Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

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

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

ДОСЛІДЖЕННЯ МЕТОДІВ МАШИННОГО НАВЧАННЯ ДЛЯ ВИРІШЕННЯ ЗАДАЧІ КЛАСИФІКАЦІЇ ЩОДО РОЗПІЗНАВАННЯ ГРАФІЧНИХ ЗОБРАЖЕНЬ
(тема)

Виконав:
студент 2 курсу, групи ІНФм-22-1

Афанасьєв А.С.
(прізвище, ініціали)

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

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Кобилін О.А.
(посада, прізвище, ініціали)

Допускається до захисту

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

2024 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти другий (магістерський)

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

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Афанасьєву Анатолію Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження методів машинного навчання для вирішення задачі класифікації щодо розпізнавання графічних зображень

затверджена наказом по університету від 3 листопада 2023 року № 1280Ст

2. Термін подання студентом роботи до екзаменаційної комісії 01 січня 2023 року.

3. Вихідні дані до роботи методи машиного навчання, моделі розпізнавання зображень, моделі нейроних мереж, машина опорних векторів, багатошаровий персептрон, тестові зображення, перелік програмних засобів реалізації класфікатора

4. Перелік питань, що потрібно опрацювати в роботі
1. Проаналізувати існуючи методи класифікації даних.
2. Порівняти моделі нейроних мереж для розпізнавання зображень.
3. Дослідити математичні моделі.
4. Розробити застосунок для дослідження задачі класифікації зображення.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність проблеми розпізнавання зображень, об'єкт, мета та постановка задачі для дослідження, тестові зображення та приклади перетворення зразків, перспективи подальших дослідження, висновки, апробація результатів роботи.

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

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

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	03.11.2023	
2	Аналіз завдання, підбір літератури	04.11.23-14.11.23	
3	Аналіз літератури з досліджуваної проблеми	15.11.23-25.11.23	
4	Аналіз методів розпізнавання зображень	26.11.23-30.11.23	
5	Дослідження методів машиного навчання	01.12.23-14.12.23	
6	Програмна реалізація	15.12.23-21.12.23	
7	Оформлення пояснювальної записки	22.12.23-30.12.23	
8	Перевірка на плагіат	31.12.2023	
9	Рецензування	02.12.2023	
10	Підготовка презентації та доповіді	08.01.2023	
11	Занесення роботи в електронний архів	09.01.2024	
12	Попередній захист кваліфікаційної роботи	10.01.2024	

Дата видачі завдання 3 листопада 2023 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Кобилін О.А.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 61 с., 1 табл., 13 рис., 38 джерел.

ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ, РОЗПІЗНАВАННЯ ОБРАЗІВ, КОМПЮТЕРНИЙ ЗІР, НЕЙРОННІ МЕРЕЖІ, МАШИНА ОПОРНИХ ВЕКТОРІВ, КЛАСИФІКАЦІЯ.

Об’єктом дослідження є методи класифікації щодо розпізнавання зображень.

Метою дослідження є реалізація класифікатора графічних зображень з урахуванням обраних алгоритмів заснованих на багатошаровому персептроні та методі опорних векторів.

У ході виконання кваліфікаційноії роботи проведено експеримент і зясовано що обидва алгоритми показали точність більше 80%, але машина опорних векторів була ефективнішою на 6%. Також, розроблено застосунок, який реалізує рішення задачі класифікації графічних зображень.

Даний застосунок може використовуватися у системах технічного зору.

DATA MINING, IMAGE RECOGNITION, COMPUTER VISION, NEURAL NETWORKS, SUPPORT VECTOR MACHINE, CLASSIFICATION.

The object of research is classification methods for image recognition.

The purpose of the study is to implement a classifier of graphic images, taking into account the selected algorithms based on multilayer perceptron and support vector machine.

During the qualification work, an experiment was conducted and it was found that both algorithms showed an accuracy of more than 80%, but the support vector machine was 6% more efficient. Also, an application was developed that implements the solution to the problem of graphic image classification.

This application can be used in technical vision systems.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

СТЗ – система технічного зору

МОВ – машина опорних векторів

МН – машине навчання

HOG – Histograms of Oriented Gradients

SIFT – Scale Invariant Feature Transform

ВСТУП

В сучасному світі все більшого значення набуває використання інформаційних технологій. Людина намагається автоматизувати всі аспекти своєї діяльності. На сьогоднішній день люди все активніше намагаються делегувати різні обов’язки машинам [1]. Вже нікого не дивує наявність різноманітних «розумних помічників» як на виробництві, так і в повсякденному житті. Наприклад, Cortana від Microsoft та Siri від Apple, володіють високорозвиненими алгоритмами розпізнавання мови і можуть виконувати завдання, які ставляться до них у розмовному стилі. Іншими словами, для їх використання вже не потрібно вказувати конкретні команди - досить просто спілкуватися з ними, подавати питання чи просити про допомогу і вони розпізнають, чого ви хочете, видаючи очікуваний результат. Очевидно, що багато помічників та систем прийняття рішень з’явилися в різних сферах людської діяльності де невірне або некоректне вирішення завдань коштує дорого. Наприклад, в медицині вони допомагають ставити правильний діагноз (комп’ютерний томограф), розпізнавати хвороби та призначати лікування на підставі великого аналізу різноманітних даних. Машини справляються з цим краще за людей, оскільки вони не мають таких «людських» факторів, як втомленість чи неуважність. Проте наразі алгоритми машинного навчання ще не настільки досконалі, щоб їх можна було використовувати без нагляду фахівця в галузі. Велика кількість подібних помічників з’явилася також у військовій справі, де багато розробок спрямовані на мінімізацію втрат людських ресурсів. Таким чином, виникають різноманітні дрони, безпілотники та роботи. В літаках встановлюється штучний інтелект, що допомагає пілотові у критичних ситуаціях уникнути катастрофи. Можливості цієї технології дійсно вражають тому актуальною цю тему можна назвати не вагаючись.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

Алгоритми навчання успішно застосовуються у великій кількості різних додатків, включаючи:

– класифікацію текстів чи документів, наприклад, визначення спаму;

– обробку природної мови, наприклад, морфологічний аналіз, угруповання за членами мови, статичний аналіз;

– розпізнавання мови, синтез мови, розпізнавання людини за голосом;

– оптичне розпізнавання символів;

– завдання комп'ютерного зору, наприклад, розпізнавання зображень та осіб;

– розпізнавання шахрайства, наприклад з кредитними картками, та мережне (інтернет) втручання;

– ігри (шахи, нарди та інші);

– автономне керування машинами, навігація та переміщення роботів;

– медична діагностика;

– системи рекомендацій, пошукові движки, системи вилучення інформації.

Даний список не є вичерпним, навчальні алгоритми застосовуються у нових додатках практично щодня. Більше того, такі додатки відносяться до безлічі проблем навчання.

1.1 Огляд завдань сучасного машинного навчання

Машинне навчання розташоване на перехресті прикладної статистики, чисельних методів оптимізації та дискретного аналізу і протягом останніх 50 років воно розвинулося у самостійну математичну галузь. Означення машинного навчання може звучати як обчислювальні методи, що

використовують накопичений досвід для поліпшення продуктивності або точного передбачення [2]. У цьому контексті «досвід» вказує на інформацію, яка зазвичай представлена у вигляді електронних даних, що зібрані та доступні для аналізу. Ці дані можуть бути в формі цифрових тренувальних вибірок або інформації, зібраної через взаємодію з оточенням. Незалежно від форми, якість та обсяг аналізованих даних є надзвичайно важливими умовами для успішного передбачення алгоритмів машинного навчання. Оскільки ефективність цих алгоритмів залежить від використовуваних даних, машинне навчання тісно пов’язане із аналізом даних та статистикою. Загалом, техніки машинного навчання є методами обробки даних, які вбирають у себе основні концепції інформатики, а також ідеї з областей статистики, теорії ймовірностей та оптимізації. Машинне навчання вирішує низку основних прикладних завдань.

Поділяють два основні підходи машинного навчання: навчання з вчителем та навчання без вчителя [1, 22-24].

Навчання без вчителя – це галузь машинного навчання, що досліджує широкий спектр завдань обробки даних. У таких задачах відомо лише опис множини об’єктів навчальної вибірки, мета полягає у виявленні внутрішніх зв’язків, залежностей та закономірностей між об’єктами. Один із найвідоміших прикладів – задачі кластеризації.

Кластеризація включає в себе групування об’єктів у кластери за допомогою інформації про схожість пар об’єктів. Функціонали якості можуть бути визначені по-різному, наприклад, враховуючи відношення середніх відстаней між кластерами та внутрішніми відстанями. Прикладом такої задачі може бути визначення класів ссавців на основі фізичних параметрів, таких як довжина тіла та вага. Алгоритм шукає подібні екземпляри та розподіляє їх за класами залежно від ступеня схожості.

Навчання з вчителем – це тип алгоритмів машинного навчання, які використовують попередньо визначені вибірки даних, відомі як тренувальні,

для здійснення прогнозів. Тренувальні дані включають в себе вхідні параметри та відповідні значення.

На основі цього навчання з вчителем спрямоване на побудову моделі, яка може передбачати значення нових даних. Тестова вибірка використовується для перевірки моделі, і використання великих тренувальних вибірок допомагає моделям краще передбачати значення нових даних і одночасно залишатися більш загальними.

У навчанні з вчителем виділяються такі типи завдань, як класифікація, регресія, прогнозування та ранжування.

Завдання регресії – це пошук та передбачення реальних значень, ґрунтуючись на попередньому досвіді. Наприклад, передбачення цінових показників на фондовому ринку на основі даних попередніх днів або вартості нерухомості з аналізу ринку. У випадку завдань ранжування особливість полягає в тому, що відповіді потрібно одержати одразу для численних об’єктів та подальше їх сортування за значеннями відповідей. Це може включати в себе завдання класифікації чи регресії та часто використовується в інформаційному пошуку і текстовому аналізі. Такі завдання вимагають обробки великих обсягів даних, які потрібно впорядковувати, щоб отримати вибірку найбільш релевантних та корисних даних. Прикладами можуть бути документи та запитання до пошукових систем. Компанії, такі як Google та bing, вирішують ці завдання, шукаючи оптимальні алгоритми. Завдання прогнозування відрізняються тим, що об’єктами є відрізки часових рядів, які обриваються у той момент, коли необхідно зробити прогноз на майбутнє. Для їх вирішення часто використовують методи регресії або класифікації, і у другому випадку йдеться, скоріше, про завдання прийняття рішень.

1.2 Уявлення про класифікацію

Класифікація – один із розділів машинного навчання, присвячений вирішенню наступного завдання: є безліч об’єктів, розділених на класи. Задано кінцеве безліч об’єктів котрим відомо до яких класів вони відносяться [2, 25-27]. Ця множина називається навчальною вибіркою. Класова приналежність інших об’єктів невідома. Потрібно побудувати алгоритм, здатний класифікувати довільний об’єкт із вихідної множини.

Класифікація – задача машинного навчання, що найчастіше зустрічається. Класифікатором називається відображення представлене на формулі:

$$\hat{c}: X \to C, C = \{C_1, C_2, \ldots, C_n\}, \tag{1.1}$$

де $\hat{c}$ – функція класифікатора;

X – множина об’єктів навчальної вибірки;

C – кінцеве та зазвичай невелика множина міток класів.

Знак $\hat{}$ означає, що $\hat{c}(x)$ – оцінка істиною, але невідомої функції *c(x)*.

$$(x, c(x)), x \in X, \tag{1.2}$$

де x – об’єкт;

$c(x)$ – істинний клас об’єктів x.

Під навчанням класифікатора розуміється створення функції $\hat{c}$. Вона, як найкраще, наближається до вихідної функції (і не лише на навчальному наборі, але, в ідеальному випадку – на всьому просторі об’єктів). У простому випадку є всього два класи, які зазвичай називають позитивним і негативним. Двокласову класифікацію часто також називають бінарною класифікацією або концептуальним навчанням, а позитивний клас можна з належним підставами назвати концепцією. Фільтрація спаму – приклад бінарної

класифікації, в якому спам традиційно вважається позитивним класом, а неспам – негативним (очевидно, що слово «позитивний» тут не означає «добрий»). Серед інших прикладів бінарної класифікації можна назвати медичну діагностику (позитивним класом тут є конкретне захворювання) і виявлення шахрайства з кредитними картками.

1.2.1 Оцінка якості класифікації

Оцінити якість класифікаторів можна за допомогою таблиці, яка називається таблицею спряженості або матрицею неточностей. Кожен рядок цієї таблиці відповідає фактичним класам з тестового набору, а кожний стовпець – класам, передбаченим класифікатором. У останньому стовпці та останньому рядку розташовані маргінали – сума елементів відповідного стовпця або рядка. Маргінали важливі, оскільки дозволяють оцінити статистичну значущість. Числа на головній діагоналі – правильні передбачення, а на побічній – помилки [30-35].

