Київський національний університет імені Тараса Шевченка
Факультет інформаційних технологій
Кафедра програмних систем і технологій

УДК 004.75;004.042 *На правах рукопису*

ВИПУСКНА КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

Тема: “Модуль виявлення аномалій для хмарної Industrial IoT системи”

Спеціальність – 121 “Інженерія програмного забезпечення”

ПОЯСНЮВАЛЬНА ЗАПИСКА

БР.ІПЗ – 30.00.00.000

Студент
ІПЗ-43 _____ /Ернест МІДОЯН/

Науковий керівник
асист. _____ /Кирило КАДОМСЬКИЙ/

Консультант
з питань нормоконтролю
фахівець _____ /Тамара ЧАПОВСЬКА/

Допускається до захисту
Завідувач кафедри
д.т.н., проф. _____ /Олексій БИЧКОВ/

Київ – 2021

Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій

Кафедра програмних систем і технологій

Освітньо-кваліфікаційний рівень бакалавр

Спеціальність 121 “Інженерія програмного забезпечення”

ЗАТВЕРДЖЕНО

Зав. кафедри програмних систем і технологій

_____ (Олексій БИЧКОВ)

ЗАВДАННЯ

НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Мідояну Ернесту Рубеновичу

(прізвище, ім’я, по-батькові)

1. **Тема бакалаврської роботи** “ Модуль виявлення аномалій для хмарної Industrial IoT системи ”

   керівник роботи Кадомський Кирило Костянтинович

   затверджені наказом вищого навчального закладу від “11” листопада 2020 р. № 6
2. **Строк подання студентом роботи** 04.06.2021 р.
3. **Вихідні дані до проекту (роботи)** Теоретичні концепції методів виявлення аномалій та практична реалізація у вигляді модуля виявлення аномалій
4. **Зміст пояснювальної записки (перелік питань, які потрібно розробити)**
   1. Огляд систем Industrial IoT.
   2. Аналіз методів виявлення аномалій.
   3. Проведення експериментального дослідження обраної системи та оцінка ефективності алгоритму виявлення аномалій.
   4. Розробка програмної реалізації модуля виявлення аномалій.

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

1. Виявлення аномалій за допомогою автоенкодеру (рис. 1.8, стор. 22).
2. Архітектура LSTM автоенкодера (рис. 2.1, стор. 25).
3. Розподіли помилки реконструкції сигналу (рис. 2.6, стор. 31).
4. Штучно внесена у сигнал аномалія (рис. 2.8, стор. 33).
5. Діаграма прецедентів (рис. 3.1, стор. 39).
6. Діаграма розгортання (рис. 3.2, стор. 40).
7. Діаграма прецедентів (рис. 3.3, стор. 41).

6. Консультанти з роботи із зазначенням розділів роботи, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
РОЗДІЛ 1	Кирило КАДОМСЬКИЙ		
РОЗДІЛ 2	Кирило КАДОМСЬКИЙ		
РОЗДІЛ 3	Кирило КАДОМСЬКИЙ		
РОЗДІЛ 4	Кирило КАДОМСЬКИЙ		

7. Дата видачі завдання 20.11.2020

Керівник _____ Кирило КАДОМСЬКИЙ

Завдання прийняв до виконання _____ Ернест МІДОЯН

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назви етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Уточнення постановки задачі	25.11.2020-03.12.2020	виконано
2	Аналіз літератури	12.12.2020-01.01.2021	виконано

3	Аналіз існуючих методів виявлення аномалій	08.01.2021-19.01.2021	виконано
4	Пошук підходящої промислової системи	10.02.2021-24.02.2021	виконано
5	Проведення експериментального дослідження	03.03.2021-30.03.2021	виконано
6	Розробка програмного забезпечення	01.04.2021-30.04.2021	виконано
7	Тестування розробленого програмного забезпечення	01.05.2021-14.05.2021	виконано
8	Оформлення і друк пояснювальної записки	16.05.2021-26.05.2021	виконано
9	Оформлення презентації	27.05.2021-03.06.2021	виконано
10	Отримання рецензії	10.06.2021	виконано
11	Затвердження пояснювальної записки роботи завідувачем кафедри	14.06.2021	виконано
12	Захист дипломної роботи	23.06.2021	виконано

Студент – бакалавр _____ Ернест МІДОЯН

Керівник роботи _____ Кирило КАДОМСЬКИЙ

АНОТАЦІЯ

Випускна кваліфікаційна бакалаврська робота: 62 сторінки, 22 рисунки, 2 додатки, 10 джерел.

Тема: Модуль виявлення аномалій для хмарної Industrial IoT системи.

Об’єкт дослідження: динамічні процеси в системах Industrial IoT.

Мета роботи: дослідити межі застосування методів виявлення аномалій у системах Industrial IoT.

Предмет дослідження: методи виявлення аномалій в даних (спостереженнях) виробничого процесу системи Industrial IoT.

Результати дослідження:

Досліджено різні підходи до задачі виявлення аномалій у промислових системах. Запропонована та застосована власна модель глибинного навчання для вирішення задачі виявлення аномалій на прикладі обраної промислової системи. Реалізовано модуль виявлення аномалій з використанням розробленої моделі.

Висновок

В результаті роботи було досліджено різні підходи до задачі виявлення аномалій, реалізований модуль виявлення аномалій з використанням розробленої моделі.

ПРОМИСЛОВИЙ ІНТЕРНЕТ РЕЧЕЙ, ВИЯВЛЕННЯ АНОМАЛІЙ, ГЛИБИННЕ НАВЧАННЯ, АВТОЕНКОДЕР, ВЕБ-ТЕХНОЛОГІЇ.

АННОТАЦИЯ

Выпускная квалификационная бакалаврская работа: 62 страницы, 22 рисунка, 2 дополнения, 10 источников.

Тема: Модуль выявления аномалий для облачной Industrial IoT системы.

Объект исследования: динамические процессы в системах Industrial IoT.

Цель работы: исследовать пределы применимости методов выявления аномалий в системах Industrial IoT.

Предмет исследования: методы выявления аномалий в данных (наблюдениях) производственного процесса Industrial IoT.

Результаты исследования:

Исследованы различные подходы к задаче обнаружения аномалий в промышленных системах. Предложена и применена собственная модель глубокого обучения для решения задачи обнаружения аномалий на примере выбранной промышленной системы. Реализован модуль обнаружения аномалий с использованием разработанной модели.

