МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
Факультет електроніки та інформаційних технологій

Кафедра комп'ютерних наук

Кваліфікаційна робота магістра

МОДЕЛЬ І МЕТОД РОБАСТНОГО МАШИННОГО НАВЧАННЯ ДЛЯ РОЗПІЗНАВАННЯ ДЕФЕКТІВ ТРУБОПРОВОДУ СИСТЕМИ ВОДОВІДВЕДЕННЯ

Здобувач освіти гр. ІН.м-01н А.М. Кудрявцев

Науковий керівник,
кандидат технічних наук, доцент В.В. Москаленко

Завідувач кафедри
доктор технічних наук, професор А.С. Довбиш

СУМИ 2022

Сумський державний університет

(назва вузу)

Факультет *ЕлІТ* Кафедра *Комп'ютерних наук*

Спеціальність *«122 - Комп'ютерні науки»*

Затверджую:

зав.кафедрою _____

_____

"_____" _____ 20___р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТОВІ

Кудрявцеву Антону Михайловичу

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) *Модель і метод робастного машинного навчання для розпізнавання дефектів трубопроводу системи водовідведення*

затверджую наказом по інституту від "____" _____ 20 ___ р. № _____

2. Термін здачі студентом закінченого проекту (роботи)_____

3. Вхідні данні до проекту (роботи) _____

_____

_____

_____

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) аналіз проблеми аналізу трубних інспекцій; 2) формалізована постановка задачі дослідження; 3) опис інформаційної технології; 4) опис програмної реалізації; 6) аналіз результатів.*

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

_____

6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання _____

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів дипломного проекту (роботи)	Термін виконання проекту (роботи)	Примітка
1.	Аналіз проблеми аналізу трубних інспекцій		
2.	Формалізована постановка задачі дослідження		
3.	Опис інформаційної технології		
4.	Опис програмної реалізації		
5.	Аналіз результатів		
6.	Оформлення пояснювальної записки до кваліфікаційної магістерської роботи		

Студент – дипломник _____
(підпис)

Керівник проекту _____
(підпис)

РЕФЕРАТ

Записка: 55 стор., 17 рис., 3 табл., 1 додаток, 24 джерела

Об'єкт дослідження — слабоформалізований процес розпізнавання дефектів стічних труб.

Мета роботи — розробка ефективних моделі та алгоритму навчання для розпізнавання дефектів стічних труб за даними відеоінспекції в умовах неповної визначеності.

Результати — розроблено метод машинного навчання глибокої мережі, що полягає в ініціалізації вагових коефіцієнтів на нерозмічених даних з використанням самонавчання та уточненні вагових коефіцієнтів з використанням контрастно-центрованої функції втрат з регуляризуючою складовою для бінаризації ознакового опису і побудови радіально-базисних інформаційно-екстремальних вирішувальних правил.

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ, МАШИННЕ НАВЧАННЯ, МОДЕЛІ МАШИННОГО НАВЧАННЯ, PYTHON, NUMPY, KERAS, OPENCV

ЗМІСТ

ВСТУП

Системи водовідведення здійснюють транспортування стоків від підприємств та домогосподарств до очисних споруд і забезпечують захист інфраструктури від підтоплення, екологічного забруднення та руйнування дорожнього покриття внаслідок підмивання грунту. Системи водовідведення швидко зношуються і забиваються і потребують частого моніторингу. Тому муніципальні служби регулярно здійснюють перевірку функціонально стану мережі трубопроводів системи водовідведення стоків. Найкращим з точки зору інформативності та економічної ефективності методом інспекції стічних труб є відеоінспекція та кодування відео даних у звіти згідно стандартів.

Звіти про інспекцію забезпечують ефективне планування ремонтних заходів, особливо в умовах обмежених ресурсів. Однак досі формування звітів здійснюють ліцензовані фахівці, що проглядають відео. Такий процес є дорогим і повільним. Існуючі технології автоматизації щодо обробки відео з використанням технологій штучного інтелекту не забезпечують прийнятної достовірності і вимагають додаткової перевірки звітів людиною. Основні ускладнення пов’язані з неповною визначеністю даних, що обумовлена шумами, артефактами, непередбачуваними неполадками камери чи втратою видимості, схожістю бризків води і піни на дефекти. Крім того існує висока варіативність конфігурації складних дефектів, часта зміна контексту внаслідок поворотів камери і зміни матеріалу труби, а також схожість одних дефектів на інші при їх засвічуванні, що обумовлює неоднозначність кодування. Тому розробка нових моделей і методів інтелектуального аналізу даних відеоінспекції стічних труб за умов високої варіативності спостережень та обмеженого обсягу розмічених даних є актуальним завданням.

Мета роботи – розробка ефективних моделі та алгоритму навчання для розпізнавання дефектів стічних труб за даними відеоінспекції в умовах неповної визначеності.

Об’єкт дослідження — слабоформалізований процес розпізнавання дефектів стічних труб.

Предмет дослідження — моделі і методи інформаційної технології машинного навчання та розпізнавання дефектів стічних труб на кадрах відеоінпекції.

Методи дослідження — методи технології нейронних мереж, інформаційно-екстремальної інтелектуальної технології.

Для досягнення мети дослідної роботи здійснюється формування вхідних навчаних та тестових вибірок з наборів даних SEWER-ML, розроблено модель та метод машинного навчання. Здійснюється оцінка ефективності запропонованих алгоритмів за тестовими даними.

Робота має теоретичний та прикладний характер, запропоновано нову нейромережеву модель та новим багатофазний метод її навчання і порівняно запропонований підхід з традиційним. Результати, отримані на вибірках з відкритих наборів даних SEWER-ML підтверджують придатність моделей і алгоритмів навчання для практичного використання. Запропонований алгоритм навчання використовує більще 200 розмічених навчальних зразків на клас та 10000 нерозмічених зразків, забезпечуючи 93% правильного розпізнавання дефектів стічних труб.

РОЗДІЛ 1
АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Сучасний стан та тенденції розвитку систем моніторингу труб системи водовідведення

Основною складовою інфраструктури водовідведення є стічні труби, що швидко зношуються та замічуються, а з часом і руйнуються. В деяких випадках, якщо дефектив виявлені вчасно, то ремонт можна здійснити за спеціальними технологіями без розриття трубопроводу, що проходить під дорожнім покриттям. Існує багато видів дефектів, кожне з яких має різну тяжкість, а за їх сукупністю можна спрогнозувати час ремонтних робіт. Розрізняють структурні, конструктивні та експлуатаційні дефекти (рис. 1.1) [1, 2].

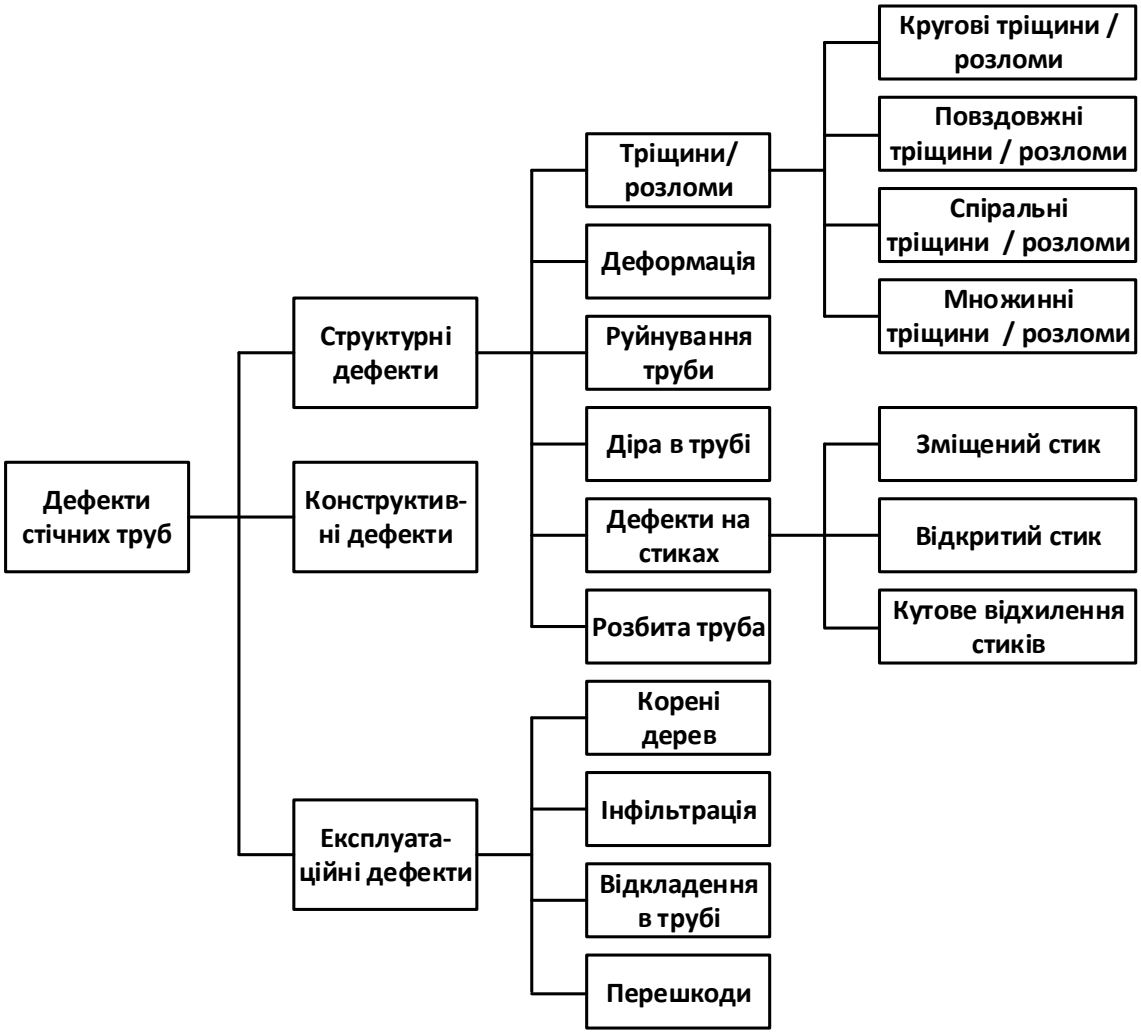


Рисунок 1.1 – Основні типи дефектів стічних труб

Група структурних дефектів згідно поширених стандарту PACP та MCCS характеризує різні типи пошкодження труб [2] : тріщина, розлом (зазор більше 3 мм), розбита труба, діра в трубі, деформована труба, зруйнована труба, зміщений чи відкритий стик, пошкодження внутрішньої поверхні. Найбільш вразливою частиною трубопроводу є місця стику труб, в даній ділянці виникає переважна більшість дефектів. В нових версіях стандартів враховуються властивості внутрішнього покриття (Lining Features), пошкодження зварного шву (Weld Failure), точковий (ділянковий) ремонт (Point Repair). Варіативність структурних дефектів породжується різницею механізмів і ймовірностей утворення та розвитку структурних дефектів, які мають свої особливості для жорстких (rigid) та гнучких (flexible) труб.

Жорсткі труби включають в себе: азбестоцементні Asbestos Cement (AC), бетонні Concrete Pipe (CP, RCP), глазуровані керамічні Vitrified Clay (VCP), чавунні Cast Iron (CAS) та інші. Вони більш схильні утворювати тріщини та розломи. Якщо їх залишити без уваги і не вжити заходів, то потім ці дефекти можуть розвинутися у розбиті труби, діри чи деформації, та, зрештою, призвести до руйнування труби.

Гнучкі труби включають в себе: гофровані металічні (Corrugated Metal Pipe), труби з рівного та гофрованого пластику (Plastic), стальні труби (Steel Pipe), труби з ковкого чавуну (Ductile Iron Pipe), армовані пластикові труби (Reinforced Plastic Pipe), бітумно-волоконні (Orangeburg), стальні спірально-зварені труби, скловолоконні армовані та інші. У відповідь на зовнішнє навантаження вони деформуються, спочатку не утворюючи тріщин та розломів. Ступінь допустимої деформації без утворення інших дефектів залежить від матеріалу труби, навантаження та інших факторів. У трубах з гнучких матеріалів, окрім деформацій, також можуть утворюватися тріщини та розломи, вони можуть розбиватися та у них можуть утворюватися діри, та, зрештою, вони також можуть руйнуватися.

На сьогоднішній день існує чотири основні групи методів для виконання інспекції стічних труб : методи, основані на використанні технічного зору, тобто відеокамер; структурні та застилаючі методи оцінювання цілісності стінок труб; специфічні методи, спрямовані на детектування специфічних дефектів; гібридні

методи, що поєднуєть декікальа інших. На рис. 1.2 показано узагальну схему класифікації методів інспекції стічних труб.

