МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ЦЕНТР ЗАОЧНОЇ, ДИСТАНЦІЙНОЇ ТА ВЕЧІРНЬОЇ ФОРМ НАВЧАННЯ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК
СЕКЦІЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРОЕКТУВАННЯ

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: «Інтелектуальна система розпізнавання дефектів стічних труб»

за спеціальністю 122 «Комп'ютерні науки»,
освітньо-професійна програма «Інформаційні технології проектування»

Виконавець роботи: студент групи ІТ.мз-92с Москаленко В'ячеслав Васильович

Кваліфікаційну роботу захищено на засіданні ЕК з оцінкою _____ «___» лютого 2021 р.

Науковий керівник _____ завідувач кафедри, д.т.н., професор, Довбиш А.С.
(підпис)

Голова комісії _____ Шифрін Д.М.
(підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Суми-2021

Сумський державний університет
Центр заочної, дистанційної та вечірньої форм навчання
Кафедра комп'ютерних наук
Секція інформаційних технологій проектування
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
Зав. секцією ІТП

_____ В. В. Шендрик
«___» _____ 2020 р.

ЗАВДАННЯ

на кваліфікаційну роботу магістра студентові

Москаленко В'ячеслав Васильович
(прізвище, ім'я, по батькові)

1 Тема проекту *Інтелектуальна система розпізнавання дефектів стічних труб*
затверджена наказом по університету від «16» листопада 2020 р. №1773-III

2 Термін здачі студентом закінченого проекту «_25__» ___січня___2021 р.

3 Вхідні дані до проекту вибірка зображень інспекції стічних труб та розмітка дефектів

4 Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) аналіз проблеми та постановка задачі, інформаційна технологія розпізнавання дефектів стічних труб, реалізація інформаційної системи розпізнавання дефектів стічних труб.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

_____

6. Консультанти випускної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

Дата видачі завдання _____.

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів випускного проекту	Термін виконання етапів проекту	Примітка
1	Аналіз проблеми	20.05.20-26.06.20	
2	Формування мети і задач	23.05.20-26.05.20	
3	Розробка інформаційної технології розпізнавання дефектів стічних труб	27.05.20-12.06.20	
4	Реалізація інформаційної системи розпізнавання дефектів стічних труб	13.06.20-30.06.20	
5	Тестування та завершення роботи	23.06.20-29.06.20	
6	Оформлення пояснювальної записки	10.11.20-01.12.20	

Магістрант _____ Москаленко В.В.

Керівник роботи _____ зав. кафедри, д.т.н., професор Довбиш А.С.

РЕФЕРАТ

Тема кваліфікаційної роботи магістра «Інтелектуальна система розпізнавання дефектів стічних труб».

Пояснювальна записка складається зі вступу, 3 розділів, висновків, списку використаних джерел із 20 найменувань, 1 додатку. Загальний обсяг роботи − 66 сторінок, у тому числі 52 сторінок основного тексту, 3 сторінок списку використаних джерел, 10 сторінок додатку. Пояснювальна записка включає в себе 24 рисунків та 2 таблиці.

Кваліфікаційну роботу магістра присвячено розробці інформаційної системи розпізнавання дефектів стічних труб на зображеннях відоінспекції. В роботі проведено аналіз проблеми розпізнавання функціонального стану стічних труб та моделій і методі існучих алгоритмів штучного інтелекту. Були сформульовані мета та задачі, вирішення яких необхідне для її досягнення.

У роботі виконано розробку інтелектуальної інформаційної системи, а саме запропоновано модель і метод машинного навчання, сформновано навчальну та тестову вибірки, здійснено навчання та тестування моделі. Виконано короткий опис програмної реалізації інформаційної системи.

Результатом проведеної роботи є розроблені модель та метод машинного навчання для розпізнавання дефектів стічних труб, програмна реалізація та результати валідації на тестових даних.

Практичне значення роботи полягає у підвищенні рівня автономності процесу відомоніторингу стічних труб і підвищення точності прийняття рішень у порівняння з традиційним підходом.

Ключові слова: стічні труби, відео інспекція, розпізнавання дефектів, згорткові нейронні мережі, інформаційно-екстремальна інтелектуальна технологія.

ЗМІСТ

ВСТУП

Стічні труби є важливим інфраструктурним об’єктом, що потребує частого моніторингу. Традиційним підходом до діагностики стічних систем є відеомоніторинг з кодуванням всіх знайдених дефектів згідно існуючих стандартів. Поширеними стандартами є британський стандарт MSCC5 та розроблені на основі нього американські стандарти PACP6 та PACP7. Кодування стану труб в рамках даних стандартів вимагає уважного перегляду і детального аналізу даних відео обстеження. Одним із шляхів підвищення продуктивності та зниження вартості інспекції є використання моделей і методів машинного зору та машинного навчання.

Сучасним напрямком удосконалення моделей аналізу візуальних даних є використання глибоких моделей аналізу даних, які забезпечують ефективну апроксимацію складних залежностей. При цьому деякі класи дефектів труб зустрічаються рідко, але мають високу варіативність, що призводить до незбалансованості даних і малого обсягу розмічених зразків для покриття складних дефектів. Це обмежує використання традиційних методів глибокого навчання, які потребують великих обсягів репрезентативних розмічених даних. Тому розроблення ефективних глибоких моделей аналізу даних за умов обмеженого обсягу розмічених даних є актуальною задачею для підвищення рівня автоматизації моніторингу стічних труб.

Головним завданням магістерської роботи є аналіз існуючих підходів до аналізу даних відеомоніторингу стічних труб, а також розробка моделей та методів аналізу даних для інтелектуальної системи розпізнавання дефектів стічних труб на основі глибокого машинного навчання.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Сучасний стан та тенденції розвитку технологій аналізу даних інспекції стічних труб

Проблеми з старими каналізаційними мережами стали головним викликом для муніципалітетів по всьому світу, оскільки вони досягають кінця своєї проектної експлуатації. При цьому, посилені вимоги та обмеження бюджету ще більше ускладнюють вирішення цього питання [1]. Крім того, через погіршення стану каналізаційних мереж протягом терміну їх служби, навіть якщо вони сьогодні перебувають у дуже хорошому стані, вони все одно потребуватимуть все більших інвестицій із часом внаслідок старіння. Щоб уникнути подальших важких та вартісних пошкоджень, необхідно забезпечити своєчасний та всеосяжний контроль за станом каналізаційних систем через регулярні інспекції.

Підземну інфраструктуру складно інспектувати, оскільки вона прихована під землею. Така ситуація робить витрати на ремонт та заміну каналізаційних труб досить високими. Прямий людський огляд підземних інфраструктур не є практичним, на відміну від доріг та тунелів, через велику кількість занурених під землю труб та мереж, а також через небезпечне та шкідливе для здоров'я середовище стічної системи, обмежену видимість та малі розміри труб. Згідно з програмою сертифікації оцінки стану трубопроводів (PACP), у системах водовідведення існує велика кількість потенційних дефектів, і виявити їх усіх за допомогою традиційних методів може виявитися неможливим [2]. Дефекти труб водовідведення каналізації можна класифікувати на структурні, конструктивні (операційні) та технічного обслуговування (рис. 1.1).

Вказані вище виклики завжди були головними мотиваторами для розробки більш вдосконалених методів інспекції стічних труб. Розвиток сенсорних та оптичних технологій полегшив та прискорив впровадження та удосконалення нових способів виявлення дефектів.

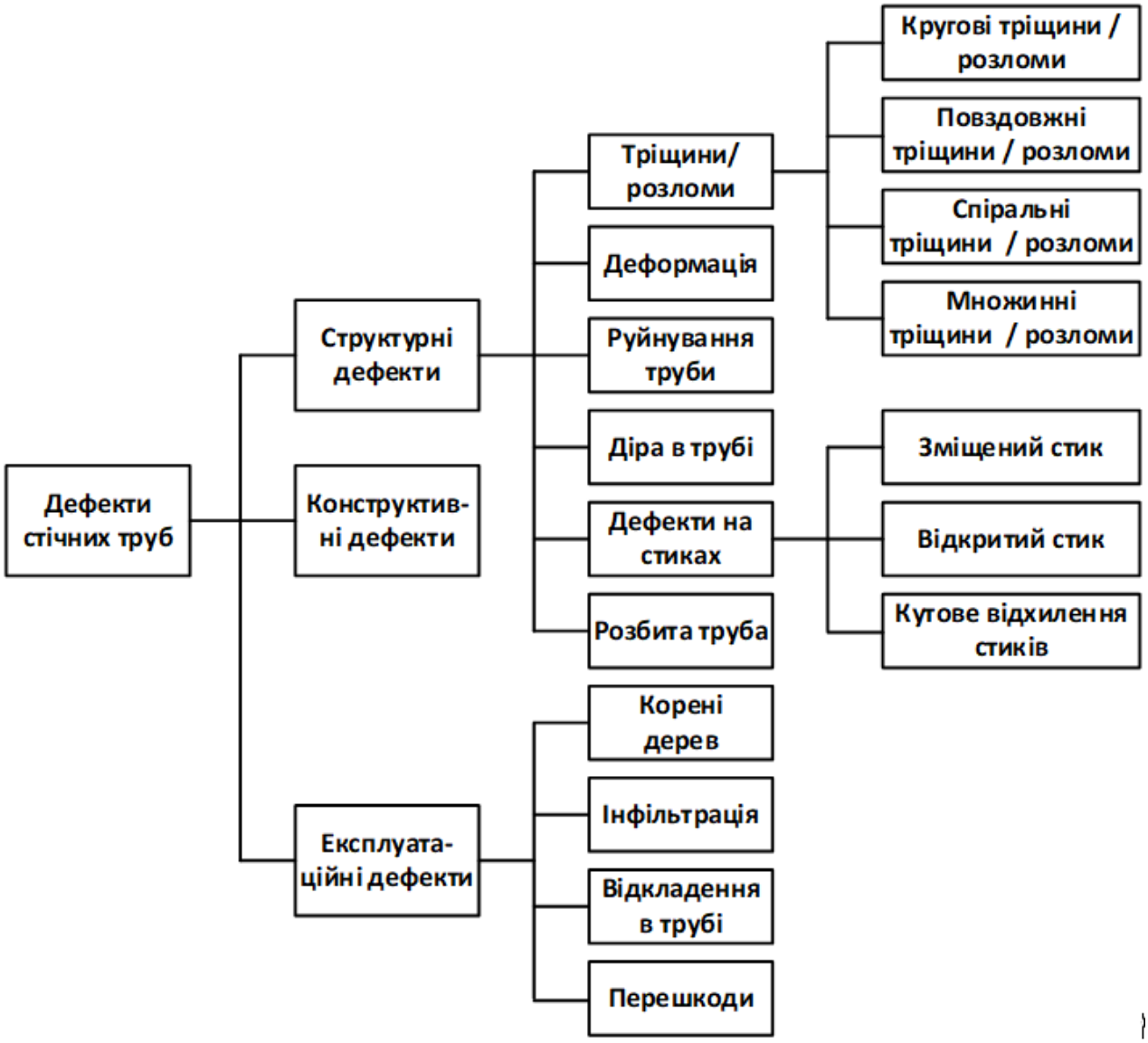


Рисунок 1.1 – Основні типи дефектів стічних труб згідно стандарту PACP

Методи перевірки стану стічних труб можуть бути класифіковані на чотири категорії. Перша включає технології, основані на візуальному спостереженні, які використовують відеокамеру і повністю залежать від її здатностей. Друга група охоплює структурні та геотехнічні інструменти, призначені для перевірки цілісності труб та стану грунту навколо них. Третя категорія включає спеціалізовані технології для певних типів дефектів. Наостанок, четверта група складається з гібридних технологій, які комбінують два або більше інструменти (рис. 1.2) [4].

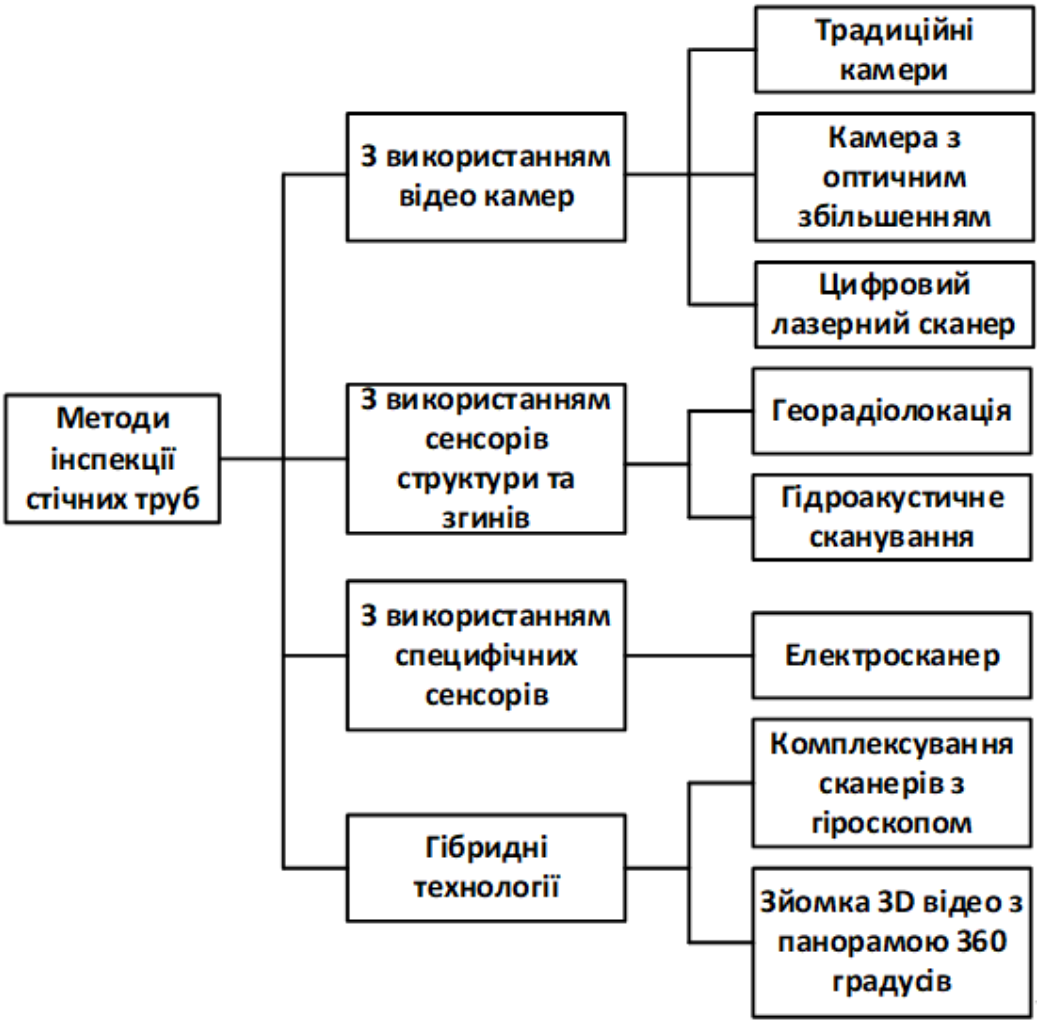


Рисунок 1.2 – Види інструментів інспекції трубопроводів

Інструменти інспекції на основі камер є одним із найбільш зручних і придатних методів контролю при оцінці стану систем збору стічних вод. Відеоспостереження залишається найпоширенішим в практиці інспекції каналізаційних трубопроводів, незважаючи на відсутність геометричних орієнтирів, суб'єктивне судження та залежність від кваліфікації оператора, якості зображення та якості освітлення. Сучасні досягнення в технологіях зйомки та візуалізації привели як до зниження вартості використання камер, так і до якості отримуваної інформації. Муніципалітети, як правило, цікавляться звітами про інспекцію, що включають карту дефектів і підтверджуючі візуальні дані для прийняття рішень. Кожен візуальний доказ може містити велику кількість інформації, таку як текстура, форма, розташування, тяжкість тощо. При цьому варто зауважити, що апаратне забезпечення систем візуальної інспекції стічних труб досягло рівня насичення технічною новизною і подальші

удосконалення стосуються пдвищення рівня автоматизації аналізу візуальної інформації.

