МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ВИПУСКНА РОБОТА

на тему:

«Інформаційна система завадозахищеного розпізнавання контексту спостережень в задачі інспекції труб»

Перевірив **Довбиш А.С.**

Керівник роботи **Москаленко В.В.**

Студента групи КБ-71 **Меняк А.В.**

СУМИ 2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Затверджую_____

Зав. кафедрою Довбиш А.С.

"_____" _____2021 г.

ЗАВДАННЯ
до кваліфікаційної роботи бакалавра

Студента четвертого курсу, групи КБ-71 спеціальності "Кібербезпека" денної форми навчання Меняка Артема Володимировича.

Тема: "Інформаційна система завадозахищеного розпізнавання контексту спостережень в задачі інспекції труб"

Затверджена наказом по СумДУ

№_____ від _____2021 г.

Зміст пояснювальної записки 1) Аналіз проблеми діагностики стічних труб; 2) Інформаційна система розпізнавання контексту спостережень на зображеннях інспекції труб; 3) Реалізація інформаційної системи для діагностики стічних труб.

Дата видачі завдання "_____" _____ 2021 г.

Керівник випускної роботи _____ Москаленко В.В.

Завдання прийняв до виконання _____ Меняк А.В.

РЕФЕРАТ

Записка: 54 стор., 20 рис., 1 додаток, 16 джерел.

Об’єкт дослідження — тенденції та стан сучасних систем діагностики труб, моделі та методи машинного навчання для аналізу зображень, ідеї та методи машинного навчання з самоучителем, алгоритми оптимізації нейромереж, класифікатори, типи класифікаторів.

Мета роботи — розробка завадозахищеної інформаційної системи для розпізнавання контексту спостережень в задачі інспекції стічних труб.

Методи дослідження — нейронні мережі, навчання з самоучителем, фото-відеоінспекція труб.

Результати — в середовищі Google Colab розроблено модель машинного навчання, яка на основі двох вибірок (тестової та тренувальної) розпізнає дефекти та розташування камери відносно безпосередньо стічної труби.

НЕЙРОННІ МЕРЕЖІ, МАШИННЕ НАВЧАННЯ, НАВЧАННЯ З САМОУЧИТЕЛЕМ, GOOGLE COLAB, ВІДЕОІНСПЕКЦІЯ, СТІЧНІ ТРУБИ, РОЗПІЗНАВАННЯ ДЕФЕКТІВ, PYTHON, SIMCLR

ЗМІСТ

ВСТУП

Важливість підземної інфраструктури не можна ігнорувати в сучасному світі. Обов'язкові ресурси, необхідні для будь-якого суспільства, що включають в себе комунікації, воду, каналізаційні системи та електроенергію розташовані у величезному підземному лабіринті. Основне використання підземної каналізації полягає у передачі стічних вод із споруд на очисні споруди або відведені місця захоронення. Неможливість транспортувати стічні води через структурні або експлуатаційні дефекти може призвести до значних проблем.

Більшість колекторів великого діаметру прокладені 20-30 чи більше років тому та знаходяться не в найкращому технічному стані. Найбільш серйозною проблемою експлуатації є руйнування верхнього зводу в результаті зносу. Якщо стан таких колекторів не контролювати, їх руйнування може спричинити провали. Саме вони стають причиною погіршення екологічного стану в різних населених пунктах та несуть загрозу для життя людей[1].

Дослідження також показують, що 40,3% очисних споруд, насосних станцій та резервуарів у більшості країн знаходяться у неприйнятному для нормальної експлуатації стані[2].

Муніципалітети виділяють значні кошти для оцінки та відновлення каналізації та трубопровідних систем, тому, щоб передбачити бюджет процесу відновлення або заміни проблемних ділянок, необхідно провести ретельну перевірку[2].

До 1960-х років перевірка каналізаційних трубопроводів була складним завданням . У більшості випадків, працівникам було важко отримати доступ до місць, де вимагалася перевірка. У відповідь на цю проблему була розроблені спеціальні методи, діагностики. Це, в свою чергу, призвело до розвитку технології, що вдосконалили підходи до перевірки каналізації. Відеоінспекція (CCTV) була винайдена в 1960-х роках[3]. Це був надзвичайно важливий винахід, оскільки для діагностики не треба було руйнувати працездатні труби.

Діагностика дає можливість отримати інформацію про стан верхнього зводу колектора та створити план ремонтних робіт, які дозволяють запобігти аварій та трагедій.

Ще одна розповсюджена проблема при експлуатації каналізаційних мереж – засмічення. Вони можуть з’явитися як в каналізації житлових будинків, так і в мережах виробничих приміщень. У будь-якому випадку, необхідно оперативно прибрати засмічення з мінімальними витратами. Відчутно знизити витрати допомагає відео діагностика труб, у ході якої виявляється точне місце утворення засмічення. Діагностика проводиться без розрізання трубопроводу, що дозволяє заощадити час та кошти. Діагностика також проводиться в наступних випадках:

При здачі в експлуатацію нових або відремонтованих мереж для визначення якості виконаних робіт.

Для виявлення місцезнаходження та протяжності труб, місцезнаходження колодязів, якщо технічна документація втрачена.

Для визначення стану каналізаційних мереж, можливості їх подальшої експлуатації, плану ремонтних робіт.

З наведеної вище інформації можна скласти наступні вимоги для інформаційної системи, яка буде розроблюватися складання:

Точне місцезнаходження інженерних систем.

Наявність будь-яких дефектів або пошкоджень (тріщини, деформації і т.д.).

Наявність всередині об’єкта стороннього предмета.

Стан труб для виявлення необхідності проведення ремонту або заміни пошкодженої ділянки.

1 ІНФОРМАЦІЙНИЙ ОГЛЯД

1.1 Сучасні тенденції і стан розвитку систем діагностики сточних труб

Станом на 2021 рік популярними методами для дослідження стану труб залишаються:

- Сателітні системи.
- Обладнання яке проштовхується.
- Самохідні системи на основі роботів.

Роботизоване обладнання в свою чергу ділиться на плаваюче та колісне. Перше застосовується в частково заповнених трубах, а інше в незаповнених трубопроводах. До складу сателітних систем входить декілька камер для дослідження розгалужень труб. Системи що проштовхуються являють собою камеру, закріплену на жорсткому кабелі.

1.2 Відеоінспекція труб

Відеоінспекція - це техніка, яка дозволяє реєструвати відеозображення в підземних трубах. Це особливо корисно для перевірки стану труб, які можуть бути занадто вузькими або небезпечними для людей. У перші часи використання цієї методики камери кріпились на кабелі між двома люками. Але з часом відеокамери почали встановлювати на гусеничні платформи або плавучі пристрої. Оператори мають можливість керувати рухами робота та камери на великій відстані. Камера зафіксовує внутрішню поверхню та стан трубопроводу, надаючи інформацію вище нормального рівня стоку. Згодом фахівці використовують отримане відео для аналізу, розшифровки та прийняття рішень щодо стану трубопроводу. На рис. 1.1 можна побачити кадр даного дослідження.

Хоча деякі складні технології були запроваджені для перевірки каналізації, система відеоспостереження все ще залишається найчастіше використовуваною технікою.

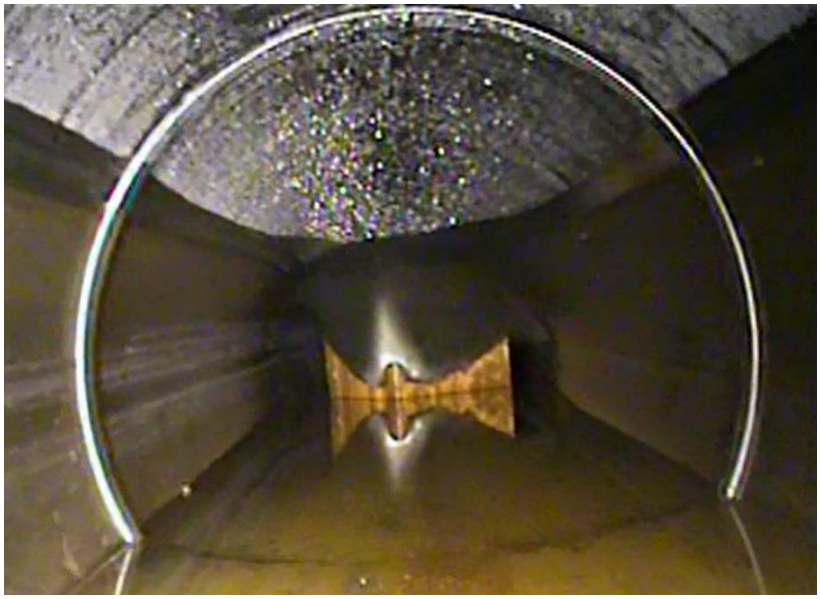


Рисунок 1.1 – Кадр з відеоінспекції труби

1.3 Сучасні методи дослідження сточних труб

1.3.1 Лазерний профайлер

Новим кроком у технології інспекції каналізації є лазерний профайлер. Лазерний профайлер – це технологія, здатна виявляти і кількісно визначати зміни у вертикальній і горизонтальній формі трубопроводів. Існує два типи лазерних профайлерів: двовимірний (2-D) лазерний профайлер та тривимірний (3-D) лазерний профайлер [4]. Двовимірна технологія лазерного профайлера базується на кільці світла, генерованому лазером, навколо стінки трубопроводу. Камера відеоспостереження, яка кріпиться на той самий сканер, виявляє кільце світла і зберігає лазерне зображення для подальшого аналізу. Використовуючи лише відеоспостереження, оператор може не спостерігати жодного прогину вздовж трубопроводу аналізуючи записане відео. Однак використання двовимірного лазерного профайлера чітко представляло б реальний стан трубопроводу [4]. Тривимірний лазерний профайлер є більш досконалою системою. Він використовує лазерну точку та пучки, які мають приймач і двосторонній передавач. Вихідні дані інспекції – це тривимірна діаграма координат X, Y та Z трубопроводу (рис. 1.2).

Лазер має здатність виявляти будь-які малі зміни в структурі трубопроводу, які можуть бути пропущені при використанні відеоспостереження. Натомість, відеокамери ефективно виявляють тріщини та розриви. Комбіноване використання цих двох технологій дозволяє зекономити ресурси та створює більш точні плани ремонту каналізаційних труб [4].