Вывод

В результате работы были исследованы различные подходы к задаче обнаружения аномалий, реализован модуль обнаружения аномалий с применением разработанной модели.

ПРОМЫШЛЕННЫЙ ИНТЕРНЕТ ВЕЩЕЙ, ОБНАРУЖЕНИЕ АНОМАЛИЙ, ГЛУБОКОЕ ОБУЧЕНИЕ, АВТОЕНКОДЕР, ВЕБ-ТЕХНОЛОГИИ.

ANNOTATION

Final qualifying bachelor's thesis: 62 pages, 22 images, 2 addition, 10 sources.

Topic: Anomaly detection module for a cloud Industrial IoT system.

Object of research: dynamic processes in Industrial IoT systems.

Purpose: explore the limits of the applicability of anomaly detection methods in the Industrial IoT systems.

Subject of research: methods for detecting anomalies in the data (observations) of the Industrial IoT production process.

Research results:

Various approaches to the anomaly detection in industrial systems have been researched. A deep learning model is proposed and applied to detect anomalies in a selected industrial system. An anomaly detection module has been implemented using the deep learning model.

Conclusion

As a result, various approaches to the anomaly detection were researched, an anomaly detecting module was implemented based on the proposed deep learning model.

INDUSTRIAL INTERNET OF THINGS, ANOMALY DETECTION, DEEP LEARNING, AUTOENCODER, WEB TECHNOLOGIES.

ЗМІСТ

Стр.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

IOT	-	internet of things (інтернет речей)
IIOT	-	industrial internet of things (промисловий інтернет речей)
СУБД	-	система управління базами даних
БД	-	база даних
ПЗ	-	програмне забезпечення
DL	-	deep learning (глибинне навчання)
API	-	Application Programming Interface
UML	-	Unified Modeling Language
HTTP	-	HyperText Transfer Protocol
HTTPS	-	HyperText Transfer Protocol Secure
JSON	-	JavaScript Object Notation
SQL	-	Structured Query Language
REST	-	Representational State Transfer

ВСТУП

Сучасний світ живе у епоху третьої промислової революції, що почалася у середині минулого століття, характерними рисами якої є автоматизція виробничих процесів, розвиток інформаційно-комунікаційних процесів. На сьогодні новим трендом є четверта промислова революція (Industrie 4.0) [1], концепція якої була вперше сформульована в 2011 році в Німеччині. Особливістю є перехід до повністю автоматизованих промислових виробництв, управління якими здійснюється у реальному часі з урахуванням можливих змін зовнішніх чинників.

Однією з важливих технологій, що лежить в основі сучасних промислових систем є технологія інтернету речей (IoT), що слугує для обміну інформацією між пристроями, датчиками, системами тощо. Ключем для створення автоматизованих підприємств є її різновид – промисловий інтернет речей (industrial IoT).

У даній роботі розглядається одна з головних проблем, що стоїть перед автоматизованими промисловими системами, а саме виявлення аномальної поведінки, що може призвести до значних дефектів та порушень у їх роботі. Сьогодні існують різноманітні підходи до задачі виявлення аномалій, які показали різну ефективність під час досліджень. Головним недоліком існуючих методів є їх специфічність для конкретної предметної області та конкретної системи, що розглядається. Тобто актуальним залишається питання визначення меж застосування класичних методів виявлення аномалій у різних по своїй природі системах.

У ряді промислових систем [2] класичні архітектури автоенкодерів є неефективними. В роботі пропонується модифікація архітектури автоенкодера на основі довгої короткочасної пам’яті, яка забезпечує значне підвищення ефективності виявлення аномалій в одній з промислових систем, але більш широкі перспективи застосування та обмеження таких моделей в системах Industrie 4.0 залишаються невивченими.

В даному дослідженні ставиться задача бенчмаркінгу моделі глибинного навчання в системі IIoT, яка має суттєво інші динамічні характеристики і в якій існуючі методи моделювання виявилися неефективними.

Мета і задачі дослідження

Метою бакалаврської роботи є дослідити межі застосування моделей автоенкодерів у задачі виявлення аномалій на прикладі промислової системи. Створити власний програмний застосунок, а саме модуль виявлення аномалій, з використанням розробленої моделі глибинного навчання.

Задачі роботи:

- Аналіз літературних та інтернет джерел щодо предметної області.
- Огляд існуючих рішень щодо задачі виявлення аномалій.
- Реалізація моделі глибинного навчання на основі архітектури автоенкодера.
- Оцінка ефективності роботи моделі, порівняння з іншими алгоритмами.
- Проектування та реалізація модуля виявлення аномалій із застосуванням створеної моделі.

Об’єктом дослідження є динамічні процеси в системах промислового інтернету речей.

Предметом дослідження є методи виявлення аномалій в даних (спостереженнях) виробничого процесу промислового інтернету речей.

Методи дослідження – пошук та ознайомлення з інформацією, аналіз даних, дослідження існуючих методів виявлення аномалій, огляд та вивчення літератури, перегляд лекцій та відео матеріалів, дослідження існуючих архітектурних рішень для реалізації програмного застосунку.

Новизна одержаних результатів – запропоновано архітектурну модифікацію моделі глибинного навчання – автоенкодера для вирішення задачі виявлення аномалій в промислових системах, досліджено межі її застосування, проведено порівняння з іншими алгоритмами.

Практичне значення одержаних результатів полягає у тому, що розроблений програмний застосунок можливо використовувати для виявлення аномалій. Є можливість широкої кастомізації застосунку для конкретних промислових систем, включаючи заміну використовуваної моделі, встановлення порогових значень, інформування різними комунікаційними каналами.

Публікації

За результатами наукового дослідження, проведеного у бакалаврській роботі, підготовлено тези для 8-мої Східно-Европейської конференції Математичні та програмні технології Internet of Everything, а також підготовлена стаття на тему «методи управління і оцінювання в умовах невизначеності (проблеми динаміки керованих систем)» для міжнародного науково-технічного журналу “Проблеми управління та інформатики” Інститут космічних досліджень НАН України и ДКА України.

Особистий внесок:

- Запропонована архітектурна модифікація моделі автоенкодера, досліджено ефективність її застосування на прикладі промислової системи.
- Запропоновано алгоритм вставки аномалій у певний проміжок сигналу з параметрами амплітуди, різкості переходу між аномальним та нормальним сигналом.