В роботах [1, 3] було висунуто пропозицію щодо методу виявлення регіонів інтересу, зокрема тих, які містять тріщини, під час роботизованої інспекції підземних труб. Пропонований підхід базується на застосуванні алгоритму Canny для виявлення країв, який є стандартним інструментом в області комп'ютерного зору. Такий метод вимагає ручного налаштування параметрів для відповідного типу труб та не бере до уваги контекстуальну інформацію, що ускладнює його практичне застосування. Також відсутній детальний аналіз виявлених регіонів інтересу для їхнього кодування згідно із діючими стандартами інспекції труб.

У працях [4, 5] запропоновано використовувати фільтри Габора (Gabor filter) для екстракції ознак, що описують спостереження під час інспекції сточних труб у поєднанні з класичними моделями машинного навчання (рис. 1.3).

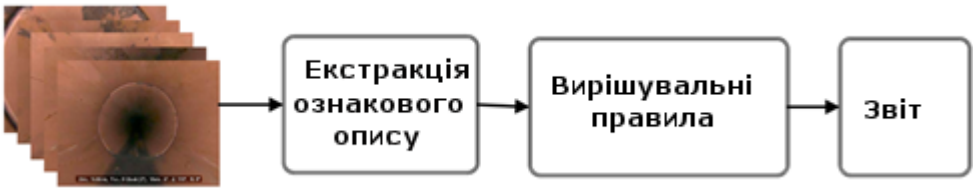


Рисунок 1.3 – Ілюстрація етпаів аналізу даних в рамках класичного підходу

Покращення інформативності екстрагованих ознак з використанням фільтрів Габора потребує значно більше обчислювальних ресурсів у порівнянні з алгоритмами побудови ієрархії ознакового опису. Крім того, популярний алгоритм випадкового лісу не дозволяє проводити деталізований аналіз з одночасним урахуванням контекстуальної інформації. Чітке розмежування моделі на екстрактор вектора ознак та класифікатор векторів ознак вносить певні обмеження і не дозволяє автоматизувати процес навчання цілої моделі за методом end-to-end.

В роботах [5, 6] було висунуто пропозицію щодо застосування глибоких нейронних мереж для аналізу візуальних даних, які надають ієрархічне представлення ознак з необхідним рівнем інваріантності та інформативності (див. рис. 1.4).

Рисунок 1.4 – Підхід до аналізу даних на основі глибокого машинного навчання

Використання глибокого машинного навчання придатне не усіх випадках. Дана технологія вимагає вибору архітектури мережі, оскільки від вибору архітектури залежить вимогливість до обсягу розмічених даних та узагальнююча здатність моделі.

Таким чином, вибір архітектури моделі аналізу даних виманає аналізу процесу інспекції і його декомпозиція на задачі, кожну з яких можна ефективно вирішити з використанням багатошарової нейронної мережі за умов обмеженого обсягу розмічених даних.

1.2 Аналіз методів розпізнавання зображень

Ефективність розпізнавання складних візуальних об’єктів залежить від способу ознакового опису зображень. При цьому питаннями формування ознакового опису займається наука про подання даних або наука про навчання ознакового опису. В даній галузі дослідники сформульовали основні принципи, що лежать в основі ефективного ознакового подання спостережень. Ці принципи є універсальними та дають основу для проектування систем інтелектуального аналізу даних різного призначення. До таких принципів належать [7]:

– множинність пояснювальних факторів (першопричин);

– ієрархічна організація пояснювальних факторів (першопричин);

– простота залежності високорівневого подання факторів;

– спільність факторів під час розв’язання різних задач;

– гіпотеза багатовидів;

– гіпотеза про природну кластеризацію;

– навчання з частковим залученням вчителя;

– просторово-часова зв’язаність;

– розрідженість ознакового подання.

Вхідні дані відбивають результат впливу багатьох незалежних пояснювальних факторів. Тому врахування кожного нового фактору потребує його узагальнення у конфігурації з іншими факторами. На цьому базується ідея розподіленого подання вхідних даних. Для кодування різних спостережень чи частин спостереження можна повторно використовувати кожен параметр моделі, навіть якщо ці спостереження не є близькими сусідами в просторі категорій. На відміну від алгоритмів з локальним узагальненням розподілене подання забезпечує активацію експоненційно більшої кількості ознак (або прихованих змінних) вхідним сигналом. У випадку локального узагальнення, локальні частини вхідного простору асоціюються з персональним набором параметрів, що обмежує ємність моделі подання даних [7, 8].

Принцип ієрархічної організації пояснюючих факторів полягає в тому, що більш складні поняття формуються з простих, тобто ієрархічно. Тобто високорівневий ознаковий опис утворюється шляхом композиції низькорівневих ознак розпізнавання (рис. 1.1). Функція f, що описує екстрактор ознак, у загальному випадку є композицією з n шарів трансформації простору ознак $f = f1 \circ f2 \circ ::: \circ fn$. При одній і тій же кількості параметрів ієрархічні моделі порівняно з однорівневими характеризуються більшою інформаційною ємністю. При цьому ознакове подання може будуватися з метою вирішення одного з двох основних завдань : формування інваріантноного ознакового опису або розділення пояснюючих факторів. Інваріантні ознаки повинні бути максимально не чутливі до неінформативних і нецікавих для подального оброблення змін вхідних даних. В результаті розділення пояснювальних факторів формуються множина незалежних ознак, тобто факторів. Такі ознаки

можуть бути корисними для ситуацій апріорної невизначеності щодо задачі та відповідно інформативності ознак, тобто отриманий ознаковий опис зручний для повторного використання.

Чим простіші залежності між сформованими ознаками і цільовою змінною тим ефективнішим можна вважати сформаований ознаковий опис. Тому синтез вирішувальних правил потрібно здійснювати на основій набільш обчислювально ефективних підходів – лінійні правила чи система порогів. Це дозволяє прискорити машинне навчання та визначити необхідність удосконалення екстрактора ознак. Виключенням є випадок багатозадачної моделі, де проміжний шар прихованих змінних потрібний для узгодження обох задач без реінжинірингу екстрактора ознак.

Гіпотеза пророднього кластеризації даних полягає в тому, що спостереження об’єднують людиною в клас на основі їх статистичної схожості. Тому локальні зміни в багатовидах як праивло не призводять до зміни класу, а лінійна ітерполяція між спостереженнями різних класів проходить через області зі зниженою щільністю ймовірності.

Для ефективного використання нерозмічених навчальних зразків в задачах начання з учителем можна використати статистичний зв'язок з навчанням без вчителя. На цьому основано ідею навчання з частковим залучення учителя, згідно якої роздільна гіперповерзня між вибірками різних класів проходить через області простору ознак з низькою щільністю ймовірності [6]. Ще одним шляхом підвищення інформативності ознакового опису є посилення деяких факторів, що є спільними для декількох задач, шляхом навчання багатозадачної моделі. Даний підхід лежить в основі методу переносу знань до наступних задач.

Гіпотеза багатовидів є основою для методів зниження розмірності простору ознак. Дана гіпотеза полягає в тому, що основна щільність ймовірності зосереджена в околі регіонів набагато меншої розмірності ніж оригінальний простір ознак [7, 9]. Якщо вибірка має обмежений обсяг, то для уникнення проблеми «прокляття розмірності», що призводить до збільшення

простору розсіювання даних, слід позбавлятися від надлишкових ознак (рис. 1.5).

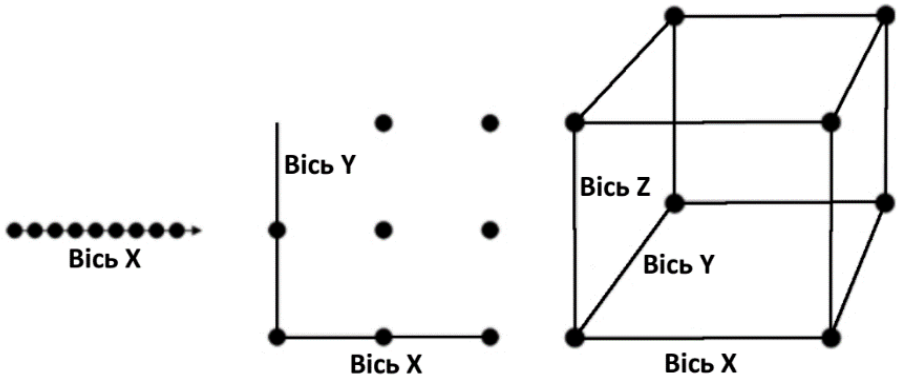


Рисунок 1.5 – Ілюстрація збільшення розсіювання точок даних внаслідок збільшення кількості вимірів – проблема «прокляття розмірності»

Для боротьби з проблемою «прокляття розмірності» було розроблено алгоритми, що грунтуються на ядрах (kernel machines). Однак в основу цих алгоритмів покладені властивість гладкості цільової функції та лінійна залежність факторів. Однак якщо на вхід алгоритму надходять «сирі» дані, то замість гладкості функції ми отримаємо множину піків і впадин, що може зростати експоненційно з кількістю впливаючих факторів. Тому ядерні методи придатні лише для попередньо підготованого ознакового простору.

Під час формування моделі аналізу даних необхідно максимально враховувати всю доступну апріорну інформацію. До такої можна віднести і обмеження на зміни даних «крізь час і простір». Сусідні ділянки даних в просторі чи часі повинні відповідати тим самим категоріям або викликати невеликий рух по поверхні багатовиду. Одним і шляхів реалізації цього обмеження є введення регуляризуючої складової для штрафування за різницю ознак в послідовні моменти часу чи обмеження розміру локальних рецептивних полів нейронів. Крім того цей сами прийнцип обгрунтовує розширення навчальних наборів даних методами аугментації, що основані на допустимих викривленнях.

Розрідженість ознакового подання є ще однією характеристикою біологічного прототипу, яку успішно моделюють в моделях аналізу даних. Розрідженість полягає в тому, що більшість ознак спостереження близькі до

нуля, тобто невелика кількість факторів є знаничимими. Дана властивість породжує ефект редукції причини, коли дві апріорно не пов'язані причини активують один фактор, якщо з'являється спостережння однієї події. Розрідженість досягають за рахунок різного роду регуляризації. При цьому сам процес пошуку розрідженого коду є складною нелінійною задачею [7].

Зниження розмірності вхідного простору ознак та формування інваріантного ознакового подання дослідниками було запропоновано аріхтектуру сіамських нейромереж [10]. Модель використовує два ектрактори ознак зі спільними параметрами (рис. 1.6).

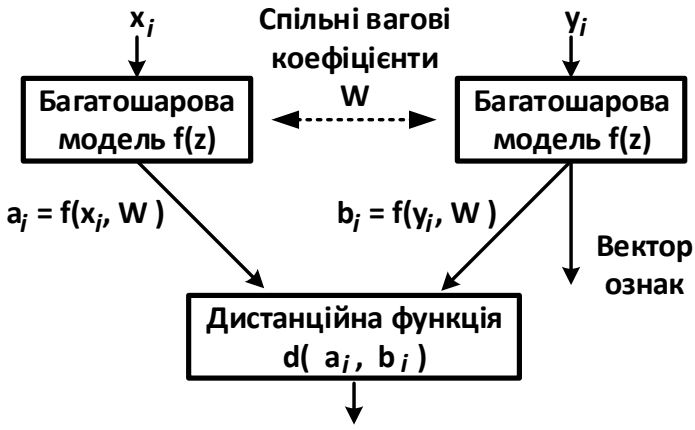


Рисунок 1.6 – Структура моделі сіамської мережі

Процес оптимізації параметрів W полягає у одночасній мінімізації відстані між зразками одного класу та максимізації відстані між зразками різних класів. При цьому для семантично однакових спостережень вихід мереж повинен бути близький до 0, а для семантично різних спостережень вихід мережі повинен бути близький до 1. Як функція втрат під час навчання використовуюь контрастну, триплентну функції чи їх модифікаці. Тобто навчена сіамська нейромережа може інтерпретуватися як функція (метрика) подібності для алгоритму найближчих сусідів чи як ядро для ядерних методів побудови вирішувальних правил. Екстрактор ознак сіамської мережі є повномінною мережею, що може повторно використовуватися як embedding для споріднених задач.

Найвищих результатів у задачах інтелектуального оброблення зображень

вдалося досягти за допомогою згорткових нейронних мереж. У цих моделях, кожен нейрон має обмежене рецептивне поле і послідовно сканує зображення, тобто кожен нейрон не зв’язаний з усіма пікселями. Такий підхід відповідає принципу просторової зв’язаності в 2D та 3D топології. Це дозволяє істотно знизити кількість параметрів для оптимізації і обчислювальну складність екстрації ознак, завдяки повторному використанню ваг (параметрів) нейронів.

За останні 30 років було розроблено велику кількість архітектурних вдосконалень згорткових моделей штучної нейронної мережі. Покращення торкнулися алгоритмів оптимізація параметрів, регуляризація мережі, структурних елементів тощо.

Останні досягнення в машинному зорі пов’язані з удосконаленням згорткових мереж за рахунок структурних змін існуючих та розробки нових блоків або модулів на мікро і макрорівні (рис. 1.7).

В залежності від типу архітектурних змін, згорткові мережі можна розділити на декілька різних груп [12]:

– використанні фільтрів різного масштабу, що також називають проторовими модифікаціями (spatial exploitation);

– збільшення кількості шарів (depth exploitation), тобто заходи щодо боротьби з затуханням градієнту на нижніх шарах;

– використання множинних шляхів в згортковій нейромережі (multi-path exploitation) за рахунок з’єднань з пропуском шарів та регуляризації на зразок dropout і drop-connection;

– збільшення кількості вузлів (фільтрів) і з’єднань на рівні кожного шару (width exploitation), що розглядається як збільшення ширини нейронної мережі;

– впровадження методів зважування ознак (feature map exploitation), що дозволяє підвищити їх інформативність, наприклад, шляхом використання модуля Squeeze-Excitation;

– додавання нових каналів (channel boosting exploitation) з різних джерел на основі переносу знань, агрегації результату генеративних моделей, нових модельностей та інше;

– застосування м’яких масок для фокусування уваги на інформативних ділянках зображення або карти ознак (attention exploitation), що дозволяє зменшити кількість хибних кореляцій.

