МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет

Факультет електроніки та інформаційних технологій

Кафедра прикладної математики та моделювання складних систем

«До захисту допущено»
Завідувач кафедри
_____ _Ігор КОПЛИК_
(підпис) (Ім'я та ПРІЗВИЩЕ)
___ _____20__ р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 113 «Прикладна математика», освітньо-професійної програми «Наука про дані та моделювання складних систем» на тему:

«Порівняльний аналіз ефективності методів класифікації зображень»

Здобувача групи ПМ-01 Пустового Юрія Михайловича

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Пустовий Юрій
(підпис) (Ім'я та ПРІЗВИЩЕ здобувача)

Керівник: док. фіз.-мат. наук, професор Олександр ЛИСЕНКО _____
(підпис)

Суми – 2024

АНОТАЦІЯ

Кваліфікаційна робота: 43 с., 8 рисунків, 11 джерел.

Мета роботи: Виконати порівняльний аналіз ефективності різних методів класифікації зображень

Об’єкт дослідження: Зображення різних категорій, алгоритми для задачі класифікації зображень.

Предмет дослідження: Якісні характеристики алгоритмів для задачі класифікації зображень

Методи дослідження: Згорткова нейронна мережа (CNN), метод опорних векторів (SVM), метод k-найближчих сусідів (k-NN), дерево рішень (Decision Trees)

Використовуючи датасет CIFAR-10, який містить 60,000 кольорових зображень, що відносяться до десяти класів, проведено порівняльний аналіз ефективності різних методів класифікації зображень. Класифікацію зображень проведено на основі наступних алгоритмів: згорткової нейронної мережі (CNN), методу опорних векторів (SVM), методу k-найближчих сусідів (k-NN), дерева рішень (Decision Trees).

З'ясовано, що алгоритм, який побудовано на згортковій нейронній мережі (CNN) показує найвищу загальну точність, мінімізує помилкові спрацьовування. В роботі запропоновано рекомендації щодо вибору алгоритму для задачі класифікації виходячи з завдань конкретної задачі.

Ключові слова: КЛАСИФІКАЦІЯ, КЛАСИФІКАЦІЯ ЗОБРАЖЕНЬ, АЛГОРИТМИ КЛАСИФІКАЦІЇ, ЗГОРТКОВА НЕЙРОННА МЕРЕЖА.

ЗМІСТ

ВСТУП

Порівняльний аналіз ефективності методів класифікації зображень є важливим аспектом досліджень в області машинного навчання та комп'ютерного зору. Основною метою такого аналізу є визначення найефективніших алгоритмів для різних типів завдань класифікації зображень.

Одним з популярних підходів є використання згорткових нейронних мереж (CNN), які показують високі результати завдяки своїй здатності виявляти складні патерни в зображеннях [1]. Зокрема, архітектури як AlexNet, VGG, ResNet, і Inception продемонстрували відмінні результати в різних задачах класифікації та обробки зображень [2].

Для оцінки ефективності різних методів класифікації використовують різноманітні метрики, такі як точність, F1-міра, та час виконання. Наприклад, алгоритми YOLO (You Only Look Once) та SSD (Single Shot MultiBox Detector) відзначаються високою швидкістю обробки, що є важливим для реальних застосувань, де критичним є час виконання, тоді як Faster R-CNN забезпечує високу точність за рахунок більшого часу на обробку [1].

Крім CNN, традиційні методи машинного навчання, такі як SVM (Support Vector Machine) та KNN (k-Nearest Neighbors), також використовуються для порівняння, хоча зазвичай вони поступаються глибинним моделям у складних завданнях класифікації зображень [2].

Таким чином, вибір алгоритму для конкретної задачі класифікації зображень залежить від вимог до точності, швидкості та обчислювальних ресурсів. Застосування сучасних інформаційних технологій і систем значно полегшує процес проведення складних розрахунків та підвищує достовірність отриманих результатів, що має велике значення для практичного використання цих методів у різних галузях [1].

РОЗДІЛ 1.
КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ (ОГЛЯД ЛІТЕРАТУРИ)

1.1 ОСНОВНІ ОЗНАЧЕННЯ

Класифікація зображень — це процес визначення категорії, до якої належить зображення з визначеного набору категорій. Даний процес важливий для ряду наукових, промислових та соціальних застосувань, таких як медична діагностика, автоматичне тегування фотографій у соціальних мережах, безпека та моніторинг, автономні транспортні засоби, тощо [1], [2].

Класифікація зображень є частиною ширшого поля комп'ютерного бачення і машинного навчання. Вона використовує алгоритми та моделі для ідентифікації та категоризації об'єктів або особливостей зображення [2].

Застосування класифікації зображень має широкий діапазон. В медицині, наприклад, вона може використовуватися для ранньої діагностики хвороб, таких як рак шкіри, шляхом аналізу зображень шкірних уражень [2]. У сфері безпеки, класифікація може допомогти у розпізнаванні підозрілих об'єктів або дій. В автомобільній промисловості, системи класифікації можуть використовуватися в автономних транспортних засобах для розпізнавання дорожніх знаків, пішоходів і інших автомобілів [1], [2].

1.1.1 Типи даних для класифікації зображень

Класифікація зображень може здійснюватися на основі різних типів даних, кожен з яких має свої особливості та обмеження.

Градація сірого є базовим типом зображень, в яких кожний піксель може приймати значення від чорного до білого. Градації сірого широко використовуються в медицині, наприклад, для рентгенівських зображень [2], [1].

У кольорових зображеннях кожен піксель містить інформацію про кольори, зазвичай в RGB (Червоний, Зелений, Синій) просторі. Кольорові зображення важливі для розпізнавання об'єктів в натуральних умовах, наприклад, в автономних автомобілях [2].

Також використовують динамічні зображення. Це послідовності кадрів, що дають можливість аналізувати зміни в часі. Відеоаналіз може застосовуватися в системах спостереження та безпеки [1].

Високодеталізовані або високодинамічні зображення зображення містять додаткову інформацію про яскравість та контраст, що може бути корисно в складних освітлених умовах [1].

Мультиспектральні та гіперспектральні зображення зображення містять інформацію з різних сегментів електромагнітного спектру і використовуються в геологічних дослідженнях та сільському господарстві [2].

Кожен тип даних вимагає відповідних методів предобробки та алгоритмів класифікації. Наприклад, для кольорових зображень можуть використовуватися алгоритми, які враховують кольорову інформацію, тоді як для градацій сірого важливіше може бути текстурна інформація [1].

1.1.2 Препроцесинг даних для класифікації зображень

Цифрове подання зображень є ключовим аспектом для їх подальшої обробки і класифікації. Ось декілька ключових аспектів цього подання:

Розмірність

Зображення зазвичай представляються як двовимірні матриці для градацій сірого і тривимірні матриці для кольорових зображень (ширина х висота х канали кольору). Відео представляє собою чотиривимірний тензор (ширина х висота х канали кольору х час) [2].

Бітова глибина

Кількість бітів, що використовуються для кодування інтенсивності кожного пікселя, може варіюватися. Зазвичай, вона складає 8 бітів (256 рівнів сірого) для градацій сірого і 24 біти (8 бітів на канал) для кольорових зображень [1].

Кольорові простори

Крім RGB, існують інші кольорові простори, такі як HSV (Hue, Saturation, Value), LAB та YCbCr, кожний з яких має свої переваги в певних задачах класифікації [1].

Висока динамічна діапазонність (HDR)

В HDR-зображеннях динамічний діапазон значно вищий, що дозволяє зберегти деталі в дуже темних і дуже світлих областях [1].

Формати файлів

Зображення можуть зберігатися в різних форматах файлів, таких як JPEG, PNG, TIFF тощо. Кожен формат має свої особливості стиснення та збереження даних [2].

Метадані

Зображення можуть містити метадані, такі як час створення, геопозиція, параметри камери та ін., які можуть бути важливими для деяких задач класифікації [2].

Розуміння цих аспектів є критично важливим для вибору відповідних алгоритмів предобробки та методів класифікації. Наприклад, зображення з високою бітовою глибиною або HDR можуть вимагати спеціалізованих методів обробки для ефективної класифікації.

1.1.3 Особливості та властивості зображень

Зображення мають ряд ключових особливостей і властивостей, які варто взяти до уваги під час їх аналізу та класифікації. До цих особливостей належать:

Текстура

Текстура зображення описує "шорсткість" або "гладкість" ділянок. У математичному виразі текстура може бути описана за допомогою статистичних методів, таких як ко-варіація, кореляція, ентропія тощо. Текстура відіграє важливу роль у завданнях комп'ютерного зору, таких як розпізнавання об'єктів, виявлення дефектів на поверхні, ідентифікація патернів, медичний аналіз зображень та інше. Існують різні методи аналізу текстури, включаючи статистичні, структурні, модельні та методи перетворення [3], [4].

