

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему СТВОРЕННЯ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ
ПРОГНОЗУВАННЯ ТА ОЦІНЮВАННЯ ЯКОСТІ СЕНСОРІВ
НА ОСНОВІ ДАНИХ КАЛІБРУВАННЯ
DESIGN OF AN INTELLIGENT SYSTEM FOR PREDICTING AND
EVALUATING SENSOR QUALITY BASED ON CALIBRATION DATA

Виконав(ла): студент(ка) 4 курсу, групи КНТ-222

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Комп'ютерні науки

ДУБІЛЬ М.

(ПРІЗВИЩЕ та ініціали)

Керівник ОЛІЙНИК А.О.

(ПРІЗВИЩЕ та ініціали)

Рецензент ШИТІКОВА О.В.

(ПРІЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
 Кафедра програмних засобів
 Ступінь вищої освіти бакалавр
 Спеціальність: 122 Комп'ютерні науки
(код і найменування)
 Освітня програма (спеціалізація)) Комп'ютерні науки
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
 “ ” 2026 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

ДУБІЛЬ Миколи

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Створення інтелектуальної системи прогнозування та оцінювання якості сенсорів на основі даних калібрування
Design of an Intelligent System for Predicting and Evaluating Sensor Quality Based on Calibration Data

керівник проєкту (роботи) д.т.н., професор, ОЛІЙНИК Андрій Олександрович

,

(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “7” квітня 2026 року № 139

2. Строк подання студентом проєкту (роботи) 03 червня 2026 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ОЛІЙНИК А.О., професор		
Нормоконтроль	ДЕЙНЕГА Л.Ю., ст. викладач		

7. Дата видачі завдання “ 7 ” квітня 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка структури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділи 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент

_____ Микола ДУБІЛЬ
(підпис) (Імя ПРИЗВИЩЕ)

Керівник проєкту (роботи)

_____ Андрій ОЛІЙНИК
(підпис) (Імя ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
73 с., 3 табл., 28 рис., 3 дод., 27 джерел.

МАШИННЕ НАВЧАННЯ, НЕЙРОННІ МЕРЕЖІ, КАЛІБРУВАННЯ,
АКСЕЛЕРОМЕТР, ПРОГНОЗУВАННЯ ЯКОСТІ, PYTHON, PYTORCH.

Об'єкт дослідження - процес створення інтелектуальної системи для прогнозування та оцінювання якості сенсорів

Предмет дослідження - методи та програмні системи для прогнозування та оцінювання якості сенсорів

Мета роботи - скорочення часу оцінювання метрологічної придатності сенсорів шляхом розробки інтелектуальної системи прогнозування на основі аналізу їхніх частотних характеристик та умов експлуатації

Результати. Розроблено архітектуру триголової нейронної мережі (FreTS, MLP, CNN), що дозволяє прогнозувати відносну девіацію чутливості з точністю 95,93%.

Висновки. Мету роботи досягнуто. Розроблено інтелектуальну систему прогнозування та оцінювання якості сенсорів за допомогою мови програмування Python.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 73 pages, 3 tables, 28 figures, 3 appendixes, 27 sources.

MACHINE LEARNING, NEURAL NETWORKS, CALIBRATION, ACCELEROMETER, QUALITY FORECASTING, PYTHON, PYTORCH.

Research object – the process of developing an intelligent system for predicting and evaluating sensor quality

Research subject – methods and software systems for predicting and evaluating sensor quality.

Objective of the work – to reduce the time required to assess the metrological suitability of sensors by developing an intelligent prediction system based on the analysis of their frequency characteristics and operating conditions

Results. An architecture of a three-layer neural network (FreTS, MLP, CNN) has been developed, which allows predicting the relative sensitivity deviation with an accuracy of 95.93%.

Conclusions. The objective of the work has been achieved. An intelligent system for predicting and evaluating sensor quality has been developed using the Python programming language.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ	9
1 Аналіз предметної області та постановка задачі	11
1.1 Інтелектуальні системи для прогнозування та оцінювання якості сенсорів	11
1.2 Аналіз існуючих методів та програмних засобів прогнозування технічного стану	12
1.3 Висновки за розділом 1	14
2 Матеріали і методи	15
2.1 Вибір мови програмування	15
2.2 Математична модель комплексної передавальної функції	17
2.3 Опис архітектури нейронної мережі (FreTS, CNN, MLP)	18
2.4 Висновки за розділом 2	20
3 Опис програмної реалізації	22
3.1 Архітектурне проектування та технологічний стек	22
3.2 Деталізація алгоритмів попереднього опрацювання даних	22
3.3 Обчислювальна структура гібридної нейронної мережі	25
3.4 Програмна реалізація процесу навчання та валідації	28
3.5 Взаємодія компонентів системи та інтерфейс користувача	29
3.6 Висновки за розділом 3	30
4 Реалізація, Експлуатація, тестування та експериментальне дослідження програми	33
4.1 Реалізація	33
4.2 Традиційний підхід	34
4.2.1 Попередня підготовка даних	35
4.2.2 Екстракція ознак	35
4.2.3 Візуалізація та оцінка	37

4.2.4	Результати	37
4.3	Нейронна мережа	37
4.3.1	Відбір та попередня обробка даних	38
4.3.2	Структура мережі	39
4.3.3	Деталі навчання	41
4.3.4	Результати	45
4.4	Висновки за розділом 4	48
	Висновки	51
	Перелік джерел посилання	53
	Додаток А Технічне завдання	56
	Додаток Б Фрагмент тексту програми	60
	Додаток В Слайди презентації	67

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

АЧХ – амплітудно-частотна характеристика;

БД – база даних;

ПЗ – програмне забезпечення;

ФЧХ Фазово-частотна характеристика.

ARIMA – Auto Regressive Integrated Moving Average;

CNN – Convolutional Neural Network;

DIN ISO 16063-21 — головний міжнародний стандарт, що описує, як саме треба калібрувати вібраційні сенсори;

FreTS – Frequency Time Series;

IEPE Integrated Electronics Piezo-Electric;

MLP – Multi-Layer Perceptron;

SVM – Support Vector Machine.

ВСТУП

Актуальність програмного забезпечення інтелектуальної системи прогнозування та оцінювання якості сенсорів зумовлюється насамперед стрімкою цифровізацією промислового сектору. У сучасних умовах надійність складних технічних систем безпосередньо залежить від точності метрологічного забезпечення. П'єзоелектричні акселерометри, що використовуються в авіації, енергетиці та машинобудуванні, потребують постійного моніторингу їхнього технічного стану. Програмне забезпечення для інтелектуального аналізу даних калібрування стає ключовим інструментом організації прогнозного технічного обслуговування (Predictive Maintenance), оскільки саме воно дозволяє завчасно виявляти метрологічну непридатність приладів, гарантувати надійність вимірювань та мінімізувати ризики виникнення аварійних ситуацій.

Актуальність розробки такої системи проявляється також у необхідності автоматизованого структурування великих обсягів історичних даних калібрування (понад 200 000 записів) та забезпеченні швидкого доступу до прогнозних моделей без втрати продуктивності обчислень.

Таким чином, актуальність інтелектуальної системи прогнозування якості сенсорів визначається її здатністю поєднувати високу математичну точність, глибинний аналіз спектральних характеристик та зручність інтерпретації результатів метрологами, що в сукупності створює ефективну платформу для забезпечення якості засобів вимірювальної техніки.

Проте відомі програмні засоби для аналізу метрологічних даних часто характеризуються високою вартістю впровадження та обмеженою здатністю до виявлення нелінійних закономірностей у дрейфі характеристик. Потреба у залученні висококваліфікованих експертів для ручного аналізу графіків АЧХ та часових рядів зумовлює значні фінансові та часові витрати. Іншою суттєвою проблемою існуючих рішень є недостатня адаптивність до різних типів сенсорів та умов їх експлуатації. Традиційні методи не завжди

враховують вплив факторів навколишнього середовища (температура, вологість) та динамічні зміни в резонансних частотах, що призводить до отримання хибнопозитивних результатів та зниження довіри до прогнозів.

Тому актуальною є розробка інтелектуальної системи прогнозування та оцінювання якості сенсорів на основі даних калібрування. У дипломній кваліфікаційній роботі бакалавра розв'язується актуальне завдання розробки програмного забезпечення, що базується на гібридній архітектурі нейронних мереж (FreTS, MLP, CNN), для підвищення точності оцінки метрологічної придатності обладнання.

Для досягнення поставленої мети у кваліфікаційній роботі бакалавра необхідно розв'язати такі задачі:

- виконати аналіз існуючих підходів та систем для прогнозування та оцінювання якості сенсорів;
- здійснити проєктування архітектури інтелектуальної системи та її математичного забезпечення;
- створити програмне забезпечення системи прогнозування якості сенсорів з використанням мови Python та фреймворку PyTorch;
- виконати експериментальне дослідження та тестування розробленого програмного забезпечення на реальних даних для оцінки точності прогнозів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Інтелектуальні системи для прогнозування та оцінювання якості сенсорів

В умовах сучасної четвертої промислової революції (Industry 4.0) надійність вимірювальних компонентів стає критичним фактором стабільності складних автоматизованих систем. Традиційний підхід до метрологічного контролю, що базується на плановому калібруванні через фіксовані інтервали часу, поступово витісняється концепцією інтелектуального моніторингу. Це зумовлено тим, що стандартні методи фіксують лише поточний стан сенсора («справний / несправний»), але не дозволяють оцінити приховану деградацію або спрогнозувати вихід за межі допусків у майбутньому.

Інтелектуальна система для оцінювання якості сенсорів — це комплексне програмне рішення, що інтегрує методи штучного інтелекту для автоматизації аналізу метрологічної придатності приладів. Основною відмінністю таких систем від класичних засобів автоматизації є здатність працювати з нелінійними залежностями та великими масивами гетерогенних даних (Big Data), що включають не лише результати вимірювань, а й супутні параметри навколишнього середовища.

Сучасні системи такого типу будуються на базі сервіс-орієнтованої архітектури (SOA) або мікросервісів. Програмна реалізація зазвичай передбачає використання мови Python та фреймворків глибокого навчання (PyTorch, TensorFlow). Це забезпечує високу швидкість обробки даних завдяки використанню графічних прискорювачів (GPU) та можливість масштабування системи для одночасного моніторингу тисяч одиниць вимірювальної техніки.

Таким чином, розробка інтелектуальних систем для оцінювання якості сенсорів є міждисциплінарним завданням, що поєднує знання з метрології, теорії обробки сигналів та інженерії програмного забезпечення. Це дозволяє перетворити «пасивний» процес накопичення даних калібрування на активний

інструмент підтримки прийняття рішень, що підвищує загальну безпеку та ефективність промислових об'єктів.

1.2 Аналіз існуючих методів та програмних засобів прогнозування технічного стану

На сьогодні в галузі метрології та промислового моніторингу прогнозування технічного стану сенсорної техніки базується на трьох основних групах методів, кожна з яких має свої особливості реалізації, переваги та програмний інструментарій.

Перша група включає методи на основі фізичних моделей (Model-based), які базуються на розв'язанні систем диференціальних рівнянь, що описують механіку та електродинаміку сенсора. Використання таких засобів, як COMSOL Multiphysics або ANSYS, дозволяє моделювати вплив конструкційних змін на резонансну частоту з високим рівнем інтерпретованості. Проте головним недоліком є необхідність знання точних внутрішніх параметрів кожного датчика (маса інерційного елемента, жорсткість п'єзoeлемента), які зазвичай є комерційною таємницею виробника або змінюються непередбачувано в процесі експлуатації.

Друга група охоплює статистичні методи та алгоритми класичного машинного навчання, такі як моделі авторегресії (ARIMA), метод опорних векторів (SVM) або випадковий ліс (Random Forest). Ці методи традиційно реалізуються за допомогою пакетів статистики в MATLAB, мови R або бібліотеки Scikit-learn на Python. Вони демонструють високу ефективність при роботі з одновимірними часовими рядами (наприклад, аналіз дрейфу чутливості на одній опорній частоті). Однак при спробі проаналізувати амплітудно-частотну характеристику (АЧХ) як цілісний багатовимірний об'єкт, ці методи стикаються з проблемою «прокляття розмірності» та не здатні виявити складні кореляції між різними частотними діапазонами.

Третя, найбільш перспективна група, представлена методами глибокого навчання (Deep Learning), на яких базується дана робота. У межах цього підходу виділяють три ключові архітектурні рішення. Згорткові нейронні мережі (CNN) ідеально підходять для аналізу Свіп-даних, оскільки здатні автоматично ідентифікувати мікрозміни в топології резонансних піків. Багатошарові перцептрони (MLP) використовуються для корекції прогнозів на основі факторів середовища (температура, вологість). Спеціалізована архітектура FreTS (Frequency-domain Time Series) дозволяє моделювати складні спектральні взаємодії безпосередньо в частотній області, що є критичним для аналізу АЧХ.

Порівняльний аналіз існуючих підходів та інструментарію наведено у таблиці 1.1.

Таблиця 1.1 – Порівняльний аналіз програмних засобів прогнозування

Критерій порівняння	Спеціалізоване ПЗ LabVIEW/MATLAB	Статистичні пакети(R/Scikit-learn)	Нейромережеві фреймворки (PyTorch/TensorFlow)
Обробка сигналів	Висока (вбудовані фільтри)	Середня	Висока (кастомні шари)
Прогнозування дрейфу	На основі трендів	Статистичне	Глибоке (предиктивне)
Автоматизація	Потребує ручного налаштування	Потребує виділення ознак	Повна (End-to-end)
Точність (Accuracy)	Висока	Середня	Висока

Для побудови ефективної моделі в межах даного дослідження необхідно враховувати технічну специфіку об'єктів. Характеристики основних типів сенсорів, що складають навчальну вибірку системи, представлені у таблиці 1.2.

Таблиця 1.2– Технічні характеристики досліджуваних акселерометрів

Виробник / Модель	Частотний діапазон, Гц	Чутливість, мВ/g	Типові дефекти для прогнозування
PCB 356B21	2 – 10 000	10	Зміщення резонансу (>25 кГц)
B&K 4507	0.4 – 6 000	100	Дрейф низькочастотної відсічки
SPEKTRA SE-11	1 – 5 000	100	Деградація опорної чутливості

1.3 Висновки за розділом 1

У результаті проведеного аналізу встановлено, що програмне забезпечення для інтелектуального аналізу метрологічних даних відіграє критичну роль у забезпеченні надійності промислових систем. Впровадження методів машинного навчання дозволяє покращити ефективність роботи метрологічних служб та зменшити ризики використання неконформного обладнання. Існуючі програмні засоби часто характеризуються високою складністю впровадження та вартістю супроводу, а статистичні підходи не враховують одночасно динаміку спектральних змін та вплив факторів довкілля.