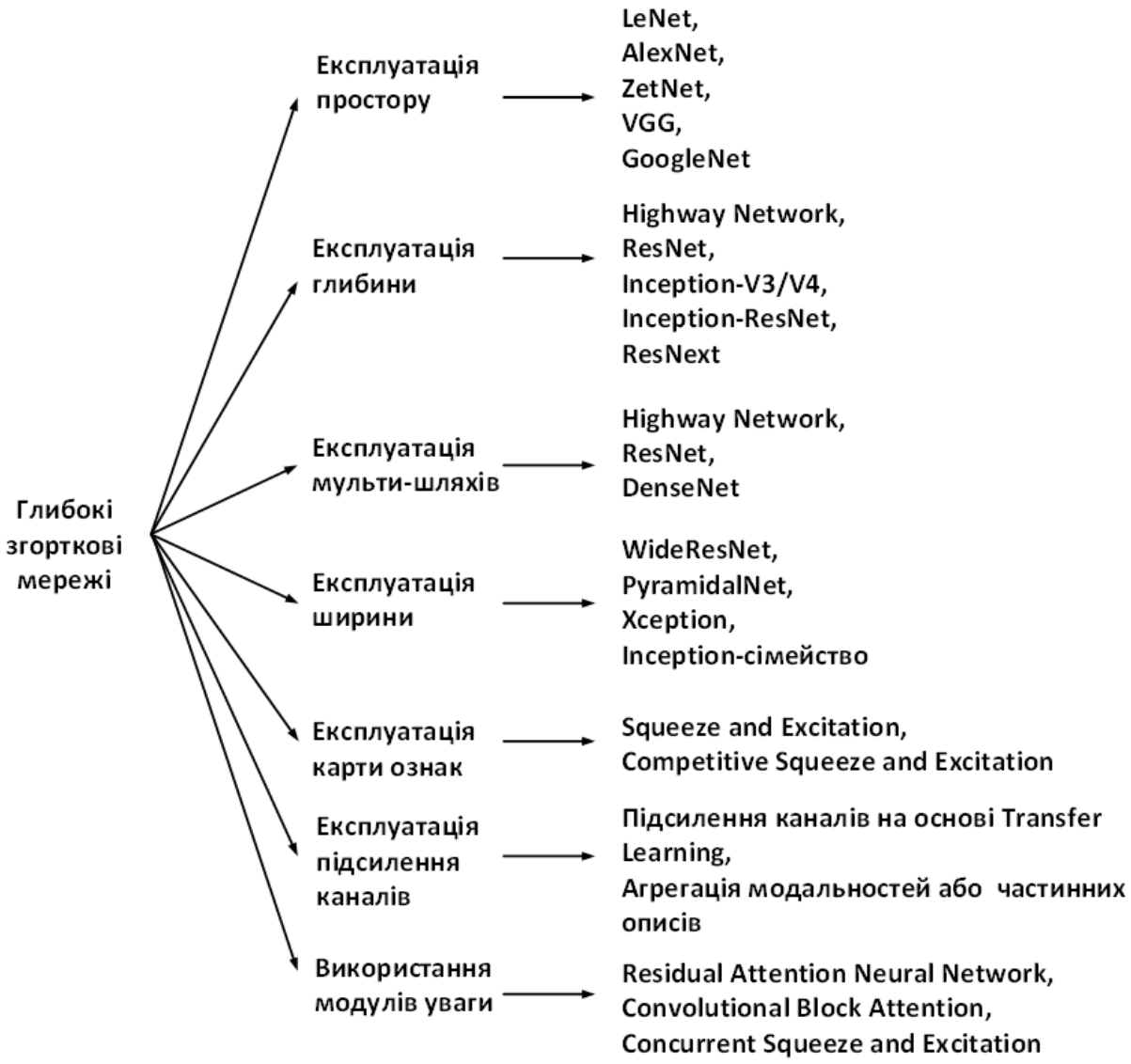


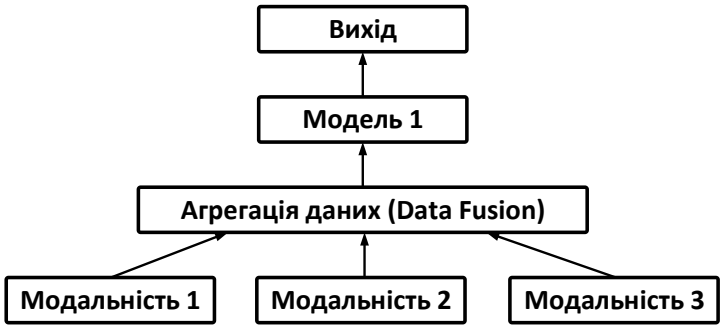
Рисунок 1.7 – Таксономія багатошарових згорткових нейронних мереж за способом їх удосконалення

Для аналізу багатоспектральної або багатомодальної візуальної інформації використовуються методи об’єднання (fusion) джерел різної модальності/спектру для їх обробки в рамках єдиної моделі. Прикладом подібної задачі є розпізнавання об’єктів на сцені, яка фіксується кольоровою (RGB) камерою, тепловізором та короткохвильовим радаром. Існує три основні методи об’єднання модельностей [13]:

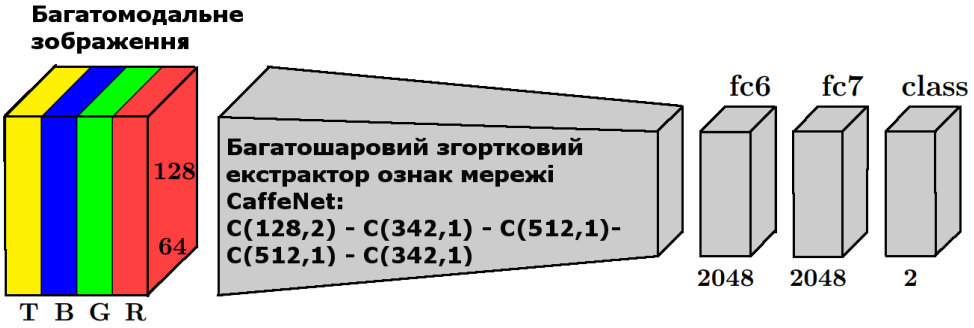
– раннє об’єднання (Early Fusion) (рис. 1.8);

– пізнє об’єднання (Late Fusion) (рис. 1.9);

– проміжне об’єднання (Intermediate fusion) (рис. 1.10).



а



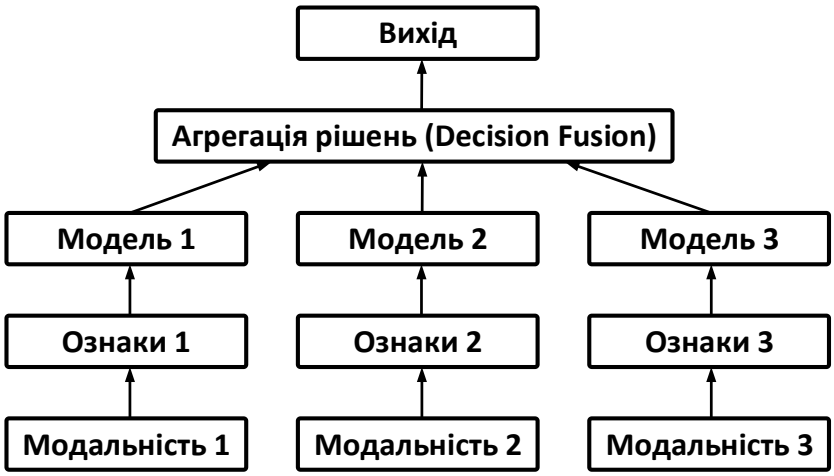
б

Рисунок 1.8 – Ілюстрація методу раннього об’єднання модальностей : *а* – загальний випадок; *б* – приклад згорткової мережі з додатковим тепловим каналом на її вході

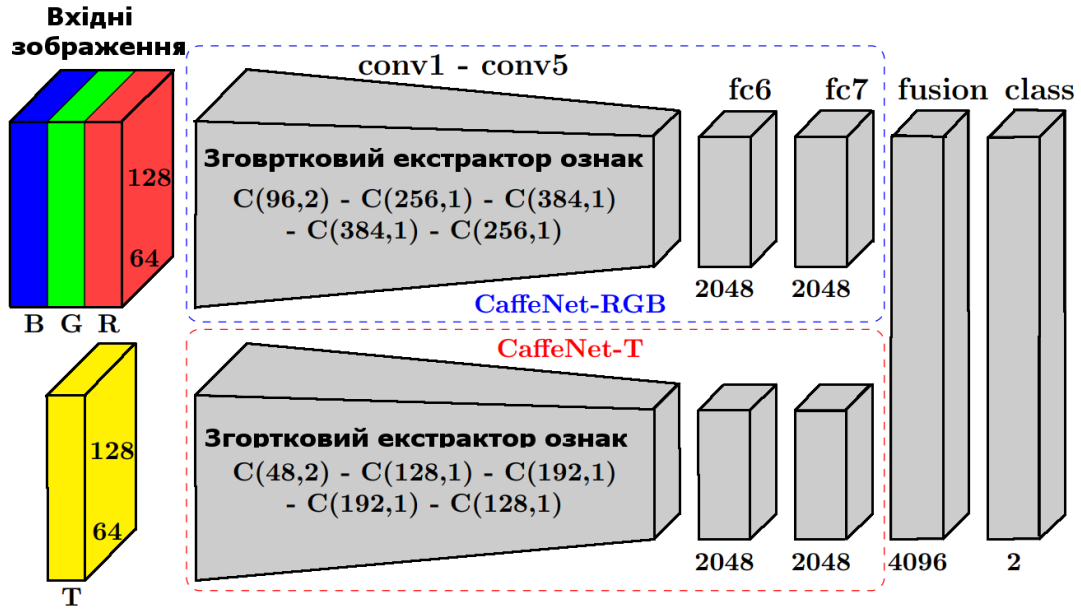
Звичайно існують більш складні способи об’єднання джерел інформації та комплексуваня моделей. На рис. 1.10 проілюстровано змішану схему, або схему з проміжною агрегацією модальностей.

Для локалізації об’єкта інтересу на зображенні використовують моделі і методи детектування, що використовують сформований екстрактор ознакового опису зображень. Найпростішим методом детектування об'єкта інтересу на зображенні є алгоритм ковзного вікна (sliding window). Однак, у цього методу є певний недолік. Проходження вікном всіх положень на зображенні на різних масштабах вимагає досить великої кількості часу O(h*w), де h і w – висота і ширина зображення. Для прискорення алгоритму сканування зображення було запропоновано модифікацію методу sliding window – метод адаптивного кроку ковзного вікна (англ. Run-Time Adaptive Sliding Window, RASW). RASW –

ефективний метод зміни кроку руху скануючого вікна під час обробки без втрати точності. Алгоритм дозволяє знизити витрати часу на обробку областей зображення, які не містять шуканих об'єктів, за рахунок збільшення кроку сканування залежно від ймовірності належності об'єкта інтересу [14].



а



б

Рисунок 1.9 – Ілюстрація методу пізнього об’єднання модальностей : *а* – загальний випадок; *б* – приклад агрегації виходів двох згорткових мереж зі входами різної модальності

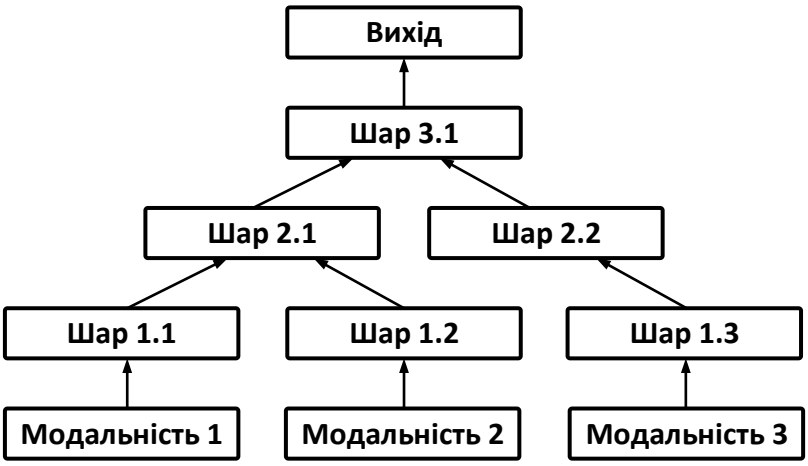


Рисунок 1.10 – Ілюстрація методу проміжного об’єднання модальностей у загальному випадку

Розвиток сучасних алгоритмів детектування об'єктів, заснованих на згорткових нейромережах, розгалужується в два ключові напрямки (рис. 1.11):

– нейромоделі з формувачем пропозицій областей інтересу, які в наступних модулях розпізнаються для визначення класу;

– нейромоделі, що роз’язують задачу з кінця-в-кінець шляхом застосування класифікаційних і регресійних виходів на кінці мережі, які одночасно забезпечують визначення категорій і обмежувальних рамок об’єктів.

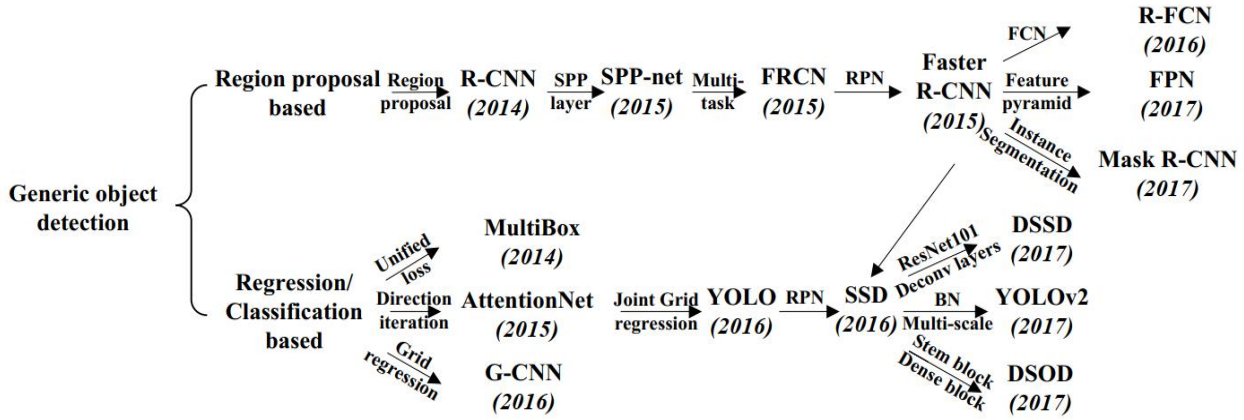


Рисунок 1.11 – Схема розвитку моделей детектування об’єктів на зображенні з використанням згорткових нейронних мереж [13]

До методів з генертором пропозиції регіонів відносяться R-CNN, SPP-мережа, Fast R-CNN, Faster R-CNN, R-FCN, FPN та Mask R-CNN. До методів, основаних на регресії і класифікації, відносяться MultiBox, AttentionNet, G-CNN, YOLO, SSD, YOLOv2, DSSD та DSOD. При цьому найбільш

популярними є три архітектури : SSD, YOLOv2 та Faster R-CNN. Для забезпечення високої точності детектування, зокрема локалізації, модель Faster-RCNN потребує менше ресурсів ніж модель SSD. Однак для задач, які не вимагають високої точності локалізації чи аналізу малорозмірних об’єктів, моделі SSD мають перевагу в швидкодії.

Архітектури моделей поступово досягають насичення технічною новизною. При цьому домінуючим методом машинного навчання є застосування алгоритму зворотного поширення помилки, проте він і має ряд недоліків:

– повільна збіжність;

– потреба значного обсягу розмічених навчальних даних та обчислювальних ресурсів для забезпечення прийнятного результату;

– невизначеність щодо оптимальних значень гіперпараметрів;

– резервування інформаційної ємності відбувається за рахунок гіперпараметризації, що збільшує накладні витрати і обумовлює схильність до перенавчання.

Одним із шляхів підвищення ефективності машинного навчання є гібридизація і комплексування різних підходів і протоколів оптимізації параметрів моделі аналізу даних. Активно досліджуються альтернативні функції втрат, методи регуляризації та пошукової оптимізації. Основною метою є зниження трудомісткості навчання з одночасним підвищення точністних характеристик за умов обмеженого обсягу розмічених навчальних даних.

В праці [10] було показано, що використання дискретного двійкового подання даних забезпечує регуляризуючі та метарегуляризуючі ефекти. Даний підхід є ефективний з точки зору боротьби з перенавчанням і трудомісткості побудови вирішувальних правил. Схожий підхід розроблено під час розв’язання задач багатокласової класифікації шляхом її заміни еквівалентною множиною двох-класових задач. Одним з найбільш ефективних методів зведення багатокласової класифікації до серії двохкласових є двійкове

кодування міток класів кодами, що виправляють помилки (Error Correcting Output Codes, ECOC) [12].

Коди, що виправляють помилки, використовуються для кодування класів. Для цілей навчання мітку чи номер класу записують у вигляді двійкового числа з C-розрядів. Кожен розряд може відповідати виходу окремої моделі бінарного класифікатора чи одному з виходів одієї моделі. Навчена модель (моделі) формуватимуть на виході двійковий код, який можна порівнювати з двійковим кодом кожного з класів. Вхідний зразок відноситься до того класу до якого мінімальна відстань Хемінга (рис. 1.12) [7].