Структура та обсяг роботи

Робота викладена на 62 сторінках друкованого тексту, який складається із вступу, чотирьох розділів, висновків, списку використаних джерел (10 найменувань). Робота містить 3 таблиці, 22 рисунки та 2 додатки.

РОЗДІЛ 1
АНАЛІЗ ПІДХОДІВ ДО ВИЯВЛЕННЯ АНОМАЛІЙ ДЛЯ СИСТЕМ IIOT

1.1 Системи IIoT

З бурхливим технологічним прогресом людства і розвитком промислового виробництва актуальним стає поняття Industrie 4.0 - складова більш глобального поняття «четвертої промислової революції». Характерною рисою є перехід до повністю автоматизованим виробництвам, на яких керівництво всіма процесами здійснюється в режимі реального часу і з урахуванням мінливих зовнішніх умов. Експерти виділяють чотири складові Industrie 4.0, серед яких провідне місце займає інтернет речей (IoT).

Промисловий інтернет речей не має стандартно визначеної універсальної архітектури роботи, якої потрібно суворо дотримуватися. Архітектура IIoT залежить від його функціональності та реалізації в різних секторах. Тим не менше, існує основний технологічний процес, на основі якого будується IIoT.

Системи IIoT зазвичай задумуються як багатошарова модульна архітектура цифрових технологій. Рівень пристрою (device layer) відноситься до фізичних компонентів: кібер-фізичних систем, датчиків або машин. Мережевий рівень (network layer) складається з фізичних мережевих шин, хмарних обчислень та протоколів зв'язку, які агрегують та транспортують дані до сервісного рівня (service layer), який складається з додатків, які маніпулюють та об'єднують дані в інформацію, яка може відображатися на інформаційній панелі. Найвищим шаром стеку є рівень вмісту (content layer) або користувальницький інтерфейс.

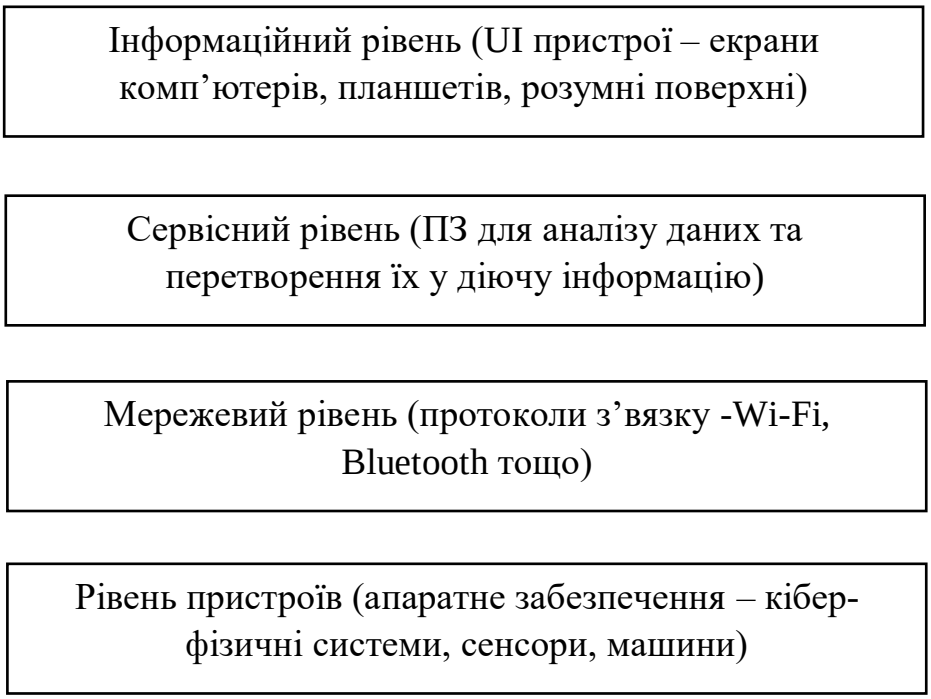


Рис. 1.1 Архітектура систем IIoT

Однією з головних цілей є автоматичний моніторинг та виявлення аномальних подій, змін і девіацій на зібраних даних. Виявлення аномалій включає всі методи, спрямовані на ідентифікацію шаблонів даних, які відхиляються від очікуваної поведінки (норми). З огляду на велику кількість приладів і датчиків на типовому підприємстві, надзвичайно важливим є ефективний аналіз масиву даних в реальному часі для запобігання критичних ситуацій.

1.2 Задача виявлення аномалій у промислових системах

Виявлення несправностей або виявлення аномалій у промислових процесах є однією з важливих задач, що стоїть в рамках четвертої промислової революції (Industrie 4.0). Детектори аномалій у промислових системах, що базуються на порогових значеннях, вимагають великих зусиль та часу для роботи або мають схильність втрачати бажані події, оскільки сучасні промислові системи генерують великий обсяг даних у вигляді часових рядів, мають різноманітні виробничі процеси та рідкі появи аномалій. Враховуючи цей факт, актуальним залишається питання

дослідження можливості застосування різноманітних підходів та методів виявлення аномалій.

1.2.1 Типи аномалій

Аномалії можуть бути двох видів: одновимірні та багатовимірні. Одновимірні аномалії можна знайти при розгляді значень в просторі з однією ознакою. Багатовимірні аномалії можна знайти в n-мірному просторі (з n-ознак). Перегляд розподілу в n-мірних просторах може бути дуже складним для аналізу людиною, тому необхідно мати модель, яка зможе вирішити це завдання. Залежно від середовища аномалії можуть поділятися на: точкові, контекстні або колективні.

- Точкові аномалії - це окремі точки даних, які знаходяться далеко від решти розподілу. Також відомі як глобальні аномалії, ці відхилення існують далеко за межами цілого набору даних (рис. 1.2).