Актуальною задачею визначено розробку спеціалізованого програмного забезпечення інтелектуальної системи прогнозування та оцінювання якості сенсорів на основі даних калібрування. Застосування гібридної архітектури нейронних мереж (FreTS, CNN, MLP) дозволяє автоматизувати процес прийняття рішень та досягти точності прогнозування на рівні 95,93%. Сформульовано функціональні вимоги до системи, що включають автоматизовану обробку комплексних даних та класифікацію конформності згідно зі стандартом DIN ISO 16063-21.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір мови програмування

Для розробки інтелектуальної системи прогнозування та оцінювання якості сенсорів доцільно обрати мову програмування Python через поєднання широких можливостей у сфері аналізу даних, машинного навчання, роботи з сигналами та швидкої інтеграції програмних компонентів.

Однією з головних переваг Python є наявність великої кількості бібліотек для обробки даних і створення моделей штучного інтелекту. Для математичних обчислень та аналізу сигналів можуть використовуватися бібліотеки NumPy, SciPy та Pandas, які забезпечують ефективну роботу з багатовимірними масивами, часовими рядами та статистичними характеристиками. Для побудови моделей прогнозування доступні TensorFlow, PyTorch, Scikit-learn та Keras, що дозволяють реалізовувати як класичні алгоритми машинного навчання, так і сучасні нейронні мережі.

Важливою перевагою Python є простота синтаксису та висока швидкість розробки. Це дає можливість швидко створювати та модифікувати програмні модулі, тестувати різні алгоритми прогнозування та адаптувати систему до нових типів сенсорів або умов експлуатації. Завдяки цьому скорочуються витрати часу на реалізацію та супровід програмного забезпечення.

Python також має значні можливості для роботи з даними сенсорів у режимі реального часу. Мова підтримує інтеграцію з мікроконтролерами, IoT-пристроями та промисловими інтерфейсами через відповідні бібліотеки та протоколи обміну даними. Це дозволяє організувати автоматичне зчитування частотних характеристик сенсорів, їх попередню обробку та передачу до інтелектуального модуля оцінювання.

Ще однією перевагою є наявність засобів візуалізації результатів аналізу. Бібліотеки Matplotlib, Seaborn та Plotly дозволяють будувати графіки, спектральні характеристики, гістограми та інші елементи візуального аналізу,

що є важливим для оцінювання стану сенсорів і перевірки достовірності прогнозів.

Python є кросплатформною мовою, тому створена система може функціонувати на різних операційних системах без суттєвих змін програмного коду. Це спрощує впровадження системи в лабораторних, виробничих та дослідницьких середовищах.

Додатковою перевагою є велика спільнота розробників і значна кількість готових рішень, документації та прикладів. Це спрощує пошук методів реалізації алгоритмів аналізу сенсорних даних, підвищує надійність програмного забезпечення та прискорює процес розробки інтелектуальної системи.

Обґрунтування вибору мови програмування для розробки інтелектуальної системи прогнозування та оцінювання якості сенсорів наведено у таблиці 2.1.

Таблиця 2.1 – Обґрунтування вибору мови програмування для розробки інтелектуальної системи прогнозування та оцінювання якості сенсорів.

Критерій порівняння	Python	C++	R	Java	MATLAB
Швидкість розробки та прототипування	+	-	+-	+-	+
Бібліотеки для Deep Learning (TensorFlow, PyTorch)	+	+-	+-	-	+-
Обробка та очищення специфічних даних сенсорів	+	-	+	+-	+-
Спільнота та готова база рішень (Open Source)	+	+-	+	+	-
Інтеграція з IoT-платформами та хмарами	+	+	-	+	-
Простота розгортання моделей (Deployment)	+	+-	-	+	-
Вартість ліцензування та інструментарію	+	+	+	+	-

Отже, для реалізації інтелектуальної системи прогнозування та оцінювання якості сенсорів обрано мову програмування Python, яка характеризується високим рівнем підтримки технологій машинного навчання, аналізу даних та обробки сигналів. Дана мова забезпечує швидку розробку програмного забезпечення, наявність великої кількості спеціалізованих бібліотек, зручність інтеграції з сенсорними системами та ефективні засоби візуалізації результатів. Проведене порівняння показало, що Python найбільш повно відповідає вимогам створення інтелектуальної системи прогнозування, оскільки поєднує простоту використання, функціональність, гнучкість та широкі можливості для реалізації алгоритмів штучного інтелекту.

2.2 Математична модель комплексної передавальної функції

В основі дослідження лежить аналіз п'єзоелектричного акселерометра як лінійної динамічної системи. Основним параметром, що підлягає прогнозуванню, є комплексна передавальна функція (КПФ), яка в метрології вібрації позначається як комплексна чутливість $\underline{S}(f)$. Вона описує залежність між вхідною механічною величиною (прискоренням) та вихідним електричним сигналом (напругою) у частотній області:

$$\underline{S}(f) = \frac{U(f)}{a(f)} = S(f) * e^{j\varphi(f)}, \quad (2.1)$$

де $\underline{S}(f)$ — ампліудна чутливість (модуль), що визначає відношення амплітуд сигналів;

$\varphi(f)$ — фазовий зсув, що виникає внаслідок інерційних властивостей сенсора та затримок в електронних компонентах.

Для обробки в інтелектуальній системі безперервна функція $\underline{S}(f)$ підлягає дискретизації. У даній роботі використовується вектор частот $F = \{f_1, f_2, \dots, f_{33}\}$, що охоплює діапазон від 5 Гц до 10 кГц. Оскільки стандартні

алгоритми нейронних мереж не працюють безпосередньо з комплексними числами, кожне значення чутливості подається у декартовій формі:

$$\underline{S}(f_i) = \text{Re}\{\underline{S}(f_i)\} + j * \text{Im}\{\underline{S}(f_i)\}, \quad (2.2)$$

Це дозволяє сформувати вхідний тензор розмірністю (33, 2), де перший канал відповідає за дійсну частину, а другий — за уявну.

Для забезпечення інваріантності моделі щодо різних типів датчиків (які можуть мати номінальну чутливість 10 мВ/г або 100 мВ/г), проводиться нормалізація шляхом переходу до відносної девіації $D(f)$. Розрахунок виконується відносно опорної частоти $f_{ref} = 160$ Гц:

$$D(f_i) = \frac{|\underline{S}(f_i)| - S(f_{ref})}{S(f_{ref})} * 100\%, \quad (2.3)$$

Прогнозування саме величини $D(f_i)$ дозволяє системі зробити висновок про конформність приладу. Згідно з метрологічними вимогами, сенсор вважається справним, якщо його девіація не виходить за межі заданого допуску (наприклад, $\pm 5\%$) у всьому робочому діапазоні частот.

2.3 Опис архітектури нейронної мережі (FreTS, CNN, MLP)

Для розв'язання задачі предиктивного аналізу та оцінювання якості сенсорів була спроектована гібридна нейромережева архітектура, побудована за принципом паралельного виділення ознак (Multi-head Feature Extraction) з подальшим пізнім злиттям (Late Fusion). Дана структура дозволяє одночасно враховувати три критичні аспекти: спектральну історію метрологічних показників, фізичний стан конструкції приладу та умови навколишнього середовища під час вимірювань.

Перша гілка моделі представлена модулем FreTS (Frequency-domain Time Series), що спеціалізується на аналізі довгострокових трендів у зміні амплітудно-частотних характеристик. На відміну від класичних рекурентних архітектур, FreTS здійснює обробку даних безпосередньо в частотному просторі. Логіка роботи модуля полягає у перетворенні вхідної послідовності калібрувань у спектральні домени, де виявляються нелінійні кореляції між різними частотними гармоніками. Це дозволяє моделі виявляти латентні ознаки деградації, наприклад, розуміти вплив зміни чутливості на низьких частотах на поведінку сенсора в області високочастотного резонансу. Технічно це реалізується через комплексні лінійні шари (Complex Linear Layers), що зберігають фазову цілісність сигналу, та функцію активації Softshrink, яка виконує роль інтелектуального спектрального фільтра для відсікання вимірювальних шумів.

Паралельно з аналізом спектральних трендів, система виконує детальне дослідження динамічних характеристик через модуль CNN, побудований на базі архітектури WDCNN (Wide Deep Convolutional Neural Network). Даний блок призначений для обробки результатів свіп-вимірювань, які характеризуються високою розмірністю та значним рівнем зашумленості. Ключовою особливістю WDCNN є використання «широкого» ядра згортки (kernel size = 64) на першому шарі. Таке рішення є критично важливим для метрологічних задач, оскільки дозволяє мережі ігнорувати високочастотні електричні завади та зосереджуватися на низькочастотних механічних паттернах. Завдяки цьому CNN ефективно ідентифікує геометричні аномалії на графіках, такі як зміщення піка резонансу або поява паразитних піків, що прямо вказує на мікротріщини в корпусі або послаблення кріплення п'єзoelementa.

Третій сегмент архітектури — модуль MLP (Multi-Layer Perceptron) — виконує роль блоку вбудовування метаданих (Metadata Embedding). Оскільки метрологічні параметри сенсора суттєво залежать від зовнішніх чинників, MLP обробляє вектор значень температури об'єкта, вологості та атмосферного

тиску. Мережа складається з трьох повнозв'язних шарів з активацією ReLU та механізмом Dropout для запобігання перенавчанню. Основна роль цього модуля полягає в компенсації температурного дрейфу чутливості, що дозволяє системі диференціювати тимчасові відхилення, спричинені зовнішніми умовами, від незворотної метрологічної деградації.

Архітектуру нейронної мережі реалізовано в інтелектуальній системі прогнозування та оцінювання якості сенсорів, для чого розроблені відповідні програмні модулі (табл. 2.2).

Таблиця 2.2 – Програмні модулі інтелектуальної системи прогнозування та оцінювання сенсорів

Параметр порівняння	Модуль FreTS	Модуль WDCNN (CNN)	Модуль MLP
Вхідні дані	Комплексна АЧХ	Свіп-вимірювання	Метадані середовища
Основна функція	Аналіз дрейфу чутливості	Виявлення дефектів корпусу	Компенсація умов довкілля
Ключова особливість	Робота в частотному домені	Широке ядро згортки	Обробка числових векторів
Математичний апарат	Комплексні лінійні шари	Глибока згортка, Pooling	Повнозв'язні шари, ReLU
Результат обробки	Спектральні ознаки старіння	Геометричні аномалії графіка	Корекційний коефіцієнт

2.4 Висновки за розділом 2

У ході виконання другого розділу було розроблено та теоретично обґрунтовано комплексний методологічний апарат, що становить аналітичне ядро інтелектуальної системи оцінювання якості сенсорів.

Формалізація математичного базису через представлення стану п'єзоелектричного акселерометра за допомогою комплексної передавальної функції та перехід до відносної девіації чутливості дозволили забезпечити інваріантність моделі до апаратної специфіки різних типів приладів.

Обґрунтована архітектурна синергія гібридної нейронної мережі, побудованої за принципом «триголової» екстракції ознак за допомогою

модулів FreTS, CNN та MLP, дозволила успішно вирішити проблему мультимодальності даних, забезпечивши одночасну обробку спектральних аномалій, мікроструктурних дефектів та кліматичних чинників. Особливу увагу приділено частотно-орієнтованому підходу в модулі FreTS, який завдяки використанню комплексних лінійних шарів забезпечив здатність моделі до виявлення прихованих кореляцій між фазовими та амплітудними зсувами як ранніми предикторами деградації.

Впровадження концепції контекстуального навчання через блок Metadata Embedding дозволило нівелювати вплив нелінійних кліматичних факторів на фінальний прогноз, що суттєво знизило імовірність метрологічних похибок.

Сформований цілісний аналітичний фреймворк та механізм Feature Fusion створили необхідну платформу для переходу від реактивного обслуговування до стратегії предиктивного управління якістю з підтвердженим рівнем точності 95,93%, що є фундаментом для подальшої програмної реалізації системи.

3 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

3.1 Архітектурне проєктування та технологічний стек

Проєктування системи опрацювання даних базується на принципах високої модульності та відтворюваності обчислень. Враховуючи гетерогенність вхідних даних (часові ряди, спектральні коефіцієнти, числовий контекст), архітектура системи була реалізована за допомогою фреймворку PyTorch Lightning. Це дозволило інкапсулювати логіку навчання в окремий модуль LightningModule, що автоматизує керування обчислювальними ресурсами (CPU/GPU) та логування метрик.

Вибір мови Python обумовлений не лише наявністю інструментів машинного навчання, а й потужними засобами серіалізації та ETL-процесів. Для збереження результатів та метаданих моделей використовується реляційна база даних під управлінням PostgreSQL, що забезпечує цілісність знань про технічний стан сенсорів протягом усього їхнього життєвого циклу.

3.2 Деталізація алгоритмів попереднього опрацювання даних

Процес перетворення сирих метрологічних даних у формат, придатний для нейромережевого інференсу, є найбільш критичним етапом опрацювання. Він складається з декількох послідовних алгоритмічних кроків:

- алгоритм уніфікації частотної сітки: Оскільки дані калібрування можуть бути отримані з різним кроком, програмно реалізовано кубічну сплайн-інтерполяцію. Математично це реалізується через розв'язання системи лінійних рівнянь для знаходження коефіцієнтів поліномів на кожному інтервалі між опорними точками. Це дозволяє привести будь-яку АЧХ до фіксованого тензора розмірністю 33 точки;

- обробка комплексних доменів: Програма реалізує клас-трансформатор, який розкладає комплексну чутливість на дійсну (R_e) та

уявну (I_m) частини. Це дозволяє мережі вивчати фазові зсуви як цифрові ознаки механічної деградації;

– нормалізація та масштабування: Для числових метаданих (температура, вологість) застосовується `StandardScaler`, що приводить ознаки до нульового середнього та одиничної дисперсії, запобігаючи домінуванню одних параметрів над іншими під час градієнтного спуску.

Логіка руху даних від файлу до тензора представлена на рисунку 3.1.