Контраст і яскравість

Контраст зображення відноситься до різниці між найсвітлішими та найтемнішими пікселями, що дозволяє виділити об'єкти на зображенні. Яскравість характеризує абсолютну величину інтенсивності пікселів, впливаючи на загальне сприйняття зображення. Високий контраст може допомогти в ідентифікації об'єктів на зображенні, тоді як яскравість є важливою в медичних зображеннях для коректної інтерпретації даних [4].

Кольорові характеристики

Кольорові характеристики зображень можуть суттєво впливати на результати класифікації. Різні кольорові простори, такі як RGB, HSV, LAB та YCbCr, можуть бути використані залежно від специфіки задачі. Наприклад, HSV простір дозволяє розділяти кольори за відтінком, насиченістю та яскравістю, що може бути корисним для сегментації зображень [4].

Геометричні характеристики

Геометричні характеристики включають форму, розмір і орієнтацію об'єктів на зображенні. Вони використовуються для розпізнавання об'єктів і аналізу сцени, наприклад, у біологічних дослідженнях для класифікації різних типів клітин за формою [3].

Залежності між сусідніми пікселями

Інформація про просторові залежності між пікселями є критично важливою для алгоритмів класифікації. Згорткові нейронні мережі (CNN)

ефективно враховують ці залежності, що дозволяє їм розпізнавати складні структури та патерни на зображеннях [4].

Шум і артефакти

Зображення можуть містити шуми та артефакти, які створюють додаткові виклики в процесі класифікації. Це може бути шум від сенсора, артефакти стиснення або інші види перешкод. Обробка шуму і артефактів є важливою частиною предобробки зображень для підвищення точності класифікації [4].

Розуміння і врахування цих особливостей і властивостей є критично важливим для ефективної класифікації зображень. Вони впливають на вибір методів предобробки та саму ефективність алгоритмів класифікації.

1.1.4 Алгоритми класифікації зображень

k-NN (k-Nearest Neighbors)

Основний принцип: Класифікація відбувається на основі "близькості" об'єктів в просторі ознак. Для кожного нового об'єкта шукаються k найближчих сусідів з відомими класами. Відстань між об'єктами зазвичай вимірюється за допомогою евклідової відстані, але можуть використовуватися й інші метрики відстані [6], [7].

$$\text{Class}(x) = \text{mode}\{yi \mid d(x, xi) \leq k\}$$

де d(x, xi) - це функція відстані між об'єктом x і даними x_i, а y_i - мітки класів найближчих сусідів.

Функції відстані:

- Евклідова відстань: $d(x, y) = \sqrt{\sum_{i=1}^{n}(xi - yi)^2}$
- Манхеттенська відстань: $d(x, y) = \sum_{i=1}^{n} \mid xi - yi \mid$
- Відстань Чебишева: $d(x, y) = \max i \mid xi - yi \mid$

Переваги:

- Простий у розумінні та імплементації.
- Не вимагає тренування моделі, оскільки всі дані використовуються під час класифікації.
- Точно працює для простих задач та добре підходить для невеликих наборів даних.

Недоліки:

- Потребує значних обчислювальних ресурсів для пошуку найближчих сусідів, особливо на великих наборах даних.
- Чутливий до викидів та шумів в даних, що може знизити точність класифікації.
- Продуктивність різко падає з підвищенням розмірності простору ознак (проблема "прокляття розмірності").

Дерева рішень

Основний принцип: Класифікація здійснюється шляхом проходження по "дереву рішень", де кожен вузол представляє собою умову або питання про ознаку об'єкта. Кожна гілка дерева відповідає результату цієї умови, а листові вузли представляють класи об'єктів [6], [7], [9].

Дерева рішень не використовують одну математичну формулу для передбачення, а будують модель, яка поділяє простір ознак на прямокутники, і кожного прямокутника видає однаковий прогноз. Процес поділу на підпростори ґрунтується на критеріях, таких як мінімізація ентропії або максимізація приросту інформації [6], [7], [9]:

$$IG(Dp, f) = I(Dp) - \sum_{j=1}^{m} \frac{N_j}{N} * I * (D_j)$$

де IG - приріст інформації, D_p і D_j - розділи даних до і після поділу, I - міра невизначеності (наприклад, ентропія), N_j - кількість точок у розділі D_j, N - загальна кількість точок в D_p, f - ознака для поділу.

Критерії поділу

Найчастіше використовувані критерії визначення точок поділу включають:

- Ентропія: Міра невизначеності чи випадковості.
- Індекс Джіні: Захід чистоти/неоднорідності прогнозованої змінної в підмножинах.
- Помилка класифікації: Простий, але рідше використовуваний критерій.

Дерева рішень легко інтерпретуються і можуть обробляти як числові, так і категоріальні дані. Вони можуть створювати складні структури, приймаючи рішення з урахуванням чітко визначених правил, що робить їх потужним інструментом у багатьох сценаріях аналізу даних.

Переваги:

- Легко інтерпретується та візуалізується.
- Може працювати з даними різних типів (числові та категоріальні).
- Не вимагає значної предобробки даних.
- Швидкість прогнозування порівняно висока.

Недоліки:

- Схильність до перенавчання, особливо на малих вибірках.
- Може бути нестабільним при змінах у даних (зміна одного значення може суттєво змінити структуру дерева).
- Погано працює з високорозмірними даними без додаткової оптимізації.

SVM (Support Vector Machines)

Основний принцип: SVM шукає оптимальну гіперплощину, яка максимально розділяє класи в просторі ознак. Гіперплощина визначається так, щоб

максимізувати відстань (маржу) між найближчими точками класів (підтримуючі вектори). Використовуються ядрові функції для обробки нелінійних розділень [6], [7], [9].

Математично гіперплощину можна описати як вирішення наступного завдання оптимізації:

$$\text{Minimize: } \frac{1}{2}\left|\left|w\right|\right|^2 \text{ за умови } y_i * (w * x_i + b) \geq 1, \forall_i$$

де:

- w - ваговий вектор гіперплощини,
- x_i - вхідні вектори,
- y_i — позначки класів, які можуть бути +1 або -1,
- b - зміщення щодо початку координат.

Ядровий трюк

У випадках, коли дані не лінійно розділені, SVM використовує так званий "ядровий трюк", який дозволяє виконувати лінійну класифікацію у високорозмірному просторі, в якому вихідні дані можуть бути розділено лінійно.

$$K(xi, xj) = \phi(xi) \cdot \phi(xj)$$

де ϕ — функція, яка відображає вхідні дані у високорозмірний простір.

Приклади популярних ядер:

- Поліноміальне ядро: $(xi \cdot xj + 1)d$
- Радіальне базисне (RBF) ядро: $\exp(-\gamma \parallel xi - xj \parallel 2)$

Переваги:

- Ефективний на високорозмірних просторах ознак.
- Висока точність класифікації, особливо для задач з чітким розділенням класів.

- Гнучкість через використання різних ядерних функцій (лінійне, поліноміальне, радіально-базисне тощо).

Недоліки:

- Велика обчислювальна складність при великих датасетах.
- Вибір ядерної функції може бути неочевидним і вимагати багато експериментів.
- Чутливість до масштабування ознак, що може вимагати попередньої нормалізації даних.

Згорткові нейронні мережі (CNN)

Основний принцип: Використовуються для врахування просторової структури зображення за допомогою конволюційних та пулінгових шарів. Згорткові шари застосовують фільтри для виділення ознак, а пулінгові шари зменшують розмірність даних, зберігаючи важливу інформацію. CNN є глибинними моделями, здатними вивчати складні патерни в даних [1], [2], [4], [10].

Згортка нейрона мережа визначається так:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n) * K(i-m, j-n)$$

де *I* - це вхідне зображення, *K* - ядро згортки, а *S(i, j)* - елемент вихідної матриці або карти ознак. Ця операція дозволяє локалізувати просторові ієрархії ознак у зображенні, що робить CNN ефективним для аналізу візуального контенту.

Функція активації ReLU (Rectified Linear Unit) застосовується після кожної операції згортки і задається формулою:

$$R(z) = max(0, z)$$

Функція ReLU нелінійно перетворює вхідні дані, обнуляючи всі негативні значення та зберігаючи позитивні значення без змін. Це усунення проблеми зникаючого градієнта під час навчання мережі.

Пулінг (Max Pooling). Максимальний пулінг - це процес зменшення розмірності кожної карти ознак, зберігаючи лише найбільш значущі значення:

$$P(i,j) = max(L(x,y))$$

де L являє собою матрицю ознак у вікні пулінгу x, y, причому x і y проходять через певне підмножина області карти ознак. Максимальний пулінг допомагає зробити уявлення ознак інваріантним до масштабу та невеликих зрушень, зменшуючи при цьому кількість параметрів та обчислень.

Переваги:

- Відмінна здатність до узагальнення, особливо для складних задач класифікації зображень.
- Можливість автоматичного виділення складних ознак без потреби в ручному визначенні.
- Добре працює з великими датасетами завдяки здатності навчатися на великих обсягах даних.

