Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп’ютерної інженерії та управління
Кафедра Комп’ютерних інтелектуальних технологій та систем

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

рівень вищої освіти другий (магістерський)

Інтелектуальна система моніторингу та прогнозування стану обладнання
(тема)

Виконав:
студент II курсу, групи КІТм-22-1
Віталій ПУШКО
(власне ім'я, прізвище)
Спеціальність 123 Комп’ютерна інженерія
(код і повна назва спеціальності)
Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма Комп'ютерні інтелектуальні технології
(повна назва освітньої програми)

Керівник проф. Олександр БЕЗСОНОВ
(посада, власне ім'я, прізвище)

Допускається до захисту

Зав. кафедри (підпис) Олег РУДЕНКО (власне ім'я, прізвище)

2023 р.

Харківський національний університет радіоелектроніки

Факультет	Комп’ютерної інженерії та управління
Кафедра	Комп’ютерних інтелектуальних технологій та систем
Рівень вищої освіти	другий (магістерський)
Спеціальність	123 Комп’ютерна інженерія
Тип програми	освітньо-професійна
Освітня програма	Комп’ютерні інтелектуальні технології

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)

«_____»_____2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Пушко Віталію Валерійовичу
(прізвище, ініціали)

1. Тема роботи (проекту) Інтелектуальна система моніторингу та прогнозування стану обладнання

затверджена наказом університету від « 03 » листопада 2023 р. № 1290ст

2. Термін подання студентом роботи до екзаменаційної комісії « 13 » січня 2024 р.

3. Вихідні дані до роботи (проекту)

Набір даних IMS NASA «Bearings»

Мова програмування Python

4. Перелік питань, що потрібно опрацювати в роботі

Аналіз предметної області та постановка задачі

Аналіз вхідних даних

Опис методу вилучення ознак перехідних процесів з сигналів віброакустичних сенсорів

Тестування моделей штучних нейронних мереж. Аналіз результатів

Розробка прототипу системи моніторингу

Опис програмної частини системи

Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням кафедри) _____

Презентація 11 слайдів

6. Консультанти розділів роботи (п.6 включається до завдання за наявністю консультантів згідно до наказу, зазначеному у п.1)

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

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної області та актуальності теми роботи	6.11 - 10.11.2023	виконано
2	Огляд сучасної літератури за напрямком дослідження	11.11 - 17.11.2023	виконано
3	Постановка задачі дослідження	18.11.2023	виконано
4	Аналіз вхідних даних з набору IMS NASA Bearings	19.11 - 23.11.2023	виконано
5	Вилучення ознак перехідних процесів у вхідних даних	24.11 - 27.11.2023	виконано
6	Огляд моделей штучних нейронних мереж	27.11-30.11.2023	виконано
7	Підготовка даних, навчання та аналіз отриманих результатів роботи моделей штучних нейронних мереж	1.12-8.12.2023	виконано
8	Розробка структури системи моніторингу	9.12 - 15.12.2023	виконано
9	Розробка програмного забезпечення системи моніторингу	16.12 - 22.12.23	виконано
10	Оформлення пояснювальної записки	23.12 - 31.12.23	виконано
11	Оформлення презентації	1.01- 4.01.2024	виконано

Дата видачі завдання « 06 » листопада 2023 р.

Студент _____
(підпис)

Керівник роботи _____ _____
(підпис) (посада, ініціали, прізвище)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 100 сторінок, 32 рисунки, 2 таблиці, 2 додатки, 16 джерел.

ШТУЧНИЙ ІНТЕЛЕКТ, КЛАСИФІКАЦІЯ, ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ, РЕКУРЕНТНІ НЕЙРОННІ МЕРЕЖІ, ОБРОБКА ДАНИХ, ІНТЕЛЕКТУАЛЬНА СИСТЕМА, МУЛЬТИАГЕНТНА СИСТЕМА, ПРЕДИКТИВНЕ ОБСЛУГОВУВАННЯ.

Мета роботи – дослідження методу вилучення ознак перехідних процесів з гармонічних сигналів, створення прототипу системи моніторингу та прогнозування стану обладнання шляхом аналізу інформації яка надходить з різноманітних сенсорів.

Об’єкт дослідження – закономірності в гармонічних сигналах з сенсорів.

Предмет дослідження – використання нейронних мереж для визначення поточного стану та рівня зношеності обладнання на основі даних які надходять з вібраційних та акустичних сенсорів.

В ході роботи розроблено прототип системи моніторингу стану та прогнозування рівня зношеності обладнання за допомогою нейронних мереж. Було досліджено доцільність використання різних архітектур нейронних мереж. Проаналізовано характеристики сигналів та запропоновано метод пошуку та вилучення ознак перехідних процесів.

ABSTRACT

Explanatory note of master's qualification work: 100 pages, 32 figures, 2 tables, 2 appendices, 16 sources.

ARTIFICIAL INTELLIGENCE, CLASSIFICATION, CONVOLUTIONAL NEURAL METERINGS, RECURRENT NEURAL METERINGS, DATA PROCESSING, INTELLIGENT SYSTEM, MULTI-AGENT SYSTEM, PREDICTIVE MAINTENANCE.

The main purpose of this work is to study the method of extracting signs of transient processes from harmonic signals, creating a prototype of system for monitoring and forecasting the state of equipment by analyzing information received from various sensors.

The object of research is the patterns in harmonic signals from sensors.

The subject of the research is the usage of neural networks to determine the current state and level of wear of equipment based on data received from vibration and acoustic sensors.

In the course of the work, a prototype of the system for condition monitoring and forecasting the level of wear of equipment was developed using neural networks. The expediency of using various neural network architectures was investigated. The characteristics of the signals are analyzed and a method for searching and identifying signs of transient processes is proposed.

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп’ютерної інженерії та управління
Кафедра Комп’ютерних інтелектуальних технологій та систем

АНОТАЦІЯ

КВАЛІФІКАЦІЙНОЇ РОБОТИ

рівень вищої освіти другий (магістерський)

Інтелектуальна система моніторингу та прогнозування стану обладнання
(тема)

Виконав:
студент II курсу, групи КІТм-22-1
Віталій ПУШКО
(власне ім'я, прізвище)

Спеціальність 123 Комп’ютерна інженерія
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Комп'ютерні інтелектуальні технології
(повна назва освітньої програми)

Керівник проф. Олександр БЕЗСОНОВ
(посада, власне ім'я, прізвище)

2023 р.

АНОТАЦІЯ

Пушко В. В. Інтелектуальна система моніторингу та прогнозування стану обладнання.

Актуальність теми дослідження.

Четверта індустріальна революція, або Industry 4.0 змінює підходи в роботі промислового бізнесу. Сучасні виробництва стають все більш автоматизованими та характеризуються значно меншим залучанням людини у виробничі процеси. Методологія Industry 4.0 означає активне залучання інтелектуальних систем у процеси контролю виробництва з високим рівнем роботизації та автоматизації і не тільки. Такі системи дозволяють збирати, зберігати та аналізувати дані з сенсорів встановлених на обладнанні. Важливою інформацією є поточний стан роботи обладнання та рівень його зносу. Аналіз накопичених даних робить можливим впровадження більш гнучкого підходу до планування обслуговування обладнання. Такий підхід має назву предиктивного або прогнозного.

Концепція предиктивного обслуговування обладнання направлена за зменшення витрат на обслуговування та підвищення ефективності за рахунок більш точного планування, оптимізації та попередження простоїв. Основна мета в пошуку тієї точки часу, в якій показники роботи обладнання починають падати та планування технічного обслуговування в той момент, коли це найбільш рентабельне. Системи предиктивного обслуговування дають можливість будувати реалістичні прогнози і на підставі чіткого розрахунку керувати всім виробничим процесом. Організації, які успішно впроваджують передиктивне обслуговування, можуть отримати конкурентну перевагу, оскільки вони здатні надавати надійніші та ефективніші послуги чи продукти.

Одним з типів даних які можуть свідчити про стан обладнання є дані з таких вібраційних та акустичних сенсорів. Сучасний розвиток технологій штучного інтелекту, зокрема нейронних мереж глибинного навчання, дозволяє за їх допомогою аналізувати таки типі даних. А саме, постає питання визначення

поточного стану обладнання та моніторинг перехідних процесів на основі даних отриманих з сенсорів у реальному часі або збережених.

Метою даної роботи є дослідження задачі класифікації даних отриманих з акустичних та вібраційних сенсорів за допомогою штучних нейронних мереж. Створення прототипу програмно-апаратної системи моніторингу поточного стану обладнання.

Об’єктом дослідження є нейронна мережа для класифікації стану обладнання на основі даних с сенсорів.

Предметом дослідження є аналіз вхідних даних, методи отримання унікальних властивостей з даних які характеризують поточний стан, пошук закономірностей в даних які свідчать про наявність перехідних процесів в роботі обладнання. Використання різних типів нейронних мереж. Аналіз отриманих результатів для отримання оптимальної моделі нейронної мережі.

В роботі розглядається методика пошуку специфічних ознак у вхідних даних. Розглядаються частото часові характеристики вхідних даних. Враховується те що вхідні дані являють собою послідовність у якій ознаки можуть змінюватись у часі та попередні дані мають кореляцію з наступними. Шляхом аналізу спектрограм отриманих з вибірок даних визначаються смуги частот в яких інформація найбільш схильна до змін.

У якості вхідного набору даних для опрацювання методики використовується набір даних «NASA Bearings Dataset» наданий Центром Інтелектуального Обслуговування Систем (IMS) Університету Цинциннаті. Набір містить дані які були отримані з акселерометрів під час навантажувальних випробувань підшипників. Набір даних має велику цінність завдяки добре описаному ходу та результатам випробувань. З набору можна вилучити характерні ознаки роботи підшипника на початку, прослідкувати зміну ознаки впродовж та наприкінці випробування.

Підхід до первинного аналізу вхідних даних полягає в оцінюванні спектральної щільності потужності або спектральної густини як окремих частин так і набору даних загалом. Пошук ознак перехідних процесів та аномалій. За

результатами первинного аналізу формуються смуги частот в яких спостерігається мінливість в даних. Основна мета первинного аналізу це підготовка та фільтрація вхідних даних для подальшого формування окремого набора даних для навчання та тестування моделей нейронних мереж.

Отримання набору даних для навчання нейронних мереж відбувається шляхом застосування механізму «ковзного вікна». Вхідні дані розбиваються на вікна з використанням перекриття. Для кожного вікна отримується спектрограма з використанням короткочасного перетворення Фур'є. Отримана спектрограма додатково масштабується до шкали Mel, і тільки для смуг частот які були отримані при попередньому аналізі. Параметри підбираються таким чином щоб сформувати оптимальну розмірність вхідних даних для роботи нейронної мережі. Велика розмірність призводить до значного обчислювального навантаження, низька до втрати важливих ознак. Тому треба дотримуватись балансу.

Розмічення отриманого набору даних для тренування нейронної мережі залежить від типу вхідних даних. За результатами первинного аналізу набір даних розбивається на інтервали де спостерігається штатна робота та на інтервали де спостерігається присутність несправності. Залежно від цього формується два окремі набори: інтервальний та помилковий. Інтервальний набір містить ознаки які повільно змінюються у часі, такі що можуть свідчити про рівень зносу. Помилковий набір містить ознаки які характеризують стан несправності та зазвичай швидко змінюються у часі. Мітка вибірки даних має або номер класу інтервалу або номер класу помилки. Отримані розмічені набори даних приймають участь тренуванні двох екземплярів нейронної мережі. Однієї для формування прогнозу інтервалу роботі, другої для формування прогнозу помилки яка настає або вже настала.

В роботі проаналізовано роботу трьох моделей нейронних мереж. Однієї згорткової, та двох згортково-рекурентних. Зроблена оцінка якості моделей по різним показникам. Отримані результати занесені в таблицю. Зроблено висновки згідно яких кожна модель має певні переваги та недоліки. Згортково-

рекурентні демонструють більшу якість ніж згорткова але потребують значно більше часу на обчислення.

В роботі запропоновано прототип системи моніторингу обладнання. Нейронна мережа є інтелектуальним компонентом системи. Інші компоненти включать в себе агенти збору даних, сервер обробки даних, веб-інтерфейс користувача який може бути частиною системи підтримки прийняття рішень.

Агенти системи відповідають за збір даних с сенсорів та представлення цих даних на власній веб сторінці з якої сервер може отримувати певні дані, за потребою. Для передачі аудіо даних в реальному часі запропоновано використовувати протокол RTP. У якості агентів в прототипі системи виступали SBC Raspberry Pi.

Сервер системи складається з наступних компонентів:

– база даних. Містить інформацію про агентів, результати прогнозу нейронної мережі, дані отримані від додаткових сенсорів, часову мітку;

– логіка нейронної мережі. Відповідає за формування прогнозів шляхом опрацювання даних які надходять з сенсорів у реальному часі;

– веб сторінка моніторингу обладнання. На веб сторінці сервера відображається інформація про прогноз поточного рівня зносу обладнання, попередження у разі виявлення ознак присутності несправності, код несправності, інформацію з додаткових сенсорів.

Програмна частина прототипу системи розроблена на мові програмування Python. Вибір цієї мови обумовлений наявністю достатньої кількості бібліотек для роботи з даними, нейронними мережами, базою даних та організації веб сервера. Було задіяно бібліотеки TensorFlow, Librosa, Flask, SQLAlchemy. У якості бази даних була обрана компактна вбудована СУБД SQLite. Вибір типу бази даних обумовлений кросс-платформеністю, автономністю та відкритістю. Реалізація веб сервера виконана за рахунок функціонала бібліотеки Flask.

Досліджена в роботі методика попередньої обробки даних з сенсорів та використання нейронних мереж для обробки цих даних виявилась цілком оправданою. Запропонована система може застосовуватись як самостійна

частина так і бути додатковим компонентом існуючої інтелектуальної системи обслуговування обладнання.

ШТУЧНИЙ ІНТЕЛЕКТ, КЛАСИФІКАЦІЯ, ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ, РЕКУРЕНТНІ НЕЙРОННІ МЕРЕЖІ, ОБРОБКА ДАНИХ, ІНТЕЛЕКТУАЛЬНА СИСТЕМА, МУЛЬТИАГЕНТНА СИСТЕМА, ПРЕДИКТИВНЕ ОБСЛУГОВУВАННЯ.

Публікації здобувача за темою роботи:

1. Сердюк Н., Пушко В. Проблематика селекції та класифікації аудіо подій у зашумленому середовищі. *«Проблеми Інформатизації».* Тези доповідей десятої міжнародної науково-технічної конференції. м. Харків, 24-25 листопада 2022. С.5

2. Аксак Н., Пушко В. Використання нейропроцессорів у вбудованих системах. *«Тренди та перспективи розвитку мультидисциплінарних досліджень».* Матеріали II міжнародної студентської наукової конференції. м. Хмельницький, 25 листопада 2022. С.13

3. Аксак Н., Пушко В. Багатоагентні системи в Industry 4.0. Матеріали 27-го Міжнародного молодіжного форуму «Радіоелектроніка та молодь у XXI столітті»: тези доповідей, 10-12 травня 2023 р. – Харків: ХНУРЕ, 2023. Т 5.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ І ТЕРМІНІВ

ШІ - Штучний інтелект

ШНМ - Штучна нейронна мережа

АЦП - Аналогово цифровий перетворювач

AI - Artificial Inteligence

ANN - Artificial Neural Network

CNN - (Convolutional Neural Network) Згорткова нейронна мережа

RNN - (Reccurent Neural Network) Рекурентна нейронна мережа

CRNN - (Convolutional Recurrent Neural Networks) Згортково-рекурентна нейрона мережа.

CPU - центральний процесор (Central Processing Unit)

RAM - Random Access Memory

SBC - Small Board Computer

PCM - Pulse Code Modulation

FFT - Fast Fourier Transform

STFT - Short-time Fourier Transform

DAC - Digital to analog converter

PdM - Predictive Maintenance

IoT - Internet of things

DL – Deep learning

DNN – Deep Neural Network

ВСТУП

В умовах автоматизації та роботизації виробничих процесів прогнозування відмов та перманентний моніторинг стану обладнання відіграють вирішальну роль у підвищенні ефективності виробництва. Запобігання непередбаченим відмовам, скорочення часу простоїв та зниження витрат на обслуговування та ремонт це основна мета цифровізації процесів обслуговування обладнання.

Одним із перспективніших напрямів у галузі технічного обслуговування обладнання є передиктивне обслуговування (Predictive Maintenance). Його основною концепцією є виявлення потенційних проблем з обладнанням шляхом аналізу даних із різних датчиків, забезпечуючи надійність критично важливих виробничих та технологічних процесів для діяльності підприємства. Удосконалення та автоматизація процесів технічного обслуговування з залучанням елементів штучного інтелекту також стають частиною цієї концепції.

В останні роки використання штучного інтелекту та штучних нейронних мереж в процесах аналізу великих об’ємів даних стало популярним напрямом досліджень. Існуючі методи контролю стану обладнання часто ґрунтуються на моніторингу параметрів роботи, таких як температура, тиск та вібрація. Використання звуку роботи обладнання як джерела даних може бути корисним доповненням завдяки більшій розмірності прихованих ознак які можна вилучити та класифікувати за допомогою нейронних мереж. Переваги використання штучного інтелекту полягають в тому що він дозволяє автоматизувати процес аналізу даних з сенсорів для своєчасного виявлення потенційних проблем, забезпечуючи більш точні та ефективні результати, а також значно зменшує час, необхідний для аналізу та обробки даних в цілому.

Метою даної роботи є дослідження методу обробки та класифікації даних з акустичних та вібраційних сенсорів за допомогою нейронних мереж, розробка прототипу інтелектуальної системи для моніторингу та прогнозування стану обладнання.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз предметної області

В останні роки використання штучного інтелекту процесах цифровізації у промисловості стає дедалі більш поширеним. Поряд з терміном Industry 4.0 стоять такі поняття як «розумна фабрика» (Smart Factory), «розумне виробництво» (Smart Manufacturing), «фабрика майбутнього» (Factory of the Feature). Національний інститут стандартів і технологій США (NIST) визначає термін Smart Manufacturing так: це «повністю інтегровані корпоративні виробничі системи, які здатні в реальному масштабі часу реагувати на мінливі умови виробництва, вимоги мереж поставок і задовольняти потреби клієнтів» [1]. У цьому визначенні головне: «в реальному масштабі часу», тобто максимально оперативно, досягаються названі цілі за рахунок інтенсивного і всеосяжного використання інформаційних технологій і кіберфізичних систем на всіх етапах виробництва продукції та її поставки.

Розвиток технологій Інтернету Речей (IoT) забезпечив можливість налагодження взаємодії між обладнанням. За допомого даних отриманих з різноманітних сенсорів, сучасні виробництва здатні випускати продукцію майже не залучаючи у процес людину. Зростають потоки інформації, які потребують обробки у реальному часі, що в свою чергу вимагає використання методів штучного інтелекту на певних етапах та процесах. Ці величезні обсяги даних, містять дуже корисну інформацію та цінні знання, які можуть підвищити продуктивність виробничих процесів, динаміку системи, а також можуть бути застосовані для підтримки прийняття рішень у кількох сферах, головним чином у технічному обслуговуванні та моніторингу стану обладнання.

Серед видів технічного обслуговування обладнання є наступні [2]:

– реактивний. Напрацювання на відмову.
– превентивний (Prescriptive). Планово-попереджувальних ремонти.

– за станом (Condition-Based). Проведення ремонтів за потребою.

– предиктивний (Predictive Maintenance, PdM). Ремонти за прогнозом.

На рисунку 1.1 представлені залежність вартості ремонту від типу обслуговування.