Рисунок 1.2 – Види інструментів інспекції трубопроводів

Найбільш простим і поширеним методом інспекції стічних труб є використання відеокамер, що закріплені на гусеничне шасі чи поплавок. Такий пристрій називають квалером. Кравлером дистанційно керує оператор через ккабельне з’єднання для повороту і фокусування камери на підозрілих ділянках. Камера фікусує стан стінок труби, що знаходяться вище рівня стічних вод. Сам рівень стічних вод теж може давати інформацію про просадку труби підв впливом зовнішнього навантаження. Отримані відео детальніше аналізують сертифіковані спеціалісти для кодування функціонального стану труби згідно існуючих стандартів. На рис. 1.3 показано кадр відеоінспекції стічних труб.

Для збільшення інформативності відеоінспекції деякі виробники кравлерів її здійснюють з використанням двох відеокамер високої роздільної здатності з кутом огляду 360 градусів – камери типу “риб’яче око”. Круговий огляд трубопроводу

дозволяє здійснювати цифрове сканування і реконструювати як фронтальний вид так і розгортку вигляду стінок з бічних сторін (рис. 1.8).

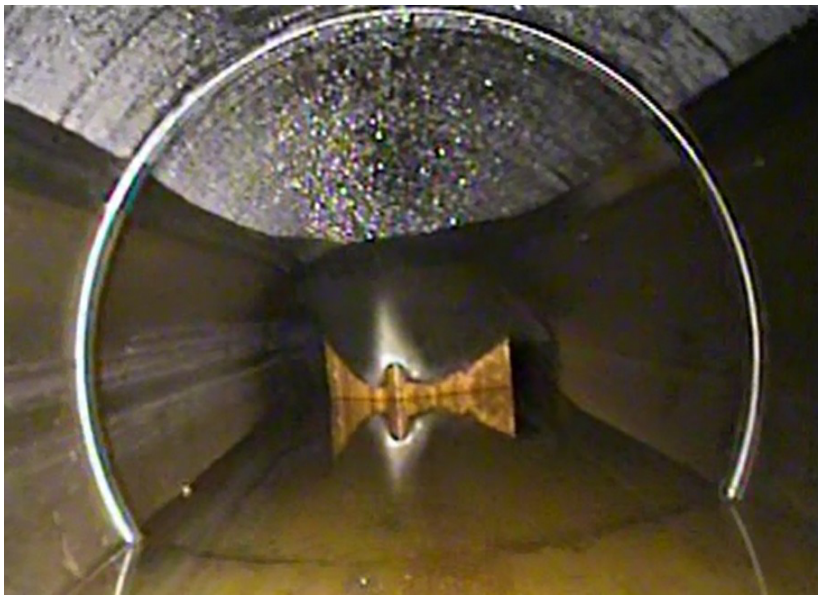


Рисунок 1.3 – Кадр відеоінспекції стічної труби

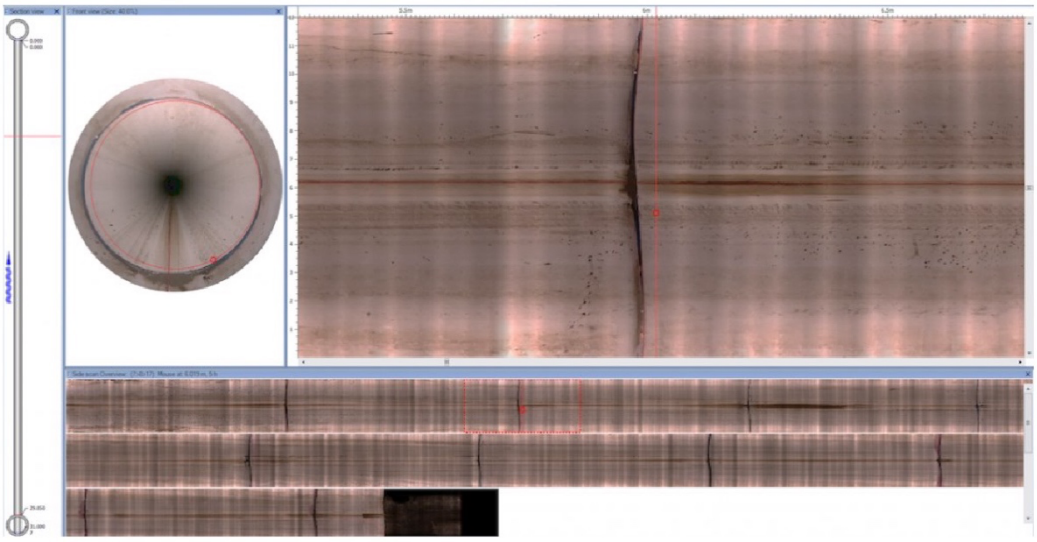


Рисунок 1.4 – Результат цифрового сканування бічних стінок труби

Незважаючи на надання детального панорамного виду метод цифрового сканування не значно підвищує точність виявлення дефектів, проте швидкість інспекції знижується в 2-3 рази порівняно з використанням звичаної відеокамери [3].

Для врахування невидимих факторів руйнування стічних труб використовують відеокамери, що фіксують випромінювання в інфрачервоному спектрів. Дана

інформація дозволяє виявити місця витоку тепла, проте необхідно враховувати матеріал стічних труб, тип і вологість грунту прокладання труб, вологість і температура повітря та інші атмосферні характеристики [4, 5]. На рис. 1.5 показано приклад суміщення кадру звичайної та інфрачервоної камери під час інспекції системи водовідведення.

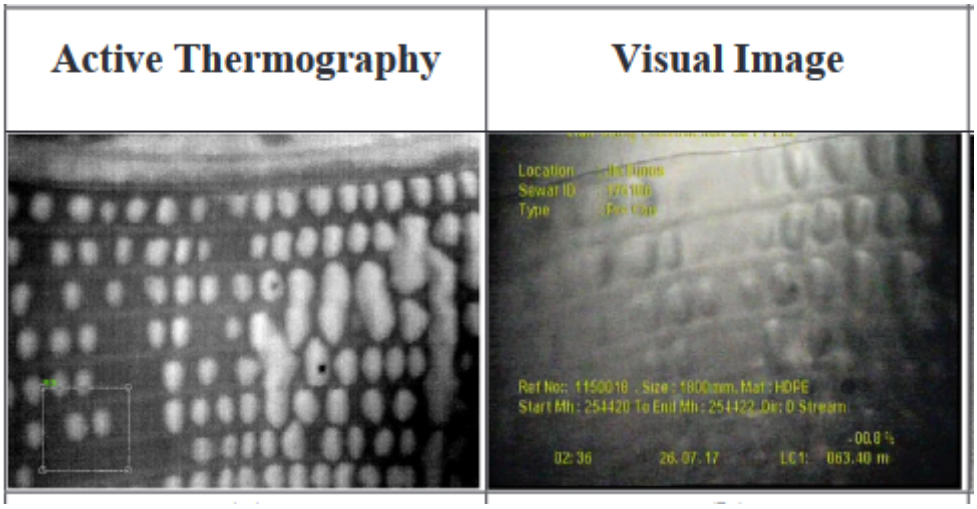


Рисунок 1.5 – Результат інфрачервоного сканування бічних стінок труби

Для підвищення точності фіксації геометричних характеристик стінок стічної труби у деяких кравлерах використовується лазерний профайлер, що фіксує бідь-які зміни у вертикальній і горизонтальній формі трубопроводу. Розрізняють двовимірні та тривимірні лазерні профайлери. Двовимірні профайлери формують лазерне кільце на стінках трубопроводу, що фіксується відеокамерою [6]. Тривимірні профайлери використовують лазерні пучки, що фіксуються примачами, що забезпечує формування тривимірної діаграми (рис. 1.6).

Для отримання діагностичної інформації з внутрішньої частини стінок стічної труби використовують акустичні сонари на основі ультразвукового випромінювача і приймача. Відбивання хвиль залежить від зміни щільності матеріалу. Такий метод може використовуватися для інспекції труб з будь-якого матеріалу і він забезпечує досить точну кількісне і якісне оцінювання значних відкладень та осадів на стінках, однак малочутливий до дрібних структурних дефектів. Як правило швидкість руху крвалера з акустичних сканером становить біля 100 мм за секунду [3]. На рис. 1.7 показано зображення, сформоване ультразвуковим сканером.

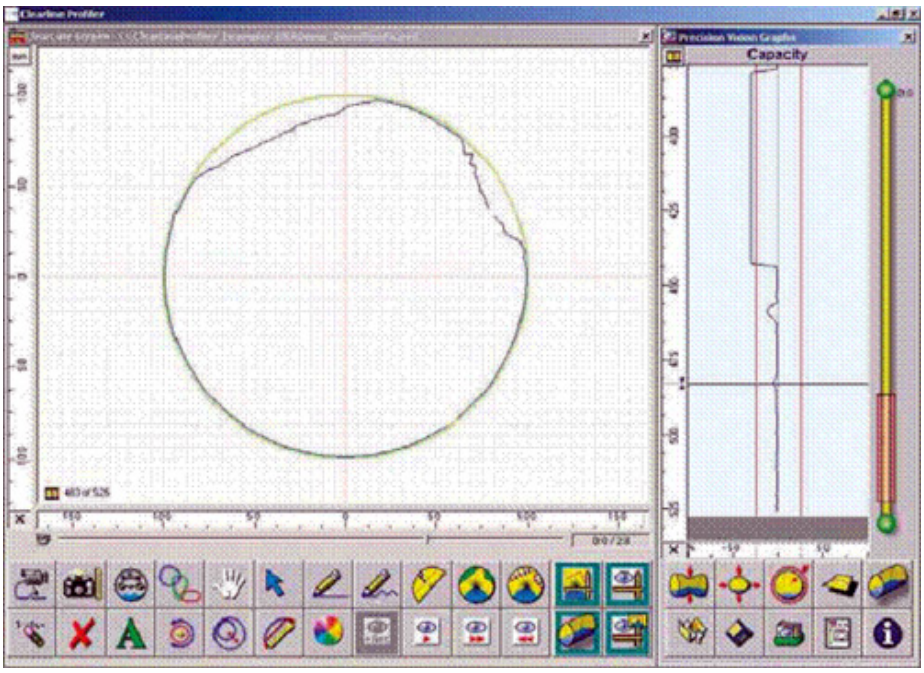


Рисунок 1.6 – Результат дослідження лазерним профайлером

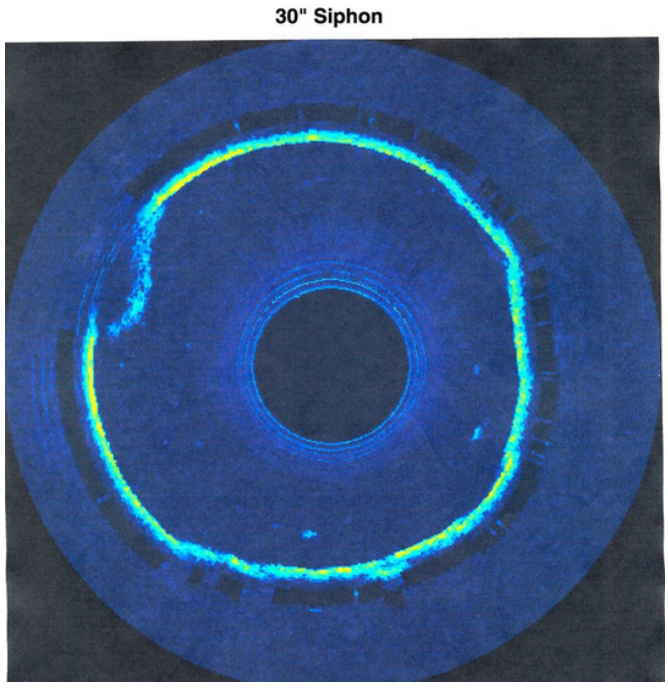


Рисунок 1.7 – Результат дослідження сонарним профайлером

Для глибшого сканування стінок трубопроводів також використовують методи підповерхневого сканування на основі георадарів. Система сканування містить декілька антен, що випромінюють радіохвилі, а сканування здійснюється за принципами радіолокації. Така система надає інформацію про товщину стінок,

підкладки, стан грунту та інше. Як правило, георадар є допоміжною системою для кравлерів з відеоспостереженням.

Існує цілий ряд методів, що спрямовані на детектування специфічних дефектів. Наприклад, електросканер, що використовується для виявлення порушення цілісності струмонепровідних труб на основі електрострумів, що фіксуються зондом поза межами труби. Однак подібні методи можуть виявляти поодинокі типи дефектів, але потребують високої кваліфікації для інтерпретації результатів.

Для компенсації недоліків і поєднання переваг різних методів реалізують гібридні методи, однак такі технології знаходяться на стадії прототипів і мають високу ціну. Прикладом такого підходу є об’єднання цифрового сканування і гороскопу, або камер відеоспостереження та лазерних профайлерів, або поєднання відразу декількох технологій.