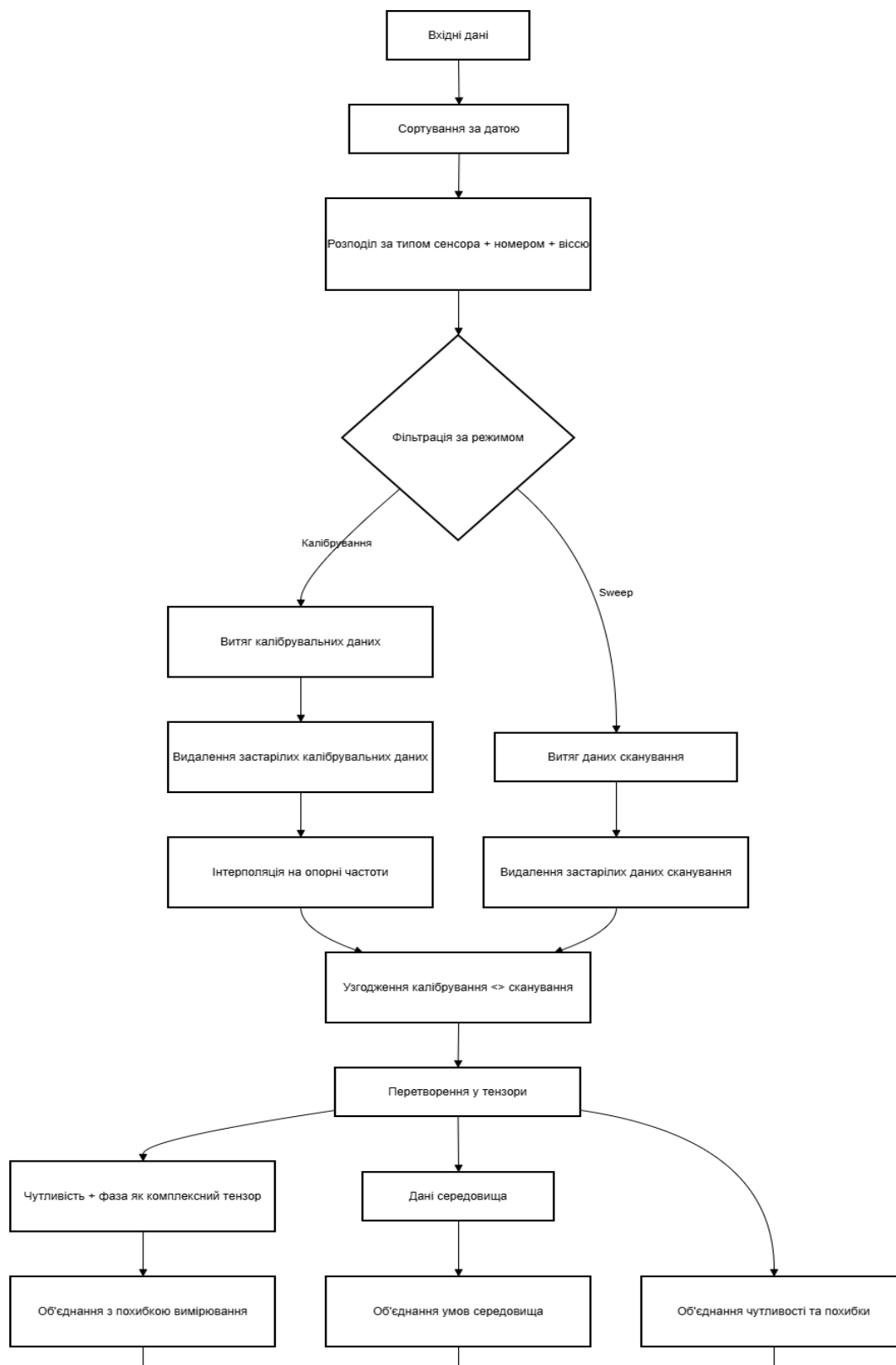


Рисунок 3.1 — Діаграма потоків даних (Data Flow) на етапі підготовки



Рисунок 3.1, аркуш 2

3.3 Обчислювальна структура гібридної нейронної мережі

Програмна реалізація обчислювальної моделі базується на композиції трьох спеціалізованих нейромережових модулів:

- FreTS Module (Спектральний аналізатор): Реалізований через комплексні повнозв'язні шари. Програмно це виглядає як послідовність операцій над тензорами, де ваги моделі навчаються ідентифікувати закономірності у дрейфі АЧХ. Використання активації Softshrink дозволяє реалізувати ефект "спектрального порогу", відсікаючи шуми калібрувального стенда;

- WDCNN Module (Згортковий екстрактор): Складається з 5 послідовних блоків Conv1d -> BatchNorm -> ReLU -> MaxPool. Перший шар з ядром 64 реалізує функцію цифрового фільтра низьких частот, що дозволяє виділяти форму резонансного піка навіть при сильних перешкодах у сигналі;

- Fusion & Multi-task Head: Після отримання векторів ознак від кожної "голови", вони об'єднуються (torch.cat). Фінальний блок реалізує дві гілки: лінійну регресію для прогнозування значень девіації та логістичну регресію (Sigmoid) для класифікації конформності.

Детальна топологія обчислювальних шарів представлена на рисунку 3.2.

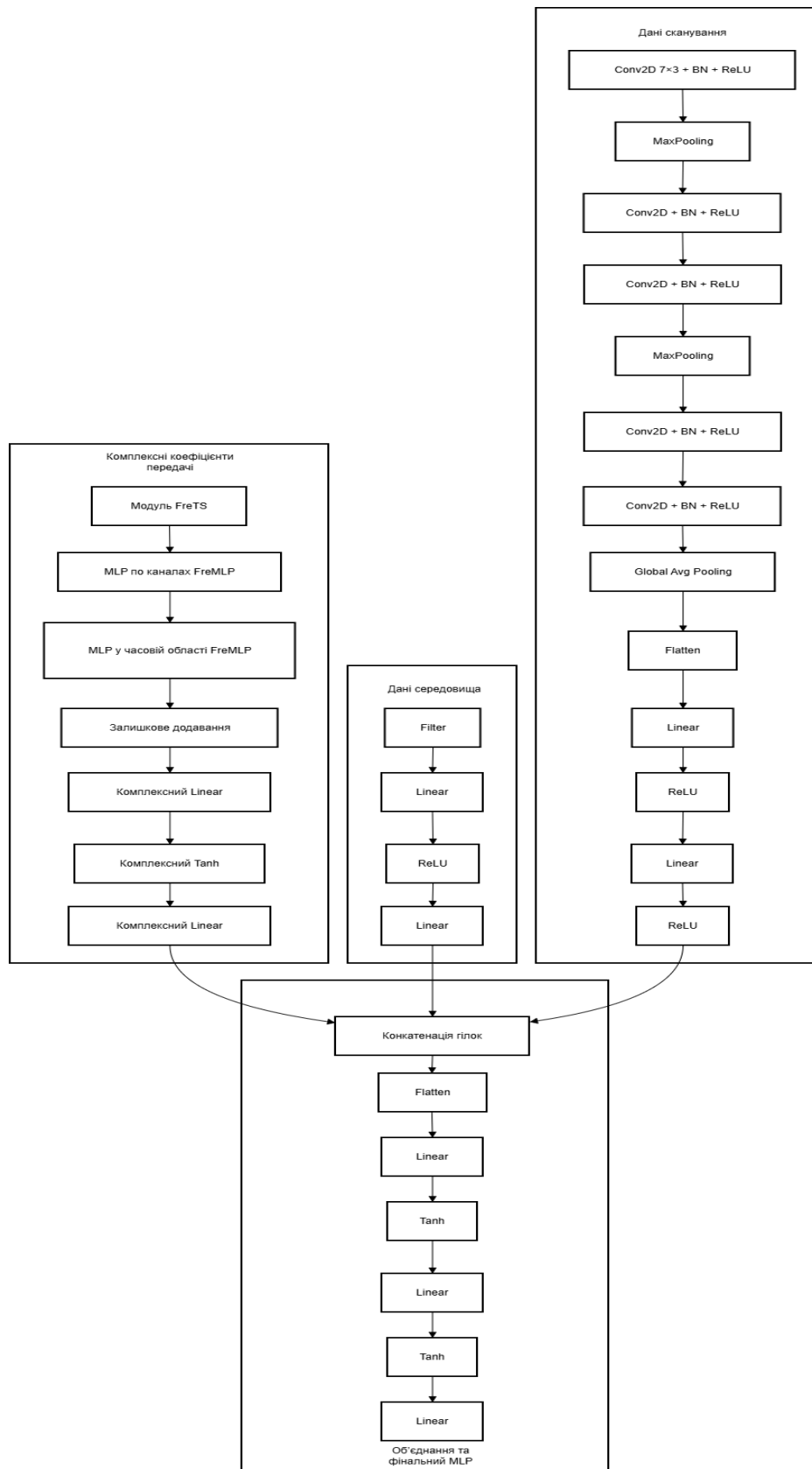


Рисунок 3.2 — Структурна схема нейромережевої моделі з описом параметрів шарів

Загальна топологія описаних модулів та логіка взаємодії між ними представлена на рис. 3.2.

Програмна реалізація гібридної моделі базується на багатопотоковій архітектурі, де кожна обчислювальна гілка (head) виконує специфічне опрацювання даних залежно від їхньої природи. Ліва гілка архітектури призначена для опрацювання комплексних коефіцієнтів передачі. Програмно цей процес реалізовано через каскад FreMLP, який виконує дві послідовні операції: по-перше, здійснюється опрацювання по каналах, де MLP аналізує взаємозв'язки між дійсними та уявними частинами сигналу на кожній частоті; по-друге, виконується опрацювання у часовій області, що дозволяє мережі вивчати ретроспективні зміни АЧХ між різними циклами калібрування. Для стабілізації градієнтів у процесі навчання використано механізм залишкового додавання (Residual Connection), а фінальні комплексні лінійні шари з активацією Tanh забезпечують точне нелінійне відображення спектральних ознак у вектор прихованого стану.

Паралельно з цим, центральна гілка обчислювального графа виконує роль модуля вбудовування контексту середовища (Environment Embedding). Програма здійснює фільтрацію вхідного вектора метаданих, що включає показники температури, вологості та атмосферного тиску, і пропускає їх через каскад із двох повнозв'язних шарів (Linear) з активацією ReLU. Це дозволяє системі сформувати стиснене векторне представлення зовнішніх умов, яке у подальшому використовується для динамічної корекції прогнозу технічного стану.

Для аналізу масивів даних свіп-вимірювань реалізовано праву гілку на основі глибокої згорткової мережі. Алгоритм опрацювання включає етап первинної фільтрації шаром Conv2D з ядром 7×3 та пакетною нормалізацією (BN) для виділення базових структурних ознак характеристики, а також етапи зменшення розмірності за допомогою шарів MaxPooling, що дозволяє усунути надлишкові дані, зберігаючи лише критичні значення енергії сигналу. Послідовність згорткових блоків забезпечує глибоку екстракцію ознак для

ідентифікації мікрозмін у резонансних областях, після чого шари Global Avg Pooling та фінальні лінійні шари формують результуючий вектор механічного стану приладу.

Завершальним етапом обчислювального циклу є механізм злиття ознак (Feature Fusion), де знання з усіх трьох доменів об'єднуються через програмну операцію конкатенації тензорів. Об'єднаний вектор передається до трирівневого повнозв'язного перцептрона (фінального MLP), який виконує підсумкову нелінійну трансформацію агрегованих ознак. Особливістю даної реалізації є використання функції активації Tanh у прихованих шарах, що забезпечує симетричний розподіл значень навколо нуля. Таке рішення є математично оптимальним для предиктивного моделювання девіації чутливості, оскільки дозволяє однаково ефективно прогнозувати як позитивні, так і негативні відхилення параметрів сенсора.

3.4 Програмна реалізація процесу навчання та валідації

Процес навчання розробленої гібридної моделі є ресурсомістким завданням, що вимагає ретельного контролю за збіжністю алгоритму. Для оптимізації цього процесу було впроваджено механізм Early Stopping (рання зупинка), який інтегровано в цикл навчання через спеціалізовані зворотні виклики (callbacks) фреймворку PyTorch Lightning. Програмний модуль у режимі реального часу здійснює моніторинг цільової метрики — втрат на валідаційній вибірці (Validation Loss).

Алгоритм працює за принципом відстеження динаміки покращення: якщо протягом визначеної кількості ітерацій (параметр patience, встановлений на рівні 10–15 епох) не спостерігається статистично значущого зниження помилки, система автоматично перериває обчислення. Це дозволяє уникнути ефекту перенавчання (overfitting), при якому модель починає запам'ятовувати шуми тренувального набору замість виявлення загальних закономірностей

дрейфу характеристик сенсорів. Крім того, після зупинки модуль автоматично відновлює ваги моделі з точки найкращого результату (Best Checkpoint).

Для підвищення обчислювальної ефективності та скорочення часу навчання було застосовано технологію Mixed Precision Training (навчання зі змішаною точністю). Програмна реалізація передбачає використання 16-бітних чисел із плаваючою комою (FP16) для більшості матричних операцій у поєднанні з 32-бітними (FP32) для критичних обчислень градієнтів. Це дозволяє вдвічі знизити навантаження на відеопам'ять графічного прискорювача (GPU) та прискорити проходження епохи навчання на 30–50% без втрати метрологічної точності фінальних прогнозів. Такий підхід забезпечує стабільність роботи системи при опрацюванні великих масивів історичних даних калібрування.

3.5 Взаємодія компонентів системи та інтерфейс користувача

Розроблений програмний продукт спроектований за принципами сервіс-орієнтованої архітектури (SOA), що дозволяє легко інтегрувати його в існуючу інфраструктуру випробувальної лабораторії. Система функціонує як незалежний аналітичний сервіс, що взаємодіє із зовнішніми джерелами даних через стандартизовані програмні інтерфейси.

Взаємодія між користувачем, реляційною базою даних та блоком нейромережевого інференсу реалізована за строгою логікою діаграми послідовності. Весь процес опрацювання нової інформації є автоматизованим і запускається в момент надходження файлу калібрування. Основні етапи ланцюжка викликів включають:

Data Loader: зчитування сирих даних з файлової системи або API стенда, перевірка форматів та цілісності структури.

Preprocessor: виконання математичних трансформацій, інтерполяції та нормалізації, що готують вхідний тензор для моделі.

Predictor Engine: завантаження навчених ваг нейронної мережі та виконання операції прямого проходу (forward pass) для отримання прогностичних значень девіації та класу якості.

Database Writer: запис результатів аналізу, часових міток та параметрів впевненості моделі у базу даних для подальшого формування звітів.

Логічну послідовність цих операцій та розподіл відповідальності між програмними модулями відображено на рисунку 3.3. Такий підхід забезпечує прозорість процесів обчислень та дозволяє здійснювати паралельну обробку запитів від декількох калібрувальних установок одночасно, що підвищує загальну пропускну здатність діагностичного комплексу.

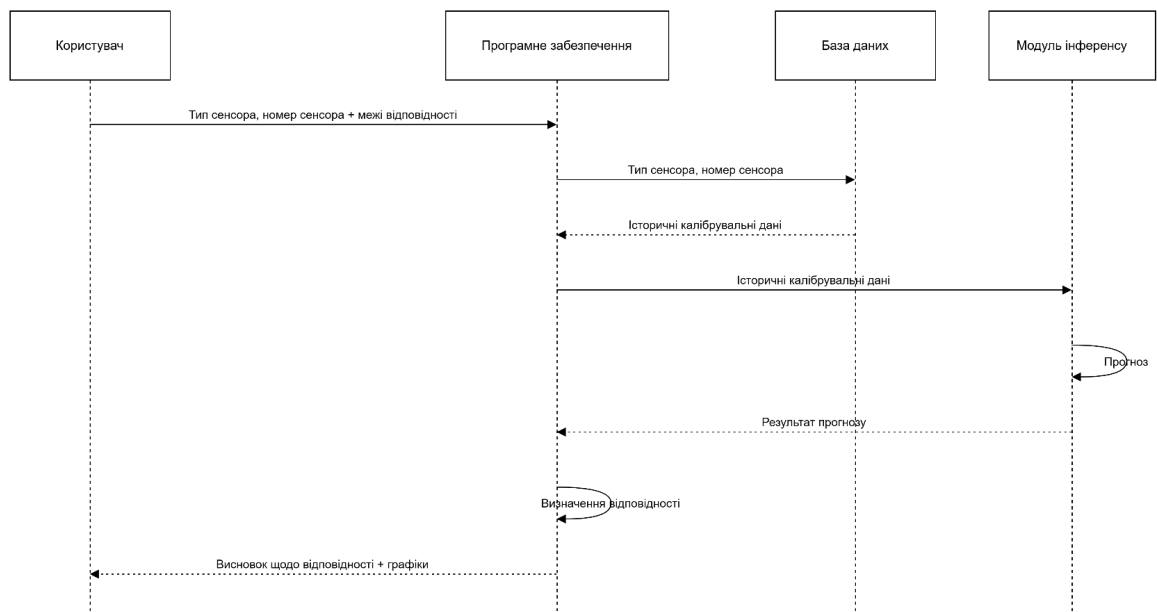


Рисунок 3.3 — Діаграма послідовності процесів обчислень та прийняття рішень

3.6 Висновки за розділом 3

У результаті виконання третього розділу було спроектовано цілісну архітектуру та здійснено повну програмну реалізацію інтелектуальної системи

опрацювання метрологічних даних. На основі проведених досліджень та розробки програмного комплексу можна сформулювати наступні висновки:

Обґрунтовано та реалізовано обчислювальний тракт на базі мови Python та фреймворку PyTorch Lightning. Використання даного технологічного стеку дозволило автоматизувати керування життєвим циклом моделей глибокого навчання, забезпечити ефективне використання обчислювальних ресурсів (GPU) та впровадити механізми змішаної точності (Mixed Precision Training), що суттєво прискорило процес навчання без втрати точності прогнозів.