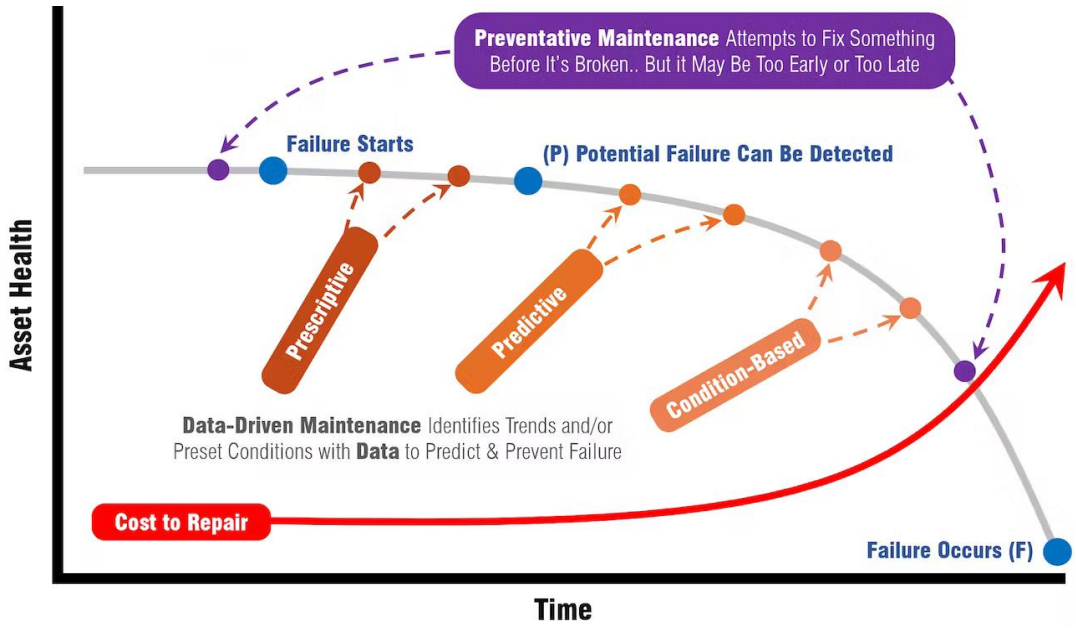

Рисунок 1.1 - Типи обслуговування, вартість ремонту

Предиктивний метод заснований на постійному моніторингу обладнання або машин, як і у методі обслуговування за станом, але використовуються інструменти статистики та прогнозування, щоб визначити коли дії з технічного обслуговування необхідні, тому технічне обслуговування можна запланувати. Крім того, це дозволяє виявляти несправності на ранній стадії на основі історичних даних, використовуючи такі інструменти прогнозування, як методи машинного навчання. Все це потребує аналізу великого обсягу даних які надходять з сенсорів. Такий підхід ще називається «data-driven» - коли основна увага приділяється самим даним, виявлення патернів, залежності та кореляцій в цих даних. Такий підхід до обслуговування доцільно використовувати на розумних виробництвах.

Методи та підходи які забезпечують інформацію про стан обладнання можуть включати такі як вібраційний моніторинг, термометрія, електричні

сигнали, контроль тиску. Однак цих методів не завжди може бути достатньо щоб виявити провісники відмови обладнання. Одними із додаткових а інколи провідними методами можуть бути методи, які використовують штучний інтелект, що базується на аналізі акустичних сцен роботи обладнання. Використання звукових сигналів для прогнозування відмови обладнання має низку переваг, таких як висока швидкість збору даних та можливість моніторингу обладнання в режимі реального часу. Аналіз накопичених даних, у майбутньому, дозволяє проводити роботу з виявлення прихованих закономірностей.

На рисунку 1.2 представлена діаграма типу даних для моніторингу та їх відповідність моделі обслуговування які спостерігаються з ростом зносу обладнання с часом.

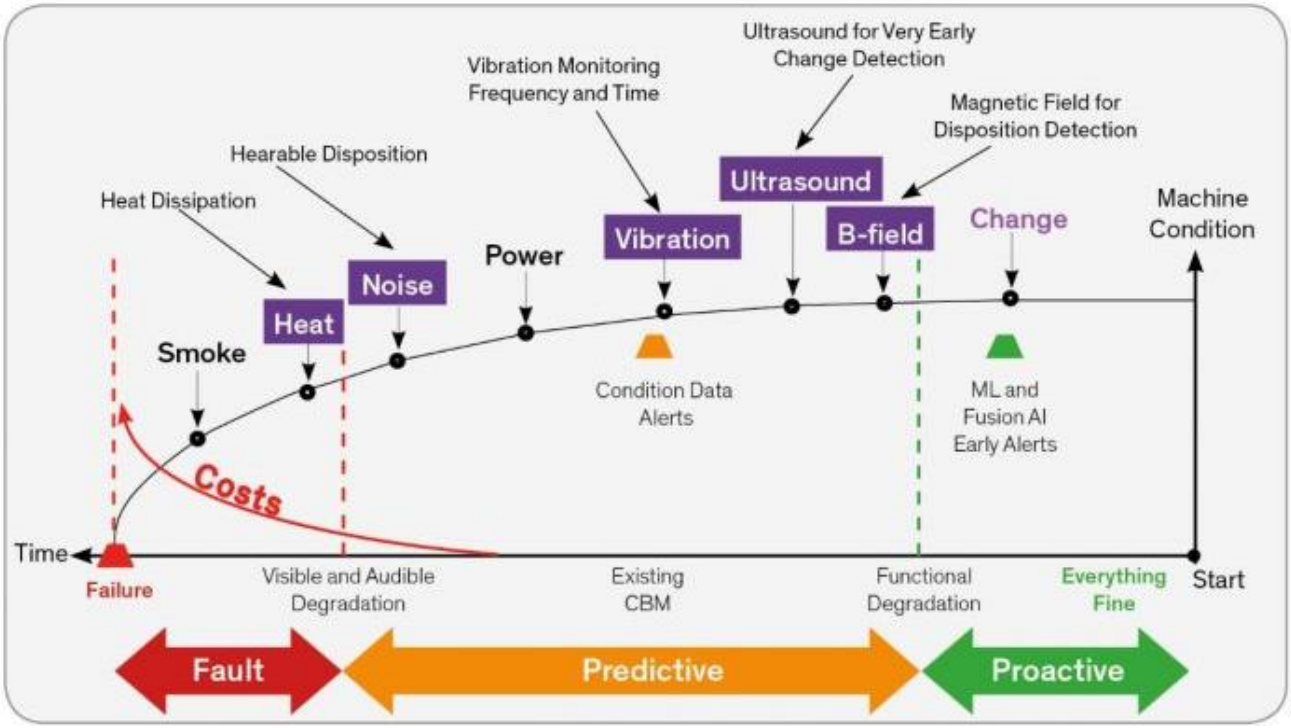

Рисунок 1.2 - Типи даних для моніторингу

Для більшості обладнання де використовуються елементи тертя діаграма є відповідною. Спочатку спостерігається вібрація, потім характерний шум(звук) далі нагрів, дим та відказ в роботі. Треба відмітити що часові відрізки між появою різних ознак нерівномірні. На рисунку 1.3 представлена діаграма

інтервалів появи ознак для підшипників кочення [3]. Весь інтервал від початку отримування ознаки несправності до відказу в роботі називається P-F інтервалом.

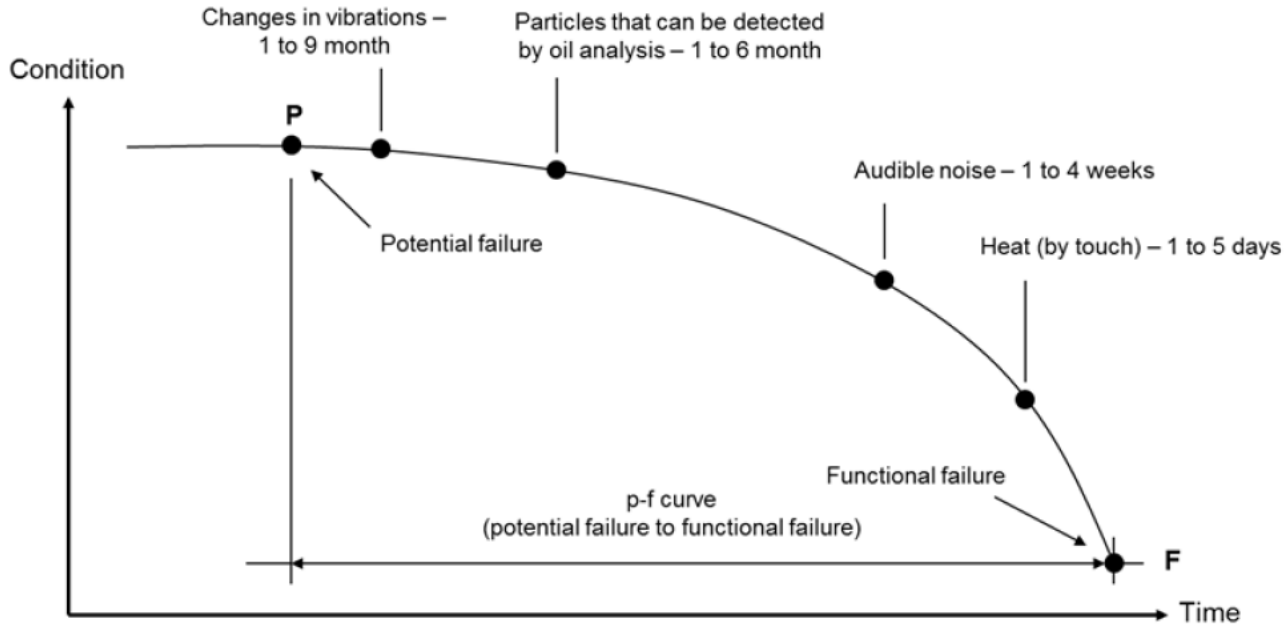

Рисунок 1.3 - P-F інтервал для підшипника кочення

Згідно графіків на рисунку 1.2 - 1.3 звук який сигналізує про несправність в роботі обладнання з’являється набагато пізніше ніж вібрації але не настільки близько до точки відмови ніж показники температури. Шум в роботі підшипника з’являється за 1-4 тижня до відмови, тоді як вібрація з’являється за 1-9 місяців. Якщо це взяти до уваги то стає очевидним що данні отримані з акустичних сенсорів більше підходять на роль основної ознаки для аналізу ніж інші, наприклад вібраційні та температурні.

Існує доволі багато підходів до аналізу звукових даних. Один з таких підходів – використання штучних нейронних мереж (ANN) глибокого навчання (DL) для отримання ознак із звукових даних та класифікації цих ознак для визначення стану обладнання. Цей підхід дозволяє створювати точні та ефективні моделі прогнозування відмов обладнання.

Поєднання висновків отриманих за методом аналізу акустичної сцени з даними отриманими з інших сенсорів дає більш чітку картину стану обладнання.

Мета полягає в тому, щоб мати можливість зібрати якомога більше інформації про стан обладнання в режимі реального часу, використовуючи не тільки дані з різноманітних датчиків а й дані з акустичної сцени роботи обладнання та далі зіставити їх з ефективністю роботи виробництва загалом та, як приклад, швидкістю зношування компонентів. Маючи у своєму розпорядженні інформацією про те, яке обладнання має певний ступіть зносу, відповідно можна запланувати роботи з технічного обслуговування на період, коли вони будуть найбільш рентабельні. Таким чином незаплановані тривалі простої перетворюються в більш короткі планові і час доступності обладнання збільшується.

1.2 Постановка задачі

Основним інструментом в PdM є аналіз даних, зокрема моніторинг параметрів роботи обладнання, таких як температура, вібрації, електричні сигнали, стан змащення, тиск тощо. Зібрані дані аналізуються з використанням алгоритмів машинного навчання та статистичного аналізу з метою виявлення аномалій, які можуть вказувати на початок проблеми з обладнанням.

У даній роботі пропонується розглянути один з підходів до PdM який полягає у класифікації даних отриманих з віброакустичних сенсорів за допомогою штучних нейронних мереж. Це дозволить виявляти аномалії у звуковому сигналі, які можуть вказувати на початок проблем з обладнанням та дає можливість оцінювати рівень зносу.

Враховуючи значний прогрес у використанні ШНМ у таких областях як розпізнаванні зображень, класифікації аудіоподій у навколишньому середовищі та особливі успіхі у створенні речових аналізаторів(голосових асистентів) Google Assistant, Amazon Alexa, Microsoft Cortana, Apple Siri можна зробити висновки що зараз стає цілком можливим аналіз більш спеціалізованих даних, таких як звук роботи обладнання. Також слід відмітити що найбільші хмарні провайдери вже пропонують сервіси для роботи з ШІ у тому числи з використанням

машинного навчання (ML, machine learning). Прикладами таких сервісів може бути Amazon Rekognition, Amazon Connect Voice ID які спеціалізуються на обробці зображень та голосу людини. Але це потребує додаткових зусиль та затрат. Крім того використання хмарних обчислень потребує стабільного зв'язку з мережею інтернет, що в умовах промислового об'єкту може бути неможливим.

Пропонується розробити прототип системи моніторингу та прогнозування стану обладнання з використанням звукових даних роботи обладнання. Розробка системи потребує вирішення наступних супутніх задач:

– попередній аналіз вхідних даних та вилучення ознак. На даному етапі необхідно проаналізувати енергетичні спектри вхідних даних у часовій та частотній області за допомогою швидкого та короткочасного перетворення Фур'є та елементів описової статистики. Враховуючи те що звукові дані можуть містити багато зайвої інформації потрібно визначити смуги частот та які містять ознаки перехідних процесів роботи обладнання;

– перетворення вхідних даних та формування вибірок для навчання та тренування нейронної мережі. На цьому етапі потрібно створити новий набір даних з вилученням тільки тієї частини вхідних даних які відповідають критеріям які були отримані на попередньому етапі, тим самим забезпечити фільтрацію вхідних даних;

– тестування моделей нейронних мереж. Провести навчання та зробити аналіз якості нейронних мереж. Пропонується провести аналіз якості згорткової та згортково-рекурентних мереж з застосуванням LSTM та GRU шарів;

– розробити програмну реалізацію модулів системи. На цьому етапі потрібно розробити програмні модулі для роботи нейронної мережі, забезпечити взаємозв'язок з базою даних, створити веб сторінку моніторингу та забезпечити логіку оновлення даних;

– описати компоненти системи. Протестувати систему на реальних даних. Описати принцип роботи, компоненти та взаємодію агентів збору даних з сервером обробки та аналізу даних.

У якості вхідного набору даних для опрацювання методики можна використати публічний набір даних з NASA Prognostics Center of Excellence - «Bearings Data Set» [4], зібраний Центром Інтелектуального Обслуговування Систем (IMS) Університету Цинциннаті, далі «NASA Bearings». Набір містить дані які були отримані з акселерометрів під час ресурсних випробувань підшипників. Завдяки добре описаному ходу та результатам випробувань а також чистоті даних використання набору має рацію. Набор містить характерні ознаки зміни рівня зношеності підшипника під час випробувань та трьох типів несправності наприкінці, які сигналізували його вихід з ладу.

Для перевірки роботи системи в умовах наближених до реальних необхідно провести випробування з використанням набору даних отриманих саме з акустичної сцени роботи обладнання. Дані з отримані з акселерометрів, та наприклад мікрофону це окремі випадки вимірювання коливань системи. У випадку акселерометру вимірюється вібрація досліджуваного об’єкта безпосередньо, а у випадку акустики вібрація повітря навколо об’єкта. Обидва випадки можуть використовуватись для отримання даних про стан обладнання.

Програмну частину пропонується реалізувати за допомогою мови програмування Python з використанням бібліотек для роботи з штучними нейронними мережами, базами даних, веб сторінками.

2 ПОПЕРЕДНІЙ АНАЛІЗ ВХІДНИХ ДАНИХ

В якості вхідних даних у даній роботі використовуються показники вібрації отримані з акселерометрів PCB 353B33. Потоки даних можуть надходити як у реальному часі так і бути попередньо записані.

2.1 Характеристики вхідних даних

Вібраційні та акустичні дані поєднує те що вони являються числовими рядами, виміряні в різни проміжки часу. У випадку вібрації кожне окреме значення буде мірою зсуву, прискорення об’єкта, у випадку звука це зміна тиску повітря. В обидвох випадках вимірюється механічні коливання середовища. Аналогові електричні сигнали отримані з вібро або акустичних сенсорів відображають зміну напруги у часі. Щоб обробити такі дані за допомогою машинного навчання, їх спочатку потрібно перетворити в цифровий формат а потім отримати унікальні ознаки. Зміни у навколишньому середовищі спочатку перетворюються на аналогові електричні сигнали, а потім оцифровуються за допомогою аналого-цифрового перетворювача (АЦП)(DAC) (Рис. 2.1).

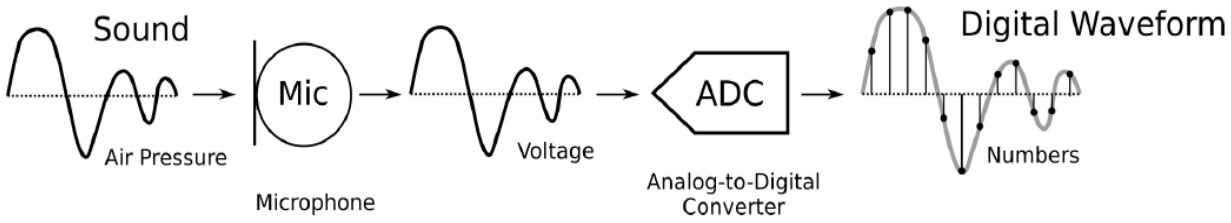

Рисунок 2.1 - Процес оцифрування звуку

Згідно теореми Котельникова-Шенона для того щоб мінімально точно відновити цифрове представлення сигналу з аналогового, то треба, щоб частота дискретизації була мінімум вдвічі більша за максимальну частоту вхідного сигналу.

У процесі оцифрування сигнал квантується в часі на певній частоті дискретизації, а амплітуда квантується на певній бітовій глибині. На рисунку 2.2 зображено отриманий масив даних. Типова частота дискретизації для звукових даних становить 44100 Гц, а бітова глибина – 16 біт, як це використовується у форматі Audio CD. З такими параметрами відбувається захват більшості сприйманої людиною інформації з точки зору акустики. Для даних сенсорів вібрації все залежить вид конкретної моделі сенсора. Наприклад, згідно специфікації [5] для моделі PCB 353B33, за допомогою якого отримувався набір даних «NASA Bearings», максимальний частотний діапазон являє 0.35 – 12кГц.

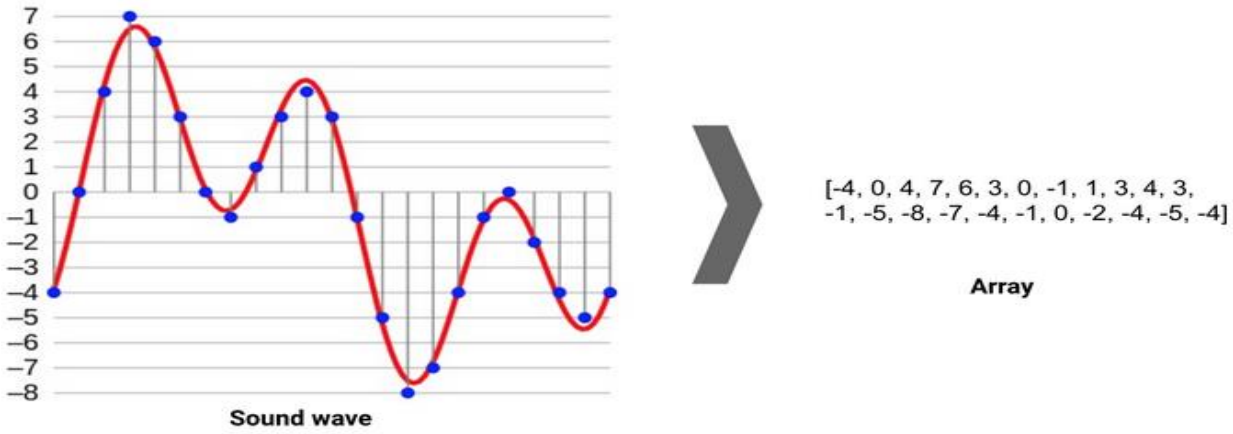

Рисунок 2.2 - Звуковий сигнал як масив даних

У цьому представленні дані є одновимірною послідовністю чисел, чисельним рядом. Такий формат використовувався як вхідний та може перетворюватись далі, у цифровому вигляді. Цифрові дані можна зберігати без стиснення у форматі WAV PCM або з використанням стиснення без втрат - формат FLAC або з використанням стиснення з втратами – формат MP3. Стиснення з втратами видаляє інформацію, яку не розрізняє людське вухо, і може стискатися краще ніж без втрат. Але стиснення з втратами може додати артефакти стиснення, і його краще уникати для завдань машинного навчання. Записи можуть мати кілька каналів аудіо даних, але для навчання ШНМ достатньо одного каналу.

2.2 Аналіз вхідних даних

Для того щоб успішно використовувати методи машинного навчання потрібно зробити попередній аналіз та отримати унікальні ознаки (Feature extraction) з вхідних даних, будь то файл або потік у реальному часі. Отримання унікальних ознак це процес пошуку та вилучення лише важливої інформації з сигналу, що і є ознакою. Використовуючи ці ознаки проводиться ідентифікація певної події у загальній вібро-акустичній сцені.

Серед основних методів оцінки характеристик та отримання ознак з вхідних даних є наступні:

– середньоквадратична енергія, RMSE (Root Mean Square Energy) – міра енергії сигналу, виражена у величинах амплітуди. RMSE є квадратним коренем із середнього квадрата значення амплітуди за певний проміжок часу. Він є одним із найбільш поширених показників рівня гучності звуку. Корисний для оцінки енергії аудіосигналу в конкретному часовому інтервалі;

Обчислюється за формулою:

$$RMS = \sqrt{\frac{1}{N}\sum_{N}|x(n)|^2}, \qquad 2.1$$

Де $x(n)$ є виваженим значенням частоти.