Недоліки:

- Висока обчислювальна вартість, потребує потужного апаратного забезпечення (GPU).
- Ризик перенавчання при недостатньому датасеті.
- Складність налаштування архітектури мережі та параметрів навчання.

1.2 АНАЛІЗ МЕТОДІВ КЛАСИФІКАЦІЇ

1.2.1 Критерії оцінки

Об'єктом аналізу є зображення, які можуть варіюватися від простих зображень, таких як рукописні цифри, до складних медичних знімків або сателітних фотографій. Важливо розуміти характер об'єкта аналізу для вибору найбільш відповідного методу класифікації.

Мета аналізу полягає у визначенні найефективніших методів класифікації для вирішення конкретної задачі. Це може бути досягнуто шляхом оцінки різних метрик, таких як точність, швидкість обробки, ресурсоємність та інші показники продуктивності.

Для об'єктивного аналізу методів класифікації необхідно визначити чіткі критерії оцінки. Найбільш поширені критерії включають:

- **Точність (Accuracy):** Пропорція правильних прогнозів до загальної кількості прогнозів.
- **Точність (Precision):** Пропорція правильних позитивних прогнозів до всіх позитивних прогнозів.
- **Повнота (Recall):** Пропорція правильних позитивних прогнозів до всіх реальних позитивних випадків.
- **F1-міра:** Гармонійне середнє точності і повноти.

Вибір методу класифікації може значно вплинути на загальну ефективність системи. Наприклад, неправильний вибір методу може призвести до підвищення часу обробки або зниження точності класифікації. Тому важливо враховувати всі аспекти задачі та можливості кожного методу.

Кожен метод класифікації має свої вимоги до ресурсів та обмеження. Наприклад, згорткові нейронні мережі (CNN) потребують значних обчислювальних ресурсів і потужного апаратного забезпечення для ефективного навчання та роботи. Інші методи, такі як k-NN, можуть бути менш вимогливими

до апаратних ресурсів, але потребують оптимізації для роботи з великими наборами даних.

1.2.2 Методологія порівняння методів класифікації

Для проведення порівняльного аналізу методів класифікації важливо визначити чітку методологію. Одним з популярних підходів є крос-валідація, яка забезпечує більш точну оцінку продуктивності моделей на основі поділу даних на тренувальні та тестові набори декілька разів і усереднення результатів. Інший підхід — A/B тестування, яке дозволяє безпосередньо порівняти два методи на окремих підмножинах даних. Також можуть використовуватися інші статистичні методи для оцінки стабільності та узагальнення моделей [5], [6].

Обрані метрики повинні відображати різні аспекти продуктивності методів класифікації. Основні метрики включають: точність (Accuracy), точність (Precision), повноту (Recall), F1-міру.

Для об'єктивного порівняння важливо використовувати однаковий датасет для всіх методів. Це забезпечує, що всі методи класифікації оцінюються за однаковими умовами і на однакових даних, що дозволяє зробити справедливі висновки про їх продуктивність. Використання стандартних наборів даних, таких як CIFAR-10 для класифікації зображень, є поширеною практикою [5].

Проведення експериментів

Кожен метод тренується і тестується на однаковому датасеті, використовуючи однакові метрики для оцінки. Це включає настройку гіперпараметрів моделей, навчання на тренувальних даних та оцінку на тестових даних. Важливо забезпечити, щоб усі методи класифікації мали однакові умови навчання і тестування для об'єктивного порівняння [6].

Аналіз результатів

Після проведення експериментів результати аналізуються для визначення переваг та недоліків кожного методу. Результати можуть бути представлені у

вигляді таблиць, графіків або інших візуалізацій для полегшення порівняння. Важливо зробити висновки про те, який метод є найбільш ефективним в різних умовах, наприклад:

- **При обмеженій кількості даних:** Методи, які краще працюють з малими вибірками.
- **При високих вимогах до точності:** Методи з найвищою точністю класифікації.
- **При обмежених обчислювальних ресурсах:** Методи, які потребують менше ресурсів для навчання і роботи [5], [6].

1.2.3 Оцінка ефективності методів класифікації

Кожен метод може показувати високу ефективність в одних умовах і бути непридатним в інших. Тому важливо зрозуміти, в яких ситуаціях використовувати певний метод. Наприклад, метод k-NN може бути ефективним для невеликих наборів даних, але не підходить для великих обсягів через високу обчислювальну складність. Аналіз чутливості допомагає зрозуміти, як змінюється ефективність методу при зміні параметрів задачі, таких як кількість даних або складність моделі [5], [6].

Метрика затрат

Не всі методи однаково "дешеві" з точки зору обчислювальних затрат. Оцінка ефективності повинна включати не тільки точність, але і ресурси, необхідні для тренування та впровадження моделі. Наприклад, CNN вимагають значних обчислювальних ресурсів і часу на навчання, тоді як SVM може бути менш вимогливим при відповідному виборі ядерної функції [6].

Важливо оцінити, наскільки метод є стійким до шуму, викидів в даних або змін у розподілі даних. Це особливо актуально в реальних умовах, де дані рідко бувають "чистими". Методи, які демонструють високу робастність, можуть бути

більш надійними для практичного застосування, навіть якщо вони не мають найвищої точності на чистих даних [5], [6].

Деякі методи можуть бути теоретично ефективними, але складними для інтеграції в існуючу систему. Наприклад, інтеграція складних нейронних мереж може вимагати значних змін в інфраструктурі системи, що може бути непрактичним для деяких організацій. Цей аспект теж варто враховувати при оцінці ефективності методів [5].

Для деяких застосувань (наприклад, медичних або юридичних) важлива не лише висока точність прогнозу, але і можливість інтерпретації результатів. Методи, які дозволяють легко інтерпретувати результати, такі як дерева рішень, можуть бути переважними у цих сферах. Натомість, складні нейронні мережі можуть бути менш прозорими і важче інтерпретуються [5], [6].

По завершенню цього етапу ми матимемо збалансований погляд на ефективність різних методів класифікації зображень, що дозволить найбільш обґрунтовано підійти до вибору конкретного методу для реалізації.

РОЗДІЛ 2:
РЕАЛІЗАЦІЯ КОМП'ЮТЕРНОГО ЕКСПЕРИМЕНТУ ТА АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

Цей розділ фокусується на практичному використанні розглянутих в першому розділі методів та алгоритмів. Тут описані етапи комп'ютерного експерименту, використані датасети, метрики оцінки, а також аналіз отриманих результатів.

2.1 ВИБІР ТА ПІДГОТОВКА ДАТАСЕТУ

В даному експерименті було вибрано датасет CIFAR-10, який є одним з найбільш поширених у задачах класифікації зображень. Датасет містить 60,000 кольорових зображень розміром 32x32 пікселі, які розділені на 10 різних класів: автомобілі, птахи, коти, олені, собаки, жаби, коні, кораблі, літаки та вантажівки. Кожний клас має рівну кількість зразків — 6,000.

Підготовка даних

1. **Нормалізація**: Для прискорення процесу навчання та зменшення можливих перешкод всі пікселі нормалізуються, тобто приводяться до діапазону від 0 до 1.
2. **Розбиття на вибірки**: Датасет розбивається на тренувальну, валідаційну та тестову вибірки.

Вибір метрик оцінки

Для оцінки якості моделі використовуються такі метрики, як точність (Accuracy), F1-метрика, та матриця невідповідностей (Confusion Matrix).

2.2 ЕКСПЕРИМЕНТАЛЬНА ПЕРЕВІРКА РОЗРОБЛЕНИХ МОДЕЛЕЙ

2.2.1 CNN модель

Області застосування

CNN (Convolutional Neural Networks) найкраще підходять для задач, де важливо враховувати просторову структуру даних. Вони ефективно використовуються для класифікації зображень, обробки відео, розпізнавання облич, аналізу медичних зображень (наприклад, рентгенівських знімків), та в системах автоматичного керування, де необхідно виділяти та аналізувати складні патерни візуальних даних.

Неефективне застосування

Методи CNN можуть бути неефективними для задач з невеликою кількістю даних, оскільки для їх навчання зазвичай потрібні великі обсяги мічених даних. Крім того, тренування CNN вимагає значних обчислювальних ресурсів, що може бути проблематичним для застосувань з обмеженими ресурсами. Вони також можуть бути менш ефективними для задач, де просторові взаємозв'язки між ознаками не мають значення або де дані не мають регулярної структури, як у випадку з деякими типами табличних даних.

Структура нейроної мережі

Розглянемо структуру згорткової нейроної мережі:

```
model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
```

```
    layers.Dense(10, activation='softmax')
])
```

Структура нейронної мережі:

1. Шар згортки (Conv2D):
   layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3))
   - Призначення: виконує згортку вхідного зображення з використанням 32 фільтрів розміром 3x3.
   - Функція активації: ReLU (Rectified Linear Unit) додає нелінійність.
   - Вхід: зображення розміром 32x32 з 3 каналами (RGB).
   - Вихід: набір зображень (feature maps) з 32 каналами.