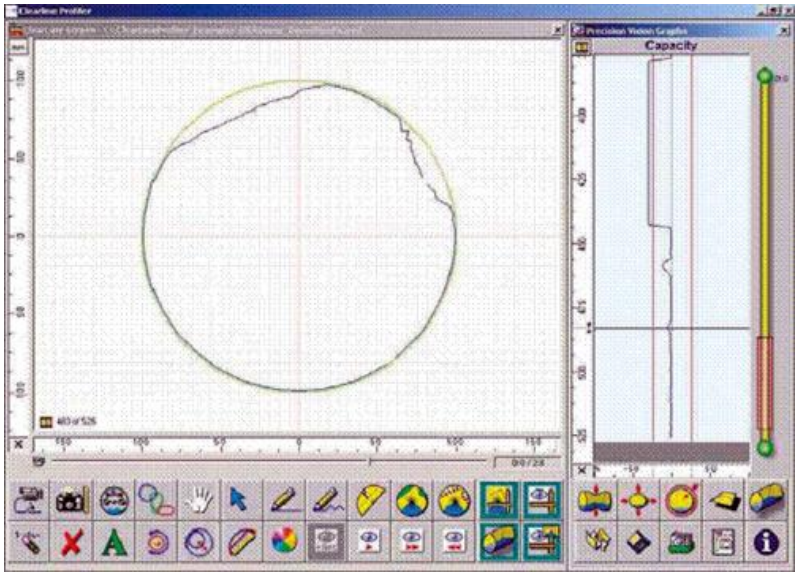


Рисунок 1.2 – Результат дослідження лазерним профайлером

1.3.2 Сонар

Сонар – це застосування акустичних технологій. Він заснований на реалізації звукової енергії де величина частоти вища, ніж може почути людина. Звукові хвилі проходять крізь внутрішню частину труби. Хвилі відбиваються щоразу, коли відбувається зміна щільності матеріалу. Частина відбитих хвиль проходить через нове середовище, тоді як інші повертаються на поверхню. Вибір частоти хвилі значно впливає на зображення, створене датчиком ехолота. При збільшенні акустичної частоти проникаюча сила зменшується. Крім того, швидкість руху також може впливати на якість зображення. Рухаючись зі швидкістю 100 мм в секунду пристрій може виявляти критичні дефекти, але може не виявляти дрібні дефекти [5].

Різноманітні сектори промисловості, зокрема медицина, космічна промисловість та нафтогазова галузь, активно використовують технологію

сонара. Галузь інспекції каналізаційних систем може використовувати ехолокаційні пристрої для аналізу труб різного матеріалу, при цьому найчастіше їх використовують для кількісного вимірювання осадових відкладів, таких як жири та відходи.

В складних роботизованих системах ехолокатори використовуються разом з лазерними профілерами та відеокамерами. Основне застосування датчика ехолота полягає в оцінці обсягу осадових відкладів під рівнем стоку. Сонар може надавати спеціалістам узагальнену оцінку обсягу осаду та визначати ступінь заповнення труби. Застосування гідролокатора разом з лазерними профілерами та відеокамерами дозволяє створювати 3D-модель, яка відображає поточний стан трубопроводу.

Ряд зображень, отриманих ехолокатором, показує реальний стан труби під рівнем потоку, а лазерні профілі та відеокамери забезпечують інформацію вище рівня потоку. Інтегроване використання лазерного профайлера та ехолота надає дані про геометричну форму трубопроводу, виявляючи будь-які деформації, руйнування внаслідок корозії стінок та осідання відкладень на дні труби [5].

1.3.3 Цифрове сканування

Цифрове сканування використовує камери. Камери розташовані на гусеничному пристрої, який переміщується по трубопроводу. Для цифрового сканування застосовуються два види високоякісних камер. Ці камери надають дані про частини трубопроводу і створюють круговий обзор трубопроводу, аналогічний до того, який здійснює відеоспостереження. Темп перевірки за цим методом може бути в два або три рази швидшим, ніж зазвичай при відеоінспекції. Оператору надається панорамний перегляд трубопроводу, оскільки цифрове сканування автоматично надає таку інформацію.

Однак він не надає інформацію нижче лінії стічних вод та висновки записаних даних носять суб’єктивний характер [6].

1.3.4 Інфрачервоне сканування

Інфрачервоні промені переходять від теплих до холодних об'єктів. Цей принцип працює для будь-якого матеріалу, хоча кожен матеріал зберігає тепло по-різному, в залежності від його ізоляційних властивостей. Цю ідею використовують для перевірки стану каналізації. Інфрачервоне сканування використовує камеру, щоб заміряти інфрачервону частоту випромінювання на поверхні трубопроводу. Система складається з інфрачервоного датчика, оптичного об'єктива, мікропроцесора, монітора, пристрою для збору даних, обладнання для аналізу зображень і приймачів [6].

1.3.5 Радар (Pipe Penetrating Radar)

Цей метод використовує принципи радіолокаційної системи, в якій антена генерує високочастотні радіохвилі. Використовуючи радар в трубі, сигнал проникає через стінки труби в навколишній грунт. Система може використовувати дві або три антени для детектування різних частот, що допомагає оцінити обстановку та структуру самої труби.

Робот SewerVUE, що використовує концепцію PPR, здатний надати дані про товщину стін труб, ізоляції, люків та стану облицювання для не металевих труб. Цей робот також обладнаний системами відеоспостереження та LIDAR сканером [7].

1.4 Автоматизація процесу інспекції стічних труб

Автоматизація застосовується для отримання швидких, точних та надійних результатів. Багато дослідників використовують саме цей метод в даній галузі. Вона використовується при оцінці стану автострад, мостів, водопровідних мережі, каналізаційні мережі та тунелі. Існує багато методів автоматизації, які можна використовувати для цієї мети.

Людський зір отримує зображення у 3-D. Люди розпізнають оточення за допомогою інформації, отриманою з допомогою переважно зору. Потім інформація перекладається за допомогою бази даних, доступної в їхньому мозку.

Подібним чином застосовуються комп'ютерні методи обробки зображень для імітації використання людського мозку в переклад цифрових зображень. На відміну від загального людського зору, комп’ютерна обробка зображень здебільшого базується на 2-D зображеннях. Він використовує кілька алгоритмів для застосування декількох операцій над цифровими зображеннями. Це одночасно найпопулярніша та найскладніша тема в галузі інформаційних технологій.

Величезні зусилля докладаються до впровадження обробки зображень в автоматизації перевірки стічних труб. Експерти сходяться на думці, що автоматизація перевірки трубопроводів може заощадити значний час і гроші. Це також підвищує точність і гарантує правильну послідовність дій при виконанні робіт. Інструмент комп'ютерного зору використовує методи математики, штучного інтелекту та розпізнавання образів. Після застосування певної форми алгоритму на виході може бути зображення, набір характеристик або параметри, які відносяться до вихідного зображення [8]. Це дозволяє комп’ютеру зрозуміти зміст зображення шляхом визначення певних параметрів.

1.5 Постановка задачі

Оскільки стічні труби один з найважливіших об’єктів інфраструктури великих міст, то для них необхідний постійний нагляд та моніторинг. Зазвичай, методом для аналізу стану зливних стуб є огляд за допомогою різноманітних відеотехнічних засобів, таких як просувні камери, або спеціальні роботизовані камери на дистанційному керуванні. Всі результати досліджень автоматично або в ручному режимі кодуються в один із стандартів (MSCC5, PACP6, PACP7) [2]. Для правильного кодування результатів досліджень нам необхідно мати інформацію про місцезнаходження, відносне положення камери, безпосередньо

самі пошкодження труб. Але зазвичай всі ці дані недоступні без конкретних значень місцезнаходження камери. Саме тому, задача розпізнавання контексту поточних спостережень стічних труб є надзвичайно необхідною на даний час.

Аналіз проблеми діагностики стічних каналізаційних труб показав, що на даний час, винайдена достатня кількість рішень для їх ручної перевірки. Використання людської праці для інспектування міських колекторів недоцільне, оскільки:

– надто висока вартість послуг спеціалістів з відеодіагностики;
– довготривалість процедури;
– дуже суб’єктивні результати перевірки.

Проаналізувавши всі недоліки була отримана наступна задача: Побудувати алгоритм машинного навчання, який би задовольняв наступним критеріям:

– простота;
– висока точність;
– прийнятна швидкість навчання алгоритму;
– завадозахищеність.

2 ІНФОРМАЦІЙНА СИСТЕМА РОЗПІЗНАВАННЯ КОНТЕКСТУ СПОСТЕРЕЖЕНЬ НА ЗОБРАЖЕННЯХ ІНСПЕКЦІЇ СТІЧНИХ ТРУБ

2.1 Основні положення машинного навчання

Машинне навчання (або Machine Learning, ML) належить до методів штучного інтелекту, які вчать комп'ютер самостійно вирішувати різні завдання. Комп'ютери проводять аналітичну роботу і визначають закономірності набагато швидше за людей завдяки заздалегідь завантаженим даним і спеціальним алгоритмам.

Алгоритми визначаються залежно від того, яке завдання необхідно вирішити і якими даними володіють розробники. Набір навчальних даних надають алгоритмам, які з їхньою допомогою обробляють різні запити.

Як правило, комп'ютерам потрібен великий обсяг інформації і статистики, щоб навчитися створювати правильні й потрібні прогнози.

Види машинного навчання:

- Класичне:

Класичне навчання буває з учителем і без нього. Якщо машина тренується виконувати завдання з учителем, вона отримує розмічені дані. Таким чином, комп'ютеру вдається швидше видавати результати.

Якщо алгоритми тренуються без учителя, їм доводиться самостійно аналізувати інформацію й шукати закономірності. Такий підхід може тривати довше, однак розробникам не потрібно готувати базу даних заздалегідь.

- З підкріпленням:

Навчання з підкріпленням використовується для того, щоб роботи навчилися виживати в різних середовищах і адаптуватися до ситуації. Такий метод також застосовують для навчання персонажів в іграх і безпілотних автомобілів. Їм необхідно узагальнити ситуацію й отримати з неї вигоду.

- Ансамблі:

Якщо розробники використовують метод ансамблів, вони збирають разом кілька машин із різними методами навчання. Далі машини навчаються виправляти помилки одне одного.

- Глибоке:

Глибоке навчання використовується для нейромереж. Завдяки йому нейронні мережі виконують завдання комп’ютерного зору, розпізнавання мови й машинного перекладу.

2.2 Ідеї та методи машинного навчання з самоучителем

Існують такі види навчання:

- навчання по розмічених даних (Supervised Learning / SL) - навчальна вибірка складається з пар (x, y), де x – опис об'єкта, y – його мітка, і необхідно навчити модель y = f (x), яка за описами отримує мітки. Часто таке навчання називають «навчанням з учителем».

- навчання з частково розміченими даними (Semi-Supervised Learning / SSL) - навчальна вибірка складається з даних з мітками і без міток (останніх, як правило, істотно більше), необхідно також навчити модель y = f (x).

- навчання по нерозміченим даними (Unsupervised Learning / UL) – дані тільки об'єкти (без міток), необхідно ефективно описати, як вони розташовуються в просторі описів. Наприклад, типові завдання навчання по нерозмічену даними - кластеризація, зниження розмірності, знаходження аномалій, оцінка щільності і т.д.

В останні роки надзвичайно популярним стає напрямок самонавчання, або навчання з самоучителем (self-supervised learning). Тут є величезна нерозмічена вибірка, необхідно сформувати для кожного об'єкта псевдо-мітку (pseudo label) і вирішити отриману SL-завдання, але нас цікавить не стільки якість рішення придуманого нами завдання (його називають pretext task), скільки уявлення (representation) об'єктів, яке буде вивчено в ході її рішення. Це уявлення можна в подальшому використовувати вже при вирішенні будь-якої задачі з мітками (SL),

яку називають подальшим завданням (downstream task). Одна з головних причин самонавчання - невеликий обсяг розмічених даних (в цьому випадку виникає спокуса використання розділеного дані, можливо з іншого домену). На відміну від навчання з частково розміченими даними в самонавчанні використовуються абсолютно довільні нерозмічені дані (що не мають відношення до розв'язуваної задачі).