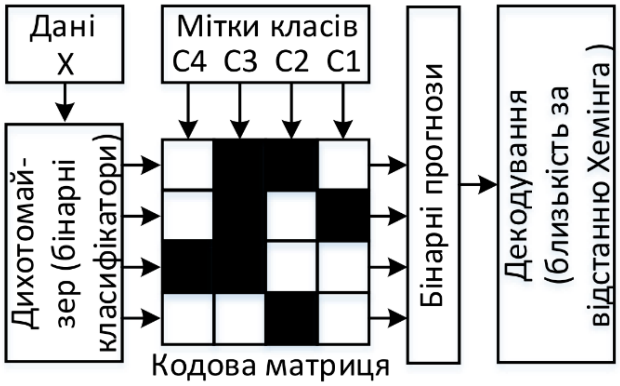


Рисунок 1.12 – Ілюстрація щодо кодування номеру класу кодами, що виправляють помилки

Найпоширенішим способом вибору кодів для кодування класів є побудова кодової матриці Адамара. Ця матриця і використовується для порівняння прогнозованого коду з кодовою матрицею і забезпечую максимальну кодову відстань, як між рядками так і між стовпцями. Чорні комірки матриці (рис. 1.12) позначають одиниці, а білі – нулі. Теоретично велика міжрядкова відстань забезпечує ефективне самовиправлення помилок, а велика міжстовбцева відстань забезпечує некорельованість кожного з виходів, що формує розряд двійкового коду. Однак ECOC в такому виконанні не враховується структура і варіативності класів розпізнавання, що знижує його ефективність в практичних задачах класифікаційного аналізу.

Таким чином, попереднє навчання глибоких екстракторів ознакового опису може бути виконано в рамках методології сіамських нейронних мереж і

даний підхід має геометричний зміст. При цьому побудову вирішувальних правил як і точну настройку варто здійснювати в рамках методів теорії інформації і кодування, оскільки це забезпечує регуляризуючий ефект.

1.3 Сучасний стан і тенденція розвитку методів класифікаційного аналізу даних

Нехай дано алфавіт класів розпізнавання $\{X_m^o \mid m=\overline{1,M}\}$, де M — кількість класів, що характеризують дефект або контекст спостереження та навчальна матриця $\| y_{m,i}^{(j)} \mid i=1,N,\ j=1,n \|$, де N – кількість ознак розпізнавання; n – кількість реалізацій образу. Відомий вектор параметрів функціонування системи розпізнавання, який у загальному випадку має таку структуру:

$$g =< g_1,..., g_{\xi_1},..., g_{\Xi_1},\ f_1,..., f_{\xi_2},..., f_{\Xi_2} >,\ \Xi_1+\Xi_2=\Xi,$$

де $< g_1,..., g_{\xi_1},..., g_{\Xi_1} >-$ генотипні параметри функціонування нечіткого регулятора, які впливають на параметри розподілу реалізацій образу;
$< f_1,..., f_{\xi_2},..., f_{\Xi_2} >-$ фенотипні параметри функціонування нечіткого регулятора, які прямо впливають на геометрію контейнера класу розпізнавання.

При цьому відомі обмеження на відповідні параметри функціонування

$$R_{\xi_1}(g_1,..., g_{\xi_1},..., g_{\Xi_1}) \le 0;\ R_{\xi_2}(f_1,..., f_{\xi_2},..., f_{\Xi_2}) \le 0.$$

Треба на етапі машинного навчання побудувати оптимальне в інформаційному розумінні розбиття класів розпізнавання.

Для досягнення поставленої мети необхідно вирішити такі завдання:

1) розробка моделі класифікаційного аналізу зображень;
2) розробка методу навчання моделі класифікаційного аналізу;
3) оцінка функціональної ефективності моделі класифікаційного аналізу;
4) програмно реалізувати запропоновані алгоритми;
5) виконати тестування розроблених алгоритмів на тестових даних;
6) зробити висновки на основі отриманих результатів.

2 ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ РОЗПІЗНАВАННЯ ДЕФЕКТІВ СТІЧНИХ ТРУБ

2.1 Модель і метод навчання моделі класифікаційного аналізу зображень відеоінспекції стічних труб

Методи теорії інформації і кодування забезпечують побудову оптимальних в інформаційному сенсі вирішувальних правил за найскладніших у статистичному сенсі випадків. Синтез вирішувальних правил в рамках геометричного підходу знижує вимоги до обсягу навчальних даних, оскіль не потребує оцінювання щільності ймовірності в кожній ділянці простору. Пропонується поєднати ці два підходи для максимізації інформаційно спроможності системи розпізнавання дефектів стічних труб за умов обмеженого обсягу розміних навчальних зразків.

Як сучасний метод геометричного підходу пропонується використати сіамьскі нейронні мережі, що навчаються на триплетах для максимізації відстані між зразками різних класів і мінімізації відстані між зразками одного класу. Для поєднання з теорією інформації і кодування пропонується пропонується формувати дискретне (двійкове) ознакове подання для реалізації кодів, що виправляють помилки (error-correcting output codes). При цьому для підвищення завадозахищеності пропонується побудова радіально-базисних меж класів в двійковому просторі Хемінга, що оптимізуються за інформаційним критерієм ефективності [15].

Запропонований метод складається з 5 основних етапів (фаз) (рис. 2.1). Перший етап включає в себе різноманітні операції з даними, такі як повороти, шкалювання, зміну яскравості і додавання шуму. Наступні 3 етапи – це безпосередньо навчання класифікатора. На останньому етапі вирішальні правила коригуються враховуючи дисперсію спостережень всередині класів в двійковому просторі Хеммінга [16].

Другий етап навчання моделі аналізу зображення проводиться із застосуванням адаптивних алгоритмів зворотного поширення помилки, з яких найбільш популярним є Adam [17]. На вхід моделі подається міні-пакет, в

якому *M* зображень кожного класу. Триплетна функція втрат розраховується за формулою (2.1).

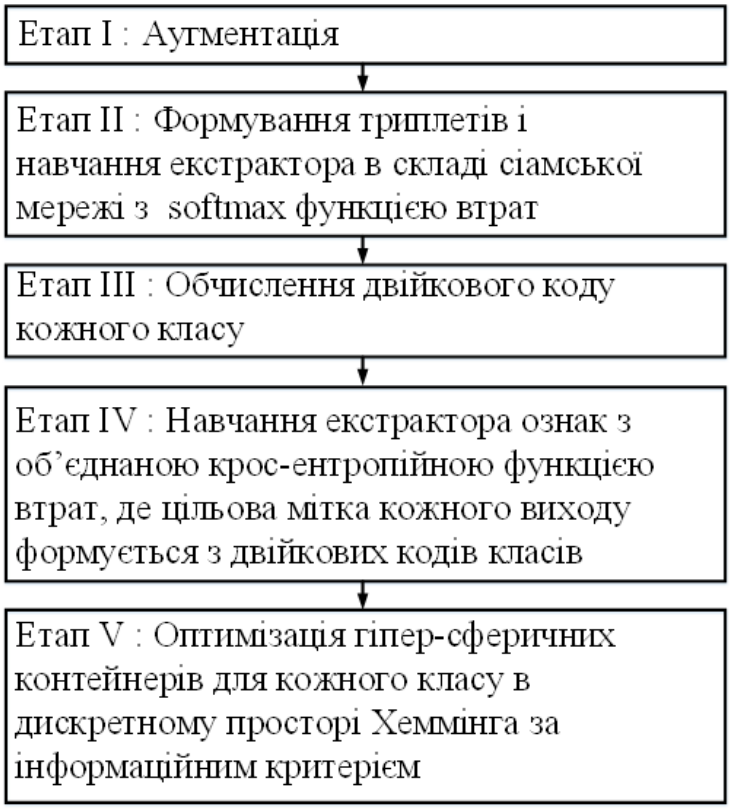


Рисунок 2.1 – Етапи запропонованого методу навчання

$$L = -\log \frac{\exp(\left\| f(x_a) - f(x_{ep}) \right\|)}{\exp(\left\| f(x_a) - f(x_{ep}) \right\|) + \exp(\left\| f(x_a) - f(x_{shn}) \right\|)} \tag{2.1}$$

де f(x) – функція екстракції ознакового опису, тобто модель залежності між між вхідним зображенням та вихідним вектором ознак; x_a – зображення з міні-пакету, що обирається випадковим чином; x_{ep} – найближчий сусідній зразок з міні-пакету, що належить до того ж самого класу, тобто

$$x_{ep} = \arg \min_{x:C(x)=C(x_a)} \left\| f(x_a) - f(x) \right\|, \tag{2.2}$$

де C(x) – функція що повертає клас зображення; x_{shn} – зразок зображення з міні-пакету, що є найближчим серед зразків протилежних класів, проте знаходиться далі ніж зразок, тобто

$$x_{shn} = \arg \min_{\substack{C(x)\neq C(x_a) \\ x: \|f(x_a)-f(x)\|>\|f(x_a)-f(x_p)\|}} \|f(x_a)-f(x)\|. \quad (2.3)$$

Для підвищення ефективності апроксимації дискретного подання даних пропонується додати регуляризуючу складову, що враховуватиме помилку округлення вихідного сигналу мережі до двійкового коду. В результаті функція втрат набуде наступного вигляду [15]

$$L = -\log \frac{\exp(\|f(x_a)-f(x_{ep})\|)}{\exp(\|f(x_a)-f(x_{ep})\|)+\exp(\|f(x_a)-f(x_{shn})\|)} + \\ +\lambda\left(f(x_a)^T(e-f(x_a))+f(x_{ep})^T(e-f(x_{ep}))+f(x_{shn})^T(e-f(x_{shn}))\right), \quad (2.4)$$

де $e=[1,1,...,1]^T$ – одинична матриця-стовбець; λ – коефіцієнт регуляризації.

Наступна фаза методу навчання призначена для визначення двійкового коду кожного класу з метою реалізації кодів, що виправляють помилки (error-correcting output codes). Для врахування структури і варіативності класів коди формуватимуться на основі статистики ознакового опису, що формується навченим екстрактором сіамської мережі. Якщо дано навчальну матрицю $\{x_{z,s} \mid z=\overline{1,Z}, s=\overline{1,n_z}\}$, де n_z – кількість зразків z-го класу, то в результаті застосування порогу на значення ознак сформується двійкова навчальна матриця $\{b_{z,s,i} \mid z=\overline{1,Z}, s=\overline{1,n_z}, i=\overline{1,N}\}$ з розмірністю N. На основі двійкової

матриці можна обчислити опорний (або еталонний) двійковий вектор (код) для кожного класу розпізнавання.

Таким чином, двійкова матриця формується шляхом подавання кожного зображення з множини $\{x_{z,s}\}$ на вхід моделі і округлення виходу сігмоїдного шару до вектора цілих чисел

$$b_{z,s,i} = \begin{cases} 1, & \text{if } f(x_{z,s,i}) > 0.5; \\ 0, & \text{otherwise.} \end{cases}, \tag{2.5}$$

Двійковий опорний (еталонний) вектор b_z для z-го класу можна визначити шляхом порозрядного порівняння частоти двійкових одиниць в z-му класі з фоновою частотою одиниць в навчальній вибірці

$$b_{z,i} = \begin{cases} 1, & \text{if } \frac{1}{n_z}\sum_{s=1}^{n_z} b_{z,s,i} > \frac{1}{Z}\sum_{c=1}^{Z}\frac{1}{n_c}\sum_{s=1}^{n_c} b_{c,s,i}; \\ 0, & \text{otherwise.} \end{cases}, \tag{2.6}$$

Обчислений еталонний вектор z-го класу b_z можна використовувати як розмітку (label) зразка для подальшого навчання з використанням Joint Binary Cross Entropy Loss, яка для кожного вхідного зразка *x* може бути обчислена за формулою

$$L = -\sum_{i=1}^{N}\left(b_i \log f_i(x) + (1 - b_i)\log(1 - f_i(x))\right), \tag{2.7}$$

де $f_i(x)$ – значення i-го виходу сігмоїдного шару для вхідного зображення x;

b_i – значення i-го розряду еталонного вектора класу, до якого належить зображення *x*.

Остання фаза машинного навчання пов’язана з оптимізацію радіусу контейнерів за інформаційним критерієм для врахування меж відхилення двійкового подання спостережень кожного класу від відповідних еталонних векторів

$$E_z^* = \max_{\{d\}} E_z(d), \qquad (2.8)$$

де $\{d\} = \{0,1,...,\left(\sum_i b_{z,i} \oplus b_{c,i} - 1 \right)\}$ – набір концентричних радіусів з центром b_z розподілу даних в z-му класі, які обчислюються за правилом (2.6);

E_z – інформаційний критерій для z-го класу, який є функцією від характеристик точності [18].

Таким чином, для побудови інформативного ознакового опису і завадозахищених вирішувальних правил варто будувати модель шляхом поєднання ідей та методів сіамських мереж та інформаційно-екстремального навчання.

2.2 Критерій оптимізації вирішувальних правил

Оцінка Важливим питанням інформаційного синтезу моделей аналізу даних є оцінка функціональної ефективності процесу навчання, яка визначає максимальну асимптотичну достовірність рішень, що приймаються на екзамені. Як критерій функціональної ефективності можуть використовуватися різні інформаційні критерії, які задовольняють таким властивостям інформаційних мір [7, 17]:

- інформаційна міра є величина дійсна і знакододатна як функція від імовірності;

- кількість інформації для детермінованих змінних ($p_i = 1$ або $p_i = 0$) дорівнює нулю;

- інформаційна міра має екстремум при значенні ймовірності $p_i = \frac{1}{m}$, де m – кількість якісних ознак розпізнавання.

Серед інформаційних мір для оцінки функціональної ефективності моделі

аналізу даних перевагу слід віддавати статистичним логарифмічним критеріям, які дозволяють працювати з навчальними вибірками відносно малих обсягів [6]. Серед таких критеріїв найбільшого використання знайшла ентропійна міра. Подамо нормований ентропійний критерій ефективності навчання моделі розпізнавати спостереження класу X_m^o у вигляді :

$$E_m^{(k)} = \frac{I_m^{(k)}}{I_{\max}^{(k)}} = \frac{H_m^{(k)} - H_m^{(k)}(\gamma)}{H_m^{(k)}}, \qquad (2.9)$$

де $I_m^{(k)}$ – кількість умовної інформації, що обробляється на k-му кроці навчанні моделі розпізнавати спостереження класу X_m^o;

$I_{\max}^{(k)}$ – максимально можлива кількість умовної інформації, одержаної на k-му кроці навчанні моделі розпізнавати спостереження одного із класів із заданого алфавіту $\{X_m^o\}$, $m = \overline{1, M}$;

$$H_m^{(k)} = -\sum_{l=1}^{M} p(\gamma_{l,k}) \log_2 p(\gamma_{l,k}) \qquad (2.10)$$

апріорна (безумовна) ентропія, що існує на k-му кроці навчання моделі розпізнавати реалізації класу X_m^o;

$$H_m^{(k)}(\gamma) = -\sum_{l=1}^{M} \sum_{m=1}^{M} p(\gamma_{l,k}) p(\mu_{m,k} / \gamma_{l,k}) \log_2 p(\mu_{m,k} / \gamma_{l,k}) - \qquad (2.11)$$

апостеріорна (умовна) ентропія, що характеризує залишкову невизначеність після k-го кроку навчання системи розпізнавати реалізації класу X_m^o;

$p(\gamma_{l,k})$ – безумовна ймовірність прийняття на k-му кроці навчання гіпотези $\gamma_{l,k}$;

$p(\mu_{m,k}/\gamma_{l,k})$ – апостеріорна ймовірність прийняття на k-му кроці навчанні рішення $\mu_{m,k}$ за умови, що прийнята гіпотеза $\gamma_{l,k}$.

Для двохальтернативної системи оцінок ($M = 2$) і рівноймовірних гіпотез, що характеризує найбільш важкий у статистичному сенсі випадок прийняття рішень, після відповідної підстановки ентропій (2.2) і (2.3) у вираз (2.1) та заміни відповідних апостеріорних ймовірностей на апріорні за формулою Байєса ентропійний критерій набуває вигляду [17]

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

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

$$\left. + \frac{D_{2,m}^{(k)}(d)}{\alpha_m^{(k)}(d) + D_{2,m}^{(k)}(d)} \log_2 \frac{D_{2,m}^{(k)}(d)}{\alpha_m^{(k)}(d) + D_{2,m}^{(k)}(d)} \right), \qquad (2.12)$$

де $\alpha_m^{(k)}(d)$ – частота помилок першого роду (false postive rate) – точністна характеристика рішення на k-му кроці навчання;

$\beta_m^{(k)}(\mathrm{d})$ – частота помилок другого роду (false negative rate);

$D_{1,m}^{(k)}(\mathrm{d})$ – чутливість (true positive rate, sensitivity) або перша достовірність;

$D_{2,m}^{(k)}(\mathrm{d})$ – специфічність (true negative rate, specificity) або друга достовірність;

d – дистанційна міра, яка визначає радіуси гіперсферичних контейнерів, побудованих в радіальному базисі простору Хеммінга.