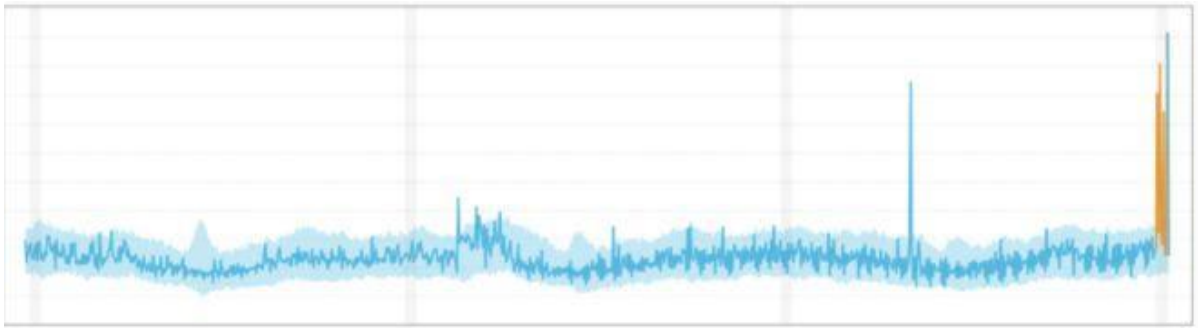


Рис. 1.2 Точкові аномалії

- Контекстуальними аномаліями можуть бути шуми в даних, наприклад, пунктуаційні символи при здійсненні аналізу тексту або фоновий шум при розпізнаванні мови. Ці аномалії, які також називаються умовними відхиленнями, мають значення, які суттєво відхиляються від інших точок даних, що існують у тому ж контексті. Аномалія в контексті одного набору даних може не бути аномалією в іншому. Ці відхилення є загальними для даних часових рядів, оскільки ці набори даних є записами певних величин за певний

період. Значення існує в межах глобальних очікувань, але може виглядати аномальним за певних сезонних моделей даних (рис. 1.3).

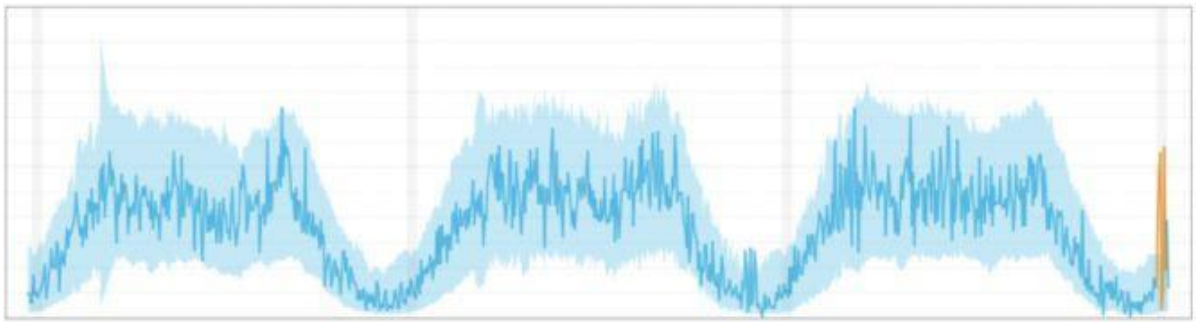


Рис. 1.3 Контекстуальні аномалії

- Коли підмножина точок даних у наборі є аномальною для всього набору даних, ці значення називаються колективними аномаліями. У цій категорії індивідуальні значення не є аномальними ні в глобальному, ні в контекстному плані. Такі типи можна побачити, якщо вивчати різні часові ряди разом. Індивідуальна поведінка може не відхилятися від норми в певному наборі даних часових рядів. Але в поєднанні з іншим набором даних часових рядів стають очевидними більш суттєві аномалії (рис 1.4).

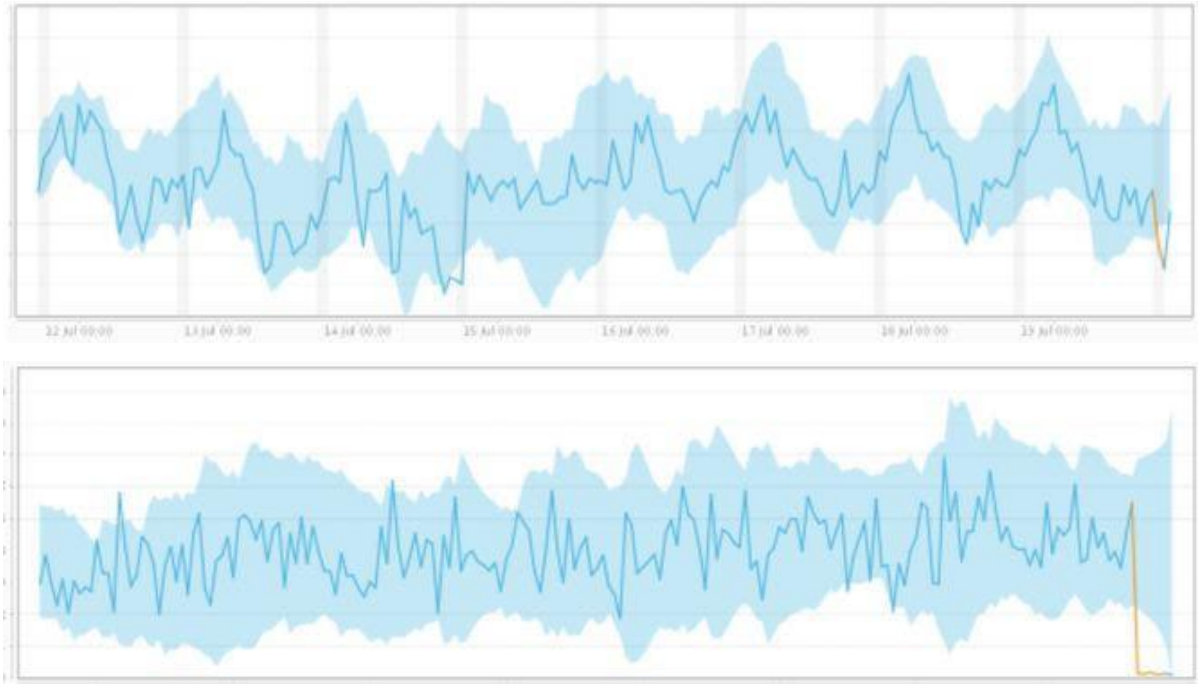


Рис. 1.4 Колективні аномалії

1.2.2 Постановка задачі для промислових систем

Розглядається клас промислових систем Industrie 4.0, в яких можуть бути присутні аномалії різної природи:

- Система може мати кілька режимів роботи. Несанкціонована зміна режиму може привести до відмови або поломки обладнання, швидкого зносу його функціональних компонентів.
- В межах одного режиму можуть спостерігатися відхилення спостережуваного процесу від нормального поведінки системи:
    - Аномалії, пов'язані з відхилення амплітуди від нормального сигналу.
    - Аномалії, пов'язані з відхилення в часі, тривалості або послідовності етапів процесу роботи системи.