Попереднє завдання (pretext task) – задача з штучно створеними мітками (псевдо-мітками), на якій навчається модель, щоб вивчити нормальні представлення (representations) об'єктів. Наприклад, є надія, що в нейронній мережі на попередній задачі навчаться початкові і середні шари, їх потім заморожують і навчають останні шари.

Псевдо-мітки (pseudo labels) – мітки, які виходять автоматично, без ручної розмітки, але навчання ним сприяє формуванню хороших представлень.

Подальша задача (downstream task) – задача на якій перевіряють якість отриманих уявлень. Майже у всіх експериментах в статтях по самонавчанню на отриманих ознакових представленнях в наступних завданнях навчають прості моделі: логістичну регресію або метод найближчого сусіда. Таким чином, самонавчання - це напрямок в глибокому навчанні, яке прагне зробити глибоке навчання процедурою попередньої обробки даних, тобто мережі потрібні для формування ознак, вони навчаються на дешевій розмітці великих наборів спочатку нерозмічену даних, а сама задача вирішується простою моделлю.

В останні роки стало значно менше робіт з вибору попередніх завдань, оскільки основним напрямком в самонавчанні стало порівняльне навчання (Contrastive Learning). На вхід нейромережі подається пара об'єктів і вона визначає, схожі вони чи ні. Як об'єкт тут подається аугментований патч з зображення, схожі повинні бути патчі з одного зображення, а не схожі - з різних. В якості функцій що оптимізуються використовується взаємна інформація або пов'язані з нею функції, наприклад, її нижня оцінка InfoNCE:

$$-E_x\left[\log\frac{\exp\left(f(x)^T f(f^+)\right)}{\exp\left(f(x)^T f(f^+)\right)+\sum_{j=1}^{N-1}\exp(f(x)^T f(x_j))}\right]$$

де x – обраний патч, який називають якірним, x^+ - схожий на нього, x_j – несхожий (їх $N-1$ штука), $f()$ кодувальник об’єкта (представлення його у вигляді вектора).

Під таку схему підходять наступні сучасні методи:

- Invariant Information Clustering (IIC) – оптимізується взаємна інформація, використовується мережа, що здійснює м'яку (soft) класифікацію.
- Deep InfoMax (DIM) - оптимізується MI (може бути реалізовано різними способами, наприклад через InfoNCE) між входом нейромережі (подаємо зображення) і вивчається високорівневим представленням. Щоб уявлення задовольняло певним статистичним властивостям використовується adversarial learning.
- Augmented Multiscale Deep InfoMax (AMDIM) - Просунута реалізація попереднього підходу.
- Momentum Contrast (MoCo) – У порівняльному навчанні чим більше негативних прикладів, тим краще. Зазвичай це число обмежується розміром батча. У методі MoCo підтримується динамічна черга негативних прикладів (коли надходить черговий батч прикладів, дуже старий батч видаляється з черги). Одна з фішок методу - спосіб поновлення параметрів кодувальників (через велику чергу пропускати градієнти занадто довго).
- SimCLR: A Simple Framework for Contrastive Learning of Visual Representations – Цей метод опублікований на початку 2020 року, це просто бібліотека, в якій все правильно реалізовано: береться стандартна схема порівняльного навчання, кодувальник на базі ResNet, представлення пропускається ще через двошарову мережу для отримання векторів над якими вже вимірюється Contrastive loss (саме в наявності такої мережі полягає основна новизна) [9].

2.3 Машинне навчання на основі SimCLR

SimCLR пропонує простий фреймворк для порівняльного навчання візуальних представлень. Він вивчає представлення для візуальних 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.

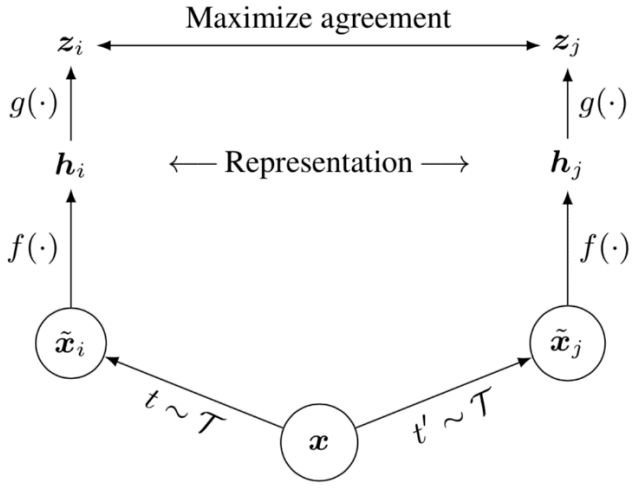


Рисунок 2.1 – Схематичне зображення SimCLR

Даний фреймворк працює в наступні три кроки:

1) Випадково випробовують міні-батч зразків n, та кожний зразок застосовується з двома різними операторами аугментування, в результаті маємо $2n$ аугментованих зразків.

$$\tilde{x}_i = t(x), \qquad \tilde{x}_j = t'(x), \qquad t, t' \sim T$$

де 2 різних оператори аугментування, t та t', відібрані з одного набору аугментацій T. Аугментування даних включає додаткове обрізання, збільшення розміру з випадковим повертанням, спотворення кольорів та розмиття Гауса.

2) Дано одну позитивну пару, інші $2(n-1)$ точки даних розглядаються як негативні зразки. Представлення отримується за допомогою базового кодувальника $f(.)$:

$$h_i = f(\tilde{x}_i), \qquad h_j = f(\tilde{x}_j)$$

3) Вираховується contrastive loss за допомогою подібності косинуса $sim(.,.)$. Зверніть увагу, що втрата діє над додатковою проекцією представлень через $g(.,.)$, а не безпосередньо з поданням g. Але лише представлення h використовується для подальших завдань.

$$z_i = g(h_i), \qquad z_j = g(h_j), sim(z_i, z_j) = \frac{z_i z_i}{\|z_i\|\|z_j\|}$$

$$L_{i,j} = -\log \frac{\exp\left(sim(z_i, z_i)/\tau\right)}{\sum_{k=1}^{2n} 1_{[k \neq i]} \exp\left(sim(z_i, z_i)/\tau\right)}$$

де $1_{[k \neq i]}$ функція індикатор: 1 якщо $k \neq i$, у противному випадку 0. τ – гіперпараметр температури [10].

На рис. 2.2 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 2.2 – Алгоритм роботи SimCLR

2.4 Класифікація

Задача класифікації – це задача, у якій існує певна множина об’єктів, що певним чином поділені на класи. Дана скінченна множина об’єктів, для котрих відомо до яких саме класів вони відносяться. Ця множина називається вибіркою. Класова належність інших об’єктів невідома. Потребується побудувати алгоритм, що здатний класифікувати довільно обраний об’єкт з наданої множини.

Класифікація це підкатегорія машинного навчання з учителем, де головною метою є передбачити певний певну категорію (клас) нового об’єкта, базуючись на попередніх спостереженнях [14].

Існують чотири головних типи класифікаційних задач з якими можна працювати:

- Бінарна класифікація (Binary Classification) – найпростіший варіант, який може бути основою для вирішення більш складних задач.
- Мультикласова класифікація (Multi-class Classification) – коли кількість класів досягає багатьох тисяч, задача класифікації стає набагато складнішою.
- Суміжні класи (Multi-label Classification) – певний об’єкт може відноситися одночасно до декількох класів.
- Незбалансовані класи (Imbalanced Classification) – сценарій, при якому кількість об’єктів у класах не рівна [14].

2.4.1 Бінарна класифікація

Бінарна класифікація відноситься до таких задач класифікації, в яких є лише 2 класи. Зазвичай, такі задачі включають в себе один клас який є нормальним станом та другий клас що є не нормальним станом.

Наприклад, для поштових листів «не спам» це нормальний стан, а «спам» не нормальний стан. Або «вірус не знайдено» це нормальний стан для дослідження медичних аналізів, а «вірус знайдено» - не нормальний стан.

Зазвичай, клас для нормального стану помічається 0, а клас з не нормальним станом – 1.

Популярні алгоритми, що можуть використовуватися для бінарної класифікації включають в себе:

- Logistic Regression
- k-Nearest Neighbors
- Decision Trees
- Support Vector Machine
- Naive Bayes

Деякі алгоритми спеціально розроблені для задачі бінарної класифікації і в початковому вигляді не підтримують більше ніж 2 класи. До таких алгоритмів можна віднести Logistic Regression та Support Vector Machines [15].

2.4.2 Мультикласова класифікація

Мультикласова класифікація відноситься до таких задач класифікації, в яких існує більше ніж дві мітки класів.

До таких задач можна віднести:

- Розпізнавання сортів рослин.
- Класифікація облич.

На відміну від бінарної класифікації, мультикласова класифікація не має «нормального» та «не нормального» станів. Навпаки, зразки маркуються як належні до одного класу з певної множини інших класів.

Кількість класів може бути надзвичайно великою. Наприклад, модель може розпізнати фото, що належить до одного з тисяч або десятків тисяч облич у системі розпізнавання облич.

Задачі, що включають в себе передбачення послідовності слів, такі як моделі для перекладу, так само можуть бути розглянуті як спеціальний вид мультикласового розпізнавання. Кожне слово у послідовності що повинно передбачатися, включає в себе задачу мультикласового розпізнавання, де розмір

словника позначає розмір можливої кількості класів що можуть бути передбачені [15].

Деякі алгоритми, що використовуються для бінарної класифікації можуть бути використані також і для мультикласової класифікації. Використовуються наступні алгоритми:

- k-Nearest Neighbors.
- Decision Trees.
- Naive Bayes.
- Random Forest.
- Gradient Boosting.

Алгоритми, що призначені лише для бінарної класифікації можуть бути адаптовані для задач мультикласової класифікації. Це значить, що використовується стратегія, в якій декілька бінарних класифікаторів для кожного класу протиставляються решті класів (one-vs-rest) або одна модель для кожної пари класів.

- One-vs-Rest – один бінарний класифікатор для кожного класу протиставляється всім іншим класам.
- One-vs-One – один бінарний класифікатор для кожної пари класів.

До бінарних класифікаторів, що можуть використовувати наступні стратегії відносять:

- Logistic Regression.
- Support Vector Machine

2.4.3 Суміжні класи

Класифікація суміжних класів це вид класифікації, в якому два або більше класів та об’єкт може бути помічений як належний до декількох класів одночасно.

Наприклад, в задачі класифікації фотографій, де на заданому фото можуть бути декілька об’єктів та модель класифікатора розпізнає присутність декількох відомих об’єктів, таких як «людина», «машина», «будинок», «річка».

Алгоритми класифікації, що використовуються для бінарного або мультикласового класифікування не можуть бути використовуватися безпосередньо для задачі класифікації суміжних класів [15]. Для цього використовуються спеціальні форми алгоритмів, так звані версії для суміжних класів, до яких відносять:

- Multi-label Decision Trees
- Multi-label Random Forests
- Multi-label Gradient Boosting

2.4.4 Незбалансовані класи

Незбалансовані класи відносяться до класифікаційних задач, де кількість прикладів в кожному класі відрізняється.

Зазвичай, задачі з незбалансованими класифікаторами це задачі для бінарної класифікації, де більшість об’єктів вибірки належить до нормального класу, а решта до ненормального класу.