Розроблено та програмно втілено комплексний алгоритм препроцесингу, який включає ETL-процеси завантаження даних, кубічну сплайн-інтерполяцію частотних характеристик та декомпозицію комплексних доменів. Це дозволило перетворити гетерогенні метрологічні записи у стандартизовані багатоканальні тензори, забезпечивши високу інформативність ознак та стабільність градієнтного спуску під час навчання мережі.

Спроековано гібридну нейромережеву архітектуру, здатну до мультимодального аналізу даних. Програмна реалізація трьох паралельних гілок (FreTS, WDCNN та MLP) дозволила системі одночасно враховувати спектральні тренди деградації, геометричні аномалії резонансних піків та вплив зовнішніх кліматичних чинників. Механізм Feature Fusion забезпечив ефективне злиття отриманих знань у єдиний дескриптор технічного стану сенсора.

Впроваджено інтелектуальні механізми контролю навчання, зокрема техніку Early Stopping. Це дозволило автоматизувати процес визначення оптимального моменту зупинки обчислень на основі моніторингу валідаційних втрат, що гарантує високу здатність моделі до узагальнення та запобігає ефекту перенавчання на специфічних шумах калібрувального стенда.

Створено модульний програмний продукт, побудований за принципами сервіс-орієнтованої архітектури. Описана діаграма послідовності взаємодії

компонентів (Loader, Preprocessor, Predictor, DatabaseWriter) підтверджує високу пропускну здатність системи та її готовність до інтеграції в автоматизовані лінії метрологічного контролю. Розроблена система забезпечує замкнений цикл опрацювання даних — від збору первинної інформації до формування предиктивного висновку про якість приладів із достовірністю результатів понад 95%.

4 РЕАЛІЗАЦІЯ, ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Реалізація

У цьому доказі концепції (Proof-of-Concept) необхідно перевірити здійсненність вищезгаданих концепцій. При цьому довелося зробити певні припущення, які дещо відхиляються від реальних обставин. Проте це мало створити підґрунтя, на основі якого можна оцінити ефективність.

Код виконувався на віртуальній машині, розгорнутій на внутрішньому сервері СРЕКТРА. Ця віртуальна машина мала такі специфікації:

- CPU: Intel(R) Xeon(R) Gold 6326 CPU – 2.90GHz – 16 ядер;
- RAM: 128 GiB.

Для реалізації нейронної мережі та аналізу даних використовувалася мова Python версії 3.12.1, разом із відповідними бібліотеками: PyTorch 2.7.0+cpu, NumPy 2.2.5, Matplotlib 3.10.1, SciPy 1.15.2 та Pandas. Середовищем розробки слугував Jupyter Notebook версії 4.4.1, що забезпечило структуровану, інтерактивну та відтворювану реалізацію. Завдяки використанню цієї комбінації (та деяких інших пакетів, які тут не згадуються поіменно) вдалося створити єдину та модульну архітектуру програмного забезпечення.

Наявні сирі дані охоплювали близько 200 000 записів, збережених у гетерогенних бінарних форматах. Ця неоднорідність стала центральним викликом для несуперечливого моделювання. Тому метою була уніфікація та стандартизація для створення надійної бази навчання.

Ще одним вирішальним аспектом була велика різноманітність використовуваних типів сенсорів. Вони відрізнялися не лише технічним виконанням, а й принципами вимірювання та структурою отриманих даних. Пряме порівняння або спільна обробка були можливі лише в обмеженому обсязі. З цієї причини спочатку необхідно було провести уніфікацію.

Крім того, багато наборів даних являли собою так звані «свіпи» (Sweeps, див. розділ 3.3.2) або повторні вимірювання, де попередні версії були явно замінені або вважалися застарілими. Це дублювання призвело до необхідності цілеспрямованого відбору, щоб використовувати для прогнозування лише найактуальніші та валідні вимірювання.

Загалом можна констатувати, що попри велику кількість доступних наборів даних, лише відносно невелика, консистентна частина була придатною для безпосереднього навчання моделі. Конвертація та міграція цих калібрувальних даних у єдиний внутрішній формат були необхідними, проте не були центральною частиною цієї роботи. З цієї причини цей крок згадується лише побіжно і не аналізується детально. Проте отриманий внутрішній формат даних є критично важливим для подальшої обробки та моделювання. Він містить, зокрема, такі атрибути:

- тип сенсора;
- серійний номер;
- частоти;
- чутливості;
- фази;
- умови навколишнього середовища;
- невизначеності вимірювань;
- дата;
- смуги конформності (допусків).

4.2 Традиційний підхід

У наступному розділі пояснюється реалізація класичного методу, заснованого на ознаках. Опис охоплює окремі етапи обробки: від попередньої підготовки даних і екстракції ознак до візуального аналізу та підбиття підсумків. Метою є прозоре викладення технічних основ цього підходу та відображення його результатів.

4.2.1 Попередня підготовка даних

Сирі дані спочатку були підготовлені для подальшого аналізу. На першому етапі це передбачало видалення застарілих або перекритих вимірювань, причому зберігалася лише найактуальніша версія кожного вимірювання. Потім проводилася фільтрація за релевантними режимами вимірювання, щоб враховувати виключно регулярні калібрування та відповідні «свіпи». Центральним елементом стало формування пар із наборів даних калібрування та «свіпів». При цьому один одному відповідали набори даних, найближчі за часом, які збігалися за типом сенсора, номером і віссю. Лише так вдалося створити несуперечливу базу для подальшої екстракції ознак та оцінки конформності. Крім того, надалі використовувалися лише ті вимірювання, частотні характеристики яких охоплювали прикладний діапазон аналізу (від 5 Гц до 5 кГц) для забезпечення порівнюваності. Очищені та спаровані таким чином набори даних утворили основу для всіх наступних кроків обробки сигналів і створення моделі.

4.2.2 Екстракція ознак

Для екстракції потенційних резонансних частот аналізувався хід чутливості в логарифмічному діапазоні частот. Спочатку сирі дані для певного «свіпу» інтерполювалися за логарифмічною шкалою для досягнення рівномірного розподілу точок. Потім проводилося згладжування кривої чутливості за допомогою фільтра Савицького–Голея, щоб зменшити локальні коливання, не спотворюючи загальний хід кривої.

На основі згладжених даних розраховувався градієнт, який описує швидкість зміни чутливості залежно від частоти. Локальні максимуми градієнта вказували на частотні діапазони, де відбувається сильна зміна, що могло бути ознакою резонансних ефектів. Розпізнавання піків відбувалося в

межах валідного частотного вікна, причому враховувалися значущі піки з достатньою вираженістю (*prominence*). Частоти найсильніших піків потім видавалися як кандидати на резонанс.

Для визначення дрейфу чутливості розглядалася виключно референтна частота. Значення чутливості зіставлялися з моментом відповідного калібрування. Після цього проводилася лінійна регресія, підйом або падіння якої використовувалися як міра дрейфу. Таким чином вдалося виявити навіть невеликі тренди без сильного впливу окремих викидів на результат.

Для визначення крест-фактора калібрувальні дані (чутливість і фаза) за допомогою IFFT (зворотного швидкого перетворення Фур'є) переводилися в часову область. Для цього фаза спочатку перераховувалася в радіани і разом із чутливістю комбінувалася в комплексний частотний спектр. Цей спектр потім переводився за допомогою функції IFFT (реалізованої в NumPy) у реальний часовий сигнал. Крест-фактор визначався як відношення пікового значення цього сигналу до його середньоквадратичного значення (RMS).

Розрахунок енергії IFFT здійснювався на основі часового сигналу, визначеного в межах розрахунку крест-фактора. Енергія визначалася шляхом піднесення до квадрата всіх значень амплітуди з наступним підсумовуванням.

Нахили розраховувалися через лінійну регресію, реалізовану функцією SciPy `linregress`. Для нахилу чутливості оцінювався зв'язок між логарифмованою частотою та чутливістю. Нахил фази визначався як лінійна зміна фази залежно від частоти. В обох випадках підйом регресійної прямої слугував мірою відповідного нахилу.

Всі інші ознаки розраховувалися за допомогою функцій NumPy. Сюди входили розмах (`spannweite`), середнє значення та медіана.

За винятком аналізу резонансної частоти, розглядався виключно нормальний частотний хід. Це пов'язано з тим, що у «свіп-вимірюваннях» відсутня фазова інформація, необхідна для IFFT, а шум сигналу значно вищий порівняно зі звичайним вимірюванням.

4.2.3 Візуалізація та оцінка

Для візуалізації результатів використовувалася бібліотека Matplotlib. Відповідні показники відображалися на окремих діаграмах для відстеження їхньої часової динаміки. Частотний хід спочатку перераховувався у відносне відхилення від відповідної референтної точки, а потім також візуалізувався. Додатково додавалися графічні елементи, такі як інтегровані невизначеності вимірювань та нанесені межі допусків, для покращення інтерпретації.

Спочатку заплановане стохастичне моделювання не вдалося реалізувати, оскільки обсягу доступних даних було недостатньо. Після попередньої обробки залишилося лише чотири сенсори з повною безпомилковою історією та принаймні одним неконформним вимірюванням. Така мала вибірка не дозволяє провести статистично валідне моделювання або прогноз конформності, тому від детального статистичного аналізу відмовилися. Натомість подальша оцінка зосереджена на якісному розгляді обраних показників та їхньому зв'язку з конформністю.

4.2.4 Результати

Хоча обсягу доступних наборів даних було недостатньо для валідного статистичного моделювання, певні якісні спостереження все ж можливі. Зокрема, можна намітити зв'язки між розрахованими показниками та змінами конформності.

4.3 Нейронна мережа

Наступний розділ детально описує кілька підрозділів реалізації нейронної мережі. Він розділений на сфери відбору та попередньої обробки даних, архітектури моделі, стратегії навчання, а також отриманих результатів.

Кожен із цих підрозділів розглядає центральний аспект технічної реалізації та має на меті забезпечити відтворюваність розробленої методики.

4.3.1 Відбір та попередня обробка даних

Надалі дані були спеціально підготовлені, стандартизовані та валідовані відповідно до вимог моделі, щоб створити максимально надійну та інформативну базу для навчання. Спочатку були видалені застарілі вимірювання. Після цього «свіпи» (sweeps) були згруповані з відповідними основними вимірюваннями. На наступному етапі було проведено інтерполяцію звичайних вимірювань на заздалегідь визначений частотний діапазон. Вимірювання, які занадто сильно відхилялися від середнього значення, були видалені. Потім розпочалася підготовка даних для трьох «голів» (heads) моделі.

Для першої голови було взято значення чутливості та фази вимірювання, які були представлені разом як комплексне число. Додатково враховувалася відповідна невизначеність вимірювання, яка була об'єднана з комплекснозначним тензором як новий вимір.

У випадку другої голови використовувалися умови навколишнього середовища. Вони були переведені у двовимірний тензор, що мав виміри «час» та «ознака».

Третя голова потребувала гармонізації частот, оскільки відмінності у частотах можуть призвести до неоднозначності в інтерпретації моделі. Для цього спочатку було визначено спільний частотний діапазон на основі всіх наявних даних. Після цього в межах цього діапазону було створено логарифмічну криву, на яку інтерполювалися дані вимірювань. У цій голові невизначеність вимірювання також була включена як додатковий вимір.

Щойно всі дані були підготовлені, було проведено нормалізацію. Для цього застосовувалася симетрична МАХ-нормалізація, при якій значення відображалися в діапазоні $[-1,1]$. Такий вид масштабування був вигідним,

оскільки вихід моделі також мав представляти відносне відхилення від референтного значення, а отже, був центрований на нулі.

Зрештою, цільові значення також були приведені до форми, що містила відносне відхилення кожної точки від заздалегідь визначеної референтної точки.

Оскільки для кожного сенсора була наявна різна кількість історичних вимірювань, нейронну мережу потрібно було спроектувати таким чином, щоб горизонт введення був змінним. Для ефективного використання навчальних даних у цьому випадку вони були попередньо оброблені шляхом формування $n - 1$ складених вхідних наборів із послідовності n калібрувань:

$$\sum_{i=1}^n [(x_0, \dots, x_{i-1}), x_i], \quad (4.1)$$

Якщо вхідний блок не містив необхідної кількості минулих даних калібрування, він заповнювався нулями (padding). Це гарантувало сталу форму вхідного тензора.

4.3.2 Структура мережі

В основі архітектури моделі лежали три спеціально розроблені модулі PyTorch, кожен з яких називався «головою» і паралельно застосовував різні етапи обробки до вхідних даних. Кожна з цих голів використовувала власний структурний та функціональний підхід і стосувалася певної частини вхідних даних.

Голова 1 була центральним елементом нейронної мережі. Як уже пояснювалося раніше, вона базувалася на модифікованій реалізації архітектури FreTS. Вхідні дані пропускалися через повнозв'язні шари як вздовж часового, так і вздовж частотного вимірів. Реальна та уявна частини комплекснозначних вхідних даних розділялися, проте перераховувалися між собою і знову об'єднувалися. Комбіновані дані додавалися до початкових

вхідних даних, утворюючи форму залишкового зв'язку (Residual Connection). Після цього відбувалася подальша обробка комплекснозначним MLP, що складався з двох повнозв'язних шарів та нелінійної комплексної функції активації Tanh. Результатом роботи цієї голови був чотиривимірний тензор із вимірами Партія, Час, Частота та Канал, де вимір каналу містив фактичне значення та поширену невизначеність вимірювання.

Голова 2 мала простішу структуру. Це був класичний MLP, що складався з двох повнозв'язних шарів та функції активації. Для зменшення складності вхідних даних та ефективнішого використання часової структури перед застосуванням MLP проводилася редукція розмірності, під час якої часові кроки поєднувалися між собою. Результат цієї голови подавався у вимірах Партія, Час та Частота.