2. Шар максимального об'єднання (MaxPooling2D):
   layers.MaxPooling2D((2, 2))

   - Призначення: зменшує розмірність даних, вибираючи максимальні значення в кожному вікні розміром 2x2.
   - Вхід: набір зображень (feature maps) з 32 каналами.
   - Вихід: набір зображень (feature maps) з 32 каналами, але розмір зображень зменшено вдвічі.

3. Шар згортки (Conv2D):
   layers.Conv2D(64, (3, 3), activation='relu')

   - Призначення: виконує згортку з використанням 64 фільтрів розміром 3x3.
   - Функція активації: ReLU.
   - Вхід: набір зображень (feature maps) з 32 каналами.
   - Вихід: набір зображень (feature maps) з 64 каналами.

4. Шар максимального об'єднання (MaxPooling2D):
   layers.MaxPooling2D((2, 2))

- Призначення: зменшує розмірність даних, вибираючи максимальні значення в кожному вікні розміром 2x2.
- Вхід: набір зображень (feature maps) з 64 каналами.
- Вихід: набір зображень (feature maps) з 64 каналами, але розмір зображень зменшено вдвічі.

5. Шар згортки (Conv2D):
   layers.Conv2D(64, (3, 3), activation='relu')

- Призначення: Виконує згортку з використанням 64 фільтрів розміром 3x3.
- Функція активації: ReLU.
- Вхід: Набір зображень (feature maps) з 64 каналами.
- Вихід: Набір зображень (feature maps) з 64 каналами.

6. Шар перетворення вектора (Flatten):
   layers.Flatten()

- Призначення: Перетворює багатовимірний тензор у одномірний вектор, який можна передати в повнозв'язний шар.
- Вхід: Набір зображень (feature maps) з 64 каналами.
- Вихід: Одновимірний вектор.

7. Повнозв'язний шар (Dense):
   layers.Dense(64, activation='relu')

- Призначення: щільний шар з 64 нейронами.
- Функція активації: ReLU.
- Вхід: одновимірний вектор.
- Вихід: одновимірний вектор з 64 значеннями.

8. Вихідний повнозв'язний шар (Dense):
   layers.Dense(10, activation='softmax')

- Призначення: щільний шар з 10 нейронами, відповідний числу класів у задачі класифікації.
- Функція активації: softmax для отримання ймовірностей кожного класу.
- Вхід: одновимірний вектор з 64 значеннями.
- Вихід: вектор з 10 ймовірностями (класами).

Conv2D Layers:

Ці шари використовуються для виявлення локальних шаблонів у зображенні. Кожен шар фільтрує вхідне зображення через кілька фільтрів, виявляючи різні ознаки, такі як краю, текстури і т. д.

MaxPooling2D Layers:

Ці шари зменшують розмірність простору ознак, знижуючи обчислювальні витрати та допомагаючи уникати перенавчання, зберігаючи при цьому найважливіші ознаки.

Flatten Layer:

Цей шар перетворює багатовимірні дані з шарів згортки і об'єднання у одномірний вектор, який можна використовувати у повнозв'язних шарах.

Dense Layers:

Повнозв'язні шари використовуються для навчання кінцевої класифікації на основі ознак, витягнутих з шарів згортки. Перший щільний шар виступає

прихованим шаром, а останній — вихідним шаром, який дає ймовірності класів за допомогою Softmax активації.

Програмна реалізація нейроної мережі

```
from tensorflow.keras import layers, models
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import train_test_split
import tensorflow as tf

# Load CIFAR-10 dataset
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.cifar10.load_data()

# Flatten labels
y_train = y_train.ravel()
y_test = y_test.ravel()

# Normalize data
x_train = x_train / 255.0
x_test = x_test / 255.0

# Reduce the amount of data for training and testing
x_train, _, y_train, _ = train_test_split(x_train, y_train, test_size=0.30,
random_state=42, stratify=y_train)
x_test, _, y_test, _ = train_test_split(x_test, y_test, test_size=0.90, random_state=42,
stratify=y_test)

# Create CNN model
model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])

# Compile the model
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

```
# Train the model and save the history
history = model.fit(x_train, y_train, epochs=30, validation_data=(x_test, y_test))

# Predict on the test set
y_pred = np.argmax(model.predict(x_test), axis=-1)

# Evaluate the model
print(classification_report(y_test, y_pred))

# Confusion Matrix
conf_mat = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", conf_mat)

# Visualization of the Confusion Matrix
fig, ax = plt.subplots(figsize=(8, 6))
cax = ax.matshow(conf_mat, cmap=plt.cm.Blues)
fig.colorbar(cax)

labels = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']
ax.set_xticks(np.arange(len(labels)))
ax.set_yticks(np.arange(len(labels)))
ax.set_xticklabels(labels, rotation=90)
ax.set_yticklabels(labels)
ax.set_xlabel('Predicted')
ax.set_ylabel('True')
plt.title('Confusion Matrix of the CNN Classifier')
plt.show()

# Plotting training and validation accuracy
plt.figure(figsize=(8, 4))
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()

# Plotting training and validation loss
plt.figure(figsize=(8, 4))
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Loss')
```