Наприклад:

- Розпізнавання шахрайства.
- Розпізнавання проникнень на службові об’єкти.
- Медичні тести.

Ці проблеми можуть бути розглянуті як задачі бінарної класифікації, але можуть потребувати спеціальних технік.

Використовуються спеціальні алгоритми, що приділяють більше уваги саме класу меншості. До таких алгоритмів відносять:

- Cost-sensitive Logistic Regression.
- Cost-sensitive Decision Trees.
- Cost-sensitive Support Vector Machines.

2.5 Алгоритми оптимізації нейромереж

2.5.1 Adam

Adam (adaptive movement estimation) - це алгоритм оптимізації на основі градієнтів першого порядку стохастичної цільової функції, базується на адаптивних оцінках моментів нижчого порядку. Метод дуже простий у реалізації, ефективний у застосуванні, має невеликі вимоги до пам'яті, інваріантний діагональному масштабуванню градієнтів і добре підходить для проблем, що мають великі розміри даних або параметрів. Цей метод також підходить для нестаціонарних цілей і проблем, пов'язаних з дуже шумними або розрідженими градієнтами. Гіперпараметри для цього методу мають інтуїтивну інтерпретацію та зазвичай потребують незначного налаштування. Емпіричні результати показують, що Adam ефективний у практичному використанні і показує кращі результати порівняно з методами стохастичної оптимізації. [11] Дуже важливим є питання кількості ітерацій, яку потрібно зробити, навчаючи штучну мережу.

Використовуючи великі моделі та набори даних, ми демонструємо, що Adam може ефективно вирішувати практичні проблеми глибокого навчання.

Розробники алгоритму описують Adam як комбінацію переваг інших двох розширень стохастичного градієнтного спуску (SDG). А саме:

- Адаптивний градієнтний алгоритм (AdaGrad), що впроваджує параметрову швидкість навчання, яка в свою чергу покращує продуктивність у вирішенні задач з розсіяними градієнтами.
- RMSProp (Root Mean Square Propagation) – який також пропонує попараметрову швидкість навчання, яка залежить від середньої ваги останніх коливань градієнтів (тобто наскільки швидко вони змінюються). Це значить що алгоритм добре працює на різноманітних проблемах в онлайн сфері [16].

Порівняльну характеристику алгоритмів можна побачити на рис. 2.3.

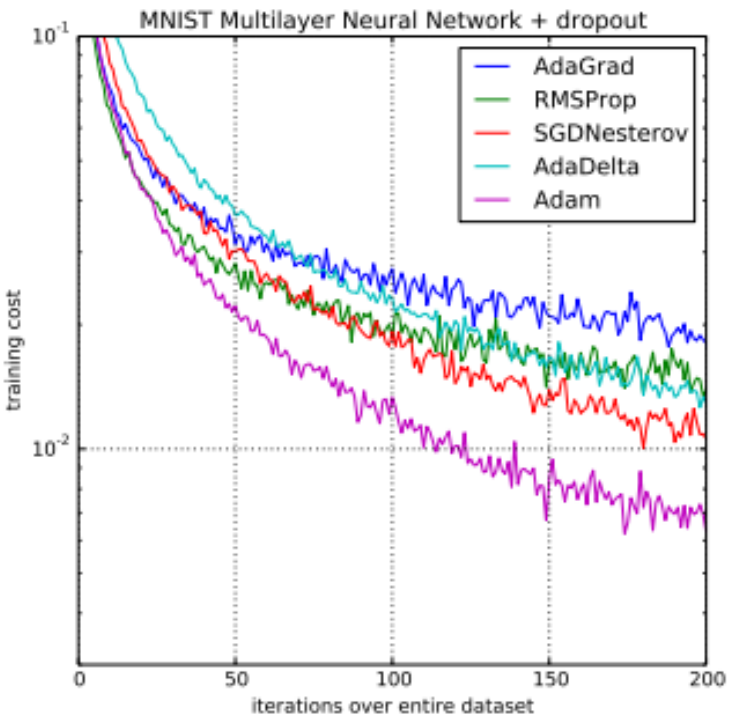


Рисунок 2.3 – Порівняльна характеристика оптимізаційних алгоритмів

На практиці, даний алгоритм рекомендований в якості алгоритму за замовчуванням та часто працює набагато краще ніж RMSProp.

Якщо дані параметри $w^{(t)}$, а функція втрат $L^{(t)}$, де t відображує індекс поточної ітерації, перерахунок алгоритмом Adam задається формулами

$$m_w^{(t+1)} \leftarrow \beta_1 m_w^{(t)} + (1-\beta_1)\nabla_w L^{(t)}$$

$$v_w^{(t+1)} \leftarrow \beta_2 v_w^{(t)} + (1-\beta_2)(\nabla_w L^{(t)})^2$$

$$\widehat{m_w} = \frac{m_w^{(t+1)}}{1-\beta_1^{t+1}}$$

$$\widehat{v_w} = \frac{v_w^{(t+1)}}{1-\beta_2^{t+1}}$$

$$w^{(t+1)} \leftarrow w^{(t)} - \eta \frac{\widehat{m_w}}{\sqrt{\widehat{v_w}} + \epsilon}$$

де є являє собою малий додаток, що використовується для запобігання діленню на 0, а β_1 та β_2 є коефіцієнтами забування для градієнтів відповідно. Піднесення до квадрату та квадратний корінь вираховуються поелементно [11].

2.5.2 SDG

Стохастичний градієнтний спуск (stochastic gradient descent) – ітеративний метод оптимізації градієнтного спуску за допомогою стохастичного наближення. Використовується для прискорення пошуку цільової функції шляхом використання обмеженого за розміром тренувального набору, який вибирається випадкового при кожній ітерації.

$$w_{N+1} = w_N - \alpha \nabla_w E(w; x^{(i)}; y^{(i)}),$$

де $(x^{(i)}; y^{(i)})$ – i-й навчальний набір.

SGD не здійснює зайвих обчислень, оскільки на відміну від класичного градієнтного спуску, функція помилок алгоритму рахується не по всьому навчальному набору, а тільки по одному прикладу, а значить, алгоритм стає значно швидшим, а також допускає можливість навчання на ходу, тобто нові приклади можуть подаватися на вхід безпосередньо в процесі навчання [12].

Але, внаслідок того, що на кожному кроці SGD обчислення градієнта виконується на основі різних прикладів вихідного набору даних, оновлення вагових коефіцієнтів супроводжується частими коливаннями цільової функції, як показано на рис. 2.4. Таким чином, з одного боку, SGD дозволяє швидко рухатися до потенціально найкращим локальним мінімумам, а з іншого боку, великі коливання значно сповільнюють східність.

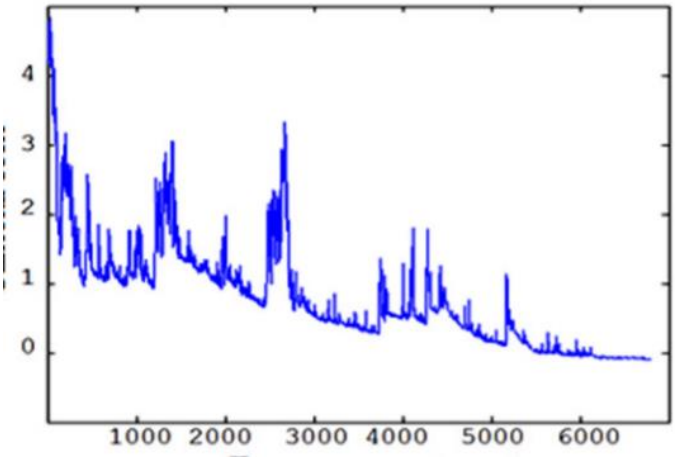


Рисунок 2.4 – Графік залежності цільової функції в залежності від ітерації навчання в алгоритмі SGD

Було доведено, що якщо внести в процес навчання динамічне зменшення швидкості навчання, то алгоритм SGD досягає рішення з точністю, аналогічною класичному градієнтному спуску [12].

2.6 Критерії валідації моделі аналізу даних

Крос-валідація, яку іноді називають перехресною перевіркою, це техніка валідації моделі для перевірки того, наскільки успішно застосовується в моделі статистичний аналіз, чи здатний він працювати на незалежному наборі даних. Зазвичай крос-валідація використовується в ситуаціях, де метою є передбачення, і хотілося б оцінити, наскільки предиктивна модель здатна працювати на практиці. Один цикл крос-валідації включає розбиття набору даних на частини, потім побудова моделі на одній частині (тренувальному наборі), і валідація моделі на іншій частині (тестовому наборі). Щоб зменшити розкид результатів, різні цикли крос-валідації проводяться на різних розбиттях, а результати валідації усереднюються по всіх циклах [13].

До розповсюджених типів крос-валідації відносяться: крос-валідація по K блокам (K-fold cross-validation), яка проілюстрована на рис. 2.5; валідація послідовним випадковим семплуванням (random subsampling)

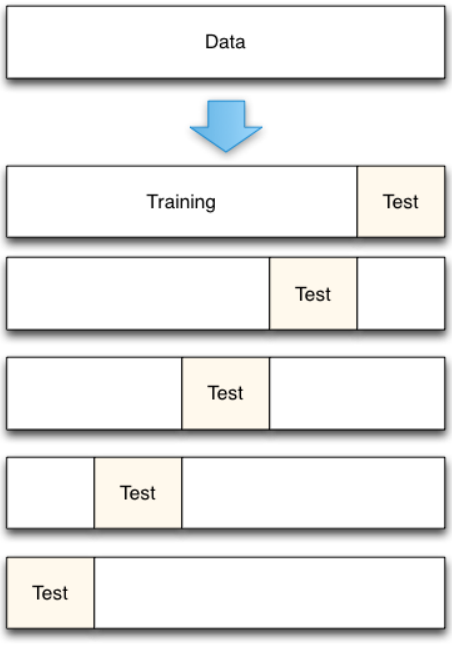


Рисунок 2.5 – Схема крос-валідації по K блокам

Як можемо бачити на рис. 2.5, вхідний набір даних розбивається на K однакових по розміру блоків. Із K блоків один залишається для тестування моделі, а решта блоків K – 1 використовуються як тренувальний набір. Процес повторюється K разів, і кожен з блоків використовується один раз як тестовий набір. Отримуємо K результатів, по одному на кожний блок, вони усереднюються або комбінуються яким завгодно іншим способом та дають одну оцінку. Перевагою такого способу перед випадковим семплуванням (random subsampling) в тому, що всі спостереження використовуються і для тренування, і для тестування моделі, і кожне спостереження використовується лише один раз. Часто спостерігається крос-валідація на 10 блоках, але чітких рекомендацій по кількості блоків немає [13].

В методі валідації послідовним випадковим семплуванням (random subsampling) випадковим чином розбивається набір даних на тренувальний та тестовий набори. Для кожного такого розбиття, модель підганяється під тренувальні дані, а точність передбачення оцінюється на тестовому наборі. Результати потім усереднюються по всім розбиттям. Перевага такого методу перед крос-валідацією на K блоках у тому, що пропорції тренувального та тестового наборів не залежать від кількості повторень (блоків). Недолік метода в тому, що деякі спостереження можуть ні разу не потрапити до тестового набору, коли інші можуть потрапити туди більше одного разу. Іншими словами, тестові набори можуть перекриватися. Окрім того, оскільки розбиття проводиться випадково, результати будуть відрізнятися в результаті повторного аналізу.

