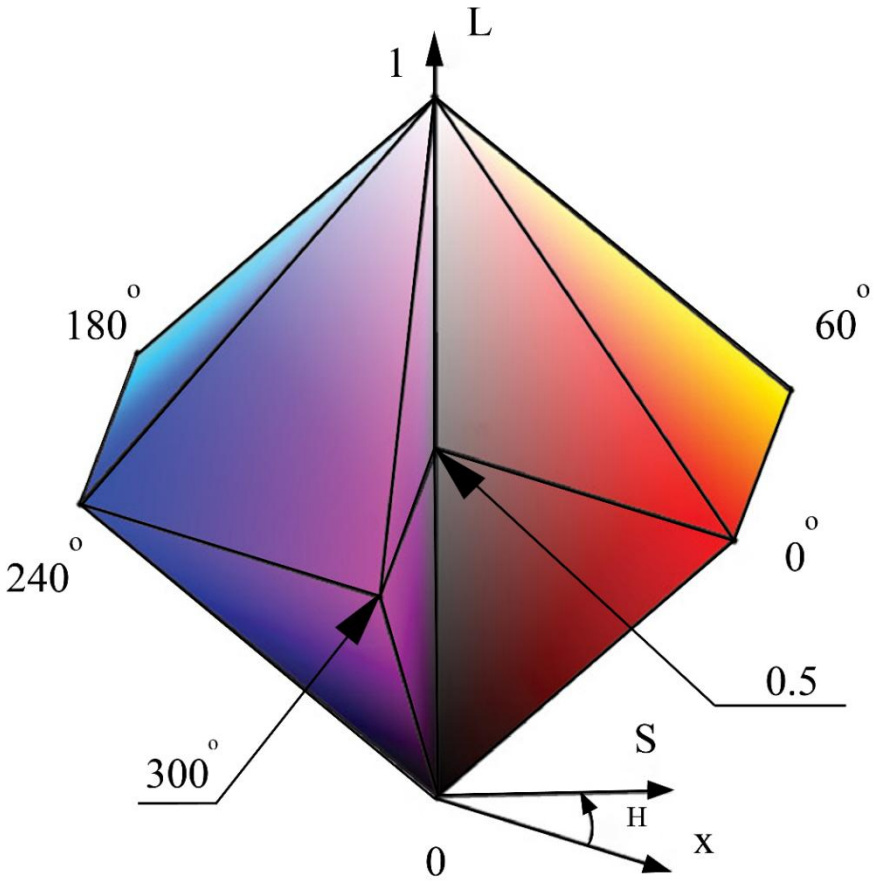


Рисунок 1.6 – Графічне представлення моделі HSL із кольоровим градієнтом

Модель LAB базується на людському сприйнятті кольору. При однаковій інтенсивності око людини сприймає промені зеленого кольору найбільш яскравими, дещо менш яскравими – червоного кольору і ще менш яскравими – синього. Яскравість при цьому є характеристикою сприйняття, а не характеристикою самого кольору. При розробці моделі LAB переслідувалася мета математичного коректування нелінійності сприйняття кольору людиною.

Цій моделі віддають перевагу в основному професіонали, оскільки вона забезпечує доступ до всіх кольорів, працюючи з досить великою швидкістю. А також вона відрізняється незвичайною побудовою, на відміну від інших колірних моделей. Побудова кольорів тут, так як і в RGB, базується на злитті трьох каналів.

Будь-який колір у колірній моделі LAB обумовлюється параметрами L – яскравість (Lightness) і хроматичними складовими – двома декартовими координатами: ***a*** (змінюється від зеленого до червоного, через сірий) і ***b*** (змінюється від синього до яскраво-жовтого через сірий). L здійснює контроль за яскравістю кольорів, утворених ***a*** і ***b***. Білий колір співставляється з максимальною інтенсивністю. L лежать у межах 0…100, a і b – в межах [–200; 200]. Якщо ***a*** і ***b*** рівні 0, змінюючи L, отримуємо зображення, що містить градації сірого (grayscale).

На відміну від кольорової моделі RGB, яка є, по суті, набором апаратних даних для відтворення кольору на папері чи на екрані монітора (колір може залежати від типу друкарської машини, марки фарб, вологості повітря в цеху чи виробника монітора і його налаштувань), LAB визначає колір. Оскільки яскравість у моделі LAB цілком відділена від кольору, то це робить модель зручною для регулювання тонової характеристики (підвищення контрасту, виправлення похибки тонових діапазонів) і видалення кольорового шуму (у т. ч. розмиття растру й видалення регулярної структури зображень у форматі JPEG), різкості та інших тонових характеристик зображення.

Ураховуючи позитивні характеристики, а саме величезний колірний обхват, модель LAB широко застосована в програмному забезпеченні для обробки зображень, через яку відбувається конвертація даних між іншими кольоровими просторами під час їхньої підготовки, наприклад із RGB-сканера в CMYK (CMYK – субтрактивна колірна модель, використовується у поліграфії, перш за все при повноколірному друці, застосовується в друкарських машинах і кольорових принтерах) друкуючого пристрою (рис. 1.7).

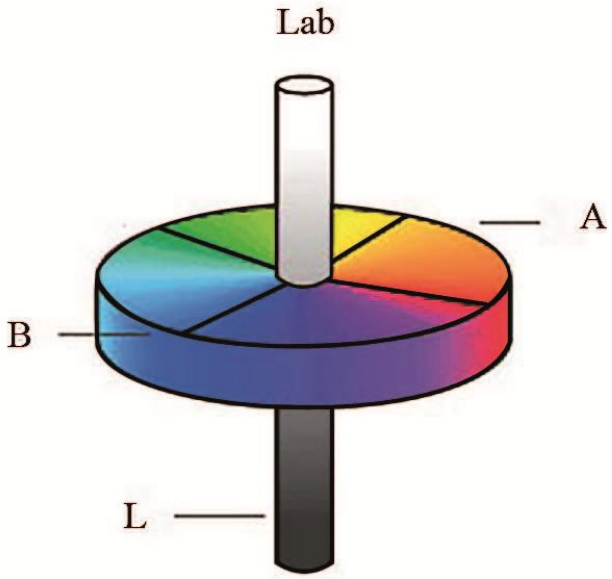


Рисунок 1.7 – Графічне представлення моделі LAB

1.2 Базові перетворення зображень

1.2.1 Конвертування зображення до чорно-білого варіанту

Кожен піксель зображення формується за допомогою поєднання чотирьох каналів: ARGB (Alpha, Red, Green, Blue), альфа-каналу, червоного, зеленого й синього. Альфа-канал відповідає за прозорість пікселя (100 % – піксель повністю непрозорий, 0 % – повністю прозорий). Поєднання значень RGB-каналів визначає колір пікселя. Кожен канал несе в собі 8 біт

інформації (1 байт), відповідно, інтенсивність каналу може змінюватися в діапазоні від 0 до 255. Повністю піксель займає в пам'яті 32 біта (4 байти). Для того щоб перетворити кольорове зображення в чорно-біле (рис. 1.8), потрібно знайти середнє арифметичне значення R, G і B-каналів пікселя і потім це значення привласнити RGB-каналам цього ж пікселя (тобто воно буде однакове). Альфа-канал залишаємо без змін. Таку операцію потрібно виконати з кожним пікселем зображення:

$$WBpixel_x = \frac{R_x + G_x + B_x}{3}.$$

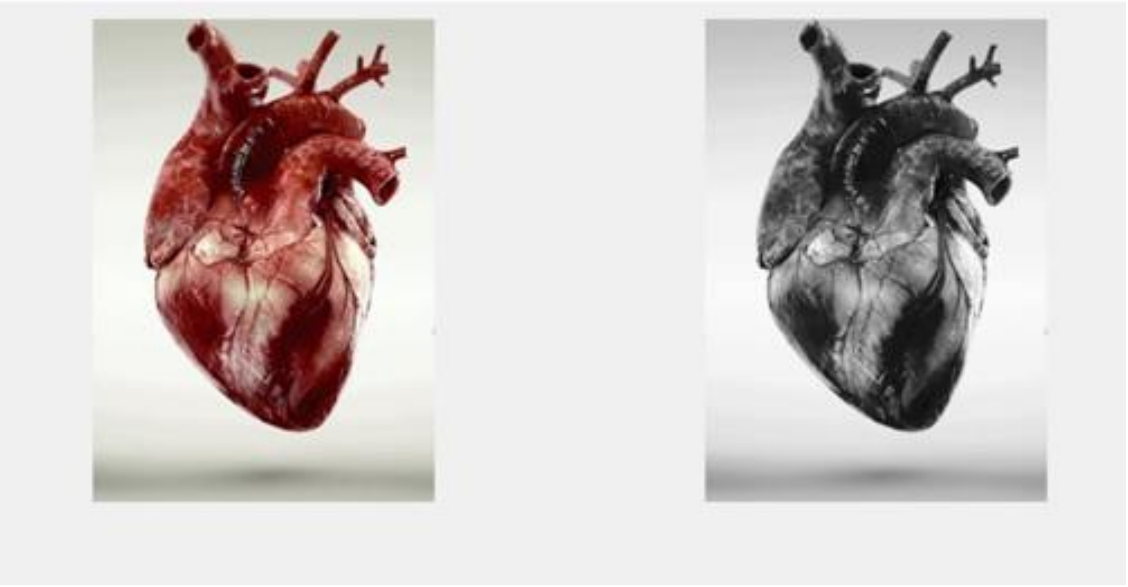


Рисунок 1.8 – Результат перетворення кольорового зображення на чорно-біле

Процес бінаризації – це зміна кольорового (або в градаціях сірого) зображення на двокольорове чорно-біле.

Головним параметром такого перетворення є поріг t-значення, з яким порівнюється яскравість кожного пікселя. За результатами порівняння пікселю присвоюється значення 0 або 1. Існують різні методи бінаризації, які можна умовно поділити на дві групи – глобальні та локальні. У першому випадку величина порога залишається незмінною протягом усього процесу

бінаризації. У другому зображення розбивається на області, у кожній з яких обчислюється локальний поріг.

Головна мета бінаризації – це радикальне зменшення кількості інформації, з якою доводиться працювати. Просто кажучи, удала бінаризація сильно спрощує подальшу роботу із зображенням. З іншого боку, невдачі в процесі бінаризації можуть призвести до спотворень, таких як розриви в лініях, утрата значущих деталей, порушення цілісності об'єктів, поява шуму й непередбачуване спотворення символів через неоднорідності фону.

Різні методи бінаризации мають свої слабкі місця: такі як втрата дрібних деталей, «злипання» навколишніх символів або поява помилкових об'єктів у разі неоднорідностей фону з низькою контрастністю. На рис. 1.9 представлений приклад бінаризації зображення з рис. 1.8 із пороговим значенням $t = 127$:

$$if\ R_x < t \rightarrow R = 0; B = 0; G = 0;$$
$$if\ R_x > t \rightarrow R = 255; B = 255; G = 255.$$

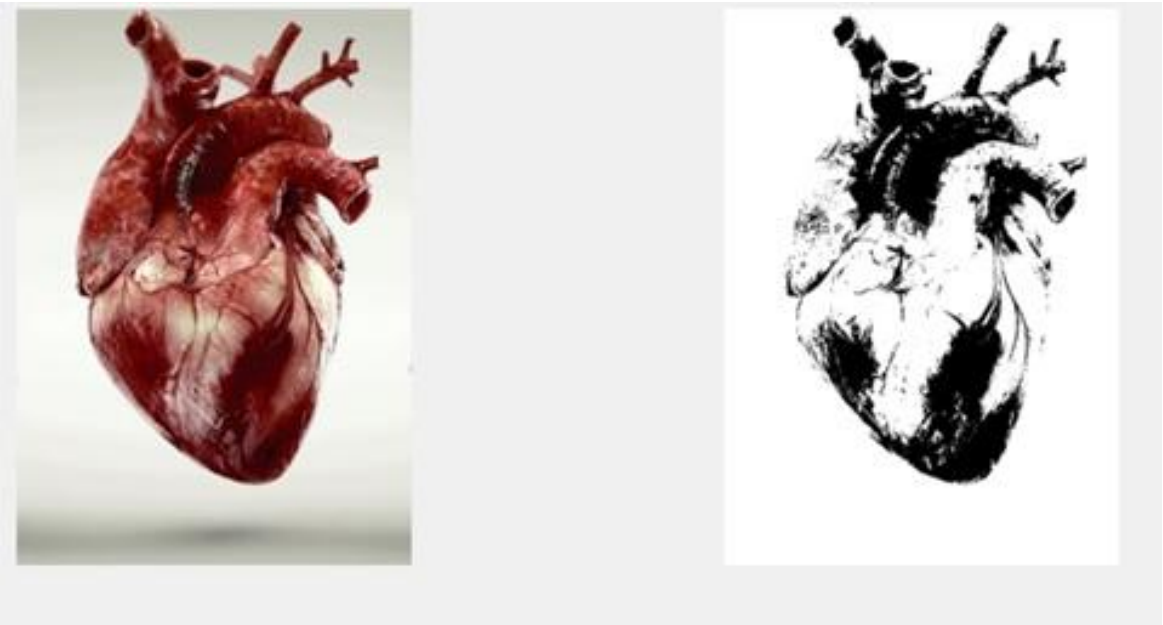


Рисунок 1.9 – Результат бінаризації зображення

1.2.2 Побудова лінійної і кумулятивної гістограм зображення

Для цифрового зображення формату градації сірого, шкала яскравості якого належить цілочисельному діапазону $0 \ldots 255$, гістограма являє собою таблицю з 256 чисел. Кожне з них показує кількість точок (пікселів) у кадрі, що мають певну яскравість.

Лінійна гістограма визначає повний перебір матриці зображення. Значення елементів матриці у свою чергу є індексами масиву гістограми. При виборі будь-якого елементу матриці до відповідного елементу масиву гістограми додається одиниця. У підсумку, після повного перебору матриці кожен елемент масиву відображає загальну кількість елементів матриці з відповідним рівнем яскравості (рис. 1.10). У кумулятивної гістограми будь-яке значення елемента масиву дорівнює сумі всіх попередніх (рис. 1.11).

Інверсія кольору потрібна для зміни кольорів на протилежні. При цьому значення яскравості зображення змінюється на різницю цього значення й максимуму на 256-кроковій шкалі значень кольору. Наприклад, якщо значення пікселя складає 250, то його протилежність при цьому правилі буде 5. Виконується заміна світлих кольорів на темні і навпаки (рис. 1.12) за формулами:

$$R_{invert} = 255 - R_X,$$
$$B_{invert} = 255 - B_X,$$
$$G_{invert} = 255 - G_X.$$

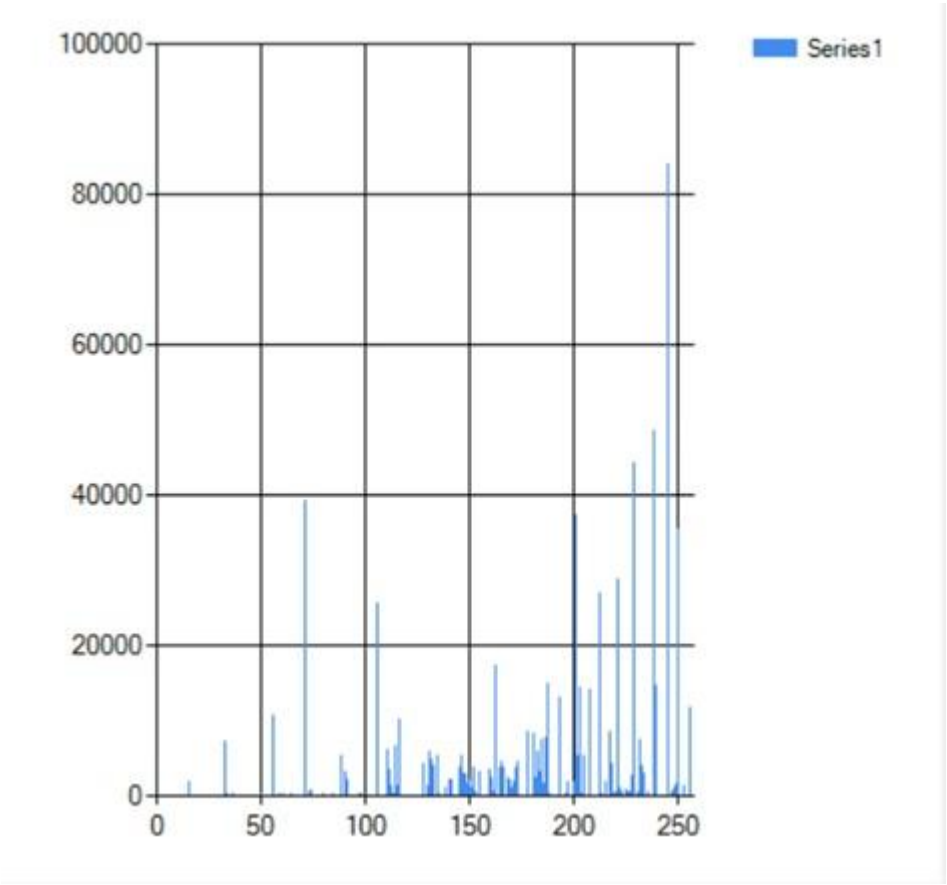


Рисунок 1.10 – Лінійна гістограма кількості різних кольорів для зображення серця (див. рис. 1.8)

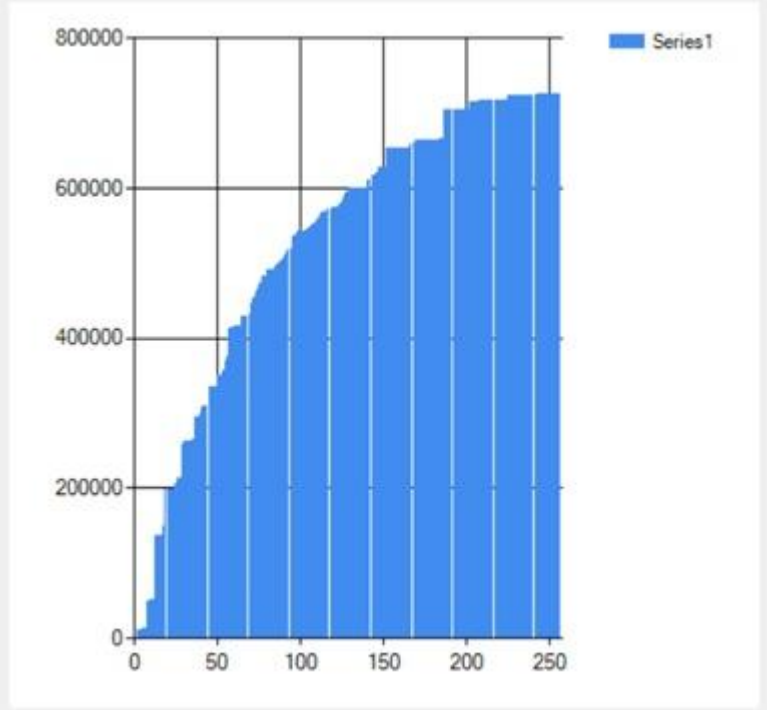


Рисунок 1.11 – Кумулятивна гістограма

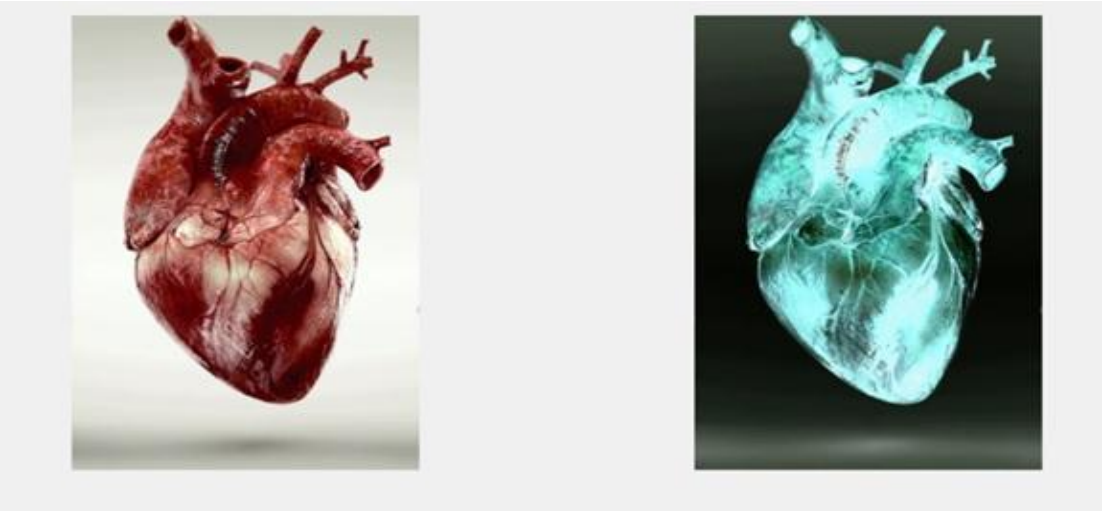


Рисунок 1.12 – Результат інверсії

1.2.3 Лінійне контрастування

Лінійне контрастування використовують для узгодження динамічного діапазону вхідного сигналу зображення та динамічного діапазону яскравості екрану пристрою відображення. Якщо для цифрового значення кожного відліку зображення виділяється 1 байт (8 біт) пристрою пам’яті, то вхідний або вихідний сигнали можуть набувати одного з 256 значень. Діапазон можливих значень аналогового сигналу знаходиться у межах від 0 до 255. Зазвичай, значення 0 відповідає рівню чорного, а значення 255 – рівню білого.

Припустимо, що мінімальна та максимальна яскравості первинного зображення (рис. 1.13, *а*) мають значення min(x) та max(x) відповідно. Якщо ці параметри або один із них істотно відрізняються від граничних значень діапазону сигналів яскравості на екрані пристрою, то відтворене зображення виглядає як малоконтрастне (ненасичене), незручне для сприйняття та дослідження (рис. 1.13, *б*).

Під час лінійного контрастування здійснюють лінійне поелементне перетворення вхідної величини x відповідно до рівняння

$$y = a \cdot x + b,$$

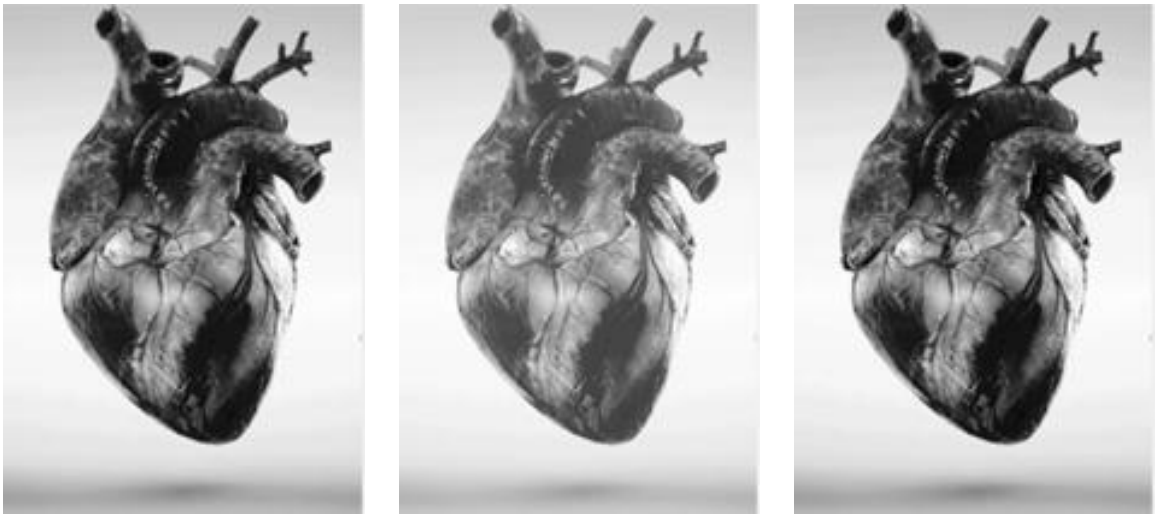
де параметри a та b визначаються бажаними значеннями мінімальної min(y) та максимальної max(y) яскравості в кінцевому зображенні. Для знаходження зазначених параметрів потрібно розв'язати систему рівнянь:

$$\begin{cases} \min(y) = a * \min(x) + b \\ \max(y) = a * \max(x) + b \end{cases}.$$

Після підстановки знайдених значень $\min(y)$, $\max(y)$ рівняння для лінійного контрастування набуває остаточного вигляду:

$$y = \frac{x - \min(x)}{\max(x) - \min(x)} (\max(y) - \min(y)) + \max(y).$$

При порівнянні зображень (вхідного та вихідного) можна бачити кращу візуальну якість обробленого зображення. Поліпшення обумовлене узгодженням динамічного діапазону вхідного зображення та динамічного діапазону екрана пристрою внаслідок здійснення лінійного контрастування (рис. 1.13, *в*).



а *б* *в*

Рисунок 1.13 – Лінійне контрастування: а – вхідне зображення у відтінках сірого; б – ненасичене зображення; в – контрастоване зображення

1.2.4 Збільшення зображень

Масштабування зображень є досить важливим завданням при їхньому аналізі. Це завдання нерозривно пов'язане з проблемою відновлення даних, оскільки при збільшенні фізичних розмірів зображення завжди виникають проміжні пікселі, значення яких невідоме. Визначення рівнів яскравості нових пікселів і є основна розв'язувана задача. Однак обидва пропоновані далі методи зручно застосовувати тільки для одновимірних масивів, тому спочатку необхідно провести операцію відновлення даних по рядках, ігноруючи рядки тільки з новими пікселями (усіма нульовими значеннями), а потім виконати ту ж операцію для стовпців отриманої матриці.