Технології відеозйомки покищо залишаються найбільш ефективними з точки зору кількості типів дефектів, що можуть бути виявлені, вартості і простоти експлуатації. Інформативність технології відеоінспекції підвищують за рахунок збільшення роздільної здатності камер та впровадження інтелектуальних інформаційних технологій для автоматизації аналізу відеоспостережень [6]. Такі стартап компанії, як SewerAI, Molfar.AI та WinCan пропонують використання технології штучного інтелекту для кодування відеінспекції в звіти згідно стандартів [6, 7]. Однак досі існує потреба у висококваліфікованих операторах для остаточної перевірки звітів.

Таким чином, відеоінспекції характеризується найменшими витратами на збір даних про функціональний стан стічного трубопроводу і забезпечує виявлення переважної кількості дефектів. Однак найбільш трудозатратним етапом залишається кодування відеоспостережень в звіти про інспекцію. Технології штучного інтелекту для автоматизації кодування звітів є недосконалими і підвищення їх ефективності є актуальною задачею.

1.2 Моделі і методи класифікаційного аналізу даних

Детектування дефектів стічних труб з використанням технології машинного зору досліджується давно. У праці [2] було досліджено використання детектування країв за алгоритмом Canny для виявлення потенційних зон інтересу. Даний підхід вимагає використання великої кількості ручних настройок та правил для врахування контексту і типу матеріалу труб. Крім тоого не було розглянуто етап класифікаційного аналізу дефектів в зоні інтересу.

У працях [2, 3] для ознакового опису спостережень під час інспекції стічних труб було запропоновано використовувати фільтри Габора (Gabor filter), а класифікаційний аналізу здійснювати на основі класичних моделей і методів машинного навчання (рис. 1.8).

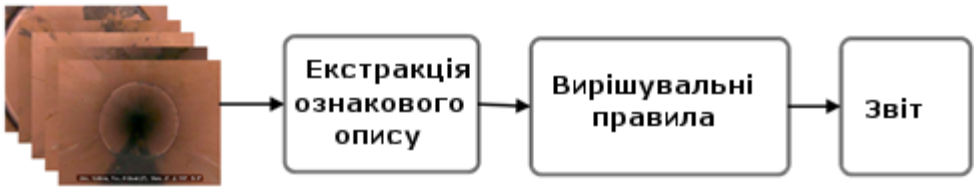


Рисунок 1.8 – Підхід до аналізу кадрів відеоінспекції на основі Класичного машинного навчання

Підвищення інформативності ознакового подання відеоспостережень потребує значних зусиль щодо проектування і настройки екстрактора на основі фільтрів Габора для врахування контексту, матеріалу і впливу артефактів. Крім того відокремення етапу екстракції ознакового опису від етапу машинного навчання зменшує можливості щодо автонастройки і зменшує рівень автоматизації процесу синтезу моделі розпізнавання дефектів.

Для реалізації машинного навчання за принципом з-кінця-в-уінець у працях [3, 4] запропоновано використовувати глибокі моделі аналізу візуальних даних, що забезпечують ієрархічне ознакове подання з необхідним рівнем інваріантності і інформативності (рис. 1.9). Але навіть глибоке машинне навчання не забезпечує повної автоматизації синтезу оптимальної моделі. Існує потреба у виборі архітектури мережі, способу ініціалізації вагових коефіцієнтів та цілої низки гіперпараметрів. При

цьому чим менший обсяг розміченої навчальної вибірки тим більша чутливість до вибору архітектури, гіперпараметрів та початкової ініціалізації.

Рисунок 1.9 – Підхід до аналізу даних на основі глибокого машинного навчання

Найбільш універсальними підходами для зменшення простору пошуку оптимальних параметрів моделі є вибір однієї з попередньо навчених моделей на певному великому наборі даних. Такий метод називається переносом знань (Transfer Learning). Проте ефективність даного підходу залежить від схожості наборів даних, на яких навчається модель-донор, на навчальні набори цільової задачі. Часто між задачами наскільки малий статистичний зв'язок, що навіть запозичення вагових коефіцієнтів нижніх шарів не підвищує швидкість навчання і точність результуючої моделі. Інший підхід до ініціалізації вагових коефіцієнтів мережі полягає у використанні самонавчання на нерозмічених навчальних даних [7], або самонавчання на синтетичних шумових і текстурних шаблонах [8].

Існує три основні підходи до реалізації алгоритмів самонавчання : генеративні, контрастні та генеративно-контрастні [7]. Генеративне самонавчання полягає у навчанні кодера та декодера, де кодер перетворює вхідний вектор x в прихований вектор z, а декодер реконструює x з z. До генеративних моделей відносяться знешумлюючі та варіаційні автоенкодери, які відновлюють вхідний розподіл даних, проте ігнорують інформацію про задачу, яку необхідно вирішувати з використанням отриманого ознакового подання. В рамках констратного навчання здійснюється навчанання кодувальника $f(x)$ формувати векторне подання вхідних даних таким чином, що дає змогу розрізняти подібність об’єктів, з’ясування відповідності однаковим та різним категоріям чи доменам. Тобто контрастне навчання співпадає з метою класифікаційного аналізу даних і на практиці забезпечує формування

інформативного ознакового подання. Крім того навчання такої модел дозволяє позбавитися декодера, що знижує обчислювальну трудомісткість методу. На рис. 1.10 показано процес обчислення функції втрат під час констратного навчання, де показано, що через один і той же кодувальник пропускається якірний зразок x , позитивний зразок x^+, що належить тій же категоріх чи домену, та негативний зразок x^-, що належить іншій ніж якірний зразок категорії чи домену.

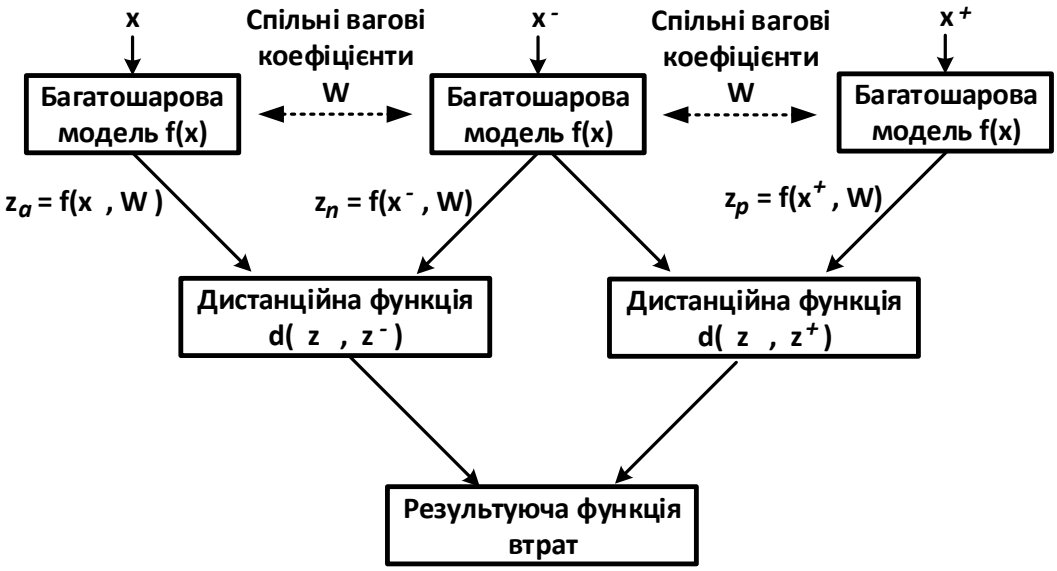


Рисунок 1.10 – Ілюстрація до контрастного методу машинного навчання

Якірний, позитивний і негативний зразки пропускаються через кодувальник і формують відповідні ознакові подання z, z^+ та z^-, між якими обчислюються косинусна чи евклідова відстань і диференційована функція втрат. У загальному випадку для контрастного навчання використовується функція шумового контрастного оцінювання [7]

$$J = E_{x,x^+,x^-}[-\log\left(\frac{e^{f(x)^T f(x^+)}}{e^{f(x)^T f(x^+)} + e^{f(x)^T f(x^-)}}\right)], \tag{1.1}$$

Під експонентами знаходяться косинусні відстані, що при нормалізації вихідного вектора дорівнюють скалярному добутку векторів ознак відповідних зразків. Якщо використовувати відстань Евкліда, то необхідно змінити знак у

підекспонентному виразі. Якщо є можливість отримати велику кількість негативних зразків для якірного зразка, то можна використати функцію втрат InfoNCE, що забезпечує оцінювання взаємної інформації між позитивною парою і відповідними K негативними парами [7],

$$J = E_{x,x^+,x^k}[-\log\left(\frac{e^{f(x)^T f(x^+)}}{e^{f(x)^T f(x^+)} + \sum_{k=1}^{K} e^{f(x)^T f(x^k)}}\right)]. \qquad (1.2)$$

Для забезпечення побудови оптимального ознакового подання як правило спираються на гіпотезу багатовидів та принцип інформаційного пляшкового горла. При цьому враховується необхідність надмірності розмірності ознакового подання для реалізації приципів завадозахищеного кодування [9, 10]. Професор Н. Тішбі [9] запропонував шукати баланс між компресією даних X у ознакове подання $\tilde{X}$ та збереженням максимальної кількості інформації про цільову змінну Y

$$J = I(\tilde{X}; X) - \beta I(\tilde{X}; Y), \qquad (1.3)$$

де $I(\bullet)$ – взаємна інформація між двома змінними;

$\beta > 0$ – параметр компромісу між складністю ознакового подання та кількістю збереженої актуальної для задачі інформації.

Для стиснення інформації і впровадження кодів, що виправляють помилки у працях [11] було запропоновано різноманітні варіанти формування дискретного ознакового подання спостережень. 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 коригують помилки (Error Correcting Output Codes, ECOC) [10]. Під час використання кодів, що коригують помилки, номер класу записують у вигляді *k*-значного двійкового числа. Для цього здіснюють навчання *k*-класифікаторів, кожен із яких

розпізнає один із *k* розрядів номера класу. За результатами розпізнавання вхідного вектора кожним із класифікаторів однозначно відновлюють номер класу, до якого він належить. Якщо окремі класифікатори помиляються, то номер класу відновлюють методом заміни одержаного номера номером класу, що найближчий до одержаного за метрикою. У працях [12] показано недоліки повнозв'язних вихідних шарів з функцією softmax порівняно зі дистанційними функціями належності або схемою ECOC. Основним недоліком методу ECOC є ігнорування структури класів під час побудови кодової матриці без можливості оптимізації коду кожного класу в процесі навчання. Крім того, відсутність кодового радіуса для кожного номера класу, що вказує на кратність помилок, які можуть бути виправлені, ускладнює виявлення викидів або новизни в даних.

Таким чином, глибоке машинне навчання є найбільш сучасним і ефективним підходом до синтезу моделей аналізу даних, зокрема класифікаційних. При цьому в глибокому навчанні особливого значення набуває початкова ініціалізація вагових коефіцієнтів, особливо в умовах обмеженого обсягу розмічених навчальних даних. Класифікаційні вирішувальні правила можуть бути побудовані за принципами кодів, що виправляють помилки, однак в цьому випадку формування кодової матриці необхідно поєднати з етапом формування ознакового опису для врахування внутрішньої структури даних.

1.3 Формалізована постановка задачі

Нехай дано відкритий набір даних SEWER-ML з якого відібрано і верифіковано з фахівцями по 300 зразків на клас для алфавіту з 13 класів. Алфавіт класів включає такі класи дефектів: 1) порушення цілісності прокладки на стику; 2) відкритий стик; 3) зміщений стик; 4) тріщина; 5) розлом; 6) розбита труба; 7) дірка в трубі; 8) пошкодження поверхні; 9) прикріплені відкладення; 10) осад; 11) тоненькі корені; 12) значне розростання коренів; 13) нормальний стан труби. Оскільки по протоколу інспекції оператор кравлера зупиняє, повертає камеру і фокусує на всіх підозрілих ділянках, то розпізнавання здійснюватиметься шляхом класифікації вього відеокадру. Також для цілей ініціалізації екстрактора ознак дано 10000 нерозмічених навчальних

зразків з набору даних SEWER-ML.

Дано структурований вектор параметрів функціонування моделі розпізнавання шкідливого програмного забезпечення, який у загальному випадку має структуру

$$g =< e_1,...,e_{\xi_1},...,e_{\Xi_1}, f_1,...,f_{\xi_2},...,f_{\Xi_2} >, \quad (1.4)$$

$$\Xi_1 + \Xi_2 = \Xi \ ,$$

де $< e_1,...,e_{\xi_1},...,e_{\Xi_1} >$ – параметри функціонування системи розпізнавання дефектів, що впливають на формування ознакового опису відеокадрів інспекції;

$< f_1,...,f_{\xi_2},...,f_{\Xi_2} >$ – параметри функціонування системи, які прямо впливають на ефективність правил розпізнавання дефектів.