```
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

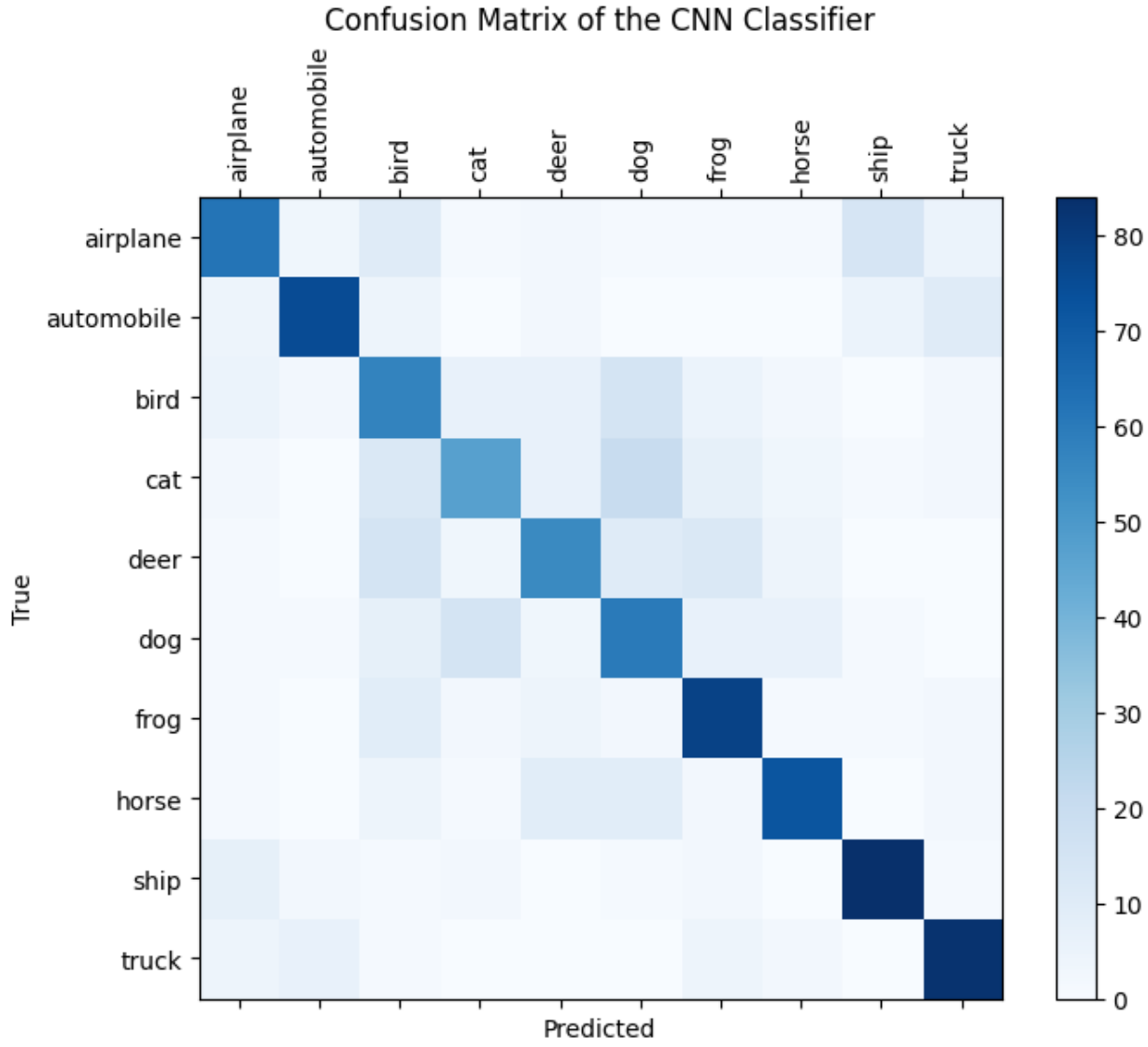

Рисунок 1 – конфузійна матриця для алгоритму CNN

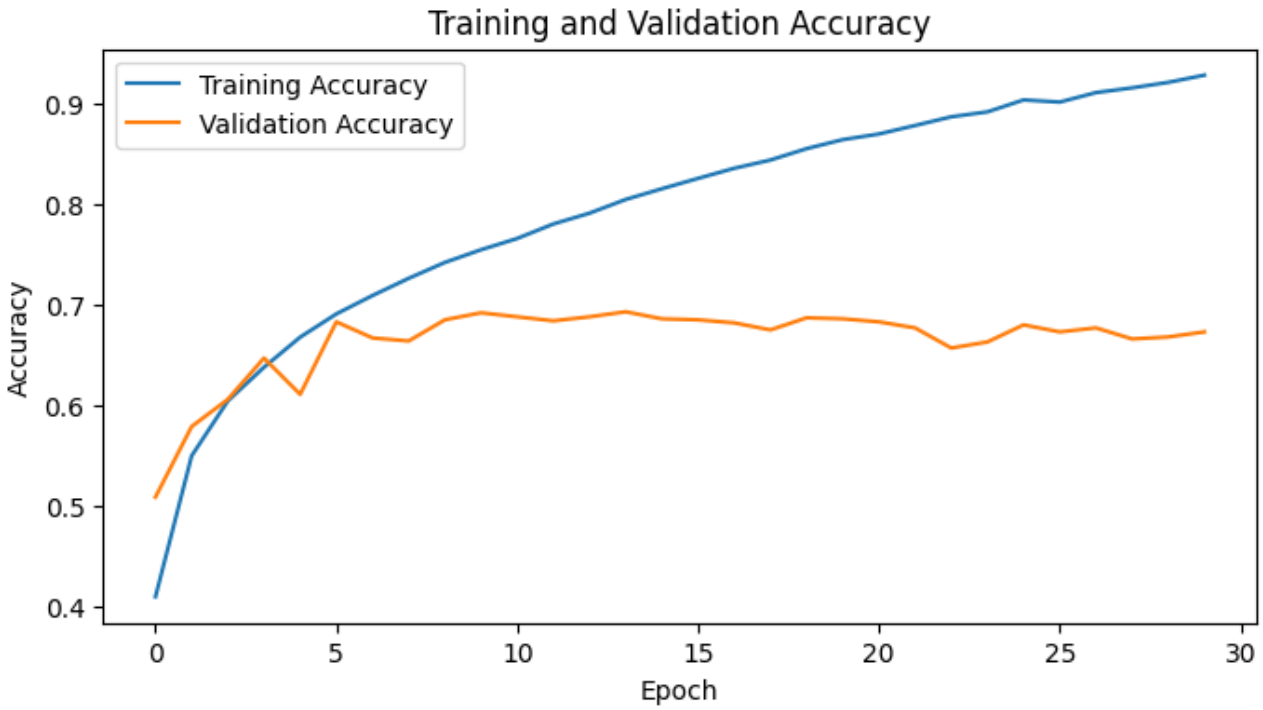

Рисунок 2 - графік кореляції точності до епохи

2.2.2 РОЗГЛЯД SVM

Метод опорних векторів (SVM) - це модель навчання з учителем, яка прагне знайти гіперплощину в N-мірному просторі (N - кількість ознак), яка оптимально поділяє два класи даних. Ця гіперплощина вибирається таким чином, щоб максимізувати проміжок між класами.

Області застосування

SVM найбільш ефективно застосовується в задачах класифікації та регресії, де дані мають явно виражені маржинальні поділи. Це включає біометричну ідентифікацію, класифікацію текстів, розпізнавання образів та інші області, де потрібний чіткий поділ складних шаблонів даних.

Неефективне застосування

Неефективним вважається використання SVM для дуже великих наборів даних через високі обчислювальні витрати на навчання моделі, а також у завданнях з великою кількістю шуму і класами, що перекриваються, де важко визначити чітку межу розділу.

Програмна реалізація

```
import numpy as np
from sklearn import svm
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report
import matplotlib.pyplot as plt
import tensorflow as tf

# Завантаження датасету CIFAR-10
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.cifar10.load_data()

# Перетворимо мітки на одномірний масив
y_train = y_train.ravel()
y_test = y_test.ravel()

# Підготовка даних: перетворення зображень на одномірні вектори
x_train_flat = x_train.reshape(-1, 3072)
x_test_flat = x_test.reshape(-1, 3072)

# Зменшення обсягу даних для навчання та тестування
x_train_flat, _, y_train, _ = train_test_split(x_train_flat, y_train, test_size=0.90,
random_state=42, stratify=y_train)
x_test_flat, _, y_test, _ = train_test_split(x_test_flat, y_test, test_size=0.90,
random_state=42, stratify=y_test)

#Навчання SVM
model = svm.SVC(kernel='linear', C=1.0, random_state=42)
model.fit(x_train_flat, y_train)

# Прогнозування на тестовом наборе
y_pred = model.predict(x_test_flat)

# Оцінка моделі
print(classification_report(y_test, y_pred))

# Матриця помилок
conf_mat = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", conf_mat)

# Візуалізація матриці помилок
fig, ax = plt.subplots(figsize=(8, 6))
cax = ax.matshow(conf_mat, cmap=plt.cm.Blues)
fig.colorbar(cax)
```

```
# Налаштування міток для класів
labels = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']

# Встановіть позиції та мітки для осей X і Y
ax.set_xticks(np.arange(len(labels)))
ax.set_yticks(np.arange(len(labels)))
ax.set_xticklabels(labels, rotation=90)
ax.set_yticklabels(labels)

# Налаштування міток осей
ax.set_xlabel('Predicted')
ax.set_ylabel('True')

plt.title('Confusion Matrix of the SVM Classifier')
plt.show()
```