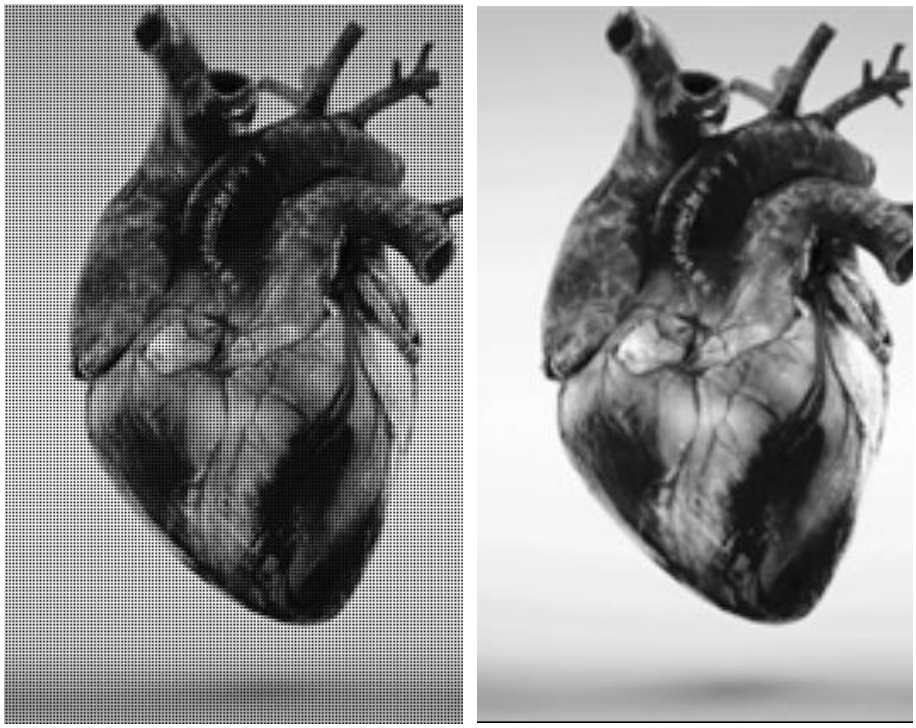
Екстраполяція – метод наукового прогнозування, що полягає в поширенні висновків, одержуваних зі спостереження над однією частиною явища, на іншу його частину.

У широкому сенсі екстраполяцією можна вважати доказову аналогію, коли переносяться властивості з одного об'єкта на інший. Екстраполяцією іноді називають перенесення властивостей змінності математичних моделей, наприклад рівнянь, на процеси, що описуються цими моделями.

Для збільшення зображення методом екстраполяції можна використовувати такий алгоритм (рис. 1.14, *а*):

$$R_{new} = \frac{R_X + R_{x+1}}{2}.$$

Метод інтерполяції заснований на тій же логіці, що й метод екстраполяції. Інтерполяція – це знаходження проміжних значень деякої закономірності (функції) за рядом відомих її значень. Інтерполяція більш доказовий спосіб припущення, ніж екстраполяція, тому що результат функції лежить у діапазоні значень досліджених параметрів.



а *б*

а – збільшене зображення методом екстраполяції; б – збільшене зображення методом інтерполяції

Рисунок 1.14 – Масштабування зображень

Для збільшення зображення методом інтерполяції можна використовувати такий алгоритм (рис. 1.14, *б*):

$$R_{new} = \frac{R_{X-1} + (R_{x-1} - R_{X-2})}{2}.$$

1.3 Оператори знаходження границь

Границі (також зустрічається – краї, англ. edge) – це такі криві на зображенні, уздовж яких відбувається різка зміна яскравості або інших видів неоднорідностей. Простіше кажучи, границя – це різкий перехід/зміна яскравості.

Причини виникнення границь:

– зміна освітленості;

– зміна кольору;

– зміна глибини сцени (орієнтації поверхні).

Виходить, що границі відображають важливі особливості зображення і тому цілями перетворення зображення в набір кривих є:

– виділення істотних характеристик зображення;

– скорочення обсягу інформації для подальшого аналізу.

Алгоритм Кенні (Canny) відомий як оптимальний детектор границь [2]. Критеріїв, що застосовуються у цьому методі, три. Першим і найбільш очевидним є низький рівень помилок. Дуже важливо, щоб присутні на зображенні кордони не були пропущені й не було помилкових виявлень. Другим критерієм є хороша локалізованість граничних точок, тобто відстань між виявленими граничними точками й фактичними точками границі має бути мінімальною. Третій критерій – одне виявлення на одну границю; він був уведений, щоб виключити можливість неодноразового виявлення одних і тих же границь.

Алгоритм виділення границь Кенні спочатку згладжує зображення, щоб усунути шум. Далі він знаходить градієнт зображення, щоб підсвітити області з високими просторовими похідними. Далі алгоритм проходить по цих областях і пригнічує всі пікселі, які не в максимумі. Градієнтний масив далі зменшується гістерезисом. Гістерезис використовується, щоб відстежити, чи залишилися пікселі, що не були придушені. Гістерезис використовує два порога, і якщо величина нижче першого порогу (T1), то вона встановлюється в нуль (робиться не граничною). Якщо величина вище високого порогу, вона робиться граничною. І якщо величина між цими двома порогами, то вона встановлюється в нуль у тому випадку, якщо немає шляху від цього пікселя до пікселя з градієнтом вище другого порогу T2. Нижче наведені кроки алгоритму.

Крок 1. Першим кроком є фільтрація шуму в оригінальному зображенні. Оскільки фільтр Гауса можна обчислити за допомогою простої маски, він використовується в алгоритмі Кенні. Після того як відповідна

маска була розрахована, згладжування Гауса може бути виконане за допомогою стандартних методів згортки. Маска згортки, як правило, набагато менша, ніж власне зображення. У результаті, маска рухається над зображенням, маніпулюючи квадратом пікселів за один раз. Чим більше ширина маски Гауса, тим менше чутливість детектора до шуму. Локалізація помилки у виявленні границь так само трохи збільшується зі збільшенням ширини гаусіана. Маска-гаусіан показана на рис. 1.15.

2	4	5	4	2
4	9	12	9	4
5	12	15	12	5
4	9	12	9	4
2	4	5	4	2

Рисунок 1.15 – Маска-гаусіан

Крок 2. Знаходження границі шляхом узяття градієнту зображення. Оператор Собеля виконує 2-D-просторове вимірювання градієнта в зображенні. Тоді наближені градієнти абсолютних величин у кожній точці можуть бути знайдені. Оператор Собеля використовує пару 3x3 масок згортки (рис. 1.16): оцінки градієнта в напрямку X (стовпці) і оцінки градієнта в Y-напрямку (рядки).

–1	0	+1
–2	0	+2
–1	0	+1

Gx

+1	+2	+1
0	0	0
–1	–2	–1

Gy

Рисунок 1.16 – Маски згортки

Величина, або границя сили градієнта, потім апроксимується за формулою

$$|G| = |Gx| + |Gy|.$$

Крок 3. Знаходження напрямку границі тривіальне, оскільки градієнти в напрямках X та Y відомі. Коли градієнт у напрямку X дорівнює нулю, напрямок границі має дорівнювати 90 або 0 градусам, у залежності від того, чому дорівнює значення градієнта в напрямку осі Y. Якщо G_Y має нульове значення, напрямок границі буде дорівнювати 0 градусів. В іншому випадку напрямок границі дорівнюватиме 90 градусам.

Крок 4. Зв'язування напрямку границі з напрямком, який може бути відстежений у зображенні. Таким чином, якщо пікселі 5×5 зображення розташовані як зазначено нижче, видно, що для пікселя «*a*» існують тільки чотири можливі напрямки опису навколишніх пікселів: 0 градусів (у горизонтальному напрямку), 45 градусів (уздовж позитивної діагоналі), 90 градусів (у вертикальному напрямку) або 135 градусів (по негативній діагоналі):

x	x	x	x	x
x	x	x	x	x
x	x	***a***	x	x
x	x	x	x	x
x	x	x	x	x

Тепер орієнтація границі повинна бути обрана в одному із цих чотирьох напрямків, у залежності від того, який напрямок знаходиться ближче (наприклад, якщо кут орієнтації виявляється +3 градуси, зробити його 0 градусів). Для цього представляють напівколо, розділене на 5 регіонів (рис. 1.17).

Будь-який напрямок краю, що входить у жовтий діапазон (від 0 до 22,5 та від 157,5 до 180 градусів), установлюється в 0 градусів. Будь-який напрямок краю, що потрапляє в зелений (від 22,5 до 67,5 градуса), установлюється на 45 градусів. Будь-який напрямок краю в синьому діапазоні (від 67,5 до 112,5 градусів) установлюється в 90 градусів. І,

нарешті, кожний напрямок краю в червоному діапазоні (від 112,5 до 157,5 градусів) установлюється в 135 градусів.

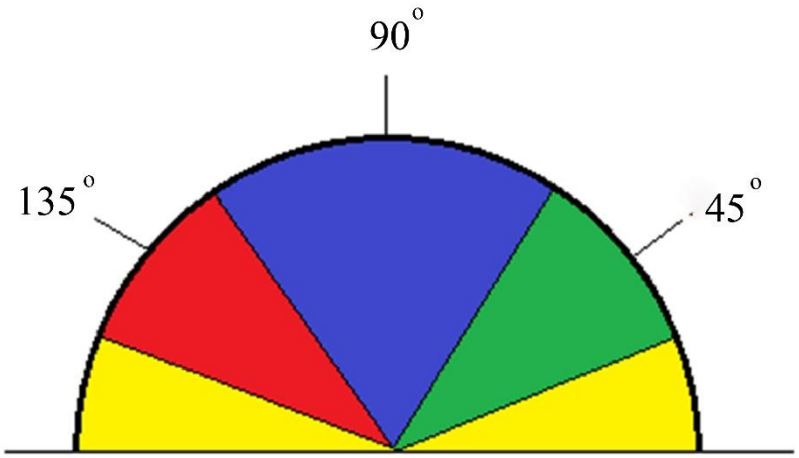


Рисунок 1.17 – Напівколо, розділене на 5 регіонів

Крок 5. Після того як відомі напрямки границь, застосовуємо немаксимальне подавлення. Воно використовується для відстеження уздовж границі в напрямку краю і подавлення будь-яких значень пікселів (установлюючи їх рівними 0), які не є граничними. Це дасть тонку лінію в результуючому зображенні.

Крок 6. Гістерезис використовується як засіб усунення смуг. Смуга – це розбиття контуру границі, викликане оператором вихідного коливання вище й нижче порогового рівня. Якщо єдиний поріг T1 застосовується до зображення і границя має середню силу, рівну T1, то через шум будуть випадки, коли границя опускається нижче порогового рівня. У рівній мірі вона буде також поширюватися вище порога прийняття границі, схожого на пунктирну лінію. Щоб уникнути цього, гістерезис використовує 2 пороги: високий і низький. Будь-який піксель у зображенні, який має значення більше, ніж T1, імовірно є крайовим пікселем і негайно позначається як такий. Потім будь-які пікселі, які з'єднуються із цим крайовим пікселем і які мають значення більші, ніж T2, також вибираються як крайові пікселі. Для

початку руху вздовж краю необхідний градієнт Т2, а для закінчення – градієнт нижче Т1.

Оператор Робертса [3]. Нехай область 3×3, показана на рис. 1.18, являє собою значення яскравості в околі деякого елемента зображення.

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

Рисунок 1.18 – Значення яскравості в околі деякого елемента зображення z_5 *у вигляді матриці* 3×3

Один із найпростіших способів знаходження перших часткових похідних у точці z_5 полягає в застосуванні наступного перехресного градієнтного оператора Робертса:

$$Gx = (z_9 - z_5),$$
$$Gy = (z_8 - z_6).$$

Ці похідні можуть бути реалізовані шляхом обробки всього зображення за допомогою оператора, який описується масками на рис. 1.19, використовуючи процедуру фільтрації, описану раніше.

–1	0
0	+1

0	–1
+1	0

Рисунок 1.19 – Маски оператора Робертса

Реалізація масок розмірами 2×2 не надто зручна, тому що в них немає чітко вираженого центрального елемента, що істотно відображається на результаті виконання фільтрації. Але цей «мінус» породжує дуже

корисну властивість даного алгоритму – високу швидкість обробки зображення.

Оператор Превітта. Оператор Превітта [4], так само як і оператор Робертса, оперує з областю зображення 3×3, представленою на рис. 1.18, тільки використання такої маски задається іншими виразами:

$$Gx = (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3),$$
$$Gy = (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7).$$

У цих формулах різниця між сумами верхнього й нижнього рядків в околі 3×3 є наближеним значенням похідної по осі X, а різниця між сумами першого й останнього стовпців цього околу – похідною по осі Y. Для реалізації цих формул використовується оператор, описуваний масками (рис. 1.20), який називається оператором Превітта.

–1	–1	–1
0	0	0
1	1	1

–1	0	1
–1	0	1
–1	0	1

Рисунок 1.20 – Маски оператора Превітта

Оператор Собеля. Оператор Собеля [5] теж використовує область зображення 3×3. Він досить схожий на оператор Превітта, а видозміна полягає у використанні вагового коефіцієнта 2 для середніх елементів:

$$Gx = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3),$$
$$Gy = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7).$$

Це збільшене значення використовується для зменшення ефекту згладжування за рахунок надання більшої ваги середнім точкам. Маски, які використовуються оператором Собеля, показані на рис. 1.21.

–1	–2	–1
0	0	0
1	2	1

–1	0	1
–2	0	2
–1	0	1

Рисунок 1.21 – Маски оператора Собеля

Розглянуті вище маски застосовуються для отримання складових градієнта G_x і G_y. Для обчислення величини градієнта ці складові необхідно використовувати спільно:

$$\nabla f \approx |Gx| + |Gy|, \text{ або } f = \sqrt{Gx^2 + Gy^2} .$$

Приклади застосування описаних методів та алгоритмів наведені на рис. 1.22…1.24.

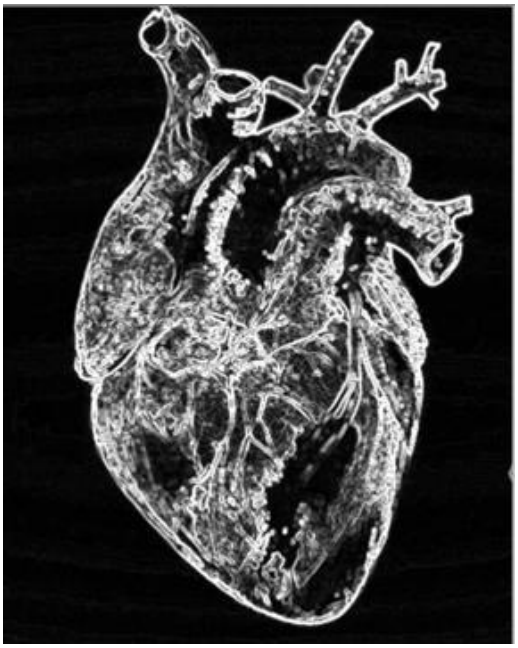


Рисунок 1.22 – Визначення кордонів із використанням алгоритму Собеля з матрицею 3×3 *у відтінках сірого*

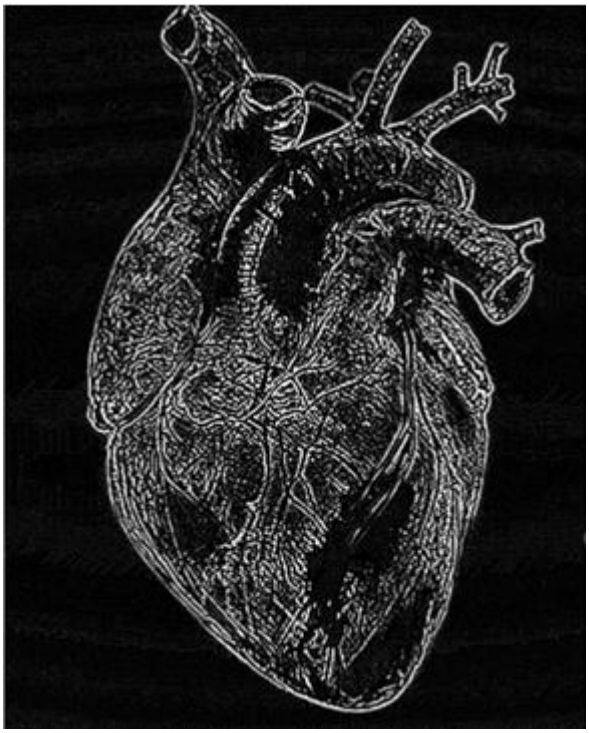


Рисунок 1.23 – Визначення кордонів з використанням алгоритму Лапласа

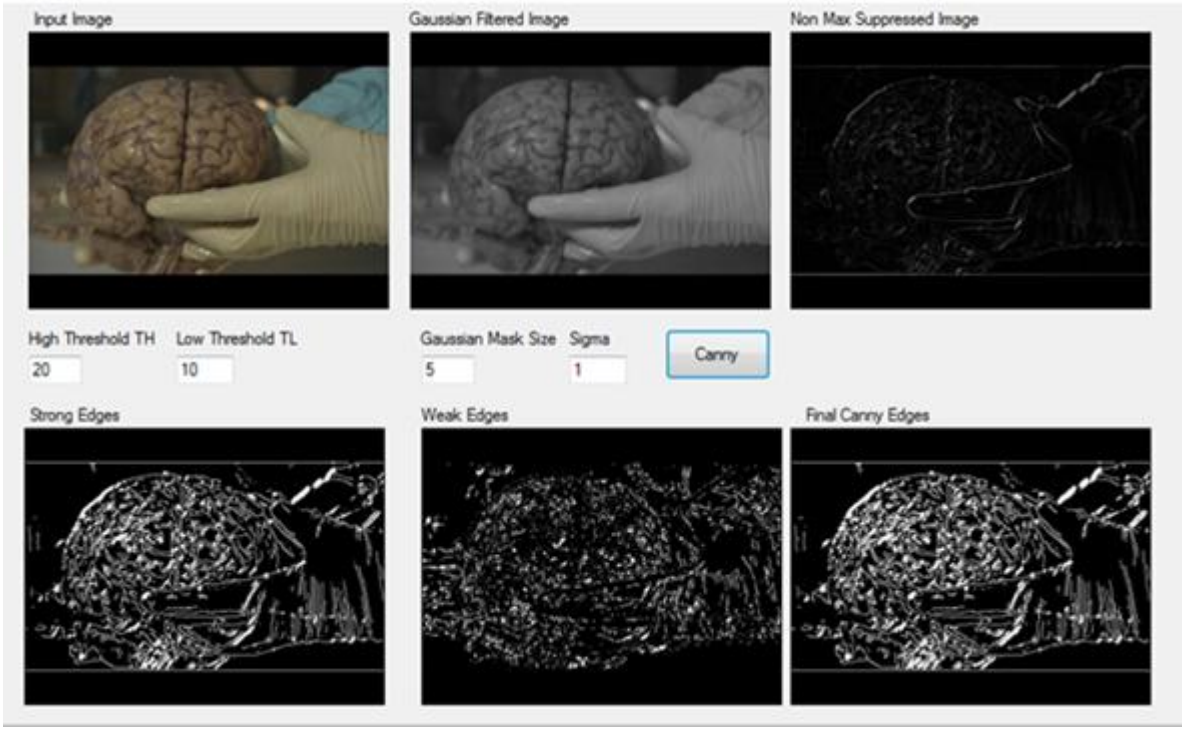


Рисунок 1.24 – 5 кроків алгоритму Кенні

2 МЕТОДИ ОБРОБКИ ЗОБРАЖЕНЬ У СИСТЕМАХ КОМП'ЮТЕРНОГО ЗОРУ

2.1 Рівні опрацювання зображень. Класифікація систем розпізнавання зображень

Галузь комп'ютерного зору може бути охарактеризована як молода, різноманітна, що динамічно розвивається. І хоча існують більш ранні роботи, можна сказати, що лише з кінця 1970-х почалося інтенсивне вивчення цієї проблеми, коли комп'ютери змогли управляти обробкою великих наборів даних, таких як зображення. Однак ці дослідження зазвичай починалися з інших сфер, і, отже, немає стандартного формулювання проблеми комп'ютерного зору.

Також немає стандартного формулювання того, як повинна вирішуватися проблема комп'ютерного зору. Замість цього існує маса методів для вирішення різних суворо визначених завдань комп'ютерного зору, де методи часто залежать від завдань і нечасто можуть бути узагальнені для широкого кола застосування. Багато методів і додатків усе ще знаходяться в стадії фундаментальних досліджень, але все більша кількість методів застосовується в комерційних продуктах, де вони часто складають частину більш великої системи, яка може вирішувати складні завдання [6].

Обробка сигналів є галуззю, що пов'язана з комп'ютерним зором. Розвиток штучних інтелектуальних систем потребує можливості з візуальної орієнтації у просторі, аналізу сцен, пошуку об'єктів з оцінюванням їх геометричних форм і кількісних характеристик. У комп'ютерному зорі реалізовані методи обробки двовимірних чи багатовимірних сигналів. Особливістю цих методів є їхня нелінійність [7].

Фізика тісно пов'язана з комп'ютерним зором. Методи комп'ютерного зору потребують досконального розуміння процесу, у якому електромагнітне випромінювання відбивається поверхнею об'єктів та вимірюється давачем зображення, щоб отримати відеодані. Цей процес ґрунтується на оптиці і фізиці твердого тіла. Складніші давачі зображення також потребують знань із квантової механіки для повного розуміння процесу формування зображення.

Багато питань комп'ютерного зору вивчаються з використанням математики і базуються на статистиці, геометрії.

Розпізнавання зображень дає комп’ютерам можливість ідентифікувати об’єкти на будь-якому зображенні. Моделі комп’ютерного зору можуть аналізувати зображення, щоб розпізнавати або класифікувати об’єкт на зображенні, а також реагувати на ці об’єкти. Розпізнавання зображень є підмножиною комп'ютерного зору. Вони значною мірою залежать від методів машинного навчання та використовують існуючі моделі, навчені на позначеному наборі даних, щоб ідентифікувати та виявляти об’єкти на зображенні чи відео [8].

Автоматичне планування (ухвалення рішень) необхідне в робототехниці для пересування робота крізь деяке середовище. Вхідні дані надаються системами комп'ютерного зору і діють як відеосенсор.

Досягнення нейробіології, зокрема вивчення біологічного зору, використовується в системах комп'ютерного зору. Результати досліджень призвели до створення штучних систем, що наслідують роботу і функціонування аналогічних біологічних систем (рис. 2.1.)

З комп’ютерним зором також пов’язано декілька сфер (рис.2.2).

Обробка зображень — будь-яка форма обробки інформації, для якої вхідні дані представлені зображенням, наприклад фотографіями або відеокадрами (рис. 2.2).

Рисунок 2.1 – Галузі, що використовуються комп'ютерним зором

Рисунок 2.2 – Галузі, пов'язані з комп'ютерним зором

Аналіз зображень – метод дослідження, який вивчає предмет, уявно чи реально розчленовуючи його на складові елементи, як-от частини об'єкта, його ознаки, властивості, відношення, відтак розглядає кожен із виділених елементів окремо в межах єдиного цілого, в основному зосереджений на роботі з двовимірними зображеннями, тобто як перетворити одне зображення на інше.

Машинний зір – зосереджується на застосуванні, в основному, промисловому, наприклад автономні роботи і системи зорової перевірки та вимірювання.

Візуалізація – процес побудови графічного образу даних, що допомагає у процесі загального аналізу даних виявляти аномалії структури, яка початково була пов'язана з процесом створення зображень, але іноді мала справу з обробкою та аналізом.

Комп'ютерний зір – теорія і технологія створення машин, які можуть здійснювати виявлення, відстеження і класифікацію об'єктів. Як наукова дисципліна, комп'ютерний зір відноситься до теорії і технології створення штучних систем, які отримують інформацію із зображень. Зосереджується на обробці тривимірних сцен, спроектованих на одне чи декілька зображень.

Розпізнавання образів — це розділ теорії штучного інтелекту, що вивчає методи класифікації об’єктів.

Комп’ютерний зір містить методи та алгоритми трьох рівнів опрацювання зображення: низького, середнього та високого (рис. 2.3…2.6). Від етапу попередньої обробки залежить якість обробки на наступних рівнях. Основними проблемами при опрацюванні зображень є значна зашумленість, розмитість, наявність затемнених або навпаки – занадто освітлених ділянок. Ці недоліки значно зменшують якість сегментації (середній рівень опрацювання), а відповідно – визначення числових характеристик об’єктів та подальше їх розпізнавання [9, 10].

На будь-яке зображення діють шуми різної природи, які впливають на подальші етапи опрацювання: сегментацію та розпізнавання. Наприклад, шум може бути ідентифікований як складова частина досліджуваного об’єкту, що може значно зменшити якість та точність обробки. Загалом виділяють два основних види шумів: адитивний гаусовий та імпульсний шум [11]. Найчастіше шум з’являється на етапі формування та передачі зображення по каналах зв’язку [10]. Фільтрацію зображень застосовують для зменшення рівня гаусових та імпульсних шумів.