При цьому відомі обмеження на відповідні параметри функціонування :

$$R_{\xi_1}(e_1,...,e_{\xi_1},...,e_{\Xi_1}) \le 0;\ R_{\xi_2}(f_1,...,f_{\xi_2},...,f_{\Xi_2}) \le 0.$$

Необхідно знайти оптимальні значення параметрів вектора g (1), що забезпечують максимальне значення мікро-усередненого значення F1-міри для класифікатора дефектів на стінках труби

$$Fl = \frac{2\sum_{m}^{M} TP_m}{2\sum_{m}^{M} TP_m + \sum_{m}^{M} FP_m + \sum_{m}^{M} FN_m}, \quad (1.3.2)$$

$$g^* = \arg\max_{G} \{Fl(g)\}. \quad (1.5)$$

де TP_m – значення лічильника правильно-позитивних рішень для m-го класу на тестовій вибірці;

FP_m – значення лічильника хибно-позитивних рішень для m-го класу на тестовій вибірці;

FN_m – значення лічильника хибно-негативних рішень для m-го класу на тестовій вибірці.

При функціонуванні системи розпізнавання безпосередньо в робочому режимі необхідно забезпечити максимальну точність класифікації дефектів трубопроводу водовідведення.

РОЗДІЛ 2
ОПИС ТЕХНОЛОГІЇ МАШИННОГО НАВЧАННЯ ДЛЯ РОЗПІЗНАВАННЯ ДЕФЕКТІВ ТРУБОПРОВОДУ СИСТЕМИ ВОДОВІДВЕДЕННЯ

2.1 Модель і метод навчання класифікатора дефектів на кадрах відеоінспекції трубопроводу системи водовідведення

Для екстракції ознакового опису кадрів відеоінспекції трубопроводу системи водовідведення пропонується використовувати згорткову багаташарову нейронну мережі. Для економії ресурсів і перевірки концепції пропонується використати мережу MobileNet з коефіцієнтом кількості фільтрів, що дорівнює 0,5, та роздільною здатністю 224x224 пікселів. Для порівняння з традиційним підходом пропонується одночасно побудувати нейромережу з таким же екстрактором ознак, але з класичним вихідним шаром (рис. 2.1).

MobileNet екстрактор
Global Average Pooling 2D шар
Dropout шар (rate=0.5)
Dense шар (128 вузлів)
ReLU шар
Dense шар (M вузлів)
Softmax шар

а

MobileNet екстрактор
Global Average Pooling 2D шар
Dropout шар (rate=0.5)
Dense шар (128 вузлів)
Сігмоїдний шар
Округлюючий шар
Радіально-базисний шар

б

Рисунок 2.1 – Архітектура згорткової мережі :
а – традиційний варіант; б – модифікований варіант

Для реалізації інформаційного пляшкового горла та кодів, що виправляють помилки пропонується здійснювати дискретизацію (бінаризацію) ознак на виході сігмоїдного шару і врахувати це у функції втрат під час навчання. До двійкового ознакового опису застосовується радіально-базисна функція належності до класів, що розділяються гіперсферичними контейнерами в просторі Хемінга. Центр контейнера описується вектором-прототипом класу, а межі контейнера визначаються кодовим радіусом. Функція належності $\mu_m(z)$ до z-го класу для N-вимірного двійкового

вектора z може бути обчислена за формулою

$$\mu_m(z) = 1 - \sum_{i=1}^{N} z_i \oplus z^*_{m,i} / d^*_m, \qquad (2.1)$$

де z^*_m – оптимальний двійковий еталонний вектор класу X^o_m;

d^*_m – оптимальний кодовий радіус в просторі Хеммінга для контейнера для класу X^o_m.

Запропонований метод поділяється на три етапи машинного навчання (рис. 2.2) [18, 19]. На першому етапі передбачається використання нерозмічених навчальних даних для ініціалізації (попереднього навчання) вагових коефіцієнтів згорткової мережі шляхом процедури контрасного самонавчання (self-supervised learning). Для реалізація самонавчання був обраний фреймворк SimCLR (A Simple Framework for Contrastive Learning of Visual Representations), де згорткова мережа формує ознакове подання пар аугментованих зразків, для яких обчислюється констрастна функція втрат.

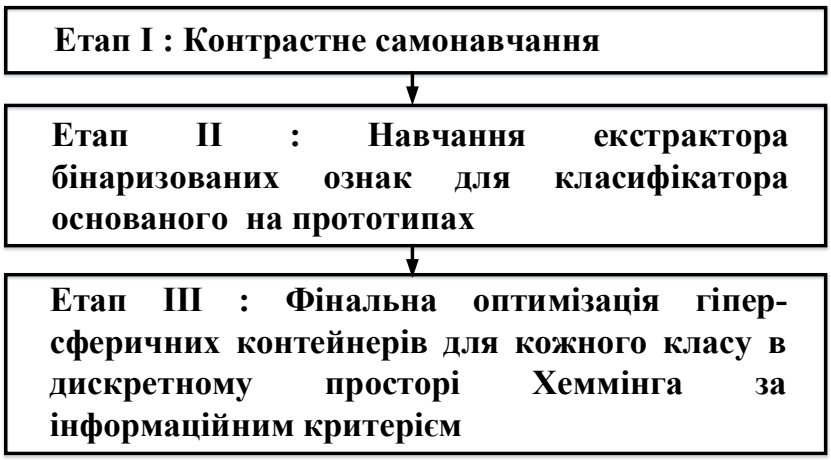


Рисунок 2.2 – Етапи запропонованого методу навчання

SimCLR - це простий фреймворк для порівняльного навчання візуальних представлень. Його основна задача - вивчення візуальних представлень вхідних даних, максимізуючи узгодження між різними аугментованими представленнями

однієї вибірки через contrastive loss у прихованому просторі. На рис. 2.1 схематично зображено принцип роботи даного фреймворку.

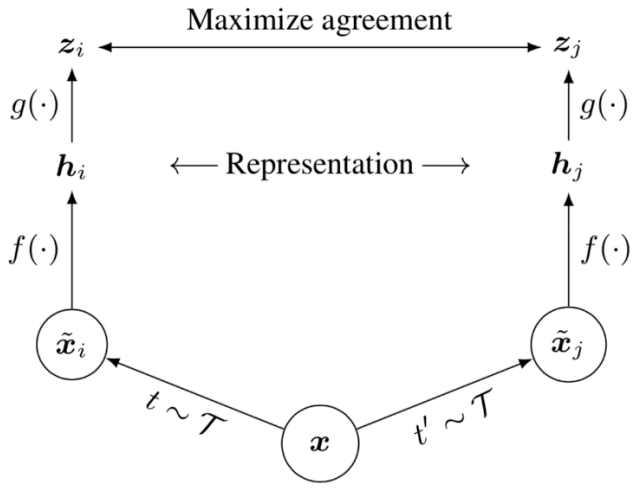


Рисунок 2.2 – Схематичне зображення методу навчання в рамках SimCLR

Даний фреймворк працює в наступні кроки:

1) Випадково випробовують міні-батч з n зразків, та до кожного зразка застосовується два різні оператори аугментування, в результаті маємо $2n$ аугментованих зразків

$$\widetilde{x_i} = t(x),\ \widetilde{x_j} = t(x),\ t_i t' \sim T \tag{2.2}$$

де t та t' – да різні оператори аугментування, відібрані з одного набору аугментацій T.

До аугментуваці даних необхідно додаткове обрізання, збільшення розміру з випадковим повертанням, спотворення кольорів та розмиття Гауса.

2) Дано один позитивний парний зразок та $2(n-1)$ негативні зразки. Ознакове подання отримується за допомогою базового екстрактора ознак $f(.)$

$$h_i = f(\widetilde{x_i}),\ h_j = f(\widetilde{x_j}) \tag{2.3}$$

3) Обчислюється констрастна функція втрат (contrastive loss) L за допомогою косинусної міри подібності $sim(.,.)$. Зверніть увагу, що функція втрат обчислюється і для додаткової проекцієї ознакового подання через шари $g(.,.)$. Але лише ознакове подання h з базового екстрактора ознак використовується для подальших завдань

$$L_{ij} = -log \frac{\exp\left(\frac{sim(z_i z_j)}{\tau}\right)}{\sum_{k=1}^{2n} 1_{k \neq j} \exp\left(\frac{sim(z_i z_j)}{\tau}\right)} \quad (2.3)$$

де $1_{k \neq j}$ функція індикатор: 1 якщо $k \neq j$, у противному випадку 0;

- $z_i = g(h_i)$;
- $z_j = g(h_j)$;
- $sim(z_i z_j) = \frac{z_i z_j}{|z_i||z_j|}$;
- τ – гіперпараметр температури [10].

4) Здійснюється корекція вагових коефіцієнтів нейронної мережі на основі обчислення градієнту функції L за кожним з параметрів і зворотнім поширенням помилки (backpropagation algorithm)

На рис. 2.3 зображено псевдокод роботи алгоритму SimCLR.

Algorithm 1 SimCLR's main learning algorithm.

input: batch size N, constant τ, structure of $f, g, \mathcal{T}$.
for sampled minibatch $\{\boldsymbol{x}_k\}_{k=1}^{N}$ **do**
  for all $k \in \{1, \ldots, N\}$ **do**
    draw two augmentation functions $t \sim \mathcal{T}$, $t' \sim \mathcal{T}$
    # the first augmentation
    $\tilde{\boldsymbol{x}}_{2k-1} = t(\boldsymbol{x}_k)$
    $\boldsymbol{h}_{2k-1} = f(\tilde{\boldsymbol{x}}_{2k-1})$ # representation
    $\boldsymbol{z}_{2k-1} = g(\boldsymbol{h}_{2k-1})$ # projection
    # the second augmentation
    $\tilde{\boldsymbol{x}}_{2k} = t'(\boldsymbol{x}_k)$
    $\boldsymbol{h}_{2k} = f(\tilde{\boldsymbol{x}}_{2k})$ # representation
    $\boldsymbol{z}_{2k} = g(\boldsymbol{h}_{2k})$ # projection
  end for
  for all $i \in \{1, \ldots, 2N\}$ and $j \in \{1, \ldots, 2N\}$ **do**
    $s_{i,j} = \boldsymbol{z}_i^\top \boldsymbol{z}_j / (\|\boldsymbol{z}_i\| \|\boldsymbol{z}_j\|)$ # pairwise similarity
  end for
  define $\ell(i, j)$ **as** $\ell(i, j) = -\log \frac{\exp(s_{i,j}/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(s_{i,k}/\tau)}$
  $\mathcal{L} = \frac{1}{2N} \sum_{k=1}^{N} [\ell(2k-1, 2k) + \ell(2k, 2k-1)]$
  update networks f and g to minimize $\mathcal{L}$
end for
return encoder network $f(\cdot)$, and throw away $g(\cdot)$

Рисунок 2.3 – Алгоритм роботи SimCLR

Ціль наступного етапу - це навчання, яке спрямовано на ефективне використання розмічених навчальних даних для підвищення компактності розподілу класів і збільшення буферних зон між класами. Враховуючи гіперсферичність (радіальний базис) вирішувальних правил в рамках інформаційно-екстремальної технології, використовується контрастно центрованої функції втрат (contrastive-center loss). В даній функції використовується множина векторів-прототипів $\{\overline{z}_j \mid j = \overline{1, M}\}$, кожен з яких характеризує центр розподілу відповідного класу розпізнавання. Ці вектори прототипи не є "замороженими" і оптимізуються в процесі машинного навчання. Контрастно-центрована функція втрат обчислюється за формулою [14]

$$L_{contr\_center} = -\frac{1}{n} \sum_{i=1}^{n} \left(\frac{\left\| f(x_i) - \overline{z}_{y_i} \right\|_2^2}{\sum_{j=1, j \neq y_i}^{k} \left\| f(x_i) - \overline{z}_j \right\|_2^2 + 1} \right), \tag{2.4}$$

де $f(x)$ – функція екстрактора ознак, що встановлює залежність між вхідним зображенням та вектором ознак на виході сигмоїдного шару;

y_i – номер класу для x_i вхідного зображення;

$\overline{z}_{y_i}$ – вектор-прототип y_i класу, що шукається в процесі машинного навчання;

$\overline{z}_j$ – вектор-прототип j-го класу, що не співпадає з класом y_i;

n – кількість зразків в пакеті;

$\|\cdot\|_2^2$ – квадрат відстані Евкліда.