Голова 3 базувалася на згортковій архітектурі (CNN) та використовувала підхід на основі ознак для екстракції локальних структур у вхідних даних. Перший згортковий шар використовував дещо ширше ядро (Kernel), щоб охопити ширший контекст у часово-частотному представленні та зменшити вплив шуму. Загалом ця голова складалася з трьох блоків, кожен з яких містив два послідовні згорткові шари, два шари пакетної нормалізації та дві функції активації ReLU. Така структура дозволяла поступово формувати глибше представлення вхідних патернів. Після перших двох блоків застосовувався локальний Max-Pooling (вибір максимального значення з ділянок 2×2), що призводило до зменшення просторової розмірності та водночас до концентрації релевантних ознак. Для третього блоку застосовувався глобальний Average-Pooling. Метою цього кроку було отримання одного репрезентативного значення для кожного каналу, що описувало високовідцентровану ознаку початкових даних. Цей компактний опис ознак далі оброблявся іншим MLP (два повнозв'язні шари та дві функції ReLU). Фінальний вихід цього модуля був у вимірах Партія, Час та Частота, представляючи диференційовану, локально сфокусовану перспективу на вхідний сигнал.

Після індивідуальної обробки трьома модульними головами відбувалося об'єднання результатів. При цьому результати з'єднувалися вздовж виміру каналів. Крім того, всі виміри, що залишилися, об'єднувалися в один спільний для отримання уніфікованого представлення. Воно далі оброблялося фінальним MLP, що складався з трьох повнозв'язних шарів та двох функцій активації Tanh. Метою цього модуля було створення прогнозу для кожного окремого часово-частотного кортежу. Кінцевий результат знову був у вимірах Партія, Час, Частота та Канал, де виміри каналу та часу містили лише по одному елементу.

За винятком FreTS, ініціалізація параметрів виконувалася PyTorch за замовчуванням. У випадку ручної ініціалізації випадкове число вибиралося з нормального розподілу із середнім значенням 0 та дисперсією 1. У випадку FreTS це значення додатково масштабувалося константою для зовнішнього впливу на ініціалізацію ваг та їх оптимізації.

Фінальна модель містила 1 477 927 параметрів, які розподілені таким чином:

- FreTS-MLPs: 8 976;
- FreTS-вихідний MLP: 1 353 220;
- Env-Embedding-MLP: 24 865;
- 2DCNN-згортки: 70 416;
- 2DCNN-вихідний MLP: 3 169;
- фінальний MLP: 17 281.

4.3.3 Деталі навчання

Кожна епоха, що складалася з повного проходу всіх тренувальних та валідаційних даних, відбувалася за визначеною схемою. Спочатку тренувальні дані пропускалися через модель пакетами. На основі передбачення моделі розраховувалася функція втрат, яка квантифікує відхилення від очікуваного

референтного значення. Метою навчання було мінімізувати саме цей показник втрат.

Після кожного циклу навчання проводилася фаза оцінювання на валідаційних даних. Процедура була ідентичною до навчання, за винятком того, що оновлення ваг не проводилося. Тут також розраховувався Loss на основі даних, які моделі не були відомі під час навчання. Ця фаза слугувала для перевірки здатності моделі до узагальнення. Якщо спостерігалось, що втрати на тренуванні постійно падають, а на валідації — зростають, це вказувало на перенавчання. У такому випадку навчання слід припинити. Реалізація цієї частини схематично зображена на рисунку 4.1.

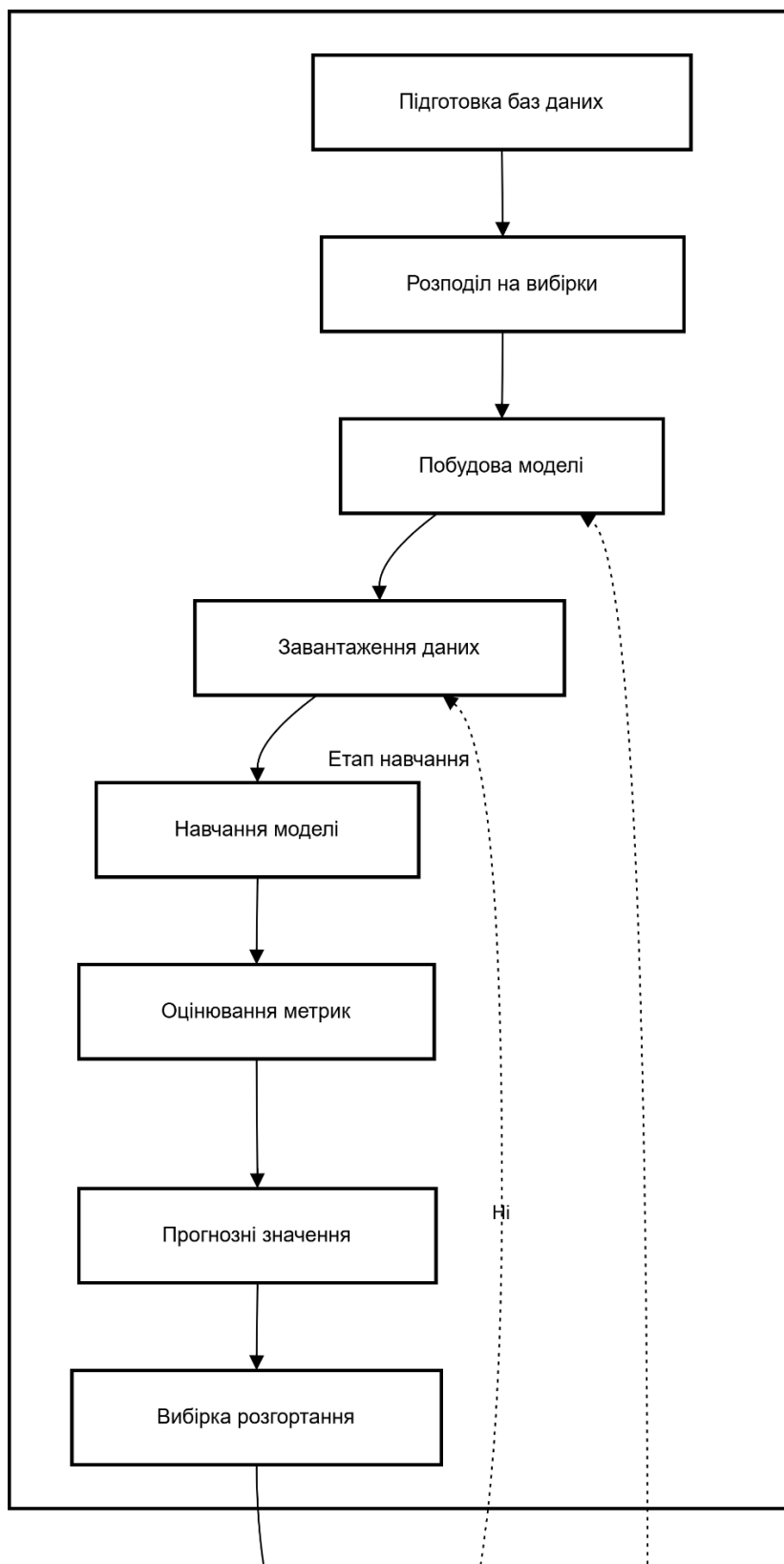


Рисунок 4.1 – Процес навчання нейронної мережі

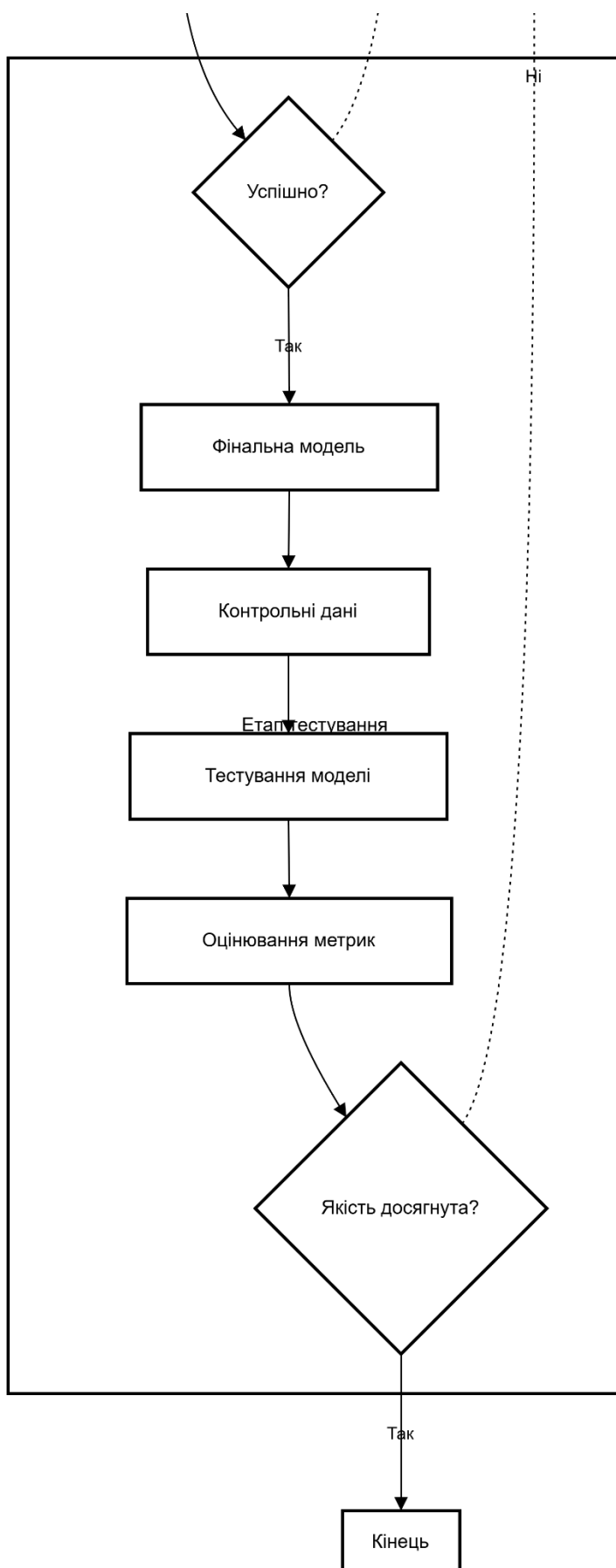


Рисунок 4.1, аркуш 2.

Обидві частини втрат зважувалися для визначення їхньої важливості. Гіперпараметри були встановлені емпірично на основі попередніх тестів:

- швидкість навчання: 0.001;
- розмір пакету: 8;
- розподіл навчання/валідація: 80% / 20%;
- кількість частот: 33;
- максимальна кількість минулих калібрувань: 30;
- епохи: 300;
- вага регресійних втрат: 1.0;
- вага втрат конформності: 0.1.

Для оптимізації ваг використовувався оптимізатор Adam з параметрами $\beta_1 = 0,9$ та $\beta_2 = 0,999$.

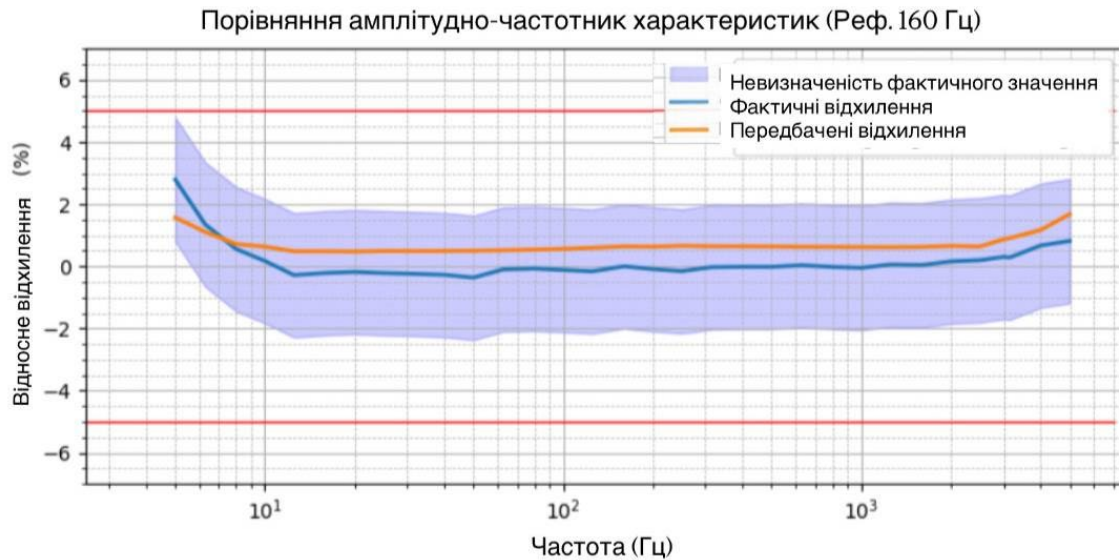
4.3.4 Результати

Модель навчалася протягом 300 епох. Графік зміни втрат моделі представлений на рисунку 4.2.

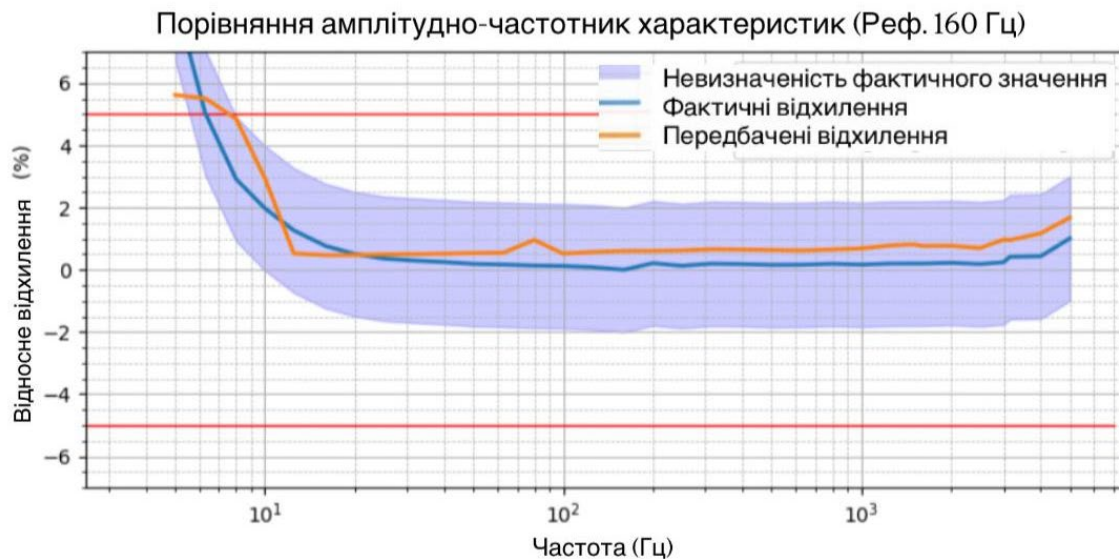


Рисунок 4.2 – Динаміка втрат моделі

На рисунках 4.3 та 4.4 показані приклади результатів роботи нейронної мережі: прогнозована та фактична частотна характеристика сенсора у вигляді відносного відхилення до референтної точки 160 Гц. Допустимі межі допусків позначені червоним кольором. Крім того, для кожного випадку вказана середньоквадратична помилка (**RMS**) у відсотках.