З описаних видів, найбільш складними в рамках виявлення аномалій, є останні. Виробничі процеси часто є дуже динамічними і мінливими, найбільш інформативними є просторово-часові характеристики, якісне моделювання яких важко. В даному дослідженні розглядаються обидва зазначених види аномалій.

На вхід моделі надходять сенсорні дані датчиків промислової системи в рамках IIoT (промисловий інтернет речей), які є аналоговими часовими рядами з достатньою частотою дискретизації. Часові ряди – це послідовності вимірювань, упорядкованих в невипадкові моменти часу (рис. 1.5). На відміну від аналізу випадкових вибірок, аналіз часових рядів грунтується на припущенні, що послідовні значення у файлі даних спостерігаються через рівні проміжки часу (тоді як в інших методах нам не важлива і часто не цікава прив'язка спостережень до часу).

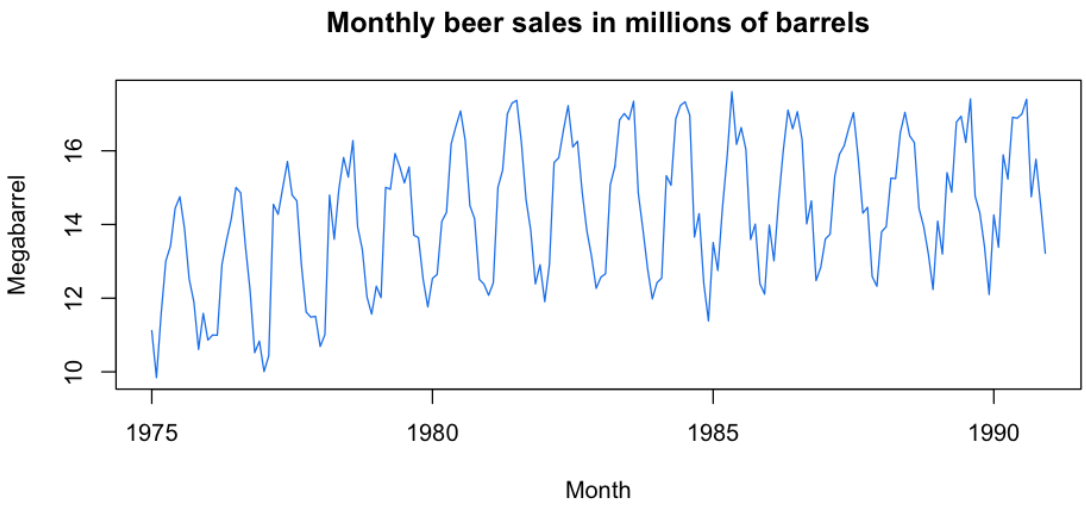


Рис. 1.5 Приклад часового ряду на графіку

Шляхом обробки даних необхідно вирішити задачу бінарної класифікації, тобто отримати поділ безлічі часових рядів на два класи - нормальні і аномальні, виявити межі аномальних інтервалів з точністю до одного кроку дискретизації.

Розроблена модель повинна відповідати всім вимогам сучасних промислових систем, а саме мати можливість застосування в реальному часі, мати низьку обчислювальною складністю, підтримувати велику кількість сенсорів, обробку в один прохід.

1.3 Виявлення аномалій за допомогою DL автоенкодерів

Задачу виявлення аномалій можна вирішувати за допомогою спеціальної архітектури штучних нейронних мереж – автоенкодера (автокодувальник). Це нейронні мережі прямого поширення, що відновлюють вхідний сигнал на виході. Він відновлюється з помилками через втрати під час кодування, аде для їх мінімізаціїї мережа повинна навчатися відбирати якомога більш важливі ознаки. Загальна структура автоенкодера (рис. 1.6) складається з наступних частин:

- Кодувальник (енкодер). Приймає вхідні дані високої розмірності та перетворює їх на приховані низькорозмірні дані. Вхідна розмірність даних до мережі кодувальника більша, ніж розмірність вихідного сигналу.

- Декодувальник (декодер). Отримує вхідні дані з виходу кодувальника. Завданням декодувальника є реконструкція вхідних даних. Вихідний розмір мережі декодувальника більший за розмір вхідного сигналу.
- Код. Являє собою прихований шар, що містить стиснене представлення вхідних даних. Кількість прихованих одиниць у коді називається розміром коду.

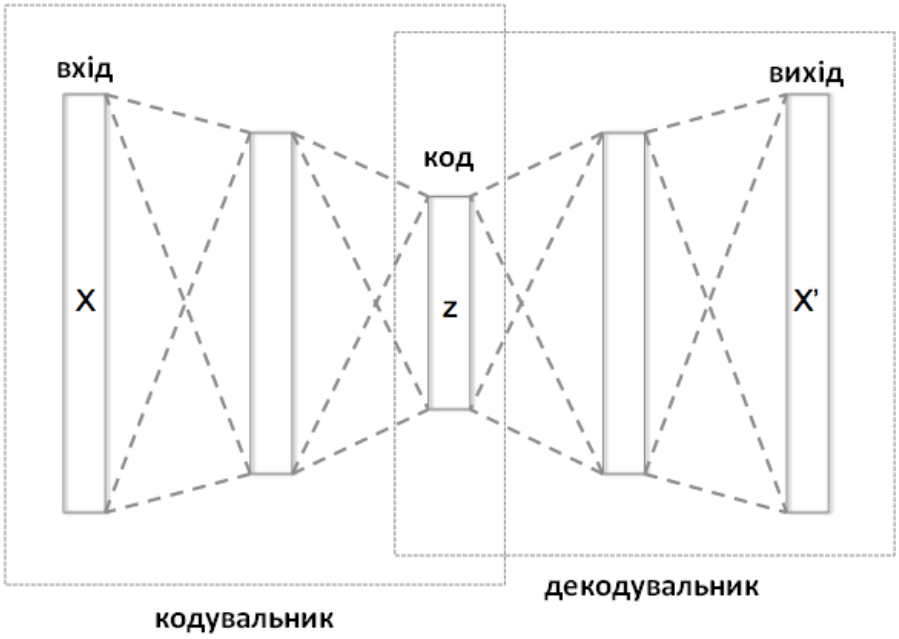


Рис. 1.6 Схематична структура автоенкодера

Нехай існує кодувальник g та декодувальник f. Кодувальник переводить вхідний сигнал в його стиснуте представлення (код): $z = g(x)$, а декодувальник відновлює вхідний сигнал за його кодом $x = f(z)$. Змінюючи f та g, автокодувальник намагається вивчити тотожну функцію $x = f(g(x))$, мінімізуючи помилку L. При цьому сімейство функцій f та g якось обмежені, щоб автокодувальник відбирав найбільш важливі властивості сигналу.