– спектральний центроїд характеризує те, на який частоті сконцентрована енергія сигналу. Для низькочастотних звуків спектральний центроїд буде близьким до 0 Гц, а високочастотних звуків - до частоти дискретизації. Розраховується як зважене середнє частоти кожного із спектральних компонентів сигналу, де вага кожної компоненти дорівнює її амплітуді. Обчислюється за формулою:

$$f_c = \frac{\sum_k S(k)f(k)}{\sum_k S(k)}, \qquad 2.2$$

Де S(k) – спектральна величина елемента k;

f(k) – частота елемента k.

– спектральна ширина. Ширина діапазону частот, у якому зосереджена більшість енергії сигналу. Вона показує, як швидко зменшується спектральна щільність потужності сигналу. Спектральна ширина показує, який діапазон частот включений спектр сигналу і наскільки ці частоти відрізняються від центральної частоти. Визначається як ширина смуги енергії сигналу на половині максимальної точки;

– спектральний спад. Частота на яку припадає певний відсоток загальної суми амплітуд спектра. Зазвичай, спектральний спад визначається як частота, де накопичена енергія становить 85% від загальної енергії спектра. Використовується для оцінки висоти звуку;

– швидкість пересічення нуля. Ознака відображає кількість разів, коли амплітуда аудіо сигналу перетинає горизонтальну вісь (нульове значення амплітуди) за певний проміжок часу. Визначається як відношення числа перетинів сигналу через нуль до довжини аудіо сигналу в секундах. Може вказувати на ступінь монотонності або гладкості звукового сигналу;

– спектрограма масштабована до шкали Mel.

Для оцінки частотно енергетичного вмісту сигналу у часі отримується спектрограма. Найпоширенішим способом обчислення спектрограми з звукового сигналу є використання короткочасного дискретного перетворення Фур'є (STFT, Short-Time Fourier Transform). STFT працює шляхом поділу звуку на короткі послідовні фрагменти, кадри, та обчислення швидкого перетворення Фур'є (FFT, Fast Fourier Transform). Щоб зменшити кількість артефактів на межі фрагментів вони перекриваються (зазвичай на 50%), і перед обчисленням FFT застосовується віконна функція Ханна або Хемінга.

Формула обчислення STFT:

$$X(m,\omega) = \sum_{n} x[n]\omega[n-m]e^{-jwn}, \qquad 2.3$$

Де $X(m, \omega)$ –матриця, спектр сигналу в момент часу m та частоти ω;

$x[n]$ – сигнал;

$\omega[n-m]$ – віконна функція;

e^{-jwn} – число яке представляє значення магнітуди частотного спектра для відповідної частоти та часу в часовому сигналі.

Результати обчислення швидкого перетворення Фур'є для невеликих кадрів сигналу можна масштабувати до шкали Mel (Рис. 2.3) що є компактним представленням спектральних характеристик сигналу та моделюють властивості сприйняття звуку людиною. Хоча в даному випадку вібро-акустичний сигнал не буде сприйматись та оцінюватись людиною, деяке стискання в частотній області яке дає Mel масштабування буде корисним для зменшення розмірності. Особливо вважаючи що цей функціонал вже доступний у бібліотеках Python. Обчислюються за формулою:

$$m = 1125 l n\left(1 + \frac{f}{700}\right), \qquad 2.4$$

Де f- частота вхідного сигналу.

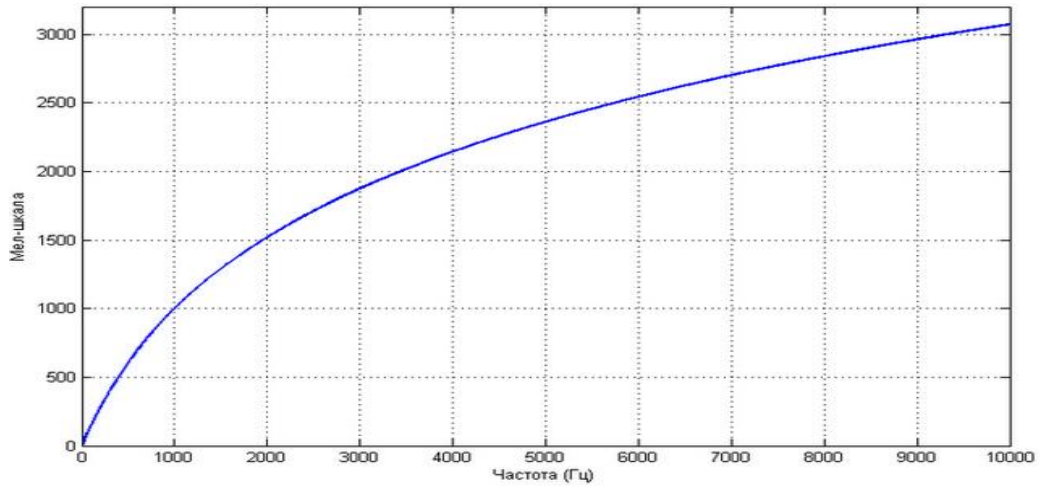

Рисунок - 2.3 Шкала Mel

Існує компроміс між частотною (спектральною) роздільною здатністю та часовою роздільною здатністю отриманою за допомогою STFT. Чим довше вікно FFT, тим краще роздільна здатність по частоті, але тим гірша роздільна здатність

у часі. У більшості випадків типовий вибір довжини вікна становить 20 мс. Подібна довжина кадру часто використовується для аудіо подій. STFT повертає комплексні числа, що описують фазу та величину кожного діапазону частот. Спектрограма обчислюється шляхом зведення в квадрат абсолютної величини та відкидання інформації про фазу. Це називається лінійною спектрограмою або іноді просто спектрограмою. Спектрограма приведена до шкали Mel називається Mel-спектрограмою.

Логарифмічне представлення Mel-спектрограми використовується для зменшення різниці в амплітуді між низькочастотними та високочастотними компонентами, щоб зменшити вплив високочастотного шуму та підвищити динамічний діапазон даних. Логарифмування також скорочує кількість несуттєвої інформації.

2.3 Методика вилучення ознак з вхідного сигналу

В даному підрозділі буде розглянуто методику аналізу та отримання ознак з вхідних даних. У якості вхідних даних буде розглянуто дані отримані вібраційних сенсорів PCB 353B33 під час ресурсних випробувань підшипників Rexnord ZA2115 які надано у наборі даних «NASA Bearings». Будуть зроблені порівняння характеристик сигналу на початку та наприкінці тесту, де було виявлено несправність. Несправність призводить до наростаючих показників вібрації та закінчується виходом підшипника з ладу наприкінці тесту.

На рисунках 2.4-2.8 зображено графіки середньоквадратичної енергії (RMSE), спектрального центроїда. спектральної ширини, спектральний спаду, швидкість пересічення нуля. Розрахунки та візуалізації характеристик і ознак вхідних звукових даних будуть робитись за використанням мови програмування Python та бібліотек Librosa, Matplotlib, Numpy та інших. Довжина запису звукових даних 5 секунд.

Для аналізу записів було розроблено графічну утиліту яка відображає амплітудно частотну спектрограму частини запису – кадру, який складається з

декількох відділків у часі – семплів. Утиліта дозволяє програвати запис сигналу, переміщуватись на певний кадр, задавати розмір для функції швидкого перетворення Фур'є. Результати дослідження запису показано на рисунку 2.9.

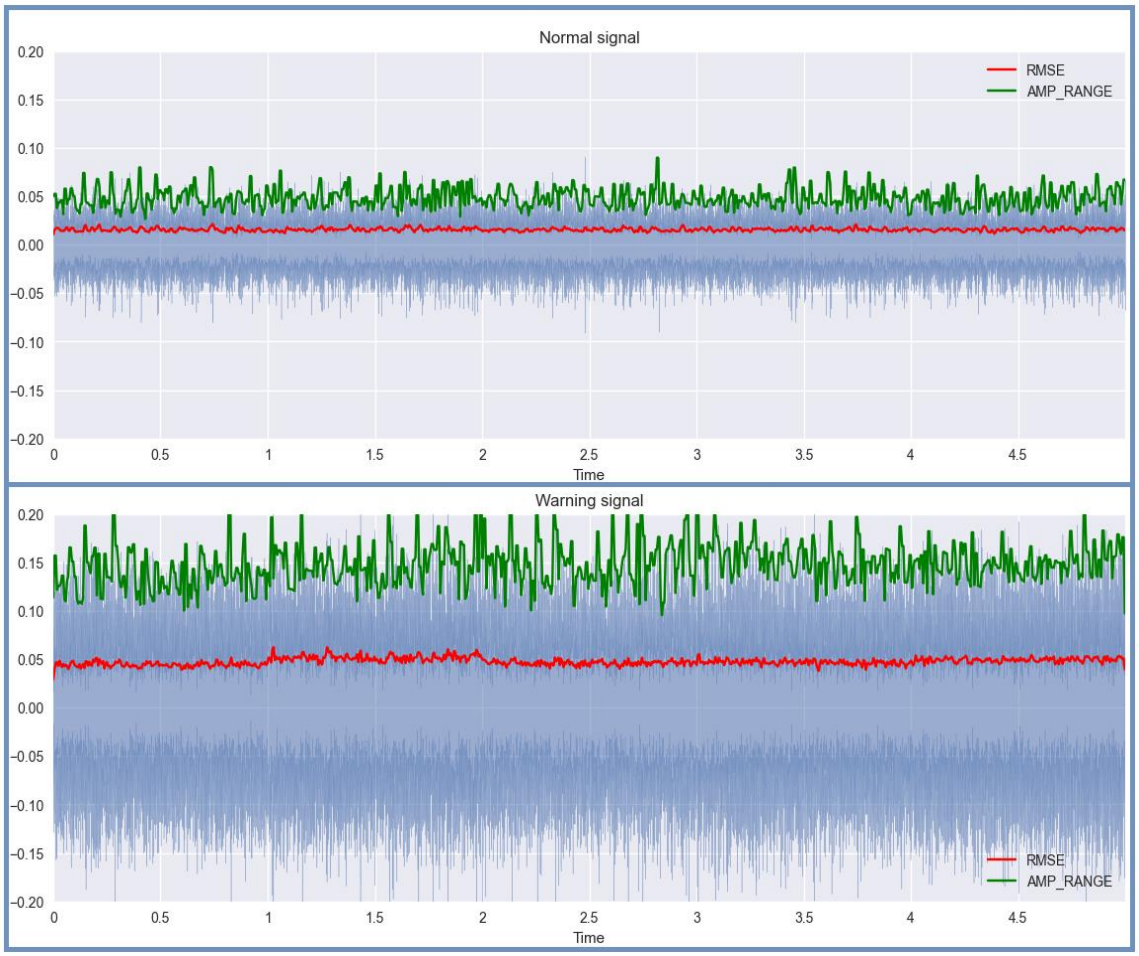

Рисунок 2.4 - Середньоквадратична енергія (RMSE)

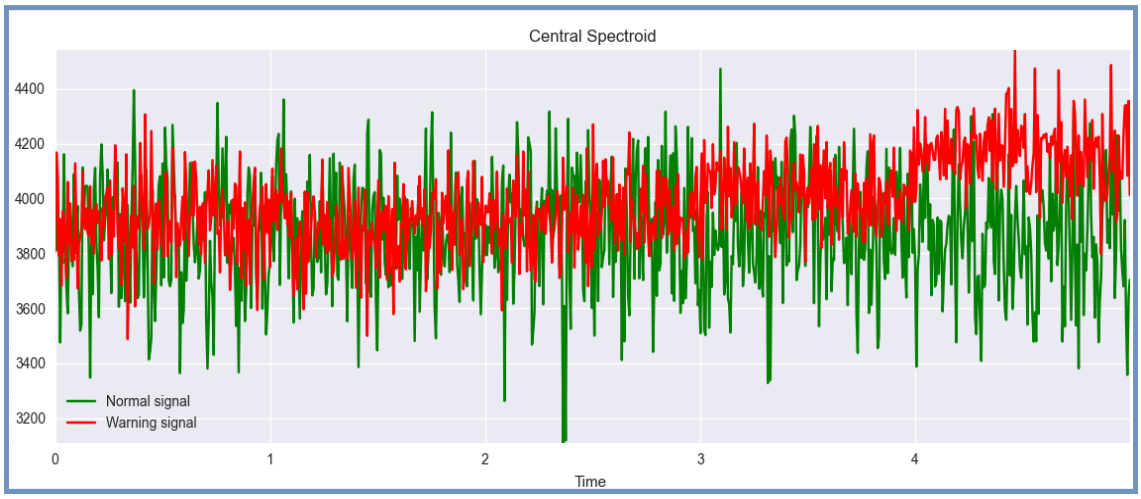

Рисунок 2.5 - Спектральний центроїд

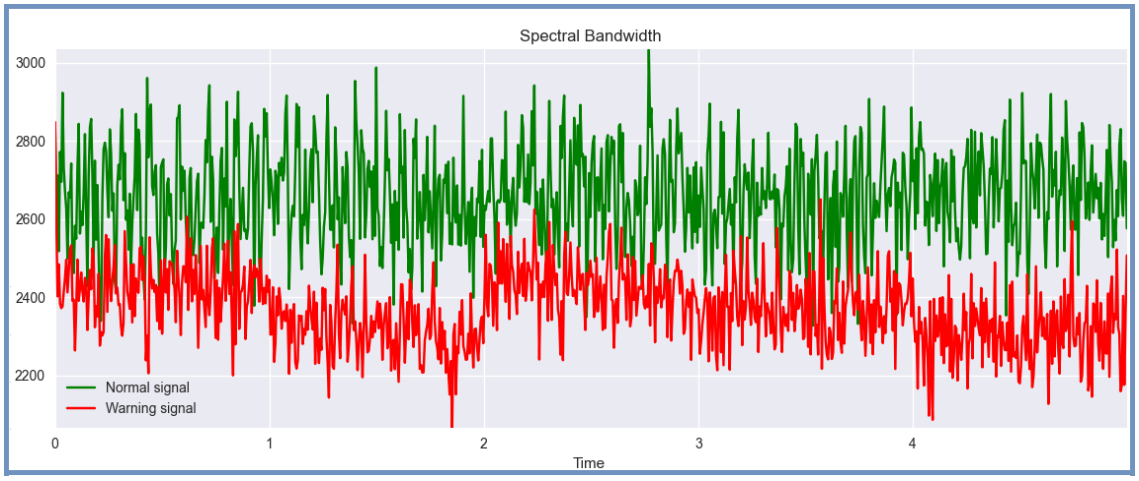

Рисунок 2.6 - Спектральна ширина

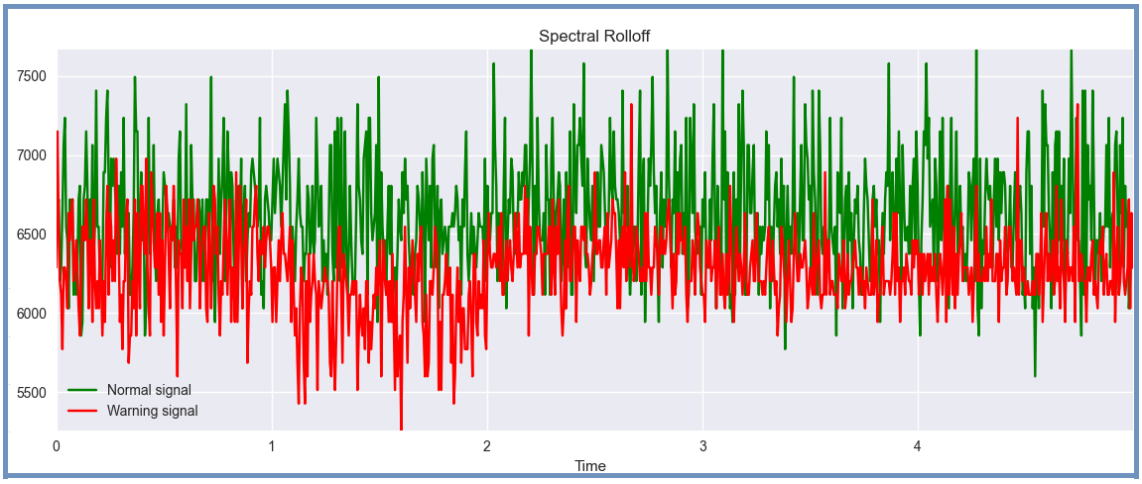

Рисунок 2.7 - Спектральний спад

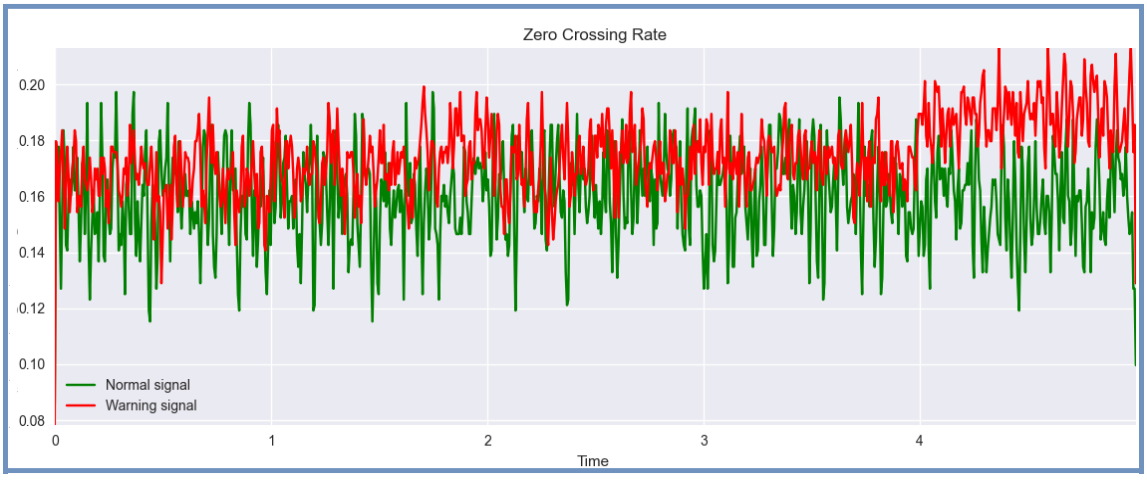

Рисунок 2.8 - Швидкість пересічення нуля

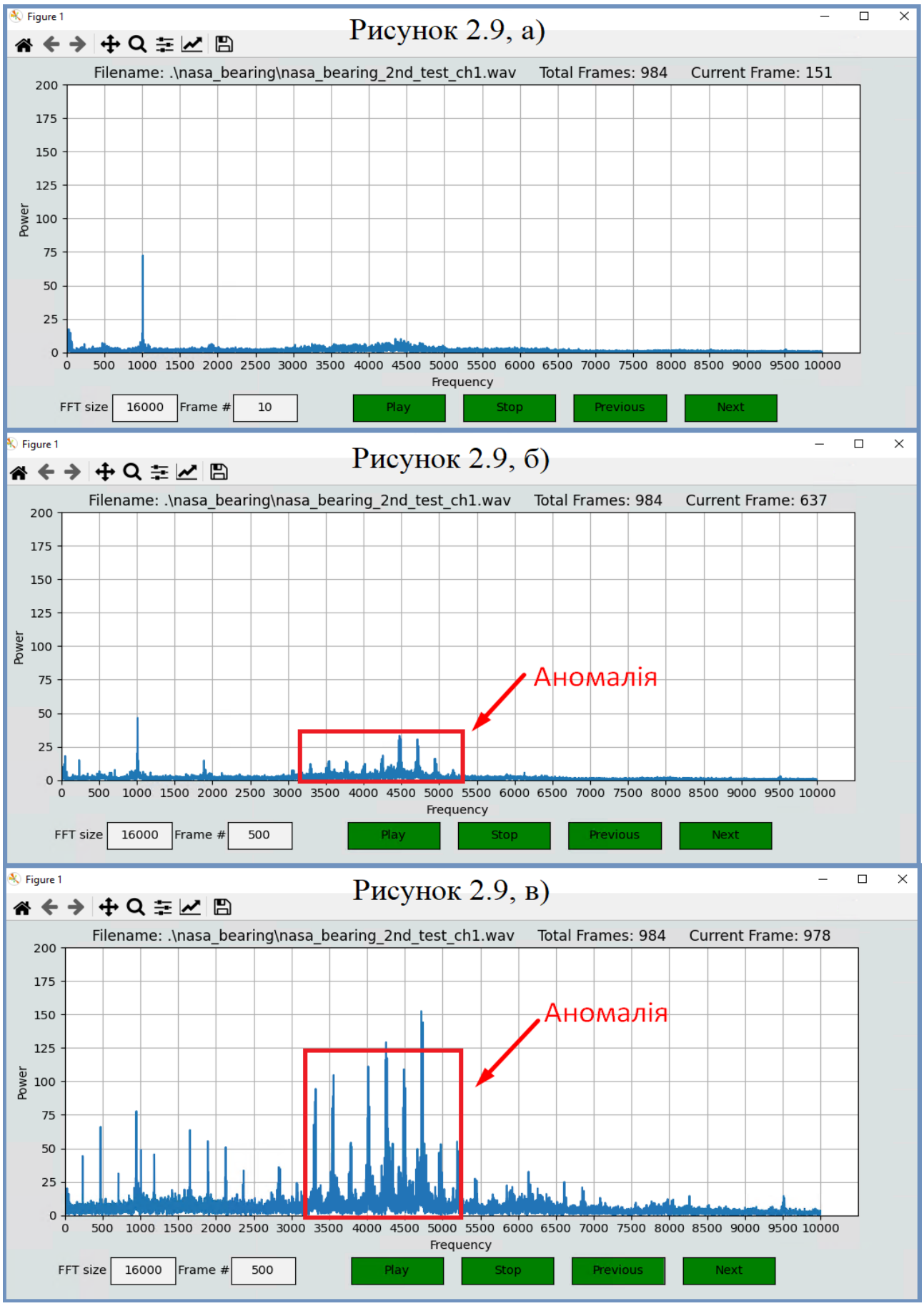

Рисунок 2.9 - Швидке перетворення Фур’є (FFT)

а) – початок тесту; б) – середина тесту; в) – кінець тесту

З вищезазначених графіків можна зробити висновки що сигнали монотонні, основна енергія зосереджена у зоні низьких частот, шумоподібні.