Оскільки точністні характеристики є функціями відстані від еталонних від геометричних центрів контейнерів (кодів) відповідних класів розпізнавання, то критерій (2.4) слід розглядати як нелінійний і взаємно-неоднозначний функціонал від точністних характеристик, що потребує знаходження в процесі навчання робочої (допустимої) області для його визначення. Робочу область

визначення функції інформаційного критерію можна задати у вигляді системи нерівностей

$$\begin{cases} \beta_m^{(k)}(d)<0,5; \\ \alpha_m^{(k)}(d)<0,5; \\ d < d(x_c \oplus x_m), \end{cases}$$

де $d(x_c \oplus x_m)$ – відстань Хемінга між центрами сусідніх класів.

Таким чином, двійкові опорні (еталонні) вектори (коди) класів є алфавітом повідомлень, закодованих кодами Хемінга з метою забезпечення виправлення помилок, що виникають під впливом впливу завад під час передачі повідомлення. При цьому радіус контейнера класу відповідає максимальній кількості помилок, які можуть бути виправлені для кожного з класів без втрати інформації. Процес оптимізації радіусу радіально-базисних вирішувальних правил (контейнерів) спрямований на підвищення ефективності рішень за умов перетину класів з різною дисперсією розподілу даних, а також для виявлення аномалій (новизни) в даних. При цьому загладжуючий ефект логарифмічного інформаційного критерію забезпечує підвищення узагальнюючої здатності отриманої моделі.

2.3. Критерії валідації моделей класифікаційного аналізу даних

Оцінку ефективності навченої моделі здійснюють на зовнішній незалежній розміченій вибірці, яку називають тестовою. Прогнозовані порівнюють з правильними (ground truth) мітками, що наносилися експертом, для обчислення метрик продуктивності моделі.

Метрики продуктивності моделі обчислюються на основі підрахунку результатів статистичних тестів. Для одного тестового зразка можливі 4-ри можливі результати статистичного тесту [7, 20]:

1) правильно позитивний результат (True Positive, або скорочено TP), тобто в тестовому зразку дійсно присутній об’єкт і його розпізнано;

2) неправильно негативний результат (False Negative, або скорочено FN), тобто в тестовому зразку дійсно присутній об’єкт, але його не було розпізнано;

3) правильно негативний результат (True Negative, або скорочено TN), тобто в тестовому зразку дійсно не присутній об’єкт, але його і не було розпізнано;

4) неправильно позитивний результат (False Positive, або скорочено FN), тобто в тестовому зразку дійсно не присутній об’єкт, але його.

Важливими характеристиками моделі аналізу даних вважаються чутливість, специфічність, точність, частота помилок першого та другого роду, правильність, збалансована правильність та F1-міра. На основі цих характеристик приймають рішення про придатність моделі до практичного використання.

Чутливість (Sensitivity, або True Positive Rate або Recall) – кількість правильно позитивних результатів, що поділена на кількість позитивних зразків, тобто

$$\text{Sensitivity} = \text{Recall} = TP / (TP+FN).$$

Специфічність (або True Negative Rate) – кількість правильно негативних результатів, поділена на реальну кількість негативних зразків, тобто

$$\text{Specificity} = TNR = TN / (FP+TN).$$

Точність (або Precision) – кількість правильно позитивних результатів, поділена на загальну кількість позитивних прогнозів, тобто

$$\text{Precision} = TP/(TP+FP).$$

Частота помилок першого роду (або False Positive Rate) – кількість неправильно позитивних результатів, поділена на загальну кількість правильно негативних результатів, тобто

$$FPR = FP/(FP+TN).$$

Частота помилок другого роду (або False Negative Rate) – кількість хибно негативних, поділена на загальну кількість істинно позитивних, тобто

$$FNR = FN/(FN+TP).$$

Правильність (або Accuracy) – частка правильних прогнозів від загальної кількості прогнозів, тобто

$$Accuracy = (TP+TN) / (TP+TN+FP+FN)$$

Однак правильність може бути адекватною метрикою тільки якщо кількість зразків кожного класу однакова, тобто класи є збалансованими.

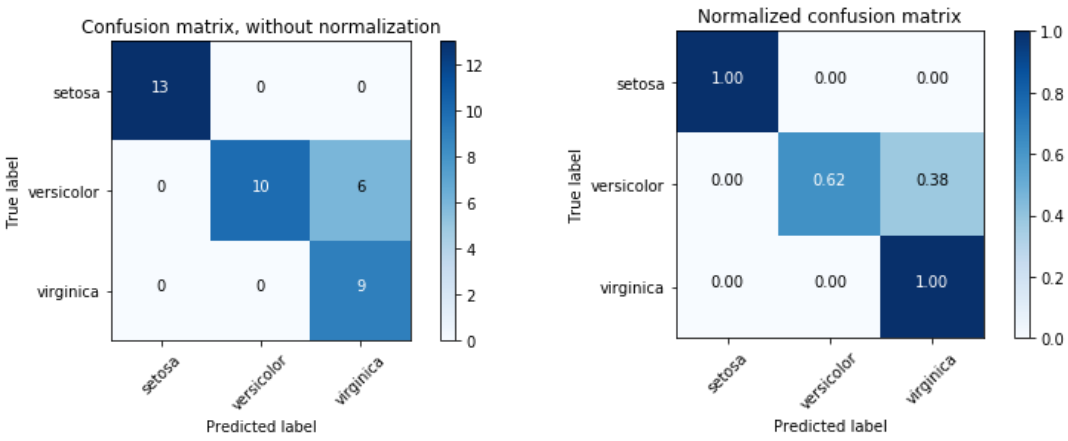
Збалансована правильність (або Balanced Accuracy) – середнє арифметичне між чутливістю та специфічністю, тобто

$$Balanced\ Accuracy = (Sensitivity + Specificity)/2$$

Міра F1 (F) – гармонічне середнє між точністю та чутливістю, значення якої лежить в діапазоні [0, 1], тобто

$$F = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}}$$

Для візуалізації продуктивності класифікаційних алгоритмів штучного інтелекту також набуло поширення використання матриці помилок (Confusion matrix). Ця матриця має конкретний формат. Кожен рядок матриці представляє екземпляри фактичних класів, а стовбець представляє екземпляри прогнозованих класів. Приклад матриці помилок для 3-х класів розпізнавання показано на рис. 2.2.



а *б*

Рисунок 2.2 – Матриця помилок : *а* – оригінальна матриця; *б* – нормалізована матриця

Діагональні елементи нормалізованої матриці містять відповідні значення чутливості (recall або sensitivity). Позадіагональні елементи відповідають частці помилкових класифікацій. На основі записів матриці можна обчислити чутливість, специфічність та точність.

Таким чином, існує багато метрик оцінки продуктивності моделей класифікаційного аналізу. Серед цих метрик продуктивності найбільш інформативними є точність та акуратність прогнозів на тестовій вибірці.

3 РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ РОЗПІЗНАВАННЯ ДЕФЕКТІВ СТІЧНИХ ТРУБ

3.1 Формування вхідного математичного опису

Звіт про інспекцію стічних труб формується у вигляді одного із стандартів, серед яких найбільш популярними є MSCC5, PACP6 та PACP7 [15]. Кожен запис звіту складається з коду та мета-інформації, що враховує локалізацію та тяжкість. Оскільки камера в трубі може змінювати орієнтацію, а сама труба може бути досить зруйнованою, то для кодування дефектів необхідно мати інформацію про контекст кадру, тобто інформацію про орієнтацію камери, умови спостереження і тяжкі руйнування чи вихід за контекст задачі (наприклад, камера увімкнута на вулиці).

Алфавіт класів для розпізнавання контексті спостереження містить 9 основних контекстів (рис. 3.1) [16]. Клас X_1^o відповідає орієнтації камери вздовж труби. Класи X_2^o, X_3^o, X_4^o та X_5^o відповідають орієнтаціям камери вліво, вправо, вгору та вниз відповідно, де внаслідок не повного повороту зрозуміло яка саме це сторона труби (гора, низ, ліво чи право). Клас X_6^o характеризує орієнтацію камери суто до стінки зліва, справа, знизу чи згори, де внаслідок повного повороту не можна сказати з впевненістю яка саме це сторона труби важко зрозуміти яка саме це частина труби (гора, низ, ліво чи право). Клас X_7^o характеризує точку входу в трубу чи точку виходу з трубу. Клас X_8^o характеризує ситуації, що виходять за контекст спостереження чи характеризують втрату видимості. Клас X_9^o характеризує значне руйнування труби, що перешкоджає руху і призводить до передчасної зупинки інспекції.

Крім контексту зображення необхідно кожен кадр аналізувати на предмет наявності дефектів. В дослідженні з усього різноманіття труб розглядаються лише бетонні (Concrete Pipe) та скло-керамічні (Vitrified Clay) труби, для яких характерна широка номенклатура можливих дефектів. Алфавіт класів містить 9-ть найбільш поширених дефектів стічних труб (рис. 3.2) [17].

а *б* *в*

г *ґ* *д*

е *є* *ж*

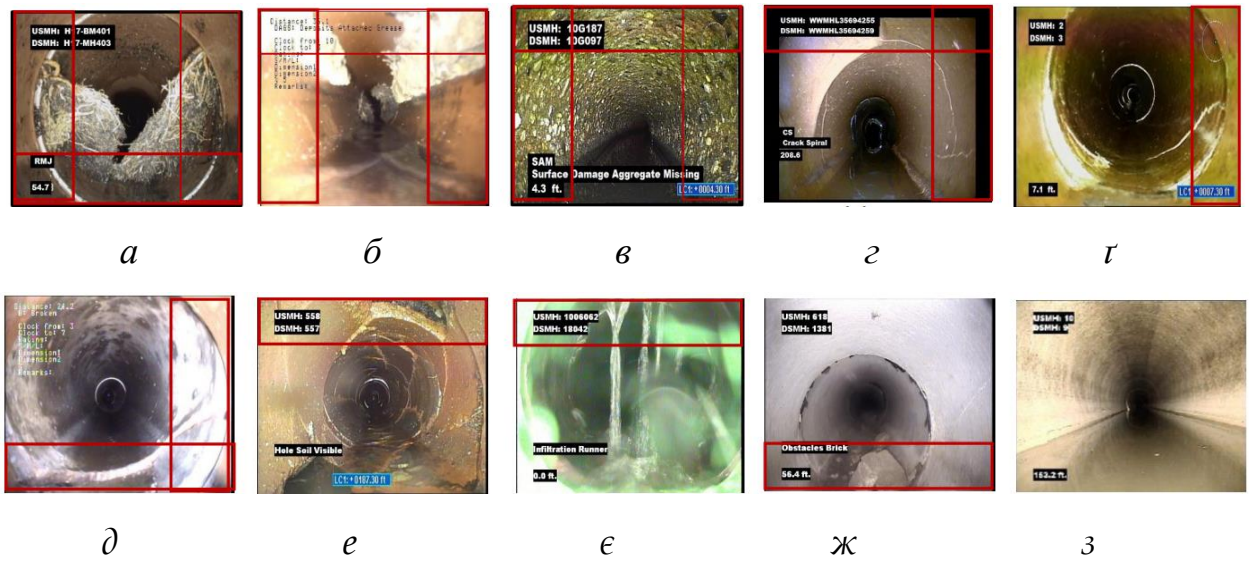
Рисунок 3.1 – Приклади зображень кожного з класів :

а – клас X_1^o; *б* – клас X_2^o; *в* – клас X_3^o; *г* – клас X_4^o; *ґ* – клас X_5^o;

д – клас X_6^o ; *е* – клас X_7^o; *є* – клас X_8^o; *ж* – клас X_9^o

Клас X_1^o відповідає дефекту “roots”, що відповідає проникненню коренів дерев в трубу. Клас X_2^o відповідає дефекту “deposits”, що призводить до зменшення робочого перерізу стічної труби. Класи X_3^o, X_4^o, X_5^o, X_6^o та X_7^o характеризують різноманітні структурні дефекти і руйнування труби такі як “surface damage”, “crack”, “fracture”, “broken” та “hole” відповідно. Клас X_8^o відповідає дефекту “infiltration”, який виникає внаслідок входження води через дефект або пошкоджений стик, або через пористу ділянку на стінці труби. Клас

X_9^o відповідає дефекту “Obstruction”, що суттєво утруднює потік або зменшує гідравлічну потужність. X_{10}^o відповідає нормальному стану труби і небхідний для регуляризації.



а *б* *в* *г* *ґ*

д *е* *є* *ж* *з*

Рисунок 3.2 – Приклади зображень кожного з класів дефектів труби :

а – клас X_1^o; *б* – клас X_2^o; *в* – клас X_3^o; *г* – клас X_4^o; *ґ* – клас X_5^o;

д – клас X_6^o ; *е* – клас X_7^o; *є* – клас X_8^o; *ж* – клас X_9^o; *з* – клас X_{10}^o

Аналіз дефектів труб відбувається не на всій фотографії, а на бокових частинах кадру, оскільки саме для них відомі значення з одометру, що спрощує їх локалізацію. Тому з кадру вирізаються прямокутні зображення, що відповідають 15% від зображення зверху, знизу та по бокам. Потім ці зображення рісайзяться до роздільної здатності 192x192 пікселів (рис. 3.3).

Рисунок 3.3 – Конвеєр обробки розмічених патчів зображення дефектів на крайових ділянках кадрів, для яких відоме значення одометра

Для кожного класу зібрано по 200 навчальних та тестових зразків. Дані для перевірки концепції надала компанія Ace Pipe Cleaning, Inc (Канзас Сіті, США). Звичайно кожен з цих класів має численну кількість підкласів, однак тут не буде розглядатися подальший аналіз класифікованого зображення. Кожне зображення використовується для розширення навчального набору в 4 рази шляхом застосування невеликих випадкових викривлень, яка включає варіацію яскравості, масштабу та поворотів до 10% від повного діапазону, а також накладання шумів типу сіль та перець.

3.2 Короткий опис програмної реалізації

На вхід системи розпізнавання дефектів стічних труб для автоматичного формування звітів необхідно подавати окрім відео файлу інспекції ще й дані від одометра для визначення відстані кожного з дефектів. Інформаційна система, що пропонується, здійснює аналіз інформації в декілька етапів, послідовність яких показана на рис. 3.4. На першому етапі відбувається класифікаційний аналіз контексту спостереження, які характеризують орієнатцію камери та ситуації, які стосуються початку чи кінця інспекції, що важливо для локалізації дефектів. Наступний етап полягає в аналізі кадрів, де камера орієнтована вперед вздовж труби, оскільки саме така орієнтація дозволяє визначити локацію дефекта, тобто дає змогу визначити чи дефект вгорі, чи внизу, справа чи зліва. Однак враховуючи те, що на кожному кадрі нас цікавлять лише ті дефекти, відстань до яких від початку інспекції ми можемо оцінити, то доцільно виконувати аналіз не всього зображення а лише патчів, що знаходяться на межі кадру, оскільки для них відоме значення одометра. Тому на класифікаційний аналіз дефектів надходять лише патчі, на межі кадру, а не повне зображення.