Рисунок 2.3 – Рівні опрацювання зображень

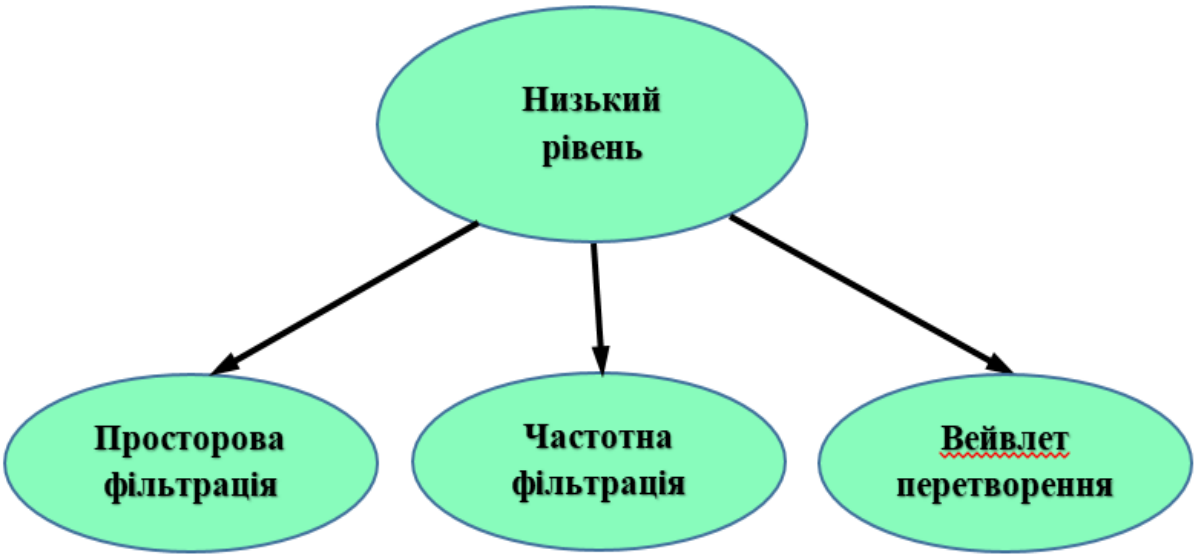


Рисунок 2.4 – Опрацювання зображень. Низький рівень

Рисунок 2.5 – Опрацювання зображень. Середній рівень

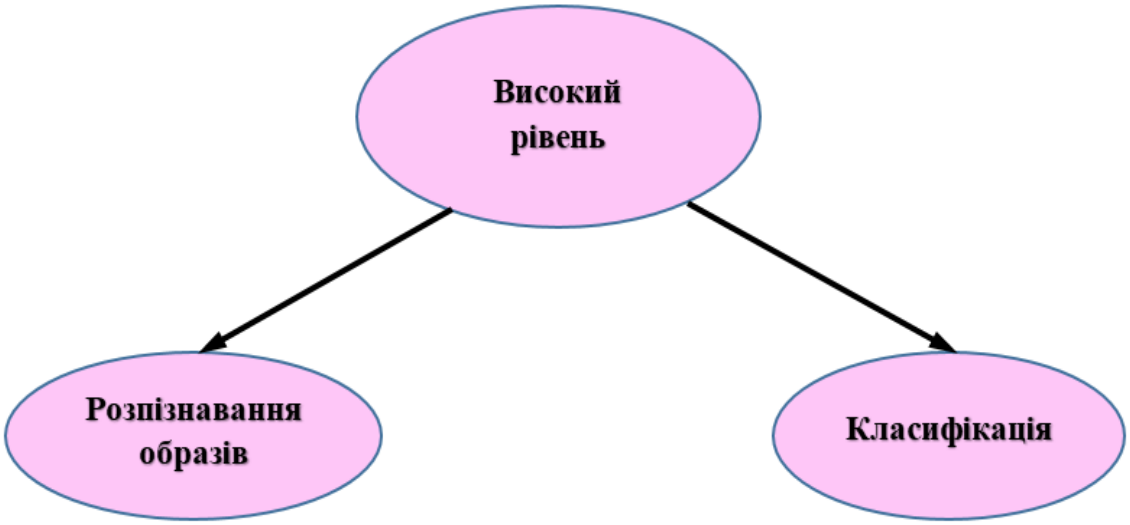


Рисунок 2.6 – Опрацювання зображень. Високий рівень

Розглянемо складові високого рівня опрацювання зображень. Типи систем розпізнавання мають декілька видів класифікації – за кількістю апріорної інформації (рис. 2.7), за характером інформації про ознаки (рис. 2.8) [12].

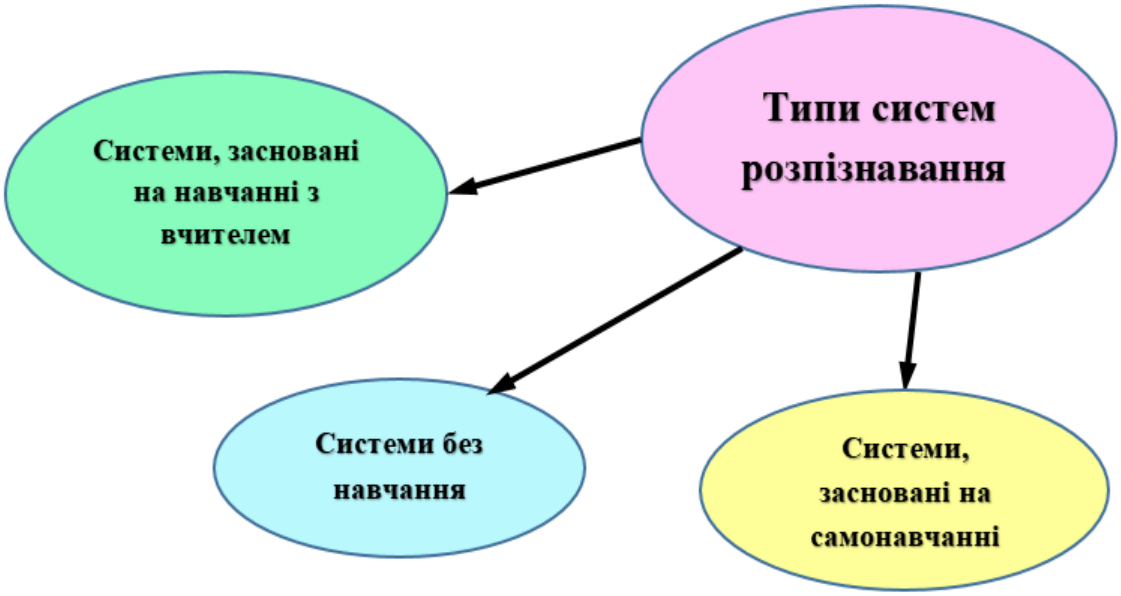


Рисунок 2.7 – Типи систем розпізнавання за кількістю апріорної інформації

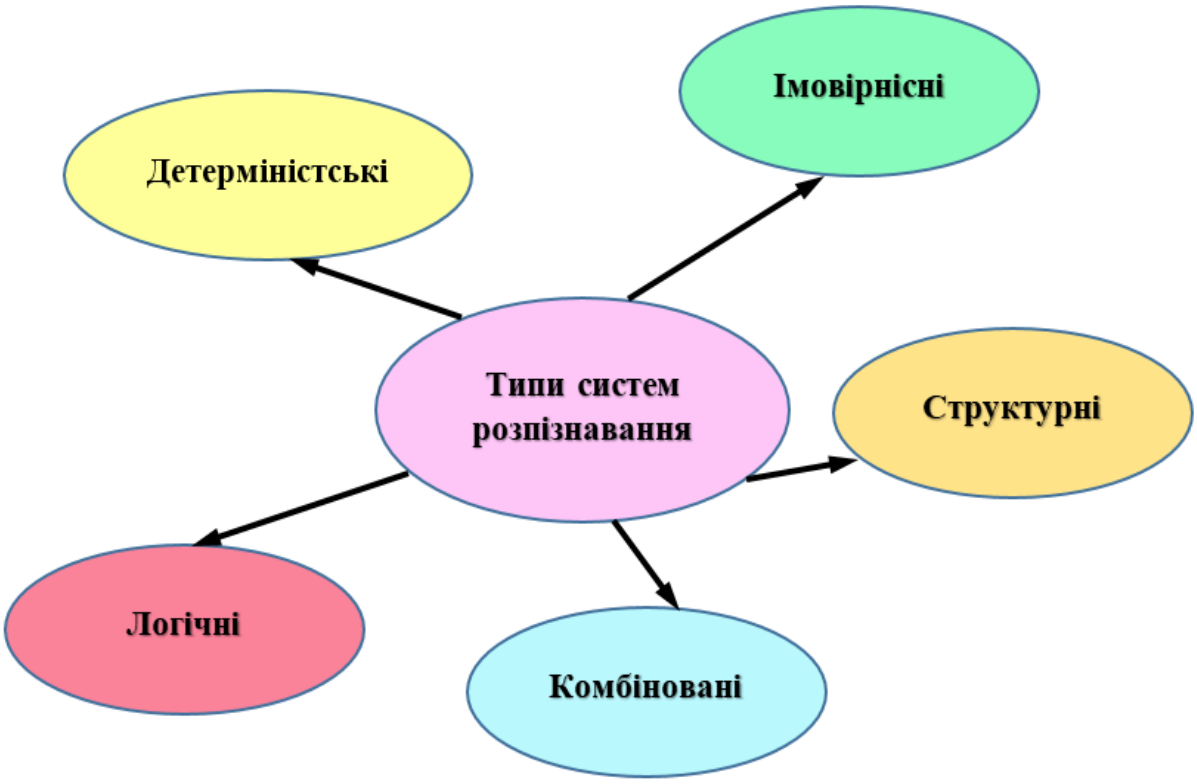


Рисунок 2.8 – Типи систем розпізнавання за характером інформації про ознаки

Детерміністський підхід (див. рис. 2.8) у теорії розпізнавання образів включає методи:

- Класифікація за допомогою вирішальних функцій
- Класифікація за допомогою функцій відстані
- Алгоритми кластеризації (векторного квантування)
- Машина (метод) опорних векторів
- Метод потенціальних функцій
- Нейронні мережі

Імовірнісний (статистичний) підхід у теорії розпізнавання образів:

- Байесовський класифікатор
- Мінімаксний критерій класифікації
- Критерій Неймана-Пірсона
- Критерії класифікації в разі нормального розподілу ознак у кожному класі
- Статистичне оцінювання імовірнісних характеристик

Методи статистичного оцінювання імовірнісних характеристик [13, 14]:

1. Відомий загальний вигляд функції щільності розподілу випадкового вектора ξ. Точний вид функції щільності розподілу імовірностей повністю визначається деяким набором параметрів. Потрібно оцінити вектор значень параметрів цієї функції щільності. У цьому випадку, як правило, використовують методи параметричного оцінювання (рис. 2.9).

2. Загальний вигляд функції щільності розподілу невідомий. Можуть бути тільки відомі деякі загальні аналітичні властивості щільності, такі як безперервність, диференцируємість, наявність обмеженого носія і т. д. Для оцінки щільності розподілу в цьому випадку, як правило, використовують непараметричні методи оцінювання (див. рис. 2.9).

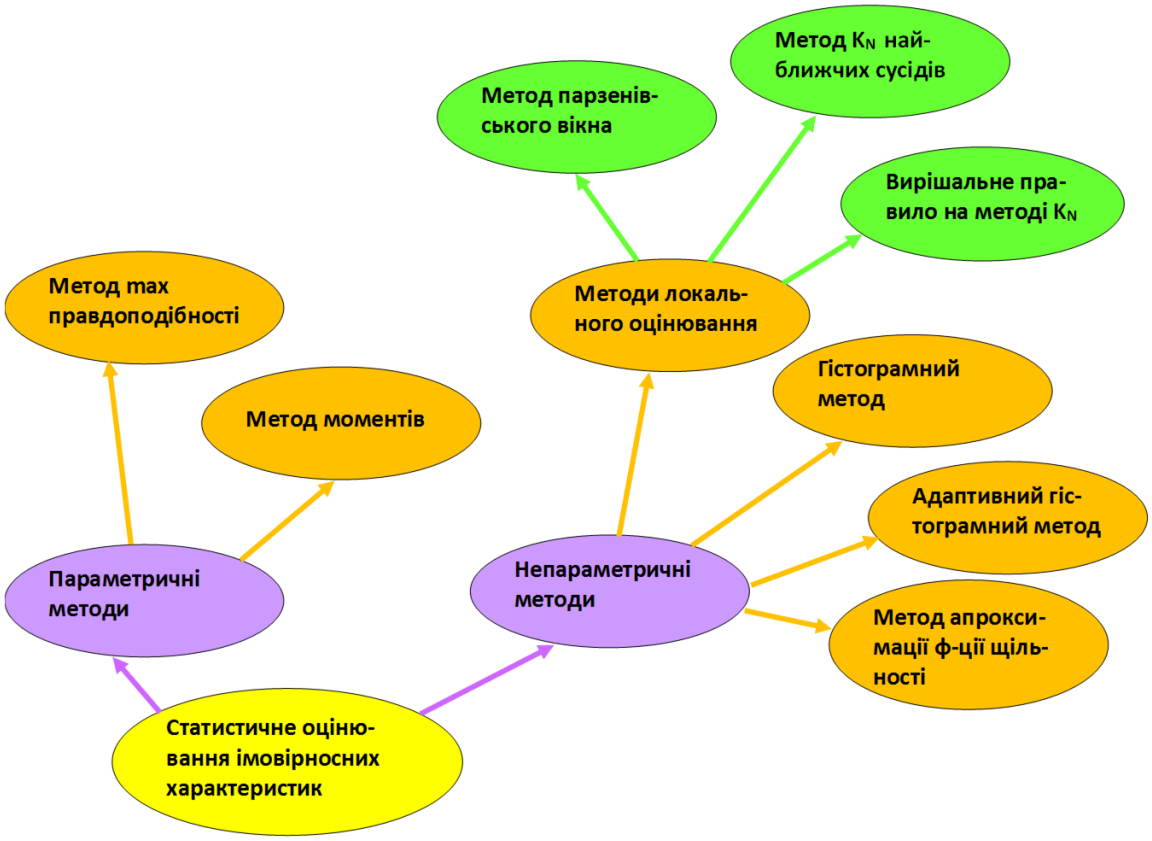


Рисунок 2.9 – Методи статистичного оцінювання імовірнісних характеристик

2.2 Основні задачі розпізнавання образів

Основні задачі теорії розпізнавання образів [15] :

- математичний опис образів;
- вибір найбільш інформативних ознак;
- опис класів розпізнаних образів;
- знаходження оптимальних вирішальних процедур;
- оцінка достовірності класифікації образів.

Під час вирішення задачі теорії розпізнавання образів проводиться ідентифікація, яка полягає в тому, щоб вирізнити певний конкретний об’єкт серед йому подібних (наприклад, упізнати серед інших людей свого друга).

Класифікація або віднесення об'єкта до того чи іншого класу. Це може бути, наприклад, задача розпізнавання літер або прийняття рішення про наявність дефекту в деякій технічній деталі. Віднесення об'єкта до певного класу відображає найтиповішу проблему класифікації. Коли говорять про розпізнавання образів, найчастіше мають на увазі саме цю проблему.

Кластерізація полягає в поділі заданого набору об'єктів на класи – групи об'єктів, схожі між собою за тим чи іншим критерієм. Цю задачу часто називають класифікацією без учителя, оскільки класи апріорно не задані.

Математична постановка задачі розпізнавання [15]. Нехай **x** – елемент простору X, відповідний образу $x \in U$, а $P: U \to X$ – оператор, що відображає x в **x**. Окрім того, X = P(U).

Припустимо, що в множині образів U у цій задачі розпізнавання нас цікавлять деякі підмножини – класи. У класичній задачі класифікації вважається, що множина класів $\Omega = \{\omega_1, ..., \omega_m\}$ є кінцевою, і класи утворюють повну групу підмножин з U (розбиття простору образів U), тобто $\bigcup_{i=1}^{m} \varpi_i = U$ і $\varpi_i \cap \varpi_j = \varnothing$ для всіх $i \neq j$.

У загальному випадку класів може бути і нескінченно багато, і вони можуть не складати повну групу множин.

Класифікувати образ $x \in U$ по класах $\omega_1, ..., \omega_m$ – це значить знайти так звану індикаторну функцію $g: U \to Y$, $Y = \{y_1, \ldots, y_m\}$, яка ставить у відповідність образу $x \in U$ мітку $y_i \in Y$ того класу ϖ_i, якому він належить, тобто $g(x) = y_i$, якщо і $x \in \varpi_i$.

Реально ми маємо справу не з усією множиною образів U, а тільки з проекцією X = P(U) – простором ознак. Тоді потрібно знайти таку функцію $\tilde{g}: X \to Y$, яка ставила б у відповідність кожному вектору $x = Px \in X$ мітку $y_i \in Y$ того класу ϖ_i, якому належить відповідний образ, тобто $\tilde{g}(\mathbf{x}) = y_i$, якщо $x = Px$, $x \in \varpi_i$. Така функція називається вирішальною.

Як правило, на етапі навчання системи розпізнавання доступна інформація про класи у вигляді деякої множини пар $(\mathbf{x}_j, y_j)$, j = 1, ..., N, де $\mathbf{x}_j = Px_j$, $y_j = g(x_j) \in Y$. Множину $\Xi = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ називають навчальною вибіркою, а пару (x_j, y_j) – прецедентом. За множиною прецедентів $(\Xi, Y) = \{(x_j, y_j): j = 1, ..., N\}$ потрібно знайти вирішальне правило – функцію $\tilde{g}(\mathbf{x})$, яка здійснювала б класифікацію елементів навчальної вибірки з найменшим числом помилок.

2.2.1 Типи вирішальних правил

Лінійний класифікатор [15]

Нехай $X1$ і $X2$ – два класи розпізнавання.

Реалізації класів розпізнавання можна розділити прямою:

$$d(\boldsymbol{x}) = w0 + w1x1 + w2x2 = 0.$$

При цьому значення коефіцієнтів $w0$, $w1$, $w2$ визначено так, що для всіх реалізацій класу $X1$ виконується нерівність $d(\boldsymbol{x}) > 0$, а для всіх реалізацій класу $X2$ – $d(\boldsymbol{x}) < 0$. Функцію $d(\boldsymbol{x})$ називають лінійним вирішальним правилом класу $X1$ (рис. 2.10).

Для N-вимірного випадку $\boldsymbol{x} = (x1, x2, \dots, xN)$ $\boldsymbol{T}$ пряма перетворюється на гіперплощину:

$$d(\boldsymbol{x}) = w0 + w1x1 + w2x2 + \cdots + wNxN = w0 + \boldsymbol{wTx} = 0$$

де $\boldsymbol{w} = (w1, w2, \dots, wN)$ – вектор коефіцієнтів.

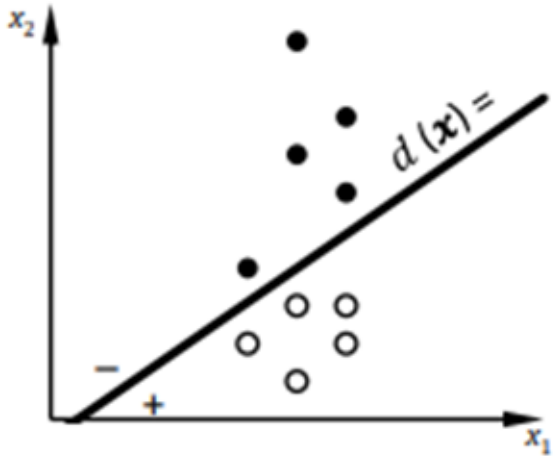


Рисунок 2.10 – Лінійне вирішальне правило

Паралельний класифікатор

Це проста, багатовимірна порогова розв'язувальна функція (рис. 2.11).

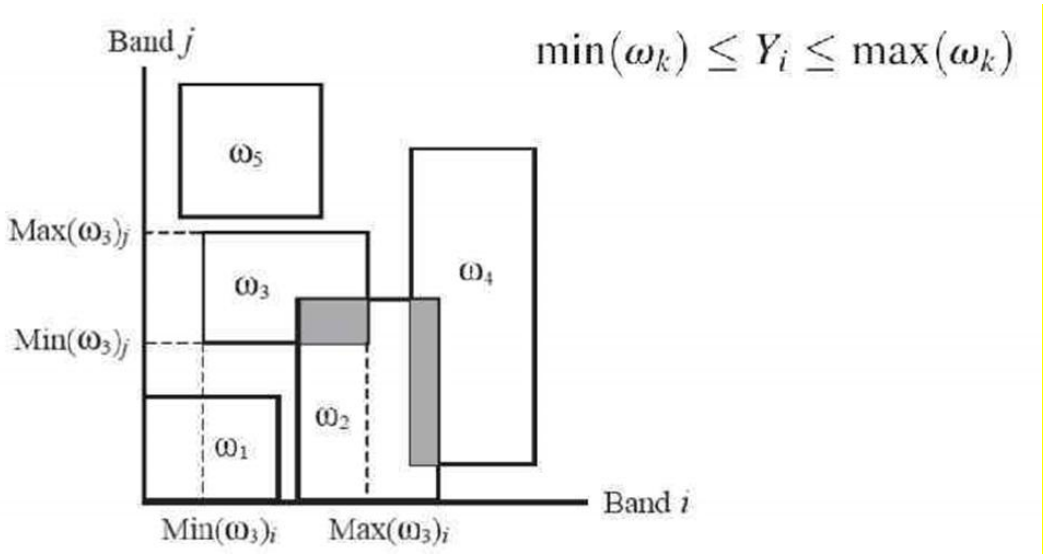


Рисунок 2.11 – Порогова розв'язувальна функція

Метод евклідових відстаней

Метод евклідових відстаней (спрощений метод найбільшої подібності) [16] (рис. 2.12) базується на однаковості стандартних відхилень в усіх кластерах.

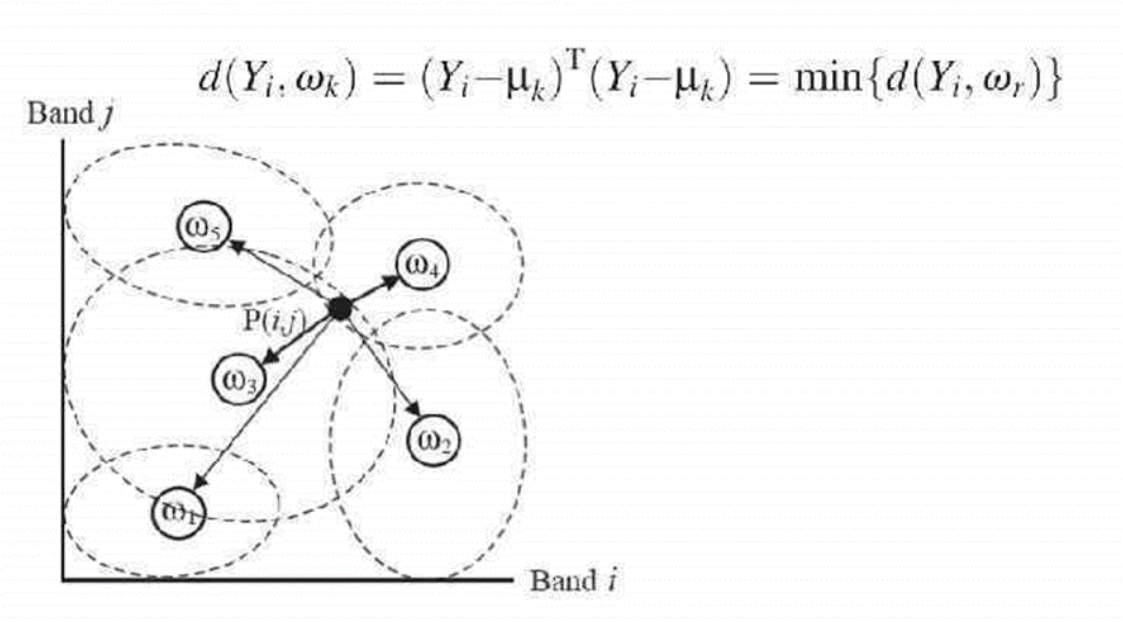


Рисунок 2.12 – Метод евклідових відстаней

Метод максимальної подібності

Базується на розв'язувальному правилі Баєса [16] за умов однакового нормального розподілу даних в усіх кластерах.

$$\text{classify}(f_1, \ldots, f_n) = \underset{c}{\text{argmax}}\ P(C = c) \prod_{i=1}^{n} p(F_i = f_i | C = c)$$

$$\delta(Y_i, \omega_k) = \ln|\Sigma_k| + (Y_i - \mu_k)^T \Sigma_k^{-1} (Y_i - \mu_k)$$

$$-\ln \frac{N_k}{N} = \min\{\delta(Y_i, \omega_r)\}$$

Крім того, існують класифікатори, що базуються на методах нечіткої логіки, нейронних мереж, лінійні класифікатори, баєсівські сітки довіри тощо.

2.2.2 Критерії якості розпізнавання зображень

Якість класифікації об’єктів істотно залежить від вибору міри їхньої близькості (подібності) [15].

Ентропійний підхід ґрунтується на понятті ентропії, введеної в теорії інформації Шеннона. У якості критерію вибору рішення виступає ентропія

значень, що зберігаються в пікселях зображення. Відповідно до принципу мінімуму ентропії, кращим значенням вважається те, для якого ентропія мінімальна.

Нормований ентропійний критерій функціональної ефективності за Шенноном має вигляд

$$E = \frac{H_0 - H(\gamma)}{H_0},$$

де $H_0 -$ апріорна (безумовна) середня ентропія;

$H(\gamma) -$ апостеріорна умовна ентропія, що характеризує залишкову невизначеність після прийняття рішень.

Байєсовські методи визначення критерію якості розпізнавання зображень є найбільш розробленими і широко вживаними на практиці. Рішення в них будується на основі апостеріорної ймовірності, що обчислюється за правилом Байеса [15].

$$E = 1 + \frac{1}{2}\left(\frac{a_m^{(k)}(d)}{a_m^{(k)}(d) + D_{2,m}^{(k)}(d)} \log_2 \frac{a_m^{(k)}(d)}{a_m^{(k)}(d) + D_{2,m}^{(k)}(d)} + \right.$$

$$+ \frac{\beta_m^{(k)}(d)}{\beta_m^{(k)}(d) + D_{1,m}^{(k)}(d)} \log_2 \frac{\beta_m^{(k)}(d)}{\beta_m^{(k)}(d) + D_{1,m}^{(k)}(d)} +$$

$$+ \frac{D_{1,m}^{(k)}(d)}{\beta_m^{(k)}(d) + D_{1,m}^{(k)}(d)} \log_2 \frac{D_{1,m}^{(k)}(d)}{\beta_m^{(k)}(d) + D_{1,m}^{(k)}(d)} +$$

$$\left. + \frac{D_{2,m}^{(k)}(d)}{a_m^{(k)}(d) + D_{2,m}^{(k)}(d)} \log_2 \frac{D_{2,m}^{(k)}(d)}{a_m^{(k)}(d) + D_{2,m}^{(k)}(d)} \right),$$

де $a_m^{(k)}(d)$ – помилка першого роду прийняття рішення на k-му кроці навчання;

$\beta_m^{(k)}(d)$ – помилка другого роду;

$D_{1,m}^{(k)}(d)$ – перша достовірність;

$D_{2,m}^{(k)}(d)$ – друга достовірність;

d – дистанційна міра, що визначає відстань контейнера, який відновлюється в процесі навчання в радіальному базисі простору ознак розпізнавання, від геометричного центру класу розпізнавання.

Приймаємо апріорною гіпотезу $\boldsymbol{\gamma_1}$ знаходження значення ознаки розпізнавання в полі допусків δ, а за альтернативну гіпотезу $\boldsymbol{\gamma_2}$ – знаходження ознаки за межами поля допусків. При цьому як апостеріорні, відповідно, приймемо гіпотези $\boldsymbol{\mu_1}$ і $\boldsymbol{\mu_2}$ (табл. 2.1)[17].

$p(\gamma_2/\ \mu_1)$ – це помилка першого роду;

$p(\gamma_1/\ \mu_2)$ – помилка другого роду;

$p(\gamma_1/\ \mu_1)$ – перша достовірність D1;

$p(\gamma_2/\ \mu_2)$ – друга достовірність D2.