На рисунку 2.9 показані спектрограми швидкого перетворення Фур’є на початку, всередині та наприкінці запису тесту. За допомогою утиліти можна простежити як змінювалась спектрограма з часом. На початку запису тесту (Рис. 2.9а) на спектрограмі присутній тільки один пік на 1000Гц. Приблизно у другий третині тесту (Рис. 2.9б) починає спостерігатись аномалія у проміжку між частотами 3500-5500Гц. Наприкінці запису (Рис. 2.9в) аномалія стає все більш чіткою, що свідчить про наявність деякого перехідного процесу, який можна охарактеризувати як розвитку несправності або зносу підшипника. Згідно записів результатів проведення тесту №2 - наприкінці тесту підшипник вийшов з ладу у зв’язку з руйнуванням зовнішнього кільця.

Для більш повного аналізу вхідних сигналів потрібно побудувати спектрограми короткочасного перетворення Фур’є (Рис. 2.10). Це корисно для аналізу спектральних характеристик сигналу у часі, що дозволяє виявляти, які частоти переважають у певні моменти часу, оцінити динаміку, рівень шумів. Верхня частота спектрограми була обмежена до 8000Гц тому що в процесі аналізу з’ясувалось що сигнали низькочастотні.

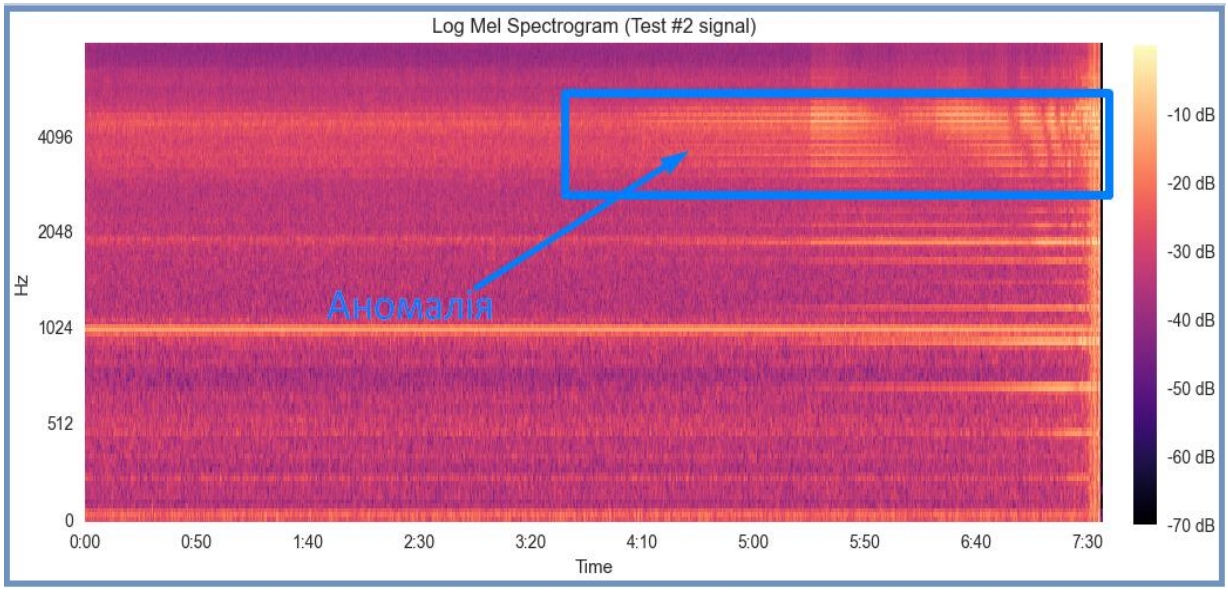

Рисунок 2.10 - Log Mel спектрограма

Шкала частоти приведена до шкали Mel а потужність логарифмована, для більшої компактності. На спектрограмі запису, наприкінці тесту (Рис. 2.10) підтверджується присутність додаткових компонентів в діапазоні 3000-5000Гц . Компоненти простежуються на протязі всього запису, монотонні, низькочастотні, потужність невелика. Також підтверджується присутність монотонного шуму на обох спектрограмах на частоті 1000Гц.

На основі отриманої інформації про характеристики сигналу робиться висновок стосовно того яка частина даних містить ознаки розвитку несправності. В даному випадку, на прикладі тесту №2 набора даних «NASA Bearings», можна зробити висновок що доцільним є використання даних Mel спектрограм з додаванням фільтру частот від 3000Гц до 5000Гц. Саме в цьому діапазоні, за результатом аналізу сигналу, спостерігається поступова мінливість у часі.

3 ОГЛЯД МОДЕЛЕЙ ШТУЧНИХ НЕЙРОННИХ МЕРЕЖ

3.1 Штучні нейронні мережі

Штучні нейронні мережі (ШНМ) - це програмна імплементація нейронних структур нашого мозку. Основним елементом ШНМ є штучний нейрон - персептрон. Обчислювальна модель штучного нейронна на основі математики та алгоритмів, названою пороговою логікою, яка була створена Ворреном Маккалохом та Волтером Піттсом ще у далекому 1943 році. Ця модель поклала шлях до початку досліджень нейронних мереж. Персептрони об’єднуються у шари які мають різні зв’язки між собою, в тому числі можуть мати зворотній зв'язок. Кожен штучний нейрон управляється активаційною функцією, яка має певний поріг. Навчання відбувається через розрахунок оптимальних ваг зв’язків між нейронами. Структуру штучного нейрона зображено на рисунку 3.1 [6].

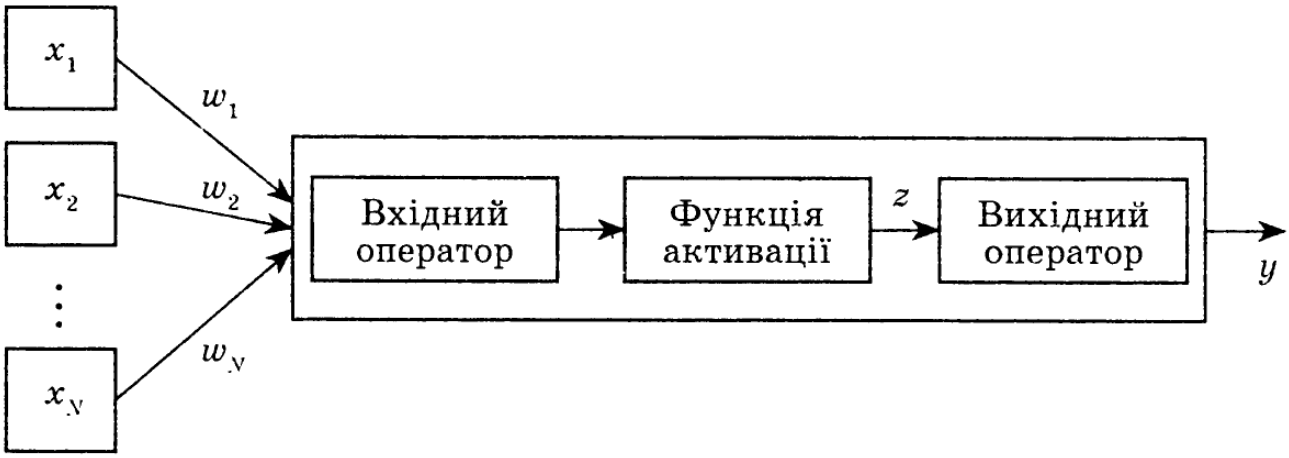

Рисунок 3.1 Структура штучного нейрона

Вхідна функція (оператор) $f_{вх}$ отримує зважені входи та подає їх до функції активації f_a, результат роботи якої може додатково проходити через вихідну функцію $f_{вих}$. Вихідний сигнал нейрона y являє собою результат перетворення вектору вхідних сигналів x та може бути записаний у наступному вигляді:

$$y = f_{\text{вих}}\left(f_{\text{а}}\left(f_{\text{вх}}(x, w)\right)\right), \tag{3.1}$$

А зважені входи $f_{\text{вх}}$ формулою:

$$f(x, w) = \sum_{i=1} w_i x_i\,, \tag{3.2}$$

Де x_i- значення n-го входу нейрона;

w_i - вагові коефіцієнти;

В глибинних нейронних (DNN) мережах використовується поєднання декількох шарів нейронів різної кількості та використання різних функцій активацій. Однією з головних завдань функції активації є додавання нелінійності до обчислення виходів нейронів. Нелінійності дозволяють нейронним мережам краще навчатися складним залежностям у даних. Від правильно підібраних функцій активації напряму залежить швидкість та якість навчання. На рисунку 3.2 представлені популярні функції активації.

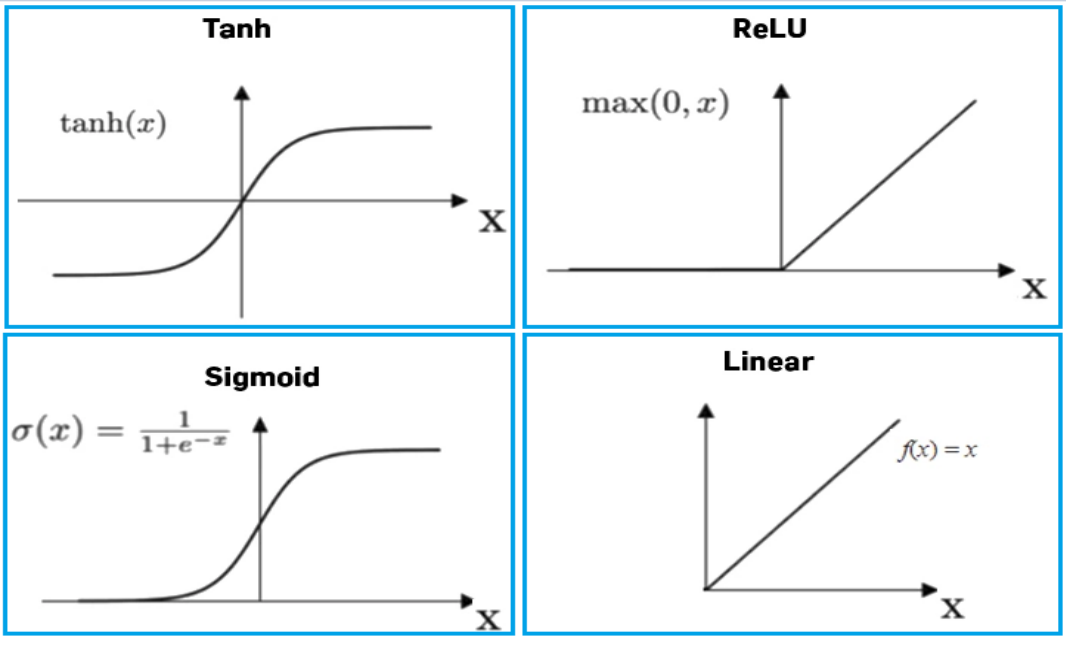

Рисунок. 3.2 - Функції активації

Опис функцій активації:

– Sigmoid - функція дозволяє перетворити будь-яке дійсне число на діапазон від 0 до 1. Ця функція використовується для задач бінарної класифікації, де вихідний сигнал можна розглядати як ймовірність належності до певного з двох класів;

– ReLU (Rectified Linear Unit) - функція повертає нуль для від'ємних значень вхідного сигналу та саме значення для додатних значень вхідного сигналу. Функція дозволяє ефективно вирішувати проблему зникненням градієнту в нейронних мережах з багатьма шарами;

– Tanh - функція подібна до Sigmoid, але повертає значення від -1 до 1. Ця функція використовується для класифікації задач, коли потрібно розрізняти між двома зворотними класами;

– Softmax - дозволяє перетворити вихідні значення нейронної мережі на дискретний розподіл ймовірностей для кількох класів. Вона зазвичай використовується для класифікації задач з декількома класами.

Таким чином, у випадку використання функції активації ReLU, сигнал який пройшов через вхідну функцію $f_{\text{вх}}$ та функцію активації $f_{\text{а}}$ можна описати наступним виразом:

$$y = max\left(0, \sum_{i=1} w_i x_i\right), \tag{3.3}$$

В процесі навчання нейронної мережі ваги зав’язків кожного нейрона поступово змінюються доти поки вихідний сигнал нейронної мережі не стає максимально наближений до очікуваного результату. Одним із основних методів навчання нейронних мереж є градієнтний спуск (Gradient descend). Завдання стоїть у мінімізації функції втрат (або цільової функції) шляхом зміни ваг зав’язків нейронів в протилежному напрямку градієнту функції втрат. Градієнтний спуск дозволяє знайти локальний мінімум функції втрат і, таким чином, налаштувати параметри нейронної мережі для досягнення добрих

показників якості нейронної мережі.

В завданнях класифікації з N класами для оцінки крос-ентропії між цими класами (Categorical Crossentropy) для нейронів в останньому шарі нейронної мережі використовується функція активації Softmax. В такому випадку значення функції втрат (loss function) можна записати у наступному вигляді:

$$L(y, \hat{y}) = \sum_{i=1}^{N} y_i * \log(\hat{y}_i), \tag{3.4}$$

Де $L(y, \hat{y})$ – значення функції втрат;

y_i - фактичний клас;

$\hat{y}_i$- клас визначений нейронною мережею.

Реалізація градієнтного методу заснованого на ітераційній процедурі корекції значень ваг виглядає у відповідності з формулою [7]:

$$w_{k+1} = w_k + \alpha_k p(w_k), k = 0,1,2 \tag{3.5}$$

Де w_k та w_{k+1} - поточне та нове значення ваг;

α_k – параметр швидкості навчання (learning rate);

$p(w_k)$ – містить градієнт функції втрат.

Для розрахунку градієнта функції втрат використовується метод зворотного поширення помилки (backpropagation). Градієнти обчислюються для кожного зв’язку в мережі, і є векторами частинних похідних функції втрат по відношенню до ваг зв’язків та використовуються для їх оновлення. Послідовно надаються вхідні дані з навчальної вибірки, і, починаючи з останнього шару, обчислюються градієнти функції помилки для кожного попереднього нейрона і, таким чином, оновлюються ваги усіх зв’язків мережі.

Показник швидкості навчання впливає на швидкість мінімізації функції

втрат та якість мережі. Занадто велике значення показника швидкості навчання може призвести до розбіжності, а надто маленька – до повільної збіжності або застрягання в локальних мінімумах. Зазвичай задається статичним на весь цикл навчання. Для покращення результатів навчання потрібно виконати процедуру пошуку більш оптимального значення.

Штучні нейронні мережі можуть бути навчені контрольованим та неконтрольованим шляхами. У першому випадку, мережа навчається шляхом передавання відповідної вхідної інформації та прикладів вихідної інформації. Так зване навчання з учителем. Найпоширенішим прикладом такого навчання є розпізнавання образів зображень. Коли вхідною інформацією може бути якесь зображення, а вихідною інформацією класифікація для відповідного зображення. У даному випадку нейрона мережа буде займатись класифікацією двовимірних об’єктів – спектрограм кадрів вхідного сигналу.

Розробка та дослідження нейронних мереж здійснюється вже протягом останніх 50 років. На сьогоднішній день досліджено багато архітектур нейронних мереж, однак серед найбільш поширених можна виділити згорткові нейронні мережі (CNN), рекурентні нейронні мережі (RNN,), та їх поєднання – згортково-рекурентні нейронні мережі (CRNN).

3.1.1 Згорткова нейронна мережа (CNN)

Згорткова нейронна мережа – тип багатошарової або глибинної нейронної мережі, яка отримала свою назву завдяки математичної операції – згортки.

Архітектура мережі була натхнена особливістю роботи зорової області кори головного мозку тварин. Окремі нейрони кори реагують на стимули лише в обмеженій області зорового поля, відомій як рецептивне поле. Рецептивні поля різних нейронів частково перекриваються таким чином, що вони покривають усе зорове поле. ЗНМ використовують порівняно мало попередньої обробки, в порівнянні з іншими алгоритмами класифікації зображень. Це означає, що мережа навчається фільтрів. Ця незалежність у конструюванні ознак від

апріорних знань та людських зусиль є великою перевагою.

Згорткова нейронна мережа має три важливі будівельні блоки:

– шар згортки (Convolutional layer) - витягує ознаки із зображення або його частин. Отримує елементи з вихідного зображення, «скануючи» зображення за допомогою фільтра, наприклад, 3×3 пікселів. Для кожної області 3×3 пікселів у зображенні операція згортки (Рис. 3.3) обчислює точкові добутки між значеннями пікселів зображення та вагами, визначеними у фільтрі;

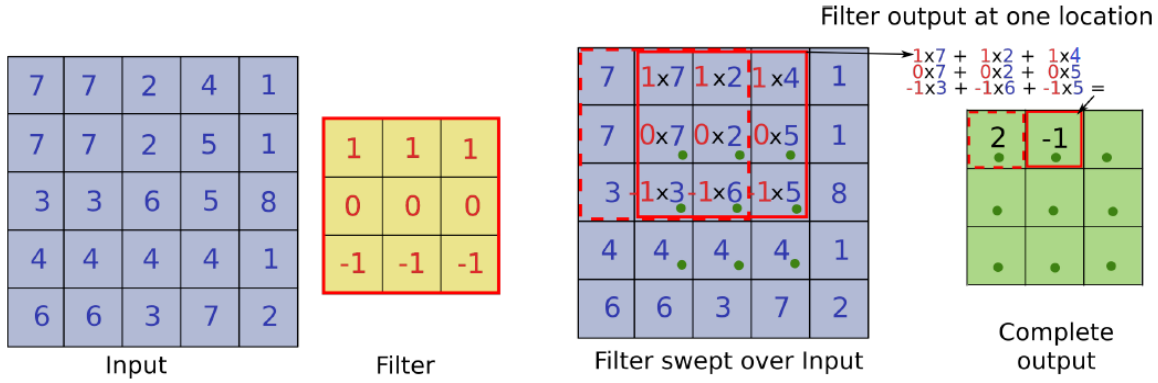

Рисунок 3.3 - Операція згортки

– шар субдискретизації або агрегувальний (pooling layer) (Рис. 3.4) - зменшує розмірність даних, щоб зосередитись на найважливіших елементах. Це прискорює навчання та зменшує загальну кількість параметрів. Зазвичай існує кілька кроків згортки та субдискретизації. Субдискретизація базується на концепції «розсувного вікна». Вона застосовує статистичну функцію до значень у межах певного розміру вікна, відомого як фільтр згортки або ядро. Це може бути, наприклад, вибірка максимального, мінімального або середнього значення в околицях вікна;

– повнозв’язний шар (fully connected, dense layer) приймає вирівняну форму ознак, визначених у попередніх шарах, і використовує їх для прогнозування зображення. Нейрони у повноз'вязному шарі мають з'єднання з усіма виходами попереднього шару, як це можна бачити у звичайних нейронних мережах.

Рисунок 3.4 - Операція субдескритизації

Поширено застосування так званого згладжувального шару (flatten layer) який згортає просторові розмірності входу в розмірність каналу, або робить одномірний масив з багатовимірного. Зазвичай розташований перед повнозв'язним шаром.