Якщо вхідні ознаки незалежні одна від одної, стиснення та подальша реконструкція представляються дуже складним завданням. Однак, якщо в даних існує якась структура (тобто кореляція між вхідними ознаками), цю структуру можна вивчити і використовувати під час проходження даних через приховані шари мережі.

Приховані шари - це ключовий атрибут автокодувальників, адже без їх присутності можливо легко навчитися просто запам’ятовувати вхідні значення та передавати їх далі по мережі.

Існують різні види автокодувальників в залежності від особливостей їх архітектури:

- Знешумлювальний автокодувальник (Denoising Autoencoder)
- Розріджений автокодувальник (Sparse Autoencoder)
- Глибокий автокодувальник (Deep Autoencoder)
- Стискаючий автокодувальник (Contractive Autoencoder)
- Згортковий автокодувальник (Convolutional Autoencoder)
- Неповний автокодувальник (Undercomplete Autoencoder)
- Варіаційний автокодувальник (Variational Autoencoder)

Автокодувальнки мають широке застосування при вирішенні наступних задач:

- Зниження розмірності. Кодувальник кодує вхідні дані у прихований шар, щоб зменшити розмірність лінійних та нелінійних даних.
- Знешумлення зображень. Пошкоджене зображення можна відновити до початкового стану.
- Генерація зображень. Варіаційні автоенкодери (Variational autoencoder) використовуються для створення зображень.
- Стиснення зображень.
- Інформаційний пошук.

Також автоенкодери є ефективними для вирішення задачі виявлення аномалій. Модель вивчає стисле представлення вхідних даних, з якого необхідно відтворити початковий сигнал. Навчаючи її тільки на нормальних даних можливо домогтися здатності реконструкції вхідних даних з високою точністю. Аномальні дані в достатній мірі відрізняються від нормальних, тому навченій моделі зі своїми вагами з’єднань не вдасться їх відтворити і подальша помилка втрати буде високою. Нормальні ж дані навпаки легко відновлюються автоенкодерами, тому втрати будуть низькими (рис. 1.8). Все, що перевищує певне порогове значення вважається аномалією, як це, наприклад, зображено на рис. 1.7.