Таблиця 1.1 – Приклад двокласової матриці неточностей

Категорії	**Передбачено (позитивний)**	**Передбачено (негативний)**	**Підсумок**
Фактично позитивний	30	20	50
Фактично негативний	10	40	50
Загальний	40	60	100

Найпростішою метрикою оцінки роботи класифікатора може бути правильність (accuracy). Вона розраховується за такою формулою:

$$Accuracy = \frac{P}{N}, \tag{1.3}$$

де *Accuracy* – правільність;

P – кількість об’єктів за якими класифікатор прийняв правильне рішення;

N – розмір навчальної вибірки.

Дана метрика є найпростішою, але має одну неприємну особливість, яку варто враховувати. Дана особливість полягає в тому, що метрика надає однакову вагу всім об’єктам, що може бути некоректним у випадку сильно відхиленої вибірки, коли більшість об’єктів належать до одного класу, тобто розподіл спрямований на один або кілька класів. У цьому випадку класифікатор має більше інформації про класи з великою кількістю об’єктів і тому він прийматиме більш адекватні рішення лише для цих класів. На практиці це означає, що при такій вибірці точність (accuracy) складатиме приблизно 80 %, але при роботі з невідхиленими класами класифікатор буде працювати некоректно.

Рішенням цієї проблеми може бути вибір більш адекватної, рівномірно розподіленої навчальної вибірки (що не завжди можливо), або використання іншого підходу до оцінки якості.

Іншою оцінкою якості класифікатора можуть служити точність (precision) і повнота (recall). Метрики використовуються як самі по собі так і в більш вдосконалених метриках, таких як *F*–мера або *R*–Precision.

Точність (precision) в межах класу – це частка об’єктів, які дійсно належать даному класу відносно всіх документів та які класифікатор відніс до цього класу, обчислюється за формулою:

$$Precision = \frac{TP}{TP + FP}, \tag{1.4}$$

де *Precision*– точність класифікатора;

TP – істино-позитивне рішення;

FP – істино-негативне рішення.

Повнота (recall) системи – це частка, знайдених класифікатором об’єктів, що належать класу щодо всіх об’єктів даного класу у тестовій вибірці:

$$Recall = \frac{TP}{TP + FN}\ , \tag{1.5}$$

де *Recall* – повнота;

TP – істино-позитивне рішення;

FN – істино-негативне рішення.

Для того, щоб простіше вважати точність і повноту, на практиці використовують матрицю неточностей, якщо кількість класів відносно невелика, даний підхід дозволяє дуже наочно представити результат роботи класифікатора.

В даному випадку, маючи таку матрицю, точність та повнота розраховуються за формулами:

$$Prescision_c = \frac{A_{c,c}}{\sum_{i=1}^{n} A_{c,i}}\ , \tag{1.6}$$

$$Recall_c = \frac{A_{c,c}}{\sum_{i=1}^{n} A_{i,c}}, \tag{1.7}$$

де $Precision_c$ – точність за класом с;

A – матриця неточностей;

$Recall_c$ – повнота по класу с.

Результативність та повнота розраховуються як середнє арифметичне точності та повноти по всіх класах. Але на практиці неможливо досягти максимальної точності та повноти одночасно, вони недосяжні, тому завжди доводиться шукати певний баланс. Для більш простого вирішення цієї задачі варто скористатися *F*–мерою. Вона представляє собою гармонійне середнє між точністю та повнотою, що відкриває цікаву особливість: *F* –міра схильна до нуля, якщо точність або повнота схильні до нуля. *F* – міра розраховується за формулою:

$$F = (\beta^2 + 1)\frac{Precision \times Recall}{\beta^2 Precision + Recall}, \tag{1.8}$$

де F – значення F – міри;

β – ваговий коефіцієнт;

Precision– порахована точність класифікатора;

Recall– порахована повнота класифікатора.

Якщо необхідно, щоб точність і повнота відігравали однакову роль при розрахунку F – мери то ваговий коефіцієнт слід прирівняти до одиниці. У випадку коли ваговий коефіцієнт приймає значення від нуля до одиниці пріоритет буде схилений в бік точності, а коли ваговий коефіцієнт приймає значення більше одиниці – до повноти. Ця міра є хорошим кандидатом на формальну метрику оцінки якості класифікатора. Вона зводить до одного числа дві інші основоположні метрики і дає уявлення про те чи йдуть зміни в алгоритмі на користь чи ні.

1.3 Уявлення про комп’ютерний зір та розпізнавання зображень

Історія комп’ютерного зору розпочалася в 2001 році, коли з’явився перший ефективний алгоритм виявлення людського обличчя, розроблений

Полом Віолою та Міхаелем Джонсоном [4, 36-37]. Їх демо-версія показувала як обличчя людей виявлялися в реальному часі через потокове відео веб-камери за допомогою їх алгоритму це було найвражаючішою демонстрацією можливостей комп'ютерного зору та потенціалу, який можна було б розвинути в подальшому. Невдовзі після цього цей алгоритм був реалізований у пакеті OpenCV, і виявлення облич стало синонімом алгоритму Віоло і Джонсона. Кожні кілька років з'являються все нові ідеї які змушують людей зупинитися і задуматися про подальший розвиток технологій. У розпізнаванні об'єктів така ідея виникла в 2005 році разом із роботою Навнеета Далаля і Біла Тріггса. їх майбутній дескриптор «Гістограма напрямлених градієнтів» (Histograms of Oriented Gradients, HOG) значно перевершив існуючі тоді алгоритми виявлення пішоходів на дорозі.

Найближче десятиріччя постачає нам нові потужні та ефективні ідеї, які роблять все попереднє абсолютно застарілим та неефективним. Так, ідея глибокого навчання – це концепція поточного десятиріччя. Алгоритми глибокого навчання існували достатньо довго, але вони стали основними в завданнях комп'ютерного зору після успіху на виклику ImageNet Large Scale Visual Recognition у 2012 році. На цьому змаганні алгоритм глибокого навчання, розроблений Олексієм Кріжевським, Іллею Суцкевером і Джефрі Гінтоном, вражав світ експертів з комп'ютерного зору за дивовижною точністю на рівні вісімдесят п'ять відсотків, що на одинадцять відсотків перевищувало другий алгоритм. У 2012 році це був єдиний алгоритм, заснований на глибокому навчанні. У наступному році всі призери цього змагання використовували глибоке навчання і вже у 2015 році кілька алгоритмів, які використовують згорткові нейронні мережі (Convolution Neural Network), в завданні розпізнавання обличь, перевищили показник точності у дев'яносто п'ять відсотків [5,6, 28-29].

З таким вражаючим успіхом у розпізнаванні зображень використання алгоритмів, що ґрунтуються на глибокому навчанні було просто неодмінним фактом. Техніки, такі як Fast R-CNN, показували вражаючі результати в

завданнях розпізнавання об’єктів багатьох класів. Сьогодні практично неможливо побудувати вдалу програму для розпізнавання зображень без використання алгоритмів на основі глибокого навчання. Без їхньої участі програма може втратити можливість поліпшити свої результати в розпізнаванні.

Алгоритми розпізнавання зображень приймають зображення на вході і видають результат, що зображено на картинці. Іншими словами, результат – це клас-мітка (наприклад, «кіт», «людина», «автомобіль», та інше). Алгоритм розпізнає вміст зображення але для цього його треба навчити це робити. Потрібно навчити алгоритм розрізняти різні класи якщо потрібно шукати котів на картинках, то треба навчати алгоритм з тисячами зображень котів і тисячами зображень, на яких котів немає. Також, важливо пам’ятати що такий алгоритм зможе розпізнавати лише котів. У цьому випадку алгоритм є більш простим та його результатом є бінарна класифікація. Звісно, існують класифікатори, які можуть розрізняти кілька класів об’єктів на зображенні, але також важливо пам’ятати, що багато алгоритмів працюють саме з бінарною класифікацією і широко використовуються і сьогодні (наприклад, розпізнавання обличчя або виявлення пішоходів).

Більшість традиційних алгоритмів для класифікації зображень слідують цій схемі тоді як алгоритми на основі глибокого навчання повністю пропускають етап виділення особливостей (параметрів). На етапі попередньої обробки вхідне зображення часто вирівнюють та нормалізують контрастність, яскравість, застосовують різні кольорові фільтри. Досить поширеною практикою є отримання середньої яскравості зображення та поділ її на стандартне відхилення. Іноді гамма-корекція призводить до кращого результату. Працюючи з кольоровими зображеннями, трансформація кольорового простору може дати хороші результати. Але неможливо точно описати набір технік та їх послідовність, які давали б кращі результати, для кожного конкретного випадку ці етапи можуть відрізнятися. Тільки методом проб та помилок можна зрозуміти яка техніка підходить або

ні. Частиною попередньої обробки зображення також є масштабування та обрізка фрагмента зображення з метою отримання картинки з фіксованим розміром. Дані маніпуляції є важливими оскільки виділення параметрів виконується на зображеннях фіксованого розміру.

Наступним етапом є виділення параметрів, оскільки вихідне зображення містить занадто багато зайвої інформації, яка не потрібна для класифікації. Тому перш за все варто спростити зображення, виділивши лише важливу інформацію, що міститься на ньому, і видаливши все інше. Наприклад, якщо потрібно знайти ґудзик на зображенні сорочки, немає необхідності відстежувати всю колірну палітру та орієнтуватися на колір ґудзиків, оскільки вони можуть дуже сильно відрізнятися. У той же час, розпізнавши краї об’єктів, алгоритм все ще зможе успішно розпізнавати ґудзики, при цьому значно спростивши зображення, що дозволить покращити продуктивність додатку.

У традиційному підході комп’ютерного зору цей етап є надзвичайно важливим для продуктивності алгоритму [4, 15-17]. Але, як виявилося, можна досягти ще кращих результатів, використовуючи більш надійні особливості зображення, ніж просто визначення країв об’єктів. У прикладі з ґудзиками хороший детектор параметрів не лише розпізнає круглу форму ґудзиків, але і визначить особливості, які відрізняють ґудзики від інших круглих об’єктів, наприклад, автомобільних шин. Ось приклад декількох відомих ознак, які використовуються в комп’ютерному зорі: ознаки Хаара, Гістограма напрямлених градієнтів, SIFT, SURF, тощо.

Після виділення ознак зображення йде сам процес навчання класифікатора. На цьому етапі алгоритм приймає як вхідний параметр вектор ознак та повертає конкретний клас-мітку. Перед тим як застосовувати алгоритм на нових зображеннях, його слід навчити, показуючи йому тисячі навчальних зображень, на яких є об’єкт, цікавить нас клас та на яких такого об’єкту немає. Різні алгоритми навчання навчаються по-різному але основним принципом є те що алгоритм розглядає вхідний вектор ознак як

точки в просторі з великою розмірністю і намагається знайти площини або поверхні які б розділили даний простір таким чином, що всі приклади, які належать одному класу, розташовані по одну сторону від роздільної поверхні.

1.4 Постановка задачі дослідження

Для досягнення поставленої мети необхідно вирішити такі завдання: розробити застосунок, який міг би вирішувати задачу класифікації зображення. У цьому завданні застосовуються не тільки методи і алгоритми машинного навчання, а й комп'ютерний зір у системах технічного зору. У повсякденному світі людина часто стикається з проблемами, пов'язаними з розпізнаванням зображення. За різними даними людина сприймає від 80% до 90% інформації з допомогою зору. Для різних роботів і сконструйованих людиною машин, цей відсоток може істотно відрізнятися, але не можна не визнавати того факту, що зір, або якийсь з його різновидів, міг би додати більше методів і підходів у вирішенні різних завдань, поставлених перед машиною у СТЗ. Слід проаналізувати доступні алгоритми класифікації та зробити висновки на основі отриманих теоретичних даних. Для цього будуть розглянуті два алгоритми: машина опорних векторів та багатошаровий персептрон. Після теоретичного аналізу належить з'ясувати переваги та недоліки двох підходів. Далі необхідно розглянути приклади та підходи у СТЗ. Корисним прикладом застосування комп'ютерного зору та розпізнавання зображень можуть бути охоронні засоби, ці методи могли б стати хорошими помічниками для служб, які займаються безпекою людей та громадян різних країн. Зараз в місцях, що охороняються, розташована велика кількість відеокамер для проведення спостереження та забезпечення безпеки людей. На виробничих об'єктах відеоспостереження відіграє роль контролю якості роботи установок. Але як уже говорилося раніше, людина не може

бути постійно сконцентрована на спостереженні за показаннями різних приладів, за переглядом відеозапису камер і тому вона може пропустити критичні моменти, які можуть призвести до катастрофи. У даній роботі для вирішення поставлених завдань будуть розглянуті три методи вилучення даних з графічного зображення.

Об’єктом дослідження є методи класифікації щодо розпізнавання зображень.

Метою дослідження є реалізація класифікатора графічних зображень з урахуванням обраних алгоритмів заснованих на багатошаровому персептроні та методі опорних векторів.

Також для досягнення поставленної мети нобхідно вирішити такі завдання:

– реалізувати класифікатор графічних зображень з урахуванням обраних алгоритмів,

– провести аналіз роботи алгоритмів та його точності класифікації.