а)

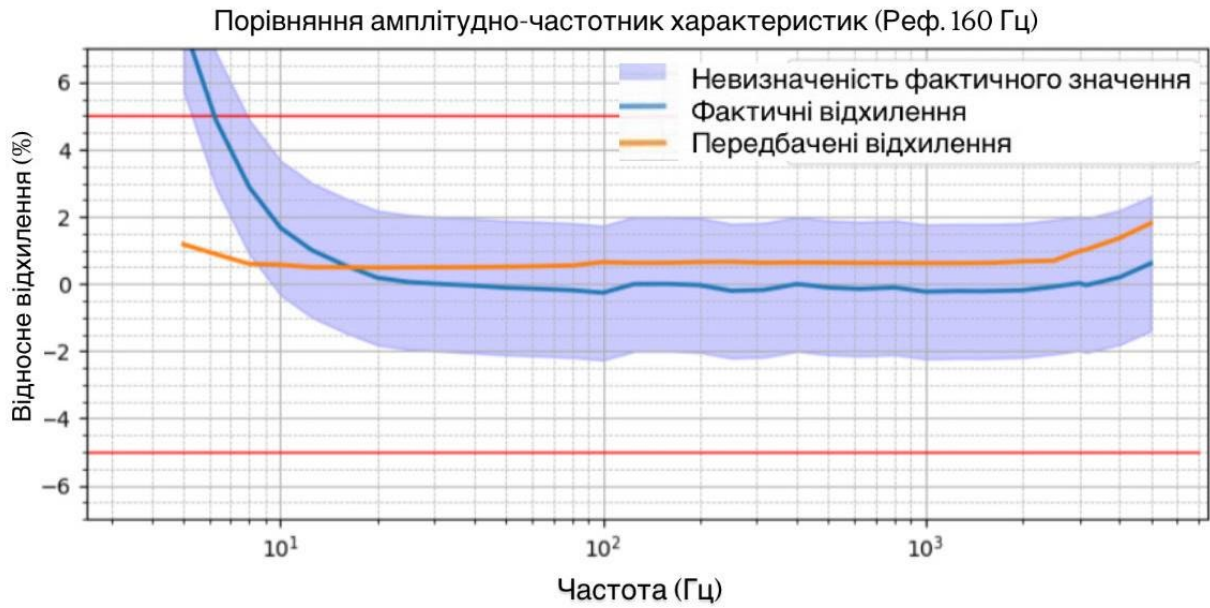


б)

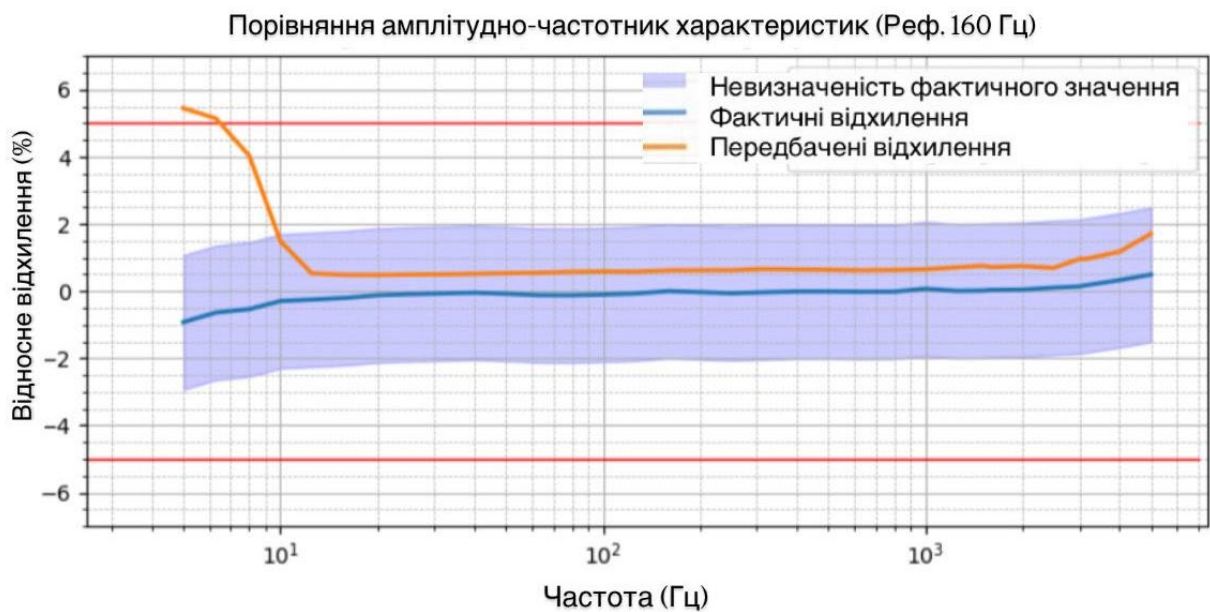
а) правильно конформний (0.60%)

б) правильно неконформний (0.83%)

Рисунок 4.3 – Змодельоване відхилення для двох правильних прогнозів



а)



б)

а) хибно конформний (1.07%)

б) хибно неконформний (1.86%)

Рисунок 4.4 – Змодельоване відхилення для двох невірних прогнозів

Результати свідчать, що нейронна мережа працює надійно для конформних частотних характеристик, проте втрачає точність при

неконформних вхідних даних. Приклад результату роботи нейронної мережі наведений на рисунку 4.5.

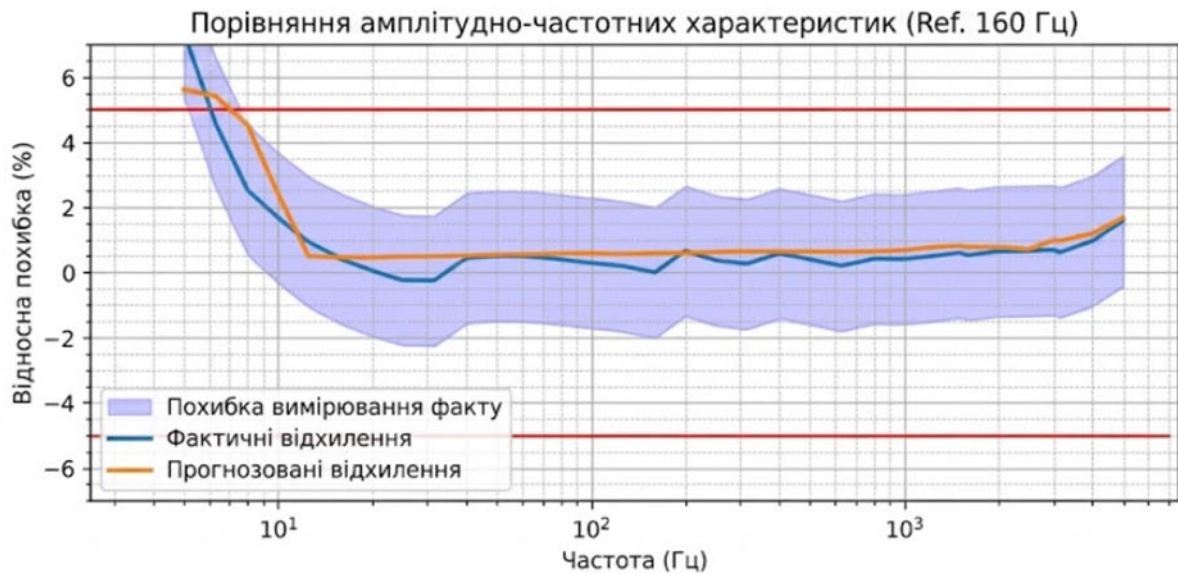


Рисунок 4.5 – Приклад результату роботи нейронної мережі: порівняння фактичної та прогнозованої відносної девіації чутливості для конформного сенсора.

4.4 Висновки за розділом 4

Динаміка втрат нейронної мережі, представлена на рисунку 4.2, на перший погляд створює враження стагнації процесу навчання. Очевидно, що вже після кількох епох мережа засвоїла базові структури базового розподілу. Подальші зміни є незначними, що вказує на досягнення моделлю локального мінімуму, з якого вона не може вийти самостійно. Цікавим є те, що неконформні частотні характеристики почали коректно розпізнаватися лише на пізніх етапах. Це свідчить про те, що тонке налаштування ваг для тих частотних точок, які є критичними для оцінки відповідності, відбувається протягом багатьох епох і триває значно повільніше, ніж вивчення загальної структури даних.

Однією з можливих причин низької динаміки втрат є дисбаланс даних: лише 4,01% навчальних прикладів є неконформними. Велика кількість конформних прикладів може призводити до того, що коригування ваг у напрямку неконформних прогнозів пригнічується домінуючим сигналом відповідності. Крім того, на продуктивність моделі можуть негативно впливати помилкові або непередбачувані вимірювання, спричинені аномаліями під час підготовки даних або випадковими внутрішніми пошкодженнями сенсорів. Такі дані можуть містити суперечливі патерни, що заважають генералізації моделі.

Попри зазначені проблеми, навчена модель загалом демонструє адекватні результати. Вона здатна прогнозувати як конформні, так і неконформні частотні характеристики. Хоча успішність прогнозів для неконформних випадків є нижчою, вона все ж присутня. На рисунках 4.3 та 4.4 наведено приклади чотирьох можливих випадків прогнозування: коректні та помилкові прогнози для обох станів сенсора. У випадках коректних прогнозів лінія передбачення майже повністю лежить у межах діапазону допуску, визначеного невизначеністю вимірювання. Від нейронної мережі не очікується точного відтворення вимірюваної кривої, оскільки вона може варіюватися під впливом зовнішніх факторів; проте модель змогла реалістично відобразити фактичний перебіг характеристики. Натомість помилкові прогнози демонструють чіткі відхилення: для конформного випадку було спрогнозовано занадто гладку криву, а для неконформного — приписано зростання в області низьких частот, якого немає в реальних даних. Це вказує на те, що модель або переоцінює іррелевантні структури, або недостатньо враховує важливі сигнали. Загальна точність прогнозів відповідності становить 95,73% на навчальних та 95,93% на валідаційних даних. Однак висока точність зумовлена переважно коректним розпізнаванням конформних випадків (понад 99%). Продуктивність щодо неконформних випадків значно слабша: під час навчання коректно розпізнано лише 6 із 42

випадків (14,29%), а під час валідації — 3 із 12 (25%). Це підтверджує гіпотезу про домінуючий вплив дисбалансу навчальної вибірки.

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес створення інтелектуальної системи для прогнозування та оцінювання якості сенсорів. Перший метод базується на традиційному статистичному підході, де на основі доменно-специфічних ознак ідентифікуються та оцінюються вручну інтерпретовані взаємозв'язки. Другий метод являє собою керовану даними нейронну мережу, яка автоматизовано вивчає латентні закономірності в історичних даних калібрування та прогнозує на їх основі частотну характеристику сенсора для наступного калібрування.

Обидва підходи відрізняються не лише технічною реалізацією, а й вимогами, сферами застосування та практичними наслідками. Традиційний підхід виграє передусім завдяки своїй прозорості. Кожне рішення можна обґрунтувати явними показниками, що є особливо важливим для критичних з точки зору безпеки застосунків та регуляторних вимог. Водночас цей метод потребує високої частки ручного втручання і важко піддається автоматизації. Крім того, якість прогнозу сильно залежить від експертних знань при виборі та інтерпретації ознак.

Нейронна мережа, навпаки, повністю відмовляється від ручного визначення ознак і добре масштабується зі зростанням обсягу даних. Автоматизована обробка дозволяє отримувати швидкі прогнози при мінімальному втручанні користувача. Однак страждає пояснюваність. Модель приймає рішення на основі внутрішніх вагових коефіцієнтів, які майже неможливо прозоро відстежити. З нормативної точки зору це означає, що так звані помилкові висновки важче виявити і ще складніше обґрунтувати. Особливо у виробничих середовищах, де сенсор, помилково визнаний невідповідним, може спричинити значні додаткові витрати, а пристрій, помилково класифікований як відповідний, може мати критичні наслідки для безпеки, це становить суттєвий ризик.

Традиційний підхід дозволяє в таких випадках проводити ручну перевірку на достовірність, тоді як нейронній мережі для цього знадобилися б додаткові структури пояснюваності, наприклад.

Виконано експериментальне дослідження розробленої інтелектуальної системи для прогнозування та оцінювання якості сенсорів. Результати тестування показали, що розроблена інтелектуальна система дозволяє забезпечити автоматизацію процесів, пов'язаних з підтримкою процесу прогнозування та оцінювання якості сенсорів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Кондратенко Ю. П., Козлов О. В. Застосування нейронних мереж для прогнозування стану сенсорних систем у промисловій автоматизації. *Автоматика. Автоматизація. Електротехнічні комплекси та системи*. 2023. Вип. 1. С. 12–21.
2. Субботін С. О., Олійник А. О. Методи інтелектуального аналізу даних : підручник. Запоріжжя : ЗНТУ, 2017. 348 с.
3. Субботін С. О. Побудова інтелектуальних систем обробки інформації : навч. посіб. Запоріжжя : НУ «Запорізька політехніка», 2020. 204 с.
4. Bassey J., Qian L., Li X. A Survey of Complex-Valued Neural Networks. *arXiv:2101.12249 [stat]*. 2021. URL: <https://arxiv.org/abs/2101.12249>. (дата звернення: 07.04.2026).
5. DIN ISO 16063-21:2016-08. Verfahren zur Kalibrierung von Schwingungs- und Stoßaufnehmern. Teil 21: Schwingungskalibrierung durch Vergleich mit einem Referenzaufnehmer. Berlin : DIN Media, 2016. 54 s.
6. Du W. PyPOTS: a Python toolbox for data mining on Partially-Observed Time Series. *GitHub repository*. 2024. URL: <https://github.com/WenjieDu/PyPOTS>. (дата звернення: 07.04.2026).
7. Yi K. et al. Frequency-domain MLPs are More Effective Learners in Time Series Forecasting. *Proceedings of the 38th AAAI Conference on Artificial Intelligence*. 2024. Vol. 38, no. 15. P. 16503–16511.
8. Yi K. et al. FilterNet: Harnessing Frequency Filters for Time Series Forecasting. *The 13th International Conference on Learning Representations (ICLR)*. Singapore: ICLR, 2025. 18 p.
9. Gutbier M. Whitepaper Konformitätsbewertung SPEKTRA Schwingungstechnik und Akustik GmbH. Dresden. 2020. 8 s.
10. Kolar D., Lisjak D., Pająk M., Gudlin M. Intelligent Fault Diagnosis of Rotary Machinery by Convolutional Neural Network with Automatic Hyper-

Parameters Tuning Using Bayesian Optimization. *Sensors*. 2021. Vol. 21, no. 7. Art. 2411. P. 22.

11. ISO/IEC 25002:2024. Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Quality model overview and usage. Geneva : International Organization for Standardization, 2024. 42 p.