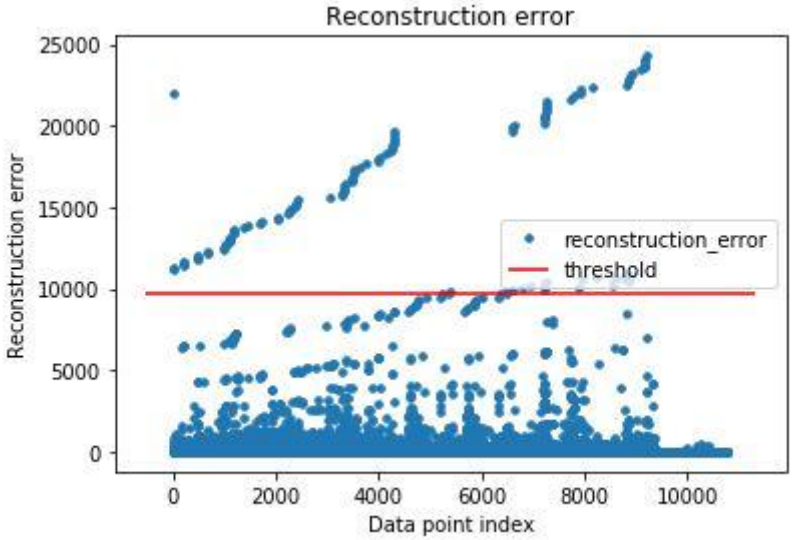


Рис 1.7 Демонстрація певного порогового значення

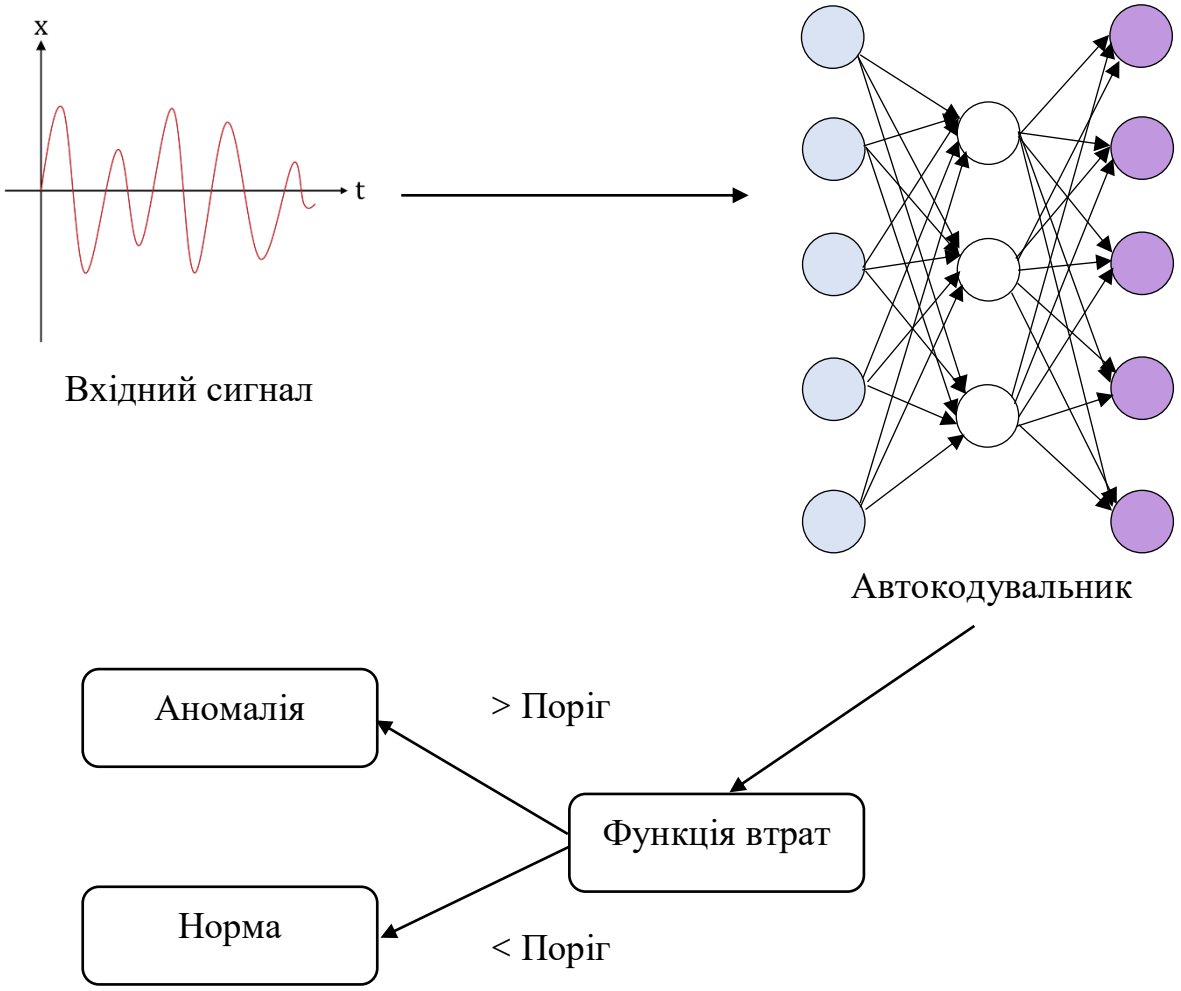


Рис 1.8 Виявлення аномалій за допомогою автоенкодеру

1.4 Висновки до розділу

У даному розділі розкрито поняття систем промислового інтернету речей (industrial IIoT), однією з головних проблем яких є ефективне виявлення аномалій. Наведена постановка задачі виявлення аномалій для промислових систем, описані різні підходи до її вирішення, класифікація типів аномалій. Більш детально розглянута архітектура нейронних мереж – автоенкодерів, наводиться їх класифікація та алгоритм виявлення аномалій.

Основними обмеженнями наведених підходів до вирішення задачі є їх вузька спеціалізація до конкретних систем і неефективність в інших схожих застосуваннях. В інших системах, з іншими динамічними характеристиками, моделі часто демонструють гірші результати. Тому актуальним є проведення дослідження меж застосування автоенкодерів для задачі виявлення аномалій, визначення найбільш робастної архітектури, що здійснюється у наступних розділах.

РОЗДІЛ 2
МОДЕЛІ ТА МЕТОДИ ДОСЛІДЖЕННЯ

2.1 Модель автоенкодера

У даній роботі завдання виявлення аномалій в промисловій системі вирішується в рамках підходу на основі моделювання (model-based diagnosis). Для моделювання системи використана архітектура мережі глибинного навчання, запропонована в дослідженні [3]. Ця модель є автоенкодером і поєднує згорткові шари (ConvNet) і довгої короткостроковій пам'яті (LSTM) (рис. 2.1).

Мережі LSTM приймають на вхід дані у вигляді багатовимірного масиву виду (sample, time_steps, features), де:

- **Sample**: кількість окремих послідовностей даних, які не пов'язані з жодною іншою послідовністю.
- **Timesteps**: кількість часових кроків в одній послідовності. Відповідає кількості кроків, які «пам’ятає» рекурентна нейронна мережа.
- **Features**: кількість різних змінних (сигналів) у рамках однієї послідовності. Відповідає кількості вимірів (розмірності) вихідних даних.

Другий параметр вказує на можливість навчання з урахуванням відміток часу, що вкрай важливо для отримання найбільш інформативних ознак з часових рядів для завдання виявлення аномалій. Рекурентна природа даного типу мереж сприяє збереженню стану блоків, що робить їх особливо придатними для аналізу змінних даних, наприклад показань різних датчиків.

Автоенкодер створює стисле представлення основних ознак вхідних даних і навчається їх відновлювати. У задачі виявлення аномалій модель навчається на спостереженнях нормального режиму роботи системи і потім застосовується для кодування / відновлення часових рядів з аномаліями. Здатність моделі відновлювати аномальні інтервали нижче, тому відбудеться зростання помилки відновлення сигналу. Таким чином, він дає інформативний критерій класифікації спостережень і виявлення аномалій.

При виборі моделі важливим фактором є робастність (robustness), тобто здатність моделі зберігати ефективність при значних змінах умов експерименту або в інших практичних додатках. Згідно [3], архітектура, показана на рис. 2.1, забезпечує оптимальний баланс між ефективністю виявлення аномалій і робастністю.

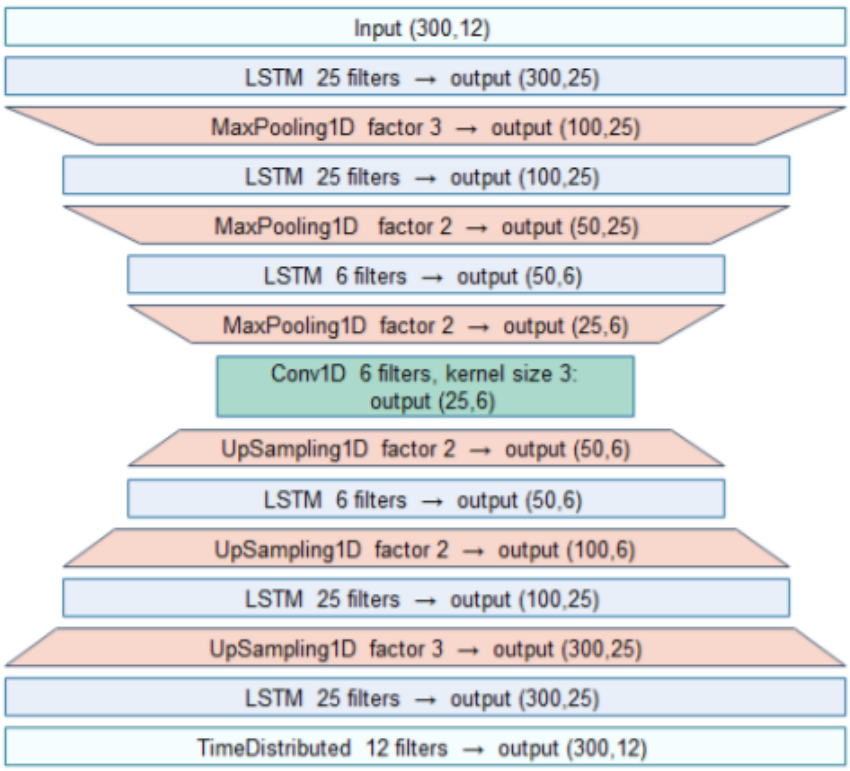


Рис. 2.1 Архітектура LSTM автоенкодера

Модель автоенкодера складається з таких шарів:

- Input – вхідний шар, який створює символічний тензорний об'єкт для використання з низкорівневими операціями TensorFlow, які приймають тензори в якості вхідних даних. Параметри шару задають форму, необхідну для LSTM.
- LSTM – шар, який реалізує довгу короткострокову пам'ять.
- MaxPooling1D – шар, який реалізує операцію підвибірки, для стиснення даних.
- UpSampling1D – шар підвищення дискретності, повторює кожен часовий крок задану кількість разів по часовій осі.