Таблиця 2.1 – Матриця помилок

Прогноз	**Реальність**	
	+	–
+	**True Positive** **TP – істинно-позитивне рішення: прогноз збігся з реальністю, результат позитивний стався, як і було передбачено ML-моделлю**	**False Positive** **FP – хибно позитивні рішення:** **помилка 1-го роду, ML-модель передбачила позитивний результат, а насправді він негативний**
–	**False Negative** **FN – хибно негативне рішення:** **помилка 2-го роду, ML-модель передбачила негативний результат, але насправді він позитивний**	**True Negative** **TN – істинно-негативне рішення:** **результат негативний, ML-прогноз збігся з реальністю**

Теоретико-інформаційні методи базуються на принципі мінімальної довжини опису. Підхід ґрунтується на алгоритмічній теорії інформації, у якій кількість інформації, що міститься в деякому наборі даних, дорівнює довжині мінімальної підпрограми, здатної відтворити ці дані.

Розпізнавання зображень відбувається без прямого звернення до алгоритмічної теорії інформації. У розглянутому підході задається опис (довжина якого замінює значення правдоподібності), за яким можна відновити вихідне зображення, використовуючи явне представлення зображення.

2.3 Попередня обробка даних

Щоб уникнути появи ситуації *«сміття на вході, сміття на виході»*, підвищити якість даних і, як наслідок, ефективність моделі, необхідно провести моніторинг працездатності даних, як можна раніше виявити проблеми і вирішити, які дії з попередньої обробки і очищення даних необхідні. Маємо таки методи [18]:

– нормалізація

– масштабування

– виключення середнього

– бінаризація

Залежно від функції, що використовується, є дві великі групи способів нормалізації: лінійні і нелінійні (рис. 2.13).

У *лінійної* нормалізації зміна змінних здійснюється пропорційно, за лінійним законом.

При *нелінійній* нормалізації в розрахункових формулах використовуються функції логістичної сигмоїди або гіперболічного тангенса.

$$X_{норм} = \frac{1}{1+e^{-a(x_{ik} - x_{ci})}}$$

де центр інтервалу, що нормалізується, зміни вхідної змінної

$$x_{ci} = (x_{\min i} + x_{\max i})/2.$$

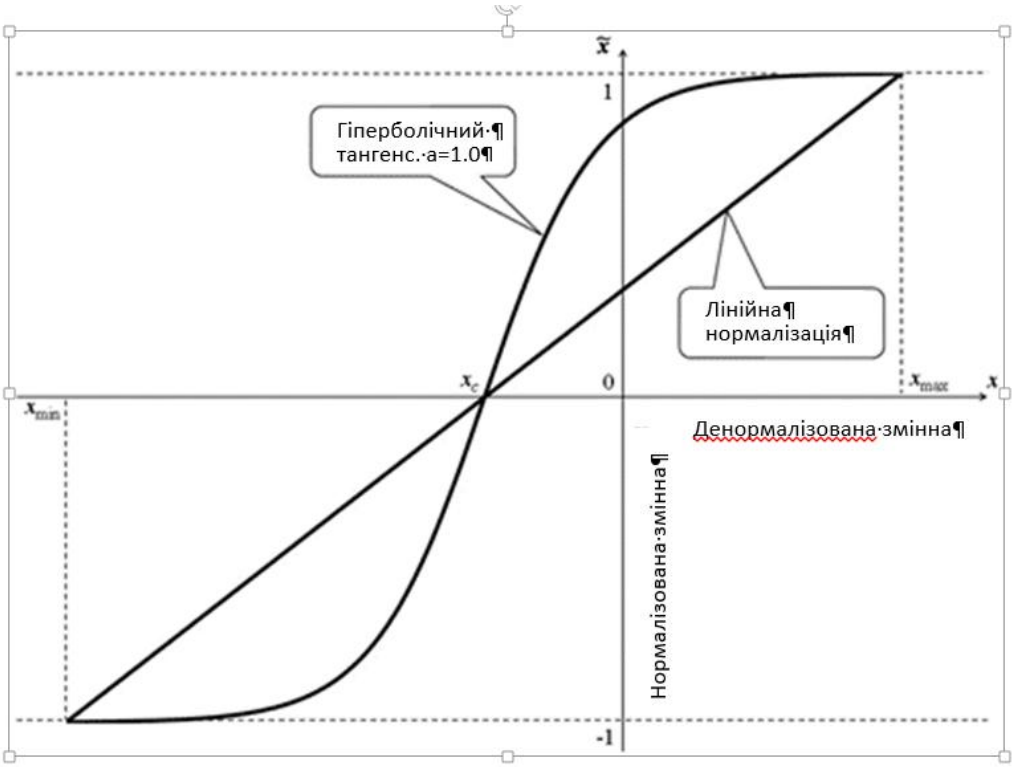


Рисунок 2.13 – Лінійна і нелінійна нормалізація

Аналітично будь-яка нормалізація зводиться до формули

$$x_{норм} = (x_i - x_{зміщ})/ x_{од},$$

де x_i – поточне значення;

$x_{зміщ}$ – величина зміщення значень;

$x_{од}$ – величина інтервалу, який буде перетворений до «одиниці».

По суті все зводиться до того, що вихідний набір значень спершу зміщується, а потім масштабується.

Робота з викидами

Найбільш масово вживаним методом автоматичного визначення викидів є міжквартильний метод (рис. 2.14) [19, 20, 21]. Його суть полягає в тому, що викидами «призначаються» дані, які більш ніж в 1,5

міжквартильних діапазонах (IQR) нижче першого квартиля або вище третього квартиля.

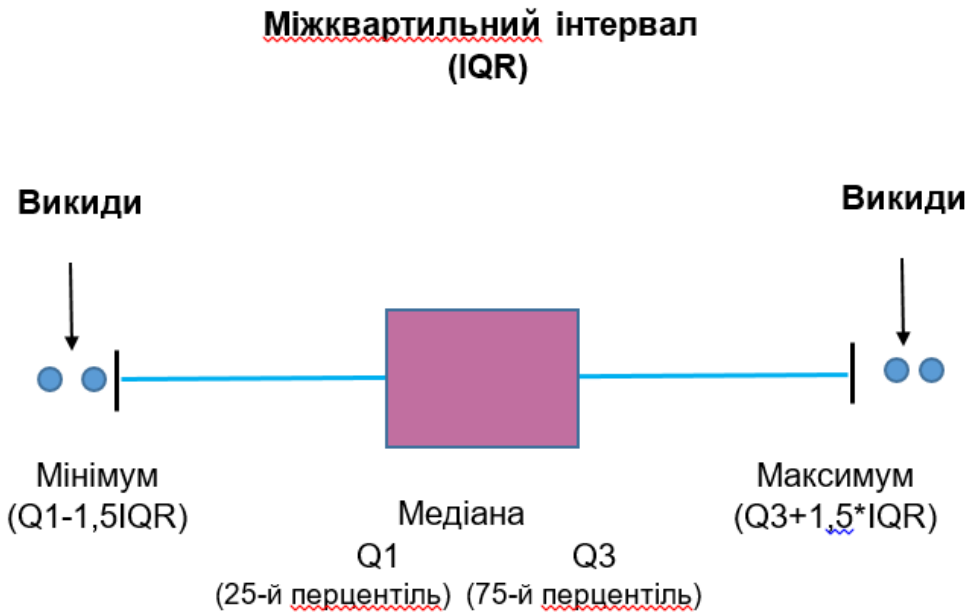


Рисунок 2.14 – Міжквартильний метод

• У разі наявності довгих хвостів (як, наприклад, при експоненційному розподілі) занадто багато даних потрапляють в такі «викиди» (рис. 2.15).

• Ще однією суттєвою проблемою метода є те, що він симетричний. Отриманий «інтервал довіри» (1,5 IQR) однаковий як для малих, так і для великих значень ознаки. Якщо розподіл не симетричний, то багато аномалій-викидів із «короткої» сторони просто будуть приховані цим інтервалом.

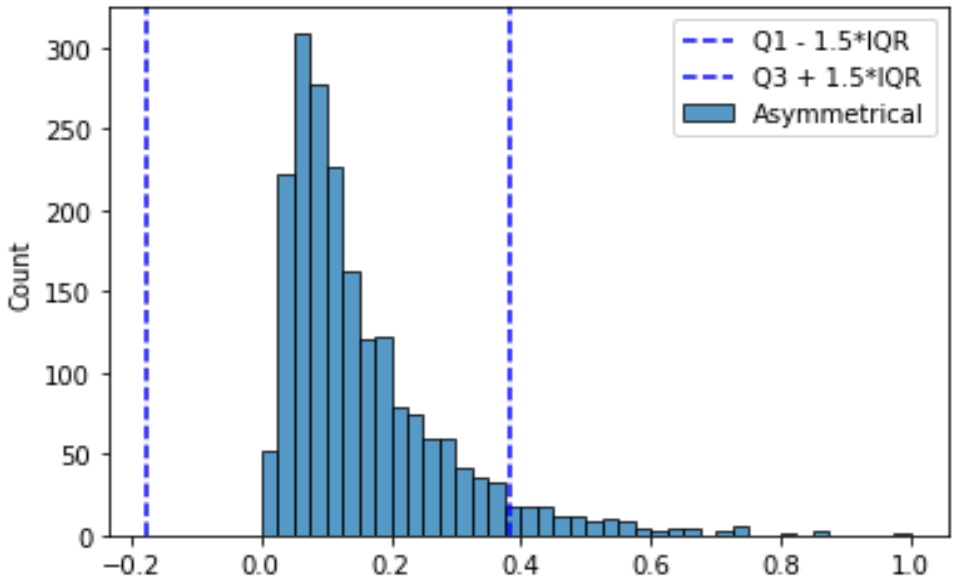


Рисунок 2.15 – Робота з викидами

Скоригований інтервал

Вирішення проблеми запропонували Міа Хаберт і Олена Вандервірен у 2007 р. Їх ідея полягає в обчисленні меж «інтервал довіри» з урахуванням асиметрії розподілу, але щоб для симетричного випадку він дорівнював 1,5 IQR.

Для визначення якогось «коефіцієнта асиметрії» вони використовували функцію medcouple (MC), яка визначається так:

$$MC = \underset{x_i \le Q_2 \le x_j}{\text{med}} \Big(h\big(x_i,\, x_j\big)\Big),$$

де $\mathrm{h}\big(x_i,\, x_j\big) = \frac{(x_i - Q_2) - (Q_2 - x_j)}{x_i - x_j}$.

Міжквартильний метод

Пошук формули для визначення кордонів «інтервалу довіри» проводився з метою зробити частку, що припадає на викиди, що не перевищує таку ж, як у нормального розподілу і 1,5 IQR – приблизно 0,7 %.

$\mathrm{MC} \ge 0 \qquad [Q_1 - 1.5e^{-4MC}IQR;\ Q_3 + 1.5e^{3MC}IQR]$

$\mathrm{MC} \le 0 \qquad [Q_1 - 1.5e^{-3MC}IQR;\ Q_3 + 1.5e^{4MC}IQR]$

Порядок попередньої обробки даних

- Центрування, якщо воно потрібне, проводити по медіані.
- Масштабувати набір даних за величиною скоригованого інтервалу.
- Якщо центрування не потрібно, то зміщувати масштабовані дані так, щоб кордони скоригованого інтервалу припадали на [0...1].

Порівняємо результати звичайних методів із новим. Порівнювати будемо з методом стандартизації (рис. 2.16, 2.17).

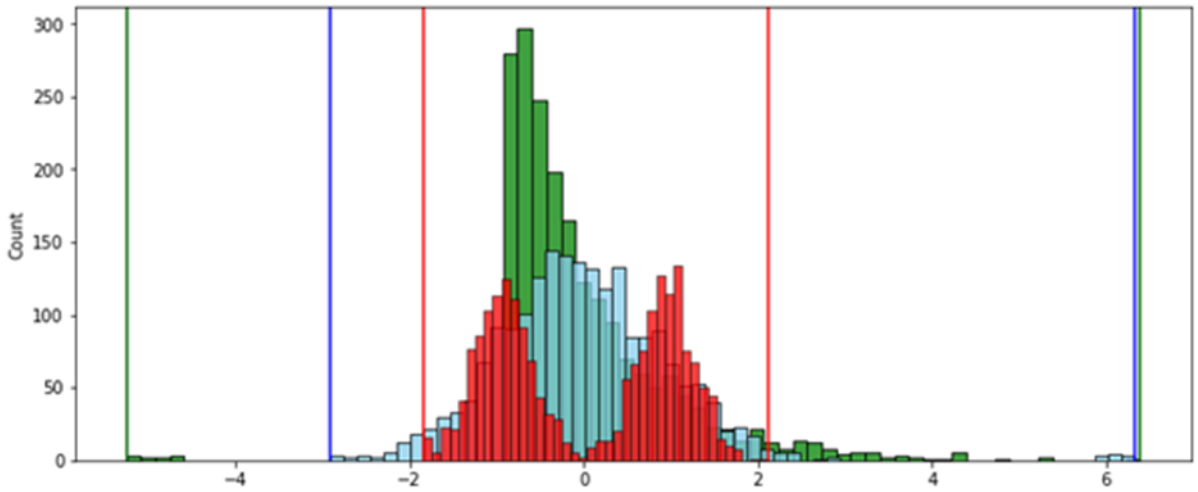


Рисунок 2.16 – Робота з викидами. Стандартизація

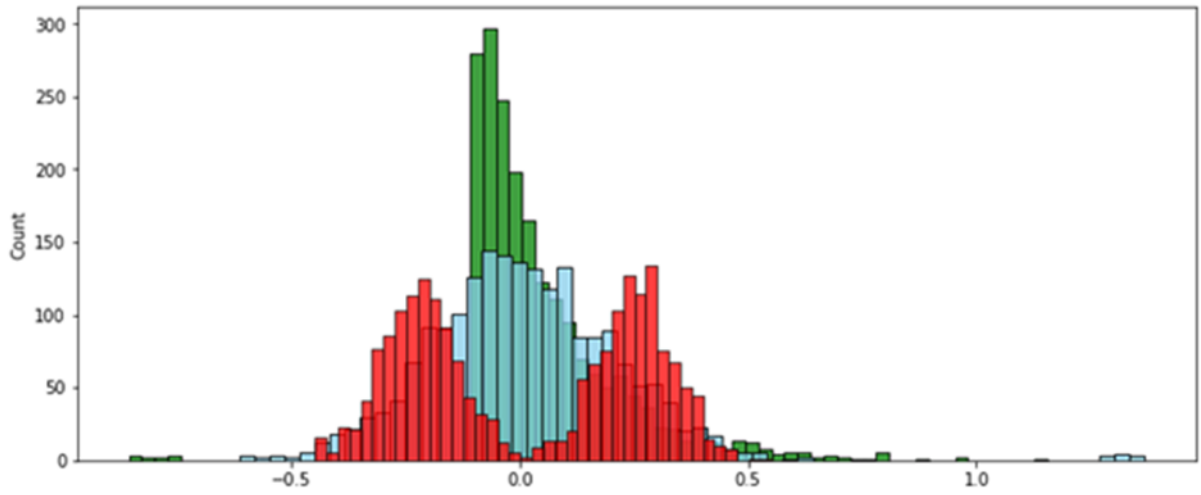


Рисунок 2.17 – Робота з викидами. Скоригований інтервал

2.4 Навчання з учителем. Навчання без учителя

Навчання з учителем (supervised learning; інша назва: контрольоване навчання) – це процес створення моделі машинного навчання, заснованої на маркованих (позначених) навчальних даних.

Навчання без вчителя (unsupervised learning; інші назви: неконтрольоване навчання, самонавчання, спонтанне навчання) – це процес створення моделі машинного навчання, не заснованої на використанні маркованих тренувальних даних. У цьому випадку, зважаючи на відсутність будь-яких відмітних міток, доводиться виявляти відносини між даними, виходячи лише із самих даних.

2.4.1 Класифікація на основі навчання з учителем

Наївний байесовський класифікатор. Це простий класифікатор, заснований на використанні теореми Байеса, яка описує ймовірність події з урахуванням пов'язаних із нею умов [22, 23].

Такий класифікатор створюється за допомогою присвоювання міток класів екземплярам задачі. Останні подаються у вигляді векторів значень ознак. При цьому передбачається, що значення будь-якої заданої ознаки не залежить від значень інших ознак. Це припущення про незалежність розглянутих ознак і становить наївну частину байесівського класифікатора.

Завдання побудови найкращого класифікатора еквівалентне такій розбивці простору R^n на непересічні частини X1 і X2, яка мінімізувала б середню помилку неправильної класифікації Q.

Розглянемо спочатку для простоти випадок двох класів {ϖ1, ϖ2}. Визначено деяке розбиття простору ознак R^n на дві області X1 и X2: X1∩ X2=∅, X1∪ X2 = R^n . Причому, будемо вважати, що область Xi є областю переваги класу ϖi, i = 1,2, тобто образ x ∈ ϖi , якщо x ∈ Xi. Тоді ймовірність неправильної класифікації можна обчислити за формулою

$$Q = \int p\,X1\,(\varpi 1\,|\mathbf{x}) f(\mathbf{x}) d(\mathbf{x}) + \int p\,X2\,(\varpi 1\,|\mathbf{x}) f(\mathbf{x}) d(\mathbf{x}).$$

Перший доданок у формулі дорівнює ймовірності віднесення вектора x класу ϖ1, якщо насправді він належить класу ϖ2 (так звана ймовірність Q2 помилки другого роду). Другий доданок – ймовірність Q1помилки першого роду. Величина Q – середня помилка неправильної класифікації. Звідси випливає, що завдання побудови найкращого класифікатора еквівалентне такій розбивці простору R^n на непересічні частини X1 і X2 , яка мінімізувала б середню помилку неправильної класифікації Q.

Це завдання узагальнюється для випадку m класів. Потрібно розбити простір R^n на такі непересічні області X1, ..., Xm, щоб середня помилка неправильної класифікації була мінімальною:

$$Q = \sum_{k=1}^{m} \int_{R^n/X_k} p(\varpi_k|\boldsymbol{x}) f(\boldsymbol{x}) d(\boldsymbol{x}).$$

Чисельна оцінка якості алгоритму. При роботі нейронної мережі можливі рішення представлені в матриці помилок (confusion matrix). Міра точності характеризує, скільки отриманих від алгоритму позитивних відповідей є правильними [24]:

$$Accuracy = P/N$$

де P – кількість даних, за якими класифікатор прийняв правильне рішення, N – розмір навчальної вибірки.

Однак ця міра не дає інформації про те, чи всі правильні відповіді повернув алгоритм.

Для визначення якості роботи нейронних мереж і порівняння їх результатів розглянуті наступні метрики: функція втрат (Loss), точність (Precision), повнота (Recall) і F1-міра. На виході нейронної мережі в задачі класифікації представлені ймовірності, які в сумі повинні дати одиницю.

Точність (precision) і повнота (recall) є метриками, які використовуються при оцінці здебільшого алгоритмів вилучення інформації. Іноді вони використовуються самі по собі, іноді у якості базису для похідних метрик, таких як F-міра або R-Precision:

– *точність системи* в межах класу – це частка даних, що дійсно належать певному класу щодо всіх даних, які система віднесла до цього класу;

– *повнота системи* – це частка знайдених класифікатором даних, що належать класу щодо всіх даних цього класу у тестовій вибірці.

У реальному житті максимальна точність і повнота недосяжні одночасно, і доводиться шукати баланс. Тому використовують метрику, яка об'єднує в собі інформацію про точність та повноту нашого алгоритму. Саме такою метрикою є *F-міра* – це гармонійне середнє між точністю і повнотою. Вона прагне до нуля, якщо точність або повнота прагнуть до нуля:

$$F=2\frac{\text{Precision}*\text{Recall}}{\text{Precision}+\text{Recall}};$$

$$F=(\beta^2+1)\frac{\text{Precision}*\text{Recall}}{\beta^2*\text{Precision}+\text{Recall}}.$$

Можливо розрахувати F-міру, надавши різну вагу точності і повноті (рис. 2.18).

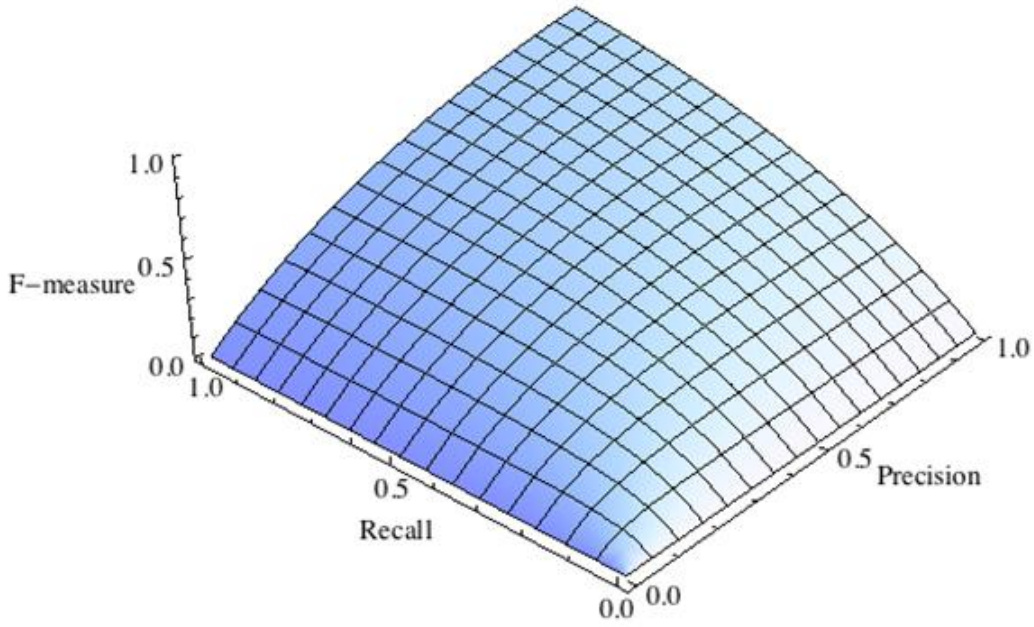


Рисунок 2.18 – Збалансована F-міра

2.4.2 Навчання без учителя

Термін навчання без вчителя (unsupervised learning) відноситься до процесу побудови моделі машинного навчання, яка не потребує залучення помічених тренувальних даних.

Машинне навчання без учителя застосовується в багатьох сферах, включаючи машинний зір, сегментування ринку, торгівлю акціями, обробку природної мови та ін.

Раніше ми мали справу з даними, з якими асоціювалися мітки (маркери). Зараз ми просто маємо дані і повинні розподілити їх за категоріями.

Коли в нас є набір даних, які не асоціюються з будь-якими мітками, ми припускаємо, що ці дані генеруються під впливом скритих змінних, керуючих їх розподілом.

У такому випадку процес навчання може слідувати ієрархічній схемі, використовуючи на початковому етапі індивідуальні точки даних. Далі можна створювати більш глибокі рівні представлення даних.

Кластеризація даних за допомогою методу k-середніх [25, 26]

Кластеризація – це процес організації даних у підгрупі, елементи якої подібні між собою згідно з деякими критеріями.

Одна зі стратегій полягає в тому, щоб центроїди розташовувалися на якомога більшій відстані один від одного для швидкої збіжності. Базовому методу k-середніх відповідає випадкове розташування центроїдів, в удосконаленому варіанті метода (k-means ++) ці точки вибираються алгоритмічно із вхідного списку точок даних.

Потім перебираються дані навчального набору і покращується стартове розбиття на кластери за допомогою віднесення кожної точки до найближчого кластерного центру.

Завершення описаного перебору всіх точок набору даних означає закінчення першої ітерації. На цьому етапі точки виявляються згрупованими на підставі початкових положень центрів кластерів.

Далі необхідно заново обчислити положення центроїдів, відштовхуючись від нових кластерів, отриманих у кінці першої ітерації. Отримавши новий набір К центрів, ми повторюємо весь процес, із набором даних і відносячи кожну точку до найближчого центроїду.

У процесі повторення описаних кроків центри кластерів поступово зміщуються до своїх стійких положень. Після виконання деякої кількості ітерацій центри кластерів перестануть зміщуватися. Це означає, що досягнуто стійке розташування центрів кластерів. Отримані К центроїдів представляють остаточну модель k-середніх.

Метод зсуву середнього (Mean Shift) – потужний алгоритм, який використовується в навчанні без учителя [27, 28, 29].

Цей непараметричний алгоритм часто застосовується для вирішення завдань кластеризації. Він називається непараметричним, оскільки в ньому не використовуються будь-які припущення щодо базового розподілу даних.

Алгоритм передбачає використання ієрархічної стратегії. В алгоритмі зсуву середнього весь простір ознак розглядається як функція розподілу ймовірності. Ми починаємо з тренувального набору даних і припускаємо, що ця вибірка відповідає функції розподілу ймовірності. У рамках такого підходу кластери відповідають максимумам базового розподілу. Якщо існують К кластерів, то в базовому розподілі існують К піків, і метод зсуву середнього ідентифікує ці піки. Метою методу зсуву середнього є ідентифікація позицій центроїдів кластерів. Для кожної точки даних навчального набору визначається її навколишнє вікно. Для цього вікна обчислюється центроїд, і положення вікна оновлюється так, щоб воно відповідало положенню нового центроїда. Далі процес повторюється для нового центроїда за допомогою визначення вікна навколо нього. У міру продовження описаного процесу ми наближаємося до піку кластера. Кожна точка даних буде переміщатися в напрямку кластера, якому вона належить. Це переміщення здійснюється в напрямку області з більш високою щільністю ймовірності. Ми продовжуємо процес зміщення центроїдів, також званих середніми, до піків кожного кластера. Оскільки середні при цьому зміщуються, метод і називається *зсув середнього*. Цей процес триває до тих пір, поки алгоритм не зійдеться, тобто поки центроїди не перестануть зміщуватися. Алгоритм зсуву середнього заснований на обчисленні оцінки щільності розподілу точок у просторі (рис. 2.19, 2.20).