Ціль крос-валідації в тому, щоб оцінити очікуваний рівень відповідності моделі даним, незалежним від тих даних, на яких модель тренувалася. Вона може використовуватися для оцінки будь-якої кількості міри відповідності, що підходить для даних та моделі [13].

3 ІНФОРМАЦІЙНА СИСТЕМА РОЗПІЗНАВАННЯ КОНТЕКСТУ СПОСТЕРЕЖЕНЬ НА ЗОБРАЖЕННЯХ ІНСПЕКЦІЇ СТІЧНИХ ТРУБ

3.1 Формування вхідних даних для інформаційної системи

Після інспекції каналізаційних труб, обов’язково повинен бути сформований звіт, у вигляді певного стандарту (MSCC5, PACP7, PACP6). Дані звіти включають в себе інформацію, що відображає тяжкість пошкодження та місцезнаходження. Для розуміння ситуації необхідно розуміти контекст безпосередньо самого кадру. Тобто розташування камери відносно стінок труби, умови всередині стічного каналу, непередбачувані ситуації.

Всього було обрано 9 класів що відображають положення камери відносно труби. Клас X1 (Collapse, рис. 3.1) – позначає завал, що унеможливлює подальший рух камери. Клас X2 (Forward, рис. 3.2) – пряме положення камери відносно труби, при якому максимально чітко видно внутрішні стінки попереду. X3 (Ignore, рис. 3.3) – дані які неможливо інтерпретувати в контексті спостереження. X4 (Manhole, рис. 3.4) – каналізаційний сервісний отвір. X5 (Semi_down, рис. 3.5), X6 (Semi_left, рис. 3.6), X7 (Semi_right, рис. 3.7), X8 (Semi_up, рис. 3.8) – стан, в якому камера знаходиться наполовину повернута до стінки труби, відповідно вниз, вліво, вправо та вгору. X9 (Side, рис. 3.9) – камера повернута повністю до стінки трубопроводу.

Рисунок 3.1 – Зразок класу X1 (Collapse)

Рисунок 3.4 – Зразок класу X2 (Forward)

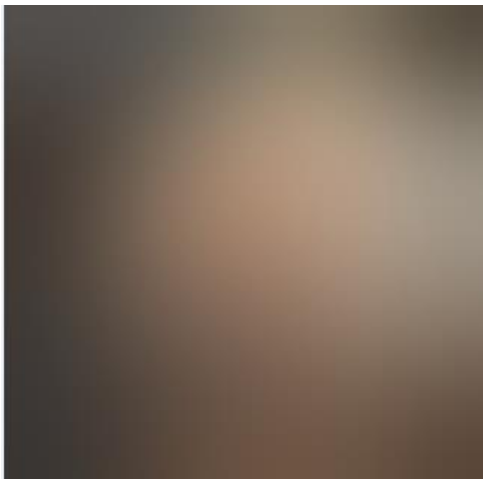


Рисунок 5.3 – Зразок класу X3 (Ignore)

Рисунок 3.6 – Зразок класу X4 (Manhole)

Рисунок 3.7 – Зразок класу X5 (Semi_down)

Рисунок 3.8 – Зразок класу X6 (Semi_left)

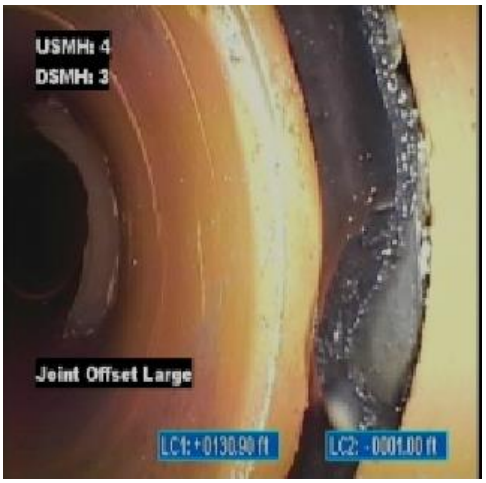


Рисунок 3.9 – Зразок класу X7 (Semi_right)

Рисунок 3.10 – Зразок класу X8 (Semi_up)

Рисунок 3.11 – Зразок класу X9 (Side)

Оскільки розмір більшості кадрів відрізняється, то під час обробки розмір кожного кадру змінюється до 170x170 пікселів. Всього для тренувального датасету було вибрано 500 кадрів для кожного класу і по 100 кадрів для тестування.

3.2 Опис програмних засобів та бібліотек

Для проекту використовується середовище для розробки Google Colab та мова програмування Python 3.8 разом з деякими бібліотеками та модулями наведеними в таблиці 3.1.

Python – це високорівнева мова програмування загального призначення із динамічною типізацією та автоматичним керуванням пам’яттю, яке оптимізовано для підвищення продуктивності розробника, простоти коду та його якості а також для забезпечення мультиплатформовості написаних на цій мові програм. Мова повністю об’єктно орієнтована, тобто взагалі все це об’єкти.

Python є мультипарадигмовою мовою програмування, яка підтримує імперативне, процедурне, структурне, об’єктно-орієнтоване та функціональне програмування. Стандартна бібліотека включає в себе великий набір корисних функцій, починаючи роботою з текстом і закінчуючи написанням складних мережевих програм. Додаткові можливості, такі як математичне моделювання, написання веб-додатків, робота з зовнішнім обладнанням, розробка ігр можуть бути реалізовані шляхом підключення широкого спектру сторонніх бібліотек, а також інтеграцією бібліотек, що написані на C/C++, при цьому сам Python може бути інтегрований в проекти що написані на цих мовах.

Google Colab – це безкоштовний хмарний сервіс на основі Jupyter Notebook. Інтерфейс даного середовища зображений на рис. 3.10.Він пропонує все необхідне машинного навчання прямо в браузері, дає доступ до неймовірно швидких GPU (Graphics Proseccing Unit) та TPU (Google Tensor Proseccing Unit). Підтримує Python 2/3 без додаткових налаштувань. Google Colab може використовуватися для:

– знайомства з TensorFlow – відкритою бібліотекою для машинного навчання;

– експериментів з TPU;

– розробки нейронних мереж;

– створення гайдів;

– різноманітних досліджень в області штучного інтелекту.

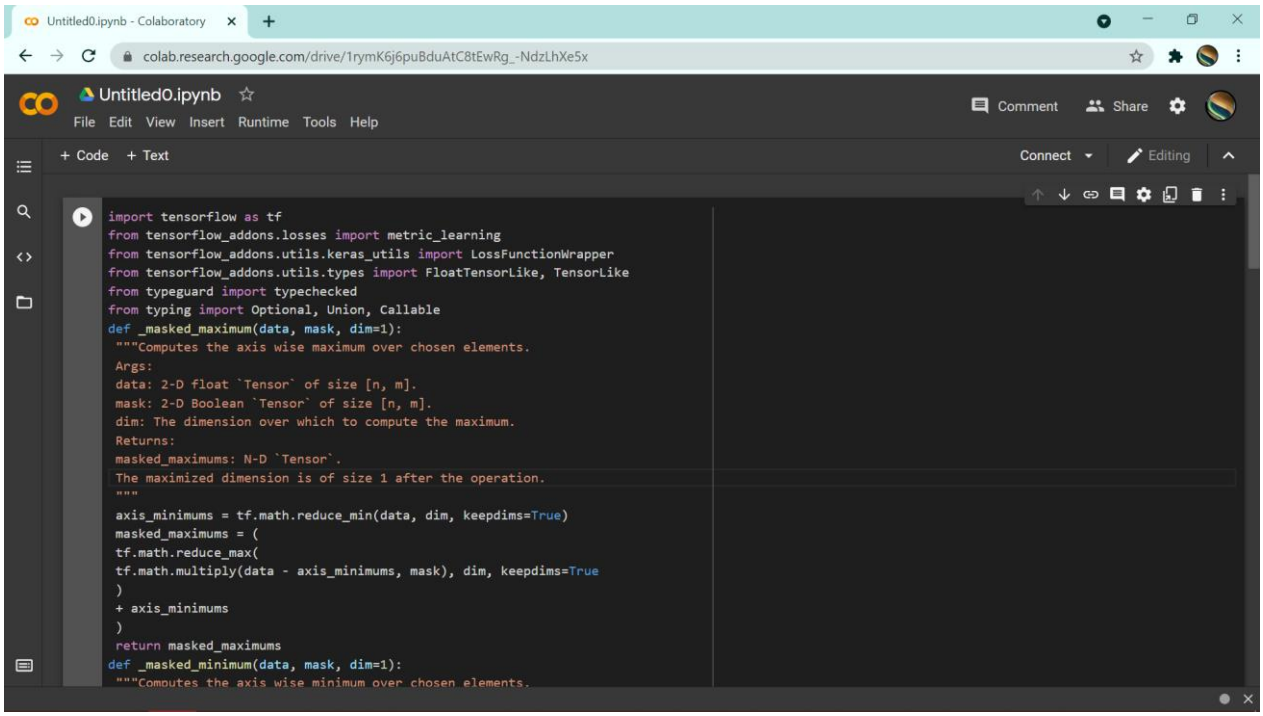


Рисунок 3.10 – Середовище Google Colab

Для роботи використовувалися наступні бібліотеки та модулі:

Таблиця 3.1 – Модулі та бібліотеки проекту.

Назва бібліотеки	Опис
numpy	Бібліотека з відкритим кодом, що підтримує багатовимірні масиви, включаючи матриці та також підтримує високорівневі математичні функції, що призначені для роботи з багатовимірними масивами. Дозволяє використовувати безліч типів даних, таким чином, дана бібліотека може безшовно та з великою швидкістю інтегруватися з багатьма базами даних.
pandas	Бібліотека для обробки та аналізу даних. Робота даної бібліотеки заснована навколо бібліотеки numpy, яка є

Назва бібліотеки	Опис
	інструментом більш низького рівня. Надає спеціальні структури даних та операції для маніпулювання числовими таблицями та числовими рядами.
matpotlib.pyplot	Модуль бібліотеки matplotlib для візуалізації даних в 2D або 3D. Дозволяє використовувати бібліотеку matplotlib майже так само, як і MATLAB. Кожна функція pyplot працює з об'єктами Figure та дозволяє змінювати їх. Наприклад, є функції для створення об'єкту Figure, для створення області побудови, візуалізації різноманітних ліній, додавання міток і т.д.
matplotlib.image	Модуль бібліотеки matplotlib для різноманітних операцій з графічними зображеннями. Дозволяє завантужування графічні зображення, змінювати їх розмір та відображати поточні операції.
tensorflow	Бібліотека з відкритим кодом, розроблена компанією Google для машинного навчання з дуже широким спектром застосування. Призначена для будування та тренування нейронних мереж з метою автоматичного знаходження та класифікування зразків, досягаючи якості людського сприйняття. До особливостей можна віднести те, що може працювати на паралельних процесорах, CPU, GPU (використовуючи архітектуру CUDA для підтримки обчислень на графічних процесорах). Доступна для 64-розрядних Linux, Windows, macOS та для популярних мобільних платформ, таких як Android та iOS. Обчислення в

Назва бібліотеки	Опис
	TensorFlow через граф станів виражаються у вигляді потоків даних. Назва «TensorFlow» походить від операцій з багатовимірними масивами даних, які і називаються «тензорами».
random	Бібліотека, що впроваджує набір псевдо-випадкових генераторів чисел для різноманітних задач.
os	Модуль стандартної бібліотеки Python, який зазвичай використовують для роботи з встановленою операційною системою, а також файловою системою ПК. Методи даного модуля дозволяють визначити тип операційної системи, отримати доступ до змінних середовища, керувати директоріями та файлами.
cv2	Це open source бібліотека комп'ютерного зору, що призначена для аналізу, класифікації та обробки зображень. Дуже розповсюджена в таких мовах програмування як, C, C++, Python та Java.
shutil	Даний модуль містить в собі широкий набір високорівневих функцій для обробки файлів, груп файлів та папок. Функції даного модуля дозволяють копіювати, переміщувати та видаляти файли та папки. Зазвичай використовується разом з модулем os, що згадувався вище.