Слід зазначити що операція згортки не є нелінійною тому додавання нелінійності робиться за рахунок використання функцій активації.

Згорткові нейронні мережі добре підходять для пошуку локальних залежностей в даних, що залежно від завдання може бути як перевагою так і недоліком. Завдяки операціям субдискретизації можна знаходити оптимальну архітектуру за співвідношенням кількості параметрів та отриманої якості.

З огляду на те що спектрограми являють собою двовимірні об'єкти, як звичайні зображення, використання загорткових мереж може бути цілком оправданим.

3.1.2 Рекурентні нейронні мережі (RNN). Блоки LSTM, GRU

Рекурентні нейронні мережі отримали свою назву через здатність обробляти з послідовні дані, де кожен елемент залежить від попередніх елементів і може впливати на майбутні. У RNN нейрони мають зворотні зв`язки що додає аналог пам'яті та дозволяє інформації "циркулювати" через мережу і зберігати попередні стани або контекст. Ця здатність дозволяє використовувати

RNN для роботи з послідовностями даних різних типів, таких як текст, звук, часові ряди. Схему RNN зображено на рисунку 3.5 [8].

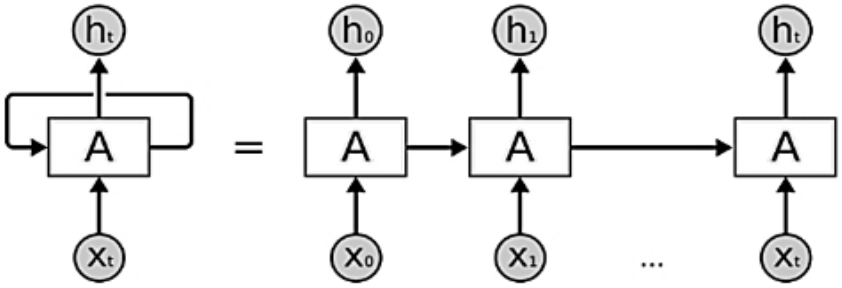

Рисунок 3.5 - Схема рекурентної ШНМ

Згідно схеми, вхідні дані X_t поступають до блоку A який повертає значення h_t. Завдяки наявності зворотного зв'язку інформація передається від одного кроку навчання мережі до іншого. Довжина такого ланцюжка у класичних RNN є обмеженою, існують проблеми затуханням градієнта на довгих послідовностях [8]. Для вирішення проблеми з градієнтом були розроблені вдосконалені блоки RNN, наприклад, довгої короткочасної пам'яті (LSTM) та вентильний рекурентний вузол (GRU).

Блок LSTM – це вузол RNN, який може запам'ятовувати значення для довгих, або коротких проміжків часу. Це забезпечується завдяки тому, що вузол не використовує функції активації в межах своїх рекурентних складових. Таким чином, значення, що зберігається, не розплющується ітеративно з плином часу, і член градієнту або штраф не має схильності розмиватися, коли для його навчання застосовується зворотне поширення у часі [7].

Кожен блок LSTM (Рис. 3.6) складається з трьох вентилів – вхідного, вихідного, забуття та комірки пам'яті (Cell State). Вентилі вирішують яку інформацію слід додати, виводити на вихід та видалити з комірки пам'яті. Кожне з цих воріт функціонує як нейрон з сигмоїдальною функцією активації. Ідея їхнього об'єднання полягає в тому, що час забувати настає тоді, коли з'являється нове значення, варте запам'ятовування. Комірка пам'яті, в свою чергу і є основним елементом блоку, в якій інформація зберігається на довгий термін.

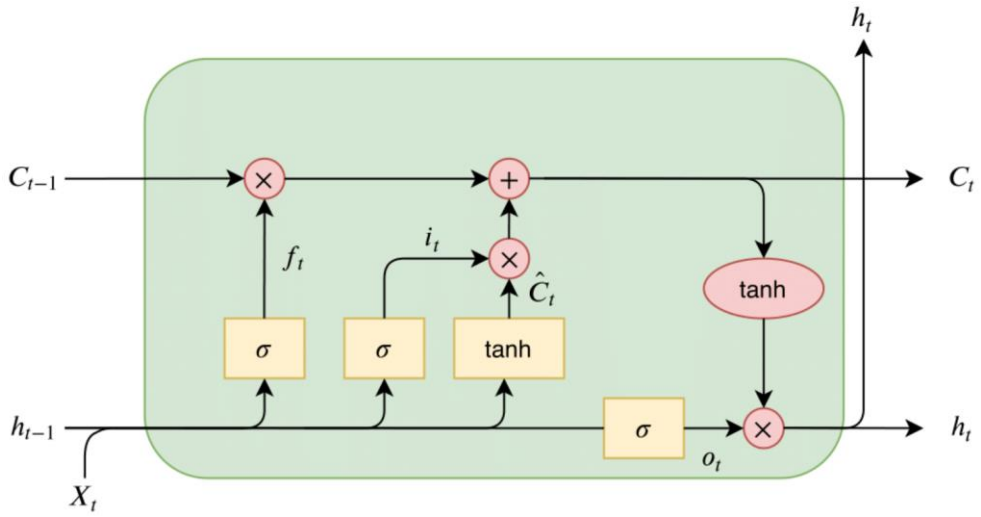

Рисунок 3.6 - Схема блоку LSTM

Роботу традиційного LSTM блоку можна описати наступним образом:

$$
\begin{aligned}
f_t &= \sigma(W_f \cdot [h_{t-1,} X_t] + b_f, \\
i_t &= \sigma(W_i \cdot [h_{t-1,} X_t] + b_i, \\
o_t &= \sigma(W_o \cdot [h_{t-1,} X_t] + b_o, \\
\hat{C}_t &= tanh(W_c \cdot [h_{t-1,} X_t] + b_c, \\
C_t &= i_t \cdot \hat{C}_t + f \cdot C_{t-1}, \\
h_t &= o_t \times \tanh(C_t)
\end{aligned}
\qquad 3.5
$$

Де f_t - вектор вентиля забуття(вага пам’ятання старої інформації);

i_t - вектор вхідного вентиля (вага отримання нової інформації);

o_t - вектор вихідного вентиля (кандидатність на вихід);

W_f, W_i, W_o – матриці ваг відповідних вентилів;

$\hat{C}_t$ - вектор активації входу комірки пам’яті;

C_t - вектор стану комірки;

h_t - вихідний вектор блоку.

На стан комірки пам’яті блоку впливають ваги W_f, W_i, W_o відповідних вентилів. Тренування цих ваг є завданням мінімізації функції втрати та

проходить за допомогою метода зворотного поширення помилки у часі.

Основна перевага LSTM полягає в їх здатності ефективно моделювати довгострокові залежності в послідовностях даних, та у стійкості до проблеми зникнення градієнта. Проте основними мінусами є ресурсозатратність, потреба в великих обсягах навчальних даних та часу на навчання. Задля того щоб зменшити обчислювальне навантаження та при цьому зберегти переваги LSTM пізніше було розроблено декілька модифікацій. Найбільшу популярність отримала модифікація під назвою «вентильний рекурентний вузол» GRU (Рис. 3.7) яка була запропонована в 2014 році [9] .

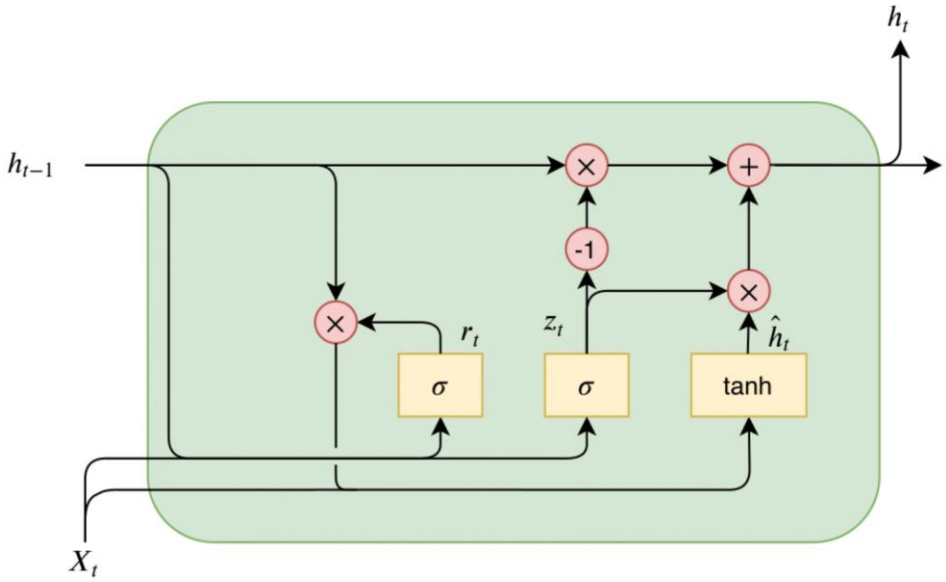

Рисунок 3.7 - Схема блоку GRU

Основна відмінність між GRU та LSTM блоками полягає в тому, що у GRU вентилі входу та забуття об’єднані в єдиний вентиль оновлення стану комірки пам’яті. Блок GRU є трохи простішим ніж LSTM та має менше параметрів, що позитивно вплинуло на швидкість навчання та обчислювальну вартість без значного зниження якості.

Якщо взяти до уваги той факт що вхідні дані з вібро-акустичних сенсорів являють собою часові ряди та такі що, у даному випадку, містять ознаки перехідних процесів, використання RNN блоків LSTM\GRU має позитивно вплинути на якість моделі ШНМ. Це можливо завдяки наявності «пам’яті» у таких блоків та можливості уловлювати контекст з довгих послідовностей.

3.2 Тестування моделей штучних нейронних мереж

В даному підрозділі буде проведено тестування трьох моделей штучних нейронних мереж: класичної згорткової та двох згортково-рекурентних (CRNN) з використанням блоків LSTM та GRU. Завданням тесту є вибір моделі нейронної мережі яка буде мати кращі показники прогнозування перехідного процесу, у даному випадку, умовно, одного з чотирьох рівнів зносу підшипника.

3.2.1 Формування набору даних для тренування

У якості вхідних даних будуть використовуватись послідовності спектрограм отриманих за допомогою короткочасного перетворення Фур'є даних з набору «NASA Bearings», запису для каналу №3 тесту №3. Спектрограми отримуються з використанням 128 Mel фільтрів для полоси частот у якому, за результатами попереднього аналізу, прослідковуються ознаки перехідних процесів - діапазоні 3-5кГц. Вхідні дані розбиваються на відрізки - кадри по 61440 семпла – що дорівнює тривалості 3 секунди з частотою дискретизації 20480Гц. Кожен наступний кадр (Рис. 3.8) має перекриття з попереднім на 2 секунди – 40960 семплів. Розмірність кожного кадру склала 117x117 пікселів.

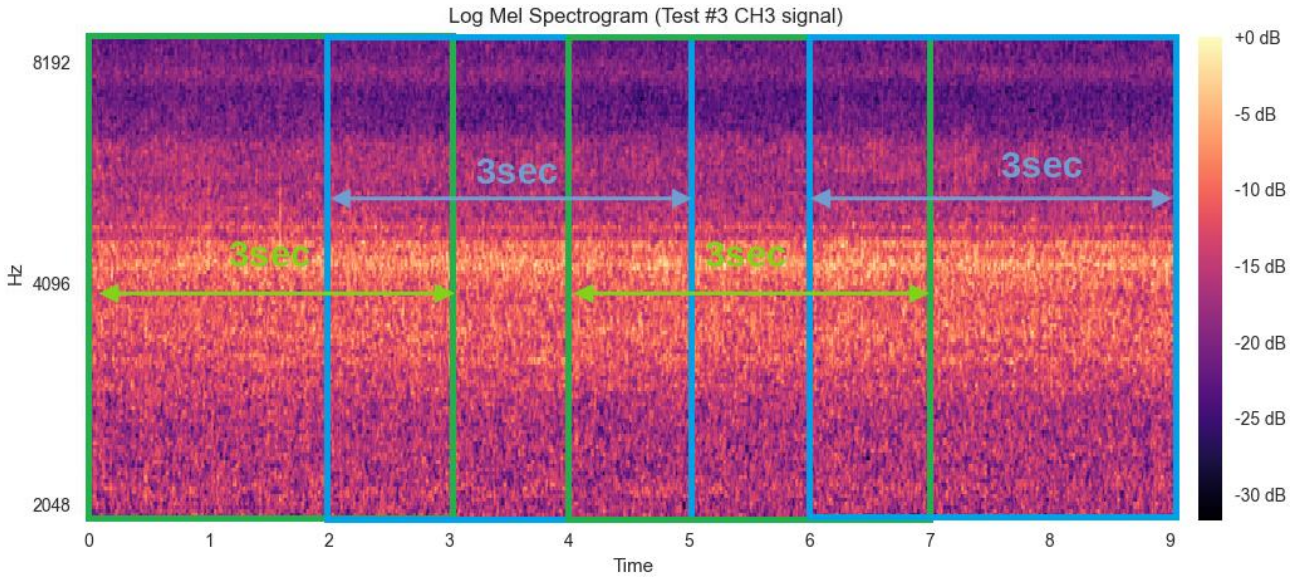

Рисунок 3.8 - Кадри спектрограм

Перекриття спектрограм робиться для додаткового зберігання попереднього контексту та для розширення тестового набору даних. Таким чином кожен кадр містить частки від попереднього та наступного кадра.

Для того щоб реалізувати прогнозування стану зносу було вирішено розбити всю послідовність у файлі запису на чотири послідовні сегменті рівної тривалості. Всім кадрам які належать до певного сегмента було призначено мітку цього сегмента. Останні 25 секунд запису містять ознаки несправності та різко наростаючий рівень шумів, тому не беруться до уваги в даному тесті. Ці відрізки будуть використані для навчання моделей прогнозування типу несправності.

Для сегментації файлів даних використовувався функціонал доступний в таких бібліотеках Python як Librosa, numpy. Отримані сегменти були розташовані в теках S1- S4 для більш зручного опрацювання бібліотекою для роботи з нейронними мережами TensorFlow.

3.2.2 Архітектури моделей для тестування

Для тестування було відібрано три моделі штучних нейронних мереж:

– згорткова нейронна мережа (CNN) запропонована в [11] має три згорткові блоки та два послідовні повнозв’язні блоки (Рис. 3.9).

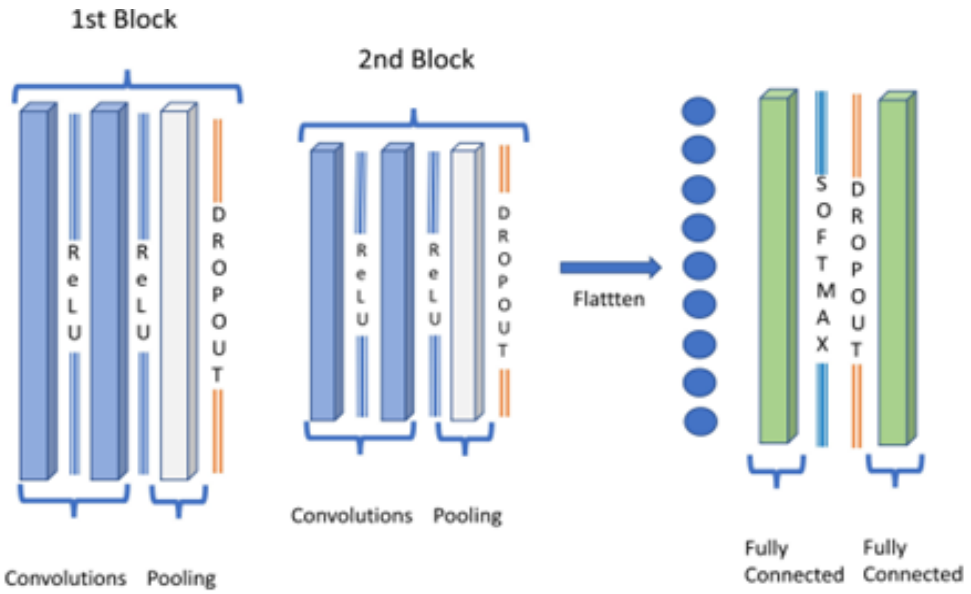

Рисунок 3.9 - Схема згорткової ШНМ

Перші два згорткові блоки містять по два шари згортки, останній один шар. Загальна кількість фільтрів в згорткових блоках складає 320. Розмір ядра 3x3.

Для зменшення розмірності даних використовується додаткові шари субдескритизації (Pooling) з ядром 2x2. У якості функції активації використовується ReLU. Останній повнозв'язний шар має функцію активації Softmax тому що вхідні дані поділені на класи - номери сегментів. Також для того щоб допомогти уникнути перенавчання моделі використовуються додаткові шари регуляризації (Droupout);

– згортково-рекурентна (CRNN) нейронна мережа (Рис. 3.10) утворена шляхом поєднання блоку згортки та блоку LSTM. Блок згортки має 128 фільтрів з ядром 3x3. Блок LSTM містить 32 нейрони. Присутній тільки один повнозв'язний шар. В згортковому блоці використовується функція активації ReLU, в повноз'вязному Softmax. Згідно дослідженню [11] поєднання згорткових та рекурентних блоків дає кращу витривалість до наявності шумів у сигналі;

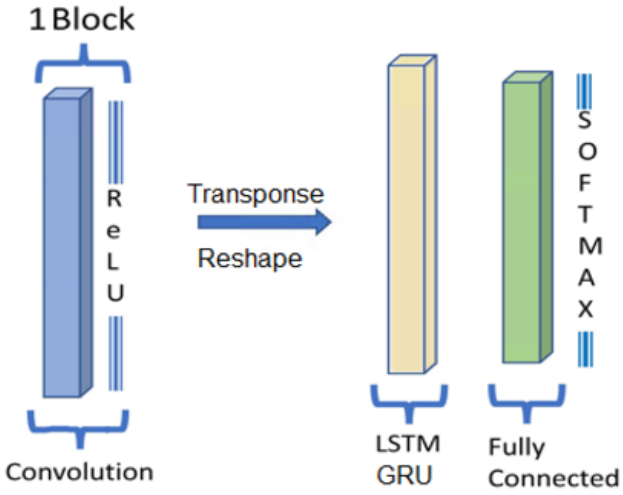

Рисунок. 3.10 Схема згортково-рекурентної ШНМ

– згортково-рекурентна (CRNN) з блоком GRU замість блока LSTM. В іншому ця модель має такі ж саму архітектуру як і мережа №2. Використання цієї мережі у тестуванні потрібно для порівняння показників швидкості навчання та якості передбачення з моделлю №2 та наскільки використання блоку GRU може мати рацію з точки зору ресурсозатратності.

3.2.2 Результати тестування моделей

Навчання кожної моделі проводилось за 100 епох. Під час навчання використовувався механізм раннього завершення (Early stopping). Цей механізм дозволяє завершити навчання якщо показники точності моделі не будуть покращуватись задану кількість епох поспіль. В якості оптимізатора навчання використовувався Adam (Adaptive Moment Estimation). Вхідні дані були розділені на навчальну та тестову вибірки в пропорції 4 до 1. Для кожної моделі обчислювались наступні показники: втрати (Loss), точність (Accuracy) по навчальній та тестовій вибіркам, Recall та F1.

Показник Recall (повнота) відображає здатність моделі до прогнозування всіх позитивних класів у наборі. Формула має вигляд:

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}, \quad 3.6$$

Це співвідношення істинно позитивних (True Positives) до загального числа істинно позитивних і помилково негативних (False Negatives) відповідей моделі під час оцінки якості (evaluation).

Показник F1 враховує як точність (accuracy), так і повноту (recall) моделі, і об'єднує їх в одне значення. Дозволяє знаходити баланс між точністю і повнотою класифікації. Корисний, коли важливо оцінити кількість помилок як помилково позитивних, так і помилково негативних. Обчислюється за допомогою формули:

$$Recall = \frac{2 * (\ Accuracy\ * Recall)}{Accuracy + Recall}, \quad 3.7$$

Додатково було проведено оцінку ресурсозатратності по таким показникам як час навчання та прогнозування, об’єм використаної оперативної пам’яті.

Результати тестування моделей відображено у Таблицю 3.1-3.2.

Таблиця 3.1 – Показники якості моделей

Band: 3000-5600 Hz	Loss	Accuracy	Recall	F1
Model_1 Conv2D + Dense	0.253	0.901	0.901	0.899
Model_2 Conv2D + LSTM	0.203	0.943	0.941	0.969
Model_3 Conv2D + GRU	0.171	0.936	0.935	0.936

Таблиця 3.2 – Показники ресурсозатратності моделей

Band: 3000-5600 Hz	E_Time	L_Time		Params	Size
		CPU	GPU		
Model_1 Conv2D + Dense	1.5 sec	20 min	2 min	1.7 M	21 MB
Model_2 Conv2D + LSTM	6.9 sec	56 min	6 min	1.9 M	23 MB
Model_3 Conv2D + GRU	6.5 sec	29 min	4 min	1.4 M	17 MB

E_Time – час затрачений на прогнозування вибірці з 100 кадрів. Зроблено виміри для навчання з використанням CPU та GPU.