2 МЕТОДИ КЛАСИФІКАЦІЇ ДАНИХ

2.1 Машина опорних векторів

Машинне навчання вирішує низку основних прикладних завдань. МОВ – це лінійна система, яка має низку привабливих особливостей. Опис роботи таких машин слід починати з питання роздільності класів, що виникає під час вирішення завдань класифікації. У цьому контексті ідея МОВ полягає у побудові гіперплощини, яка виступає в ролі поверхні рішення, що максимально розділяє позитивні та негативні приклади. Це досягається завдяки підходу, що ґрунтується на теорії статичного навчання, а саме: МОВ є апроксимуючою реалізацією методів мінімізації структурного ризику. Цей індуктивний принцип заснований на тому, що рівень помилок навчальної машини на тестових даних можна подати у вигляді суми помилок навчання та доданку, що залежить від розмірності Вапника-Червоненкіса [2]. В Оксфордському журналі було надруковано статтю О'Фалона про використання методу опорних векторів для знаходження однонуклеотидних поліморфізмів у геномі людини [3]. Хоча раніше цю тему вже було висвітлено, але вирішення цього завдання методом опорних векторів було описано вперше. Як наведено у [3, 18-21] переваги методів, заснованих на машинному навчанні, полягає в тому, що вони дозволяють комбінувати різні фактори, які впливають на правдоподібність виникнення поліморфізму в даній позиції геному, що збільшує чутливість методу до більш рідкісних поліморфізмів.

Переваги методів, що ґрунтуються на машинному навчанні, полягають в тому, що вони дозволяють комбінувати різні фактори, які впливають на ймовірність виникнення поліморфізму в даній позиції геному, що, зокрема, підвищує чутливість методу до більш рідкісних поліморфізмів. На відміну від інших методів машинного навчання, для машин опорних векторів

потрібно значно менше даних для навчання. Згідно з графіком, представленим у статті, MOB перевищує інші методи визначення поліморфізмів за чутливістю та специфічністю. Метод визначив 95,6% всіх дійсно позитивних прикладів. Графік наведено на рисунку 2.1.

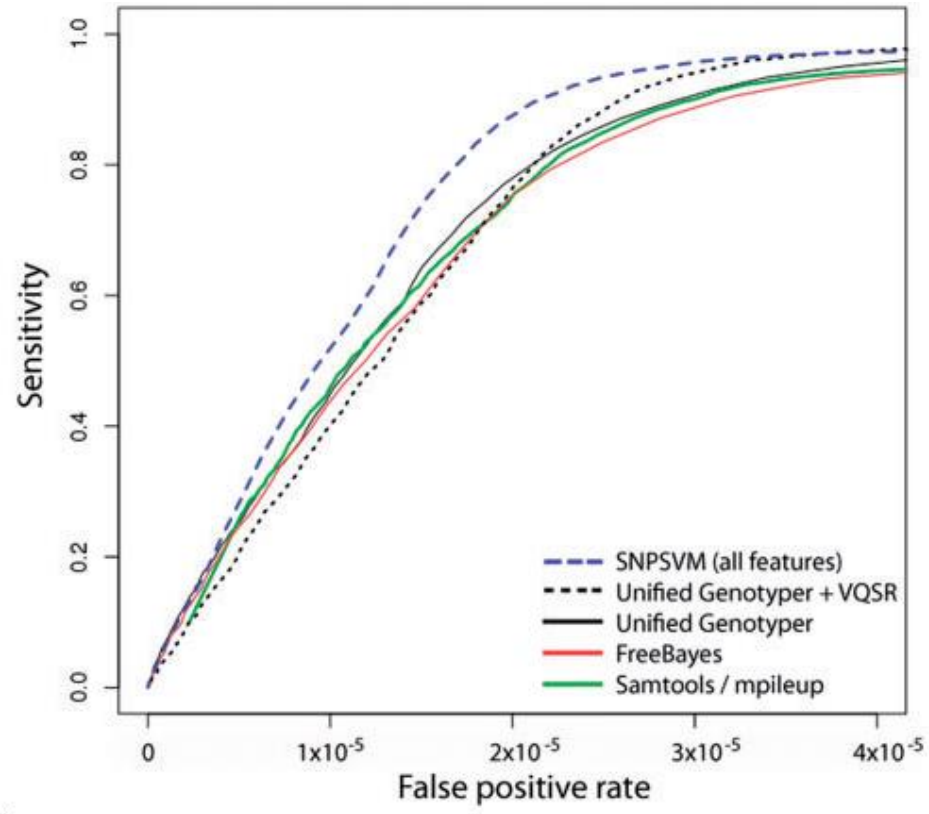

Рисунок 2.1 – Графік порівняння алгоритмів класифікації однонуклеотидних поліморфізмів у геномі людини

Серед недоліків описаного підходу слід відзначити, що модель машини опорних векторів може давати погані результати на даних, які містять різні профілі помилок. Крім того, незважаючи на багатоаспектність, реалізація виявилася на 10-20% повільнішою, ніж інші алгоритми. Обрана модель добре відповідає специфіці поставленої задачі. Ще одним прикладом використання машини опорних векторів є розв’язання завдань машинного зору [3]. Цей метод використовується для класифікації зображень. Наприклад, програма, яка аналізує зображення, відповідала б на питання: «Чи є об’єкт на зображенні людиною?» Для цього їй потрібно передати на вхід дані з

позитивним прикладом (зображення з людьми) і негативними. Завдяки машині опорних векторів класифікатори в подальшому зможуть розпізнавати людей уже на нових вхідних даних. Проте дослідження в цьому напрямку ще проводяться, і існує цілий ряд задач які ще не мають рішення. Складність даного завдання полягає в тому, що зображення динамічне, з різним освітленням, віддаленістю об'єкта від камери і кутом нахилу відносно цієї камери, а також можливими перекриттями об'єктів. Ще одна складність полягає в великій кількості шуканих об'єктів, що в свою чергу веде до необхідності обробки об'ємної бази зображень. Таким чином, час на розв'язання великих задач може виявитися неприйнятним. Це питання стало дуже важливим в завданнях, які вимагають розв'язання в реальному часі. Сьогодні для розв'язання цього завдання був розроблений метод l Latent SVM. Відмінність від звичайної машини опорних векторів полягає в тому, що додається третій параметр, прихована змінна. Це обумовлено тим, що у багатьох завдань вхідні та вихідні зв'язки залежать не тільки від пари *x* і *y*, а також від прихованих змінних. МОВ – це алгоритм машинного навчання з вчителем, який використовується в завданнях класифікації та регресійного аналізу. Особливістю цього методу є безперервне зменшення емпіричної помилки класифікації і збільшення зазору, тому цей метод також відомий як класифікатор з найбільшим зазором.

2.1.1 Лінійно-роздільні дані та оптимальна гіперплощина

Множина початкових параметрів має вигляд:

$$\{(x_i, d_i\}_{i=1}^{N}, \tag{2.1}$$

де x_i – вхідний образ прикладу i;

d_i – відповідний бажаний відгук (цільовий вихід).

Спочатку припустимо, що клас, представлений підмножиною $d_i = 1$ і клас, представлений підмножиною $d_i = -1$, лінійно розділені. Рівняння поверхні рішень у формі гіперплощини, яка виконує цей поділ, записується наступним чином:

$$w^T x + b = 0, \tag{2.2}$$

де w – вектор ваг, який нелагоджується;

x – вхідний вектор;

b – поріг.

Таким чином можна записати:

$$w^T x_i + b \geq 0 \text{ для } d_i = 1, \tag{2.3}$$

$$w^T x_i + b < 0 \text{ для } d_i = -1, \tag{2.4}$$

де w – вектор ваг, який нелагоджується;

x – вхідний вектор;

b – поріг;

d_i – відповідний бажаний відгук (цільовий вихід).

Для даного вектора ваг та порогу відстань між гіперплощиною, яка визначається формулою (2.2), та найближчою точкою з набору даних називається межею розділення і позначається символом *P*. Основною метою методу опорних векторів є пошук конкретної гіперплощини для якої межа розділення буде максимальною. При цьому умова рішення називається оптимальною гіперплощиною. На рисунку 2.2 показане геометричне представлення гіперплощини у двовимірному просторі вхідних сигналів [7].

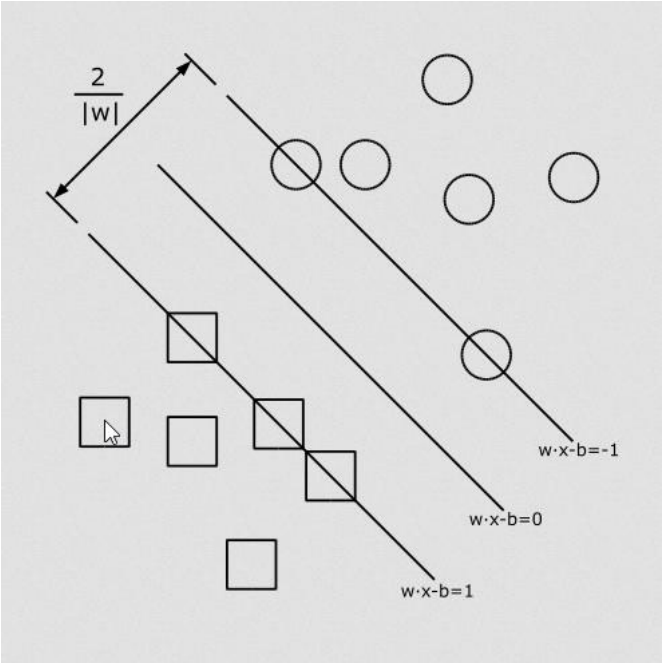

Рисунок 2.2 – Оптимальна гиперплощина

Нехай w_0 та b_0 – оптимальні значення векторів ваг та порога відповідно. Виходячи з цього, оптимальна гіперплощина, яка представляє багатовимірну лінійну поверхню рішень у просторі вхідних сигналів, можна описати рівнянням:

$$w_0^T x + b_0 = 0 \ , \tag{2.5}$$

де w_0 – оптимальний вектор ваг;

x – вхідний вектор;

b_0– оптимальний поріг.

Дане рівняння є аналогом рівняння 2.2. При цьому дискримінантна функція 2.6 визначає міру відстані алгебри від точки X до оптимальної гіперплощини:

$$g(x) = w_0^T x + b_0 \, , \tag{2.6}$$

де $g(x)$ – дискримінантна функція;

w_0 – оптимальний вектор ваг;

x – вхідний вектор;

b_0 – оптимальний поріг.

Якщо виразити X наступним чином:

$$x = x_p + r\frac{w_0}{\|w_0\|}, \tag{2.7}$$

де x– вхідний вектор;

x_p – нормальна проекція точки X на гиперплощині;

r – бажана алгебраїчна відстань;

w_0 – оптимальний вектор ваг.

Величина r є позитивною, якщо знаходиться з позитивного боку оптимальної гіперплощини, та негативної у протилежному випадку. З рівнянь 2.5 та 2.6 маємо:

$$r = \frac{g(x)}{\|w_0\|}, \tag{2.8}$$

де r – бажана алгебраїчна відстань;

$g(x)$ – дискримінантна функція;

w_0 – оптимальний вектор ваг.

Якщо $b_0>0$, то початок координат знаходиться з позитивного боку оптимальної гіперплощини, якщо ж $b_0<0$ то з негативного. Треба знайти такі параметри w_0 та b_0 оптимальної гіперплощини на основі даної множини прикладів. Дана пара параметрів повинна відповідати таким умовам:

$$w_0^T x_i + b_0 \geq 1 \text{ для } d_i = 1 \tag{2.9}$$

$$w_0^T x_i + b_0 \leq 1 \text{ для } d_i = -1 \tag{2.10}$$

де w_0 – оптимальний вектор ваг;

x – вхідний вектор;

b_0 – оптимальний поріг;

d_i – відповідний бажаний відгук (цільовий вихід).

Конкретні точки (x_i,d_i), для яких перше та друге обмеження виконуються зі знаком рівності, називають опорними векторами. Звідси методи та отримали свою назву машини опорних векторів. Ці вектори грають вирішальну роль роботі машин, а опорні вектори є тими точками, які лежать найближче до поверхні розв'язання, і, таким чином, є найбільш складними для класифікації.

Відповідно формулі 2.8, алгебраічна відстань від опорного вектора $x^{(s)}$ до оптимальної гиперплощини дорівнює:

$$r = \frac{g(x^{(s)})}{\|w_0\|} = \begin{cases} \frac{1}{\|w_0\|}, \textit{якщо } d^{(s)} = 1, \\ -\frac{1}{\|w_0\|}, \textit{якщо } d^{(s)} = -1 \end{cases} \tag{2.11}$$

де r – бажана алгебрічна відстань;

g – дискримінантна функція;

$x^{(s)}$– опорний вектор;

w_0 – оптимальний вектор ваг;

$d^{(s)}$– відгук на опорному векторі.

Нехай p – оптимальне значення межі поділу між двома класами, що становить безліч прикладів. Тоді з рівняння 2.11 випливає, що:

$$\rho = 2r = \frac{2}{\|w_0\|}, \tag{2.12}$$

де ρ – оптимальне значення меж разподілу;

r – бажана алгебрічна відстань;

w_0 – оптимальний вектор ваг.