Таким чином, дані інструменти дають можливість протестувати роботу алгоритмів, що були розроблені з метою подальшої реалізації системи для ідентифікації положення та дефектів каналізаційних труб. Код системи наведений в додатку А.

3.3 Аналіз результатів навчання інформаційної системи.

Модель була навчена визначати контекст спостережень, а саме відносне положення камери в стічній трубі. На рис. 3.11 зображено графік залежності точності тренування та валідації від кількості епох навчання, а саме 30 епох.

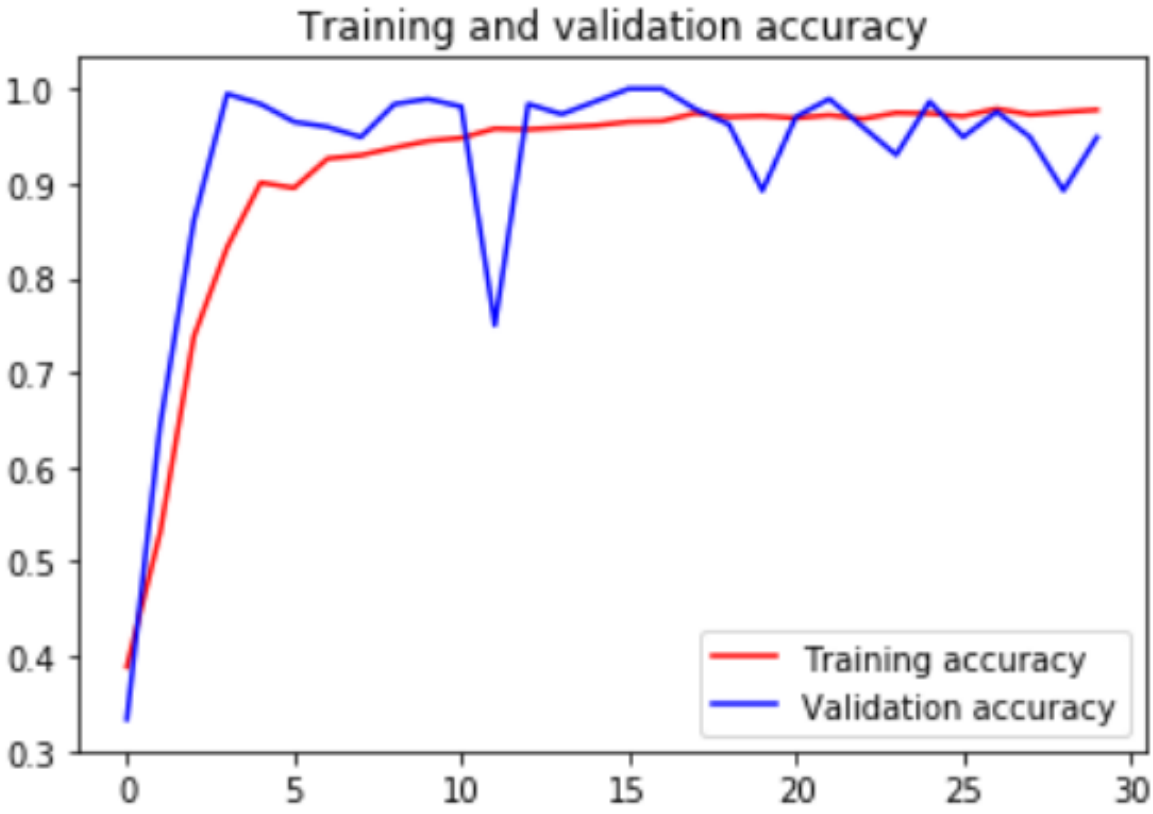


Рисунок 3.11 – Графік залежності точності тренування та валідації інформаційної моделі від кількості епох без додавання шумів

Аналіз рис. 3.11 показує, що збільшення точності за вибіркою валідації (validation accuracy) припинилося на 17 епосі і становить 97.8%. При цьому зменшення помилки за вибіркою валідації (Validation loss) аналогічно зупинилося на 17 епосі, що і можна спостерігати на рис. 3.12.

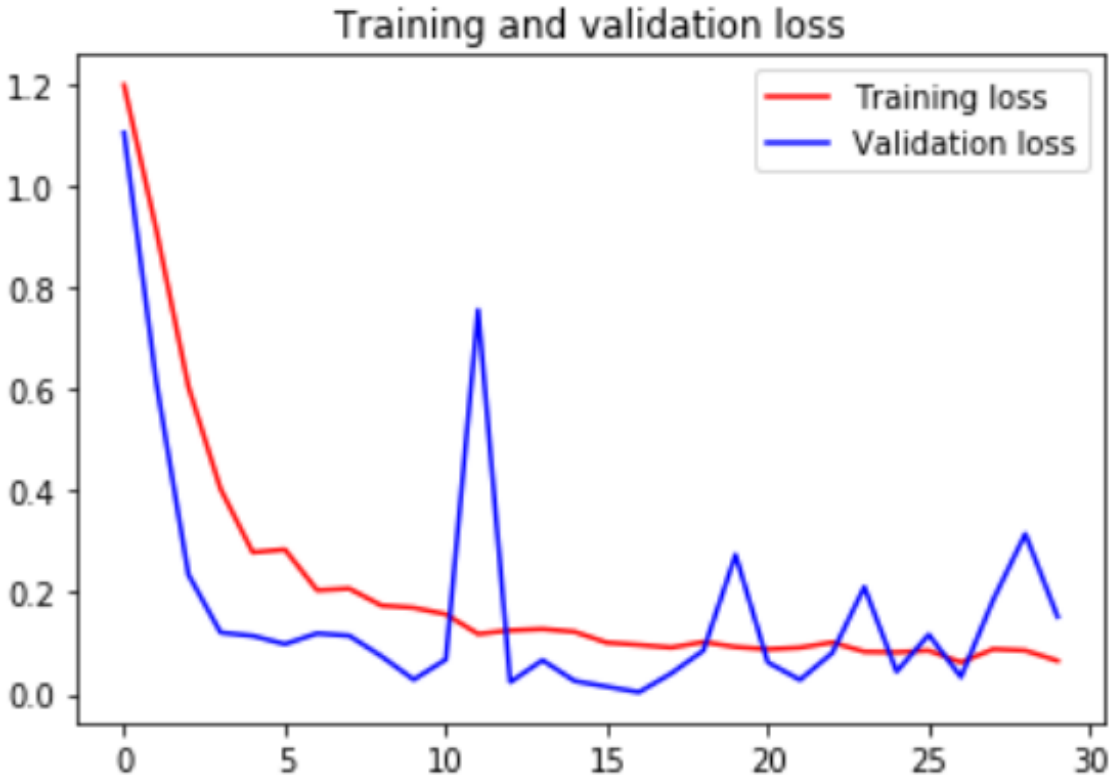


Рисунок 3.12 – Графік залежності функцій втрати на вибірках тренування та валідації від кількості епох без додавання шумів

Для демонстрації завадозахищеності розробленої системи на тестову (валідаційну) вибірку за допомогою функції sp_noise (див. Додаток А). Дана функція випадковим чином додає на зображення шуми «сіль» та «перець», які зашкодять не завадозахищеній системі у розпізнаванні контексту зображень з відеоінспекції трубопроводу. Результати роботи нейронної мережі на датасеті з додаванням шумів наведені на рис. 3.13.

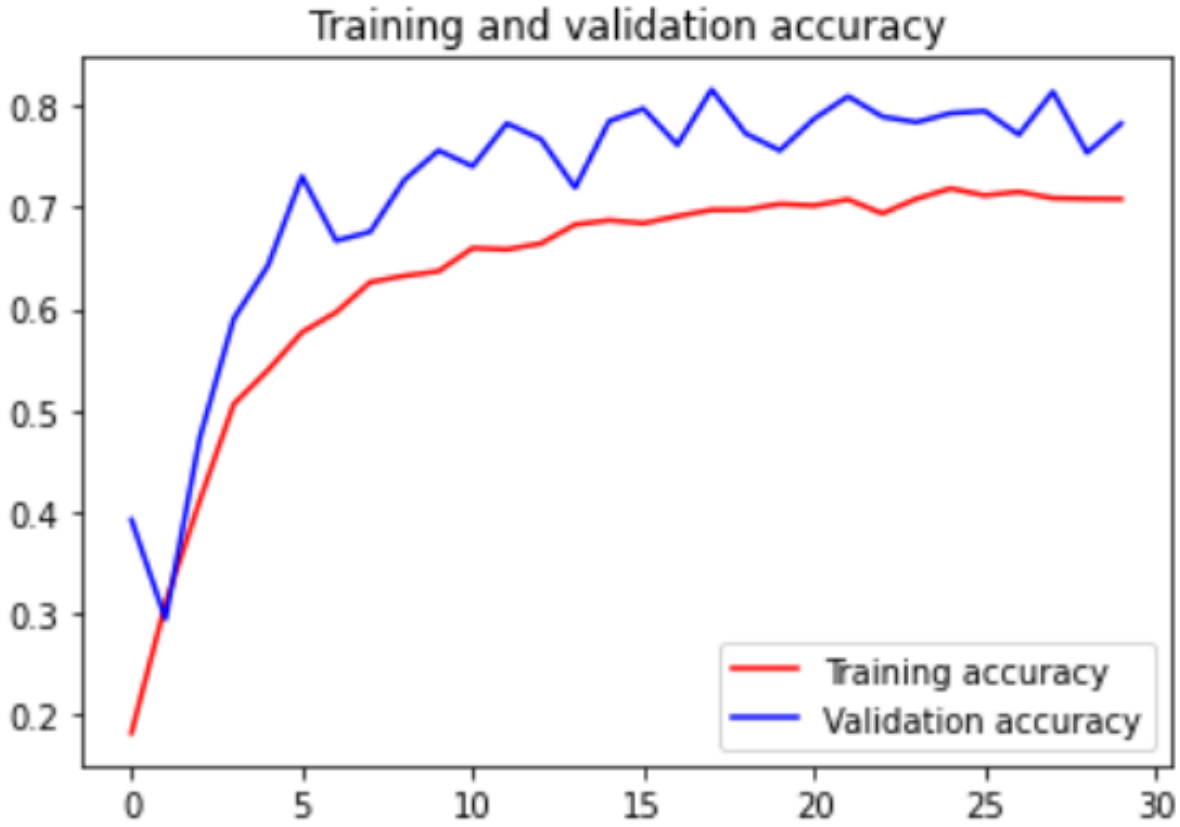


Рисунок 3.13 – Графік залежності тренувальної точності та точності валідації від кількості епох з додаванням цифрового шуму на зображення вибірки

Аналіз графіку, що зображений на рис. 3.13 показує, що точність розпізнавання нейронної мережі знизилися з 97.8% до 81%, що також є доволі високим результатом. Такі результати отримані завдяки тому, що фреймворк SimCLR заснований на сіамських мережах та аугментації, що забезпечує високу робастність моделі до викривлень та інваріативних перетворень, таким чином має високу завадозахищеність.