Як допоміжну можна розглядати кросентропійну функцію втрат, що обчислюється між мітками класів та ймовірністю належності до класів, яка оцінюється на основі розподілу відстаней Евкліда до прототипів класів

$$L_{dist\_ce} = -\frac{1}{n}\sum_{i=1}^{n} CrossEntropy\left(\frac{\exp\left(-\left\|f(x_i)-\overline{z}_{y_i}\right\|_2^2\right)}{\sum_{k=1}^{K}\exp\left(-\left\|f(x_i)-\overline{z}_k\right\|_2^2\right)}, onehot\_coding(y_i)\right),$$

де K – розмір алфавіту класів.

Для підсилення дискретизації ознак згідно принципц інформаційного пляшкового горла можна використати регуляризаційну складову шо штрафуватиме за помилку дискретизації. Пропонована регуляризуюча компонента функції втрат характеризується Ліпшицевою неперервністю (Lipschitz Continuous) і обчислюється за формулою

$$\mathrm{R_{bin}} = \frac{1}{\mathrm{n}}\sum_{\mathrm{i=1}}^{\mathrm{n}} \lambda \mathrm{f(x_i)^T(e - f(x_i)))}, \qquad (2.5)$$

e – вектор з одиниць;

λ – коефіцієнт регуляризації ($\lambda = 0{,}01$).

Для кодування кожного класу двійковим вектором-прототипом $\{b_{m,i} \mid m = \overline{1,M}, i = \overline{1,N}\}$ пропонується здійснити округлення оптимізованих на попередньому етапі навчання векторів-прототипів $\{\bar{z}_{m,i} \mid m = \overline{1,M}, i = \overline{1,N}\}$

$$b_{m,i} = \begin{cases} 1, & \text{if } \bar{z}_{m,i} > 0,5; \\ 0, & \text{otherwise.} \end{cases} \qquad (2.6)$$

В традиційних моделях аналізу даних під час побудови вирішувальних правил здійснюють калібрування порогів спрацювання з метою врахування незбалансованості класів та ознак на різних рівнях абстактності. В пропонованому методі замість цього здійснюється оптимізація за інформаційним критерієм кодових радіусів контейнерів для кожного з класів розпізнавання з метою врахування габаритиру розподілу і перетину класів під час прийняття рішень. Оптимізаційна процедура має вигляд [16]

$$E_m^* = \max_{\{d\}} E_m(d), \qquad (2.7)$$

де $\{d\} = \{0,1,...,\left(\sum_i b_{m,i} \oplus b_{c,i} - 1 \right)\}$ – набір концентричних кодових радіусів з центром в b_m;

$\sum_i b_{m,i} \oplus b_{c,i}$ – кодова відстань між вектором-прототипом m-го класу та сусіднім до нього c-го класу, $m \neq c$;

E_m – інформаційний критерій ефективності вирішувальних правил для m-го класу [12].

2.2 Критерій оптимізації вирішувальних правил

Оптимізація кодового радіусу контейнера кожного класу здійснюється прямим

послідовним перебором значень і вибору того значення, що забезпечує максимум інформаційного критерію. Як інформаційний критерій розглядається нормалізована модифікація інформаційної міри Шенона для двохальтернативних рішень, що виражений через точністні характеристики рішень [16] :

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

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

$$\left. + \frac{D_{2,m}^{(k)}(d)}{\alpha_m^{(k)}(d) + D_{2,m}^{(k)}(d)} \log_2 \frac{D_{2,m}^{(k)}(d)}{\alpha_m^{(k)}(d) + D_{2,m}^{(k)}(d)} \right), \qquad (2.8)$$

де $\alpha_m^{(k)}(d)$ – оцінка ймовірності помилок першого роду (false postive rate) на k-му кроці оптимізації;

$\beta_m^{(k)}(\mathrm{d})$ – оцінка ймовірності 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.

Точністні характеристики радіально-базисних класифікаційних правил для кожного класу можут бути обчислені на основі статистичних тестів за формулами

$$\mathrm{D_{1,m}} = \text{Чутливість}_\mathrm{m} = \frac{\mathrm{TP_m}}{\mathrm{TP_m} + \mathrm{FN_m}}, \qquad (2.9)$$

$$\mathrm{D_{2,m}} = \text{Специфічність}_\mathrm{m} = \frac{\mathrm{TN_m}}{\mathrm{TN_m} + \mathrm{FP_m}}, \qquad (2.10)$$

$$\alpha_m = \text{Оцінка ймовірності пропусків}_m = \frac{FN_m}{FN_m + TP_m}, \quad (2.11)$$

$$\beta_m = \text{Оцінка ймовірності хибних спрацювань}_m = \frac{FP_m}{FP_m + TN_m}, \quad (2.12)$$

де TP_m – кількість правильно-позитивних класифікацій зразків для вирішувальних правил m-го класу;

TN_m – кількість правильно-негативних класифікацій зразків для вирішувальних правил m-го класу;

FP_m – кількість хибно-позитивних класифікацій зразків для вирішувальних правил m-го класу;

FN_m – кількість хибно-негативних класифікацій зразків для вирішувальних правил m-го класу.

Для валідації ефективності багатокласового класифікатора можна використати мікро- або макро- усереднені значення значення інформаційного критерію, але для порівняння з результатами інших дослідників варто використовувати традиційні метрики. Для випадку незбалансованості класів чи ознакового опису широкого поширення набуло використання мікро-усередненого значення F1-міри (1.5).

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ РОЗПІЗНАВАННЯ ДЕФЕКТІВ СТІЧНИХ ТРУБ

3.1 Формування вхідного математичного опису

Набір даних SEWER-ML було використано для підготовки вхідного математичного опису інформаційної інтелектуальної системи аналізу відео зображень [6]. В наборі 1,3 мільйони зображень з 75618 відеофайлів, зібраних з датських компаній, що обслуговують сточні трубопроводи. Було виділено 13 класів розпізнавання, 12 з яких показано на рис. 3.1. В більшості стандартів протокол інспекції передбачає, що для детальнішого огляду дефекту оператор повертає камеру і оцінює його тяжкість, тому розглянемо задачу розпізнавання дефектів при повороті камери на стінку труби. Для задачі попереднього навчання без вчителя планується використати 20% датасету. А для тюнінгу під задачу розпізнавання дефектів планується обрати тільки ті зображення, що містять дефекти – вид збоку. Зображення необхідно превести до роздільною здатності 224x224 пікселя з метою зменшення витрат ресурсів на екзамен та навчання. Алфавіт сформованих класів показано в табл. 3.1. Приклади зображень кожного класу показано на рис. 3.1.

Таблиця 3.1 Структура розміченої вибірки даних для розпізнавання дефектів на стінках сточних труб

Клас	Назва дефекту	Кількість розмічених зразків
x_1^0	Порушення цілісності прокладки на стику	250
x_2^0	Відкритий стик	200
x_3^0	Зміщений стик	331
x_4^0	Тріщина	270
x_5^0	Розлом	189

Продовження таблиці 3.1

Клас	Назва дефекту	Кількість розмічених зразків
x_6^0	Розбита труба	306
x_7^0	Дірка в трубі	450
x_8^0	Пошкодження поверхні	210
x_9^0	Прикріплені відкладення	200
x_{10}^0	Осад	300
x_{11}^0	Тоненькі корені	500
x_{12}^0	Значне розростання коренів	310
x_{13}^0	Нормальний стан труби	500

Поділ вибірки на тестову і навчальну здійснюється у відношенні 70% на навчання і 30% на тестування. При цьому під час поділу використовується стратегія збереження пропорції кількості зразків кожного класу в тестовій і навчальній вибірці. Потім навчальна частина набору даних створена шляхом аугментації міноритарних класів. Використані методи аугментації:

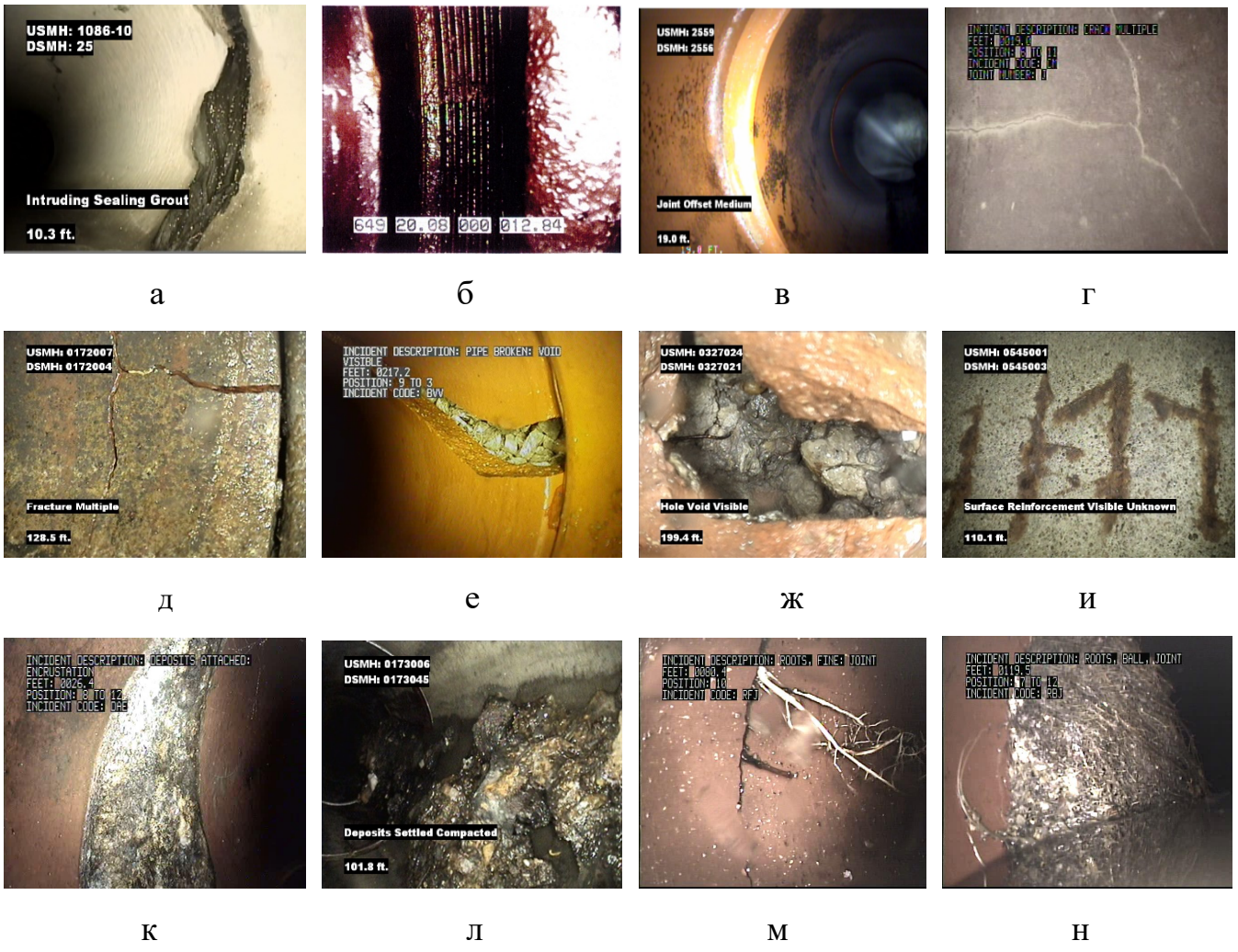
- зміна масштабу на 1…5 %;
- поворот зображення на ±18 °;
- зміна яскравості на ±5 %.

Розмір аугментованої навчальної вибірки кожного класу для класифікатора дефектів становить 500 зразків.

Процес навчання без вчителя пропонується продовжувати доки функція втрат не перестане помітно зменшуватися. Для вимірювання ефективності моделей під час навчання з учителем використовується критерій, що обчислюється на екземаційній вибірці. При цьому після оброки кожного міні-пакету пропонується заново будувати інформаційно-екстремальні вирішувальні правила і тестувати їх на тестовій вибірці.

З вхідного набору даних SEWER-ML було сформовано вибірку зображень дефектів на стінках стічних труб (рис. 2.3) [17]: X_1^o – порушення цілісності

прокладки на стику; X_2^o – відкритий стик; X_3^o – зміщений стик; X_4^o – тріщина; X_5^o – розлом; X_6^o – розбита труба; X_7^o – дірка в трубі; X_8^o – пошкодження поверхні; X_9^o – прикріплені відкладення; X_{10}^o – осад; X_{11}^o – тоненькі корені; X_{12}^o – значне розростання коренів; X_{13}^o – нормальний стан труби.



а б в г

д е ж и

к л м н

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; д – клас X_5^o; е – клас X_6^o; ж – клас X_7^o; и – клас X_8^o; к – клас X_9^o; л – клас X_{10}^o; м – клас X_{11}^o; н – клас X_{12}^o

Вибірка перелічених дефектів містить незбалансованість класів, тобто обсяги вибірки кожного класу становлять наступні відповідні значення : 250, 200, 331, 270, 189, 306, 450, 210, 200, 300, 500, 310, 500.