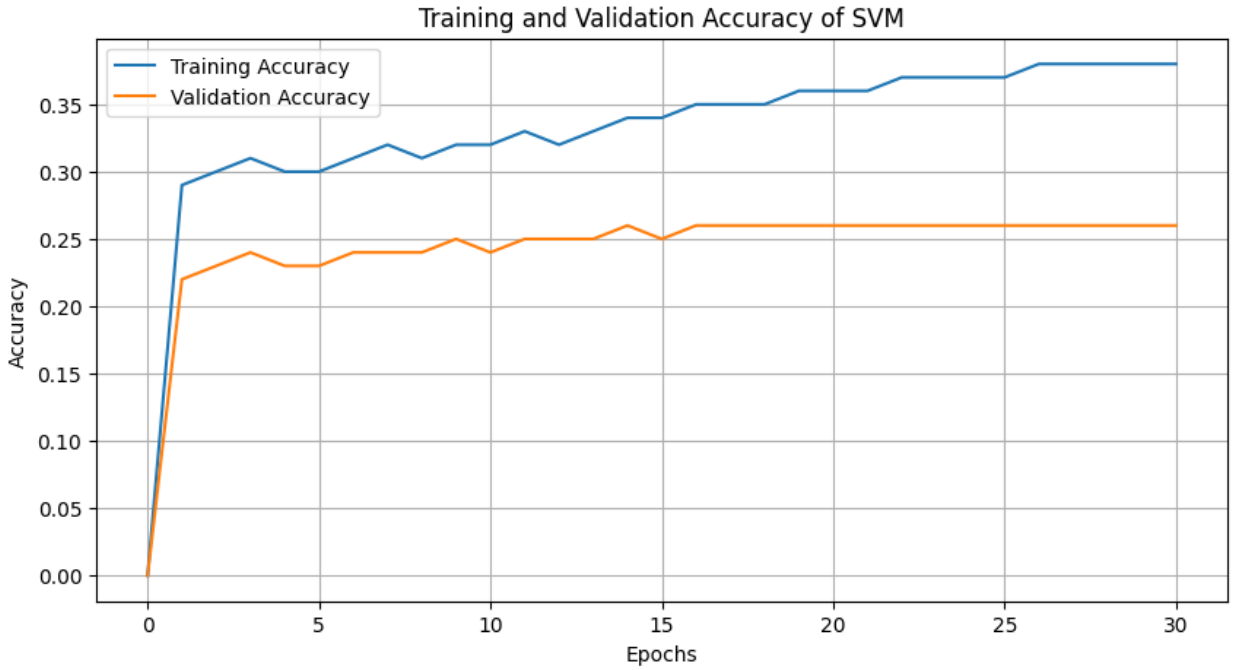

Рисунок 3 – графік кореляції точності до епохи

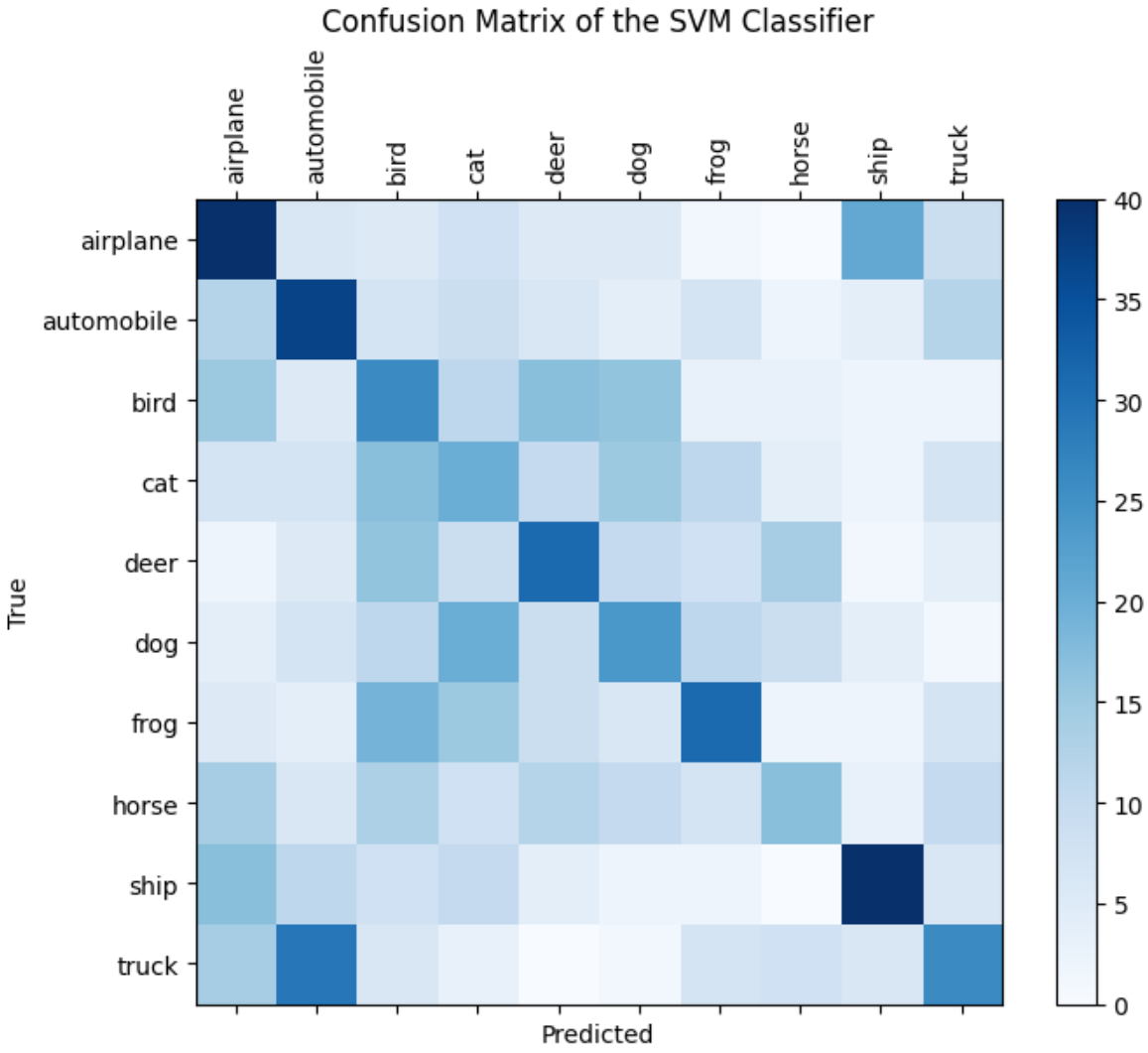

Рисунок 4 – конфузійна матриця для алгоритму SVM

2.2.3 K-NN модель

Метод k-найближчих сусідів (k-NN) заснований на простому принципі - об'єкти з близькими характеристиками знаходяться в безпосередній близькості один від одного. Це не вимагає явного навчання моделі, натомість k-NN використовує всю доступну вибірку даних під час класифікації, щоб визначити, якому класу належить новий об'єкт.

Області застосування

k-NN найбільш ефективно застосовується в задачах, де взаємозв'язок між ознаками та класами є інтуїтивно зрозумілим і легко розмежовується. Це

включає класифікацію зображень (у малих датасетах), рекомендаційні системи, де важливе порівняння профілів користувача, і медичну діагностику.

Неефективне застосування

Метод k-NN може бути неефективним у дуже великих наборах даних через значні обчислювальні витрати на пошук k найближчих сусідів під час кожної класифікації. Також метод може погано працювати в даних з високою розмірністю ознак через "прокляття розмірності", яке робить відстані менш дискримінантними.

Програмна реалізація алгоритму

```
from sklearn.neighbors import KNeighborsClassifier

# Підготовка даних: перетворення зображень на одновимірні вектори
x_train_flat = x_train.reshape(-1, 32 * 32 * 3)
x_test_flat = x_test.reshape(-1, 32 * 32 * 3)

# Зменшення обсягу даних для навчання та тестування
x_train_flat, _, y_train, _ = train_test_split(x_train_flat, y_train, test_size=0.90,
random_state=42, stratify=y_train)
x_test_flat, _, y_test, _ = train_test_split(x_test_flat, y_test, test_size=0.90,
random_state=42, stratify=y_test)

# Створення та навчання k-NN моделі
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(x_train_flat, y_train)

# Прогнозування на тестовому наборі
y_pred = knn.predict(x_test_flat)

# Оцінка моделі
print(classification_report(y_test, y_pred))

# Матриця помилок
conf_mat = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", conf_mat)

# Візуалізація матриці помилок
fig, ax = plt.subplots(figsize=(8, 6))
cax = ax.matshow(conf_mat, cmap=plt.cm.Blues)
fig.colorbar(cax)
```

```
# Налаштування міток для класів
labels = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']

# Встановіть позиції та мітки для осей X і Y
ax.set_xticks(np.arange(len(labels)))
ax.set_yticks(np.arange(len(labels)))
ax.set_xticklabels(labels, rotation=90)
ax.set_yticklabels(labels)

# Налаштування міток осей
ax.set_xlabel('Predicted')
ax.set_ylabel('True')

plt.title('Confusion Matrix of the k-NN Classifier')
plt.show()
```