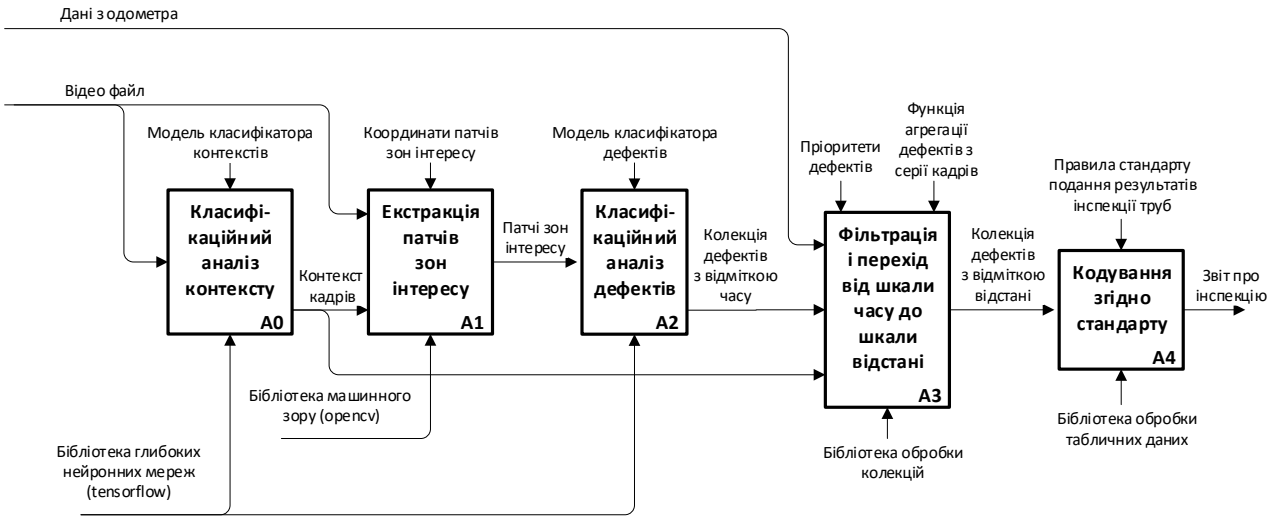


Рисунок 3.4 – Функціональна схема інформаційної технології класифікаційного аналізу стічних труб

Аналіз рис. 3.4 показує, що отримані покадрові прогнози класифікаційної моделі підлягають фільтрації і переходу від шкали часу до шкали відстані. Агрегація прогнозів для кадрів, що мають різну часову відмітку, але однакову відстань на ідикаторі одометра, дозволяє шляхом голосування відсікати хибні спрацювання і заповнювати пропуски. При цьому варто враховувати, що деякі коди дефектів мають пріоритет, і їх наявність автоматично нівелює спрацювання деяких інших дефектів. Саме тогу тяжкість дефектів, тобто їх пріоритети необхідно задавати розробников інформаційного забезпечення опирвючись на усереднений досвід існуючих стандартів кодування дефектів у звітах про інспекцію. Останній етап аналізу даних полягає у формуванні звіту шляхом відображення класів розпізнавання у коди дефектів конткретного стандарту кодування. Реалізація даної інформаційної системи потребує використання бібліотек глибокого машинного навчання, машинного зору, а також бібліотек обрабки колекцій і табличних даних.

Для експериментів і перевірки концепції як основу для екстрактора ознак пропонується використати згорткову мережу загального призначення MobileNetV2 без повнозвязних вихідних шарів з коефіцієнтом ємності, що дорівнює 0,35 [17]. Структуру запропонованої моделі класифікаційного аналізу даних показано на рис. 3.5*а*. Для визначення ефективності запропонованого

підходу варто дослідити ефективність даного екстрактор ознак з класичними вихідними шарами. На рис. 3.5*б* показано базову (baseline) модель для порівння ефективності з запропонованим підходом.

MobileNet екстрактор ознак (backbone)
Шар Global Average Pooling 2D
Шар Dropout (rate=0.5)
Повнозв’язний шар, Dense (128 вузлів)
Сігмоїдний шар, Sigmoid
Округлючий шар
Радіально-базисний шар

а

MobileNet екстрактор ознак (backbone)
Шар Global Average Pooling 2D
Шар Dropout (rate=0.5)
Повнозв’язний шар, Dense (128 вузлів)
Сігмоїдний шар, Sigmoid
Повнозв’язний шар
Нормуючий Softmax шар

б

Рисунок 3.5 – Структура моделі класифікатора : *а* – запропонована модель; *б* – модель з класичними вихідними шарами для порівняння ефективності

Розробка програмного коду здійснювалась в середовищі Jupyter Notebook та Google colab на мові програмування python 3. Розроблене програмне забезпечення потребує використання сторонніх бібліотек, опис яких наведено в табл. 3.1.

Система застосовує бібліотеку opencv-3.5 для читання вхідних даних в форматі NumPy. Потім numpy масив ініціалізує тензорні масиви, що є зручними для використання у фреймворці глибокого машинного навчання tensorflow. Для підвищення читабельності та швидкості написання коду використовується бібліотека keras, що є високорівневою обготкою для tensorflow. Попередня обробка даних выдбувається функціями бібліотеки Scikit-Learn [18].

Функція створення глибокої моделі екстракції ознакового опису теж присутня у Додатку А і називається create_embedding_model. Ця функція приймає на вхід кількість ознак, які на виході має сформувати модель, та роздільну здатність вхідного зображення. Також для порівняльного аналізу будується базова модель з традиційним повнозвязним класифікаційним softmax вихідним шаром за допомогою функції create_baseline_model [19, 20].

Таблиця 3.1 – Використані бібліотеки

Назва бібліотеки	Опис
NumPy	Бібліотека-розширення мови Python, що додає підтримку великих багатовимірних масивів і матриць, разом з великою бібліотекою високорівневих математичних функцій для операцій з цими масивами.
Scikit-Learn	Безкоштовна бібліотека (ліцензія BSD), що надає ефективні інструменти для аналізу даних, основана на NumPy, SciPy і Matplotlib. Бібліотека доступна на всіх популярних репозиторіях та може бути використана для задач різноманітних контекстах. Включає ряд пакетів: – класифікація (classification); – регресія (regression); – кластер-аналіз (clustering); – редукція розмірності (dimensionality reduction); – валідація моделей (model selection); – передобробка (preprocessing); – завантаження набору даних (data loader); – візуалізація проміжних і кінцевих результатів (flow chart); – приклади використання; – датасети (datasets); – алгоритми оптимізації (optimization). Бібліотека є добре документованою, має вечерпні туторіали, інструкції швидкого старту.

Продовження таблиці 3.1

Назва бібліотеки	Опис
Keras	Keras — відкрита нейромережева бібліотека, написана мовою Python. Вона здатна працювати поверх Deeplearning4j, TensorFlow та Theano. Спроектовану для уможливлення швидких експериментів з мережами глибинного навчання, її зосереджено на тому, щоби вона була мінімальною, модульною та розширюваною. Ця бібліотека містить численні реалізації поширених будівельних блоків нейронних мереж, таких як різного типу шар, функції втрат, оптимізатори, та безліч утиліт для спрощення роботи із зображеннями та текстом.
tensorflow	Відкрита програмна бібліотека для машинного навчання, розроблена компанією Google для розв’язку задач синтезу і тренування нейронної мережі з метою автоматичного пошуку і розпізнавання образів. Tensorflow підтримує мови програмування C++, python, java script, java. Tensorflow підтримує такі обчислювальні одиниці як CPU, GPU та TPU.
opencv	Бібліотека алгоритмів комп’ютерного зору, обробки зображень і чисельних алгоритмів загального призначення з відкритим кодом. Бібліотека реалізована на C/C++, також розробляється для Python, Java, Ruby, Matlab, Lua та інших мов. Може вільно використовуватися в академічних і комерційних цілях — поширюється на умовах ліцензії BSD.
albumentations	Бібліотека для швидкої аугментації зображень, яка характеризується простотою використання і є обгорткою інших бібліотек. Придадтна для аугментації зразків у задачах класифікації, сегментації та детектування. Бібліотека легко кастомізується і додається до інших фреймворків.

Програмний код наведено в додатку А. Призначення основних класів приведено в табл. 3.2.

Таблиця 3.2 – Основні класи програмного забезпечення

Клас	Короткий опис
SoftTripletLoss	Клас, екземпляр якого забезпечує пошук триплетів і обчислення згладженої триплетної функції втрат для попереднього навчання екстрактора ознакового опису
InformationExtremeLayer	Клас, екземпляр якого забезпечує обчислення двійкових еталонних векторів і оптимізацію радіусів для побудови інформаційно-екстремальних радіально-базисних класифікаційних вирішувальних правил

Таким чином, програмна реалізація дозволяє протестувати розроблені алгоритми з метою реалізації в майбутньої повнофункціональної системи розпізнавання дефектів стічних труб.

3.3 Результати машинного навчання

Спочатку навчимо класичну модель (рис. 3.5*б*) для класифікації контекстів спостереження. Для цього буде використано та categorical cross-entropy loss та оптимізатора Adam [16]. Розмір міні-батчу становить 32 зображення, а learning rate встановлено рівним 0,0001. На рис. 3.6 показано криві зміни точності та функції втрат на тестовій та навчальній вибірках залежно від кількості епох навчання класифікатора контекстів з класичним вихідним шаром softmax.

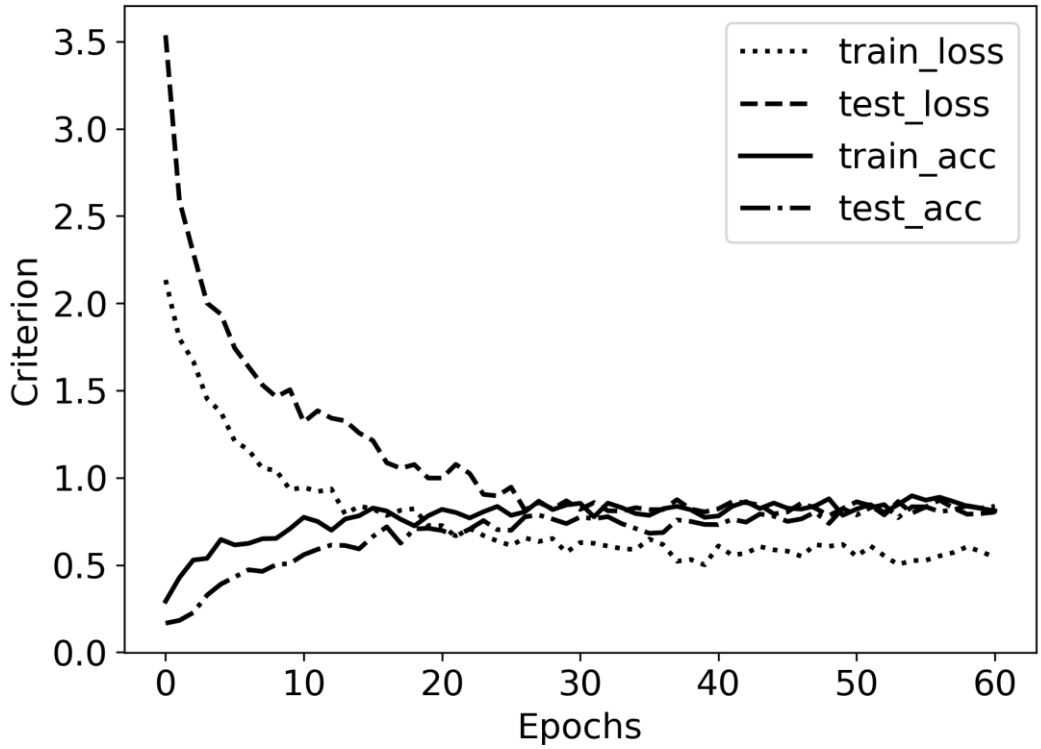


Рисунок 3.6 – Графік залежності точності та функції втрат під час навчання класифікатора контекстів від кількості епох навчання в рамках традиційного підходу

Аналіз рис. 3.6 показує, що зменшення помилки за тестовою (test_loss) вибіркою припинилося на 39 епосі. При цьому точність отриманої моделі (test_acc) становить 91%.

На рис. 3.7 показано результат навчання екстрактора ознак з сігмоїдним вихідним шаром для моделі розпізнавання контексту, де перші 30 епох навчання відбувається з використанням функції помилки (2.1) та наступні 30 епох з використанням функції помилки (2.7). При цьому кожні 5 епох будуються інформаційно-екстремальні вирішувальні правила та обчислюється усереднене за алфавітом класів значення інформаційного критерію ефективності (2.12).

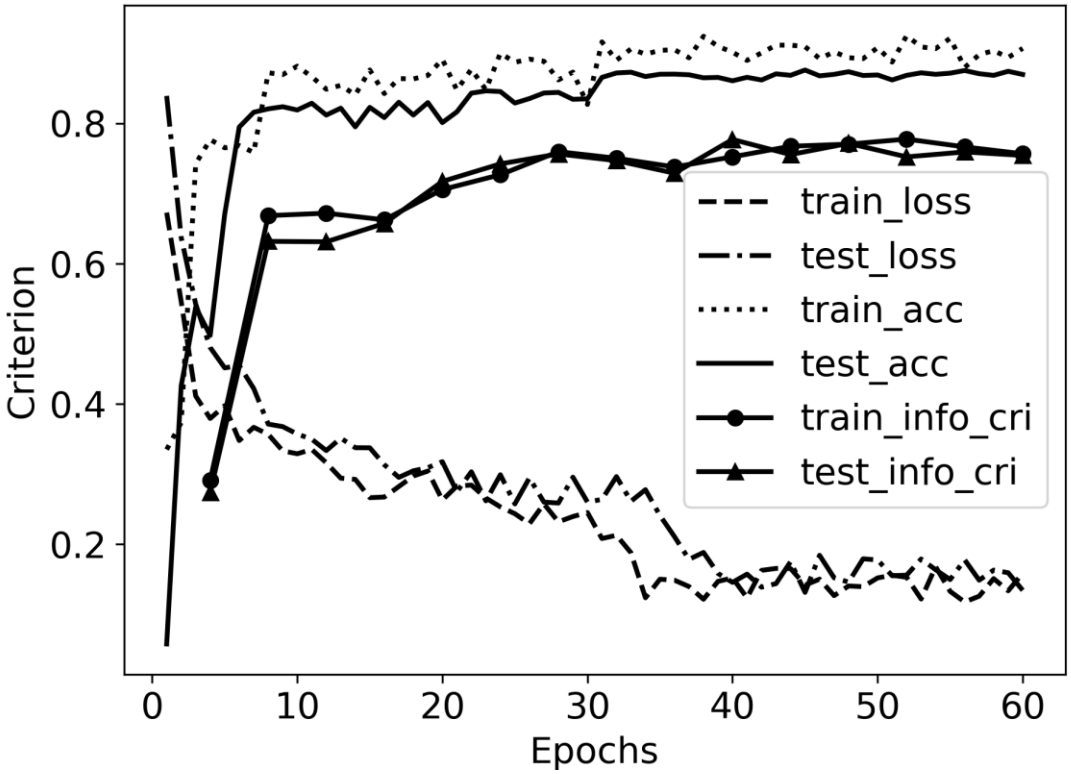


Рисунок 3.7 – Графік залежності точності, функції втрат та інформаційного критерію від кількості епох навчання в рамках запропонованого підходу

Аналіз рис. 3.7 показує, що починаючи з 7-ї епохи зменшення помилки за тестовою вибіркою дуже вповільнюється, а починаючи з 31-ї епохи відбувається різке підвищення точності вирішальних правил більше ніж на 2%. Проте після 41-ї епохи навчання покращення результату практично припиняється. Для навченої моделі точність за тестовою вибіркою становить 95,1%, а значення інформаційного критерію за навчальною і тестовою вибірками мало відрізняються і рівне 0,71. При цьому середня відстань між центрами контейнерів, яка рівна 11, перевищує значення середнього радіусу контейнерів, який рівний 6. Це свідчить про високу завадозахищеність інформаційно-екстремальних вирішальних правил.

Таким чином, використання багатофазної схеми навчання з інформаційно-екстремальними вирішувальними правилами дозволяє підвищити

точність моделі розпізнавання контекстів спостереження під час інспекції труб на 4,3% порівняно з традиційною схемою навчання з softmax вихідним шаром.

Розглянемо результати машинного навчання класифікатора дефектів на патчах кадрів у контексті орієнтації камер прямо (forward) з використанням традиційної моделі (рис.3.5а) і методу навчання. На рис. 3.8 показано зміни точності моделі на тестовій (test_acc) і навчальній (train_acc) вибірках. Кожен міні-батч містить 32 зображення, а коефіцієнт швидкості навчання дорівнює 0,0001.