З цього випливає, що максимізація межі поділу між класами еквівалентна мінімізації Евклідової норми вектора w.

2.1.2 Пошук оптимальної гіперплощини

Для того щоб знайти оптимальну гіперплощину потрібно вирішити наступне завдання: знайти оптимальне значення вектора коефіцієнтів ваг та поріг, які задовольняють умові:

$$d_i\left(w^T x_i + b\right) \geq 1 \ \textit{для}\ i = 1,2,...,N, \tag{2.13}$$

де d_i – відповідний очікуваний відклик;

w – вектор ваг;

x – вхідний вектор;

b – поріг.

Та які мінімізують функцію вартості:

$$\Phi\left(w\right) = \frac{1}{2} w^T w, \tag{2.14}$$

де $\Phi\left(w\right)$ – функція вартості;

w – вектор ваг.

Таке завдання можна вирішити за допомогою методу множників Лагранжа. Спочатку варто побудувати функцію Лагранжа:

$$J\left(w,b,a\right) = \frac{1}{2} w^T w - \sum_{i=1}^{N} a_i \left[d_i\left(w^T x_i + b\right) - 1 \right], \tag{2.15}$$

де $J(w,b,a)$ – функція Лагранжа;

w – вектор ваг;

a_i – множники Лагранжа;

d_i – відповідний бажаний відгук (цільовий вихід);

x – вхідний вектор;

b – поріг.

Розв'язання задачі умовної оптимізації визначається сідловою точкою функції Лагранжа, яку необхідно мінімізувати і одночасно максимізуючи. Продиференціювавши функцію з i, будуть отримані умови оптимальності, які використовуються для отримання умов 2.16 та 2.17 з функції Лагранжа:

$$w = \sum_{i=1}^{N} a_i d_i x_i \,, \tag{2.16}$$

$$\sum_{i=1}^{N} a_i d_i = 0 \,. \tag{2.17}$$

Вектор рішень w визначається в термінах розширення, яке включає N навчальних прикладів. Дане рішення єдине завдяки опуклості функції Лагранжа. Рішення прямої задачі допускає опуклість функції вартості та лінійність обмежень. Для такого завдання можна сформулювати подвійне завдання, яке матиме ті ж оптимальні рішення, але оптимальність її рішення забезпечуватиметься множниками Лагранжа. Таким чином визначимо двоїсте завдання: для даної навчальної множини знайти множники Лагранжа, які максимізують функцію:

$$Q(a) = \sum_{i=1}^{N} a_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} a_i a_j d_i d_j x_i^T x_j \,, \tag{2.18}$$

при наступних обмеженнях:

$$\sum_{i=1}^{N} a_i d_i = 0, \tag{2.19}$$

$$a_i \geq 0 \ \ для\, i = 1,2,\ldots,N. \tag{2.20}$$

Визначивши оптимальні множники Лагранжа, за допомогою рівняння 2.16 можна визначити оптимальний вектор коефіцієнтів ваг w_0.

2.1.3 Оптимальна гіперплощина для нерозділених даних

Коли дані нероздільні неможливо побудувати гіперплощину яка б не мала помилок при класифікації але можна спробувати мінімізувати такі помилки. Для того, щоб розглянути нерозділені дані введемо в позначення гіперплощини нову множину невід'ємних скалярних змінних $\{\xi_i\}_{i=1}^{N}$:

$$d_i\left(w_i^T + b\right) \geq 1 - \xi_i i = 1,2,\ldots,N, \tag{2.21}$$

де ξ_i – фіктивна змінна, яка визначає відхилення точок від ідеального стану лінійної роздільності.

Завдання зводиться до того, щоб знайти гіперплощину, яка мінімізує помилку класифікації на багатьох навчальних прикладах. Таке завдання мінімізації є невипуклою NP-повною задачею. Функцію мінімізації щодо вектора можна записати так:

$$\Phi\left(w,\xi\right) = \frac{1}{2} w^T w + C\sum\nolimits_{i=1}^{N} \xi_i \ . \tag{2.22}$$

де C – параметр регулюємий користувачем.

Параметр C забезпечує компроміс між складністю машини і кількістю точок, що не розділяються. Параметр можна вибрати двома способами:

експериментально, на основі безлічі рішень, та аналітично, оцінюючи розмірність та використовуючи межі ефективності узагальнення машини на основі розмірності. У будь-якому разі функція $\Phi(w,\xi)$ мінімізується по w і $\{\xi_i\}_{i=1}^{N}$ при умові 2.21 та $\xi_i \geq 0$.

Використовуючи метод множників Лагранжа і виконуючи перетворення, аналогічні тим, які використовувалися для лінійно-роздільних множин, можна сформулювати подвійне завдання для нероздільної множини: для даної навчальної множини знайти множники Лагранжа, які максимізують функцію 2.18 при обмеженнях 2.19 и 2.23:

$$0 \leq a_i \leq C \text{ для } i = 1,2,\ldots,N. \qquad (2.23)$$

У даній двоїстій задачі не фігурують ані змінні ξ_i ані їхні множники Лагранжа. Таким чином, двоїста задача для нероздільних множин аналогічна більш простій задачі для лінійно-роздільних множин у всьому крім одного, але дуже важливої відмінності, в обох випадках максимізується одна й та сама функція, але у випадку нероздільних класів обмеження 2.20 замінено більш суворим обмеженням 2.23. Крім цієї зміни, задача умовної оптимізації для нероздільних множин та обчислення оптимальних значень вектора ваг та порогу не відрізняється від випадку лінійно-роздільних множин. Опорні вектори обчислюються тим самим способом, що і раніше. При побудові машини опорних векторів виконуються дві наступні операції: нелінійне відображення вхідного вектора в простір ознак вищої розмірності та побудова оптимальної гіперплощини для розділення ознак, отриманих при першій операції. Перша операція виконується відповідно до теореми Ковера про роздільність відображень. Друга операція використовує ідею побудови оптимальної гіперплощини. Тільки за допомогою оптимальних гіперплощин можна мінімізувати VC-розмірність і досягти хорошого узагальнення, це

питання і є суттю другої операції. Важливою складовою цих операцій є оцінка ядра скалярного добутку.

2.1.4 Машина опорних векторів на основі ядер

За допомогою різних ядерних функцій можна побудувати поверхню рішень, яка у вхідному просторі була б нелінійною, проте її зображення в просторі ознак було б лінійним. За допомогою такого розкладу можна визначити двоїсту задачу умовної оптимізації для машини опорних векторів з обмеженнями 2.20 та 2.23:

$$Q(a) = \sum_{i=1}^{N} a_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{i=1}^{N} a_i a_j d_i d_j K\left(x_i, x_j\right), \tag{2.24}$$

де $K\left(x_i, x_j\right)$ – ядерна функція.

Сформульована двоїста задача має ту саму структуру що й для лінійно-роздільних множин, за винятком того, що скалярний добуток, який там використовується, замінено ядром скалярного добутку. Ядро можна уявляти як елемент симетричної матриці розмірності $N \times N$. Ядро машини опорних векторів може бути будь-якою функцією, яка є скалярним добутком в деякому просторі векторів. Дану умову можна перевірити за допомогою теореми Мерсера, ось приклад найвідоміших ядер: поліноміальне (2.25), гаусіанське (2.26), сигмоїдне (2.27).

$$K\left(x, x_i\right) = \left(x^T x_i + 1\right)^p, \tag{2.25}$$

$$K\left(x, x_i\right) = \exp\left(-\frac{\left\|x - x_i\right\|^2}{2\sigma^2}\right), \tag{2.26}$$

$$K\left(x,x_i\right)=\tanh\left(kx^Tx_i+c\right). \tag{2.27}$$

Також варто відзначити, що для сигмоїди теорема Мерсера виконується лише для деяких *k* та *c*.

Якщо спробувати пояснити роботу машини опорних векторів простими словами, можна сказати, що основна задача цього алгоритму полягає в максимізації зазору між граничними даними. Іншими словами, маючи таке завдання:

$$\min_{\theta} C\sum_{i=1}^{m}\left[y^{(i)}cost_1\left(\theta^Tx^i\right)+\left(1-y^i\right)cost_0\left(\theta^Tx^i\right)\right]+\frac{1}{2}\|\theta\|. \tag{2.28}$$

При дуже великому *C* єдиний вихід це мінімізувати параметр θ що і робить машина опорних векторів. До речі, параметр *C* можна розглядати як параметр регуляризації. При дуже великих значеннях C можна отримати так зване перенавчання, а при малих значеннях – навпаки. Тому важливо підібрати цей параметр так, щоб він показував найбільшу точність при класифікації, але при цьому не перенавчивши класифікатор. Зазвичай цей параметр визначається експериментально.

2.2 Багатошаровий персептрон

Зазвичай мережа складається з безлічі сенсорних елементів (вхідних вузлів або вузлів джерела), які утворюють вхідний шар, одного або декількох прихованих шарів обчислювальних нейронів і одного вихідного шару нейронів. Вхідний сигнал поширюється по мережі у прямому напрямку від шару до шару дані мережі називають багатошаровими персептронами, які представляють собою узагальнення одношарового персептрона.

Багатошарові персептрони успішно використовуються для вирішення різноманітних складних завдань. При цьому навчання з вчителем виконується за допомогою алгоритму зворотнього поширення помилок [2].

Даний алгоритм ґрунтується на корекції помилок і його можна розглядати як узагальнення популярного алгоритму адаптивного фільтрування – алгоритму мінімізації середньоквадратичної помилки. Навчання за допомогою алгоритму зворотнього поширення помилок передбачає два проходження через всі шари мережі: прямий і зворотний. Під час прямого проходження вхідний вектор подається на сенсорні вузли мережі, після чого поширюється по мережі від шару до шару. У результаті генерується набір вихідних сигналів, який є фактичною реалізацією мережі на даному вхідному векторі. Під час зворотнього проходження всі синаптичні ваги налаштовуються відповідно до правила корекції помилок, а саме: фактичний вихід мережі віднімається від цільової відповіді, в результаті чого виникає сигнал помилки. Цей сигнал подальше поширюється по мережі в напрямку, протилежному напрямку синаптичних зв’язків. Звідси і назва – алгоритм зворотнього поширення помилок. Синаптичні ваги налаштовуються з метою максимального наближення вихідного сигналу мережі до цільового (в статичному розумінні). Процес навчання, реалізований цим алгоритмом, називається навчанням на основі зворотнього поширення.

Більшість багатошарових персептронів мають три відмінні риси:

– кожен нейрон мережі має нелінійну функцію активації, яка повинна бути всюди диференційованою. Одним з найпопулярніших прикладів такої функції може бути сигмоїдальна або логістична;

– нейронна мережа повинна містити один або декілька прихованих шарів, які дозволяють мережі навчатися вирішувати більш складні завдання, послідовно виділяючи найважливіші ознаки з вхідного вектора;

– мережа має високий рівень зв’язку, який реалізується за допомогою синаптичних зв’язків. Зміна рівня зв’язку мережі вимагає зміни набору з’єднань або їх вагових коефіцієнтів.

Усі ці властивості разом із можливістю навчання алгоритму на власному досвіді становлять переваги багатошарового персептрону у розв’язанні різних завдань. Але при цьому ці властивості породжують і недоліки такого алгоритму: складність теоретичного аналізу алгоритму через його нелінійність і високий рівень зв’язку, складність в візуалізації процесу навчання алгоритму через наявність прихованих шарів, які є важливою складовою в аналізі роботи алгоритму, визначенні того, які ознаки вхідного вектора слід передавати прихованим нейронам. У такому випадку процес навчання стає ще більш складним, оскільки пошук відбуватиметься у великій області можливих функцій, а вибір повинен виконуватися серед альтернативних представлень вхідних векторів.

Поява алгоритму зворотнього поширення стала важливою подією в розвитку нейронних мереж, оскільки він реалізує обчислювально ефективний метод навчання багатошарового персептрона. Цей алгоритм не пропонує дійсно оптимального рішення для всіх існуючих проблем, але він розвіяв песимізм щодо навчання багатошарових машин. Багатошаровий персептрон, так само, як і простіший – одношаровий персептрон, і будь-яка інша нейронна мережа, складається з нейронів. У дуже спрощеному варіанті, нейрон в основі – це обчислювальний блок, який приймає за допомогою входу (можна провести аналогію з дендритами в людському мозку) вхідний електричний сигнал (spikes), який далі передається до виходу (аксон). У загальній моделі нейрони є вхідними параметрами, а вихідний сигнал є результатом функції активації. На рисунку 2.3 представлена проста блок-схема нейрона.