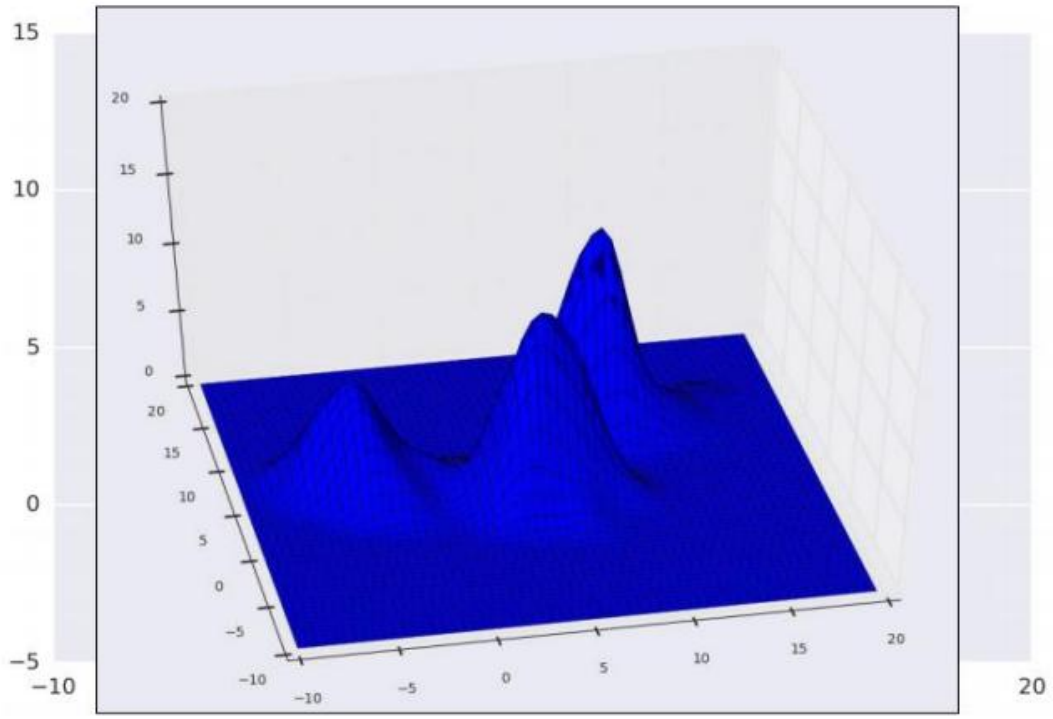


Рисунок 2.19 – Оцінки щільності розподілу точок у просторі

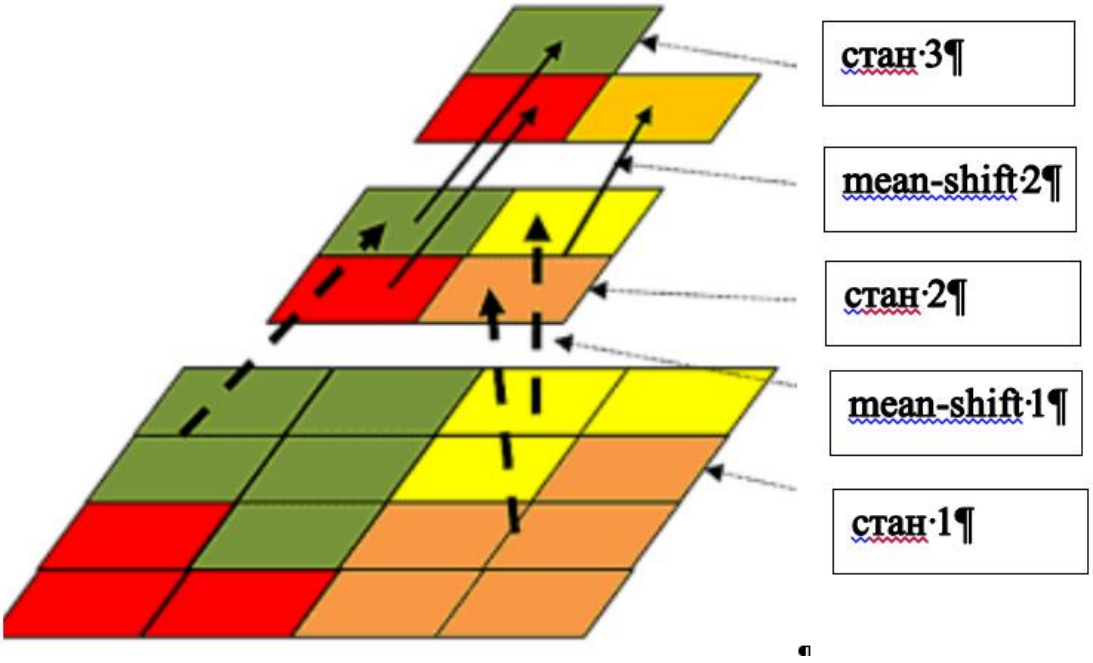


Рисунок 2.20 – Робота алгоритму зсуву середнього

2.4.3 Оцінка якості кластеризації за допомогою силуетних оцінок

Силуетна міра – це інтегральна характеристика зв'язності і поділу кластерів даних. Вона дає оцінку того, наскільки добре кожна точка даних вписується у свій кластер.

Силуетна оцінка (silhouette score) – це метрика, що вимірює ступінь подібності точки даних із власним кластером у порівнянні з іншими кластерами [30, 31].

Силуетна оцінка обчислюється для кожної точки даних:

$$\text{Силуетна оцінка} = (p - q) / \max(p, q),$$

де p – середня відстань до точок найближчого кластера, частиною якого ця точка не є, q – середня відстань до всіх точок у кластері цієї точки.

Значення силуетної оцінки – від –1 до 1. Значення, близькі до 1, указують на тісну схожість точки даних з іншими точками цього кластера, значення, близькі до –1, указують на відсутність такої подібності (рис. 2.21, 2.22).

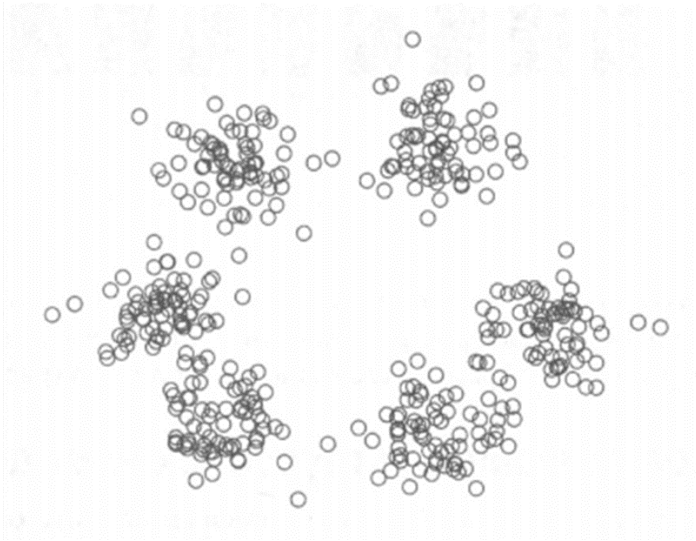


Рисунок 2.21 – Візуалізовані вхідні дані

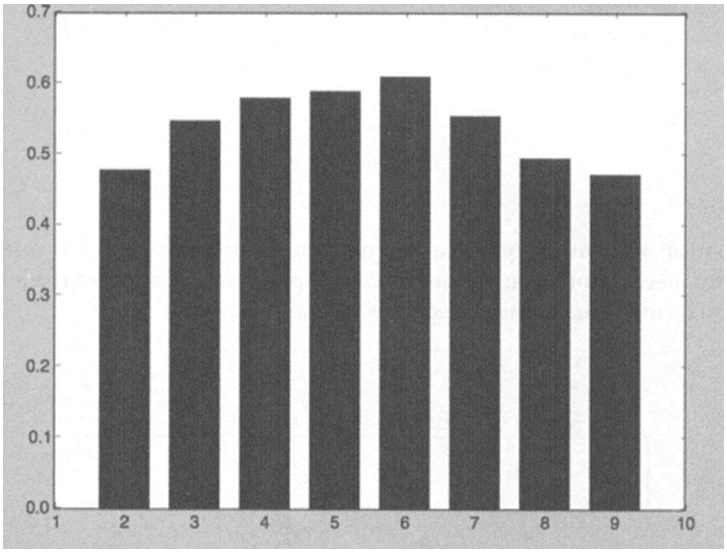


Рисунок 2.22 – Залежність силуетної оцінки від кількості кластерів

2.5 Використання нейронних мереж у розпізнаванні образів

2.5.1 Архітектура нейронних мереж

Архітектура нейронних мереж для класифікації образів. Одним із головних недоліків звичайних нейронних мереж (рис. 2.23…2.25) є те, що вони ігнорують структуру вхідних даних. Перш ніж вхідні дані передаються мережі, усі вони перетворюються на одновимірний масив. Такий підхід прийнятний для звичайних даних, але в разі зображень ситуація ускладнюється.

Рисунок 2.23 – Перцептрон, архітектура НМ для класифікації образів

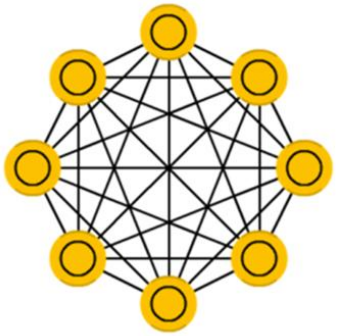


Рисунок 2.24 – Нейронна мережа асоціативної пам'яті

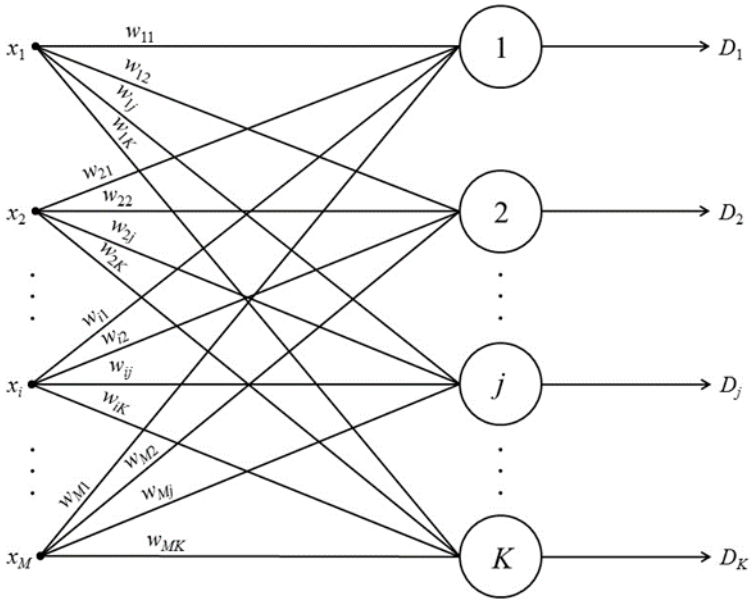


Рисунок 2.25 – Нейронна мережа Кохонена

Розглянемо зображення в градаціях сірого. Ці зображення є 2D-структурами, і ми знаємо, що просторове розташування пікселів несе в собі масу прихованої інформації. Проігнорувавши цю інформацію, ми втратимо безліч базових закономірностей (шаблонів). І тут на допомогу приходять згорткові нейронні мережі (CNN). При обробці зображень CNN (рис. 2.26) ураховують їхню 2D-структуру [32, 33].

На відміну від мереж прямого поширення, які працюють із даними у вигляді векторів, згорткові мережі працюють із зображеннями у вигляді тензорів. Тензори – це 3D-масиви чисел або масиви матриць чисел.

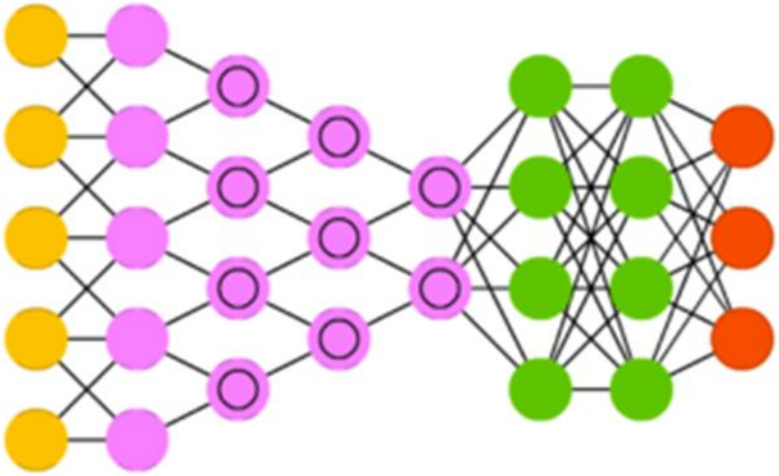


Рисунок 2.26 – Згорткова нейронна мережа

Зображення в комп'ютері представляються у вигляді пікселів, а кожен піксель – це значення інтенсивності відповідних каналів (описується цілим числом від 0 до 255). Найчастіше використовуються кольорові зображення, що складаються з RGB пікселів, які містять яскравості по трьом каналам: червоному, зеленому і синьому. Різні комбінації цих кольорів дозволяють створити будь-який із кольорів всього спектра. Тому тензори використовуються для представлення зображень: кожна матриця тензора відповідає за інтенсивність свого каналу, а сукупність усіх матриць описує все зображення.

У нейронної мережі є кілька різних типів шарів:

- згортковий шар,
- шар підвибірки,
- шар активації,
- повнозв'язний шар.

Згортковий шар

Основне призначення – виділити ознаки на вхідному зображенні і сформувати карту ознак [34, 35].

Карта ознак – це масив матриць, у якому кожен канал відповідає за яку-небудь виділену ознаку.

Згортка – це операція обчислення нового значення обраного пікселя, що враховує значення оточуючих його пікселів [36, 37] (рис. 2.27…2.29).

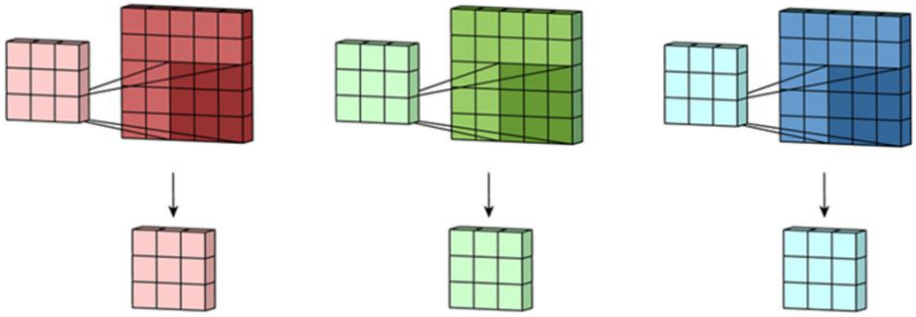


Рисунок 2.27 – Покомпонентне множення значень фільтра і значень зображення

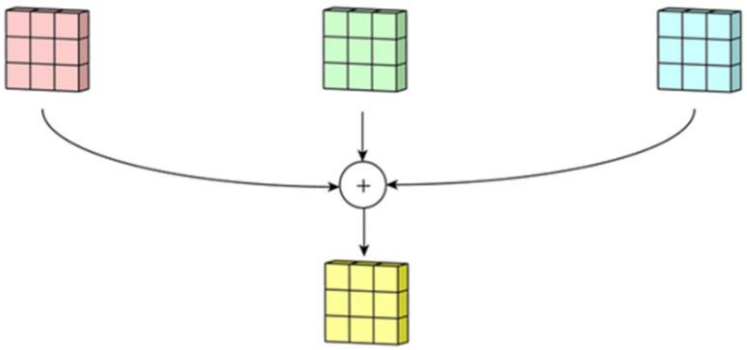


Рисунок 2.28 – Числа отриманих матриць підсумовуються в єдину матрицю

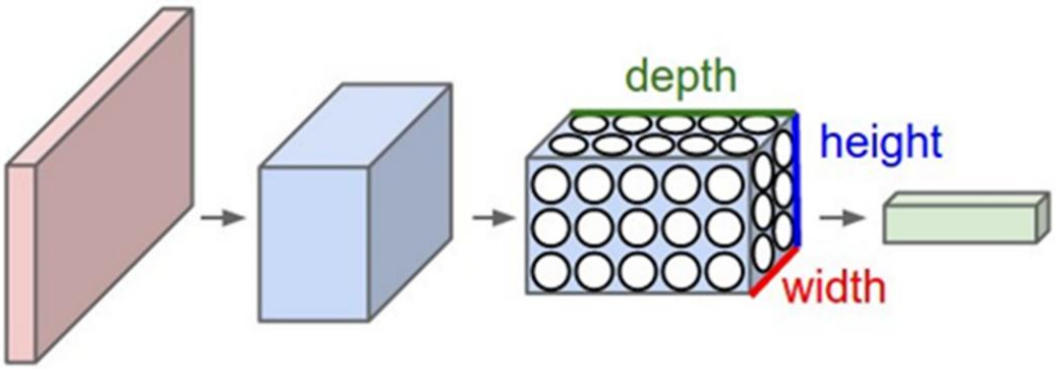


Рисунок 2.29 – Розташування нейронів у трьох вимірах – ширині, висоті,

глибині

Вхідними параметрами згорткового шару є :

- тензор розміром $W_1 x H_1 x D_1$ і
- 4 гіперпараметри: fc, fs, S, P,

де fc – кількість фільтрів, які є в шарі;

fs – розмір фільтрів (висота і ширина тензора фільтрів);

S – крок згортки, кількість пікселів, на яке переміщується матриця фільтру по вхідному зображенню;

P – доповнення нулями – кількість пікселів, які додаються з кожного краю зображення, щоб уникнути зменшення зображення на розмір фільтра.

Вихідним параметром шару є тензор розміром

$$W_2 x H_2 x D_2,$$

де $W_2 = (W_1 - fs + 2P) / S + 1,$

$H_2 = (H_1 - fs + 2P) / S + 1,$

$D_2 = fc.$

Шар підвибірки (рис. 2.30)

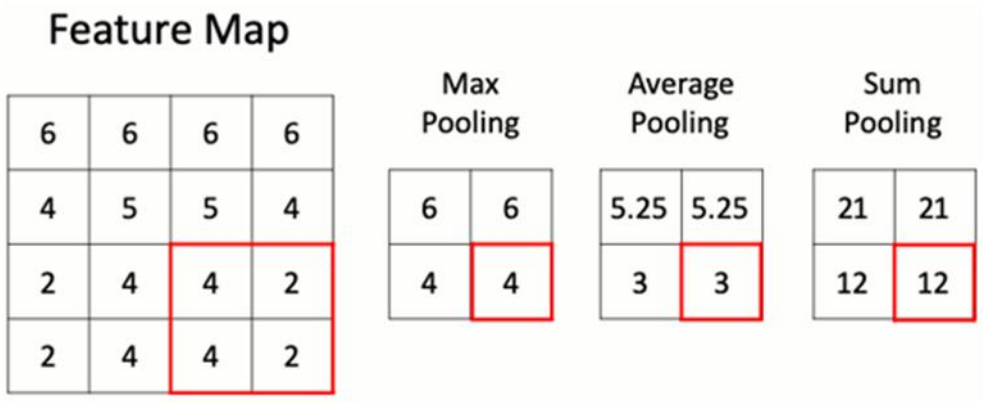


Рисунок 2.30 – Різні версії шару пулінгу

Шар підвибірки має один гіперпараметр – *крок пулінгу,* тобто число разів, у яке потрібно скоротити просторові розмірності.

Шар активації. Повнозв'язний шар

Шар активації. Цей шар використовує деяку функцію, яка застосовується до кожного числа вхідного зображення. Найбільш часто використовуються такі функції активації, як ReLU, Sigmoid, Tanh, LeakyReLU.

Повнозв'язний шар. Нейрони в повнозв'язному шарі зв'язані з усіма активаційними нейронами попереднього шару, як у звичайній нейронній мережі. Містить матрицю вагових коефіцієнтів і вектор зсувів.

Єдиним гіперпараметром шару є кількість вихідних значень. Результатом застосування шару є вектор або тензор, у якого матриці в кожному каналі мають розмір 1x1.

2.5.2 Навчання згорткової НМ

У процесі навчання використовується метод зворотного поширення помилки. При цьому оновити параметри вихідного шару досить просто, але щоб дістатися до параметрів шарів за ним, для згорткової мережі доводиться проходити через нелінійності, похідні і т. д. Формули для поновлення ваг в середині мережі виглядають складно.

Проблеми, що виникають:

- зависання в локальних мінімумах, або сідлових точках, яких для функції від великого числа змінних може бути дуже багато;
- складний ландшафт цільової функції: плато чергуються з регіонами сильної нелінійності;

• деякі параметри оновлюються значно рідше інших, особливо коли в даних зустрічаються інформативні, але рідкісні ознаки, що погано позначається на узагальнюючому правилі мережі;

• занадто мала швидкість навчання змушує алгоритм сходитися дуже довго і зависати в локальних мінімумах, занадто велика – «пролітати» вузькі глобальні мінімуми або зовсім розходитися

Для згорткових мереж використовується оптимізаційний алгоритм Adam (adaptive moment estimation). Він уміє звертатися до історії змін кожного параметра, поєднує в собі ідею накопичення руху та ідею слабшого поновлення ваг для типових ознак.

Швидкість навчання для кожного параметра адаптується на основі середнього першого моменту, також використовує середнє значення других моментів градієнтів, тобто стабілізує градієнт усередненням на декількох точках вздовж прямої, по якій ми рухаємося.

Коли є передбачення мережі, можна знайти помилку. Часто використовують бінарну крос-ентропію (метод логарифмічної правдоподібності). Ця функція вартості помилки L опукла, тому досягти глобального мінімуму буде досить легко:

$$L = -\frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)}\, log\left(a^{[L](i)}\right) + \left(1 - y^{(i)}\right) log\left(1 - a^{[L](i)}\right),$$

де n – кількість тренувальних прикладів;

a – вектор ймовірності відповідних міток передбачень;

y – вектор вірних міток (наприклад, 0 для «немає хвороби» і 1 для «є хвороба»);

L – функція вартості помилки мережі.

Крос-ентропія має справу з розподілами ймовірностей P і Q, визначаючи P, як справжнє, а Q – як наближене, і вимірює «загальну ентропію» між цими розподілами.

Крос-ентропія вимірює середнє число біт, необхідне для кодування даних, отриманих із джерела, що використовує розподіл ймовірностей P, якщо використовувана схема кодування базується на заданому розподілі ймовірностей Q. Тобто ми прагнемо зменшити число біт, необхідне для позначення події, використовуючи Q замість P.

VGG16 (Visual Geometry Group) (рис. 2.31) має 16 фільтрів розміром 3x3, розташованих один за одним із нульовим доповненням (P = 0) і одиничним кроком (S = 1) [38].

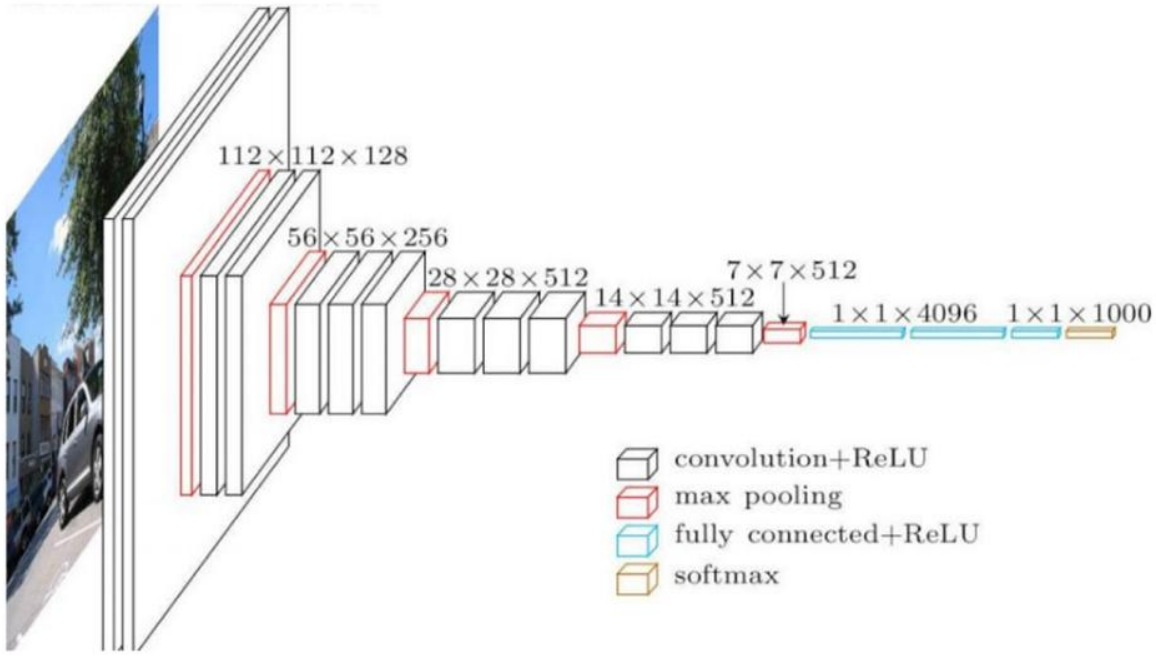


Рисунок 2.31 – Мережа VGG16

Під час навчання вхід являє собою рентгенівський знімок фіксованого розміру 224×224 px. На наступному кроці зображення пропускається через стопку згорткових шарів розміром 3×3.

Крок згортки фіксується на значенні 1 піксель. Просторове доповнення (padding) входу згорткового шару дорівнює 1 для 3x3 згорткових шарів.

Великою перевагою VGGNet стала знахідка, що кілька згорток 3x3, об'єднаних у послідовність, можуть емулювати більші рецептивні поля, наприклад, 5x5 або 7x7.

Мережа DenseNet. Мережа розділена на декілька щільно з'єднаних (dense) блоків, між якими розташовані перехідні шари. Ці шари складаються із шару нормалізації партії та згорткового шару 1×1, за яким слідує шар об’єднання 2×2 [39].

Кожен шар блока явно пов'язаний з усіма попередніми шарами в області об'єднання. Ці підключення сприяють повторному використанню функцій, тому що функції раннього шару можуть бути використані всіма іншими шарами (рис. 2.32).

Ознаки («фічі»), перш ніж вони будуть передані в наступний шар, не підсумовуються, а конкатенуються в єдиний тензор.