3.2 Короткий опис програмної реалізації

На вхід системи розпізнавання дефектів стічних труб для автоматичного формування звітів необхідно подавати дані від одометра для визначення відстані кожного з дефектів та відеофайл для кожної інспекції. Інформаційна система, що пропонується, здійснює аналіз інспеції в декілька етапів, послідовність яких показана на рис. 3.2..

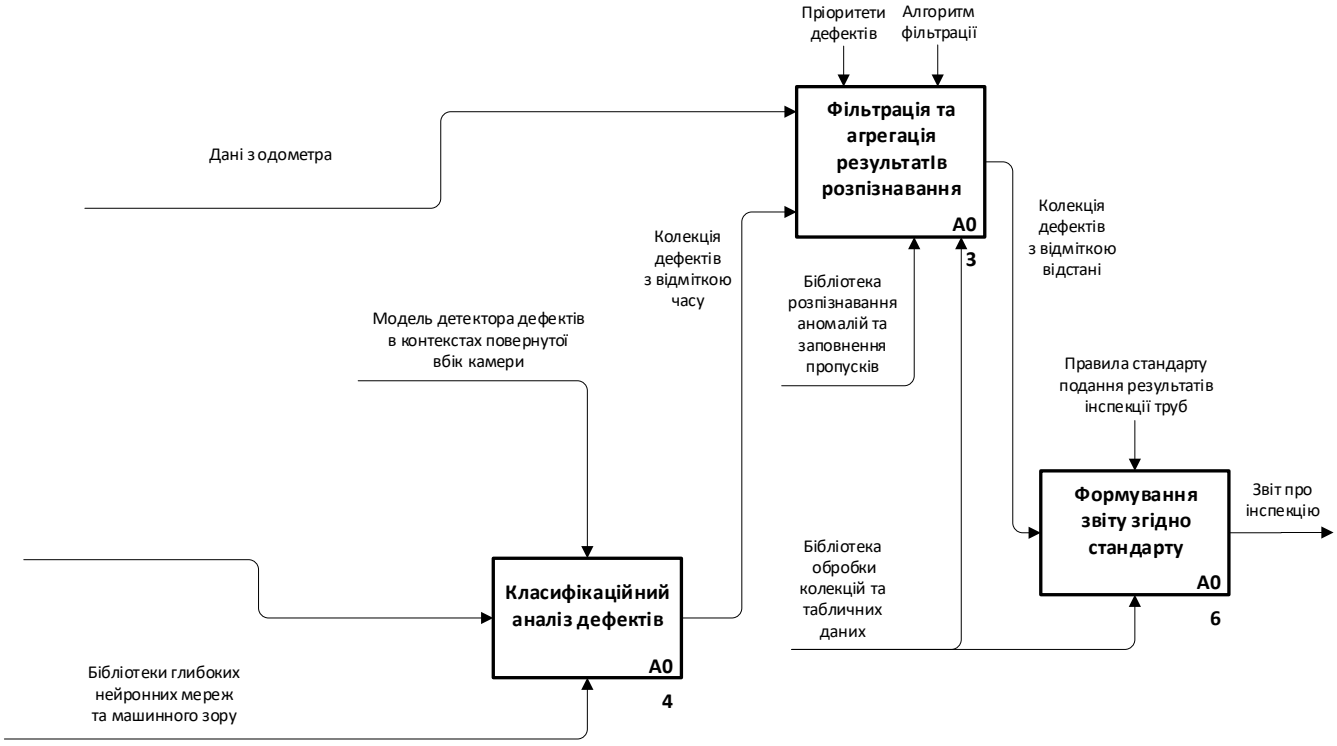


Рисунок 3.2– Функціональна схема інформаційної технології класифікаційного аналізу стічних труб

Аналіз рис. 3.2 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]. MobileNetV2 — це архітектура згорткової нейронної мережі. MobileNetV2 заснований на перевернутій залишковій структурі, де залишкові зв'язки знаходяться між вузькими шарами. Проміжний шар розширення використовує легкі глибинні згортки для фільтрації об'єктів як джерела нелінійності. В цілому архітектура MobileNetV2 містить початковий шар з повною згорткою з 32 фільтрами, за яким слідують 19 залишкових шарів вузького місця.

Jupyter Notebook та Google colab використовувалось як середеовище для розробки. Jupyter Notebook – це інтерактивний інструмент для створення добре оформлених аналітичних звітів, так як він дозволяє зберігати разом зображення, код, коментарі, формули і графіки. Гнучкий інтерфейс Jupyter Notebook дозволяє користувачам налаштовувати та організовувати робочі процеси в галузі науки про дані, наукових обчислень та машинного навчання. Google Colab – це безкоштовний інтерактивне хмарне середовище для роботи з кодом від Google. Google Colab надає все необхідне для машинного навчання прямо в браузері, дає безкоштовний доступ до швидких GPU та TPU. Головна особливість «Колабораторії» — безкоштовні потужні графічні процесори GPU та TPU, завдяки яким можна займатися не лише базовою аналітикою даних, а й складнішими дослідженнями у галузі машинного навчання. З тим, що CPU обчислює годинами, GPU або TPU справляються за хвилини або секунди. Розробка здійснювалась на мові програмування python 3. Розроблене програмне забезпечення потребує використання сторонніх бібліотек з використанням менеджеру pip, опис яких наведено в табл. 3.1.

Для зчитування системою вхідних даних використуються бібліотека opencv і обробляються в форматі NumPy масиву, однак безпосередній аналіз даних вібдувається в тензорних масивах в нейронній мережі бібліотеки tensorflow. Для простоти написання коду пропонується використовувати високорівневу обготку keras. Бібліотека Scikit-Learn може використовується для препроцесінгу отриманих даних [18].

У Додатку А присутній код функції створення глибокої моделі екстракції ознакового опису під назвою create_embedding_model. Ця функція приймає на вхід кількість ознак, яка на виході має сформувати модель, та роздільну здатність вхідного зображення. Будується базова модель для порівняльного аналізу з повнозвязним традиційним класифікаційним softmax вихідним шаром за допомогою функції create_baseline_model [19, 20].

Таблиця 3.2 – Використані бібліотеки

Назва бібліотеки	Опис
NumPy	Бібліотека мови Python[20], що дає можливість виконувати: • математичні та логічні операції над масивами; • перетворення Фур'є та процедури; • операції лінійної алгебри; • математичні та логічні операції над матрицями; • вбудовані функції для лінійної алгебри та генерації випадкових чисел.

Продовження таблиці 3.2

Назва бібліотеки	Опис
Scikit-Learn	Безкоштовна бібліотека алгоритмів[21] машинного навчання, написана на Python і побудована на основі модуля SciPy. SciPy включає інструменти для ефективного аналізу даних. Модуль scikit-learn надає розробникам безліч алгоритмів у сфері контрольованого та неконтрольованого машинного навчання у вигляді узгодженого інтерфейсу програмування. Крім того, модуль доступний за ліцензією BSD, що дозволяє його комерційне та академічне використання. Scikit-Learn включає ряд пакетів: • класифікація (classification); • регресія (regression); • кластер-аналіз (clustering); • редукція розмірності (dimensionality reduction); • валідація моделей (model selection); • передобробка (preprocessing); • авантаження набору даних (data loader); • візуалізація проміжних і кінцевих результатів (flow chart); • прикладів використання. Бібліотека має якісну документацію та інструкції швидкого старту.

Продовження таблиці 3.2

Назва бібліотеки	Опис
Keras	Keras — бібліотека мови Python з відкритим кодом для створення нейронних мереж[22]. Keras використовує TensorFlow або Theano як бекенд, де виконуються операції. Keras спрощує роботу з цими двома бібліотекам, які виконують алгоритми машиного навчання і повертають результат. Keras допомагає створювати невеликі програми Machine Learning (або швидке прототипування та експерименти), оскільки бібліотека спрощує багато кроків та заощаджує багато часу на написання коду. Keras використовується CERN, NASA, NIH та багатьма іншими науковими організаціями по всьому світу (так, Keras використовується на LHC). Keras має низький порог входу: що дозволяє для реалізувати довільні ідеї дослідження, пропонуючи додаткові високорівневі функції для прискорення циклів експериментів.
Opencv	Opencv — бібліотека комп'ютерного зору та машинного навчання з відкритим вихідним кодом. До неї входять понад 2500 алгоритмів, у яких є як класичні, і сучасні алгоритми для комп'ютерного зору машинного навчання. Ця бібліотека має реалізації на різних мовах програмування: Java, C++ та Matlab. Opencv може вільно використовуватись в комерційних та академічних цілях на умовах BSD ліцензії [24].

Продовження таблиці 3.2

Назва бібліотеки	Опис
Albumentations	Бібліотека для швидкої аугментації зображень, яка характеризується простотою використання і є обгорткою інших фреймворків. Це бібліотека з відкритим кодом, інтуїтивно зрозуміла ,швидка і добре документована. Бібліотека може одночасно доповнювати зображення та його маску сегментації, що обмежує рамку або розташування ключових точок. Albumentations придадтна для аугментації зразків у задачах класифікації, сегментації та детектування. Бібліотека легко кастомізується і додається до інших фреймворків.
Tensorflow	Tensorflow — потужна бібліотека Machine Learning від Google, що відрізняється високою продуктивністю та масштабованістю. Tensorflow має вбудовану підтримку для роботи з пристроями GPU. Крім того, існує також інструмент TensorBoard, який дозволяє нам відображати та аналізувати дані, що використовуються у процесі навчання. Tensorflow підтримує мови програмування C++, Python, JavaScript, java. Tensorflow підтримує такі типи процесів як CPU, GPU та TPU. Перевагою TensorFlow є можливість створювати абстракцій для розробки машинного навчання. Замість того, щоб мати справу з реалізаціями що дозволяє розробникам зосередитися на загальній логіці програми[23].

Програмний код наведено в додатку А. Призначення основних класів приведено в табл. 3.3.

Таблиця 3.3 – Основні класи програмного забезпечення

Клас	Короткий опис
ContrastiveModel	Клас, екземпляри якого забезпечують контрастне навчання з самоучителем (self-supervision), для попередньої ініціалізації параметрів екстрактора ознак
BinarizedPrototypeLoss	Клас, екземпляри якого забезпечують обчислення функції втрат та еталонних векторів (прототипів класів) під час навчання екстрактора ознак з учителем
InformationExtremeClassifier	Клас, екземпляри якого забезпечують оптимізацію радіусів для побудови радіально-базисних інформаційно-екстремальних класифікаційних вирішувальних правил. Одночасно даний екземпляр може уточнити і еталонні вектори класів.

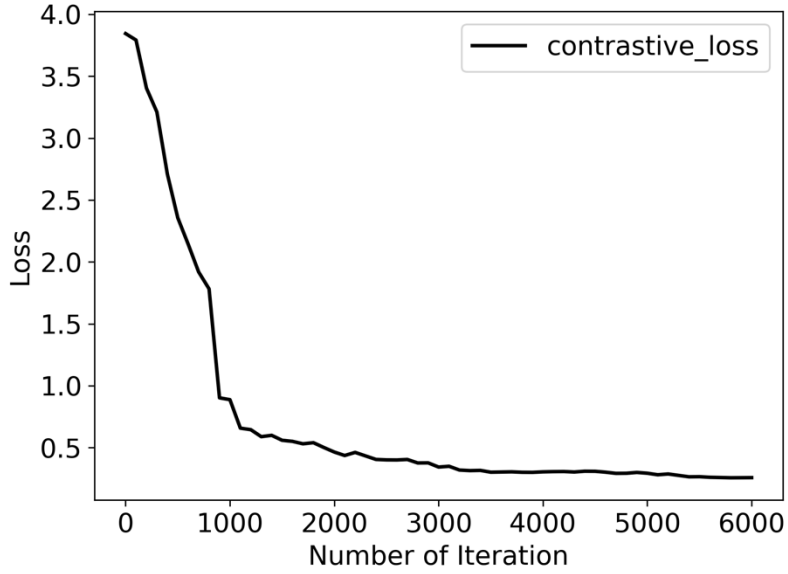
Функції gaborN_rand, gaborN_uni, perlin та perturb є основою для реалізації шуму, що використовується під час аугментації з метою підвищення узагальнюючої здатності моделі класифікаційного аналізу.

Таким чином, програмна реалізація дозволяє тестувати алгоритми розроблені з метою реалізації в майбутньої повнофункціональної системи для розпізнавання дефектів в стічних трубах.