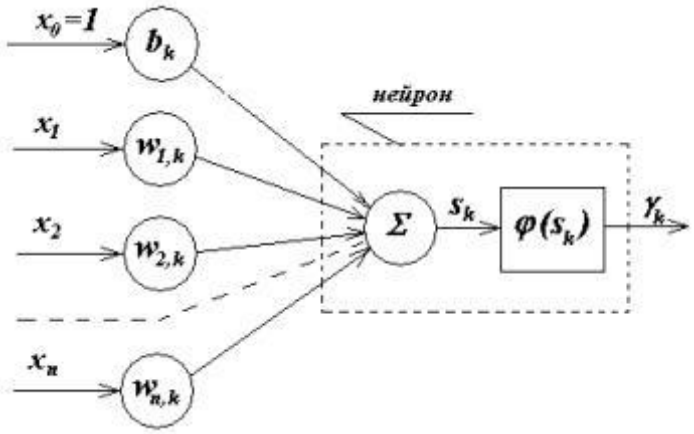

Рисунок 2.3 – Схема штучного нейрону

Якщо нейрон можна зобразити як схему він має і математичне уявлення. У формулах 2.29 та 2.30 наводиться приклад такого подання:

$$v_k = \sum_{j=0}^{m} w_{kj} x_j \,, \tag{2.29}$$

$$y_k = \varphi\left(v_k\right), \tag{2.30}$$

де v_k – потенціал активації;

w_{kj} – синаптичні ваги;

x_j – вхідні сигнали;

y_k – вихідний сигнал нейрону;

φ – функція активації.

Зокрема, з рівняння 2.30 виходить що важливою складовою нейрона є функція активації.

Функція активації визначає вихідний сигнал нейрона в залежності від індукованого локального поля. У якості аргументу вона приймає сигнал, отриманий на виході вхідного суматора нейрона. Найпоширеніші функції активації такі: жорстка порогова функція, лінійно-порогова функція, сигмоїдальна функція.

Жорстка порогова функція, або як її ще називають одиничний стрибок, – це проста кусково-лінійна функція. Вона обчислюється таким чином: якщо вхідне значення менше порогового то значення функції дорівнює мінімально допустимому, інакше – максимально допустимому.

$$\varphi(v)=\begin{cases}1, якщо\, v \geq 0;\\ 0, якщо\, v < 0\end{cases} \tag{2.31}$$

де φ – порогова функція;

v – індивідуальне локальне поле нейрона.

Дану модель називають моделлю МакКаллока-Піттса в ній вихідний сигнал нейрона приймає значення 1, якщо індуковане локальне поле цього нейрона не від'ємне, і 0 в іншому випадку. Такий вираз описує властивість «все чи нічого» моделі МакКаллока-Піттса (рис. 2.4).

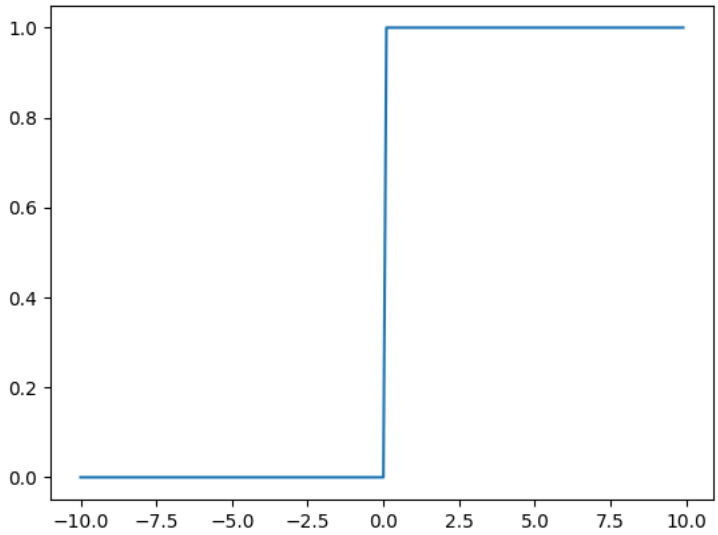

Рисунок 2.4 – Графік порогової функції активації

Лінійний поріг чи гістерезис є кусково-лінійною функцією, однак, на відміну від порогової функції, він має два лінійних відрізки, де активаційна функція тотожно рівна мінімально допустимому і максимально допустимому

значенням, і є відрізок, на якому функція неухильно монотонно зростає. Таку функцію можна розглядати як апроксимацію нелінійного підсилювача. Також існує кілька особливих форм цієї функції: якщо лінійна область оператора не досягає порогу насичення, вона перетворюється в лінійний суматор; якщо коефіцієнт підсилення лінійної області прийняти нескінченно великим, то кусково-лінійна функція дегенерується в порогову (рис. 2.5).

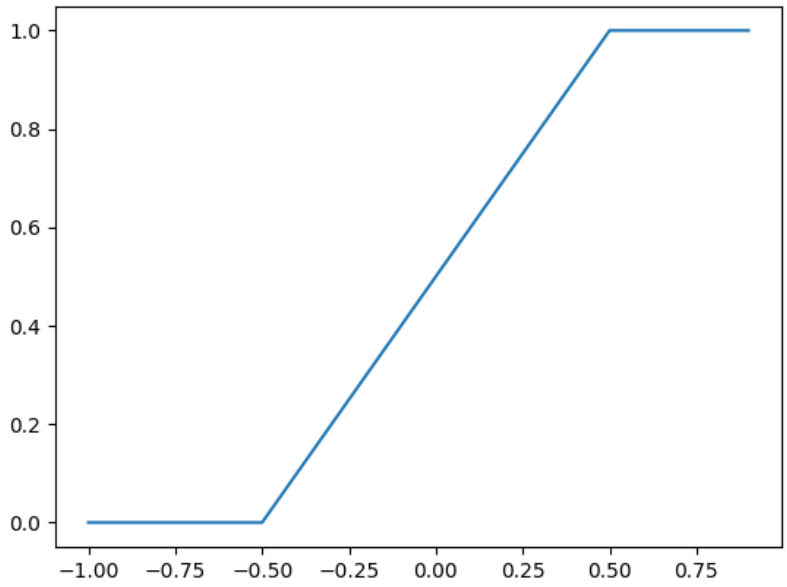

Рисунок 2.5 – Графік линійно-порогової функції активації

Логістична функція, яка є сигмоїдальною, є монотонно зростаючою та всюди диференційованою нелинійною функцією з насиченням. Її графік нагадує англійську літеру S. Дана функція є найбільш поширеною серед тих, що використовуються для створення штучних нейронних мереж і дозволяє підсилювати слабкі сигнали не насичуючись від сильних, також, вона успішно забезпечує баланс між лінійною та нелинійною поведінкою. Сигмоїд успішно вирішує проблему шумового насичення. Прикладом сигмоїдальної функції може бути логістична функція, яку можна виразити наступним виразом:

$$\varphi(v) = \frac{1}{1+e^{-av}}, \tag{2.32}$$

де *a* – параметр наклону сигмоїдальної функції.

Змінюючи цей параметр, можна будувати функції з різною крутістю. У межах, коли параметр нахилу наближається до нескінченності, сигмоїдальна функція стає пороговою. Якщо порогова функція може приймати лише значення 0 і 1, то сигмоїдальна функція приймає нескінченну кількість значень у діапазоні від 0 до 1. При цьому вона є диференційованою, а порогова такою не є (рис. 2.6).

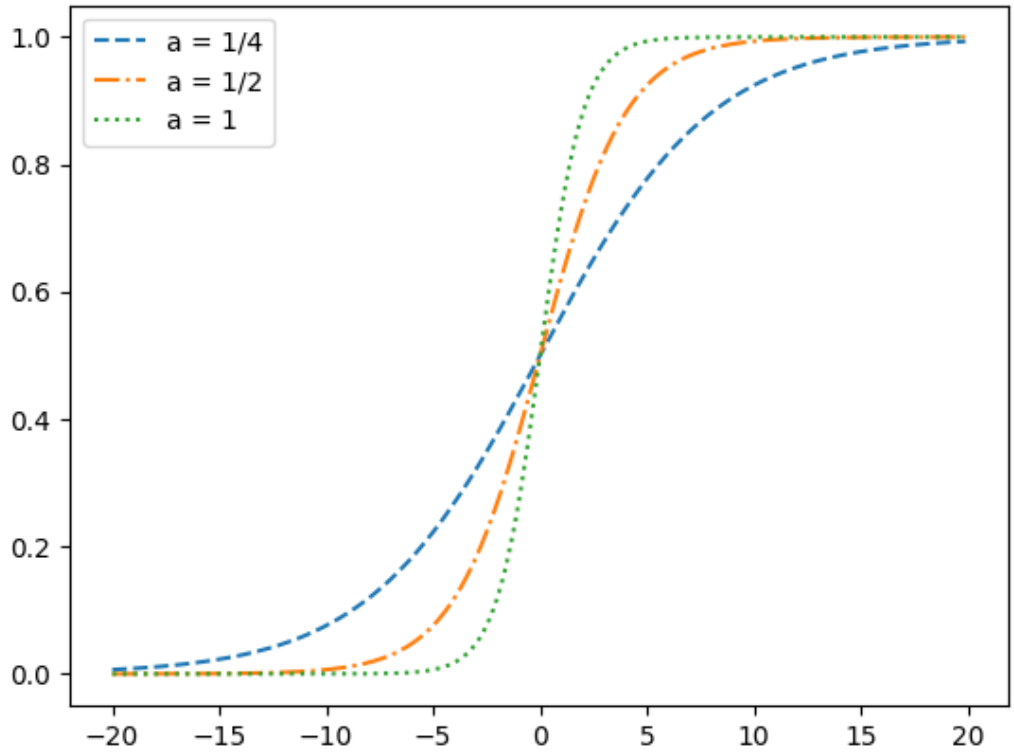

Рисунок 2.6 – Графік логістичної функції активації

Коли мова йде про навчання нейронної мережі, тут мається на увазі коригування синаптичних ваг і порогів. В ідеальному випадку нейронна мережа отримує знання про оточуюче середовище на кожній ітерації процесу навчання. З точки зору нейронної мережі, навчання можна визначити так: навчання – це процес, в якому вільні параметри нейронної мережі налаштовуються шляхом моделювання середовища у яке ця мережа

вбудована. Тип навчання визначається способом налаштування цих параметрів.

Навчання нейронної мережі можна уявити як послідовні події: в мережу надходять стимули зовнішнього середовища; внаслідок цього змінюються вільні параметри мережі; після завершення змін внутрішньої структури нейронна мережа відповідає на збудження іншим чином.

Цей список чітких правил вирішення проблеми навчання нейронної мережі називається алгоритмом навчання. Не складно здогадатися, що не існує універсального алгоритму навчання, який би підходив для всіх архітектур нейронних мереж. Існує лише набір засобів, представлений багатьма алгоритмами навчання, кожен з яких має свої переваги. Алгоритми навчання відрізняються один від одного способом налаштування синаптичних ваг нейронів. Ще однією відмінною рисою є спосіб зв'язку навчаної нейронної мережі з зовнішнім світом у цьому контексті говорять про парадигму навчання, пов'язану з моделлю оточуючого середовища, в якому функціонує дана нейронна мережа.

Алгоритм адаптації вектора вагових коефіцієнтів персептрона формулюється наступним чином. Якщо *n-й* елемент *x(n)* навчального множини правильно класифікований за допомогою вагових коефіцієнтів *w(n)*, обчислених на *n*-му кроці алгоритму, то вектор ваг не коригується, тобто діє наступне правило:

$$w(n+1) = w(n), \text{ якщо } w^T x(n) > 0 \text{ та } x(n) \in C_1, \tag{2.33}$$

$$w(n+1) = w(n), \text{ якщо } w^T x(n) \le 0 \text{ та } x(n) \in C_2, \tag{2.34}$$

де C_1 – перший клас;

C_2 – другий клас;

w – синаптичні ваги;

x – вхідні сигнали.

В іншому випадку вектор ваг персептрону піддається корекції відповідно до наступного правила:

$$w(n+1)=w(n)-\eta(n)x(n),\ якщо\ \ w^T x(n)>0\ \ та\ \ \ x(n)\in C_2, \qquad (2.35)$$

$$w(n+1)=w(n)-\eta(n)x(n),\ якщо\ \ w^T x(n)\le 0\ \ та\ \ \ x(n)\in C_1, \qquad (2.36)$$

де $\eta(n)$ – параметр швидкості навчання.

Параметр швидкості навчання залежить від номеру ітерації, тому вищеописаний алгоритм називається правилом адаптації з фіксованим збільшенням. Алгоритм адаптації вектора вагових коефіцієнтів відповідає правилу навчання на основі корекції помилок:

$$w(n+1)=w(n)+\eta\left[d(n)-y(n)\right]x(n), \qquad (2.37)$$

де $d(n)-y(n)$ – сигнал помилки.

Параметр швидкості навчання є позитивною константою яка належить інтервалу від нуля до одиниці. Обираючи значення параметра швидкості навчання з цього діапазону, слід враховувати дві взаємовиключні вимоги: усереднення попередніх вхідних сигналів, що забезпечує стійкість оцінки вектора ваг та потребує малих значень параметра швидкості навчання; швидка адаптація до реальних змін розподілу процесу, що відповідає за формування векторів вхідного сигналу x і потребує великих значень параметра швидкості навчання.

У багатошаровому персептроні для налаштування синаптичних ваг використовується механізм зворотного поширення помилок. У звичайному багатошаровому персептроні сигнали передаються виключно в прямому напрямку, від шару до шару, розрізняючи два види сигналів: функціональний сигнал прямого поширення і сигнал помилки зворотнього поширення.