L_Time – час затрачений на навчання моделі. Слід зазначити що час навчання вказаний з урахуванням механізму раннього завершення.

Отримані показники якості та ресурсозатратності моделей свідчать про перевагу CRNN моделей які поєднують згорткові (Conv2D) та рекурентні шари (LSTM\GRU). Хоча розбіжність в показниках якості невелика і всі моделі мають значення які можна вважати задовільними проте показники ресурсозататності суттєво відрізаються. Модель яка використовує LSTM блок навчається в два-три рази довше ніж згорткова модель. Значно прискорити навчання можна використовуючи сучасні GPU. Час прогнозування довший більш ніж в чотири рази, що в деяких випадках це може бути вирішальним.

Результати отримані при використанні процесора Intel(R) Core(TM) i7-10850H та графічного адаптера Nvidia Quadro T2000.

4 РОЗРОБКА ПРОТОТИПА СИСТЕМИ МОНІТОРИНГУ

4.1 Структурна схема системи моніторингу

Основне завдання системи це надання в реальному часі інформації про поточний стан обладнання. Ця інформація повинна містити прогнозний рівень зношеності обладнання, код несправності у разі виявлення, показники отримані з додаткових сенсорів. Отримана інформація має зберігатись та виводитись на інтерфейс користувача для подальшого аналізу та прийняття рішень.

З огляду на вищесказане пропонується використання клієнт-серверної схеми, компоненти якої зображено на рисунку 4.1.

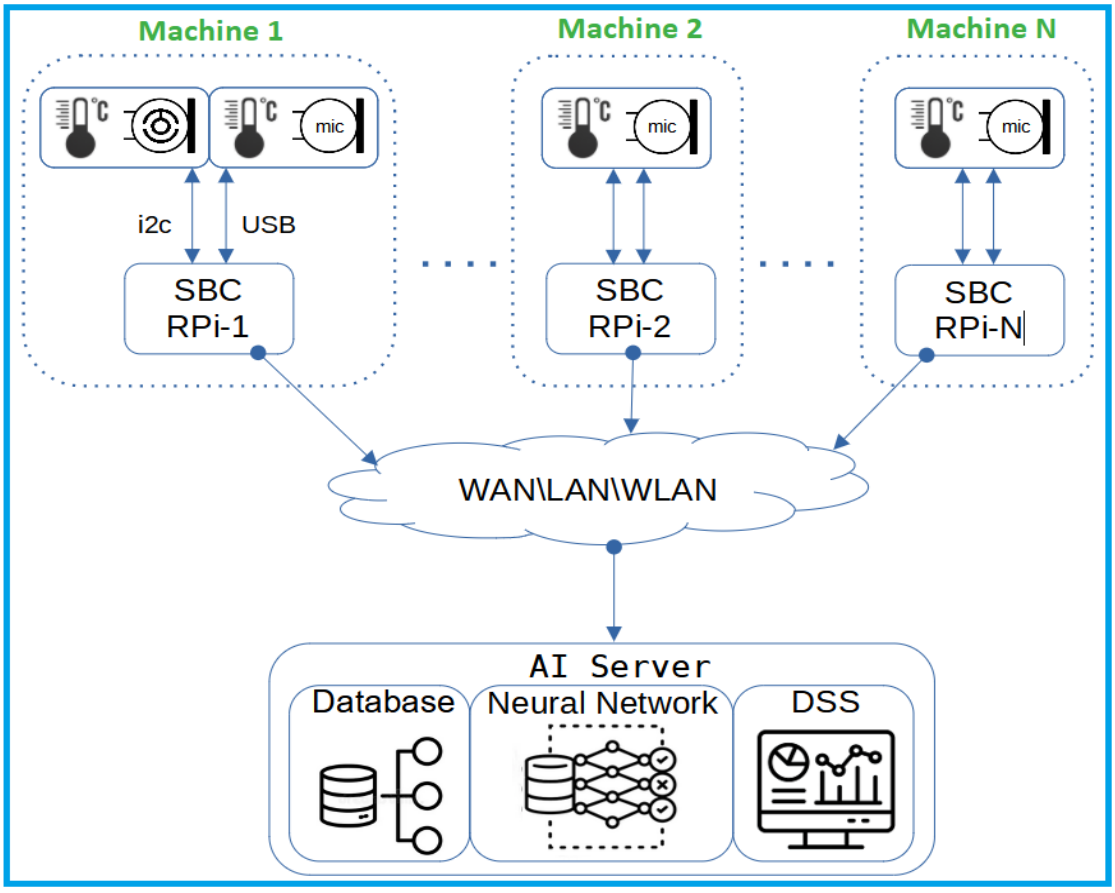

Рисунок 4.1 - Структурна схема роботи системи

Система складається з клієнтів – агентів збору даних (Machine 1-N) та

серверу обробки даних (AI Server). До агентів підключені вібраційні, акустичні, температурні та інші сенсори. Поточна інформація з сенсорів передається на сервер аналізу даних. Передача даних відбувається по дротовій (LAN) або бездротовій (WLAN) мережі. Сервер обробки даних проводить аналіз отриманих даних за допомогою штучної нейронної мережі. Отримані в результаті аналізу дані а також показники додаткових сенсорів відображаються на веб сторінці моніторингу яка розташована на сервері. В якості результатів виступають поточний рівень зносу обладнання та код несправності, у випадку якщо несправність була виявлена. Дані на веб сторінці оновлюються по мірі надходження інформації. Також сервер містить базу даних в яку зберігаються всі отримані показники.

4.2 Агенти збору даних

Основне завдання агентів це попередня обробка та відправка на сервер даних отриманих з сенсорів, у реальному часі. Приклад схеми підключення сенсорів яка використовувалась в стенді для навантажувального тестування підшипників надано на рисунку 4.2 [12]. В процесі цього тестування було отримано набір даних «NASA Bearings».

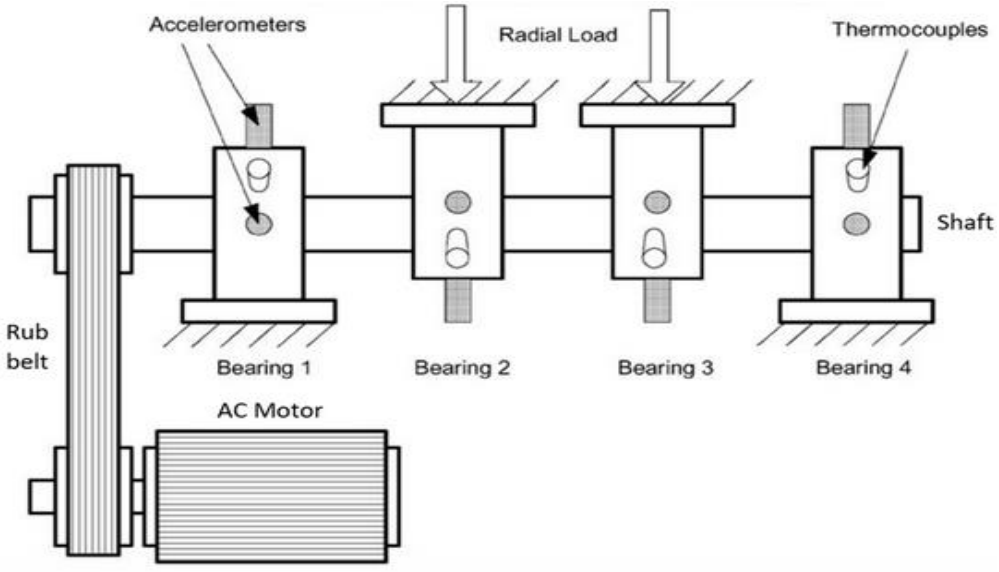

Рисунок 4.2 - Стенд випробування підшипників. Підключення сенсорів

На кожен з чотирьох кріплень підшипників було встановлено по два акселерометри типу PCB 353B33 та по одній термопарі.

В якості агентів може виступати будь який SBC (Small Board Computer), який може відповідати наступним вимогам:

– мати достатню кількість портів USB, I²C, GPIO для можливості підключення декількох сенсорів одночасно;

– наявність можливості підключення до локальної (LAN) мережі Ethernet або бездротової (WLAN) мережі Wi-Fi;

– обчислювальна потужність центрального процесора та об’єм оперативної пам’яті мають бути достатніми для роботи операційної системи з ядром Linux та програм для обробки даних з сенсорів в реальному часі, написаних з використанням мови програмування Python.

Враховуючи вищезазначені вимоги пропонується в якості SBC для агента використовувати Raspberry Pi (3, 4, Zero W).

До агентів підключаються вібраційні або акустичні сенсори. Інформація з цих сенсорів відправляються на сервер за допомогою RTP (Real-time Transport Protocol) протоколу, як аудіо потік. Інформація з додаткових сенсорів таких як, наприклад, датчики температури виводиться на власній веб сторінці агента та забирається сервером. Попередня обробка даних з сенсорів, відправка на сервер та вивід даних з сенсорів на просту веб сторінку реалізується за допомогою бібліотек Python таких як Librosa, FFMpeg-Python, Flask.

Використання декількох агентів дозволяє сформувати багатоагентну систему. Один або декілька агентів можуть бути вузлами взаємодії з сервером обробки та іншими агентами. Це може покращити надійність та ефективність задач з асинхронними, паралельними діями між агентами [15], що відповідає концепції Industry 4.0. Використання таких SBC як Raspberry Pi дозволяє підключати різноманітну апаратуру управління обладнанням та таким чином організувати зворотній зв'язок між агентами та сервером обробки даних. У разі виявлення несправності система зможе застосувати план відповідних дій в автоматичному режимі або за вказівками оператора.

4.2.1 Вібраційні та акустичні сенсори

Вибір вібраційних сенсорів залежить від специфіки обладнання та середовища яке підлягає моніторингу. Встановлення таких сенсорів передбачає безпосередній контакт. Серед варіантів які добре зарекомендували себе в умовах де не потрібен широкий динамічний діапазон можна відзначити Grove LDT0-028, SW-420. Також в якості сенсора вібрації можна розглянути використання акселерометрів, наприклад, 3-осьовий акселерометр GY-291 ADXL345.

Всі зазначені сенсори використовують шину I²C, що накладає певні обмеження по довжині кабелю та скорості передачі даних описані в стандарті [13]. Для подолання цих обмежень були розроблені рішення які дозволяють подовжити шину (bus extender), наприклад, NXP P82B715, PCA9615. Згідно специфікації [14] це рішення дозволяє використовувати кабель типу вита пара Cat5e у якості середи передачі даних на дистанції до 20 метрів. Деякі промислові акселерометри мають варіанти з підключенням по шині USB, наприклад, PCB 633A01, Amphenol 333D01. Довжина стандартного кабелю становить вже 2.9 м, а з використанням активного сягає 15м. Для більш специфічних рішень існують бездротові сенсори PCB 670A01. Зовнішній вигляд сенсорів відображено на рисунку 4.3.

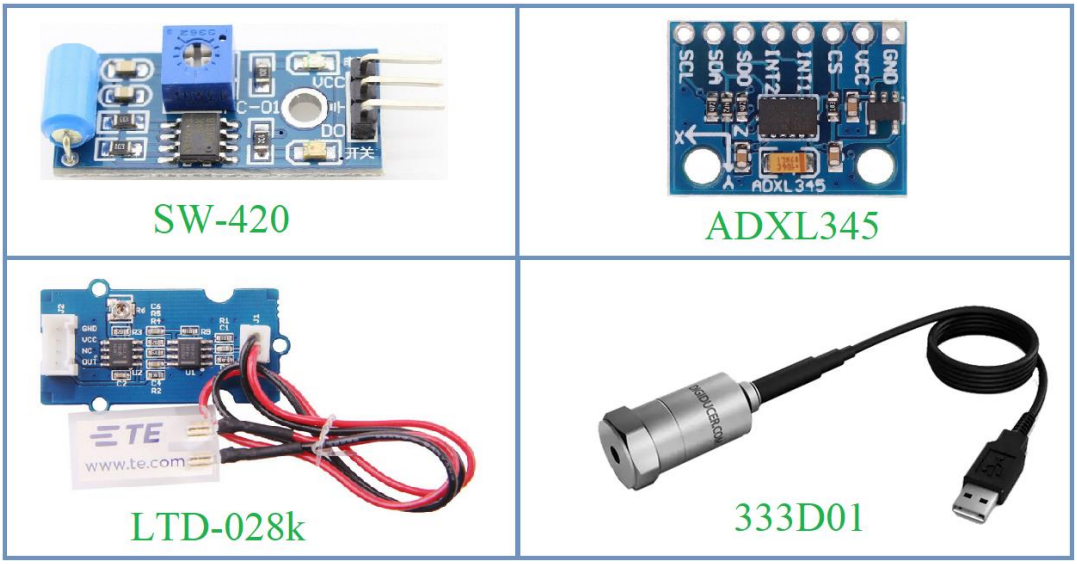

Рисунок 4.3 - Зовнішній вигляд вібраційних сенсорів, акселерометрів

Для отримання інформації з акустичної сцени роботи обладнання можна використовувати різноманітні мікрофони. На відміну від сенсорів вібрації мікрофони вимірюють не коливання об’єкту а коливання тиску середи в якій знаходяться, в даному випадку повітря.

Основними типами мікрофонів за принципом дії є динамічні та конденсаторні. Динамічні мікрофони досить міцні та надійні, вони менш вибагливі до середи використання та витривалі до постійних навантажень сильним звуковим тиском. Вони не потребують живлення для роботи та зазвичай доступніші за ціною (порівняно з конденсаторними). Мають гіршу чутливість, особливо на високих частотах. Конденсаторні мікрофони, навпаки, мають гарну чутливість та кращій частотний діапазон але потребують додаткового живлення для роботи та більш вибагливі.

Серед характеристик які мають мікрофони потрібно відзначити наступні: частотний діапазон, чутливість та спрямованість.

Частотний діапазон вимірюється в Гц, та зазвичай складає 20-20000 Гц, що включає більшість звуків які сприймаються людським вухом. Для порівняння, низькочастотні мікрофони можуть мати діапазон від 30 Гц до 15000 Гц, а високочастотні мікрофони для запису, наприклад, співу птахів та ізольованих звуків можуть мати діапазон від 5 000 Гц до 30 000 Гц.

Чутливість мікрофона вимірюється у мікровольтах на Паскаль (μV/Pa) або децибелах (дБ SPL). Це показує, наскільки ефективно мікрофон може перетворювати звуковий тиск в електричний сигнал. Вища чутливість вказує на те, що мікрофон може реєструвати слабкі звуки з великого відстані.

Спрямованість мікрофона зазвичай надається виробником у вигляді діаграми направленості та визначає, наскільки чутливо мікрофон реагує на звук з різних напрямків, відносно своєї орієнтації.

На рисунку 4.4 наведено зображення моделей мікрофонів які можна використовувати для аналізу акустичної сцени роботи обладнання.

Модель петличного мікрофона Audio-Technica ATR3350 є типовим представником мікрофонів загального призначення. Модель Fifine K031 являє

собою радіосистему з конденсаторним мікрофоном та приймачем з USB інтерфейсом. Модель PCB 130A24 поєднує мікрофон та підсилювач в одному міцному корпусі з захистом від води та пилу, розроблений для використання ззовні приміщень та в промислових умовах. Як окремий вид акустичного сенсора можна виділити багатомікрофонні системи, наприклад, ReSpeaker 4-Mic Array, спеціально розроблений для Raspberry Pi.

Рисунок 4.4 - Моделі мікрофонів

За типом підключення мікрофони можуть бути дротові та бездротові. В дротовому варіанті це може бути аналогові (Jack 3.5, XLR) або USB і навіть I²C варіанти. Бездротові працюють з допомогою різноманітних радіосистем, працюючих, зазвичай, у діапазоні UHF (УКВ) або Bluetooth (2.4 ГГц.) Довжина кабелю залежить від типу підключення та може сягати 5-30м.

4.3 Сервер обробки даних

Сервер обробки даних має забезпечувати функції отримання та попередньої обробки вхідних даних які надходить від агентів у реальному часі, аналіз цих даних, зберігання та візуалізацію отриманих результатів.

Пропонується кожну функцію зробити окремим модулем та організувати взаємодію між цими модулями. Для програмної реалізації роботи кожного модуля використати мову програмування Python. Вибір Python обумовлений наявністю великої кількості бібліотек для аналізу даних, роботи з базами даних, нейронними мережами та можливостями візуалізації отриманих результатів, в тому числі за допомогою веб інтерфейсу.

Сервер обробки даних включає три модуля:

– модуль роботи з базою даних. Цей модуль відповідає за роботу з локальною базою даних SQLite. Вибір цього типу бази даних обумовлений тим що це вбудована СУБД та не вимагає окремого сервера або налаштувань. Бази даних зберігаються в одному файлі, що робить її дуже компактною та легкою в установці та використанні. При цьому SQLite використовує мову запитів SQL, що дозволяє виконувати різноманітні операції з даними, такі як вставка, оновлення, видалення та вибірка. Для взаємодії з базою даних використовується функціонал доступний у бібліотеках Python – SQLAlchemy, Sqlite3. База даних містить таблицю з наступними полями: унікальний ідентифікатор агента, результат обробки вхідних даних нейронною мережею (прогноз), мітка часу, показники додаткових датчиків. Показники додаткових датчиків отримуються сервером з кожного агенту окремо;

– модуль аналізу даних. Відповідає за попередню обробку вхідних даних які надходять з агентів в реальному часі по протоколу RTP. Попередня обробка полягає в отримані ознак з спектрограми кадру вібро-акустичного сигналу довжиною 3 секунди та передачі цієї ознаки до обробки нейронною мережею. Кадри перекриваються між собою на дві секунди. Завдяки цьому, в реальному часі, нейронна мережа робить прогнозування кожної секунди для кадру довжиною в 3 секунди. Оброблені дані надходять на опрацювання моделлю штучної нейронної мережі. На виході мережі отримується результат прогнозу стану обладнання у вигляді масиву ймовірностей приналежності до певного класу (стану). Стан може означати клас рівня зносу та\або клас несправності. Пропонується два підходи до отримання класу стану. Перший підхід – це

використання двох моделей нейронних мереж, однієї для прогнозування стану обладнання, другої для прогнозування стану несправності. Другий підхід полягає в використання однієї моделі для прогнозування і стану зносу і стану несправності. Мінусом другого підходу є в відсутності роздільного моніторингу цих станів. Функціонал модуля реалізується за допомогою використання наступних бібліотек Python: Tensorflow, Librosa, FFMpeg, Numpy;

– модуль інтерфейсу моніторингу показників. Призначений для візуалізації отриманих в результаті аналізу даних. До цих даних відносяться прогноз стану обладнання у вигляді рівня зносу та помилки несправності. Також на інтерфейс виводиться показники додаткових сенсорів та службова інформація. Показники виводяться для кожного агента у системі у окремому блоці. Дизайн макет інтерфейсу наведено на рисунку 4.5.