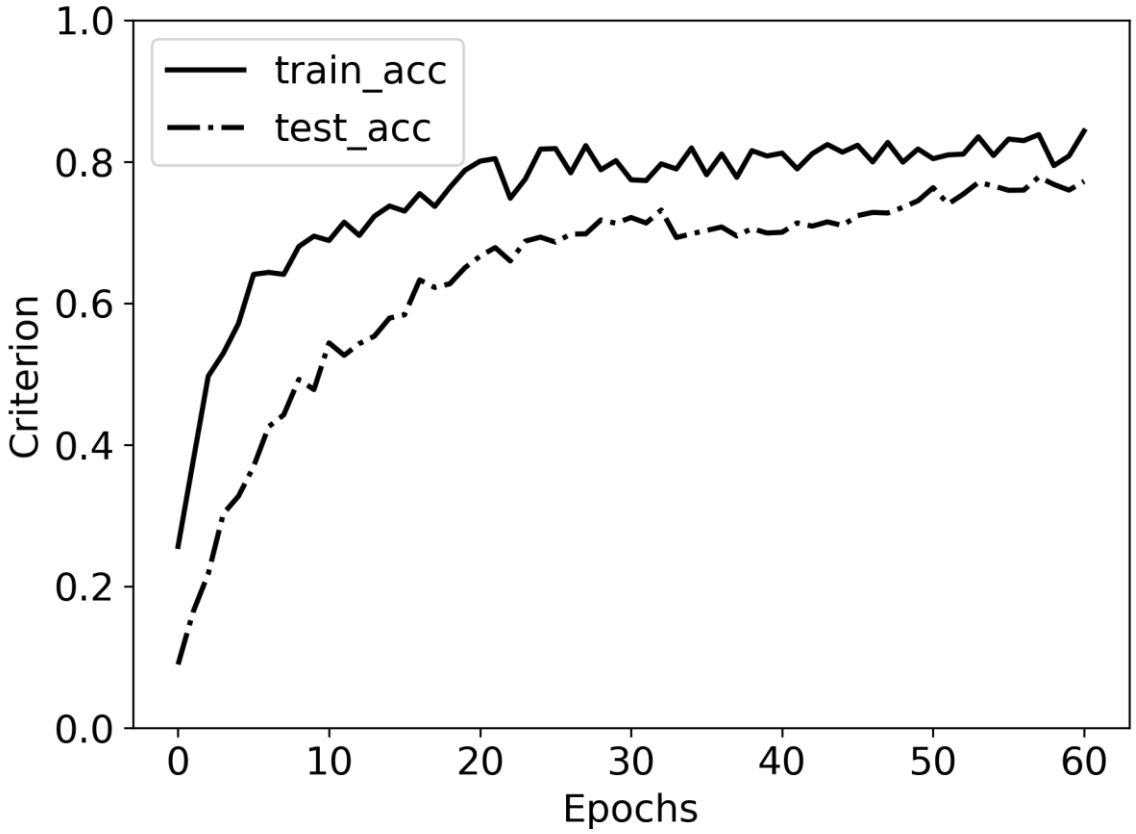


Рисунок 3.8 – Графік залежності точності та функції втрат під час навчання класифікатора дефектів від кількості епох навчання в рамках традиційного підходу

Аналіз рис. 3.8 показує, що підвищення точності зупиняється після 40 епох навчання з максимальним значенням точності результуючої моделі, що дорівнює 73% на стестовій вибірці. При цьому відстань між кривою на тестовій і на навчальні вибірці свідчить про наявність невеликого ефекту перенавчання.

На рис. 3.9 показано результати навчання запропонованого екстрактора ознак з сігмоїдним шаром і функцією втрат (2.1) для моделі детектора дефектів. При цьому кожні 5 епох відбувається побудова інформаційно-екстремальних вирішувальних правил і обчислення усередненого за алфавітом класів інформаційного критерію (2.12) на тестовій (test_info_cri) та навчальній (train_info_cri) вибірках. Після 30 епох навчання модель було збережено для наступного експерименту.

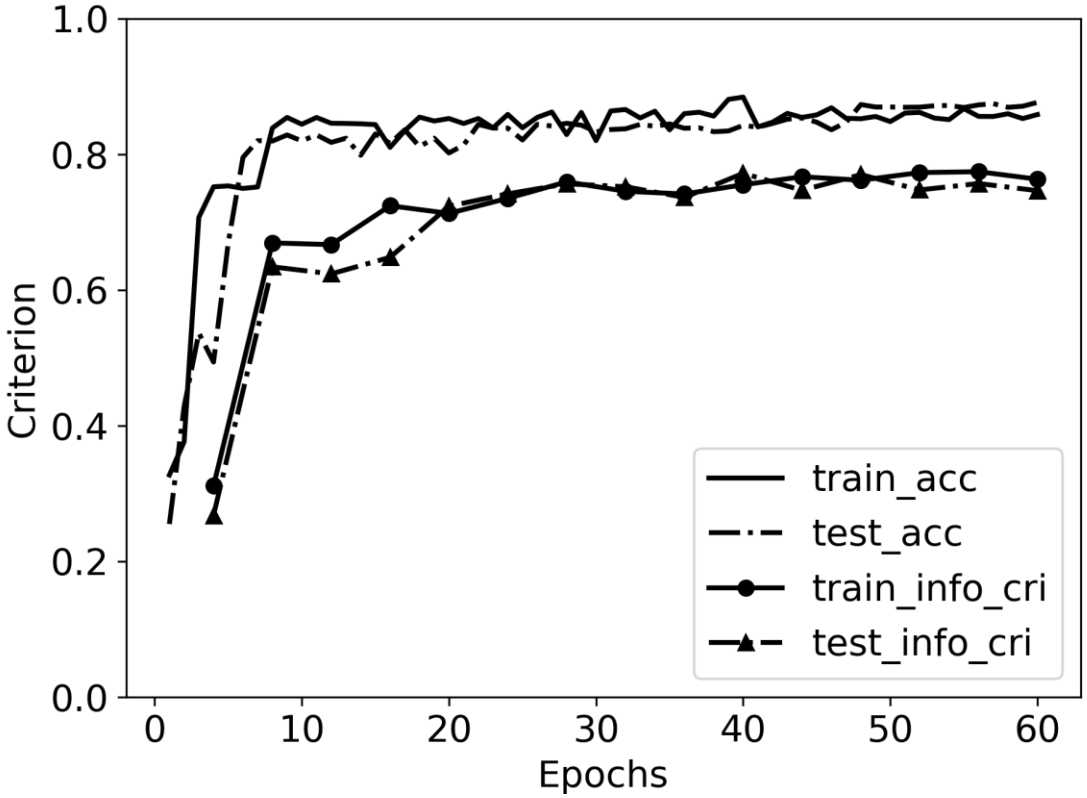


Рисунок 3.9 – Графік залежності точності та інформаційного критерію від кількості епох навчання в рамках запропонованого підходу з функцією втрат (2.1)

Аналіз рис. 3.9 показує, що з 10 епохи зростання точності значно уповільнилося, а після 30 епох було досягнуто максимальну точність, що дорівнює 85%, що на 14.1% більше ніж в базової (baseline) моделі. Однак після 40 епох навчання, подальше покращення практично припинилося. При цьому тестові і навчальні криві майже співпадають, що свідчить про високу узагальнюючу здатність отриманих вирішувальних правил. Усереднe за

алфавітом класів значення інформаційного критерію дорівнює 0,74 а середня відстань між центрами гіперсферичних контейнерів дорівнює 18 кодових одиниць, а середній радіус контейнерів класів рівний 9,1 кодових одиниць. Це свідчить про високу завадозахищеність вирішувальних правил [15].

На рис. 3.10 показано подальші результати навчання моделі, збереженої на попередньому етапі після 30 епох навчання. Наступне навчання здійснюється також 30 епох, але з використанням функції втрат (2.7) з метою зменшення внутрішньокласової дисперсії в просторі Хемінга [16].

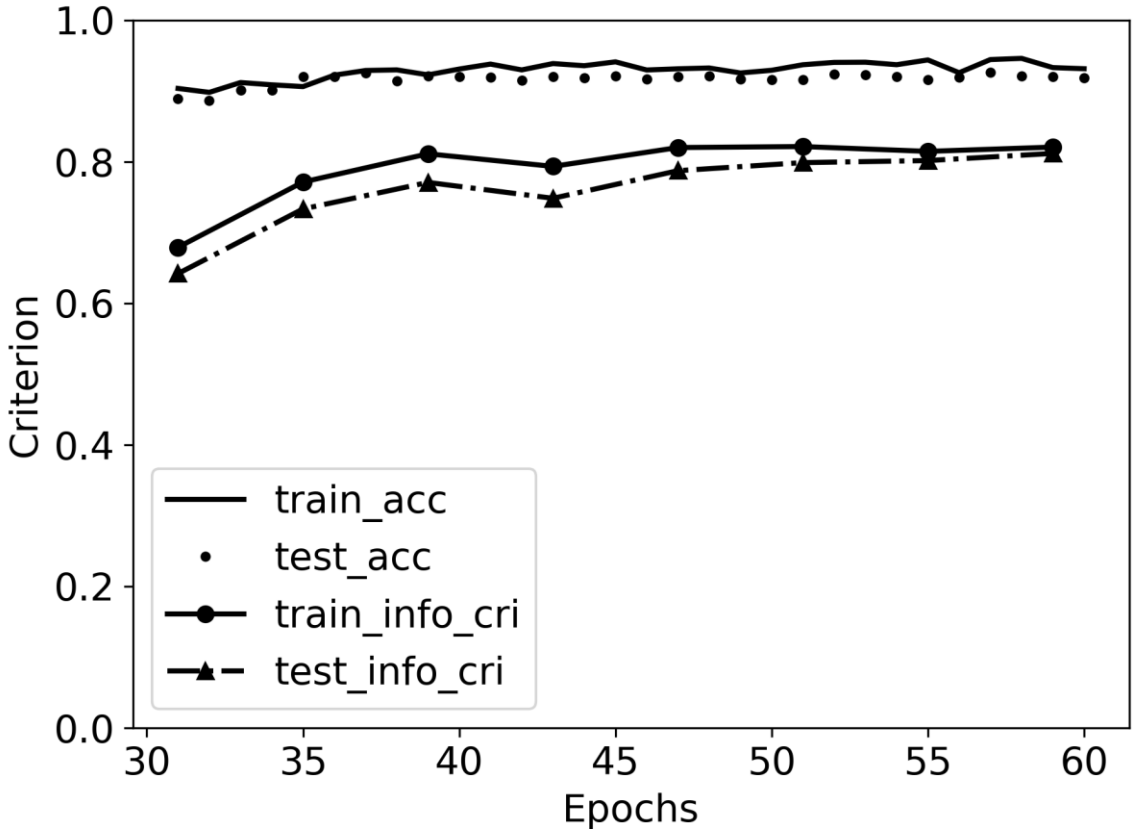


Рисунок 3.10 – Графік залежності точності та інформаційного критерію від кількості епох навчання в рамках запропонованого підходу з функцією втрат (2.7)

Аналіз рис. 3.10 показує, що використання функції joint binary cross-entropy loss (2.7) забезпечує зростання точності на 7,3% порівняно з попереднім етапом. При цьому усереднене за алфавітом класів значення інформаційного критерію (2.12) дорівнює 0,8.

Таким чином мультифазне навчання з інформаційно-екстремальними вирішувальними правилами забезпечує підвищення точності розпізнавання дефектів порівняно з традиційним підходом на 20% за туж кількість епох навчання за умов обмеженого обсягу розмічених навчальних даних [17].

ВИСНОВКИ

В магістерській кваліфікаційній роботі було проведено аналіз проблеми синтезу інтелектуальної системи розпізнавання дефектів стічних труб на основі відеоспостереження. Виконано огляд відомих рішень в галузі інтелектуального аналізу даних і обрано напрямок перспективних досліджень.

Запропоновано модель та метод навчання системи розпізнавання дефектів стічних труб з використанням багатошарових згорткових сіамських нейронних мереж та інформаційно-екстремальних класифікаційних вирішувальних правил. Поєднання цих методів і алгоритмів повинно забезпечити регуляризуючий і мета-регуляризуючий ефект за найскладніших у статистичному відношенні умов.

Результати фізичного моделювання підтверджують працездатність розроблених алгоритмів, проте безпомилкових за навчальною і тестовою вибірками моделей не було отримано, що може свідчити про неоптимальність сформованого словника ознак та інших параметрів формування вхідного математичного опису.

СПИСОК ЛІТЕРАТУРИ

1. Moradi, Saeed et al. "Review On Computer Aided Sewer Pipeline Defect Detection And Condition Assessment". Infrastructures, vol 4, no. 1, 2019, p. 10. MDPI AG, doi:10.3390/infrastructures4010010.

2. Syahrian, Nur Mutiara et al. "Vision-Based Pipe Monitoring Robot For Crack Detection Using Canny Edge Detection Method As An Image Processing Technique". Kinetik, vol 2, no. 4, 2017, p. 243. Universitas Muhammadiyah Malang, doi:10.22219/kinetik.v2i4.243.

3. Myrans, Joshua et al. "Automated Detection Of Fault Types In CCTV Sewer Surveys". Journal Of Hydroinformatics, vol 21, no. 1, 2018, pp. 153-163. IWA Publishing, doi:10.2166/hydro.2018.073.

4. Cheng, Jack C.P., and Mingzhu Wang. "Automated Detection Of Sewer Pipe Defects In Closed-Circuit Television Images Using Deep Learning Techniques". Automation In Construction, vol 95, 2018, pp. 155-171. Elsevier BV, doi:10.1016/j.autcon.2018.08.006.

5. Panella, F. et al. "Deep learning and image processing for automated crack detection and defect measurement in underground structures". ISPRS - International Archives Of The Photogrammetry, Remote Sensing And Spatial Information Sciences, XLII-2, 2018, pp. 829-835. Copernicus Gmbh, doi:10.5194/isprs-archives-xlii-2-829-2018.

6. Zhan H. Deepshoe: An improved multi-task viewinvariant cnn for street-to-shop shoe retrieval / H. Zhan, B. Shi, L. Y. Duan, A. Kot // Computer Vision And Image Understanding. – 2019. – Vol. 180. – P. 23–33. – DOI: https://doi.org/10.1016/j.cviu.2019.01.001.

7. Москаленко В. В. “Моделі і методи інтелектуального аналізу багатовимірних даних да умов апріорної невизначеності”: монографія / В. В. Москаленко. – Суми: Видавництво СумДУ, 2019. – 184 с.

8. Інтелектуальна автономна бортова система безпілотного літального апарату для ідентифікації об'єктів на місцевості [Текст]: Розробка інформаційного та програмного забезпечення бортової системи безпілотного

апарату, що функціонує в режимі самонавчання екстрактора ознакового опису середовища для навігації та класифікаційного аналізу спостережень: звіт про НДР (проміжний) / кер. В. В. Москаленко. – Суми: СумДУ, 2018. – 91 с. – Режим доступу: https://essuir.sumdu.edu.ua/bitstream-download/123456789/73907/1/ Moskalenko_V._intelektualna_1477.pdf.

9. Khan A. A Survey, of the Recent Architectures of Deep Convolutional Neural

10. Networks / A. Khan, A. Sohail, U. Zahoora, A. S. Qureshi // 2019. – P. 1–62. – Available at: https://arxiv.org/ftp/arxiv/papers/1901/1901.06032.pdf.

11. Ramachandram D. Deep Multimodal Learning / D. Ramachandram, G.W. Taylor // IEEE Signal Processing Magazine. – 2017. – P. 96 – 108.

12. Моделі і методи інформаційної технології автономного відеомоніторингу місцевості безпілотним літальним апаратом [Текст] : автореф. дис. ... канд. техн. наук : 05.13.06 / Коробов Артем Геннадійович; Нац. аерокосм. ун-т ім. М. Є. Жуковського "Харків. авіац. ін-т". – Харків, 2019. – 20 с.

13. Zhao Z. Object Detection With Deep Learning: A Review / Z. Zhao, P. Zheng, S. Xu, X. Wu // IEEE Transactions on Neural Networks and Learning Systems. – 2019. – P. 1 – 21. – DOI: 10.1109/TNNLS.2018.2876865.

14. Hu W. Learning discrete representations via information maximizing self augmented training. / W. Hu, T.Miyato, S. Tokui, E. Matsumoto // Proceedings of International Conference on Machine Learning. – Vol. 70. – P. 1558–1567.