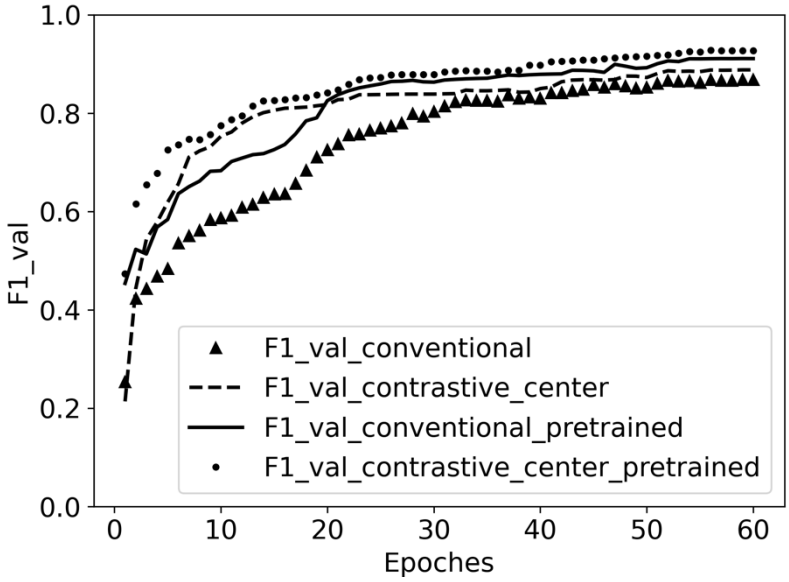
3.3 Результати машинного навчання

Перший етап машинного навчання реалізується з використанням попереднього самонавчання екстратора ознак з функціює втрат (2.3). На рис. 3.3*а* показано графік зміни функції втрат (2.3) (Loss) на тестовій вибірці від кількості ітерацій (Number of

iterations). На рис. 3.3*б* показано результати машинного навчання з і без попереднього самонавчання для традиційного і запропонованого методів навчання.



а



б

Рисунок 3.3 –Залежність критерію ефективності навчання від кількості ітерацій : а – самонавчання екстрактора ознак; б – навчання на розміченій вибірці класифікаційної моделі за традиційним і запропонованим методом з і без попереднього самонавчання

Аналіз рис. 3.3а показує, що після 5500 ітерацій зменшення функції втрат практично зупилося під час самонавчання екстрактора ознак, тому далі процес не продовжувався. Аналіз рис. 3.3б показує, що без попереднього самонавчання традиційний підхід до навчання забезпечує F1=0,868, а після самонавчання вдається досягти F1=0,911. В кінці кожної епохи вібувається синтез інформаційно-екстемальних вирішувальних правил за процедурами (2.6) та (2.8) і здійснюється валідація на тестовій вибірці, що становить 20% від повного обсягу розмічених даних. Як видно з рис. 3.3б навчання за запропонованим методом без попереднього самонавчання забезпечує F1 = 0,8881, що на 2% перевершує традиційний підхід. При цьому використання попереднього самонавчання забазпечує F1 = 0,930, що також приблизно на 2% перевищує результат, отриманий в рамках традиційного підходу.

Таким чином, отримане значення F1-метрики на тестових даних в рамках запропонованого підходу забезпечує прийнятний для практичного використання рівень точності. При цьому значення F1-метрики за запропонованими моделлю і методом навчання з попереднім самонавчання перевищує значення F1-метрики, отриманої в рамках традиційного підходу, на 7%.

ВИСНОВКИ

1. **Наукова новизна отриманих результатів** :

– вперше розроблено метод машинного навчання глибокої мережі, що полягає в ініціалізації вагових коефіцієнтів на нерозмічених даних з використанням самонавчання та уточненні вагових коефіцієнтів з використанням контрастно-центрованої функції втрат з регуляризуючою складовою для бінаризації ознакового опису і побудови радіально-базисних інформаційно-екстремальних вирішувальних правил;

– експериментально підтверджено перевагу запропонованого методу машинного навчання порівняно з традиційним підходом до глибинного навчання моделі класифікатора зображень;

– експериментально підтверджено, що використання попереднього контрастного самонавчання покращує наступні результати як традиційного, так запропонованого методів навчання.

2. Практичне значення роботи полягає в підвищенні точності класифікаційних рішень для моделі розпізнавання дефектів стічних труб, що навчається за незбалансованими даними обмеженого обсягу і значною шумовою складовою.

Подальші дослідження будуть спрямовані на поширення основних ідей запропонованого методу для навчання моделей детектування, локалізації та оцінювання розміру дефектів стічних труб.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Cheng, J. C. P. Automated detection of sewer pipe defects in closed-circuit television images using deep learning techniques [Text] / J. C. P. Cheng, M. Wang // Automation in Construction. – 2018. – Vol. 95. – P. 155-171. – DOI : 10.1016/j.autcon.2018.08.006.

2. Moradi, S. Review on Computer Aided Sewer Pipeline Defect Detection and Condition Assessment [Text] / S. Moradi, T. Zayed, F. Golkhoo // Infrastructures. – 2019. – Vol 4, No. 1. – Article Id: 10. – DOI : 10.3390/infrastructures4010010.

3. Haurum, J. B. A Survey on image-based automation of CCTV and SSET sewer inspections [Text] / J. B. Haurum, T. B. Moeslund // Automation in Construction. – 2020. – Vol. 111. – Article Id: 103061. DOI : 10.1016/j.autcon.2019.103061.

4. Czimmermann, T. Visual-Based Defect Detection and Classification Approaches for Industrial Applications—A SURVEY [Text] / T. Czimmermann, G. Ciuti, M. Milazzo, M. Chiurazzi, S. Roccella, C. M. Oddo, P. Dario // Sensors. – 2020. – Vol. 20. – Article Id : 1459. – DOI: 10.3390/s20051459

5. Dawei Li. Sewer pipe defect detection via deep learning with local and global feature fusion [Text] / Dawei Li, Qian Xie, Zhenghao Yu, Qiaoyun Wu, Jun Zhou, Jun Wang // Automation in Construction. – 2021. – Vol. 129. – Article Id: 103823. – DOI : 10.1016/j.autcon.2021.103823.

6. Duanshun Lia. Sewer damage detection from imbalanced CCTV inspection data using deep convolutional neural networks with hierarchical classification [Text] / D. Li, A. Cong, S. Guo // Automation in Construction. – 2019. – Vol. 101. – P. 199-208. – DOI : 10.1016/j.autcon.2019.01.017.

7. Liu X. Self-supervised Learning: Generative or Contrastive [Text] / X. Liu, F. Zhang, Z. Hou, L. Mian, Z. Wang, J. Zhang, J. Tang // IEEE Transaction on Knowledge and Data Engineering. – 2021. – DOI : 10.1109/TKDE.2021.3090866

8. Baradad M. Learning to See by Looking at Noise / M. Baradad, J. Wulff, T. Wang, P. Isola, A. Torralba // 35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia. – 2021. – 14 p. – arXiv preprint arXiv:2106.05963.

9. Kolchinsky A. Nonlinear Information Bottleneck [Text] / A. Kolchinsky, B. D. Tracey, D. H. Wolpert // Entropy. – 2019. – Val 21 (12). – Article Id: 1181. DOI: 10.3390/e21121181.

10. Challenging the Adversarial Robustness of DNNs Based on Error-Correcting Output Codes [Text] / B. Zhang, B. Tondi, X. Lv, M. Barni // Security and Communication Networks. – 2020. – Vol. 2020. – Article Id: 8882494. DOI : 10.1155/2020/8882494.

11. Raviv R. CodNN – Robust Neural Networks From Coded Classification / N. Raviv, S. Jain, P. Upadhyaya, J. Bruck, A. A. Jiang // 2020 IEEE International Symposium on Information Theory (ISIT). – 2020. – P. 2688-2693. – DOI : 10.1109/ISIT44484.2020.9174480

12. Li J. Towards a high robust neural network via feature matching / J. Li, Y. Guo, S. Lao, Y. Wu, L. Bai, Y. Wei // International Journal of Multimedia Information Retrieval. – 2021. – V. 10. – P. 227–237. – DOI : 10.1007/s13735-021-00219-0

13. Konkle T. Beyond category-supervision: Computational support for domain-general pressures guiding human visual system representation / T. Konkle, Alvarez G. A. // bioRxiv. – 2021. – 12 p. – DOI: 10.1101/2020.06.15.153247.

14. Qi C. Contrastive-center loss for deep neural networks / C. Qi, F. Su // IEEE International Conference on Image Processing (ICIP). – 2017. – P. 2851-2855. – DOI : 10.1109/icip.2017.8296803.

15. Haurum J. B. Sewer-ML: A Multi-Label Sewer Defect Classification Dataset and Benchmark [Text] / J. B. Haurum, T. B. Moeslund // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021, pp. 13456-13467.

16. 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. – 2020. – Vol. 1158. – pp 253-263. DOI : 10.1007/978-3-030-61656-4_16.

17. Kumar, S. S. Deep Learning–Based Automated Detection of Sewer Defects in CCTV Videos [Text] / S. S. Kumar, W. Mingzhu, D. M. Abraham, M. R. Jahanshahi,

I. Tom, J. C. Cheng // Journal of Computing in Civil Engineering. – 2020. – Vol. 34(1). – Article Id : 4019047.

18. Кудрявцев А. М. Метод самонавчання згорткового екстрактора ознак для розпізнавання багатоканальних діагностичних зображень / А. М. Кудрявцев, А. С. Москаленко, В. В. Москаленко // XIV International scientific conference «Intellectual systems of decision-making and problems of computational intelligence (ISDMCI'2018)»

19. Інтелектуальна автономна бортова система безпілотного літального апарату для ідентифікації об'єктів на місцевості [Текст] : Виготовлення дослідного зразка безпілотного апарату та впровадження результатів дослідження: звіт про НДР (остаточний) / кер. В. В. Москаленко. — Суми : СумДУ, 2020. — 79 с.

20. Python Numpy [Електронний ресурс] – Режим доступу до ресурсу: www.brutalk.com/ru/brutalk-blog/view/co-to-jest-numpy-w-pythonie---python-numpy-tutorial-60469f51d7e28.

21. Scikit Learn [Електронний ресурс] – Режим доступу до ресурсу: https://aswed.gitbooks.io/python/content/chapter1/biblioteka-scikit-learn.html.

22. Машинне навчання - корисні інструменти та бібліотеки [Електронний ресурс] – Режим доступу до ресурсу: https://nofluffjobs.com/blog/machine-learning-przydatne-narzedzia-i-biblioteki/.

23. What is TensorFlow? The machine learning library explained [Електронний ресурс] – Режим доступу до ресурсу: https://www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning-library-explained.html.

24. Шпаргалка по OpenCV — Python [Електронний ресурс] – Режим доступу до ресурсу: https://tproger.ru/translations/opencv-python-guide/.

ДОДАТОК А

```
from google.colab import drive
drive.mount('/content/gdrive/')
```

```
Mounted at /content/gdrive/
```

```
!mkdir train
!mkdir train/unlabeled
!cp /content/gdrive/MyDrive/Pipe_Hack_Data/train00.zip train00.zip
!unzip /content/train00.zip -d /content/train/unlabeled/
```

Dataset

During training we will simultaneously load a large batch of unlabeled images along with a smaller batch of labeled images.

```
builder = tfds.ImageFolder(INPUT_DIR+"/")
print(builder.info)

unlabeled_train_dataset = builder.as_dataset(shuffle_files=True, as_supervised=True, split="train")

tfds.show_examples(unlabeled_train_dataset, builder.info)
```