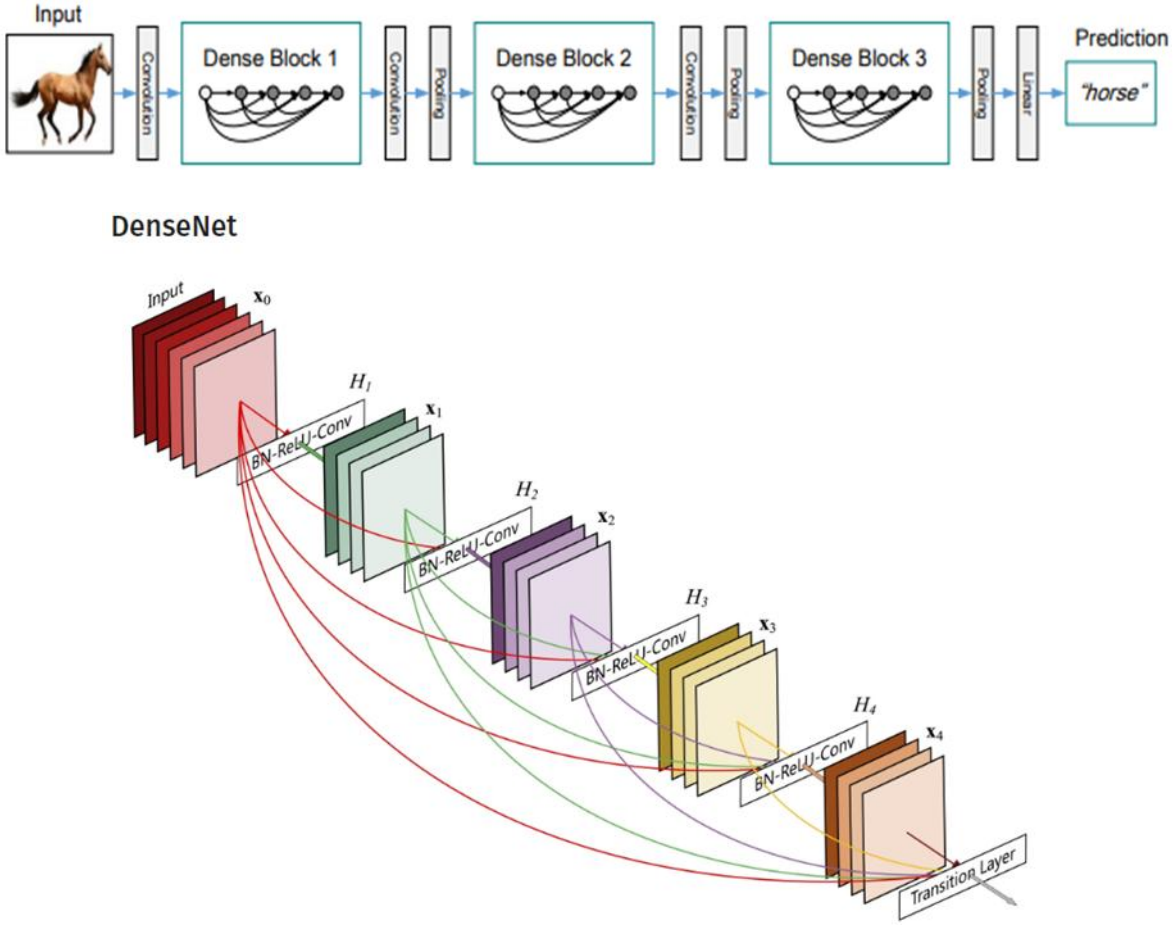


Рисунок 2.32 – Мережа DenseNet

Оскільки риси накопичуються, остаточний класифікаційний рівень має доступ до великого і різноманітного представлення ознак.

Кожен шар створює лише k карт функцій (де k зазвичай від 12 до 48), але використовує всі попередні карти функцій як вхідні дані. Це призводить до того, що кількість параметрів зростає в квадратичному масштабі.

Проміжні карти функцій є результатами операцій нормалізації та конкатенації пакетів. Ці карти функцій використовуються як під час прямого проходу, так і під час зворотного поширення (для обчислення градієнтів).

Реалізація включає спільні сховища для кожної з наступних стадій: пакетної нормалізації, конкатенації, градієнтних операцій. Ця розподілена спільна пам’ять використовується усіма шарами, щоб зберігати проміжні результати. Наступні шари замінюють проміжні результати попередніх шарів, але їх значення можна повторно заповнити під час зворотного проходу з мінімальними витратами.

Тобто об'єднані та нормалізовані карти об'єктів перераховуються за необхідності протягом зворотного розповсюдження помилки. Єдиними тензорами, які зберігаються в пам'яті під час навчання, є згортка особливостей карт та параметрів мережі.

2.6 Виявлення та відстеження об'єктів. Застосування детекторів і дескрипторів для розв’язання задачі класифікації

Детектор — алгоритм пошуку особливих точок [40, 41].

Дескриптор — опис особливої точки, що визначає особливості її околу, являє собою числовий або бінарний вектор певних параметрів. Довжина вектора і вид параметрів визначаються застосовуваним алгоритмом [40, 41].

Дескриптор дозволяє виділити особливу точку з усієї їх множини на зображенні. Це необхідно для складання ключових пар особливостей, що належать одному об’єкту при порівнянні різних зображень. На вхід дескрипторам подається зображення і набір особливих точок, а виходом є

набір векторів ознак для кожної особливої точки. Проте деякі алгоритми розв'язують одразу дві задачі: пошук особливих точок та побудову їх дескрипторів.

Одними з найпоширеніших класів алгоритмів класифікації є так звані bag-of-words (або bag-of-features, bag-of-key-points). Ідея запозичена із задачі класифікації текстів, де використовується опис у вигляді гістограм входжень певних слів із наперед складеного словника.

Алгоритм застосування детекторів і дескрипторів:

- виявлення ключових точок зображення;
- обчислення дескрипторів локальних околів особливих точок;
- кластеризація дескрипторів ключових точок, що належать всім об'єктам навчальної вибірки;
- побудова опису кожного зображення у вигляді нормованої гістограми входжень «слів» (для кожного кластеру обчислюється кількість віднесених до нього ключових точок певного зображення);
- побудова класифікатора, який використовує обчислений на попередньому кроці опис зображення.

Такий підхід отримав достатньо критики, оскільки він ніяк не враховує просторову інформацію про розподіл ключових точок зображення.

Для врахування цього аспекту запропоновано декілька підходів:

- використання так званих просторових корелограм «візуальних слів»;
- використання ієрархічної моделі, у якій об'єкт представляється P частинами, до кожної з яких відноситься Np ключових точок;
- використання схеми порівняння пірамід, у якій простір ознак розбивається на послідовність вкладених одна в одну підобластей і обчислюється зважена сума числа співпадінь на всіх рівнях розбиття, причому співпадіння на більш детальних рівнях мають більшу вагу;
- використання моделей об'єктів, що складаються з частин (part-based models).

Методи розв’язання задач виявлення об’єктів можна умовно поділити на 3 групи:

1. Методи, що використовують найбільш характерні ознаки для опису об’єктів. У якості ознак можуть бути вибрані точкові особливості об’єкта або ознаки, побудовані для зображення, що містить тільки один цей об’єкт.

2. Методи пошуку об’єктів за шаблоном.

3. Методи виявлення об’єктів, що рухаються, на базі декількох зображень або кадрів відео однієї й тої ж сцени.

1. Методи на основі характерних ознак

Спочатку будують характерні вектори ознак для особливих точок об’єкта або для всього об’єкта, а потім на основі цього будують класифікатор (можуть використовуватися також методи машинного навчання).

Оскільки об’єктів на зображенні може бути досить багато і вони можуть бути представлені в різних масштабах, то використовують техніку «ковзаючої рамки» різного розміру.

Детектори і дескриптори використовуються для виявлення та описання особливих точок. Ця інформація потім подається на вхід класифікатора, який відносить об’єкт до певного класу.

Пошук відповідного дескриптора — це пошук найближчого сусіда в просторі дескрипторів. Бібліотека FLANN (Fast Library for Approximate Nearest Neighbors) містить колекцію алгоритмів для пошуку найближчих сусідів у просторах великої розмірності.

Методи бібліотеки автоматично вибирають найкращий алгоритм пошуку й оптимальні параметри, базуючись на вхідних даних.

2. Методи пошуку за шаблоном

У якості шаблону можуть використовувати зображення, на якому присутній тільки шуканий об’єкт, або дескриптори, які притаманні цьому об’єкту.

Для розв’язання задачі, як і в інших методах, застосовується техніка «ковзаючої рамки» різних масштабів. Частина зображення, що потрапляє в рамку, співставляється із шаблоном. Результатом такого порівняння є міра схожості, у якості якої може бути вибрана, наприклад, величина, обернена до евклідової відстані.

3. Методи виявлення об’єктів, що рухаються

Алгоритм робастного оцінювання RANSAC (RANdom Sample Consensus). Під робастністю слід розуміти нечутливість до малих відхилень від припущень.

Він використовується для оцінки параметрів моделі на підставі випадкових вибірок. При зіставленні модель являє собою матрицю перетворення (гомографія). На вході алгоритму є дві множини дескрипторів, отриманих на попередньому і поточному зображенні.

Робота алгоритму полягає в багаторазовому повторенні таких етапів:

• вибір опорних точок і побудова параметрів моделі, що містить 8 невідомих параметрів. Для пошуку матриці гомографії необхідно, як мінімум, 4 пари особливих точок на зображеннях, що зіставляються. На підставі отриманих наборів будується матриця перетворення;

• перевірка побудованої моделі. Для кожної точки попереднього кадру знаходиться проєкція на поточному кадрі і виконується пошук найбільш близького дескриптора (точки) із множини дескрипторів поточного кадру. Характерна точка позначається як «викид», якщо відстань між проєкцією і відповідним дескриптором поточного зображення більше заданого порогу;

• заміщення моделі. Перевіряється, чи є побудована модель кращою серед набору попередніх моделей.

Дескриптори. Метод SURF. На вхід дескрипторам подається зображення і набір особливих точок, а виходом є набір векторів ознак для кожної особливої точки. Проте деякі алгоритми розв’язують одразу дві задачі: пошук особливих точок та побудову їх дескрипторів.

SURF (Speeded up Robust Features) шукає особливі точки за допомогою матриці Гессе. Гессіан досягає екстремуму в точках максимальної зміни градієнту яскравості. Алгоритм добре виявляє плями, кути, краї. Оскільки гессіан не інваріантний відносно масштабу, то SURF використовує різномасштабні фільтри для знаходження гессіанів. Для кожної ключової точки обчислюється напрямок максимальної зміни яскравості і масштаб, взятий із масштабного коефіцієнта матриці Гессе. Градієнт обчислюється за допомогою фільтрів (каскадів) Хаара.

Приклади дескрипторів. SIFT (Scale Invariant Feature Transform). Обчислюється різниця Гауса — попіксельне віднімання зображень.

Модифікації PCA-SIFT (PCA — Principal Component Analysis) та GLOH (Gradient Location and Orientation Histogram).

BRIEF (Binary Robust Independent Elementary Features). Алгоритм зводиться до побудови випадкового лісу або наївного Баєсівського класифікатора на тренувальній множині.

BRISK використовує шаблон шляхом аналізу точок, що розташовані рівномірно розподілено по колам, концентричним із ключовою точкою. Це забезпечує інтегрований аналіз та високу швидкість оброблення чи зберігання.

ORB (Oriented FAST and Rotated BRIEF). ORB є свого роду сплавом FAST-детектора та дескриптора BRIEF. За допомогою детектора FAST знаходяться особливі точки, після чого застосовується міра Харріса для визначення найкращих N точок. Алгоритм використовує пірамідальну структуру для виявлення особливостей різних масштабів. Для обчислення дескриптора використовується BRIEF.

KAZE. Ідея полягала у виявленні та описі 2D-особливості в нелінійних екстремумах масштабного простору, щоб отримати кращу точність локалізації. Оскільки KAZE обчислює багатомасштабні похідні (градієнти) для кожного пікселя, алгоритм більш вимогливий до продуктивності, ніж SURF.

Виявлення та відстеження об'єктів

Обчислення різниці між кадрами – найпростіша методика, яку можна використовувати для ідентифікації рухомих елементів відео. У процесі перегляду відеопотоку ми отримуємо масу інформації на основі різниці зображень між послідовними кадрами. Інформація, яку надає різниця між кадрами, корисна, але на її основі не можна побудувати надійний трекер. Вона дуже чутлива до шуму.

Відстеження об'єктів за допомогою колірних просторів. Зображення може бути представлене з використанням різних колірних просторів. Найбільш популярним із них є RGB, але колірний простір HSV точніше відповідає сприйняттю кольору людиною. Ми можемо перетворити захоплені кадри з простору RGB у простір HSV, а потім використовувати порогові значення кольорів (колірний діапазон HSV для кольору людської шкіри) для відстеження будь-якого заданого об'єкта. Для вибору відповідних діапазонів кольорів у процесі призначення порогових значень необхідно знати розподіл кольорів об'єкта.

Відстеження об'єктів шляхом віднімання фонових зображень (рис. 2.33). Метод [42, 43] відмінно справляється зі своїми завданнями в тих випадках, коли потрібно виявляти рухомі об'єкти в межах статичної сцени. Робота даного алгоритму в основному полягає у виявленні фонового зображення, побудові моделі для нього і подальшому відніманні фону з поточного кадру для отримання переднього плану. Цей передній план і відповідає рухомим об'єктам.

Рисунок 2.33 – Приклад техніки віднімання фону

Поняття оптичного потоку. В основі поняття оптичного потоку лежить припущення про те, що при зміні положення конкретного пікселя від кадру до кадру його яскравість і інтенсивність не змінюються, а найближчі точки, що належать одному об'єкту, в площині зображення рухаються з однаковою швидкістю [44].

Із кожним пікселем зображення пов'язаний деякий вектор швидкості, що визначає, яку відстань пройшов піксель протягом часового проміжку між попереднім і поточним кадрами (рис. 2.34) На основі наступного рівняння будується вектор швидкості пікселя:

$$\left(\frac{\Delta x}{\Delta t}, \ \frac{\Delta y}{\Delta t}\right) = (u, v) \ ,$$

де u, v – координати вектора швидкості.

Рівняння містить дві невідомі і являє собою оптичний потік.

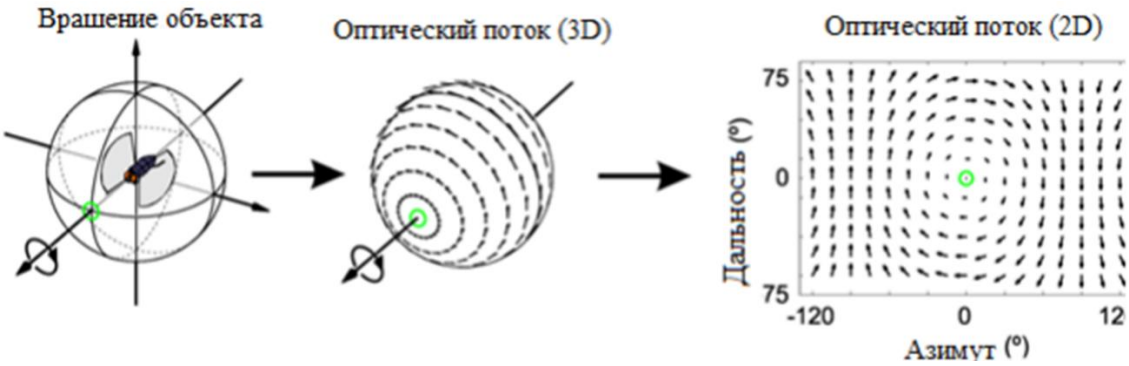


Рисунок 2.34 – Проєкція руху на двомірне поле

Алгоритм Лукаса – Канаде. Алгоритм обходить неоднозначність за рахунок використання інформації про сусідні пікселі в кожній точці. Згідно з припущенням, що в локальному околі (x, y) кожного пікселя p значення оптичного потоку однакові, то отримуємо систему із N рівнянь:

$$\begin{cases} \nabla I\,(u, v)\,(p_1) = -I_t(p_1) \\ \nabla I\,(u, v)\,(p_2) = -I_t(p_2) \\ \dots \\ \nabla I\,(u, v)\,(p_N) = -I_t(p_N) \end{cases}$$

Запишемо систему у матричному вигляді:

$$Ad = b,$$

де A – матриця градієнта для всіх пікселів, d – вектор зміщення, b – вектор змінення кольору для всіх сусідніх точок.

Відстеження об'єктів із використанням оптичних потоків. Перший крок полягає в отриманні особливих точок із поточного кадру. Для кожної точки створюється фрагмент розміром 3x3 пікселя з особливою точкою в центрі. Ми припускаємо, що всі точки в кожному фрагменті рухаються однаково. Розмір цього вікна можна регулювати в залежності від ситуації.

Для кожного фрагмента шукається відповідність у його околі в попередньому кадрі. Найкраще відповідність вибирається на підставі метрики помилки. Як тільки ми знаходимо найближчий аналогічний фрагмент, шлях між центральною точкою поточного фрагмента і відповідним йому фрагментом попереднього кадру стає вектором руху (рис.

2.35). Аналогічним чином ми обчислюємо вектори руху для всіх інших фрагментів.

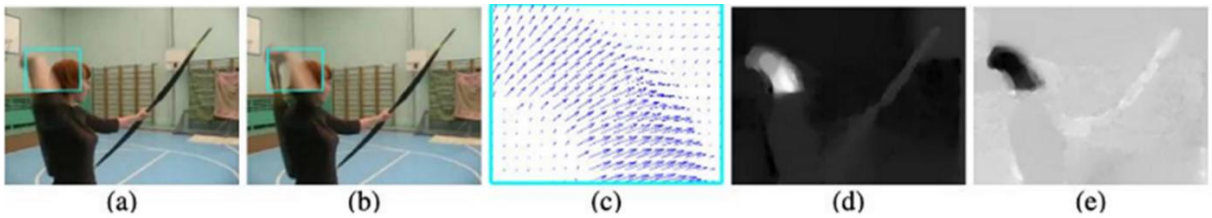


Рисунок 2.35 – Обчислення вектора руху для фрагментів

(a), (b): пара послідовних відеокадрів із площею навколо рухомої руки, окресленої блакитним прямокутником; (c): крупним планом густого оптичного потоку в окресленій області; (d): горизонтальний компонент дорівнює вектору переміщення (більша інтенсивність відповідає позитивним значенням, менша інтенсивність – до негативних значень); (e): вертикальний компонент.

Виявлення та відстеження обличь у живому потоці. Виявлення обличь – це визначення місця розташування особи на зображенні. Розпізнавання обличь – процес ідентифікації особистості.

Розглянемо автоматизацію виявлення та відстеження обличь у живому потоці з використанням каскадів Хаара. У цьому випадку каскади Хаара – це каскадні класифікатори, засновані на ознаках Хаара.

Це просто суми і різниці фрагментів зображення, і їх обчислення фактично не представляє труднощів. Щоб метод залишався надійним щодо масштабування, повторюємо всю процедуру для зображень різного розміру.

Як тільки ознаки витягнуті, пропускаємо їх через наш каскад простих класифікаторів. Перевіряємо різні прямокутні підобласті зображення і відкидаємо ті, які не містять зображень обличчя. Це дозволяє швидко отримати відповідь. Обчислення ознак вдається прискорити за рахунок використання так званих інтегральних зображень.

Використання інтегральних зображень. Якщо нам необхідно підсумувати пікселі в прямокутній області ABCD зображення, то зовсім

необов'язково проходити по всім пікселям, що належать даному прямокутнику (рис. 2.36).

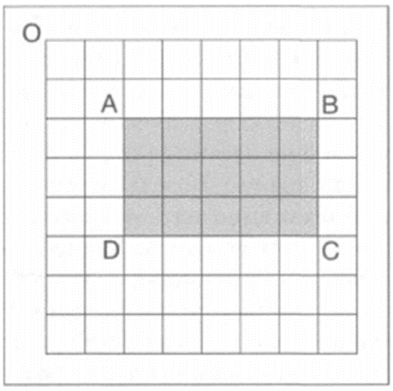


Рисунок 2.36 – Розрахунок площі прямокутної області

Нехай OP позначає область прямокутника, визначеного за допомогою його лівого верхнього кута O і точки P, розташованої на іншому кінці діагоналі. Тоді для розрахунку площі прямокутної області ABCD можна скористатися наступною формулою:

$$\text{Площа ABCD} = OC - (OB + OD - OA).$$

2.7 Короткі теоретичні відомості для виконання практичних робіт

1. *Google TensorFlow* – це відкрита і найпопулярніша бібліотека глибокого навчання для досліджень і виробництва.

Зручний спосіб опису об'єкта в реальному світі – це перерахування його властивостей або функцій. Наприклад, ви можете описати машину за кольором, моделлю, типом двигуна, пробігом. Впорядкований список функцій називається вектором функцій, і це саме те, що ви представляєте в TensorFlow-коді.

Вектор векторів однакової довжини – це синтаксис для представлення матриць у TensorFlow. Ми отримуємо доступ до елементу в матриці,

указуючи його індекси рядка і стовпця. Наприклад, перший рядок і перший стовпець указують найперший верхній лівий елемент (рис. 2.37).

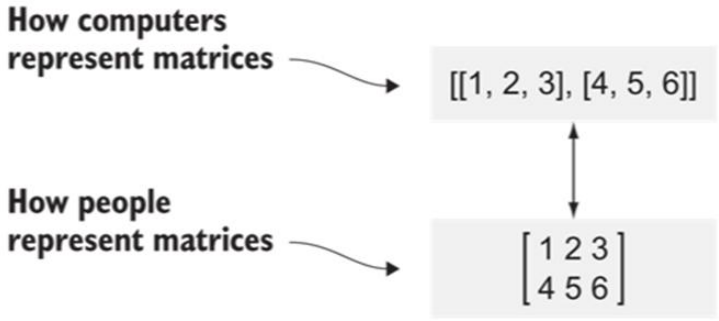


Рисунок 2.37 – Використання двох індексів при доступі до елементу матриці

Іноді зручно використовувати більше двох індексів, наприклад, при посиланні на піксель у кольоровому зображенні не тільки за його рядком і стовпцем, а і за його червоним (зеленим, синім) каналом. Тензор є узагальненням матриці, яка визначає елемент на довільне число індексів (рис. 2.38). Тензор можна представити як кілька матриць, накладених одна на одну. Щоб указати елемент, ви повинні вказати рядок і стовпець, а також до якої матриці здійснюється доступ. Отже, ранг цього тензора дорівнює 3 (див. рис. 2.38).

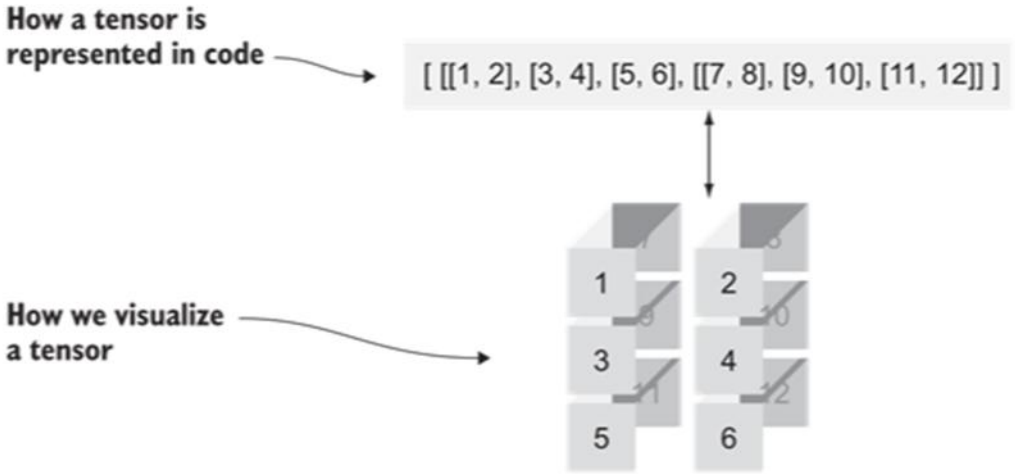


Рисунок 2.38 – Використання трьох індексів при доступі до елементу матриці

Наступна особливість TensorFlow – розуміння коду як графіка (рис. 2.39).

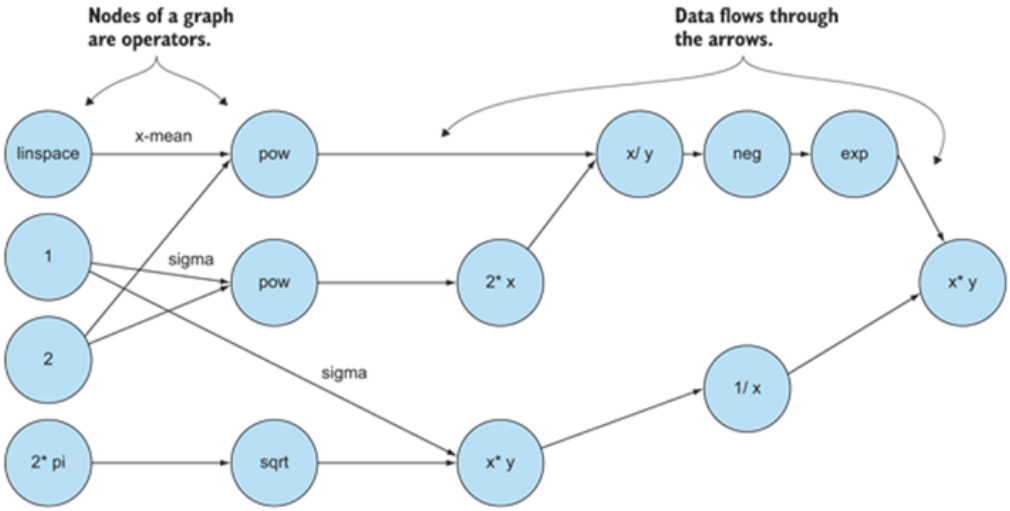


Рисунок 2.39 – Операції необхідні для побудови розподілу Гаусса

Операції представляються вузлами. Зв'язки між ними показують, як дані передаються від однієї операції до іншої. Сеанс визначає, як обладнання буде використовуватися для найбільш ефективної обробки графіка. Після обробки сеанс виводить дані в зручному форматі, такому, як масив NumPy. У сеансі необов'язково можуть бути заповнювачі, змінні і константи.

Найбільш наочне уявлення про компоненти TensorFlow ми отримуємо, розглядаючи його архітектуру (рис. 2.40).