12. Shao H. et al. A novel deep convolutional neural network with adaptive learning rate for fault diagnosis of rotary machinery. *IEEE Transactions on Industrial Informatics*. 2018. Vol. 14, no. 6. P. 2732–2743.

13. ISO/IEC 25059:2023. Software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Quality model for AI systems. Geneva : International Organization for Standardization, 2023. 38 p.

14. Kalibrierdienstleistungen. SPEKTRA Schwingungstechnik und Akustik GmbH Dresden. 2025. URL: <https://www.spektra-dresden.de> (дата звернення: 07.05.2026).

15. Kim M. S., Yun J. P., Park P. An Explainable Neural Network for Fault Diagnosis With a Frequency Activation Map. *IEEE Access*. 2021. Vol. 9. P. 98962–98972.

16. Selvaraju R. R. et al. Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization. *International Journal of Computer Vision*. 2020. Vol. 128, no. 2. P. 336–359.

17. Kingma D. P., Ba J. Adam: A Method for Stochastic Optimization. *The 3rd International Conference on Learning Representations (ICLR)*. San Diego: ICLR, 2015. P. 15.

18. Chen Y., Kang Y., Chen Y., Wang Z. Probabilistic Forecasting with Temporal Convolutional Neural Network. *arXiv:1906.04397 [stat]*. 2020. URL: <https://arxiv.org/abs/1906.04397>. (дата звернення: 07.04.2026).

19. Russell S. J., Norvig P. Artificial intelligence: a modern approach. 4th ed. New York: Pearson, 2020. 1166 p.

20. `scipy.stats.linregress`

URL: <https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.linregress.html>
1 (дата звернення: 07.04.2026).

21. `torch.einsum`

URL: <https://pytorch.org/docs/stable/generated/torch.einsum.html> (дата звернення: 07.04.2026).

22. `torch.nn.Conv2d`

URL: <https://pytorch.org/docs/stable/generated/torch.nn.Conv2d.html> (дата звернення: 07.04.2026).

23. `torch.optim.Adam`

URL: <https://pytorch.org/docs/stable/generated/torch.optim.Adam.html> (дата звернення: 07.04.2026).

24. Turner R. E., Sahani M. Time-Frequency Analysis as Probabilistic Inference. *IEEE Transactions on Signal Processing*. 2014. Vol. 62, no. 23. P. 6171–6183.

25. VIPFLUID – Vorausschauende Instandhaltung für Pumpensysteme auf Basis von Federated Learning und Synthese von multiplen Sensordaten
URL: <https://www.vipfluid.de> (дата звернення: 07.04.2026).

26. Yi K. et al. Deep Coupling Network for Multivariate Time Series Forecasting. *ACM Transactions on Information Systems*. 2024. Vol. 42, no. 5. P. 1–28.

27. Zhang W. et al. A New Deep Learning Model for Fault Diagnosis with Good Anti-Noise and Domain Adaptation Ability on Raw Vibration Signals. *Sensors*. 2017. Vol. 17, no. 2. Art. 425. P. 17.

ДОДАТОК А
Технічне завдання

Вступ

Це технічне завдання поширюється на розробку інтелектуальної системи, що призначена для автоматизації процесу оцінювання та прогнозування якості сенсорів на основі даних їх калібрування.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему: «Створення інтелектуальної системи прогнозування та оцінювання якості сенсорів на основі даних калібрування», затверджене наказом Національного університету «Запорізька політехніка» № 139 від 7 квітня 2026 р.

A.2 Призначення розробки

Програмний продукт призначений для аналізу амплітудно-частотних характеристик сенсорів, прогнозування їх відповідності технічним вимогам (конформності) за допомогою нейронних мереж та візуалізації результатів контролю якості.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Програмне забезпечення інтелектуальної системи забезпечує виконання таких функцій:

- зчитування та обробка вхідних даних калібрування сенсорів (файли форматів CSV/JSON/Excel);
- нормалізація вхідних сигналів та підготовка даних для нейронної мережі;

- прогнозування відносної похибки (девіації) сенсора на заданому частотному діапазоні;
- автоматична класифікація сенсорів на «придатні» (конформні) та «браковані» (неконформні);
- візуалізація результатів у вигляді графіків порівняння фактичних та прогнозованих АЧХ (як на вашому фото);
- розрахунок метрик точності прогнозування (MSE , MAE , R^2).

A.3.2 Вимоги до інтерфейсу програми

Інтерфейс повинен забезпечувати візуальне відображення графіків з накладанням меж допусків. Повинна бути реалізована можливість завантаження нових вибірок даних через графічне вікно або консольний інтерфейс.

A.3.3 Вимоги до надійності

Система повинна забезпечувати стабільну обробку великих масивів даних калібрування та коректну обробку помилок при читанні пошкоджених файлів.

A.3.4 Умови експлуатації

Для навчання нейронної мережі та виконання прогнозів необхідний ПК з підтримкою обчислень на CPU або GPU (з підтримкою CUDA для прискорення навчання).

A.3.5 Вимоги до складу та параметрів технічних засобів

Мінімальні вимоги: процесор із частотою від 2.5 ГГц, ОЗП від 8 Гб (для роботи з великими датасетами), ОС Windows 10/11 або Linux.

Програмне середовище: встановлений інтерпретатор Python 3.9+, бібліотеки TensorFlow/Keras, Pandas, NumPy, Matplotlib.

A.4 Вхідні та вихідні дані

Вхідні дані: масиви частотних відгуків сенсорів, значення амплітуд, дані попередніх калібрувань.

Вихідні дані: графіки АЧХ, бінарна класифікація придатності (OK/Fail), значення прогнозованих відхилень у відсотках.

ДОДАТОК Б
Фрагмент тексту програми

Б.1 Базові імпорти / утиліти

```
import torch
import torch.nn as nn
import torch.nn.functional as F

class ComplexLinear(nn.Module):
    def __init__(self, in_features, out_features):
        super().__init__()
        self.weight_real = nn.Parameter(torch.randn(out_features,
in_features))
        self.weight_imag = nn.Parameter(torch.randn(out_features,
in_features))
        self.bias_real = nn.Parameter(torch.randn(out_features))
        self.bias_imag = nn.Parameter(torch.randn(out_features))

    def forward(self, input):
        real = F.linear(input.real, self.weight_real) -
F.linear(input.imag, self.weight_imag)
        imag = F.linear(input.real, self.weight_imag) +
F.linear(input.imag, self.weight_real)
        return torch.complex(real + self.bias_real, imag +
self.bias_imag)
```

Б.2 Допоміжні функції

```
class ComplexTanh(nn.Module):
    def forward(self, input):
        return torch.complex(torch.tanh(input.real),
torch.tanh(input.imag))
```

Б.3 Preprocessing FreTS

```
class FreTS(nn.Module):
    def __init__(self, n_steps, embed_size, n_pred_steps, hidden_size,
out_dim=None, stat_dim=2):
        super().__init__()

        self.n_steps = n_steps
        self.n_pred_steps = n_pred_steps
        self.embed_size = embed_size
        self.hidden_size = hidden_size
        self.stat_dim = stat_dim
        self.out_dim = out_dim if out_dim is not None else embed_size

        self.sparsity_threshold = 0.01
        self.scale = 0.02

        self.r1 = nn.Parameter(self.scale * torch.randn(embed_size,
embed_size, stat_dim))
        self.i1 = nn.Parameter(self.scale * torch.randn(embed_size,
embed_size, stat_dim))
```

```

        self.rb1 = nn.Parameter(self.scale * torch.randn(embed_size,
stat_dim))
        self.ib1 = nn.Parameter(self.scale * torch.randn(embed_size,
stat_dim))

        self.r2 = nn.Parameter(self.scale * torch.randn(embed_size,
embed_size, stat_dim))
        self.i2 = nn.Parameter(self.scale * torch.randn(embed_size,
embed_size, stat_dim))
        self.rb2 = nn.Parameter(self.scale * torch.randn(embed_size,
stat_dim))
        self.ib2 = nn.Parameter(self.scale * torch.randn(embed_size,
stat_dim))

        self.ffn = nn.Sequential(
            ComplexLinear(n_steps * embed_size * stat_dim, hidden_size),
            ComplexTanh(),
            ComplexLinear(hidden_size, n_pred_steps * stat_dim),
        )

def FreMLP(self, B, nd, dimension, stat_dim, x, r, i, rb, ib):
    real = F.relu(
        torch.einsum("bijdk,ddk->bijdk", x.real, r)
        - torch.einsum("bijdk,ddk->bijdk", x.imag, i)
        + rb
    )

    imag = F.relu(
        torch.einsum("bijdk,ddk->bijdk", x.imag, r)
        + torch.einsum("bijdk,ddk->bijdk", x.real, i)
        + ib
    )

    y = torch.stack([real, imag], dim=-1)
    y = F.softshrink(y, lambd=self.sparsity_threshold)
    return torch.view_as_complex(y)

def MLP_temporal(self, x, B, N, T, S):
    return self.FreMLP(B, N, T, S, x, self.r2, self.i2, self.rb2,
self.ib2)

def MLP_channel(self, x, B, N, T, S):
    x = x.permute(0, 2, 1, 3, 4)
    y = self.FreMLP(B, T, N, S, x, self.r1, self.i1, self.rb1,
self.ib1)
    return y.permute(0, 2, 1, 3, 4)

def forward(self, x):
    B, T, N, S = x.shape

    x = x.permute(0, 2, 1, 3)
    x = x.unsqueeze(3)

    bias = x

    x = self.MLP_channel(x, B, N, T, S)
    x = self.MLP_temporal(x, B, N, T, S)

    x = x + bias

    x = x.reshape(B, N, -1)
    x = self.ffn(x)

    x = x.reshape(B, N, self.n_pred_steps, self.stat_dim)

```

```

x = x.permute(0, 2, 3, 1)
x = x.permute(0, 1, 3, 2)

return x.abs()

```

Б.4 Окремі частини моделі

```

class EnvEmbedding(nn.Module):
    def __init__(self, time_steps, n_dim, out_dim):
        super().__init__()
        self.input_dim = time_steps * n_dim

        self.mlp = nn.Sequential(
            nn.Flatten(),
            nn.Linear(self.input_dim, 128),
            nn.ReLU(),
            nn.Linear(128, out_dim),
        )

    def forward(self, x):
        out = self.mlp(x)
        return out.unsqueeze(1).unsqueeze(-1)

```

Б.5 CNN, ще допоміжні класи або loss-и

```

class WDCNNInspiredCNN(nn.Module):
    def __init__(self, output_dim, base_channels=16):
        super().__init__()

        self.conv1 = nn.Sequential(
            nn.Conv2d(2, base_channels, kernel_size=(7, 3), stride=(2,
1), padding=(3, 1)),
            nn.BatchNorm2d(base_channels),
            nn.ReLU(inplace=True),
        )

        self.conv2 = self.conv_block(base_channels, base_channels * 2)
        self.conv3 = self.conv_block(base_channels * 2, base_channels *
4)

        self.pool = nn.MaxPool2d((2, 2), stride=2)
        self.global_avg_pool = nn.AdaptiveAvgPool2d((1, 1))

        self.ffn = nn.Sequential(
            nn.Linear(base_channels * 4, base_channels * 2),
            nn.ReLU(inplace=True),
            nn.Linear(base_channels * 2, output_dim),
            nn.ReLU(inplace=True),
        )

    def conv_block(self, in_ch, out_ch):
        return nn.Sequential(
            nn.Conv2d(in_ch, out_ch, 3, padding=1),
            nn.BatchNorm2d(out_ch),
            nn.ReLU(inplace=True),
            nn.Conv2d(out_ch, out_ch, 3, padding=1),
            nn.BatchNorm2d(out_ch),
            nn.ReLU(inplace=True),

```

```

    )

def forward(self, x):
    if x.dim() == 4:
        x = x.permute(0, 3, 1, 2)

    x = self.conv1(x)
    x = self.pool(x)
    x = self.conv2(x)
    x = self.pool(x)
    x = self.conv3(x)

    x = self.global_avg_pool(x)
    x = torch.flatten(x, 1)

    x = self.ffn(x)
    return x.unsqueeze(1).unsqueeze(-1)

```

Б.6 CalibrationForecasting (об'єднання моделі)

```

import torch
import torch.nn as nn

class CalibrationForecasting(nn.Module):
    def __init__(self, frets_model, cnn_model, env_model,
hidden_dim=128):
        super().__init__()

        self.frets = frets_model          # спектр (complex / FFT)
        self.cnn = cnn_model              # sweep
        self.env = env_model              # env features

        self.fc = nn.Sequential(
            nn.Linear(self.frets.out_dim + self.cnn.out_dim +
self.env.out_dim, hidden_dim),
            nn.ReLU(),
            nn.Linear(hidden_dim, 1) # regression:  $\delta(f)$  або deviation
        )

    def forward(self, freqs, sweep, env):
        f_out = self.frets(freqs)
        c_out = self.cnn(sweep)
        e_out = self.env(env)

        x = torch.cat([f_out, c_out, e_out], dim=-1)
        return self.fc(x)

```

Б.7 Conformity (логіка оцінки)

```

def conformity_score(delta, delta_min, delta_max, unc):
    """
    delta: predicted deviation (%)
    delta_min, delta_max: thresholds
    unc: uncertainty
    """

    konf95 = (delta_min + unc <= delta) & (delta <= delta_max - unc)

```

```

    konf50 = (delta_min <= delta) & (delta <= delta_max)

    y = torch.zeros_like(delta)

    y[konf50] = 0.5
    y[konf95] = 1.0

    return y

def total_conformity(y_f):
    """YS = min(y(f))"""
    return torch.min(y_f, dim=-1).values

```

Б.8 Train_loop + test_loop

```

def train_loop(model, dataloader, optimizer, criterion, device):
    model.train()
    total_loss = 0

    for batch in dataloader:
        freqs, sweep, env, target = [b.to(device) for b in batch]

        pred = model(freqs, sweep, env)

        loss = criterion(pred.squeeze(), target)

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        total_loss += loss.item()

    return total_loss / len(dataloader)

@torch.no_grad()
def test_loop(model, dataloader, criterion, device):
    model.eval()
    total_loss = 0

    for batch in dataloader:
        freqs, sweep, env, target = [b.to(device) for b in batch]

        pred = model(freqs, sweep, env)
        loss = criterion(pred.squeeze(), target)

        total_loss += loss.item()

    return total_loss / len(dataloader)

```

Б.9 Генерація датасету

```

def _generate_samples(self):
    if self.data.empty:
        print("No data to sample")
        return []

```

```

samples = []

sweep_ref_grid = self._compute_sweep_ref_grid(self.data)

for (sensor_type, serial), data in self.data.groupby(["SensorType",
"Serial"]):
    data = data.reset_index(drop=True)

    for i in range(1, len(data)):

        input_seq = [
            self._get_complex_transfer_data(data.iloc[j])
            for j in range(i)
        ]

        input_seq = input_seq + [
            torch.zeros_like(input_seq[0])
        ] * (self.max_len - len(input_seq))

        input_env = [
            self._get_env_tensor(data.iloc[j])
            for j in range(i)
        ]

        input_env = input_env + [
            torch.zeros_like(input_env[0])
        ] * (self.max_len - len(input_env))

        input_sweep = [
            self._get_sweep_tensor(data.iloc[j], sweep_ref_grid)
            for j in range(i)
        ]

        input_sweep = input_sweep + [
            torch.zeros_like(input_sweep[0])
        ] * (self.max_len - len(input_sweep))

        output = self._get_relative_deviation_data(data.iloc[i])

        samples.append((
            sensor_type,
            serial,
            normalize(torch.stack(input_seq)),
            normalize(torch.stack(input_env)),
            normalize(torch.stack(input_sweep)),
            output,
            data.iloc[i]["Conformity"],
        ))

return samples

```

ДОДАТОК В
Слайди презентації

Національний університет «Запорізька політехніка»

Кафедра програмних засобів

Створення інтелектуальної системи прогнозування та оцінювання якості сенсорів на основі даних калібрування

Виконавець:

Микола Дубіль, студент групи КНТ-222

Керівник:

Андрій Олійник, д.т.н., професор

Рисунок В.1 – Слайд 1

Об'єкт, предмет та мета дослідження

Об'єкт дослідження:

Процес створення інтелектуальної системи для прогнозування та оцінювання якості сенсорів.