Функціональним сигналом є вхідний сигнал, який надходить в мережу і передається від нейрона до нейрона по всій мережі. Такий сигнал досягає кінця мережі у вигляді вихідного сигналу. Його називають функціональним через те, що призначений для виконання певної функції на виході мережі, а також через те що в кожному нейроні, через який передається цей сигнал, обчислюється певна функція з урахуванням вагових коефіцієнтів.

Сигнал помилки, з іншого боку починається на виході нейронної мережі і поширюється у зворотньому напрямку. Він отримав свою назву завдяки тому, що обчислюється кожним нейроном мережі на основі функції помилки, представленої в певній формі. Багатошаровий персептрон складається з трьох видів нейронних шарів: вхідний шар, прихований шар та вихідний шар.

Вихідний шар мережі складають вихідні нейрони, а інші нейрони відносяться до прихованих шарів. Приховані вузли не є частиною входу чи виходу нейронної мережі отже, вони і отримали свою назву. Перший прихований шар отримує дані з вхідного шару мережі та передає їх далі другому прихованому шару і так далі до самого кінця нейронної мережі.

Будь-який прихований або вихідний нейрон може виконувати два типи обчислень: обчислення функціонального сигналу та обчислення оцінки вектора градієнту необхідного для зворотного проходження через мережу. У реалізації алгоритму зворотнього розповсюдження в реальному часі більш перевагою є послідовне коригування ваг нейронної мережі. У такому режимі алгоритм циклічно обробляє приклади з навчальної вибірки. Крок за кроком реалізація виглядає наступним чином:

– ініціалізація. На цьому етапі передбачається, що немає жодної інформації про середовище, ваги та порогові значення генеруються випадковим чином з середнім значенням 0, єдине умова – індуковане поле нейрона повинне лягати на лінійну частину функції активації і не досягати області насичення;

– пред’явлення прикладів навчання. На цьому етапі в мережу подаються значення з навчального вибірки, для кожного значення виконуються спочатку прямий, а потім і зворотний проходи;

– прямий прохід. Під час прямого проходу обчислюються індуковані локальні поля та функціональні сигнали мережі, проходячи по ній шар за шаром у прямому напрямку. В кінці цього етапу обчислюється сигнал помилки, який буде використовуватися у зворотньому проході для налаштування синаптичних ваг та порогових значень нейронної мережі;

– зворотній прохід. Обчислюються локальні градієнти вузлів мережі та зміна синаптичних ваг відповідно до загального дельта-правила;

– ітерація. Послідовно виконуються прямі та зворотні проходи, представляючи мережі всі приклади навчання епохи, доки не буде досягнутий критерій зупинки.

2.3 Порівняння машини опорних векторів та багатошарового пересптрону

Багатошаровий персептрон, так само як і машина опорних векторів, є дуже популярним алгоритмом у машинному навчанні. Зараз багато завдань успішно вирішуються як першим, так і другим алгоритмом. У попередніх розділах кваліфікаційної роботи вони були розглянуті. Зараз варто поговорити про переваги та недоліки цих алгоритмів і порівняти які можливості вони пропонують для вирішення завдань класифікації графічних зображень.

Машина опорних векторів має кілька переваг:

– алгоритм ефективний на даних з високою розмірністю;

– залишається досить ефективним у випадках коли кількість параметрів перевищує кількість навчальних прикладів;

– використовує опорні вектори тим самим є ефективним з точки зору споживання пам’яті;

– гнучкий, можна користуватися різними ядерними функціями. Існує перелік часто використовуваних та перевірених ядерних функцій, але є можливість створити зовсім нову функцію;

– цей метод зводиться до вирішення задачі квадратичного програмування у випуклій області, яка завжди має єдине рішення;

– алгоритм знаходить таку роздільну площину, щоб проміжок між різними класами був максимальним, таким чином у майбутньому це дозволяє бути більш впевненим у класифікації об’єктів;

– цей метод є найшвидшим у знаходженні рішучих функцій.

Але при цьому машина опорних векторів має свій ряд недоліків, до яких відносяться:

– коли кількість параметрів набагато більша, ніж кількість навчальних прикладів, то в кінцевому підсумку алгоритм буде показувати слабку продуктивність;

– машина опорних векторів не надає безпосередніх ймовірнісних оцінок, вони обчислюються за допомогою перехресної перевірки;

– метод чутливий до шумів;

– не існує загального підходу до автоматичного вибору ядра (і побудови простору взаємодії взагалі) в разі лінійної нероздільності класів;

– цей алгоритм не може безпосередньо класифікувати декілька класів об’єктів, для багатокласової класифікації використовується підхід «один проти всіх».

Машина опорних векторів – потужний інструмент для побудови якісного класифікатора графічних зображень. Але, з іншого боку, низька продуктивність у випадках з перекосом на користь великої кількості параметрів, а також чутливість до шумів, змушують задуматися, чи відповідає цей алгоритм вимогам практичного застосування. Слід розглянути переваги та недоліки багатошарового персептрона для повного розуміння

теоретичних особливостей обох алгоритмів. Багатошаровий персептрон - досить старий алгоритм, що практично лежить в основі створення нейронних мереж. Хоча від того часу пройшло багато часу, цим алгоритмом часто користуються, особливо коли треба вирішити не надто складні завдання. У наші дні цей алгоритм отримав нове життя завдяки зростанню обчислювальних потужностей комп'ютерів і появі великої кількості даних, які потрібно аналізувати.

Переваги багатошарового персептрона включають:

– можливість навчання на нелінійних моделях;

– можливість навчання моделі в реальному часі;

– використання локальних обчислень дозволяє досягти плавного зниження продуктивності при виникненні збоїв, що забезпечує стійкість алгоритму;

– алгоритм добре підходить для паралельної архітектури, що підвищує ефективність і швидкість роботи багатошарового персептрона.

Недоліками такого алгоритму є:

– багатошаровий персептрон з прихованим шаром має невипуклу функцію втрат, в якій існує більше одного локального мінімуму. Тому різні значення ваг при ініціалізації можуть призводити до різної точності класифікації;

– алгоритм потребує ручної настройки параметрів, таких як кількість прихованих нейронних шарів або кількість ітерацій навчання;

– алгоритм чутливий до масштабування параметрів навчального набору, тому рекомендується масштабувати всі параметри перед початком навчання для отримання кращих результатів.

Отже, обидва алгоритми мають свої переваги і недоліки. Якщо порівнювати їх з точки зору класифікації графічних зображень, можна відзначити наступні ключові особливості:

– машина опорних векторів програє багатошаровому персептрону за швидкістю навчання, особливо якщо врахувати, що обчислення

багатошарового персептрону можна легко паралелити, що не так просто зробити у випадку машини опорних векторів;

– при використанні цих алгоритмів у задачі багатокласової класифікації багатошаровий персептрон матиме ще одну перевагу, оскільки МОВ є класифікатором двох класів. Її можна адаптувати для розв’язання задачі багатокласової класифікації, але при цьому швидкість навчання помітно знизиться але при передбаченні результату МОВ працює швидше.

Дуже важливим параметром при порівнянні цих алгоритмів є не тільки швидкість навчання і роботи, але і точність передбачень за даними. Цей параметр дуже важливий, оскільки швидко, але неправильно працюючий алгоритм не може бути корисним в реальних умовах. У такому порівнянні машина опорних векторів часто показує кращі результати, ніж багатошаровий персептрон. Звісно, всі ці результати варто порівнювати лише при правильно налаштованих параметрах обох алгоритмів, а точність алгоритму прямо залежить від якості поданих даних, але при інших рівних умовах машина опорних векторів все ж показує результати в точності краще, ніж звичайний багатошаровий персептрон. Але на практиці досягти значущої різниці на користь машини опорних векторів дуже непросто через велику кількість налаштовуваних параметрів, таких як вибір ядерної функції, вибір параметрів цієї ядерної функції, вибір значення роздільного параметра.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ ТА АНАЛІЗ АЛГОРИТМІВ

3.1 Аналіз засобів програмної реалізації класифікатора

На реалізацію класифікатора графічних зображень було обрано мову програмування Python [8-10]. Дана скриптова мова добре підходить для вирішення поставленої задачі, оскільки має ряд переваг порівняно з іншими мовами. До таких переваг відноситься те що інтерпретатор мови реалізований практично на всіх платформах та операційних системах. Також серед переваг можна відзначити постійну розширюваність мови, доступність вихідного коду для будь-яких маніпуляцій, та кожен програміст може завантажити його для себе та виправити будь-які особливості, які не підходили для вирішення спеціальних завдань. Також, в нагоді є те що існує обширна спільнота і велика кількість підключуваних програмних модулів та бібліотек.

Дана перевага надає можливість програмістам обирати Python як основну мову розробки для вирішення великої кількості різних завдань. Так, наприклад, за допомогою бібліотек Django та Flask розробник може створювати додатки, спрямовані на клієнт-серверну архітектуру та різноманітні вебзастосунки. Більше того, існує ряд бібліотек, які дозволяють розробляти додатки з графічним користувацьким інтерфейсом, які будуть працювати на будь-якій операційній системі без необхідності в переписуванні або перекомпіляції коду додатка.

Ще однією перевагою є простота в освоєнні цієї мови. Основам і синтаксису мови можна легко навчитися за досить короткий проміжок часу немає перегружених, складних в розумінні конструкцій, завдяки чому вона проста в освоєнні. Крім того, синтаксис сприяє написанню читабельного коду та підтримці чистоти в коді це забезпечується спеціальними правилами

пунктуації. Наприклад, кінець рядка є кінцем інструкції, вкладені інструкції об’єднуються в блоки за розміром відступів.

У даній роботі мова Python використовувалася через її потужну підтримку в області машинного навчання. Так для побудови додатка класифікації графічних зображень були використані готові бібліотеки, такі як OpenCV, NumPy, Scikit.

OpenCV призначена для обробки графічних зображень в ній зібрані зручні та корисні методи з вилучення параметрів з зображень, їх попередню обробку та корекцію основних параметрів зображень.

NumPy містить потужний математичний комплекс, призначений для виконання операцій над матрицями та векторами, що в машинному навчанні є надзвичайно важливими компонентами. У цій бібліотеці присутні функції з обробки даних, їх зручного зберігання та виконання математичних дій.

Scikit містить методи машинного навчання, в якому вже реалізовані алгоритми, такі як машина опорних векторів та багатошаровий персептрон. Крім того, в даній бібліотеці є великий комплекс методів і засобів, спрямованих на обробку та аналіз даних, а інтеграція з іншими бібліотеками робить Scikit потужним інструментом для вивчення та аналізу основних методів та алгоритмів машинного навчання. Крім того, бібліотека включає в себе безліч методів і даних, які допоможуть розробнику в освоєнні та тренуванні алгоритмів без зайвого відволікання на збір даних та їх пошук. Засоби та бібліотеки Python також містять додаткові допоміжні інструменти для візуалізації та аналізу отриманих даних. Одним з таких інструментів є matplotlib який ирішує проблему відображення та побудови графіків. На основі зазначених переваг Python був обраний для розробки додатку та аналізу методів машинного навчання у вирішенні задач класифікації. Крім того, простий синтаксис і велика кількість засобів з готовими математичними функціями, робота з векторами та матрицями не відволікають від низькорівневих особливостей системи і рутинної реалізації основних функцій математичного апарату, а дозволяють зосередитися на головному –

аналізі даних та методах машинного навчання. Для розробки було використано середовище розробки Spyder, яке є зручним рішенням для реалізації додатків на мові Python. На рисунку 3.1 показаний графічний інтерфейс Spyder.

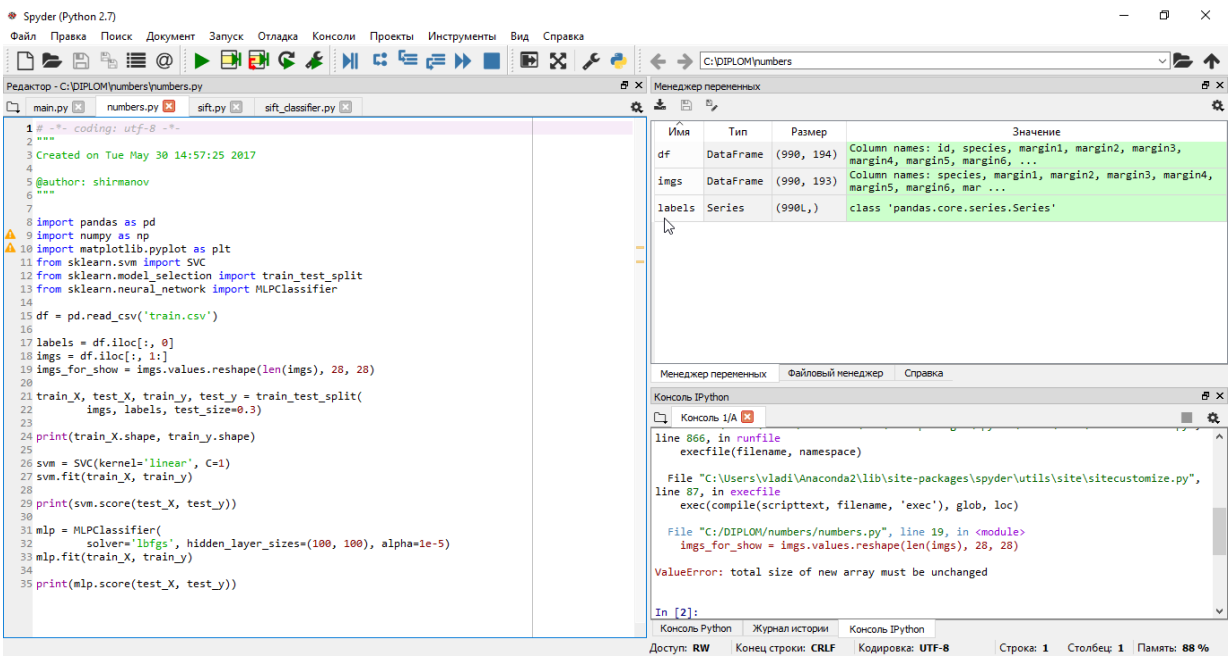

Рисунок 3.1 – Графічний інтерфейс середи розробки Spyder

Включає в себе безліч корисних функцій, які прискорюють розробку. Серед найкорисніших можна відзначити запуск та відлагодження додатків прямо з середовища також, ще однією перевагою є графічний інтерфейс системи, який має корисні дані для відлагодження додатка.