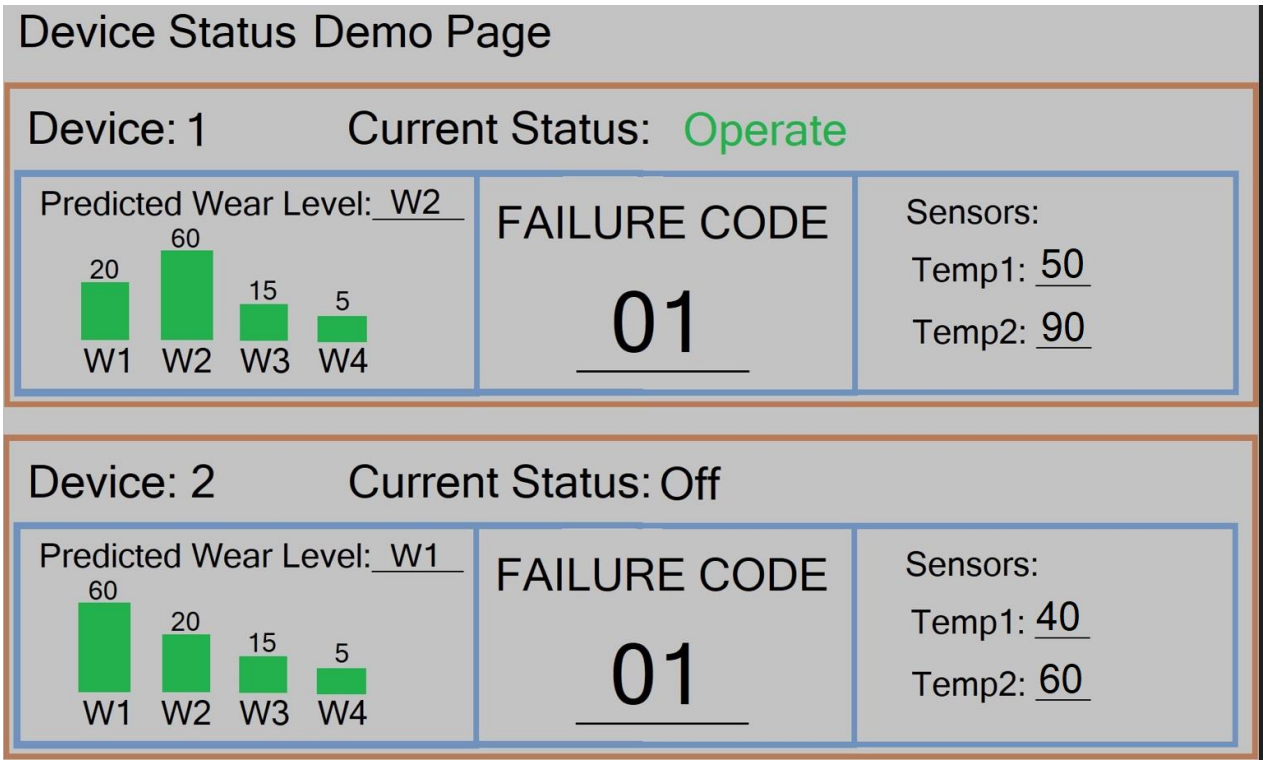

Рисунок 4.5 - Інтерфейс моніторингу показників

На інтерфейсі відображено стани для двох агентів – Device 1, Device 2. Агенти збору даних збирають показники з кожного екземпляра обладнання та додаються системи моніторингу. Показник рівня зносу пристрою має назву

«Predicted Wear Level» та має, у цьому прикладі, чотири рівня: W1-W4. Цифра над кожним зеленим стовпчиком означає вірогідність приналежності до певного класу яку надає нейрона мережа. У даному випадку мається на увазі що модель ШНМ була натренована для чотирьох ступенів зносу обладнання - класів. Завдяки тому що процес зносу обладнання має перехідний характер, а також тому що нейрона мережа має деякий рівень неточності, сусідні рівні будуть також мати деякі не нульові значення. Проте основний тренд буде зберігатись. Стовпчик з найбільшим значенням повинен вказувати на поточний рівень зносу. Оновлення рівнів зносу проходить кожну секунду, як результат аналізу кадру який містить інформацію за три останні секунди. Значення «FAILURE CODE» виводить код несправності. Тобто, у випадку коли на виході нейронної мережі буде виявлено клас помилки то цей клас буде відображено. Серед додаткової інформації на сторінці моніторингу може бути присутні показники додаткових сенсорів, наприклад температури (Temp1, Temp2), наявність зв'язку з агентом, базою даних, поточний час. Всі показники синхронно записуються до бази даних. Модуль реалізовано за допомогою бібліотеки Python – Flask.

Вимоги до апаратного забезпечення серверу залежать від кількості агентів та об'єму вхідних даних. В загальному випадку сервер має бути оснащений сучасним багатоядерним процесором x86 архітектури, об'ємом оперативної пам'яті не менше ніж 16GB, швидким накопичувачем SSD з інтерфейсом NVMe.

Окремо стоїть питання навчання та перенавчання моделей штучних мереж. Навіть за використанням сучасних центральних процесорів цей процес може бути повільним, особливо при великому обсязі даних та відносно великої кількості параметрів моделі нейронної мережі. Використання графічних процесорів Nvidia дозволяє прискорити навчання в десятки або навіть сотні разів. Компанія Nvidia також розробляє SBC серії Jetson які поєднують центральний процесор архітектури ARM та графічний процесор. Компенсувати невелику потужність ARM процесорів в процесі обробки даних нейронною мережею можна використовуючи спеціальні нейропроцессорів [16], наприклад, Google Coral TPU.

5 ОПИС ПРОГРАММНОЇ ЧАСТИНИ

5.1 Інструменти розробки програми

Програмна частина системи реалізована за допомогою мови програмування Python оскільки ця мова є найпопулярнішою мовою для роботи з ШНМ та має велику кількість бібліотек, що полегшує розробку.

Були застосовані наступні бібліотеки:

– TensorFlow. Для обробки вхідних даних та навчання ШНМ;
– Librosa. Для аналізу вхідних даних, отримання спектрограм;
– Matplotlib. Для візуалізації даних, побудови графіків;
– Flask. Для побудови інтерфейсу користувача;
– FFMPeg-python. Для відправки даних з агенту на сервер;
– SQLAlchemy, Sqlite3. Для взаємодії з базою даних SQLite.

Розробка проводилась у середовищах розробки Visual Studio Code та PyCharm. Створено окремі екземпляри програм у форматі Python (*.py), для роботи на агентах збору даних та на сервері системи моніторингу. Головна програма системи моніторингу використовує модель ШНМ для формування прогнозу рівня стану зношеності обладнання відповідно вхідним даним які надходять від агентів збору даних. Результатом її роботи є відображення на веб сторінці сервера показників рівня зношеності, коду несправності та додаткової інформації. Також головна програма відповідає за зберігання цих даних в локальній базі даних. Результатом роботи програми агента є отримання даних з сенсорів, попередня обробка даних з вібро та акустичних сенсорів та відправка їх на сервер, відображення на веб сторінці агента інформації з додаткових сенсорів. Програма тренування моделей штучних мереж створена у форматі Jupyter Notebook (*.ipynb). Робота з цим форматом більш зручна у випадках коли, наприклад, потрібно часто міняти параметри моделі та проводити аналіз отриманих результатів, робити візуалізацію та будувати графіки.

5.1.1 Бібліотека TensorFlow та API Keras

Основною метою даної роботи є дослідження ефективності використання різних архітектур нейронних мереж глибинного навчання для аналізу послідовних даних, наприклад аудіо, які можуть мати ознаки скритих перехідних процесів. Існує чимало фреймворків для роботи з нейронними мережами які підтримують мову Python та зараз активно розвиваються: Tensorflow-Keras, PyTorch, Onnx. Однією з найпопулярніших є бібліотека Tensorflow а також фреймворк Keras, котрий зарекомендував себе гарною швидкістю у роботі з ШНМ та має велику кількість документації та підтримку суспільства, що прискорює розробку програми. TensorFlow - це програмна бібліотека з відкритим вихідним кодом для чисельних розрахунків з використанням графів потоків даних. TensorFlow був створений для машинного глибокого навчання компанією Google для задоволення її потреб у системах, здатних будувати та тренувати нейронні мережі для виявляння та розшифровування образів та кореляцій, аналогічно до навчання й розуміння, які застосовують люди. Основний API для роботи з бібліотекою реалізований для Python, також існують реалізації для C#, C++, Java та інші. Підтримує обчислення на CPU, GPU та прискорювачах тензорних операцій (нейропроцессори) Google TPU (Tensor processing units).

Keras представляє собою API інтерфейс високого рівня для нейронних мереж, написаний на Python. Цю бібліотеку можна використовувати додатково до TensorFlow або Theano. З 2017 року команда TensorFlow Google вирішила підтримувати Keras в основній бібліотеці TensorFlow. Keras було замислено радше як інтерфейс, аніж як самостійну систему машинного навчання. Вона пропонує високорівневий, інтуїтивній набір абстракцій, який робить розробку моделей нейронних мереж відносно простою, незалежно від того яке обладнання для обчислень використовується. Містить численні втілення широко вживаних нейромережних будівельних блоків, таких як шари, цільові та передавальні функції, оптимізувальники та безліч інструментів для спрощення роботи із зображеннями та текстом.

5.2 Алгоритм роботи програми агента

Програма агента розроблена з використанням наступних бібліотек Python: Flask, W1thermsensor, threading, subprocess. Схема роботи програми агента представлена на рисунку 5.1.

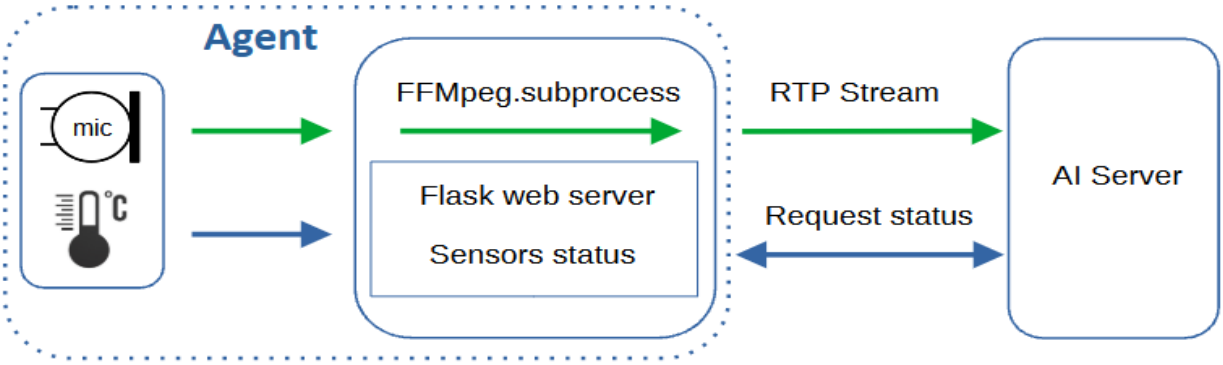

Рисунок 5.1 - Схема роботи програми агента

Основне завдання програми агента це забезпечення передачі даних з підключених сенсорів до сервера обробки даних (AI Server). Дані з акустичного сенсора передаються на сервер по протоколу RTP за допомогою програми FFMpeg яка запускається у додатковому процесі:

```
a_str=subprocess.Popen(ffmpeg.split(),stdout=subprocess.PIPE)
```

Показники з температурного сенсора DS1820 отримуються за допомогою бібліотеки W1ThermSensor наступним чином:

```
t_sensor = W1ThermSensor(Sensor.DS18B20)
```

Стан процесу який відповідає за потокову передачу (a_str) та поточні показники сенсора температури зберігаються у глобальних змінних temp_r та onair. Їх оновлення забезпечує функція update_status. Оновлення відбувається кожні 2 секунди. Функція запускається в додатковий нитці (Лістинг 5.1).

Лістинг 5.1:

```
    def update_status():
        global temp_r
        global onair
        while True:
            temp_r = round(t_sensor.get_temperature(),1)
            onair = check_stream()
            time.sleep(2)
    # Запуск функції в потоці
    status_t = threading.Thread(target=update_status,
daemon=True)
    status_t.start()
```

Сервер може запросити показники температури та стан потоку відправивши агенту веб запит (Request) методом HTTP GET наступним чином:

```
    response = requests.get(url=reqUrl)
    if response.status_code != 200:
        print('HTTP Error:', response.status_code)
    text_response = json.loads(response.text)
```

За обробку запита та передачу поточного стану змінних відповідає функціонал бібліотеки Flask:

```
    @app.route('/status', methods=['GET'])
    def status():
        if temperature_r is not None:
            return jsonify({"temp": temp_r}, {"stream": onair})
        else:
            return jsonify({"temp": "NA"}, {"stream": onair})
```

Відповідь на запит має вигляд строки у форматі JSON:

```
    [{'temp': 23.6}, {'stream': 'Stream On'}]
```

5.3 Алгоритм роботи програми сервера моніторингу

Серверна частина складається декількох компонентів та розроблена з використанням наступних бібліотек Python: Tensorflow, Librosa, SQLAlchemy, FFMpeg, Flask, Subprocess, Pandas. Схема роботи сервера (AI Server) представлена на рисунку 5.2.

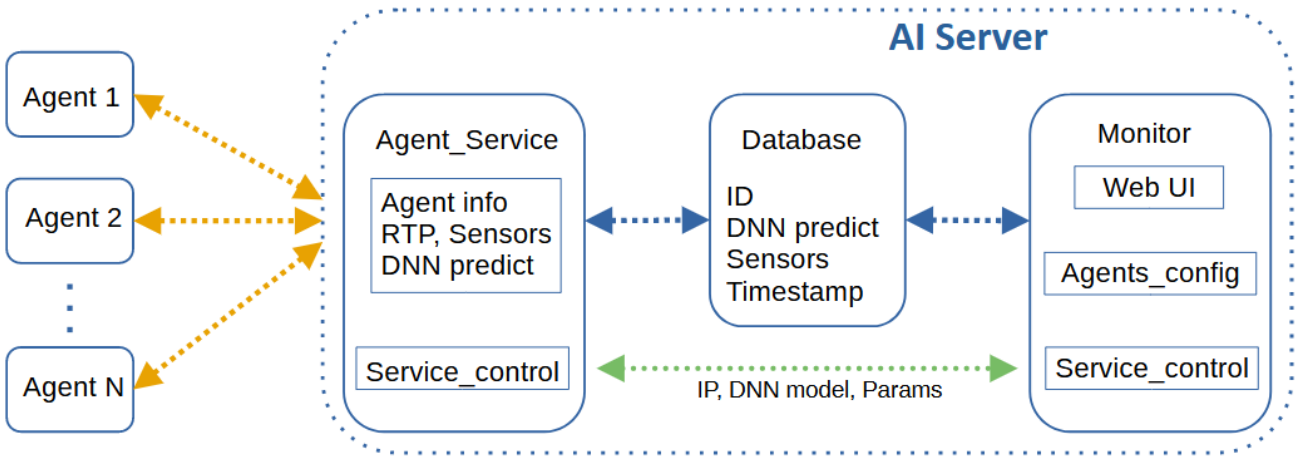

Рисунок 5.2 - Схема роботи сервера

Призначення сервера моніторингу це відображення на інтерфейсі користувача результатів прогнозу стану обладнання отриманого від опрацювання вхідної інформації нейронною мережею, поточних показників додаткових сенсорів, інформації про агента збору даних.

Функціонал сервера реалізовано двома головними компонентами – монітор (Monitor) та сервіс (Agent service) обробки даних отриманих від агентів системи. Обидва компоненти взаємодіють з базою даних SQLite.

Програма монітор відповідає за формування та оновлення веб сторінки інтерфейсу користувача та за керування сервісом обробки даних. Роль веб сервера виконує бібліотека Flask. Веб сторінка містить окремі секції для даних з кожного агента. Для відображення та оновлення даних на веб сторінці клієнта (браузера) використовується технологія SSE (Server Side Event). Після встановлення HTTP з'єднання з сервером клієнт отримує автоматичні оновлення з сервера у вигляді повідомлень у форматі JSON. Клієнт підписується на потік

/Stream автоматично. За це відповідає наступний код маршруту Flask а в програмі монітору:

```
@app.route('/stream')
def stream():
    return Response(generate_data_from_database(agents), content_type='text/event-stream')
```

Приклад повідомлення у форматі JSON:

```
{"a_id": "A1", "a_dnn": "4,88,8,0", "a_temperature": 22.2, "a_timestamp": "2023-09-24 18:59:07"}
```

Повідомлення містять інформацію про ідентифікатор агента (a_id), результати прогнозу нейронної мережі (a_dnn), показники з додаткових сенсорів (a_temperature), часову мітку (a_timestamp). За формування повідомлення відповідає функція generate_data_from_database (Лістинг 5.2).

Лістинг 5.2:

```
def generate_data_from_database(agents):
    with app.app_context():
        while True:
            for each_agent in agents:
                a_id, a_dnn, a_temperature, a_timestamp = get_latest_data_from_db(str(each_agent.agent_id))
                yield (f'data: {{"a_id": "{a_id}", "a_dnn": "{a_dnn}", '
                       f'"a_temperature": {a_temperature}, "a_timestamp": "{a_timestamp}"}}\n\n')
            time.sleep(2)
```

Для кожного агента, робиться вибірка останнього рядка за бази даних з

інтервалом в дві секунди. За це відповідає функція get_latest_data_from_db наведена у Лістингу 5.3.

Лістинг 5.3:

```
    def get_latest_data_from_db(agent_name):
        last_entry =
Device.query.filter_by(agent=agent_name).order_by(Device.id.desc()
).first()
        if last_entry:
            agent_id = last_entry.agent
            agent_dnn_predict = last_entry.dnn_status
            agent_temperature = last_entry.temperature
            agent_timestamp = last_entry.timestamp
        return agent_id, agent_dnn_predict, agent_temperature,
agent_timestamp
```

Веб сторінка моніторингу має додатковий код на JavaScript який розбирає отримане повідомлення на стороні браузера клієнта. Отримані значення записуються у відповідні елементи секції сторінки для відповідного агента. Оновлюються статус бари рівня зносу (W1-W4), поточний показник зносу, код несправності (за наявністю), показники додаткових сенсорів(температура), додається мітка часу для отриманого запису в базі даних, стан агента.

Інформація про кожного агента зберігається в файлі конфігурації. В файлі міститься інформація про назву агента та його IP адресу, модель нейронної мережі, додаткову конфігурацію. Ця потрібно для керування сервісом обробки даних (agent_service) для чого в програмі монітору створено клас agent.

Після запуску програма монітор зчитує інформацію про агентів з файлу конфігурації та запускає для кожного агента сервіс обробки даних (Лістинг 5.4).

Лістинг 5.4:

```
    for each_agent in agents:
      print(each_agent.agent_dnn)
```

```
       each_agent.process_id = subprocess.Popen(["python", "agent-
service.py",
     str(each_agent.agent_id), str(each_agent.agent_ip),
     str(each_agent.agent_dnn), str(each_agent.agent_db)])
```

Сервіс обробки даних завантажує модель нейронної мережі та додаткові параметри фільтра частот які вказана в конфігурації агента. Підключається до агента за його IP адресом. Вхідний потік даних з акустичного сенсора отримується за допомогою бібліотеки FFMpeg які агент передає в RTP потоці. Ці дані буферизуються до довжини тривалістю 3 секунди і потім подаються для обробки моделлю нейронної мережі. В подальшому відбувається зсув буфера на довжину тривалістю в одну секунду. Таким чином, кожної секунди, модель нейронної мережі робить прогноз для вікна даних в 3 секунди. Функція отримання спектрограми та фільтрації по частоті наведена у Лістингу 5.5.

Лістинг 5.5:

```
     def get_spectr(wav, min_freq, max_freq, sampling_rate,
mel_bins):
          zero_padding = tf.zeros([sampling_rate*3] -
tf.shape(wav), dtype=tf.float32)
          wav = tf.concat([zero_padding, wav], 0)
          stfts = tf.signal.stft(wav, frame_length=2048,
frame_step=512, fft_length=2048)
          spectrograms = tf.abs(stfts)
          num_spectrogram_bins, lower_edge_hertz, upper_edge_hertz,
num_mel_bins = 1025, min_freq, max_freq, mel_bins
          linear_to_mel_weight_matrix =
tf.signal.linear_to_mel_weight_matrix(num_mel_bins,
num_spectrogram_bins, sampling_rate,
          lower_edge_hertz, upper_edge_hertz)
          mel_spectrogram = tf.tensordot(spectrograms,
linear_to_mel_weight_matrix, 1)
          mel_spectrogram.set_shape(spectrograms.shape[:-
1].concatenate(linear_to_mel_weight_matrix.shape[-1:]))
```

```
    mel_spectrogram = tf.transpose(mel_spectrogram)
    return mel_spectrogram
```

Отримана спектрограма для кожного кадру отфільтровано по мінімальній (min_freq) та максимальній (max_freq) частоті. Ці характеристики отримуються під час попереднього аналізу вхідних даних з сенсорів та дозволяє отримати тільки важливі ознаки. Функцію прогнозування наведено у Лістингу 5.6.

Лістинг 5.6:

```
def predict_in(mel, model):
    mel_rs = tf.expand_dims(mel, axis=0)
    mel_rs = mel_rs[..., tf.newaxis]
    predict = model.predict(mel_rs, verbose=0)
    predict = (np.round(predict * 100)).astype(int)
    predict = ','.join(map(str, predict))
    return predict
```

Результат прогнозу (predict) проходить серію перетворень тому що спочатку являє собою numpy масив даних а для більш зручної обробки даних, в подальшому, його потрібно перетворити в строку. Ймовірності приналежності до певного класу виражено в процентному співвідношенні та розділено комами.

Інформацію з додаткових сенсорів агент виводить на власну веб сторінку. Ці дані збираються сервісом обробки даних за допомогою запита методом HTTP GET. Функцію запиту показника сенсору температури наведено у Лістингу 5.7.