ВИСНОВКИ

Отже, в рамках даної роботи було розглянуте питання стосовно методів діагностики стічних каналізаційних труб. Оскільки каналізація це одна з основних інфраструктурних одиниц, яка стоїть поряд з електропостачанням, водопостачанням, то необхідно відповідально та своєчасно контролювати та обслуговувати її.

Була розроблена завадозахищена модель розпізнавання візуального контексту для задачі інспекції стічних труб на основі багатошарової сіамської нейронної мережі. Виконано огляд сучасних тенденцій у сфері дослідження каналізації, огляд принципів машинного навчання, та популярних методів у цій сфері. Окремо було розглянуте питання навчання з самоучителем, класифікації, оптимізації нейронних мереж та критерії їх валідації.

Результати навчання виявилися дуже високими (97.9% точності) без накладання цифрових шумів. З накладанням же візуальних шумів точність впала до 81%, що досі залишається досить високим результатом і задовольняє вимогам завадозахищеності системи.

Більш високих результатів можна досягнути лише збільшенням вибірки або збільшенням кількості нейронів на шарах мережі, що призведе до значного зростання часових витрат.

СПИСОК ЛІТЕРАТУРИ

1. Ariaratnam S. T., El-Assaly A., Yang Y. ASSESMENT OF INFRASTRUCTURE INSPECTION NEEDS USING LOGISTIC MODELS / Journal of infrastructure systems. – 2001. – 160 с.

2. American Society of Civil Engineers (ASCE) Report Card for America's Infrastructure 2003 Progress report [Електронний ресурс] / ASCE. – 2004. – Режим доступу до ресурсу: http://www.asce.org/reportcard/2003

3. Abdel-Qader I., Abudayyeh O., & Kelly, M. E. Analysis of edge-detection techniques for crack identification in bridges [Електронний ресурс]/ Journal of Computing in Civil Engineering. – 2003. – Режим доступу до ресурсу: https://ascelibrary.org/doi/abs/10.1061/%28ASCE%290887-3801%282003%2917%3A4%28255%29

4. STORM PIPELINE LASER PROFILING [Електронний ресурс] / Rinker Materials SEWER. – 2009. – Режим доступу до ресурсу: https://www.yumpu.com/en/document/read/41957940/storm-sewer-pipeline-laser-profiling-rinker-materials.

5. Andrews, M. E., & Eng, P. Large diameter sewer condition assessment using combined sonar and CCTV equipment [Електронний ресурс] / APWA International Public Works Congress, NRCC/CPWA Seminar series Innovations in Urban Infrastructure – 2000. – Режим доступу до ресурсу: https://www.semanticscholar.org/paper/LARGE-DIAMETER-SEWER-CONDITION-ASSESSMENT-USING-AND-Andrews/bea315f9ae574cea76c791e495f8e8e04a0baf79

6. Eiswirth. M., Heske. C., Burn. L. S., DeSilva. D. New methods for water pipeline assessment / Eiswirth. M., Heske. C., Burn. L. S., DeSilva. D. – 2001.

7. SeverVUE [Електронний ресурс] – Режим доступу до ресурсу: http://www.sewervue.com/

8. Guo W., Soilbelman L. Automated defect detection for sewer pipeline inspection and condition assessment / Automation in Construction. – 2009. – 568 с.

9. Self-Supervised Representation Learning [Електронний ресурс] – Режим доступу до ресурсу: https://lilianweng.github.io/lil-log/2019/11/10/self-supervised-learning.html#why-self-supervised-learning

10. SimCLR [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/google-research/simclr

11. М. М. Поліщук, С. М. Костючко, М. О. Христинець ПОРІВНЯННЯ МЕТОДІВ ОПТИМІЗАЦІЇ НЕЙРОННИХ МЕРЕЖ НА ПРИКЛАДІ ЗАДАЧІ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ / Науковий журнал "Комп’ютерно-інтегровані технології: освіта, наука, виробництво". – 2019. – 45 с.

12. Song M. A mean field view of the landscape of two-layer neural networks [Електронний ресурс] / Proceedings of the National Academy of Sciences. – 2018. – Режим доступу до ресурсу: https://www.pnas.org/content/115/33/E7665

13. Elements of Statistical Learning: data mining, inference, and prediction. 2nd Edition [Електронний ресурс] – Режим доступу до ресурсу: https://web.stanford.edu/~hastie/ElemStatLearn/

14. Exploring Multi-Class Classification using Deep Learning [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/@srijaneogi31/exploring-multi-class-classification-using-deep-learning-cd3134290887

15. 4 Types of Classification Tasks in Machine Learning [Електронний ресурс] – Режим доступу до ресурсу: https://machinelearningmastery.com/types-of-classification-in-machine-learning/

16. Gentle Introduction to the Adam Optimization Algorithm for Deep Learning [Електронний ресурс] – Режим доступу до ресурсу: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

ДОДАТОК А

Setup

```
[ ] !pip install tensorflow_datasets
```

```
Requirement already satisfied: tensorflow_datasets in /usr/local/lib/python3.7/dist-packages (4.0.1)
Requirement already satisfied: importlib-resources; python_version < "3.9" in /usr/local/lib/python3.7/d
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from tensorflow_datasets)
Requirement already satisfied: tensorflow-metadata in /usr/local/lib/python3.7/dist-packages (from tenso
Requirement already satisfied: dm-tree in /usr/local/lib/python3.7/dist-packages (from tensorflow_datase
Requirement already satisfied: dill in /usr/local/lib/python3.7/dist-packages (from tensorflow_datasets)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from tensorflow_datasets
Requirement already satisfied: promise in /usr/local/lib/python3.7/dist-packages (from tensorflow_datase
Requirement already satisfied: requests>=2.19.0 in /usr/local/lib/python3.7/dist-packages (from tensorfl
Requirement already satisfied: future in /usr/local/lib/python3.7/dist-packages (from tensorflow_dataset
Requirement already satisfied: absl-py in /usr/local/lib/python3.7/dist-packages (from tensorflow_datase
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from tensorflow_datasets)
Requirement already satisfied: attrs>=18.1.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow_
Requirement already satisfied: protobuf>=3.6.1 in /usr/local/lib/python3.7/dist-packages (from tensorflo
Requirement already satisfied: termcolor in /usr/local/lib/python3.7/dist-packages (from tensorflow_data
Requirement already satisfied: zipp>=0.4; python_version < "3.8" in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: googleapis-common-protos<2,>=1.52.0 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from reques
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests>=2.
```

```
[ ] from tensorflow.keras import layers
    from tensorflow.keras import regularizers
    import tensorflow as tf
    import tensorflow_datasets as tfds
    import matplotlib.pyplot as plt
    import numpy as np
```

Define hyperparameters

```
[ ] AUTO = tf.data.AUTOTUNE
    BATCH_SIZE = 16
    EPOCHS = 5
    SEED = 26
    PROJECT_DIM = 1024
    LATENT_DIM = 512
    WEIGHT_DECAY = 0.0005

    CROP_TO = 160
    IMG_SIZE = 224
```

Load the context dataset

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

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

```
[ ] !cp -i /content/gdrive/MyDrive/context_dataset_originals_v1.0.zip /content/context_dataset_originals_v1.0.zip
```

```
[ ] !unzip context_dataset_originals_v1.0.zip -d dataset
```

```
    Показано результат, скорочений до останніх рядків (5000).
      inflating: dataset/context_dataset_originals/tap_capped/S033-066_S033-055_20200903_U_TBC_45.5.jpg
      inflating: dataset/context_dataset_originals/tap_capped/S033-602_S033-044_20200806_U_TFC_164.6.jpg
      inflating: dataset/context_dataset_originals/tap_capped/S034-070_S034-069_20200803_D_TFC_148.6.jpg
      inflating: dataset/context_dataset_originals/tap_capped/S034-253_S034-255_20200730_D_TFC_55.7.jpg
      inflating: dataset/context_dataset_originals/tap_capped/S035-280_S035-281_20170811_U_TFC_131.1.jpg
      inflating: dataset/context_dataset_originals/tap_capped/S035-321_S035-322_20170811_U_TFC_49.1.jpg
```

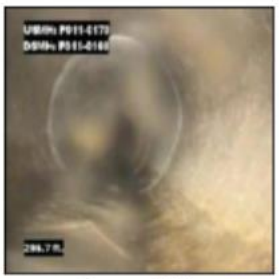
```
[ ] !mv dataset/context_dataset_originals  dataset/train
```

```
[ ] import os
    import shutil

    os.mkdir("/content/dataset/test")

    for dn in os.listdir("/content/dataset/train"):
        path = os.path.join("/content/dataset/test", dn)
        os.mkdir(path)

    source_list = []
    target_list = []
    num = 50
    for dn in os.listdir("/content/dataset/train"):
        for fn in os.listdir("/content/dataset/train/"+dn)[0:num]:
            source_list.append("/content/dataset/train/"+dn+"/"+fn)
            target_list.append("/content/dataset/test/"+dn+"/"+fn)

    for i in range(len(source_list)):
        shutil.move(source_list[i], target_list[i])
```

```
[ ] builder = tfds.ImageFolder('dataset/')
    print(builder.info)

    ssl_ds_one = builder.as_dataset(split='train', shuffle_files=True, as_supervised=True)
    ssl_ds_two = builder.as_dataset(split='train', shuffle_files=True, as_supervised=True)

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

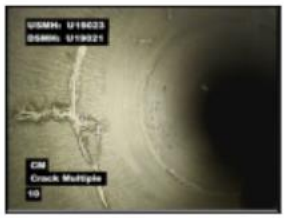
```
tfds.core.DatasetInfo(
    name='image_folder',
    version=1.0.0,
    description='Generic image classification dataset.',
    homepage='https://www.tensorflow.org/datasets/catalog/image_folder',
    features=FeaturesDict({
        'image': Image(shape=(None, None, 3), dtype=tf.uint8),
        'image/filename': Text(shape=(), dtype=tf.string),
        'label': ClassLabel(shape=(), dtype=tf.int64, num_classes=12),
    }),
    total_num_examples=72004,
    splits={
        'test': 600,
        'train': 71404,
    },
    supervised_keys=('image', 'label'),
    citation="""""",
    redistribution_info=,
)
```

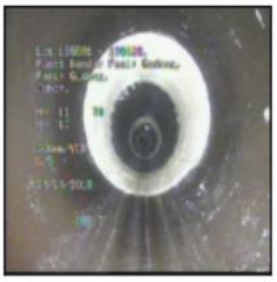

side (8)


forward (1)

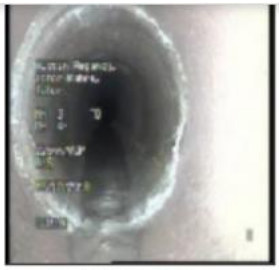

side (8)

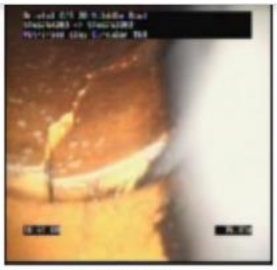

forward (1)


semi_left (5)

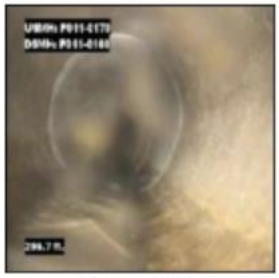

forward (1)

manhole (3)


forward (1)

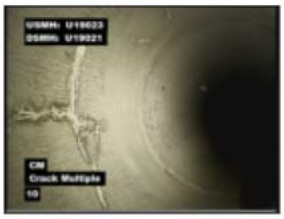

side (8)


side (8)

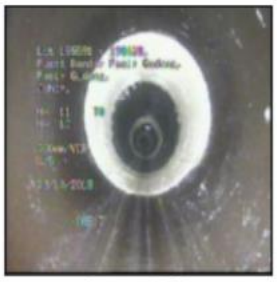

forward (1)

side (8)

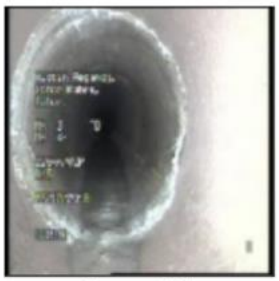
forward (1)



semi_left (5)


forward (1)

manhole (3)


forward (1)


side (8)

Defining our data augmentation pipeline