3.2 Розробка застосунку класифікації зображень

На основі графічних зображень без додаткових текстових описів були вибрані вихідні дані для поставленої задачі. Дані грають ключову роль в задачах класифікації, так само, як і в інших задачах навчання з вчителем. Для роботи були використані дані з вибірки Swedish leaf dataset (дані надано

репозиторієм www.kaggle.com.) Сайт є відомою платформою для розробників, які займаються питаннями машинного навчання.

Представлені дані включають 1584 зображення листя. Всі листя належать до різних класів дерев, які програма повинна правильно визначати, зображення вже мають невелику попередню обробку – кольорова гамма зведена до двох кольорів, чорного і білого. Таким чином, лист намальований білим кольором на чорному фоні. На рисунку 3.2 подано приклад такого зразка [11-14].

Рисунок 3.2 – Зразок даних представлений у навчальній множині

Зображення представлені у різних форматах (розширення файлів – різні); висота і ширина вимагають додаткової обробки та налаштувань. Таким чином, дані потребують додаткової обробки перед використанням в алгоритмах машинного навчання. Набір з ідентифікатором аркуша і його класом зберігається окремо у файлі з розширенням .csv. Дане розширення є зручним форматом для зберігання даних він легко читається як у Excel так і у вбудованій засобах Python. У першій реалізації в якості параметрів зображення були використані значення кольору кожного пікселя. За такого підходу спочатку знадобилося перетворити зображення до єдиного розширення, оскільки матриця чисел на вході в навчальний алгоритм повинна була мати однакові розміри для всіх навчальних прикладів. У результаті всі зображення були трансформовані так, щоб їхня розмірність складала 500x500 пікселів. Такий підхід виявився дуже неефективним. Машина опорних векторів показала всього 10% точності, а багатошаровий

персептрон – майже 4%. На рисунку 3.3 подано короткий аналіз у вигляді гістограми роботи машини опорних векторів і багатошарового персептрону.

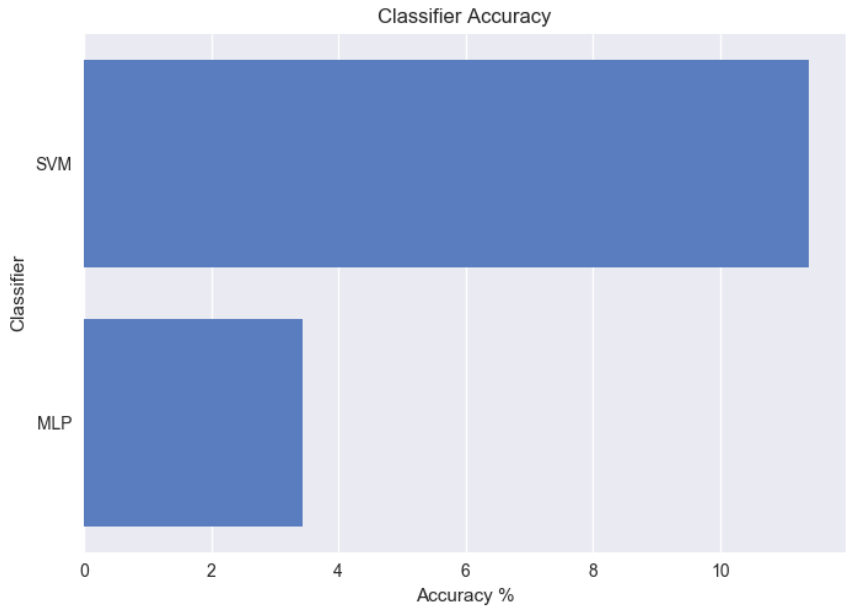

Рисунок 3.3 – Гістограма точності алгоритмів з використанням даних, отриманих із пікселів

Цей приклад ілюструє, що без правильного вилучення параметрів зображення якісна класифікація неможлива. Тому на другій ітерації реалізації використовувався підхід складання словника графічних слів та класифікації на його основі. Для цього був використаний алгоритм SIFT(Scale Invariant Feature Transform) з подальшим формуванням словника на основі отриманих даних. SIFT – це алгоритм, який працює наступним чином: визначає ключові точки на зображенні, потім перетворює ці точки в квадрати розміром 16 на 16 пікселів, після чого алгоритм обчислює градієнти для кожного пікселя (напрямок і значущість), у кінці отримується вектор параметрів, складений з 128 значень. Потім для класифікації отримані дані розбиваються на словник за допомогою будь-якого алгоритму кластеризації. Після того, як вдалося отримати словник, кожне зображення в навчальному наборі відповідає словнику, і, в кінцевому підсумку, будується новий вектор параметрів, який показує, скільки та які графічні слова представлені на

зображенні. І лише після цього цей вектор параметрів передається в алгоритм навчання для класифікації. На рисунку 3.5 представлений приклад того, як за допомогою алгоритму SIFT були вилучені параметри.

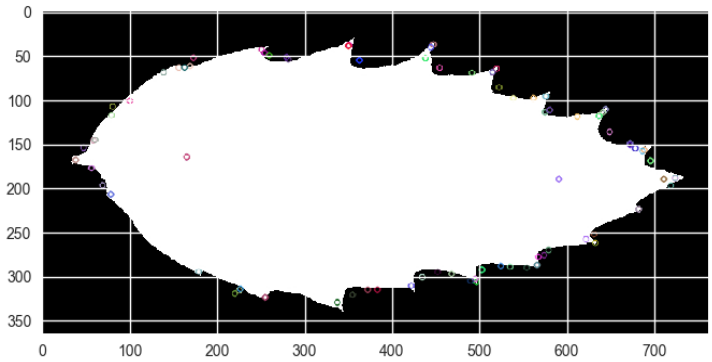

Рисунок 3.5 – Приклад вилучених параметрів за допомогою алгоритму SIFT

Після всіх виконаних дій дані параметри були передані алгоритмам навчання та отримані наступні результати, які представлені на рисунках 3.6. Машина опорних векторів показала точність 61% на контрольній вибірці, а багатошаровий персептрон склав 59%.

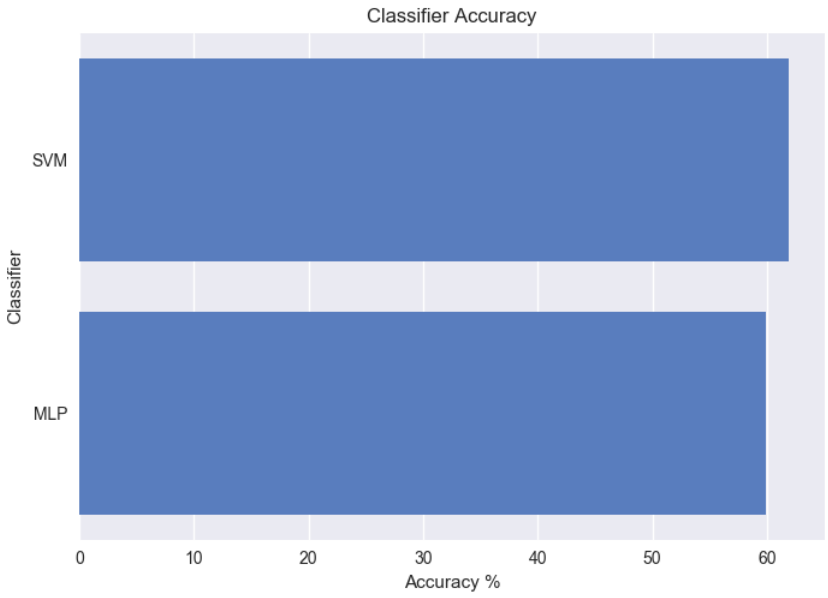

Рисунок 3.6 – Гістограма, що відображає точність роботи алгоритмів з використанням даних, отриманих за допомогою SIFT алгоритму

Результати класифікації покращилися порівняно з класифікацією на основі значень кожного пікселя, проте метод, заснований на вилученні параметрів за допомогою алгоритму SIFT та подальшої побудови словника графічних слів, також показав не дуже хороші результати. Це може бути пов’язано з особливостями даного навчального набору та вибраними зображеннями листя дерева. Справа в тому, що всі листя дуже схожі між собою і важко побудувати такий словник графічних слів, який би однозначно класифікував об’єкти. Крім того, ще однією складністю є те, що навчальний набір складається з великої кількості класів – 99. Тобто виходить, що кожному класу в наборі відповідає приблизно по 10 конкретних екземплярів. Ці причини стали перешкодою для більш точної класифікації зображень.

Далі, для підвищення значень точності класифікації був використаний інший метод вилучення параметрів зображення, який базується на перетворенні контуру листа в часовий ряд з подальшою класифікацією зображень саме за цими рядами. Такий підхід є досить популярним, і в разі представленого навчального набору він більш ніж підходить як міра відстані між двома зображеннями, які відносяться до різних класів.

За допомогою методу радіального сканування контур листа був перетворений в часовий ряд. На рисунку 3.7 показано приклад вилученого контуру та побудованого часового ряду на його основі.

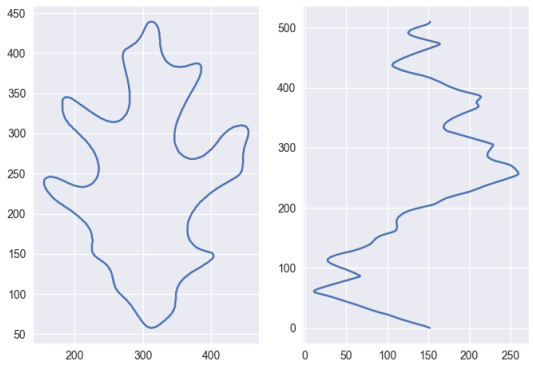

Рисунок 3.7 – Приклад перетворення контуру листа на тимчасову послідовність

Потім отримані послідовності були використані як параметри для класифікації зображень. У результаті було отримано такі результати. Машина опорних векторів показала кращу точність 87%, а багатошаровий персептрон – 81%. На рисунку 3.8 показано графік точності алгоритмів.

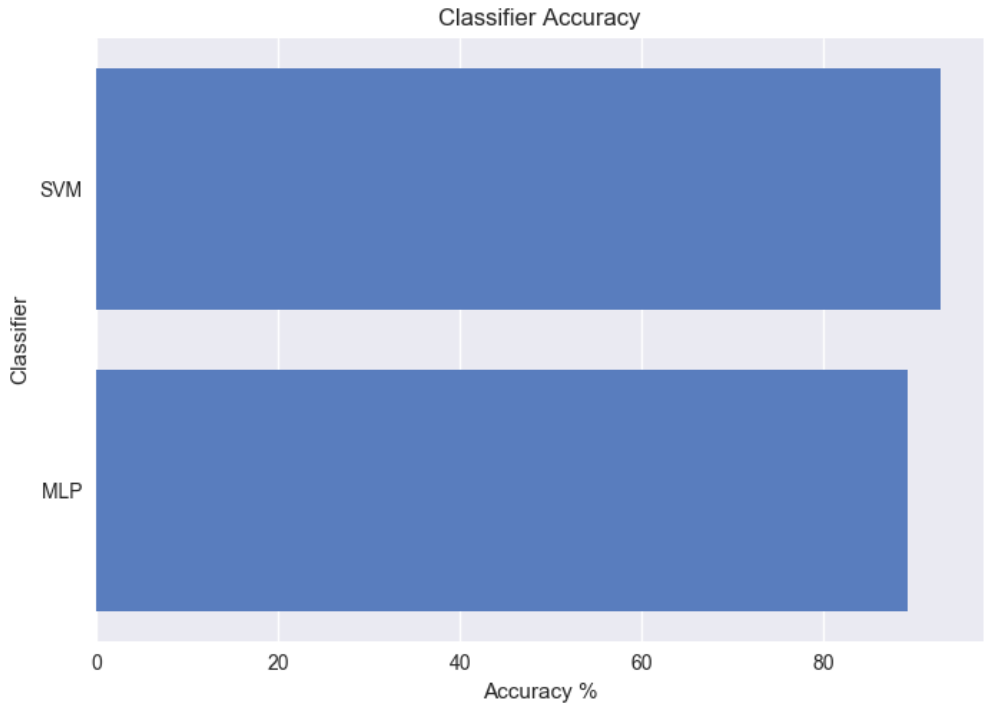

Рисунок 3.8 – Гістограма, що відображає точність роботи алгоритмів при класифікації часових послідовностей

Рисунок 3.8 наочно доводить що підсумкова точність двох алгоритмів дуже схожа. Таким чином, можна зробити висновок, що обидва ці алгоритми підходять для вирішення задачі класифікації графічних зображень.