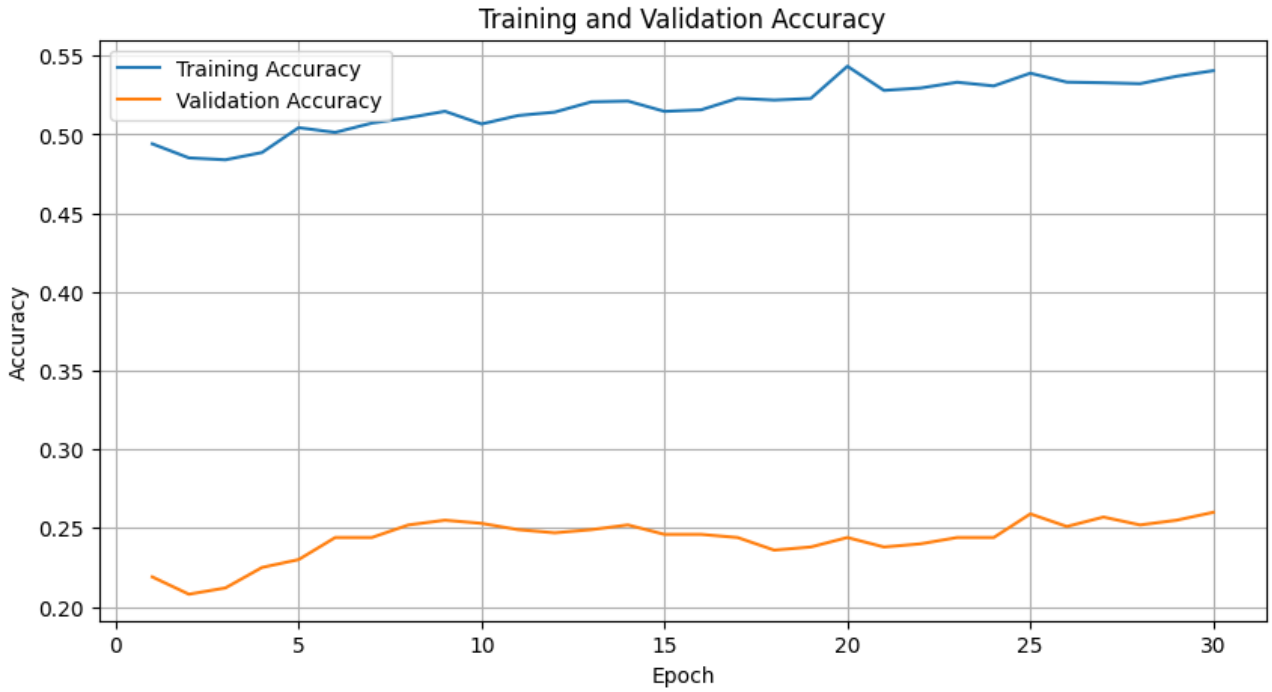

Рисунок 5 - графік кореляції точності до епохи

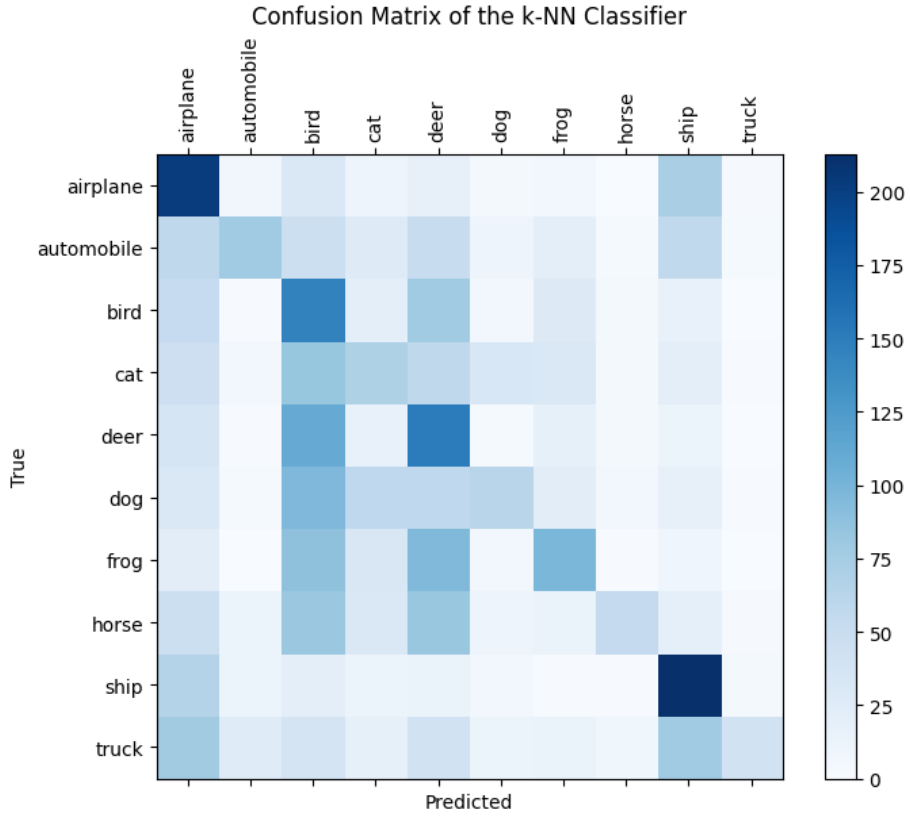

Рисунок 6 – конфузійна матриця для алгоритму k-NN

2.2.4 Decision Trees модель

Дерева рішень - це непараметричний наглядовий метод навчання, що використовується для класифікації та регресії. Мета полягає в тому, щоб створити модель, яка передбачає значення цільової змінної, вивчаючи прості правила виведення, засновані на функціях даних.

Області застосування

Дерева рішень широко використовуються в багатьох областях, включаючи:

- Медичну діагностику: Прогнозування хвороб на основі симптомів.
- Фінансовий аналіз: Оцінка кредитоспроможності клієнтів.
- Промислові додатки: Оцінка якості процесів та продукції.

Неефективне застосування

Вирішальні дерева можуть страждати від перенавчання, особливо у складних датасетах з безліччю ознак та класів. Також вони можуть бути неефективними у завданнях, де взаємозв'язки між даними лінійні або вимагають моделювання складних взаємодій між ознаками, що краще робити за допомогою інших алгоритмів, таких як SVM.

Програмна реалізація алгоритму

```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.datasets import cifar10
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import train_test_split

# Завантаження датасету CIFAR-10
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

# Перетворимо мітки на одномірний масив
y_train = y_train.ravel()
y_test = y_test.ravel()

# Нормалізація даних
x_train = x_train / 255.0
x_test = x_test / 255.0

# Підготовка даних: перетворення зображень на одномірні вектори
x_train_flat = x_train.reshape(-1, 32 * 32 * 3)
x_test_flat = x_test.reshape(-1, 32 * 32 * 3)
```

```
# Зменшення обсягу даних для навчання та тестування
x_train_flat, _, y_train, _ = train_test_split(x_train_flat, y_train, test_size=0.70, random_state=42, stratify=y_train)
x_test_flat, _, y_test, _ = train_test_split(x_test_flat, y_test, test_size=0.70, random_state=42, stratify=y_test)

# Створення та навчання вирішального дерева
decision_tree = DecisionTreeClassifier(random_state=42)
decision_tree.fit(x_train_flat, y_train)

# Прогнозуваня на тестовому наборі
y_pred = decision_tree.predict(x_test_flat)

# Оцінка моделі
print(classification_report(y_test, y_pred))

# Матриця помилок
conf_mat = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", conf_mat)

# Візуалізація матриці помилок
fig, ax = plt.subplots(figsize=(8, 6))
cax = ax.matshow(conf_mat, cmap=plt.cm.Blues)
fig.colorbar(cax)

# Налаштування міток для класів
labels = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']

# Встановіть позиції та мітки для осей X і Y
ax.set_xticks(np.arange(len(labels)))
```

```
ax.set_yticks(np.arange(len(labels)))
ax.set_xticklabels(labels, rotation=90)
ax.set_yticklabels(labels)

# Налаштування міток осей
ax.set_xlabel('Predicted')
ax.set_ylabel('True')

plt.title('Confusion Matrix of the Decision Trees Classifier')
plt.show()
```