```
[ ]  def flip_random_crop(image):
         # With random crops we also apply horizontal flipping.
         image = tf.image.random_flip_left_right(image)
         image = tf.image.random_crop(image, (CROP_TO, CROP_TO, 3))
         return image


     def color_jitter(x, strength=[0.4, 0.4, 0.4, 0.1]):
         x = tf.image.random_brightness(x, max_delta=0.8 * strength[0])
         x = tf.image.random_contrast(
             x, lower=1 - 0.8 * strength[1], upper=1 + 0.8 * strength[1]
         )
         x = tf.image.random_saturation(
             x, lower=1 - 0.8 * strength[2], upper=1 + 0.8 * strength[2]
         )
         x = tf.image.random_hue(x, max_delta=0.2 * strength[3])
         x = tf.clip_by_value(x, 0, 255)
         return x


     def color_drop(x):
         x = tf.image.rgb_to_grayscale(x)
         x = tf.tile(x, [1, 1, 3])
         return x


     def random_apply(func, x, p):
         if tf.random.uniform([], minval=0, maxval=1) < p:
             return func(x)
         else:
             return x


     def custom_augment(image):
         image = flip_random_crop(image)
         image = random_apply(color_jitter, image, p=0.8)
         image = random_apply(color_drop, image, p=0.2)
         return image
```

Converting the data into TensorFlow `Dataset` objects

```
def _normalize_img(img, label):
    img = tf.cast(img, tf.float32) / 255.
    return img #(img, label)

ssl_ds_one = ssl_ds_one.map(_normalize_img)

ssl_ds_two = ssl_ds_two.map(_normalize_img)
```

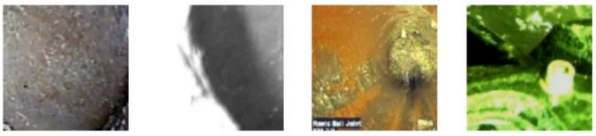
```
[ ] ssl_ds_one = (
        ssl_ds_one.shuffle(1024, seed=SEED)
        .map(custom_augment, num_parallel_calls=AUTO)
        .batch(BATCH_SIZE)
        .prefetch(AUTO)
    )

    ssl_ds_two = (
        ssl_ds_two.shuffle(1024, seed=SEED)
        .map(custom_augment, num_parallel_calls=AUTO)
        .batch(BATCH_SIZE)
        .prefetch(AUTO)
    )
```

```
[ ] # Visualize a few augmented images.
    sample_images_one = next(iter(ssl_ds_one))
    plt.figure(figsize=(10, 10))
    for n in range(4):
        ax = plt.subplot(5, 5, n + 1)
        plt.imshow(sample_images_one[n].numpy() )#.astype("int"))
        plt.axis("off")
    plt.show()
```

```
# Ensure that the different versions of the dataset actually contain
# identical images.
sample_images_two = next(iter(ssl_ds_two))
plt.figure(figsize=(10, 10))
for n in range(4):
    ax = plt.subplot(5, 5, n + 1)
    plt.imshow(sample_images_two[n].numpy() ) #.astype("int"))
    plt.axis("off")
plt.show()
```

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255]
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255]
```


```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255]
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255]
```



Defining the encoder and the predictor

```
[ ] !wget https://raw.githubusercontent.com/GoogleCloudPlatform/keras-idiomatic-programmer/master/zoo/resnet/resnet_cifar10_v2.py
```

```
[ ] import tensorflow as tf
    tf.keras.backend.clear_session()  # For easy reset of notebook state.
    from tensorflow.keras import layers
```

```
def get_encoder():
    extractor = tf.keras.applications.MobileNet(
            input_shape=(CROP_TO, CROP_TO, 3),
            alpha=0.25,
            include_top=False,
            weights='imagenet' )
    inputs = extractor.output
    x = tf.keras.layers.GlobalAveragePooling2D()(inputs)
    # Projection head.
    x = layers.Dense(
        PROJECT_DIM, use_bias=False, kernel_regularizer=regularizers.l2(WEIGHT_DECAY)
    )(x)
    x = layers.BatchNormalization()(x)
    x = layers.ReLU()(x)
    x = layers.Dense(
        PROJECT_DIM, use_bias=False, kernel_regularizer=regularizers.l2(WEIGHT_DECAY)
    )(x)
    outputs = layers.BatchNormalization()(x)
    return tf.keras.Model(extractor.input, outputs, name="encoder")
```

```
def get_predictor():
    model = tf.keras.Sequential(
        [
            layers.Input((PROJECT_DIM,)),
            layers.Dense(
                LATENT_DIM,
                use_bias=False,
                kernel_regularizer=regularizers.l2(WEIGHT_DECAY),
            ),
            layers.ReLU(),
            layers.BatchNormalization(),
            layers.Dense(PROJECT_DIM),
        ],
        name="predictor",
    )
    return model
```

Defining the (pre-)training loop

```
[ ]
def compute_loss(p, z):
    z = tf.stop_gradient(z)
    p = tf.math.l2_normalize(p, axis=1)
    z = tf.math.l2_normalize(z, axis=1)
    return -tf.reduce_mean(tf.reduce_sum((p * z), axis=1))
```

```
class SimSiam(tf.keras.Model):
    def __init__(self, encoder, predictor):
        super(SimSiam, self).__init__()
        self.encoder = encoder
        self.predictor = predictor
        self.loss_tracker = tf.keras.metrics.Mean(name="loss")

    @property
    def metrics(self):
        return [self.loss_tracker]

    def train_step(self, data):
        # Unpack the data.
        ds_one, ds_two = data

        # Forward pass through the encoder and predictor.
        with tf.GradientTape() as tape:
            z1, z2 = self.encoder(ds_one), self.encoder(ds_two)
            p1, p2 = self.predictor(z1), self.predictor(z2)
            loss = compute_loss(p1, z2) / 2 + compute_loss(p2, z1) / 2
```

```
            # Compute gradients and update the parameters.
            learnable_params = (
                self.encoder.trainable_variables + self.predictor.trainable_variables
            )
            gradients = tape.gradient(loss, learnable_params)
            self.optimizer.apply_gradients(zip(gradients, learnable_params))

            # Monitor loss.
            self.loss_tracker.update_state(loss)
            return {"loss": self.loss_tracker.result()}
```

Adding noises to images

```
[ ]  import cv2

     def sp_noise(image,prob):

         output = np.zeros(image.shape,np.uint8)
         thres = 1 - prob
         for i in range(image.shape[0]):
             for j in range(image.shape[1]):
                 rdn = random.random()
                 if rdn < prob:
                     output[i][j] = 0
                 elif rdn > thres:
                     output[i][j] = 255
                 else:
                     output[i][j] = image[i][j]
         return output

     image = cv2.imread('image.jpg',0)
     noise_img = sp_noise(image,0.05)
     cv2.imwrite('sp_noise.jpg', noise_img)
```

Networks pre-training

```
num_training_samples = tf.data.experimental.cardinality(ssl_ds_one).numpy()

steps = EPOCHS * (num_training_samples // BATCH_SIZE)
lr_decayed_fn = tf.keras.experimental.CosineDecay(
    initial_learning_rate=0.03, decay_steps=steps
)

# Create an early stopping callback.
early_stopping = tf.keras.callbacks.EarlyStopping(
    monitor="loss", patience=5, restore_best_weights=True
)

# Compile model and start training.
simsiam = SimSiam(get_encoder(), get_predictor())

simsiam.compile(optimizer=tf.keras.optimizers.SGD(lr_decayed_fn, momentum=0.6))

history = simsiam.fit(ssl_ds, epochs=EPOCHS, callbacks=[early_stopping])

# Visualize the training progress of the model.
plt.plot(history.history["loss"])
plt.grid()
plt.title("Negative Cosine Similairty")
plt.show()
```

Evaluating SSL method

```
[ ] train_ds = builder.as_dataset(split='train', shuffle_files=True, as_supervised=True)
    test_ds = builder.as_dataset(split='train', shuffle_files=True, as_supervised=True)

    def _normalize_img(img, label):
        img = tf.cast(img, tf.float32) / 255.
        img = tf.image.resize(img, [IMG_SIZE, IMG_SIZE])
        return (img, label)

    train_ds = train_ds.map(_normalize_img)
    test_ds = test_ds.map(_normalize_img)
```

```
train_ds = (
    train_ds.shuffle(1024)
    .map(lambda x, y: (flip_random_crop(x), y), num_parallel_calls=AUTO)
    .batch(BATCH_SIZE)
    .prefetch(AUTO)
)
test_ds = test_ds.batch(BATCH_SIZE).prefetch(AUTO)

backbone = tf.keras.Model(
    simsiam.encoder.input, simsiam.encoder.output
)

backbone.trainable = False
inputs = layers.Input((CROP_TO, CROP_TO, 3))
x = backbone(inputs, training=False)
outputs = layers.Dense(12, activation="softmax")(x)
linear_model = tf.keras.Model(inputs, outputs, name="linear_model")

# Compile model and start training.
linear_model.compile(
    loss="sparse_categorical_crossentropy",
    metrics=["accuracy"],
    optimizer=tf.keras.optimizers.SGD(lr_decayed_fn, momentum=0.9),
)
history = linear_model.fit(
    train_ds, validation_data=test_ds, epochs=EPOCHS, callbacks=[early_stopping]
)
_, test_acc = linear_model.evaluate(test_ds)
print("Test accuracy: {:.2f}%".format(test_acc * 100))
```