Python

```
tfds.core.DatasetInfo(
```

Encoder architecture

```
#*************************************************************************************
#************************  base_feature_extractor.py  ********************************
#*************************************************************************************

# Define the encoder architecture
def get_encoder():
    extractor = tf.keras.applications.MobileNet(
            input_shape=(image_size, image_size, 3),
            alpha=0.25,
            include_top=False,
            weights='imagenet' )
    inputs = extractor.output
    x = tf.keras.layers.GlobalAveragePooling2D()(inputs)
    outputs = tf.keras.layers.Dense(width, activation="relu", use_bias=False)(x)
    return tf.keras.Model(extractor.input, outputs, name="encoder")
```

```python
# Distorts the color distibutions of images
class RandomColorAffine(layers.Layer):
    def __init__(self, brightness=0, jitter=0, **kwargs):
        super().__init__(**kwargs)

        self.brightness = brightness
        self.jitter = jitter

    def call(self, images, training=True):
        if training:
            batch_size = tf.shape(images)[0]

            # Same for all colors
            brightness_scales = 1 + tf.random.uniform(
                (batch_size, 1, 1, 1), minval=-self.brightness, maxval=self.brightness
            )
            # Different for all colors
            jitter_matrices = tf.random.uniform(
                (batch_size, 1, 3, 3), minval=-self.jitter, maxval=self.jitter
            )

            color_transforms = (
                tf.eye(3, batch_shape=[batch_size, 1]) * brightness_scales
                + jitter_matrices
            )
            images = tf.clip_by_value(tf.matmul(images, color_transforms), 0, 1)
        return images
```

```python
# Image augmentation module
def get_augmenter(min_area, brightness, jitter):
    zoom_factor = 1.0 - tf.sqrt(min_area)
    return keras.Sequential(
        [
            keras.Input(shape=(image_size, image_size, image_channels)),
            preprocessing.Rescaling(1./127.5, offset=-1.0),   #1 / 255),
            #tf.keras.layers.Rescaling(1./127.5, offset=-1.0),
            preprocessing.RandomFlip("horizontal"),
            preprocessing.RandomTranslation(zoom_factor / 2, zoom_factor / 2),
            preprocessing.RandomZoom((-zoom_factor, 0.0), (-zoom_factor, 0.0)),
            RandomColorAffine(brightness, jitter),
        ]
    )
```

```
class ContrastiveModel(keras.Model):
    def __init__(self):
        super().__init__()
        self.temperature = temperature
        self.contrastive_augmenter = get_augmenter(**contrastive_augmentation)
        self.encoder = get_encoder()
        # Non-linear MLP as projection head
        self.projection_head = keras.Sequential(
            [
                keras.Input(shape=(width,)),
                layers.Dense(width, activation="relu"),
                layers.Dense(width),
            ],
            name="projection_head",
        )
        self.encoder.summary()
        self.projection_head.summary()

    def compile(self, contrastive_optimizer, **kwargs): #, probe_optimizer, **kwargs):
        super().compile(**kwargs)

        self.contrastive_optimizer = contrastive_optimizer
        self.contrastive_loss_tracker = keras.metrics.Mean(name="c_loss")
        self.contrastive_accuracy = keras.metrics.SparseCategoricalAccuracy(
            name="c_acc"
        )
```

```
    @property
    def metrics(self):
        return [
            self.contrastive_loss_tracker,
            self.contrastive_accuracy,
        ]

    def contrastive_loss(self, projections_1, projections_2):
        # InfoNCE loss (information noise-contrastive estimation)
        # NT-Xent loss (normalized temperature-scaled cross entropy)
        # Cosine similarity: the dot product of the l2-normalized feature vectors
        projections_1 = tf.math.l2_normalize(projections_1, axis=1)
        projections_2 = tf.math.l2_normalize(projections_2, axis=1)
        similarities = (
            tf.matmul(projections_1, projections_2, transpose_b=True) / self.temperature
        )
        # The similarity between the representations of two augmented views of the
        # same image should be higher than their similarity with other views
        batch_size = tf.shape(projections_1)[0]
        contrastive_labels = tf.range(batch_size)
        self.contrastive_accuracy.update_state(contrastive_labels, similarities)
        self.contrastive_accuracy.update_state(
            contrastive_labels, tf.transpose(similarities)
        )
        # The temperature-scaled similarities are used as logits for cross-entropy
        # a symmetrized version of the loss is used here
        loss_1_2 = keras.losses.sparse_categorical_crossentropy(
            contrastive_labels, similarities, from_logits=True
        )
        loss_2_1 = keras.losses.sparse_categorical_crossentropy(
            contrastive_labels, tf.transpose(similarities), from_logits=True
        )
        return (loss_1_2 + loss_2_1) / 2
```

```
def train_step(self, unlabeled_images):
    # Each image is augmented twice, differently
    augmented_images_1 = self.contrastive_augmenter(unlabeled_images)
    augmented_images_2 = self.contrastive_augmenter(unlabeled_images)
    with tf.GradientTape() as tape:
        features_1 = self.encoder(augmented_images_1)
        features_2 = self.encoder(augmented_images_2)
        # The representations are passed through a projection mlp
        projections_1 = self.projection_head(features_1)
        projections_2 = self.projection_head(features_2)
        contrastive_loss = self.contrastive_loss(projections_1, projections_2)
    gradients = tape.gradient(
        contrastive_loss,
        self.encoder.trainable_weights + self.projection_head.trainable_weights,
    )
    self.contrastive_optimizer.apply_gradients(
        zip(
            gradients,
            self.encoder.trainable_weights + self.projection_head.trainable_weights,
        )
    )
    self.contrastive_loss_tracker.update_state(contrastive_loss)
    return {m.name: m.result() for m in self.metrics}
```

```
# Contrastive pretraining
pretraining_model = ContrastiveModel()
pretraining_model.compile(
    contrastive_optimizer=keras.optimizers.Adam()
)
```

```
pretraining_history = pretraining_model.fit(
    unlabeled_train_dataset, epochs=num_epochs
)
```

load labeled data

```
!cp '/content/gdrive/MyDrive/context_dataset_originals_v1.0.zip' '/content/context_dataset_originals_v1.0.zip'
!unzip  '/content/context_dataset_originals_v1.0.zip' -d 'context_dataset_originals_v1.0'
```

```
import splitfolders

splitfolders.ratio('context_dataset_originals_v1.0/context_dataset_originals/',
                   output="output", ratio=(.8, 0.1,0.1))
```

```
from numpy import savetxt

# save to csv file
savetxt('gdrive/MyDrive/radiuses.csv', classifier.radiuses, delimiter=',')
```

```
from sklearn.metrics import f1_score

features = []
y_test = []
for dir in os.listdir('/content/output/test'):
    for filename in os.listdir('/content/output/test'+"/"+dir) :
        if filename.endswith(".jpg") :
            img = process_image('/content/output/test'+"/"+dir+"/"+filename)
            features.append(feature_extractor.predict(img)[0])
            y_test.append( name_to_int[dir] )

features = np.array(features)
y_test = np.array(y_test)

features[features>0.5] = 1
features[features<=0.5] = 0

y_pred = classifier.predict(features)

f1_score(y_test, y_pred, average='macro'), f1_score(y_test, y_pred, average='micro')
```

(variable) dir: str

```
def gaborN_rand(size, grid, num_kern, ksize, sigma, theta, lambd, xy_ratio = 1, sides = 1, seed = 0):
    np.random.seed(seed)
    # Gabor kernel
    if sides != 1: gabor_kern = gaborK(ksize, sigma, theta, lambd, xy_ratio, sides)
    else: gabor_kern = cv2.getGaborKernel((ksize, ksize), sigma, theta, lambd, xy_ratio, 0, ktype = cv2.CV_32F)
    # Sparse convolution noise
    sp_conv = np.zeros([size, size])
    dim = int(size / 2 // grid)
    noise = []
    for i in range(-dim, dim + 1):
        for j in range(-dim, dim + 1):
            x = i * grid + size / 2 - grid / 2
            y = j * grid + size / 2 - grid / 2
            for _ in range(num_kern):
                dx = np.random.randint(0, grid)
                dy = np.random.randint(0, grid)
                while not valid_position(size, x + dx, y + dy):
                    dx = np.random.randint(0, grid)
                    dy = np.random.randint(0, grid)
                weight = np.random.random() * 2 - 1
                sp_conv[int(x + dx)][int(y + dy)] = weight

    sp_conv = cv2.filter2D(sp_conv, -1, gabor_kern)
    return normalize(sp_conv)
```

```python
def gaborN_uni(size, grid, ksize, sigma, lambd, xy_ratio, thetas):
    sp_conv = np.zeros([size, size])
    temp_conv = np.zeros([size, size])
    dim = int(size / 2 // grid)

    for i in range(-dim, dim + 1):
        for j in range(-dim, dim + 1):
            x = i * grid + size // 2
            y = j * grid + size // 2
            temp_conv[x][y] = 1
            theta = thetas[(i + dim) * dim * 2 + (j + dim)]

            # Gabor kernel
            gabor_kern = cv2.getGaborKernel((ksize, ksize), sigma, theta, lambd, xy_ratio, 0, ktype = cv2.CV_32F)
            sp_conv += cv2.filter2D(temp_conv, -1, gabor_kern)
            temp_conv[x][y] = 0

    return normalize(sp_conv)
```

```python
def perlin(size, period, octave, freq_sine, lacunarity = 2):
    # Perlin noise
    noise = np.empty((size, size), dtype = np.float32)
    for x in range(size):
        for y in range(size):
            noise[x][y] = pnoise2(x / period, y / period, octaves = octave, lacunarity = lacunarity)
    # Sine function color map
    noise = normalize(noise)
    noise = np.sin(noise * freq_sine * np.pi)
    return normalize(noise)
```

```python
def perturb(img, noise, norm):
    noise = np.sign((noise - 0.5) * 2) * norm
    noise = np.clip(noise, np.maximum(-img, -norm), np.minimum(255 - img, norm))
    return (img + noise)
```

```
class InformationExtremeClassifier:

    def __init__(self, etalons=None, radiuses=None):
        self.etalons   = None
        self.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_train[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
        return self.etalons
```

```
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.etalo
            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 fit(self, X_train, y_train):
    self.n = len(X_train)
    if self.etalons is None :
        self.compute_etalons(X_train, y_train)
    self.compute_max_radiuses()
    self.compute_distance_matrix(X_train)
    self.optimize_radiuses(X_train, y_train)
    return self.etalons, self.radiuses

def predict(self, x_test):
    result = []
    for j in range(len(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[j], e)
            u[c] = 1 - dist/self.radiuses[c]
            #u = np.exp(u)
            #u /= np.sum(u)
        result.append( np.argmax(u) )
    result = np.array(result)
    return result
```

```python
class BinarizedPrototypeLoss(tf.keras.layers.Layer):

    def __init__(self, num_classes=1, dim_hidden=1, lambda_c=1.0, **kwargs):
        super().__init__(**kwargs)
        self.dim_hidden  = dim_hidden
        self.num_classes = num_classes
        self.lambda_c    = tf.constant(lambda_c, dtype=tf.float32)
        self.epsilon     = tf.constant(1e-6, dtype=tf.floa  float32: Any

    def build(self, input_shape):
        self.centers = self.add_weight(name='centers',
                                       shape=(self.num_classes, self.dim_hidden),
                                       initializer=tf.keras.initializers.RandomUniform(minval=0., maxval=1.),
                                       trainable=True)
        super().build(input_shape)

    def call(self, inputs, mask=None):
        embedding, labels   = inputs[0], inputs[1]
        labels_int          = tf.math.argmax(labels, axis=1)
        batch_size          = embedding.shape[0]
        if batch_size is None:
            batch_size = 1
        expanded_centers    = tf.expand_dims(self.centers, axis=0)
        expanded_centers    = tf.repeat( expanded_centers, repeats=[batch_size], axis=0 )
        expanded_embedding  = tf.expand_dims( embedding, axis=1 )
        expanded_embedding  = tf.repeat( expanded_embedding, repeats=[self.num_classes], axis=1 )
        distance_centers    = tf.math.squared_difference( expanded_embedding, expanded_centers )
        distance_centers    = tf.math.reduce_sum( distance_centers, axis=2 )
        # Contrastive Center
        distances_intra     = tf.gather( distance_centers, indices=tf.expand_dims(labels_int, axis=1), batch_dims=1 )
        distances_sum       = tf.math.reduce_sum( distance_centers, axis=1 )
        distances_sum       = tf.expand_dims( distances_sum, axis=0 )
        distances_sum       = tf.transpose( distances_sum )
        distance_inter      = distances_sum - distances_intra
        loss1               = distances_intra/( distance_inter+self.epsilon)#self.lambda_c )
```

```python
        distances_sum       = tf.transpose( distances_sum )
        distance_inter      = distances_sum - distances_intra
        loss1               = distances_intra/( distance_inter+self.epsilon)#self.lambda_c )
        # Cross Entropy over distance distribution
        loss2               = tf.compat.v1.nn.softmax_cross_entropy_with_logits_v2(labels, -1.0*distance_centers)
        loss2               = tf.expand_dims(loss2, axis=0)
        loss2               = tf.transpose(loss2)
        # Regularization for output discritization
        # *********** for features discritization ******************
        e                   = tf.ones([batch_size, self.dim_hidden] )
        dif                 = tf.linalg.matmul(embedding, e-embedding, transpose_b=True)
        loss3               = tf.linalg.tensor_diag_part(dif)
        loss3               = tf.expand_dims(loss3, axis=0)
        loss3               = tf.transpose(loss3)
        # *********** for prototypes discritization ******************
        ec                  = tf.ones([self.num_classes, self.dim_hidden] )
        difc                = tf.linalg.matmul(self.centers, ec-self.centers, transpose_b=True)
        loss3c              = tf.linalg.tensor_diag_part(difc)
        loss3c              = tf.math.reduce_sum( loss3c, axis=0 )
        # Final loss
        self.loss            = 100.0*loss1 + 10.0*loss2 + 0.01*loss3 + 0.001*loss3c
        return self.loss

    def compute_output_shape(self, input_shape):
        return tf.compat.v1.keras.backend.int_shape(self.loss)

### custom loss
def zero_loss(y_true, y_pred):
    return 0.5 * tf.math.reduce_sum(y_pred, axis=0)
```