15. Moskalenko A. Deep feature extractor with information-extreme decision rules for visual classification of sewer pipe defects and its training method / A. Moskalenko, V. Moskalenko, M. Zaretskyi, V. Lysyuk // Data Stream Mining & Processing 2020. – Lviv, Ukraine, 21-25 Aug. 2020. – pp.191-195. (Scopus & Wos).

16. Москаленко В.В. Модель і метод навчання класифікатора контекстів спостереження на зображеннях відеоінспекції стічних труб / В.В. Москаленко, М. О. Зарецький, Я. Ю. Ковальський, С. С. Мартиненко // Радіоелектронні і

комп'ютерні системи. – No 3 (95). – 2020. – P. 59-66. – DOI : 10.32620/reks.2020.3.06

17. Moskalenko, V. Sewer Pipe Defects Classification Based on Deep Convolutional Network with Information-extreme Error-correction Decision Rules / A. S. Moskalenko, V.V. Moskalenko, M. O. Zaretskyi, V. Lysyuk //Communications in Computer and Information Science (CCIS-2020) – Springer : Cham, Switzerland. – 2020. – 20 p. – DOI : 10.1007/978-3-030-61656-4_16

18. Moskalenko V. Multi-Layer Model and Training Method for Information-Extreme Malware Traffic Detector / V. Moskalenko, A. Moskalenko, A. Shaiekhov, M.Zaretskyi // The Third International Workshop on Computer Modeling and Intelligent Systems – April 27-May 1. – 2020. – p. 288-299. (Scopus).

19. Moskalenko V. V. Multi-Layer Model And Training Method For Malware Traffic Detection Based On Decision Tree Ensemble / V. V. Moskalenko, A. S. Moskalenko, M. O. Zaretsky, A. M. Kudryavtsev, V. A. Semashko // Radioelectronic and Computer Systems – No 2. – 2020. – P. 92-101.

20. Інтелектуальна автономна бортова система безпілотного літального апарату для ідентифікації об'єктів на місцевості [Текст]: Розробка інформаційного та програмного забезпечення бортової системи безпілотного апарату, що функціонує в режимі навчання вирішальних правил та класифікаційного аналізу спостережень: звіт про НДР (проміжний) / кер. В.В. Москаленко. – Суми: СумДУ, 2019. – 67 с. – Режим доступу: https://essuir.sumdu.edu.ua/bitstream-download/123456789/79046/1/Moskalenko_intelektualna_1538.pdf

ДОДАТОК А

```
"""Implements soft triplet loss."""

import tensorflow as tf
from tensorflow_addons.losses import metric_learning
from tensorflow_addons.utils.keras_utils import LossFunctionWrapper
from tensorflow_addons.utils.types import FloatTensorLike, TensorLike
from typeguard import typechecked
from typing import Optional, Union, Callable


def _masked_maximum(data, mask, dim=1):
    """Computes the axis wise maximum over chosen elements.

    Args:
      data: 2-D float `Tensor` of size [n, m].
      mask: 2-D Boolean `Tensor` of size [n, m].
      dim: The dimension over which to compute the maximum.

    Returns:
      masked_maximums: N-D `Tensor`.
        The maximized dimension is of size 1 after the operation.
    """
    axis_minimums = tf.math.reduce_min(data, dim, keepdims=True)
    masked_maximums = (
        tf.math.reduce_max(
            tf.math.multiply(data - axis_minimums, mask), dim, keepdims=True
        )
        + axis_minimums
    )
    return masked_maximums


def _masked_minimum(data, mask, dim=1):
    """Computes the axis wise minimum over chosen elements.

    Args:
      data: 2-D float `Tensor` of size [n, m].
      mask: 2-D Boolean `Tensor` of size [n, m].
      dim: The dimension over which to compute the minimum.

    Returns:
      masked_minimums: N-D `Tensor`.
        The minimized dimension is of size 1 after the operation.
    """
    axis_maximums = tf.math.reduce_max(data, dim, keepdims=True)
```

```
    masked_minimums = (
        tf.math.reduce_min(
            tf.math.multiply(data - axis_maximums, mask), dim, keepdims=True
        )
        + axis_maximums
    )
    return masked_minimums


@tf.keras.utils.register_keras_serializable(package="Addons")
@tf.function
def soft_triplet_loss(
    y_true: TensorLike,
    y_pred: TensorLike,
) -> tf.Tensor:
    """Computes the triplet loss with semi-hard negative mining.

    Args:
      y_true: 1-D integer `Tensor` with shape [batch_size] of
        multiclass integer labels.
      y_pred: 2-D float `Tensor` of embedding vectors. Embeddings should
        be l2 normalized.

    Returns:
      triplet_loss: float scalar with dtype of y_pred.
    """

    labels, embeddings = y_true, y_pred

    convert_to_float32 = (
        embeddings.dtype == tf.dtypes.float16 or embeddings.dtype == tf.dtypes.bfloat16
    )
    precise_embeddings = (
        tf.cast(embeddings, tf.dtypes.float32) if convert_to_float32 else embeddings
    )

    # Reshape label tensor to [batch_size, 1].
    lshape = tf.shape(labels)
    labels = tf.reshape(labels, [lshape[0], 1])

    # Build pairwise squared distance matrix
    pdist_matrix = metric_learning.pairwise_distance(
            precise_embeddings, squared=False
    )

    # Build pairwise binary adjacency matrix.
    adjacency = tf.math.equal(labels, tf.transpose(labels))
    # Invert so we can select negatives only.
    adjacency_not = tf.math.logical_not(adjacency)
```

```
    batch_size = tf.size(labels)

    # Compute the mask.
    pdist_matrix_tile = tf.tile(pdist_matrix, [batch_size, 1])
    mask = tf.math.logical_and(
        tf.tile(adjacency_not, [batch_size, 1]),
        tf.math.greater(
            pdist_matrix_tile, tf.reshape(tf.transpose(pdist_matrix), [-1, 1])
        ),
    )
    mask_final = tf.reshape(
        tf.math.greater(
            tf.math.reduce_sum(
                tf.cast(mask, dtype=tf.dtypes.float32), 1, keepdims=True
            ),
            0.0,
        ),
        [batch_size, batch_size],
    )
    mask_final = tf.transpose(mask_final)

    adjacency_not = tf.cast(adjacency_not, dtype=tf.dtypes.float32)
    mask = tf.cast(mask, dtype=tf.dtypes.float32)

    # negatives_outside: smallest D_an where D_an > D_ap.
    negatives_outside = tf.reshape(
        _masked_minimum(pdist_matrix_tile, mask), [batch_size, batch_size]
    )
    negatives_outside = tf.transpose(negatives_outside)

    # negatives_inside: largest D_an.
    negatives_inside = tf.tile(
        _masked_maximum(pdist_matrix, adjacency_not), [1, batch_size]
    )
    semi_hard_negatives_dist = tf.where(mask_final, negatives_outside, negatives_inside)

    # easy positives: smallest D_ap.
    easy_positives_dist = _masked_minimum(pdist_matrix, mask_positives)

    triplet_loss_1 = -tf.math.log1p(  tf.math.exp(easy_positives_dist)  )
    triplet_loss_2 = -tf.math.log1p(
tf.math.exp(easy_positives_dist)+tf.math.exp(semi_hard_negatives_dist) )
    triplet_loss = triplet_loss_1 - triplet_loss_2

    # Get final mean triplet loss
    triplet_loss = tf.reduce_mean(triplet_loss)

    if convert_to_float32:
```

```
        return tf.cast(triplet_loss, embeddings.dtype)
    else:
        return triplet_loss

class SoftTripletLoss(LossFunctionWrapper):
    """Computes the soft triplet loss with easy positive and semi-hard negative mining.
    See: https://doi.org/10.1007/978-3-030-61656-4_16.

    We expect labels `y_true` to be provided as 1-D integer `Tensor` with shape
    [batch_size] of multi-class integer labels. And embeddings `y_pred` must be
    2-D float `Tensor` of l2 normalized embedding vectors.

    Args:
      name: Optional name for the op.
    """

    @typechecked
    def __init__(
        self,
        **kwargs
    ):
        super().__init__(
            soft_triplet_loss,
            name=name
        )
```

```
import math

class InformationExtremeLayer:

    def __init__(self, etalons=None, radiuses=None):
        if etalons is not None :
            self.etalons  = etalons
            self.class_num   = len(etalons)
            self.feature_num = len(etalons[0])
        if radiuses is not None :
            self.radiuses = radiuses

    def compute_etalons(self, X_train, y_train):
        self.class_num = len( np.unique(y_train) )
        self.feature_num = len(X_train[0])
        self.counter = { i: 0 for i in range(self.class_num)}
        self.etalons = np.zeros((self.class_num, self.feature_num))
        self.center = np.zeros(self.feature_num)
        self.n = len(X_train)
        for i in range(self.n):
            x = X_train[i]
            class_id = y[i]
            self.etalons[class_id] = self.etalons[class_id] + x
            self.center = self.center + x
            self.counter[class_id] = self.counter[class_id] + 1
        self.center = self.center / self.n
        for c in range(self.class_num):
            corrected_counter = max(self.counter[c], 1)
            self.etalons[c] = self.etalons[c] / corrected_counter
            self.etalons[c] = self.etalons[c] - self.center[c]
            self.etalons[c][ self.etalons[c]  > 0 ] = 1
            self.etalons[c][ self.etalons[c] <= 0 ] = 0

    def compute_max_radiuses(self, ):
        self.max_radius = np.ones(self.class_num)*self.feature_num
        for c in range(self.class_num):
            for k in range(self.class_num):
                distance = self.get_distance(self.etalons[c], self.etalons[k])
                if c != k :
                    if distance < self.max_radius[c] :
                        self.max_radius[c] = distance
        print("self.max_radius = ", self.max_radius)

    def criterion(self, fpr, fnr, sen, spe):
        com1 = com2 = com3 = com4 = 0
        if fpr+spe > 0 :
            com1 = fpr/(fpr+spe)
            com1 = com1*math.log2(com1) if com1>0 else 0
            com2 = spe/(fpr+spe)
```

```
            com2 = com2*math.log2(com2) if com2>0 else 0
        if fnr+sen > 0 :
            com3 = fnr/(fnr+sen)
            com3 = com3*math.log2(com3) if com3>0 else 0
            com4 = sen/(sen+fnr)
            com4 = com4*math.log2(com4) if com4>0 else 0
        return 1 + 0.5*(com1+com2+com3+com4)

    def get_distance(self, b1, b2):
        return np.count_nonzero(b1!=b2)

    def compute_distance_matrix(self, X_train):
        self.dist_matrix = np.zeros((self.class_num, self.n))
        for i in range(self.n):
            x = X_train[i]
            for c in range(self.class_num):
                e = self.etalons[c]
                self.dist_matrix[c][i] = self.get_distance(x, e)

    def optimize_radiuses(self, X_train, y_train):
        self.radiuses = np.zeros(self.class_num)
        fpr = fnr = sen = spe = 0
        for c in range(self.class_num):
            c2c = np.sort(self.dist_matrix[c][y_train==c])
            c2n = np.sort(self.dist_matrix[c][y_train!=c])
            nn = min(self.counter[c], len(c2n))
            c2n = c2n[0:nn]
            E_max = 0
            for r in range(self.feature_num):
                sen = len(c2c[c2c>r])
                fpr = self.counter[c] - sen
                fnr = len(c2n[c2n>r])
                spe = self.counter[c] - fnr
                sen = sen/self.counter[c]
                fpr = fpr/self.counter[c]
                fnr = fnr/self.counter[c]
                spe = spe/self.counter[c]
                E = self.criterion(fpr, fnr, sen, spe)
                if sen > 0.5 and spe > 0.5 and r < self.max_radius[c]:
                    if E > E_max :
                        E_max = E
                        self.radiuses[c] = r
            print( "radius    ["+str(c)+"] : "+ str(self.radiuses[c]) )
            print( "criterion ["+str(c)+"] : "+ str(E_max) )
            print( "*****************************")

    def fit(self, X_train, y_train):
        self.compute_etalons(X_train, y_train)
        self.compute_max_radiuses()
```

```
        self.compute_distance_matrix(X_train)
        self.optimize_radiuses(X_train, y_train)

    def predict(self, x_test):
        u = np.zeros(self.class_num)
        for c in range(self.class_num):
            e = self.etalons[c]
            dist = self.get_distance(x_test, e)
            u[c] = 1 - dist/self.radiuses[c]
        u = np.exp(u)
        u /= np.sum(u)
        return u
```

```
def create_baseline_model(classes_num, features_num, image_shape):
     classifier = tf.keras.Sequential([
        tf.keras.applications.MobileNetV2(
            input_shape=image_shape,
            alpha=0.35,
            include_top=False,
            weights='imagenet'
        ),
        tf.keras.layers.GlobalAveragePooling2D(),
        tf.keras.layers.Dropout(0.5),
        tf.keras.layers.Dense(features_num, activation='sigmoid'),
        tf.keras.layers.Dense(classes_num, activation='softmax')

     ])
     classifier.build(image_shape)
     classifier.compile(
         optimizer=tf.keras.optimizers.Adam(0.0001),
         loss=tf.keras.losses.CategoricalCrossentropy()
     )
     return classifier

def create_embedding_model(features_num, image_shape):
     embedding = tf.keras.Sequential([
        tf.keras.applications.MobileNetV2(
            input_shape=image_shape,
            alpha=0.35,
            include_top=False,
            weights='imagenet'
        ),
        tf.keras.layers.GlobalAveragePooling2D(),
        tf.keras.layers.Dropout(0.5),
        tf.keras.layers.Dense(features_num, activation='sigmoid'),
     ])
     embedding.build(image_shape)
     embedding.compile(
         optimizer=tf.keras.optimizers.Adam(0.0001),
         loss=SoftTripletLoss()
     )
     return embedding

from keras_preprocessing.image import ImageDataGenerator
from keras.preprocessing import image
from keras.preprocessing.image import ImageDataGenerator, array_to_img, img_to_array, loa
d_img
from keras.applications import imagenet_utils
from keras.applications import MobileNet, MobileNetV2
```

```
from keras.applications.mobilenet import preprocess_input, decode_predictions
import numpy as np
import cv2

datagen = ImageDataGenerator(
        preprocessing_function = preprocess_input,
        featurewise_center=True,
        featurewise_std_normalization=True,
        rotation_range=90,
        width_shift_range=0.2,
        height_shift_range=0.2,
        zoom_range = 0.2,
        horizontal_flip=True,
        vertical_flip=True,
        validation_split=0.25)

train_generator=datagen.flow_from_directory(directory=DATASET_PATH,
                                            class_mode="sparse",
                                            target_size=(IMG_SIZE, IMG_SIZE),
                                            batch_size=BATCH_SIZE,
                                            shuffle=True,
                                            seed=13,
                                            subset="training")

valid_generator=datagen.flow_from_directory(directory=DATASET_PATH,
                                            class_mode="sparse",
                                            target_size=(IMG_SIZE, IMG_SIZE),
                                            batch_size=BATCH_SIZE,
                                            shuffle=False,
                                            seed=13,
                                            subset="validation")



from tensorflow.keras.callbacks import TensorBoard, EarlyStopping, ModelCheckpoint
import numpy as np

run=0
run = run + 1 if 'run' in locals() else 1
LOG_DIR = TRAINING_DIR
LOG_FILE_PATH = LOG_DIR + '/embedding_checkpoint-{epoch:02d}-{val_loss:.4f}.hdf5'
checkpoint = ModelCheckpoint(filepath=LOG_FILE_PATH, monitor='val_loss', verbose=1, save_
best_only=True)
early_stopping = EarlyStopping(monitor='val_loss', patience=8, verbose=1)

test_steps_per_epoch = np.math.ceil(valid_generator.samples / valid_generator.batch_size)
```

```
train_steps_per_epoch = np.math.ceil(train_generator.samples / train_generator.batch_size
)


print("test_steps_per_epoch, train_steps_per_epoch = ", (test_steps_per_epoch, train_step
s_per_epoch))


history = embedding.fit_generator(generator=train_generator,
                                  steps_per_epoch=train_steps_per_epoch,
                                  validation_data=valid_generator,
                                  validation_steps=test_steps_per_epoch,
                                  epochs=EPOCHS,
                                  verbose=1,
                                  callbacks=[checkpoint, early_stopping])
```