Лістинг 5.7:

```
def get_agent_temperature(agent_ip):
    reqUrl = 'http://' + agent_ip + ':5000/status'
    temperature = None
    response = requests.get(url=reqUrl)
    if response.status_code != 200:
        print('HTTP Error:', response.status_code)
        return -1000
```

```
        text_response = json.loads(response.text)
        for item in text_response:
            if "temperature" in item:
                temperature = item["temperature"]
        return temperature
```

Результат виконання функції get_agent_temperature та поточний прогноз нейронної мережі, отриманий від функції predict_in дані записуються до бази даних за допомогою бібліотеки sqlite3 (Лістинг 5.8).

Лістинг 5.8:

```
    conn = sqlite3.connect(agent_db)
        cursor = conn.cursor()
    insert_query = (f"INSERT INTO devices "
                            f"(agent, dnn_status, temperature) "
                            f"VALUES
('{agent_id}','{predicts}','{get_agent_temperature(agent_ip)}')")
            cursor.execute(insert_query)
            conn.commit()
```

Таким чином сервіс обробки даних заносить в базу даних дані від кожного агента. Програма монітору даних періодично запитує з бази даних останні записи для кожного агента у разі створення HTTP сесії, та передає їх як повідомлення.

5.4 Алгоритм навчання моделей штучних нейронних мереж

В роботі проаналізовано три моделі нейронних мереж: згорткової (CNN) та двох згортково-рекурентних (CRNN) мереж з блоками LSTM та GRU.

Навчання моделей та аналіз отриманих результатів проводився в середовищі розробки Visual Studio Code. Програма написана у форматі Jupyter Notebook (*.ipynb), що дає більш гнучкий підхід до розробки та відладки. Використовувались бібліотеки TensorFlow, Librosa, Numpy та Matplotlib.

Загальний алгоритм навчання та аналізу моделей наступний:

- отримання початкового набору даних з вхідних файлів з записами показників вібраційних та акустичних сенсорів;

- отримання масиву спектрограм та маркування класів;

- створення моделей ШНМ;

- тренування та зберігання моделей ШНМ з найліпшими показниками;

- оцінка якості отриманих моделей на тестових даних.

Схема алгоритму навчання моделей ШНМ представлена на рисунку 5.1.

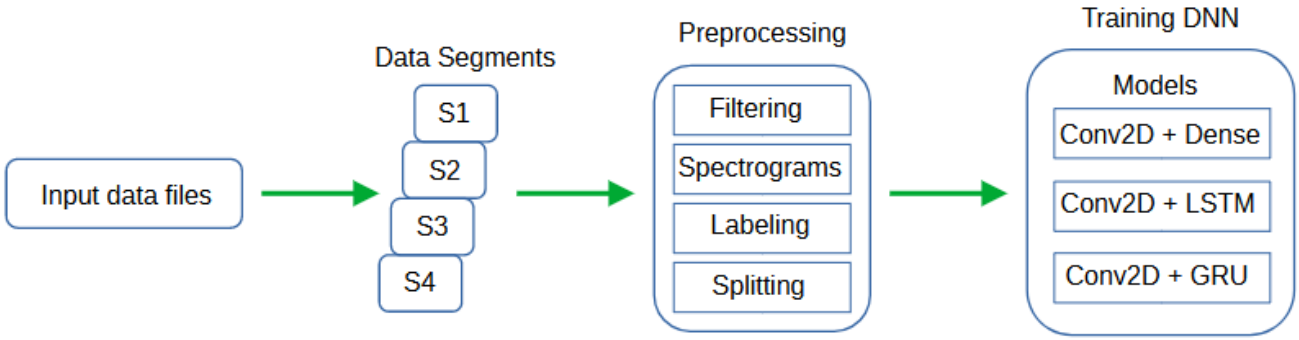

Рисунок 5.3 - Блок схема навчання моделей ШНМ

Вхідні дані потребують фільтрації та попередньої підготовки. Для того щоб мати можливість розпізнавати рівень зносу обладнання пропонується розбивати записи на сегменти. Файл з набору даних «NASA Bearings» який містить інформацію про показники вібраційного датчика на підшипнику №3 в тесті №3 можна розбити на сегменти за допомого функції (Лістинг 5.9).

Лістинг 5.9:

```
def split_audio(input_audio_path, output_dir, num_segments):
    audio, sr = librosa.load(input_audio_path, sr=None)
    segment_duration = len(audio)
    for i in range(num_segments):
        start = i * segment_duration
        end = (i + 1) * segment_duration
        segment = audio[start:end]
        segment_path =
```

```
f"{output_dir}/{input_file}_segment_{i+1}.wav"
            sf.write(segment_path, segment, sr)
```

Отримані сегменти потрібно розмістити в відповідних теках. Розміщення в теках потрібно для спрощення маркування спектрограм відповідно кожного сегмента. Це робиться під час створення набору даних для навчання та тестування ШНМ. Набір містить спектрограми заданого фільтру частот для відрізків тривалістю, еквівалентно, 3 секундам та перекриттям в 1 секунду.

Функція створення набору для тестування наведено Лістингу 5.10.

Лістинг 5.10:

```
    def create_dataset(data_dir, class_names, hop_size,
window_size):
        dataset = []
        labels = []
        for i, class_name in enumerate(class_names):
            class_dir = os.path.join(data_dir, class_name)
            for filename in os.listdir(class_dir):
                print(filename)
                file_path = os.path.join(class_dir, filename)
                features = load_and_extract_features(file_path,
hop_size, window_size)
                dataset.extend(features)
                labels.extend([i] * len(features
        return np.array(dataset), np.array(labels)
```

Де data_dir це тека яка містить теки з сегментами назви яких занесено у лист class_names. Крок перекриття та розмір вікна це hop_size та window_size відповідно. Функція load_and_extract_features (Лістинг 5.11) відповідає за вилучення ознак:

Лістинг 5.11:

```
    def load_and_extract_features(file_path, hop_size,
```

```
window_size):
         audio = librosa.load(file_path, sr=sampling_rate)[0]
         samples_per_frame = int(sampling_rate * hop_size)
         num_frames = (len(audio) - samples_per_frame) //
(samples_per_frame) + 1
         mel_spectr = []
        for i in range(num_frames):
             start_sample = i * samples_per_frame
             end_sample = start_sample + int(sampling_rate *
window_size)
             window_audio = audio[start_sample:end_sample]
             mel_spectr.append(preprocess_wav(window_audio,
sampling_rate))
            return mel_spectr
```

Функція отримання спектрограм (preprocess_wav) для певної смуги частот (MIN_FREQ, MAX_FREQ) та нормованих по шкалі Mel має таку ж саму структура як і функція get_spectr яку було наведено в попередньому підрозділі.

Створення фінального датасету для навчання показано у Лістингу 5.12.

Лістинг 5.12:

```
     dataset_f, labels_f = create_dataset(data_dir_full,
class_names, hop_size, window_size)
     from sklearn.model_selection import train_test_split
     X_train_f, X_test_f, y_train_f, y_test_f =
train_test_split(dataset_f, labels_f, test_size=0.2,
random_state=42)
```

Розділення на тренувальний набір тестовий робиться за допомогою функції train_test_split бібліотеки Scikit-learn в співвідношенні 80% та 20%.

Далі буде наведено три функції для створення моделей ШНМ.

Модель №1 – загорткова (CNN) представлена у Лістингу 5.13.

Лістинг 5.13:

```
    def create_model1(input_shape, num_classes):
        input = Input(shape=input_shape)
        conv2d_1 = Conv2D(filters=32, kernel_size=(3, 3),
padding='same', activation='relu')(input)
        conv2d_2 = Conv2D(filters=32, kernel_size=(3, 3),
padding='same', activation='relu')(conv2d_1)
        maxp2d_1 = MaxPooling2D(pool_size=(2, 2))(conv2d_2)
        dropout_1 = Dropout(0.25)(maxp2d_1)
        conv2d_3 = Conv2D(filters=64, kernel_size=(3, 3),
padding='same', activation='relu')(dropout_1)
        conv2d_4 = Conv2D(filters=64, kernel_size=(3, 3),
padding='same', activation='relu')(conv2d_3)
        maxp2d_2 = MaxPooling2D(pool_size=(2, 2))(conv2d_4)
        dropout_2 = Dropout(0.3)(maxp2d_2)
        conv2d_5 = Conv2D(filters=128, kernel_size=(3, 3),
padding='same', activation='relu')(dropout_2)
        maxp2d_3 = MaxPooling2D(pool_size=(2, 2))(conv2d_5)
        dropout_3 = Dropout(0.3)(maxp2d_3)
        flatten_1 = Flatten()(dropout_3)
        dense_1 = Dense(64)(flatten_1)
        dropout_4 = Dropout(0.5)(dense_1)
        output = Dense(units=num_classes,
activation='softmax')(dropout_4)
        model1 = tf.keras.Model(inputs=input, outputs=output)
        return model1
```

Використовуються три блоки згортки (Conv2D), два повнозв’язні блоки (Dense) поєднані з функціями підвибірки (MaxPooling) та часткового виключення (Dropout). Активаційна функція в останньому блоці – Softmax.

Модель №2 – згортково-рекурентна (CRNN), Conv2D + LSTM, представлено у Лістингу 5.14.

Лістинг 5.14:

```
    def create_model2(input_shape, num_classes):
        input = Input(shape=input_shape)
        x = Conv2D(filters=128, kernel_size=(3, 3),
activation='relu')(input)
        transp = Permute((2,1,3))(x)
        x = Reshape((x.shape[1],x.shape[2]*x.shape[3]))(transp)
        x = LSTM(32)(x)
        out = Dense(units=num_classes, activation='softmax')(x)
        model2 = tf.keras.Model(inputs=input, outputs=out)
        return model2
```

В даному випадку один блок згортки (Conv2D) поєднується з рекурентним блоком LSTM. Вихідні дані з блоку згортки проходять через процес перестановки (Permute) та зміни форми (Reshape) для того щоб отримати послідовність в LSTM.

Модель №3 – згортково-рекурентна (CRNN), Conv2D + GRU, представлено у Лістингу 5.15.

Лістинг 5.15:

```
    def create_model3(input_shape, num_classes):
        input = Input(shape=input_shape)
        x = Conv2D(filters=128, kernel_size=(3, 3),
activation='relu')(input)
        transp = Permute((2,1,3))(x)
        x = Reshape((x.shape[1],x.shape[2]*x.shape[3]))(transp)
        x = GRU(32)(x)
        out = Dense(units=num_classes, activation='softmax')(x)
        model3 = tf.keras.Model(inputs=input, outputs=out)
        return model3
```

Модель має аналогічну архітектуру моделі №2 за винятком заміни блока

LSTM на блок GRU.

Для кожної з моделей використовується оптимізатор Adam та функція втрат sparse_categorical_crossentropy, яка повертає список ймовірностей приналежності до кожного сегмента.

Навчання моделей проводилось з використанням зворотних викликів (callbacks) та контрольних точок для контролю за прогресом навчання моделі та зупинки процесу у разі його відсутності. Цей підхід дозволяє зберегти найоптимальніші ваги моделі (Лістинг 5.16).

Лістинг 5.16:

```
    filepath = "./saved_model/model1-3650.keras"
    checkpoint = ModelCheckpoint(filepath,
monitor='val_accuracy', verbose=1,save_best_only=True, mode='max')
    early_stop = EarlyStopping(monitor='val_accuracy',
patience=10, verbose=1, mode='max', restore_best_weights=True)
    callbacks_list = [checkpoint, early_stop]
    history1 = model1.fit(X_train_f, y_train_f,
                          epochs=100,
                          validation_data=(X_test_f, y_test_f),
                          callbacks=callbacks_list, verbose=1)
```

Дуже важливим етапом в розробці систем які використовують ШНМ є етап оцінки показників моделей. Для оцінки якості побудованих моделей використовуються параметри точності (Accuracy), втрат (Loss), повноти (Recall), F1. Функціє оцінки моделей наведено у Лістингу 5.17.

Лістинг 5.17:

```
    def model_evaluate(model):
        loss, accuracy = model.evaluate(X_test_f, y_test_f)
        y_pred = model.predict(X_test_f)
        recall = recall_score(y_test_f, y_pred.argmax(axis=1),
average='macro')
```

```
        f1 = f1_score(y_test_f, y_pred.argmax(axis=1),
average='macro')
        print(f"Model name:{model}")
        print("Loss:", round(loss, 3))
        print("Accuracy:", round(accuracy, 3))
        print("Recall:", round(recall, 3))
        print("F1 Score:", round(f1, 3))
```

На рисунку 5.4 представлено графіки навчання та результати оцінювання трьох моделей ШНМ за даними з набору «NASA Bearings», тест №3, канал №3 .

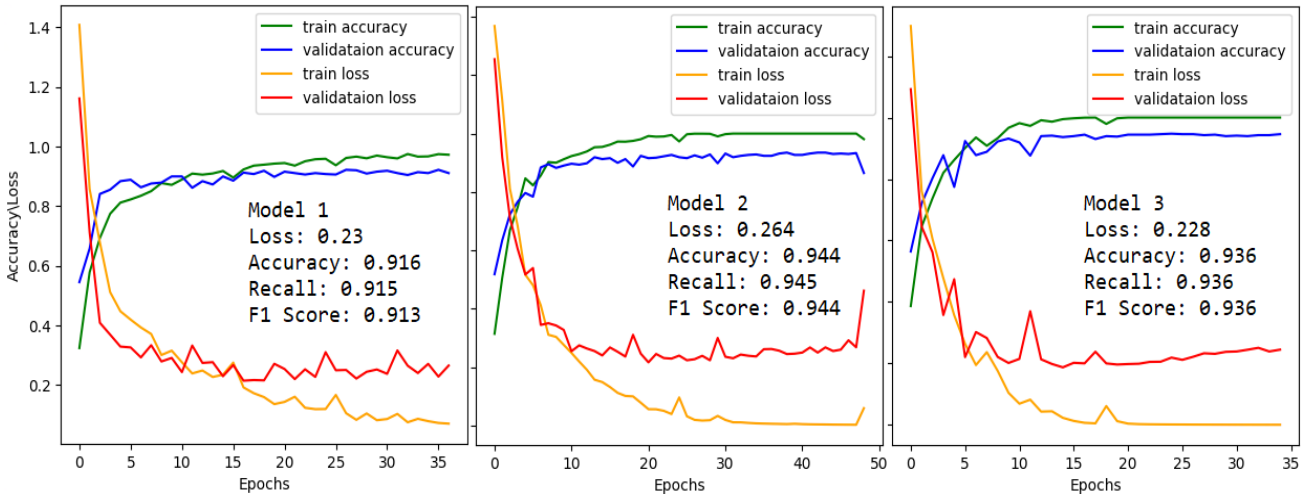

Рисунок 5.4 - Графіки навчання та оцінка моделей

Отримані результати свідчать про перевагу моделей згортково-рекурентних (CRNN) мереж над загортковою (CNN) по показникам точності та повноти на тестовому наборі даних. Слід відмітити що тривалість навчання згортково-рекурентних мереж займає більше часу (епох) а якість дуже залежить від об’єму тестового набору.

Рекомендується використовувати згортково-рекурентні моделі у випадках коли є можливість зібрати достатньо даних для навчання та у випадках коли час навчання не має великого значення. В інших випадках моделі загорткових нейронних мереж демонструють задовільні показники.

ВИСНОВКИ

В кваліфікаційній роботі було доведено важливість використання інтелектуальних систем у методі предиктивного обслуговування обладнання на сучасних високоавтоматизованих виробництвах. Продемонстрована можливість використання штучних нейронних мереж для прогнозування стану обладнання. Досліджено метод вилучення ознак перехідних процесів з гармонічних сигналів від акустичних та вібраційних сенсорів та використання їх для тренування нейронних мереж. Сформовано вимоги та розроблено прототип системи моніторингу та прогнозування стану обладнання.

Розроблена система дозволяє аналізувати сигнали сенсорів від декількох джерел одночасно, зберігати отримані результати в базі даних, та в реальному часі виводити отриману інформацію на інтерфейс моніторингу. Система складається з агентів збору даних та сервера моніторингу. За попередню обробку даних відповідають агенти збору даних які далі передають її на сервер.

Для вхідних даних з акустичних та вібраційних сенсорів проводиться попередній аналіз в частотній області виявлення та отримання параметрів перехідних процесів. Отримані результати використовуються для налаштування функції попередньої обробки даних та вилучення спектрограм, як основного джерела даних для обробки моделлю нейронної мережі.

Інтелектуальною складовою системи становить нейрона мережа, на яку покладено функцію прогнозування стану обладнання на основі вхідних даних з сенсорів. Для цього була розглянута перспектива використання згорткових та згортково-рекурентних моделей нейронних мереж. Проведено тренування та проаналізовано результати показників якості та швидкодії. За отриманими результатами зроблено висновок що у разі наявності у сервера моніторингу достатніх обчислювальних можливостей використання згортково-рекурентних мереж має переваги. В іншому випадку якість поступається швидкодії.

Головною метою роботи було продемонструвати підхід який можна

використовувати у побудові систем діагностики обладнання.

Результати роботи можуть бути корисними для інженерів, відповідальних за обслуговування та ремонт обладнання, а також для менеджерів з виробництва. Система може бути частиною більш розвинутої системи підтримки прийняття рішення, в тому числі, на основі даних про стан обладнання, та запобігати можливим простоям в роботі виробництва.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Національний інститут стандартів і технологій США (NIST) // Term Smart Manufacturing. Official site of Library NIST. URL: https://www.nist.gov/programs-projects/product-definitions-smart-manufacturing (дата звернення: 7.11.2023).

2. R. Keith Mobley. An Introduction to Predictive Maintenance, Second Edition / Mobley R. Keith. – Amsterdam : Butterworth/Heinemann, 2002. – 438 p.

3. Bengtsson M. On the importance of combining “the new” with “the old” – One important prerequisite for maintenance in Industry 4.0 [Electronic resource] / Marcus Bengtsson, Gunnar Lundström // Procedia Manufacturing. – 2018. – Vol. 25. – P. 118–125.

4. Prognostics Center of Excellence Data Set Repository - NASA // NASA. URL: https://www.nasa.gov/intelligent-systems-division/discovery-and-systems-health/pcoe/pcoe-data-set-repository/ (дата звернення: 20.11.2023).

5. Model 353B33 | PCB Piezotronics [Electronic resource] // PCB Piezotronics | Sensors to measure vibration, acoustics, force, pressure, load, strain, shock & torque. URL: https://www.pcb.com/products?m=353b33 (дата звернення: 21.11.2023).

6. Бодянский Е.В., Руденко О.Г. Искусственные нейронные сети: архитектура, обучение, применения. Харьков: Телетех, 2004. – 372 с.

7. Руденко О. Г., Бодянський Є. В. Штучні нейронні мережі: Навчальний посібник. — Харків: ТОВ "Компанія СМІТ", 2006. — 404 с.

8. Understanding LSTM Networks [Electronic resource] // Home - colah's blog. URL: http://colah.github.io/posts/2015-08-Understanding-LSTMs/ (дата звернення: 28.11.2023).

9. Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine

Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

10. Сердюк Н., Пушко В. Проблематика селекції та класифікації аудіо подій у зашумленому середовищі. *«Проблеми Інформатизації».* Тези доповідей десятої міжнародної науково-технічної конференції. м. Харків, 24-25 листопада 2022. С.5.

11. dos Santos R. Disrupting Audio Event Detection Deep Neural Networks with White Noise [Electronic resource] / Rodrigo dos Santos, Ashwitha Kassetty, Shirin Nilizadeh // Technologies. – 2021. – Vol. 9, no. 3. – P. 64.

12. Sacerdoti D. A Comparison of Signal Analysis Techniques for the Diagnostics of the IMS Rolling Element Bearing Dataset [Electronic resource] / Diletta Sacerdoti, Matteo Strozzi, Cristian Secchi // Applied Sciences. – 2023. – Vol. 13, no. 10. – P. 5977.

13. I2C-bus specification and user guide [Electronic resource] // Automotive, IoT & Industrial Solutions | NXP Semiconductors. URL: https://www.nxp.com/docs/en/user-guide/UM10204.pdf (дата звернення: 10.12.2023).

14. P82B715: I²C-Bus Extender [Electronic resource] // Automotive, IoT & Industrial Solutions | NXP Semiconductors. – Mode of access: https://www.nxp.com/docs/en/data-sheet/P82B715.pdf (дата звернення: 11.12.2023).

15. Аксак Н., Пушко В. Багатоагентні системи в Industry 4.0. Матеріали 27-го Міжнародного молодіжного форуму «Радіоелектроніка та молодь у XXI столітті»: тези доповідей, 10-12 травня 2023 р. – Харків: ХНУРЕ, 2023. Т 5.

16. Аксак Н., Пушко В. Використання нейропроцессорів у вбудованих системах. *«Тренди та перспективи розвитку мультидисциплінарних досліджень».* Матеріали II міжнародної студентської наукової конференції. м. Хмельницький, 25 листопада 2022. С.13.