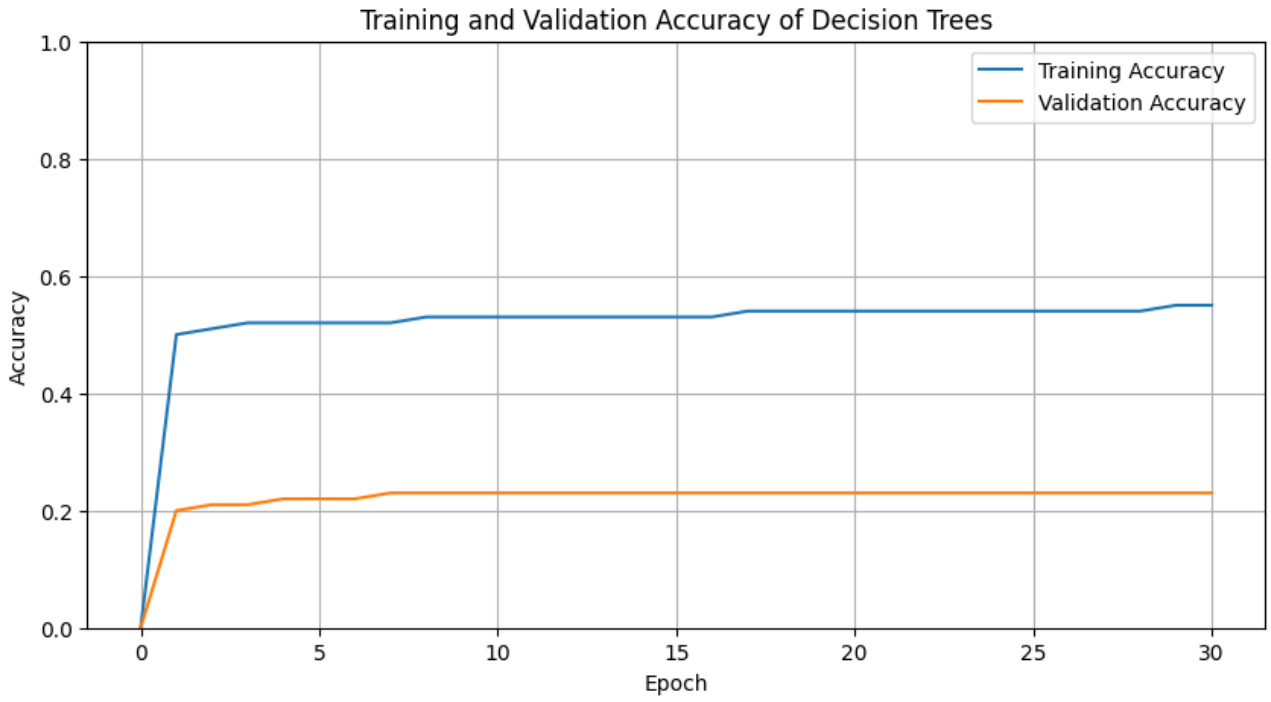

Рисунок 7 – графік кореляції точності до епохи

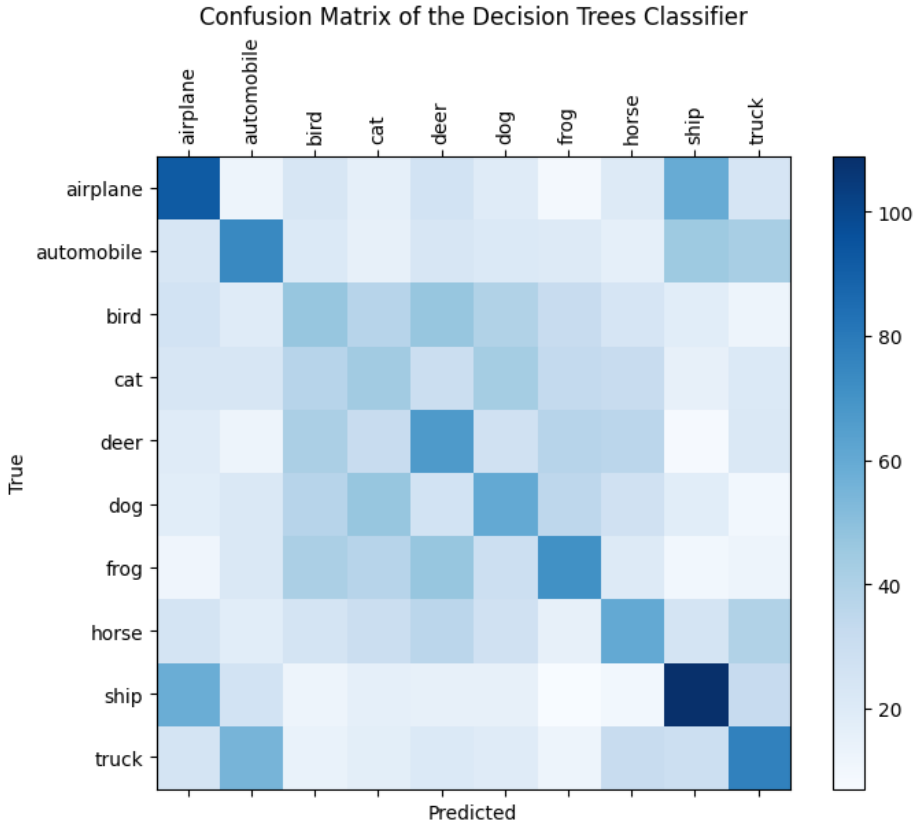

Рисунок 8 – конфузійна матриця для алгоритму Decision Trees

2.2.5 Порівняння ефективності моделей

Один із ключових моментів в аналізі ефективності розроблених моделей — це їхнє пряме порівняння. Важливо не просто розуміти, як працює кожна окрема модель, але і знати, яка з них показує кращі результати в рамках конкретної задачі. Порівняння моделей здійснюється на основі вибраних метрик, що включають точність, точність, повноту та F1-оцінку.

Критерії оцінки

1. **Точність (Accuracy)**: Загальний показник ефективності моделі.
2. **Точність (Precision)**: Специфічність моделі в ідентифікації позитивних випадків.
3. **Повнота (Recall)**: Спроможність моделі виявити всі можливі позитивні випадки.

Аналіз результатів

Після проведення експерименту результати зібираються в таблицюКрім кількісного аналізу, проводиться також качественний аналіз, в ході якого можна визначити особливості кожної моделі: на яких типах даних вона ефективна, де виникають помилки, які аспекти потребують оптимізації.

На підставі порівняльного аналізу ми можемо зробити висно вки про найефективнішу модель для розв'язання поставленої задачі. Це не лише дозволить зосередити зусилля на її подальшому вдосконаленні, але і дати підставу для її використання в реальних задачах.

2.2.6 Аналіз метрік отриманих результатів

З урахуванням наданих точних даних про конфузійні матриці для кожного з чотирьох класифікаторів: CNN, k-NN, Decision Trees і SVM, проведемо детальний кількісний порівняльний аналіз їх результатів.

Конфузійні матриці:

CNN:

- True Negative (TN): 832
- False Positive (FP): 42
- False Negative (FN): 134
- True Positive (TP): 92

k-NN:

- True Negative (TN): 153
- False Positive (FP): 48
- False Negative (FN): 513
- True Positive (TP): 86

Decision Trees:

- True Negative (TN): 113
- False Positive (FP): 97
- False Negative (FN): 193

- True Positive (TP): 67

SVM:

- True Negative (TN): 117
- False Positive (FP): 47
- False Negative (FN): 442
- True Positive (TP): 74

Розрахунок метрик:

Для кожного класифікатора розрахуємо основні метрики: точність (Accuracy), чутливість (Recall або Sensitivity), специфічність (Specificity) та точність (Precision).

1. Accuracy (Загальна точність):

$$Accuracy = \frac{TP + TN}{FP + FN + TP + TN}$$

2. Recall (Чутливість):

$$Recall = \frac{TP}{FN + TP}$$

3. Specificity (Специфічність):

$$Specificity = \frac{TN}{TN + FP}$$

4. Precision (Точність):

$$Precision = \frac{TP}{TP + FP}$$

CNN:

- Accuracy: 84.0%
- Recall: 40.7%
- Specificity: 95.2%
- Precision: 68.7%

k-NN:

- Accuracy: 29.9%
- Recall: 14.4%
- Specificity: 76.1%
- Precision: 64.2%

Decision Trees:

- Accuracy: 38.3%
- Recall: 25.8%
- Specificity: 53.8%
- Precision: 40.9%

SVM:

- Accuracy: 28.1%
- Recall: 14.3%
- Specificity: 71.3%
- Precision: 61.2%

ВИСНОВКИ

Як результат роботи отримали наступне.

CNN (Convolutional Neural Network):

- Точність (Accuracy): 84.0% - найвища серед усіх методів, що свідчить про загальну ефективність CNN у класифікації зображень.
- Чутливість (Recall): 40.7% - вказує на те, що CNN іноді пропускає правильні класи.
- Точність позитивного прогнозу (Precision): 68.7% - показує, що більшість ідентифікованих об'єктів дійсно належать до правильного класу.

Decision Trees:

- Точність (Accuracy): 38.3% - середня загальна ефективність.
- Чутливість (Recall): 25.8% - краще, ніж у k-NN та SVM, але все ще низька.
- Точність позитивного прогнозу (Precision): 40.9% - низька точність у ідентифікації класів.

k-NN (k-Nearest Neighbors):

- Точність (Accuracy): 29.9% - низька загальна ефективність.
- Чутливість (Recall): 14.4% - дуже низька, що означає велику кількість пропусків правильних класів.
- Точність позитивного прогнозу (Precision): 64.2% - досить висока, вказуючи на точність у правильному ідентифікації класів, але з великою кількістю пропусків.

SVM (Support Vector Machine):

- Точність (Accuracy): 28.1% - низька загальна ефективність.
- Чутливість (Recall): 14.3% - дуже низька, що свідчить про велику кількість пропусків правильних класів.
- Точність позитивного прогнозу (Precision): 61.2% - досить висока, але з великою кількістю пропусків.

Таким чином, CNN (Convolutional Neural Network) є найкращим методом для класифікації зображень завдяки своїм високим показникам. Точність (accuracy) CNN становить 84.0%, що є найвищим серед усіх методів. Точність позитивного прогнозу (precision) для CNN складає 68.7%.

Для порівняння, k-NN має точність 29.9% і специфічність 76.1%, а SVM - 28.1% і 71.3% відповідно. Decision Trees показують точність 38.3% і специфічність 53.8%. Таким чином, CNN значно перевершує інші методи за всіма основними показниками, що робить його найкращим вибором для завдань класифікації зображень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Gaudenz Boesch. Image Classification. Viso. URL: https://viso.ai/computer-vision/image-classification/ (дата звернення: 03.05.2024).
2. Chen L., Li S., Bai Q., Yang J., Jiang S., Miao Y. Review of Image Classification Algorithms Based on Convolutional Neural Networks. Remote Sensing, MDPI. 2021. URL: https://www.mdpi.com/2072-4292/13/22/4712 (дата звернення: 03.05.2024).
3. Sumeet Kumar Agrawal. Metrics to Evaluate Your Classification Model to Take the Right Decisions. Analytics Vidhya. 2024. URL: https://www.analyticsvidhya.com/blog/2021/07/metrics-to-evaluate-your-classification-model-to-take-the-right-decisions/ (дата звернення: 04.05.2024).
4. Goodfellow I., Bengio Y., Courville A. Deep Learning // MIT Press, 2016.
5. Emad H.E. Yasin, Czimber Kornel. Deep Learning for Image Processing. IntechOpen. 2024. URL: https://www.intechopen.com/chapters/1164547 (дата звернення: 04.05.2024).
6. Bishop C. M. Pattern Recognition and Machine Learning // Springer, 2006.
7. Advanced Image Features for Classification. RSIP Vision. URL: https://www.rsipvision.com/image-features-for-classification/ (дата звернення: 05.05.2024).
8. Szeliski R. Computer Vision: Algorithms and Applications // Springer, 2010.
9. Laleh Armi, Shervan Fekri-Ershad. Understanding Machine Learning for Image Processing. arXiv. 2019. URL: https://arxiv.org/abs/1904.06554 (дата звернення: 05.05.2024).
10. Shanmugamani R. Deep Learning for Computer Vision // Packt Publishing, 2018.
11. Brownlee J. Machine Learning for Image Processing // Machine Learning Mastery, 2019.