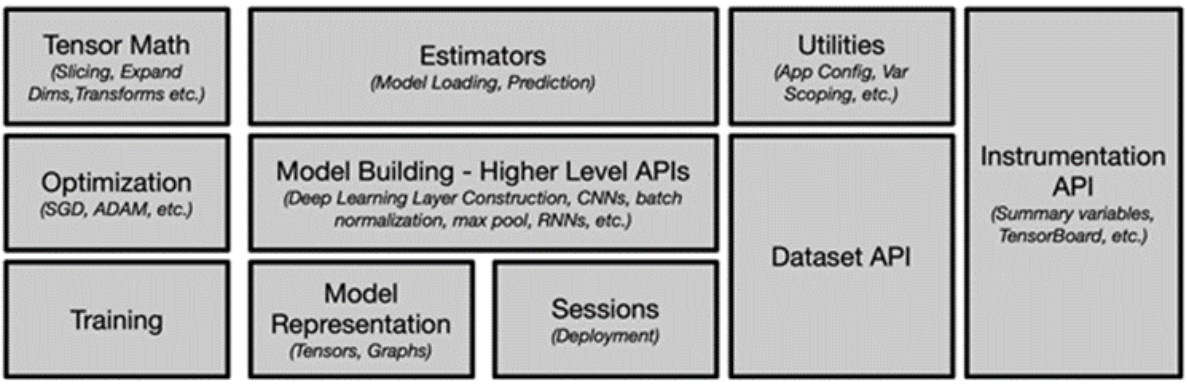


Рисунок 2.40 – Архітектура системи TensorFlow і API

Безсумнівно, TensorFlow – це не тільки уявлення обчислень у вигляді графіка. Як ви побачите в наступних розділах, деякі вбудовані функції адаптовані для області машинного навчання. Фактично, TensorFlow має кращу підтримку згортальних нейронних мереж (CNN), популярного в даний час типу моделі для обробки зображень (із багатообіцяючими результатами в аудіо і тексті).

TensorBoard надає простий спосіб візуалізувати спосіб зміни даних у коді TensorFlow, а також усувати помилки шляхом вивчення тенденцій у даних.

TensorFlow прекрасно працює із записниками Jupyter, які представляють собою інтерактивну середу для обміну та документування коду Python.

2. *Бібліотека NumPy* – бібліотека з відкритим вихідним кодом для мови програмування Python. Має наступні можливості:

– підтримка багатовимірних масивів (включаючи матриці);

– підтримка високорівневих математичних функцій, призначених для роботи з багатовимірними масивами.

Розглянемо, наприклад, обробку зображень в NumPy. Зображення є матрицею пікселів за розміром (висота х ширина). Якщо зображення чорно-біле, тобто представлене в напівтонах, кожен піксель може бути представлений як єдине число. Зазвичай між 0 (чорний) і 255 (білий).

Якщо необхідно обрізати квадрат розміром 10х10 пікселів у верхньому лівому кутку картинки, то набираємо *image [: 10,: 10]*. Якщо зображення кольорове, кожен піксель представлений трьома числами. Тут за основу береться колірна модель RGB – червоний (R), зелений (G) і синій (B). У даному випадку нам знадобиться третя розмірність, оскільки кожна клітина вміщає тільки одне число. Таким чином, кольорова картинка буде представлена масивом *ndarray* з розмірностями: (висота х ширина х 3). Ось як виглядає фрагмент зображення (рис. 2.41).

Рисунок 2.41 – представлення кольорової картинки масивом

3. *Бібліотека SciPy* є відкритим вихідним кодом для Python, поширюваним у рамках ліцензованої бібліотеки BSD для виконання математичних, наукових та інженерних обчислень.

SciPy залежить від NumPy, який забезпечує зручну і швидку маніпуляцію з N-мірним масивом. Бібліотека SciPy створена для роботи з масивами NumPy і надає безліч зручних і ефективних чисельних методів, таких як процедури чисельної інтеграції та оптимізації.

SciPy складається з підпакетів, що охоплюють різні галузі наукових обчислень. Вони узагальнені в таблиці 2.1.

4. *Бібліотека OpenCV* (open source computer vision library) – це бібліотека комп'ютерного зору з відкритим вихідним кодом. Включає в себе різні алгоритми комп'ютерного зору, розпізнавання зображень і багато іншого, що працює в реальному режимі часу.

Функціональність OpenCV забезпечується наступними складовими:

1) ядро *Cxcore*, також частково використовується в Intel Open Source Probabilistic Network Library:

– базові операції над багатовимірними числовими масивами;

– матрична алгебра, математичні функції, генератори випадкових чисел;

– DFT, DCT;

– запис/відновлення структур даних в/з XML / YAML;

– базові функції 2D-графіки;

– підтримка складніших структур даних: розріджені масиви, динамічно зростаючі послідовності, графи;

Таблиця 2.1 – Sub-пакети SciPy

scipy.cluster	Векторне квантування / K-means
scipy.constants	Фізико-математичні константи
scipy.fftpack	Перетворення Фурье
scipy.integrate	Процедури інтеграції
scipy.interpolate	Інтерполяція
scipy.io	Введення і виведення даних
scipy.linalg	Процедури лінійної алгебри
scipy.ndimage	Пакет n-мірних зображень
scipy.odr	Ортогональна регресія відстані
scipy.optimize	Оптимізація
scipy.signal	Обробка сигналів
scipy.sparse	Розріджені матриці
scipy.spatial	Просторові структури даних і алгоритми
scipy.special	Будь-які спеціальні математичні функції
scipy.stats	Статистика

2) модуль обробки зображень і комп'ютерного зору CV:

– базові операції над зображеннями (фільтрація, геометричні перетворення, перетворення колірних просторів і т. д.);

– аналіз зображень (вибір відмінних ознак, морфологія, пошук контурів, гістограми);

– структурний аналіз (опис форм, плоскі розбиття і т. д.);

– аналіз руху, стеження за об'єктами;

– виявлення об'єктів, зокрема осіб;

– калібрування камер, елементи відновлення просторової структури;

3) модуль для введення/виведення зображень і відео, призначеного для користувача інтерфейсу highgui:

– захоплення відео з камер і з відео-файлів, читання/запис статичних зображень;

– функції для організації простого UI (зараз всі демо-додатки використовують HighGUI);

4) експериментальні та застарілі функції cvaux:

– просторовий зір: стерео-калібрації, саме калібрації;

– пошук стерео-відповідності, кліки в графах;

– знаходження й опис рис обличчя;

– порівняння форм, побудова скелетону;

– приховані марківскі ланцюги;

– опис текстур.

5. *Бібліотека Pillow* – дружній форк популярної бібліотеки PIL, Python Imaging Library.

Основні функції знаходяться в модулі *Image*. Ви можете створювати екземпляри цього класу декількома способами. Шляхом завантаження зображень з файлів, обробки інших зображень, або створення зображень із нуля.

Імпорт модуля Pillow, який ви хочете використовувати:

```
from PIL import Image
```

Потім ви отримаєте доступ до функцій:

```
myimage = Image.open (filename)
myimage.load ()
```

Використовуйте метод *open* ідентифікації файлу на комп'ютері, а потім завантажте ідентифікований файл за допомогою *myfile.load().* Як тільки зображення буде завантажено, із ним можна працювати. Часто

використовується блок *try except* при роботі з файлами. Щоб завантажити зображення за допомогою *try except*, використовуйте:

```
from PIL import Image, ImageFilter


try:
  original = Image.open(«Lenna.png»)
except FileNotFoundError:
  print(«Файл не знайдений»)
```

Модуль Pillow надає наступний набір визначених фільтрів для поліпшення зображення: *BLUR, DETAIL, EDGE_ENHANCE, EDGE_ENHANCE_MORE, EMBOSS, FIND_EDGES, SMOOTH, SMOOTH_MORE, SHARPEN, CONTOUR.*

6. *Бібліотека H5py* (ієрархічний формат даних) – назва формату файлів, розробленого для зберігання великого обсягу цифрової інформації. (бібліотеки для роботи з форматом і пов'язані з ним утиліти доступні для використання під вільною ліцензією).

Надає модель наборів даних і груп. Перші являють собою в основному масиви, а другі можна розглядати як каталоги. Збереження в цей файл значно зменшує його розміри.

7. *Бібліотека Keras* – відкрита нейромережева бібліотека, націлена на оперативну роботу з мережами глибинного навчання (написана на мові Python). Вона являє собою надбудову над фреймворками Deeplearning, TensorFlow і Theano.

Спроєктована так, щоб бути компактною, модульною та розширюваною. Вона була створена як частина дослідницьких зусиль проекту ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System), а її основним автором є і підтримує її Франсуа Шолль, інженер Google. Планувалося, що Google буде підтримувати Keras в основний бібліотеці TensorFlow, однак Шолль виділив Keras в окрему надбудову,

оскільки згідно з концепцією Keras є скоріше інтерфейсом, ніж наскрізною системою машинного навчання.

У Keras, ви збираєте шари (layers) для побудови моделей (models). Модель – це граф шарів. Найбільш поширеним видом моделі є стек шарів – модель tf.keras.Sequential (https://habr.com/ru/post/482126/). Доступно багато різновидів шарів tf.keras.layers. Більшість із них використовують загальний конструктор аргументів:

– activation: установка функції активації для шару. У цьому параметрі вказується ім'я вбудованої функції чи викликається об'єкт. У параметра немає значення за замовчуванням;

– kernel_initializer і bias_initializer: схеми ініціалізації створюють ваги шару (ядро і зрушення). У цьому параметрі може бути ім'я або викликається об'єкт. За замовчуванням використовується ініціалізатор «Glorot uniform»;

– kernel_regularizer і bias_regularizer: схеми регуляризації додаються до ваг шару (ядро і зрушення), такі як L1 або L2 регуляризації. За замовчуванням регуляризація не встановлюється.

Використовуйте Keras functional API для побудови складних топологій моделей, таких як:

– моделі з декількома входами;

– моделі з декількома виходами;

– моделі із загальними шарами (один і той же шар викликається кілька разів);

– моделі з непослідовними потоками даних.

Побудова моделі з functional API працює наступним чином:

– примірник шару викликається і повертає тензор;

– вхідні і вихідні тензори використовуються для визначення екземпляру *tf.keras.Model*;

– ця модель навчається точно так само, як і Sequential-модель.

8. *Бібліотека ImageAI* – підтримує список сучасних алгоритмів машинного навчання для передбачення зображень, виявлення об'єктів, відстеження відеооб'єктів і навчання передбаченню зображень.

ImageAI підтримує прогнозування та навчання зображень із використанням 4 різних алгоритмів машинного навчання, навчених на наборі даних ImageNet 1000. ImageAI також підтримує виявлення об'єктів, виявлення відео і відстеження об'єктів із використанням RetinaNet, YOLOv3 і TinyYOLOv3, навчених на наборі даних COCO. ImageAI дозволяє навчати призначені для користувача моделі для виконання виявлення і розпізнавання нових об'єктів.

ImageAI буде забезпечувати підтримку більш широких і спеціалізованих аспектів комп'ютерного зору, включаючи розпізнавання зображень в особливих середовищах і спеціальних областях. ImageAI використовує магістраль Tensorflow для операцій комп'ютерного зору (офіційний репозиторій ImageAI на GitHub: https://github.com/OlafenwaMoses/ImageAI).

3 ПРАКТИЧНІ РОБОТИ

Зміст звіту з практичної роботи:

1) тема, мета роботи;

2) індивідуальне завдання;

3) хід роботи: опис, фрагменти коду, скріншоти результатів роботи;

4) висновки.

3.1 Практична робота 1

***Тема*. Основи аналізу та обробки зображень**

Мета роботи: здобути навички, роботи з алгоритмами перетворення кольорового зображення у чорно-біле. Ознайомитися з методами лінійного контрастування, інверсії, бінаризації зображення. Побудувати лінійну та кумулятивну гістограму зображення. Використовуючи алгоритми екстраполяції і інтерполяції, виконати двократне збільшення зображення.

Індивідуальне завдання

1. Самостійно вибрати 2 зображення різної якості.

2. Перетворити кольорове зображення в зображення у відтінках сірого.

3. Виконати лінійне контрастування, інверсію і бінаризацію зображення.

4. Побудувати лінійну і кумулятивну гістограму зображення.

5. Виконати дворазове збільшення зображення з використанням методів екстраполяції і інтерполяції.

6. Зробити висновки.

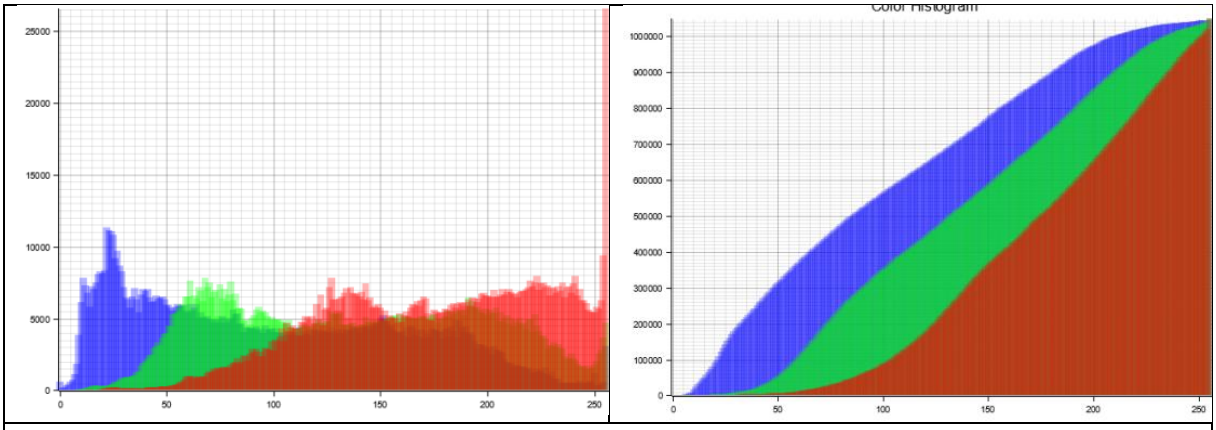
Приклади результатів роботи приведені на рис. 3.1.

Перетворення кольорового зображення в зображення у відтінках сірого

Лінійне контрастування зображення

Інверсія і бінаризація зображення

Рисунок 3.1 – Приклади результатів практичної роботи 1



Лінійна і кумулятивна гістограми зображення

Дворазове збільшення зображення

Рисунок 3.1, аркуш 2

Виконання цієї роботи дозволить глибше зрозуміти принципи обробки зображень і способи покращення візуальної якості зображень для різних завдань. Робота з методами лінійного контрастування допомагає виявити важливість регулювання контрастності зображення для підвищення чіткості та виділення деталей. Інверсія зображення, у свою чергу, дає змогу краще зрозуміти візуальні ефекти, яких можна досягти за допомогою цього

простого, але ефективного перетворення. Бінаризація зображення демонструє свою корисність у завданнях, де необхідно виділити певні елементи зображення або спростити його аналіз. Побудова лінійної та кумулятивної гістограми зображення є важливою частиною роботи, оскільки це дозволяє не лише візуалізувати розподіл інтенсивності пікселів у зображенні, але й краще зрозуміти структуру зображення та вплив різних методів обробки на цю структуру

3.2 Практична робота 2

***Тема*. Колірні моделі**

Мета роботи: Ознайомитися з різними колірними моделями зображення та алгоритмами їх перетворення.

Індивідуальне завдання

1. Самостійно вибрати зображення.
2. Перетворити колірну модель зображення з RGB в HSL, RGB в HSV, з RGB – у відтінки сірого.
3. Зробити висновки.

Приклади результатів роботи приведені на рис. 3.2.

Рисунок 3.2 – Приклади результатів практичної роботи 2

3.3 Практична робота 3

***Тема*. Оператори знаходження границь**

Мета: ознайомитися з різними алгоритмами та операторами пошуку границь.

Індивідуальне завдання

1. Самостійно вибрати зображення.

2. Виконати пошук границь зображення алгоритмами, вказаними у індивідуальному завданні (табл. 3.1).

3. Зробити висновки.

Таблиця 3.1 – Індивідуальні завдання

Варіант	Алгоритми обробки
1	Canny, Laplasian 3x3, Sobel 3x3
2	Canny, Prewitt, Kirsch
3	Canny, Roberts, Laplasian 5x5
4	Canny, Prewitt, Sobel 5x5
5	Canny, Laplasian 3x3, Kirsch
6	Canny, Prewitt, Sobel 3x3
7	Canny, Roberts, Kirsch
8	Canny, Prewitt, Laplasian 5x5
9	Canny, Kirsch, Sobel 5x5
10	Canny, Kirsch, Sobel 3x3
11	Canny, Roberts, Prewitt
12	Canny, Roberts, Sobel 3x3
13	Canny, Laplasian 3x3, Sobel 5x5
14	Canny, Roberts, Prewitt
15	Canny, Laplasian 5x5, Sobel 3x3

Приклади результатів роботи приведені на рис. 3.3.

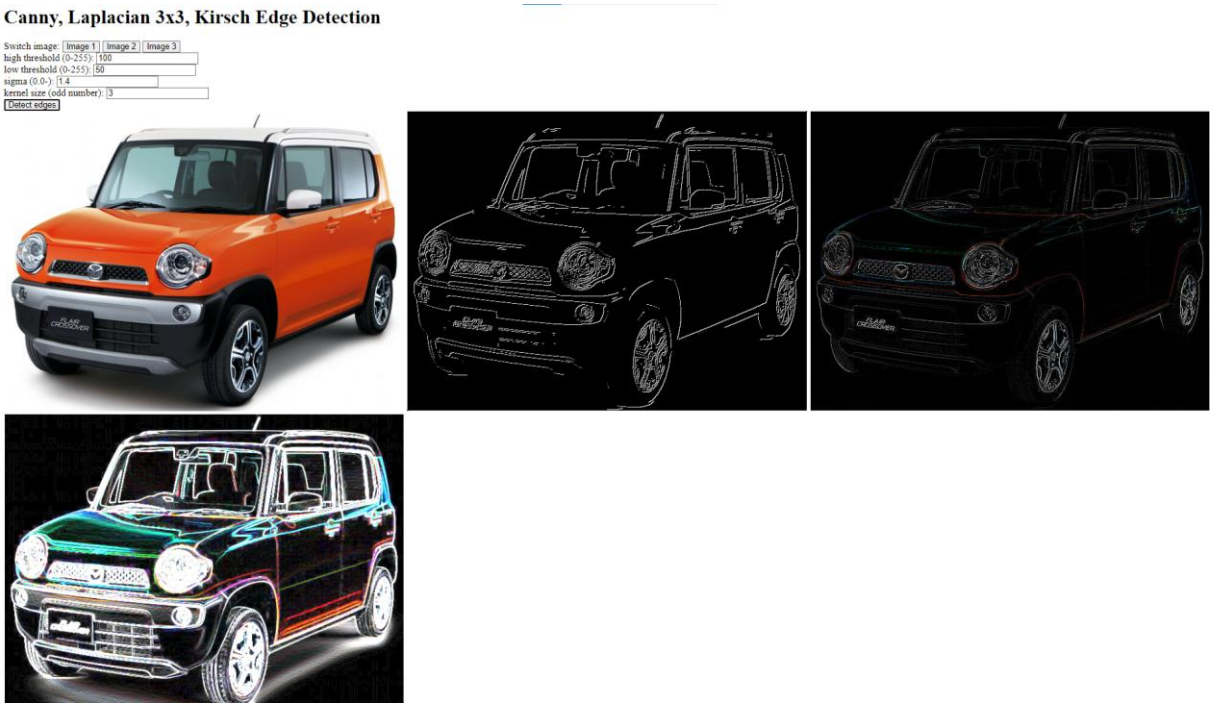


Рисунок 3.3 – Приклади результатів практичної роботи 3

3.4 Практична робота 4

Тема. **Застосування TENSORFLOW і бібліотек програмного забезпечення комп'ютерного зору**

Мета роботи: ознайомитися з використанням TensorFlow і бібліотек програмного забезпечення комп'ютерного зору.

Індивідуальне завдання

1. Самостійно вибрати вхідні дані.

2. Використати бібліотеки Google TensorFlow та Keras для відображення стеку слоїв моделі Sequential.

3. Використати бібліотеку SciPy для знаходження кластерів за допомогою k-means.

4. Використати бібліотеку OpenCV для копіювання частин зображення.

5. Використати бібліотеку Pillow для накладання фільтрів (контурний фільтр) для зображення.

6. Використати бібліотеку ImageAI для аналізу вхідного зображення.

7. Зробити висновки.

Програмне забезпечення:

1. PyCharm
2. Python
3. TensorFlow
4. Keras
5. SciPy
6. OpenCV
7. Pillow
8. ImageAI
9. Tourch
10. TourchVision
11. Scikit-image

1. Використання бібліотек Google TensorFlow та Keras для відображення стеку слоїв моделі Sequential.

Результат виконання коду див. на рис. 3.4.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 hidden_layer_1 (Dense)      (None, 32)                352

 dropout (Dropout)           (None, 32)                0

 hidden_layer_2 (Dense)      (None, 10)                330

=================================================================
Total params: 682
Trainable params: 682
Non-trainable params: 0
_________________________________________________________________
```

Рисунок 3.4 – Результат моделі Sequential

2. Використання бібліотеки SciPy для знаходження кластерів за допомогою k-means.

Результат виконання коду див. на рис. 3.5.

```
Нормалізація : [[1.20604538 3.61813613 1.29777137 3.16227766 2.41209076]
 [2.41209076 3.61813613 0.32444284 3.79473319 3.61813613]
 [1.20604538 6.03022689 0.64888568 1.8973666  1.20604538]
 [3.61813613 4.82418151 2.91998558 1.26491106 1.20604538]]
Обчислені центроїди : [[1.20604538 6.03022689 0.64888568 1.8973666  1.20604538]
 [1.80906807 3.61813613 0.81110711 3.47850543 3.01511345]
 [3.61813613 4.82418151 2.91998558 1.26491106 1.20604538]]
Кластерний індекс : [1 1 0 2]
```

Рисунок 3.5 – Виконання роботи зі знаходження кластерів

3. Використання бібліотеки OpenCV для копіювання частин зображення на прикладі зображення.

Результат виконання коду див. на рис. 3.6, 3.7.

```
Розмірність:  (512, 512, 3)
Загальна кількість пікселів:  786432
Тип даних:  uint8
```

Рисунок 3.6 – Інформація про вхідне зображення

Рисунок 3.7– Результат копіювання довільної області зображення

4. Використання бібліотеки Pillow для накладання фільтрів (контурний фільтр) для зображення.

Результат виконання коду див. на рис.3.8.

Рисунок 3.8 – Результат накладання контурного фільтру на зображення

5. Використання бібліотеки ImageAI.

Результат виконання коду див. на рис. 3.9, 3.10.

Рисунок 3.9 – Вхідне зображення для аналізу

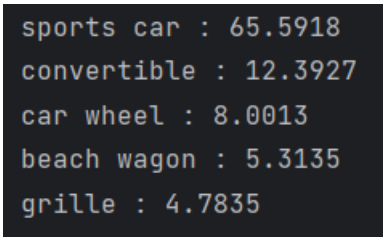


Рисунок 3.10 – Результат аналізу

3.5 Практична робота 5

***Тема*. Класифікація на основі навчання з учителем**

Мета роботи: створити наївний байесовский класифікатор і провести його тренування

Індивідуальне завдання

1. Створити новий файл Python і імпортувати необхідні пакети.
2. Створити наївний байесовский класифікатор
3. Завантажити вхідні дані (двовимірні) із вибраного файлу.
4. Навчити модель k-середніх на вхідних даних.
5. Відобразити два графіка: перший являє вхідні дані, а другий – отримані кордони кластерів.
6. Зробити висновки.

Програмне забезпечення:

1. PyCharm
2. Python
3. NumPy
4. Matplotlib
5. Scikit-learn

Результат виконання коду див. на рис. 3.11…3.14.

main.py data_multivar_nb.txt