ВИСНОВКИ

В ході виконання кваліфікаційної роботи було розроблено застосунок, який реалізує рішення задачі класифікації графічних зображень.

Під час реалізації даного застосунку були використані методи машинного навчання, спрямовані на вирішення задач класифікації. Зокрема, були проаналізовані два популярних алгоритми машинного навчання: машина опорних векторів та багатошаровий персептрон.

Обидва алгоритми показали точність більше 80%, але все ж машина опорних векторів була ефективнішою на 6% в останньому порівнянні. Слід відзначити, що машина опорних векторів навчалася швидше за багатошаровий персептрон на обраному навчальному наборі. Наприклад, під час класифікації з вилученням параметрів SIFT, машина опорних векторів навчалася за 1,39 секунди, порівняно з 15,66 секундами багатошарового персептрона. З використанням даних на основі радіального сканування були отримані такі результати: машина опорних векторів – 0,26 секунд, багатошаровий персептрон – 6 секунд.

За отриманими результатами можна відзначити, що за швидкістю навчання для розв’язання поставленої задачі на навчальному наборі Swedish leaf dataset найкращою виявилася машина опорних векторів. Крім того, машина опорних векторів простіше налаштовується. Вона має менший набір параметрів, які потрібно налаштовувати вручну для отримання хороших значень точності класифікації.

Так, наприклад, машина опорних векторів з лінійним ядром в усіх прикладах навчалася з параметром C=1, результати такого навчання були представлені вище, тоді як для багатошарового персептрона доводилося запускати додаток кілька разів з різними значеннями конфігурації прихованих нейронних шарів, щоб отримати результати точності, які максимально наближені до тих, які показувала машина опорних векторів. Для багатошарового персептрона це є досить нетривіальною задачею

правильно підібрати структуру прихованих шарів нейронної мережі. Важливим етапом вирішення задачі класифікації зображень для цих алгоритмів машинного навчання є попереднє вилучення параметрів з зображення, адже для роботи алгоритмів потрібна матриця чисел, а таких чисел зображення не містить.

Під час практичних експериментів було встановлено, що точність безпосередньо залежить від того, які дані були передані для навчання алгоритмам. Так, при використанні матриці, що складається зі значень кольору кожного пікселя зображення, були отримані наступні результати: машина опорних векторів показала час навчання більше 30 хвилин, кінцева точність становила 10%, багатошаровий персептрон навчався швидше – 22 хвилини, але його кінцева точність виявилася нижче – 4%. При використанні параметрів, отриманих за допомогою алгоритму SIFT, результати радикально змінилися. Машина опорних векторів показала час навчання 1,39 секунди і точність на рівні 61%. Багатошаровий персептрон показав час 15,6 секунд і точність 59%. Під час використання часових послідовностей, отриманих за допомогою радіального сканування, вдалося покращити результати. Наприклад, машина опорних векторів показала час 0,26 секунди і точність 87%, в той час як багатошаровий персептрон вимірювався часом 6 секунд і мав точність 81%. Таким чином, з урахуванням отриманих результатів можна зробити висновок, що при вирішенні такої задачі перш за все слід вибрати параметри для навчання класифікаційного алгоритму та обрати стратегію вилучення параметрів з зображення. Перетворення контуру листя в часову послідовність показало на практиці, що для цього конкретного набору даних цей параметр підходить краще, ніж інші які були розглянуті в даній роботі. Це пов’язано з тим, що зображений на рисунку лист володів лише однією важливою властивістю. Контури листя, які належать до одного класу, дуже схожі і не змінюються з часом, саме тому класифікатор, який обробляв цю особливість, показав найкращі результати.

Варто також відзначити і кілька зауважень щодо перспектив розвитку цієї роботи. Хоча вибірка, яка використовувалася для класифікації, і більш наближена до реальних даних з якими доведеться зіткнутися алгоритму при класифікації зображень з камер або фотографій, все ж таки вона досить спрощена. На рисунках в даній вибірці немає шумів, все листя знаходиться строго по центру зображення, палітра кольору представлена тільки двома кольорами. Всі ці нюанси роблять аналіз та попередню обробку даних зображень простіше. У наступних реалізаціях програми варто звернути увагу вже на реальніші фотографії, наприклад, різні фото дорожніх знаків, які були сфотографовано за допомогою відеореєстратора автомобіля. До того ж, для підвищення точності класифікації варто звернути увагу на алгоритм згорткової нейронної мережі. Вона має низку переваг у порівнянні з описаними в даній роботі алгоритмами. Більше того, на змаганнях із класифікації графічних зображень ImageNet Large Scale Visual Recognition Challenge в останні роки лідирують саме алгоритми побудовані на основі згорткових нейронних мереж.

Результати дослідження апробовано у вигляді тез доповідей [38].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Mohri, M., Rostamizadeh, A., Talwalkar, A. (2012). Foundations of Machine Learning, Massachusetts Institute of Technology.

2. Michie, D., Spiegelhalter, D.J., Taylor, C.C. (2009). Machine Learning, Neural and Statistical Classification.

3. O'Fallon, B.D. (2013). A support vector machine for identification of single-nucleotide polymorphisms from next-generation sequencing data Journal Bioinformatics.

4. Duda, R.O., Hart, P.E., (1973). Pattern Classification and Scene Analysis, Wiley.

5. Fletcher, R. (2008). Practical Methods of Optimization, Wiley.

6. Rumelhart, D.E., McClelland, J.L. (1986). Parallel Distributed Processing: Explorations in the Microstructure of Cognition Cambridge – MA.

7. Ye, L., Keogh, E. (2009). Time series shapelet: a new primitive for data mining, KDD `09 Proceedings of 15th ACM SIGKDD international conference on Knowledge discovery and data mining.

8. Bertsekas, D.P. (2017). Dynamic Programming and Optimal Control, Athena Scientific.

9. Reitz, K., Schlusser, T. (2016). The Hitchhiker's Guide to Python: Best Practices for Development, O'Reilly Media.

10. Müller, A.C., Guido, S. (2016). Introduction to Machine Learning with Python: A Guide for Data Scientists, O'Reilly Media.

11. Yoon, S.T., Hwang, E., (2007). A leaf image retrieval scheme based on partial dynamic time wraping and two-level filterin, CIT: 7th IEEE International Conference on Computer and Information Technology.

12. Lowe, D.G. (2004). Distinctive Image Features from Scale-Invariant Keypoints, International Journal of Computer Vision.

13. Hills, J., Baranauskas, E., Lines, J., Mapp, J., Bagnall, A. (2014). Time-series Classification with Shapelets, Journal Data Mining and Knowledge

Discovery.

14. Tvoroshenko I., Pomazan V., Gorokhovatskyi V., and Kobylin O. (2023) Application of video data classification models using convolutional neural networks, International Journal of Academic and Applied Research, 7(11), pp. 134-145.

15. Rabotiahov, A., Kobylin, O., Dudar, Z., & Lyashenko, V. (2018, February). Bionic image segmentation of cytology samples method. In 2018 14th International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET) (pp. 665-670). IEEE.

16. Кобилін, О. А., & Творошенко, І. С. (2021). Методи цифрової обробки зображень.

17. Работягов, А. В., Ляшенко, В. В., & Кобылин, О. А. (2016). Сегментация сложных изображений цитологических препаратов.

18. Lyashenko, V., Mohammad, A., & Kobylin, O. (2015). Experiments with Fusion of Images with Use of Wavelet Transformation in Problems of the Text Information Analysis.

19. Kobylin, O., Vyskrebentseva, S., & Petrova, R. (2019). Обробка даних, що містять пропуски в задачах кластеризації. Системи управління, навігації та зв'язку. Збірник наукових праць, 5(57).

20. Oleg, K., Sergii, M., & Mykhailo, S. (2017, October). Video Clustering via Multidimensional Time-Series Analysis. In Proceedings of the 9th International Conference on Information Management and Engineering (pp. 60-63). ACM.

21. Mashtalir, S., Mashtalir, V., & Stolbovyi, M. (2018, August). Representative Based Clustering of Long Multivariate Sequences with Different Lengths. In 2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP) (pp. 545-548). IEEE.

22. Bodyanskiy, Y., Kobylin, I., Rashkevych, Y., Vynokurova, O., & Peleshko, D. (2018, February). Hybrid fuzzy-clustering algorithm of unevenly and asynchronously spaced time series in computer engineering. In 2018 14th International Conference on Advanced Trends in Radioelecrtronics,

Telecommunications and Computer Engineering (TCSET) (pp. 930-935). IEEE.

23. Bodyanskiy, Y., Vynokurova, O., Kobylin, I., & Kobylin, O. (2016). Adaptive fuzzy clustering of short time series with unevenly distributed observations in Data Stream Mining tasks. Information Technology and Management Science, 19(1), 23-28.

24. Lyashenko V., Kobylin O., Selevko O. (2020) Wavelet Analysis and Contrast Modification in the Study of Cell Structures Images. International Journal of Advanced Trends in Computer Science and Engineering. 9(4). – 4701-4706.

25. Mashtalir, V., Ruban, I., & Levashenko, V. (Eds.). (2019). Advances in Spatio-Temporal Segmentation of Visual Data (Vol. 876). Springer Nature.

26. Kobylin, O., & Lyashenko, V. (2016). Contrast Modification as a Tool to Study the Structure of Blood Components.

27. Kobylin, O. A., Gorokhovatskyi, V. O., Tvoroshenko, I. S., & Peredrii, O. O. (2020). The application of non-parametric statistics methods in image classifiers based on structural description components. Telecommunications and Radio Engineering, 79(10).

28. Kobylin, O., & Lyashenko, V. (2014). Comparison of standard image edge detection techniques and of method based on wavelet transform.

29. Gorokhovatskiy, V. A., Kobylin, O. A., & Kulikov, Y. A. (2015). Application of Granulation of Feature Descriptions in Structural Image Recognition. Telecommunications and Radio Engineering, 74(6).

30. Kuzminska, O., Mazorchuk, M., Morze, N., & Kobylin, O. (2019, June). Digital learning environment of ukrainian universities: The main components to influence the competence of students and teachers. In International Conference on Information and Communication Technologies in Education, Research, and Industrial Applications (pp. 210-230). Springer, Cham.

31. Kinoshenko, D., Kobylin, O., Mashtalir, S., & Stolbovyi, M. (2019, March). Metric video retrieval speedup by irrelevant data elimination. In Eleventh International Conference on Machine Vision (ICMV 2018) (Vol. 11041, pp. 176-183). SPIE.

32. Бодянский, Е. В., Винокурова, Е. А., Пелешко, Д. Д., Кобылин, И. О., & Кобылин, О. А. (2017). Нечёткая кластеризация временных рядов с неравномерными и асинхронными тактами квантования. Системи обробки інформації, (5), 47-54.

33. Lyashenko, V., Matarneh, R., Kobylin, O., & Putyatin, Y. (2016). Contour detection and allocation for cytological images using Wavelet analysis methodology.

34. Lyashenko, V., Kobylin, O., & Ahmad, M. A. (2014). General methodology for implementation of image normalization procedure using its wavelet transform.

35. Gorokhovatskyi V., Tvoroshenko I., Kobylin O., and Vlasenko N. (2023) Search for visual objects by request in the form of a cluster representation for the structural image description, Advances in Electrical and Electronic Engineering, 21(1), pp. 19-27.

36. Yakovleva, O., Kovtunenko, A., Liubchenko, V., Honcharenko, V., & Kobylin, O. (2023). Face Detection for Video Surveillance-based Security System (COLINS-2023). In CEUR Workshop Proceedings (Vol. 3403, pp. 69-86).

37. Tvoroshenko I., Gorokhovatskyi V., Kobylin O., and Tvoroshenko A. (2023) Application of deep learning methods for recognizing and classifying culinary dishes in images, International Journal of Academic and Applied Research, 7(9), pp. 57–70.

38. Афанасьєв А., Кобилін О. (2023) Порівняльний аналіз методів машинного навчання для вирішення задачі класифікації щодо розпізнавання графічних зображень, *Proceedings of IV International Scientific and Practical Conference «Current challenges of science and education», (December 11 – 13, 2023). Berlin, Germany,* pp. 236-241.