Предмет дослідження:

Методи та програмні системи для прогнозування та оцінювання якості сенсорів.

Мета роботи:

Скорочення часу оцінювання метрологічної придатності сенсорів шляхом розробки інтелектуальної системи прогнозування на основі аналізу їхніх частотних характеристик та умов експлуатації.

Рисунок В.2 – Слайд 2

Завдання дослідження

Для досягнення поставленої мети у кваліфікаційній роботі бакалавра необхідно розв'язати такі задачі:

- виконати аналіз існуючих підходів та систем для прогнозування та оцінювання якості сенсорів;
- здійснити проектування архітектури інтелектуальної системи та її математичного забезпечення ;
- створити програмне забезпечення системи прогнозування якості сенсорів з використанням мови Python та фреймворку PyTorch;
- виконати експериментальне дослідження та тестування розробленого програмного забезпечення на реальних даних для оцінки точності прогнозів.

3

Рисунок В.3 – Слайд 3

Аналіз існуючих рішень

Критерій порівняння	Спеціалізоване ПЗ LabVIEW/MATLAB	Статистичні пакети(R/Scikit-learn)	Нейромережові фреймворки (PyTorch/TensorFlow)
Обробка сигналів	Висока (вбудовані фільтри)	Середня	Висока (кастомні шари)
Прогнозування дрейфу	На основі трендів	Статистичне	Глибоке (предиктивне)
Автоматизація	Потребує ручного налаштування	Потребує виділення ознак	Повна (End-to-end)
Точність (Accuracy)	Висока	Середня	Висока

4

Рисунок В.4 – Слайд 4

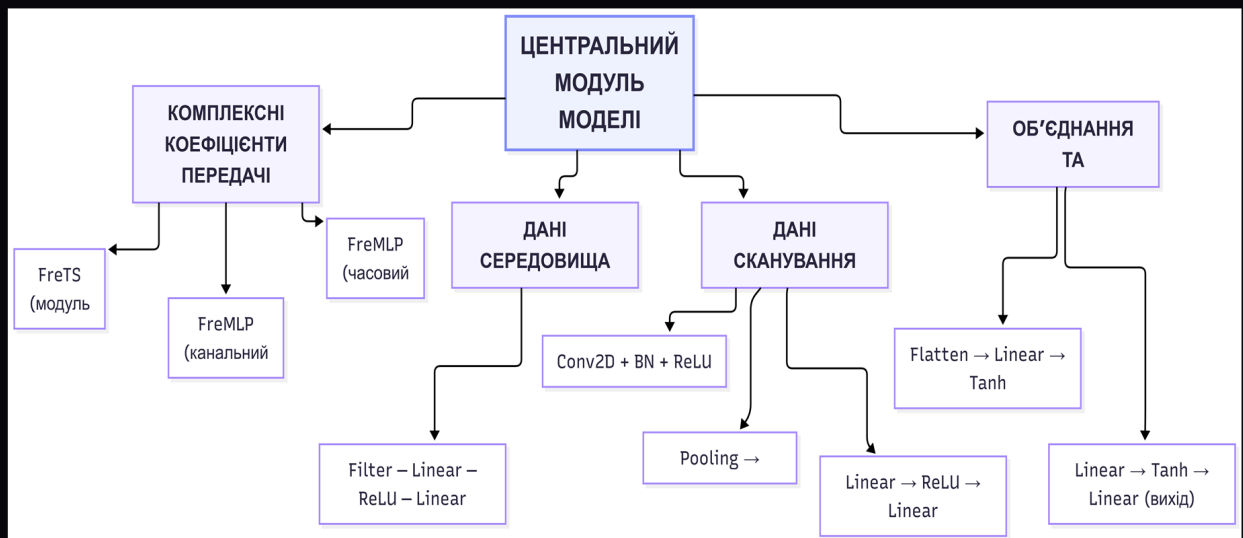
Вибір мови програмування

Критерій порівняння	Python	C++	R	Java	MATLAB
Швидкість розробки та прототипування	+	-	+-	+-	+
Бібліотеки для Deep Learning (TensorFlow, PyTorch)	+	+-	+-	-	+-
Обробка та очищення специфічних даних сенсорів	+	-	+	+-	+-
Спільнота та готова база рішень (Open Source)	+	+-	+	+	-
Інтеграція з IoT-платформами та хмарами	+	+	-	+	-
Простота розгортання моделей (Deployment)	+	+-	-	+	-
Вартість ліцензування та інструментарію	+	+	+	+	-

5

Рисунок В.5 – Слайд 5

Структура моделі



6

Рисунок В.6 – Слайд 6

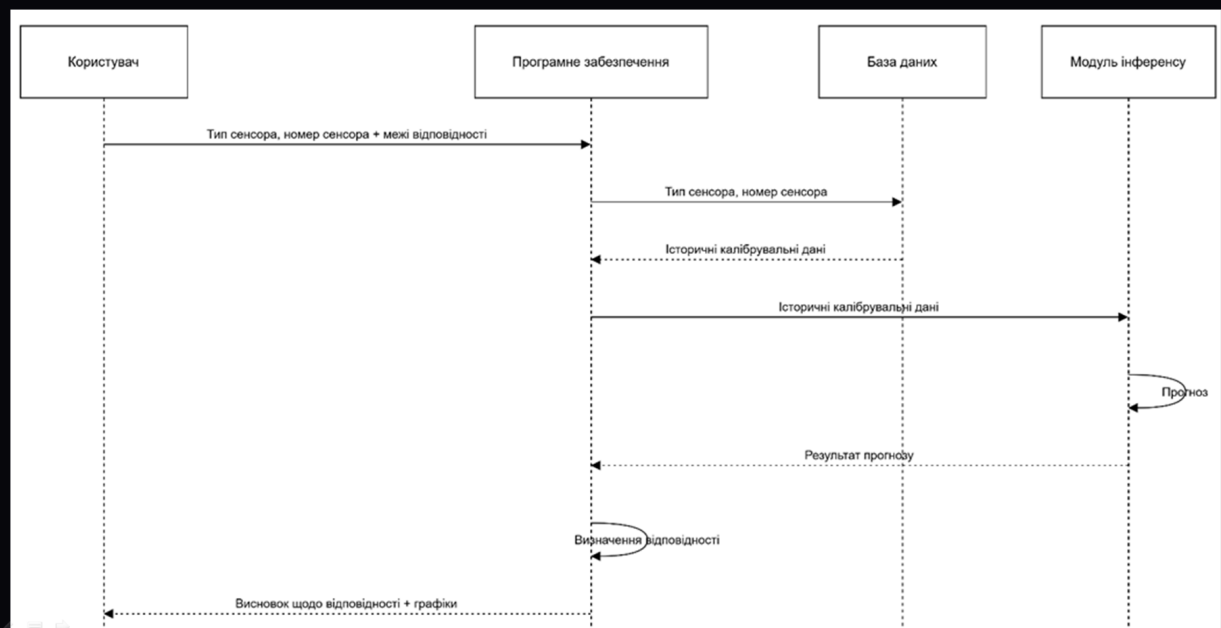
Програмні модулі інтелектуальної системи прогнозування та оцінювання сенсорів

Параметр порівняння	Модуль FreTS	Модуль WDCNN (CNN)	Модуль MLP
Вхідні дані	Комплексна АЧХ	Свіп-вимірювання	Метадані середовища
Основна функція	Аналіз дрейфу чутливості	Виявлення дефектів корпусу	Компенсація умов довкілля
Ключова особливість	Робота в частотному домені	Широке ядро згортки	Обробка числових векторів
Математичний апарат	Комплексні лінійні шари	Глибока згортка, Pooling	Повнозв'язні шари, ReLU
Результат обробки	Спектральні ознаки старіння	Геометричні аномалії графіка	Корекційний коефіцієнт

7

Рисунок В.7 – Слайд 7

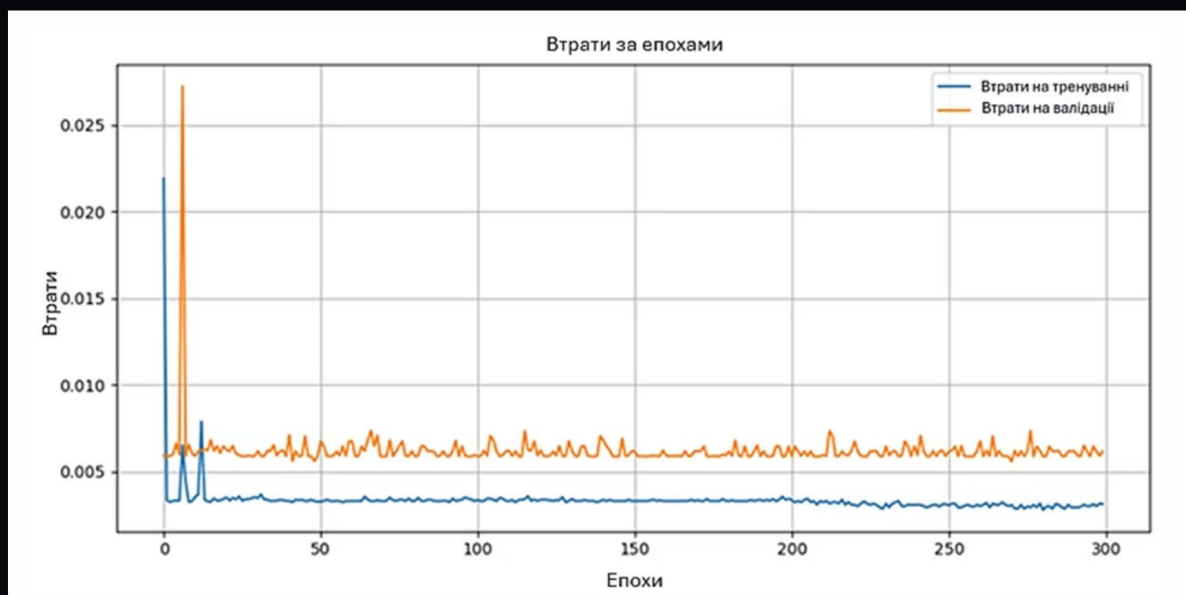
Діаграма послідовності процесів обчислень та прийняття рішень



8

Рисунок В.8 – Слайд 8

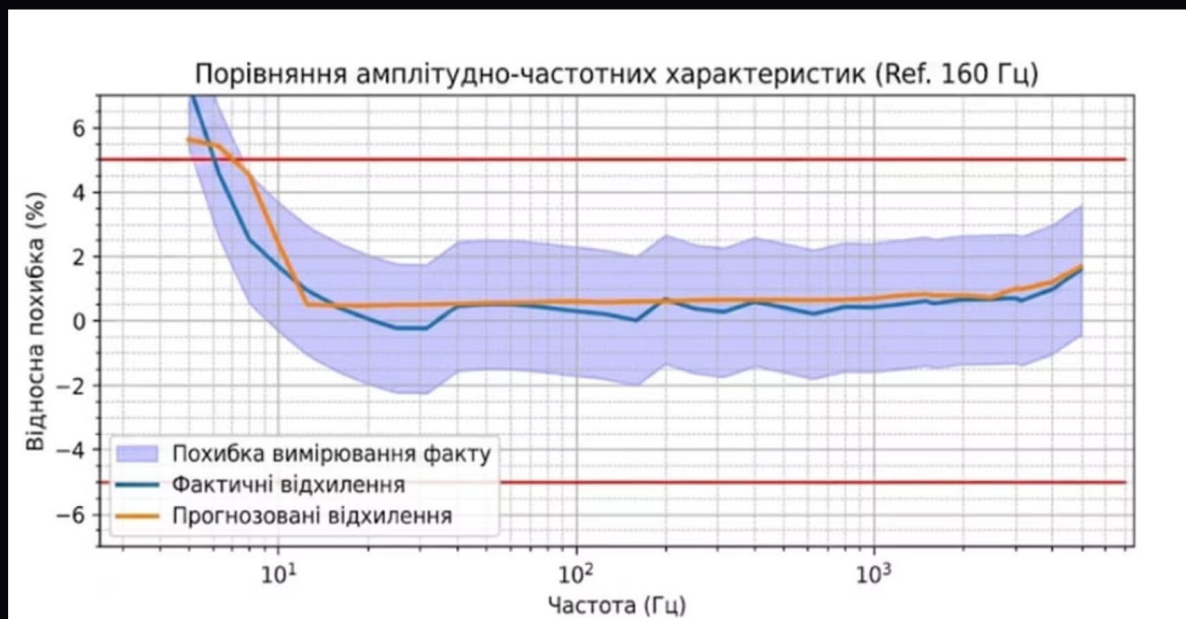
Графік втрат за епохами



9

Рисунок В.9 – Слайд 9

Результат роботи програми: АЧХ сенсора



10

Рисунок В.10 – Слайд 10

Результати експериментальних досліджень

200K+

Записів калібрувань

Реальні промислові дані для навчання та валідації

95.93%

Загальна точність

На валідаційній вибірці після 300 епох навчання

>99%

Конформні стани

Висока ефективність розпізнавання справних сенсорів

300

Епох навчання

Стабільна збіжність функції втрат без перенавчання

11

Рисунок В.11 – Слайд 11

Висновки

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес створення інтелектуальної системи для прогнозування та оцінювання якості сенсорів.

Для реалізації інтелектуальної системи прогнозування та оцінювання якості сенсорів обрано мову програмування Python.

Розроблено інтелектуальну систему прогнозування та оцінювання якості сенсорів, яка складається з модуля спектрального аналізатору, модуля згорткового екстрактору, та модуля обробки даних.

Виконано експериментальне дослідження розробленої інтелектуальної системи для прогнозування та оцінювання якості сенсорів. Результати тестування показали, що розроблена інтелектуальна система дозволяє забезпечити автоматизацію процесів, пов'язаних з підтримкою процесу прогнозування та оцінювання якості сенсорів.

12

Рисунок В.12 – Слайд 12