```
2.18,0.57,0
4.13,5.12,1
9.87,1.95,2
4.02,-0.8,3
1.18,1.03,0
4.59,5.74,1
8.25,1.3,2
3.91,-0.68,3
0.55,1.26,0
5.64,6.67,1
9.22,1.46,2
4.36,-1.27,3
2.19,2.66,0
```

Рисунок 3.11 – Приклад синтаксису вхідних даних (фрагмент)

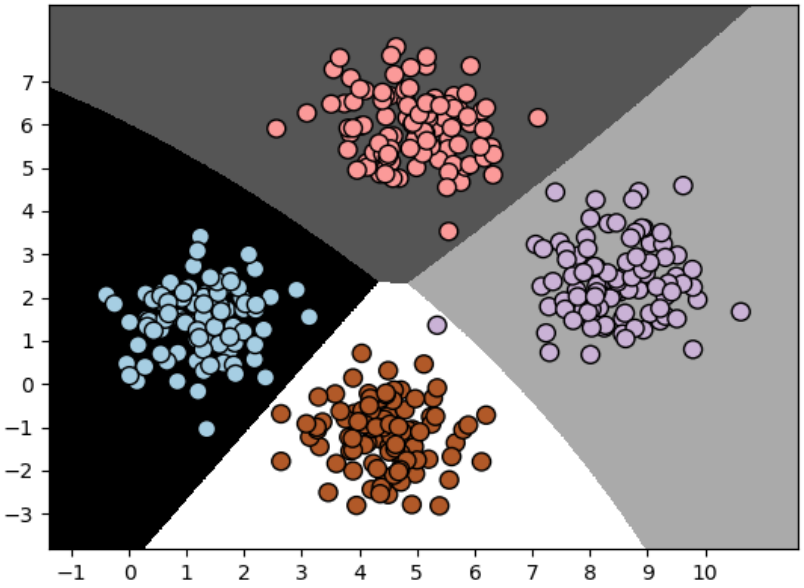


Рисунок 3.12 – Зображення тренувального прогону

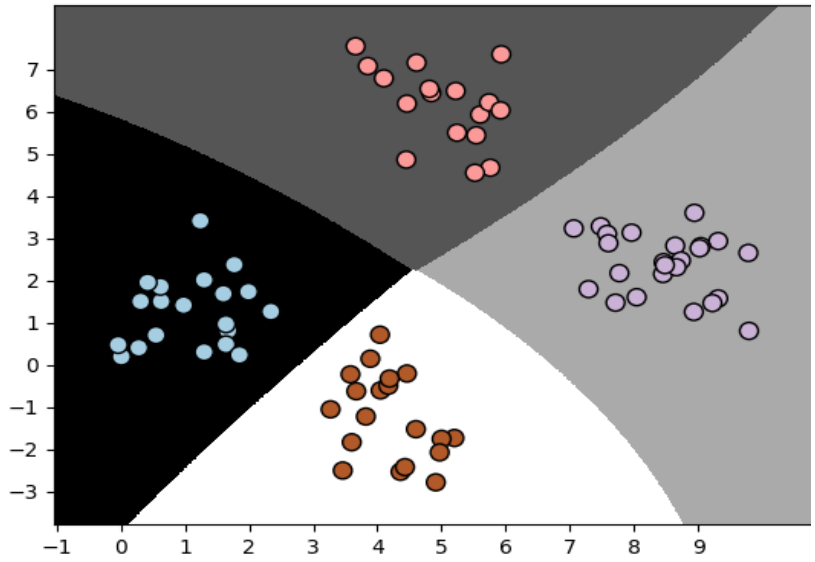


Рисунок 3.13 – Зображення тренувального прогону з перехресною перевіркою

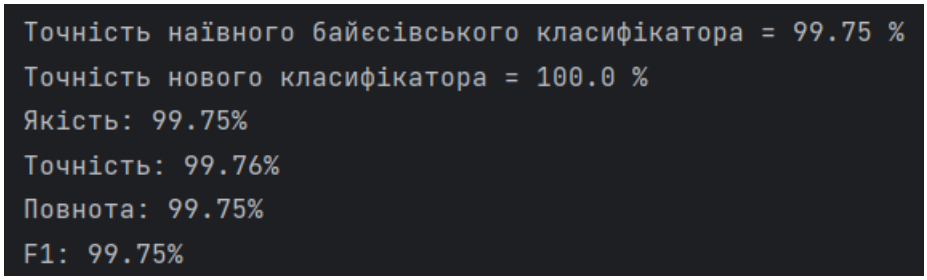


Рисунок 3.14 – Результати виконання

3.6 Практична робота 6

Тема. **Байєсовські методи визначення критерію якості розпізнавання зображень**

Мета роботи: побудувати матрицю неточностей.

Індивідуальне завдання

1. Створити новий файл Python і імпортувати необхідні пакети.
2. Створити програму матриці неточностей.
3. Завантажити вхідні дані у вигляді вибірки фактичних і передбачених міток класів.
4. Візуалізувати матрицю неточностей.
5. Зробити висновки.

Програмне забезпечення:

1. PyCharm
2. Python
3. NumPy
4. Matplotlib
5. Scikit-learn

Результат виконання коду див. на рис.3.15, 3.16.

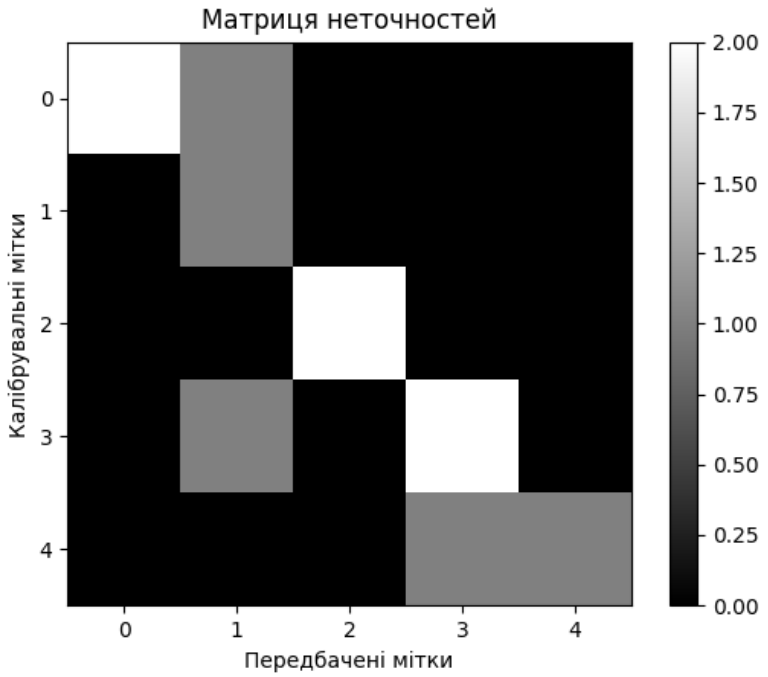


Рисунок 3.15 – Зображення оцінки класифікатора по відношенню до кожного з класів

	precision	recall	f1-score	support
Class-0	1.00	0.67	0.80	3
Class-1	0.33	1.00	0.50	1
Class-2	1.00	1.00	1.00	2
Class-3	0.67	0.67	0.67	3
Class-4	1.00	0.50	0.67	2
accuracy			0.73	11
macro avg	0.80	0.77	0.73	11
weighted avg	0.85	0.73	0.75	11

Рисунок 3.16 – Результативні дані виконання коду

3.7 Практична робота 7

Тема. Кластеризація даних за допомогою методу k-середніх

Мета роботи: провести оцінку кількості кластерів із використанням методу зсуву середнього.

1. Створити новий файл Python і імпортувати необхідні пакети.

2. Створити програму з моделлю кластеризації на основі зсуву середнього.

3. Завантажити вхідні дані з вибраного файлу.

4. Навчити модель кластеризації на основі алгоритму зсуву середнього.

5. Відобразити графік, який представляє кластери і їх центри.

6. Зробити висновки.

Програмне забезпечення:

1. PyCharm
2. Python
3. NumPy
4. Matplotlib
5. Scikit-learn

Результат виконання коду див. на рис. 3.17, 3.18.

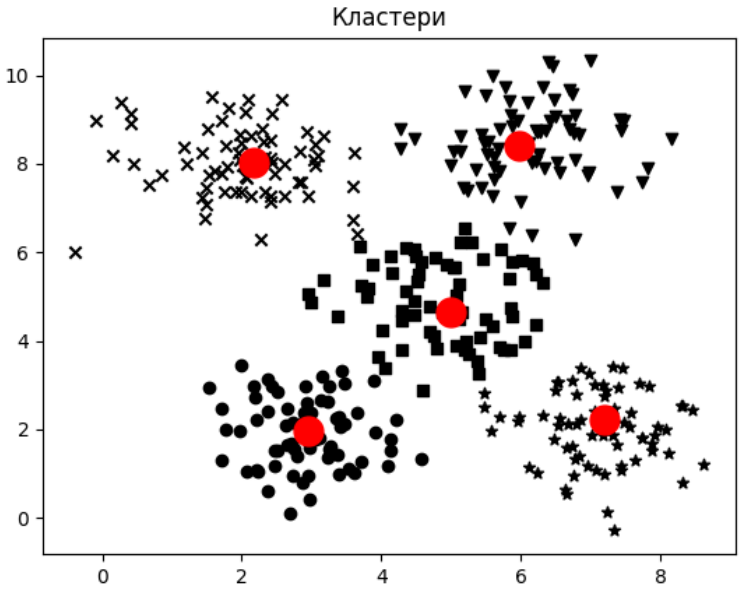


Рисунок 3.17 – Приклад зображення графіка з кластерами та їх центрами

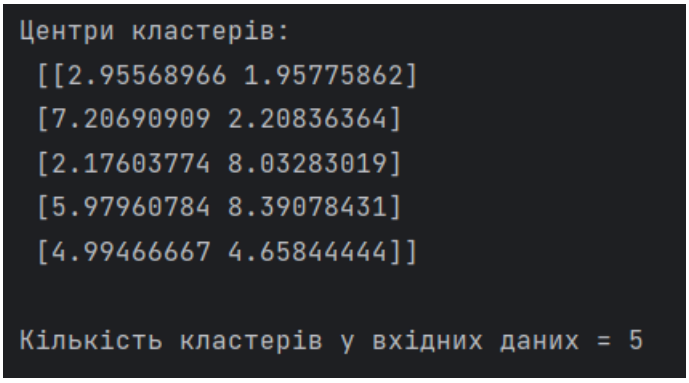


Рисунок 3.18 – Приклад результативних даних виконання коду, які інформують про координати положення центрів кластерів та кількість центрів кластерів

3.8 Практична робота 8

Тема. **Використання згорточної нейронної мережі для вирішення завдань класифікації**

Мета роботи: розробка та реалізація згорточної нейронної мережі глибокого навчання для класифікації зображень.

Індивідуальне завдання

1. Створити новий файл Python і імпортувати необхідні пакети

2.Отримати дані зображень вибраного набору.

3. Створити класифікатор зображень на основі згорточної нейронної мережі.

4. Провести процес навчання. Вивести дані про підвищення точності через кожні n ітерацій.

5. Обчислити точність класифікації, використовуючи тестовий набір даних.

6. Зробити висновки.

Програмне забезпечення:

1. PyCharm
2. Python
3. ImageAI
4. TourchVision

Результат виконання коду – див. рис. 3.19…3.21.

```
Вигляд x_train:  (60000, 28, 28, 1)
60000 Тренувальні зразки
10000 Пробні зразки
```

Рисунок 3.19 – Вигляд вибірки та кількість тренувальних/пробних зразків

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        320

 max_pooling2d (MaxPooling2D  (None, 13, 13, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 11, 11, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 5, 5, 64)         0
 2D)

 flatten (Flatten)           (None, 1600)              0

 dropout (Dropout)           (None, 1600)              0

 dense (Dense)               (None, 10)                16010

=================================================================
Total params: 34,826
Trainable params: 34,826
Non-trainable params: 0
```

Рисунок 3.20 – Вигляд відображення стеку слоїв моделі Sequential та сумарна кількість тренувальних параметрів

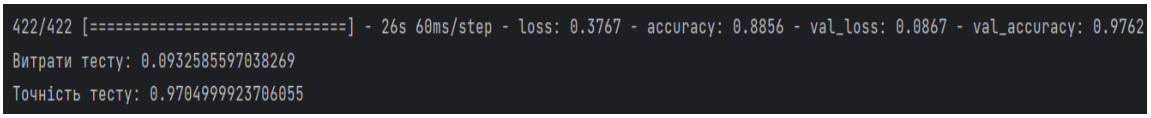


Рисунок 3.21 – Результати тесту після проведення аналізу та точність класифікації нейронної мережі для 1 епохи

3.9 Практична робота 9

Тема. Виявлення та відстеження об'єктів

Мета роботи: реалізація згорточної нейронної мережі для знаходження об'єктів на фотографії.

Індивідуальне завдання

1. Створіть нейронну мережу для розпізнавання об'єктів.
2. Завантажте файл моделі, який буде використовуватися для виявлення об'єктів.
3. Проведіть тестування мережі для виявлення одного і декількох унікальних об'єктів.
3. Зробіть висновки.

Програмне забезпечення:

1. PyCharm
2. Python
3. ImageAI
4. TourchVision

Результат виконання коду – див. рис. 3.22…3.25.

Рисунок 3.22 – Зображення 1 для виконання пошуку

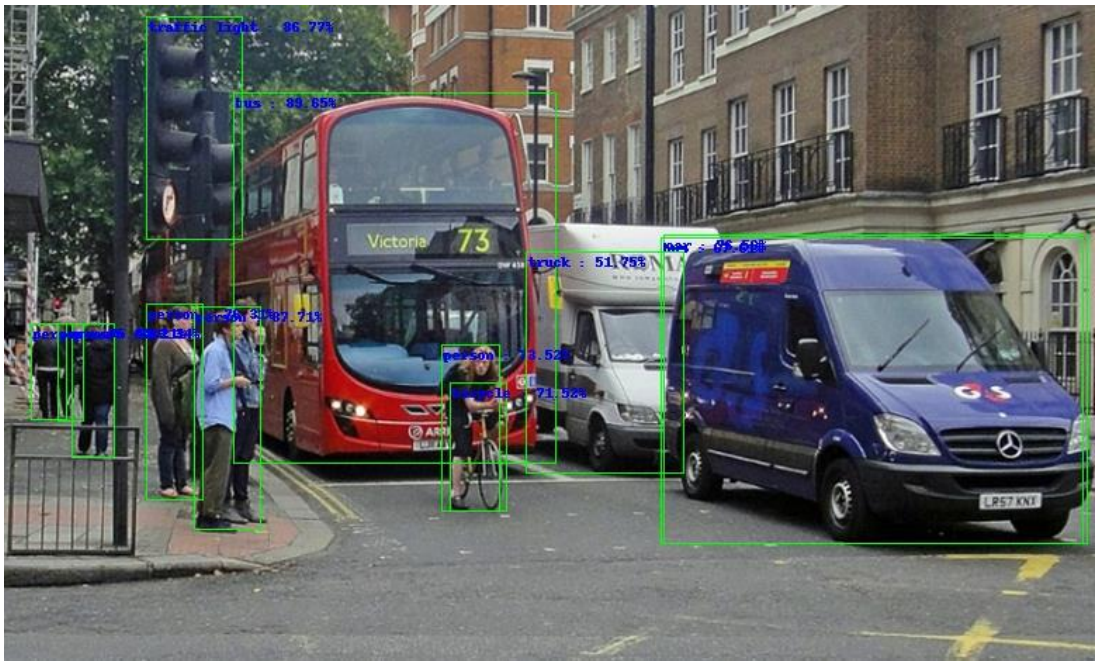


Рисунок 3.23 – Зображення 1 після розпізнавання

```
bus  :  89.65
person  :  87.71
traffic light  :  86.77
person  :  83.34
car  :  76.56
person  :  76.31
person  :  75.05
person  :  73.52
bicycle  :  71.52
bus  :  67.61
person  :  52.21
truck  :  51.75
```

Рисунок 3.24 – Консольні результати розпізнавання зображення 1

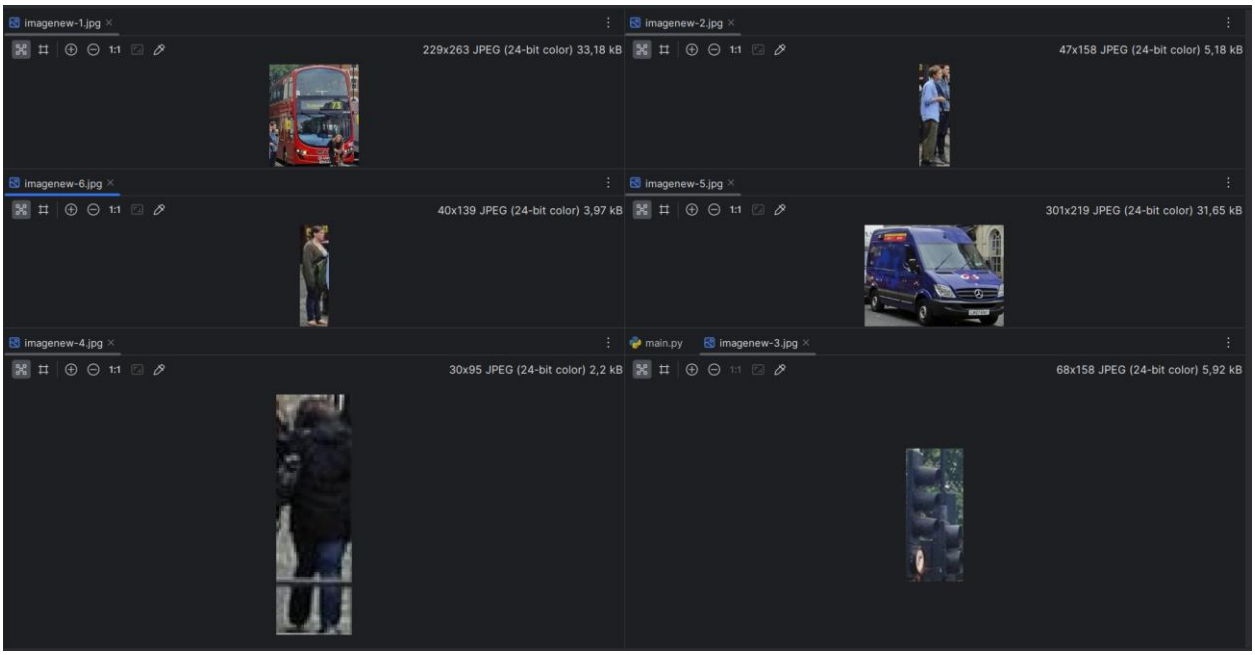


Рисунок 3.25 – Кожна розпізнана область на зображенні

СПИСОК ПОСИЛАНЬ

1. Gonzalez, R. (2018). Digital image processing. New York, NY: Pearson. ISBN 978-0-13-335672-4.

2. Canny, J., A Computational Approach To Edge Detection, IEEE Transactions on Pattern Analysis and Machine Intelligence, 1986, 8(6). pp. 679–698.

3. L. Roberts Machine Perception of 3-D Solids, Optical and Electro-optical Information Processing, MIT Press, 1965.

4. Prewitt, J. M. S. Object enhancement and extraction, in Picture Processing and Psychopictorics, B. Lipkin and A. Rosenfeld, Eds. New York: Academic, 1970, pp. 75-149.

5. Sobel, I. (2014). History and definition of the sobel operator. Retrieved from the World Wide Web, 1505, 29.

6. Szeliski, Richard. Computer Vision Algorithms and Applications. – Springer Cham, 2022.– 925 pp.– ISBN: 978-3-030-34372-9

7. Вовк, С. М. Методи обробки зображень та компютерний зір : навч. посіб. / С. М. Вовк, В. В. Гнатушенко, М. В. Бондаренко. – Дніпропетровськ : ЛІРА, 2016. – 148 с. – ISBN 978-966-383-699-7.

8. Antonio Torralba, Phillip Isola and William T. Freeman. Foundations of Computer. – MIT Press, 2024. – 840 pp.– ISBN: 9780262048972

9. Гнатушенко, В. В. Аналіз зображень. DOI: 10.5446/59673. [Електронний ресурс]

10. Піцун, О. Й. Адаптивний метод попередньої обробки гістологічних та цитологічних зображень/ О. Й. Піцун // Вісник НУ «Львівська політехніка». Комп’ютерні науки та інформаційні технології. – 2017. – № 864.

11. Бондіна, Н.М. Порівняння алгоритмів фільтрації медичних зображень за оцінками їх якості / Н. М. Бондіна, О. С. Калмичков, О. А. Козіна // Вісник НТУ «ХПІ». Серія: Інформатика та моделювання. – Харків : НТУ «ХПІ». – 2013. – № 39 (1012). – С. 15–21.

12. Постановка задач розпізнавання. [Електронний ресурс] URL: https://surl.lt/hozeaj

13. Кобилін, О. А. Методи цифрової обробки зображень : навч. посібник / О. А. Кобилін, І. С. Творошенко. – Харків : ХНУРЕ, 2021. – 124 с. – ISBN 978-966-659-295-1

14. Гороховатський В. О. (2020) Статистичне оброблення та аналіз даних у структурних методах класифікації зображень (монографія) / В. О. Гороховатський, С. В. Гадецька. – Харків, ФОП Панов А.Н. – 128 с.; DOI: 10.30837/978-617- 7859-69-6

15. Довбиш, А. С. Основи теорії розпізнавання образів : навч. посіб. У 2 ч. / А. С. Довбиш, І. В. Шелехов. – Суми : Сумський державний університет, 2015. – Ч. 1. – 109 с. – ISBN 987-966-657-596-1

16. Гороховський, О. І. Інтелектуальні системи : навчальний посібник / О. І. Гороховський. – Вінниця : ВНТУ, 2010. – 194 с.

17. Вуєвіч, Ж. (2021). Метрики оцінки моделей класифікації / Ж. Вуєвич // Міжнародний журнал передових комп’ютерних наук і застосувань. – 12(6); 599–606. DOI:10.14569/IJACSA.2021.0120670

18. Основи наукових досліджень і теорія експерименту : навч. посіб. для здобувачів освіт. ступеня «Магістр» спец. 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка» / Капаціла Ю. Б [та ін.]. – Тернопіл. нац. техн. ун-т ім. Івана Пулюя. – Тернопіль : Паляниця В. А., 2023. – 184 с. – ISBN 978-617-7875-34-4.

19. Карташов, М. В. Імовірність, процеси, статистика / М. В. Карташов. – К. : ВПЦ Київський університет, 2007. — 504 с

20. Бойко, Н. (2023). Алгоритми тренування та оцінки моделей машинного навчання для структурованого набору даних / Н. Бойко, Б. Левицький. – Information Technology: Computer Science, Software Engineering and Cyber Security, 3, 3–12, doi: https://doi.org/10.32782/IT/2023-3-1

21. Chen Y., Li C., Xu M. Business Analytics for Used Car Price Prediction with Statistical Models. 3rd International Conference on Economic Management and Cultural Industry (ICEMCI 2021), Guangzhou, China, 2021. P. 20–32. DOI: 10.2991/assehr.k.211209.090.

22. Mozina M, Demsar J, Kattan M, & Zupan B. (2004). «Nomograms for Visualization of Naive Bayesian Classifier». In Proc. of PKDD-2004, pages 337–348.

23. Наївний баєсів класифікатор. [Електронний ресурс] URL: https://surl.li/blzjva

24. Класифікація: точність, повнота, влучність і пов’язані показники. [Електронний ресурс] URL: https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall?hl=uk

25. Кононова, К. Ю. Машинне навчання: методи та моделі : підручник для бакалаврів, магістрів та докторів філософії спеціальності 051 «Економіка» / К. Ю. Кононова. – Харків : ХНУ імені В. Н. Каразіна, 2020. – 301 с.

26. 26 Основні поняття кластеризації та постановка задачі. [Електронний ресурс] URL: https://csc.knu.ua/media/study/asp/mod_probl_inf_tech_sys_analysis_ivohin/lecture/lec11.pdf (останнє звернення 18.02.2026

27. Що таке алгоритм середнього зсуву. [Електронний ресурс] URL: https://surl.li/ffyvap

28. Машинне навчання простими словами. [Електронний ресурс] URL: http://mmf.lnu.edu.ua/ar/1739

29. Prasad S. Different Types of Clustering Methods and Applications Sunit Prasad // Analytix Labs. 2020. [Електронний ресурс] URL: https://www.analytixlabs.co.in/blog/types-of-clustering-algorithms/.

30. Selecting the number of clusters with silhouette analysis on KMeans clustering: [Електронний ресурс] URL: https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html

31. Sklearn.metrics.silhouette_score [Електронний ресурс] URL: https://lijiancheng0614.github.io/scikit-learn/modules/generated/sklearn.metrics.silhouette_score.html

32. Oleksii Malyshev. Використовуємо CNN для обробки зображень [Електронний ресурс] URL: https://dou.ua/forums/topic/48368/.

33. Згорткові нейронні мережі. [Електронний ресурс] URL: https://www.youtube.com/watch?v=yviFVY-Jozs.

34. Ninh, Pham; Rasmus, Pagh (2013). Fast and scalable polynomial kernels via explicit feature maps. SIGKDD international conference on Knowledge discovery and data mining. Association for Computing Machinery.doi:10.1145/2487575.2487591

35. Théodore Bluche Deep Neural Networks – Applications in Handwriting Recognition, [Електронний ресурс] URL: https://www.tbluche.com/files/MeetupSaoPaulo2017.pdf

36. А для чого потрібний математичний аналіз у Machine Learning? (2023) [Електронний ресурс] URL: http://www.mmf.lnu.edu.ua/ar/2017

37. Олег Романів. Методи лінійної алгебри в Data Science та Machine Learning [Електронний ресурс] URL: http://www.mmf.lnu.edu.ua/algstu/2016 (2024)

38. VGG16 [Електронний ресурс] URL: https://www.mathworks.com/help/deeplearning/ref/vgg16.html

39. Densely Connected Convolutional Networks [Електронний ресурс] URL: https://arxiv.org/pdf/1608.06993

40. Р.М. Тимчишин, О.Є. Волков, О.Ю. Господарчук, Ю.П. Богачук Сучасні підходи до розв'язання задач комп'ютерного зору [Електронний ресурс] URL: http://usim.org.ua/arch/2018/6/8.pdf

41. Маленчик, Т. В. Аналіз алгоритмів виявлення та супроводження точкових об'єктів у відеопотоці / Т. В. Маленчик, О. Ю. Мирончук, О. С. Неуймін // Вісник Вінницького політехнічного інституту. – 2022. – № 648 : https://doi.org/10.31649/1997-9266-2022-165-6-48-56

42. Бабарика, А. О. Обґрунтування показника вибору оптимального алгоритму виділення фону у відеопослідовностях з камер відеоспостереження відомчих систем відеоспостереження / А. О. Бабарика // Сучасні інформаційні технології у сфері безпеки та оборони. – К. : Національний університет оборони України, 2019. – № 3 (36). – С. 97–102. DOI : http://dx.doi.org/10.33099/2311-7249/2019-36-3-97- 102.

43. Катеринчук, И. С. Усовершенствование алгоритма для обнаружения динамических объектов на видеопоследовательностях / И. С. Катеринчук, А. А. Бабарика: e-ISSN 1607-3274 Радіоелектроніка, інформатика, управління. 2020. № 3 p-ISSN 2313-688X Radio Electronics, Computer Science, Control. 2020. № 3 с.88– 96

44. Middlebury Optical flow evaluation and ground truth sequences. [Електронний ресурс] URL: https://vision.middlebury.edu/flow/floweval-ijcv2011.pdf

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

45. Конспект лекцій з вивчення дисципліни «Обробка біомедичних зображень та реконструкція об'єктів» для студентів спеціальності 163 – Біомедична інженерія, освітня програма «Інтелектуальні штучні імпланти та медичні апарати в біоінженерії» / уклад. Л.Г. Коваль. – Вінниця : ВНТУ, 2020.

46. Методичні вказівки до виконання практичних робіт з дисципліни «Обробка біомедичних зображень та реконструкція об'єктів» для студентів спеціальності 163 – Біомедична інженерія, освітня програма «Інтелектуальні штучні імпланти та медичні апарати в біоінженерії» / уклад. Л. Г. Коваль. – Вінниця : ВНТУ, 2020.

47. Путятін, Є. П. Методи та алгоритми комп’ютерного зору : навч. посібник / Є. П. Путятін, В. О. Гороховатській, О. О. Матат. – Харків : СМІТ, 2006. – 236 с.

48. Gonzalez, Rafael (2018). Digital image processing. New York, NY: Pearson. ISBN 978-0-13-335672-4.

49. Bovik A. C. Handbook of image and video processing. – Academic press, 2010.

50. Jahne B. Practical handbook on image processing for scientific and technical applications. – CRC press, 2004.

51. Parker J.R. 2010. Algorithms for Image Processing and Computer Vision. Second Edition. Wiley Publishing, Inc.

Навчальне видання

БОГДАНОВА Ліна Михайлівна
ВАСИЛЬЄВА Людмила Володимирівна

МЕТОДИ ОБРОБКИ ЗОБРАЖЕНЬ І КОМП'ЮТЕРНИЙ ЗІР

Навчальний посібник

для здобувачів другого (магістерського) рівня вищої освіти

Редагування Дудченко О. О.

48/2024. Формат 60 × 84/16. Ум. друк. арк. 6,74.
Обл.-вид. арк. 5,27. Тираж 100 пр. Зам. №

Видавець і виготівник
Донбаська державна машинобудівна академія
84313, м. Краматорськ, вул. Академічна, 72.
Свідоцтво суб'єкта видавничої справи
ДК № 1633 від 24.12.2003