- Conv1D – шар, який реалізує одновимірну операцію згортки.
- TimeDistributed – дозволяє застосувати шар до кожного часового фрагменту входу.

У задачі виявлення аномалій в окремому випадку промислової системи IIoT, дана модель демонструвала високі показники ефективності: 62.3 ± 2.1% F1-міри і 59.1% повноти з 3350 істинно-позитивними прогнозами [3].

2.2 Система Industrial IoT, що обрана для дослідження

Для даного дослідження автором роботи була обрана термаусадочна система Vega (рис. 2.2) [4], яка являє собою пакувальну машину, що може бути розміщена на великих виробничих лініях в харчовій промисловості, зокрема для пакування напоїв. Вона групує вільні пляшки або банки за встановленими розмірами упаковки, загортає їх у поліетиленову плівку, а потім термоусаджує пластикову плівку, щоб об'єднати їх в пакет. Поліетиленову плівку подають в машину з великих котушок, а потім розрізають на довжину, необхідну для обмотки плівки навколо пачки товарів. Ріжучий вузол є важливою складовою машини для досягнення мети високої готовності. Тому лезо потрібно правильно налаштувати і підтримувати. Крім того, стан леза не можна візуально оцінити під час роботи, оскільки воно укладене в металевий корпус і має значну швидкість обертання.

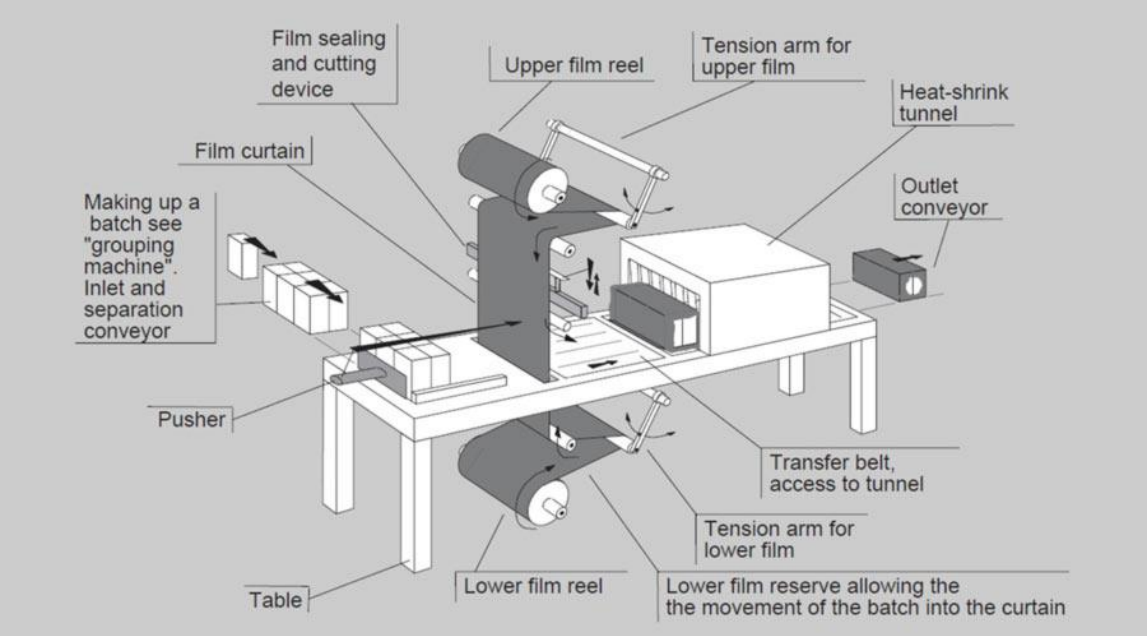


Рис. 2.2 Схема роботи системи Vega

Основними завданнями автоматизації для системи є предиктивне обслуговування (predictive maintenance) і прогнозування деградації компонентів (component degradation prediction). Їх ефективне рішення дозволяє скоротити час простою до мінімуму і уникнути виходу машини з ладу, забезпечити конкурентну перевагу перед іншими системами. Незаплановані простої через часті ремонтні роботи можуть привести до значних фінансових втрат. У загальному вигляді ці завдання формально зводяться до задачі виявлення аномалії. При цьому увага зосереджена на виявленні несправностей на ранніх етапах для оповіщення про необхідність проведення технічного обслуговування конкретних деградованих компонентах.

У термоусадочній системі Vega, створеній в рамках дослідницького проекту IMPROVE, який отримав фінансування від програми European Union's Horizon 2020 року, зібрані дані IIoT, які включають часові ряди вимірів датчиків компонентів протягом року. Дані включають в себе експерименти як з новим, так і зношеним ріжучим лезом для порівняння продуктивності машини. При цьому експерименти розділені на категорії в залежності від спостережуваного режиму роботи. Специфікація і опис режимів відсутня, відмінності між ними і динамічні

характеристики кожного необхідно виявити експериментально. Крім цього, машина може працювати з різною швидкістю.

Набір даних включає в себе 519 повних циклів роботи, кожен з яких складається з 2048 спостережень з інтервалом 4 мс., Таким чином 1 цикл роботи представляє 8-секундний інтервал. Серед отриманих параметрів - швидкість ріжучого леза, положення ріжучого диска, помилка запізнювання, крутний момент, положення і швидкість розмотування плівки. Відсутня розмітка аномальних і нормальних циклів. Система здатна працювати в 8 режимах, кожен з яких відрізняється параметрами, що спостерігаються (рис. 2.3).

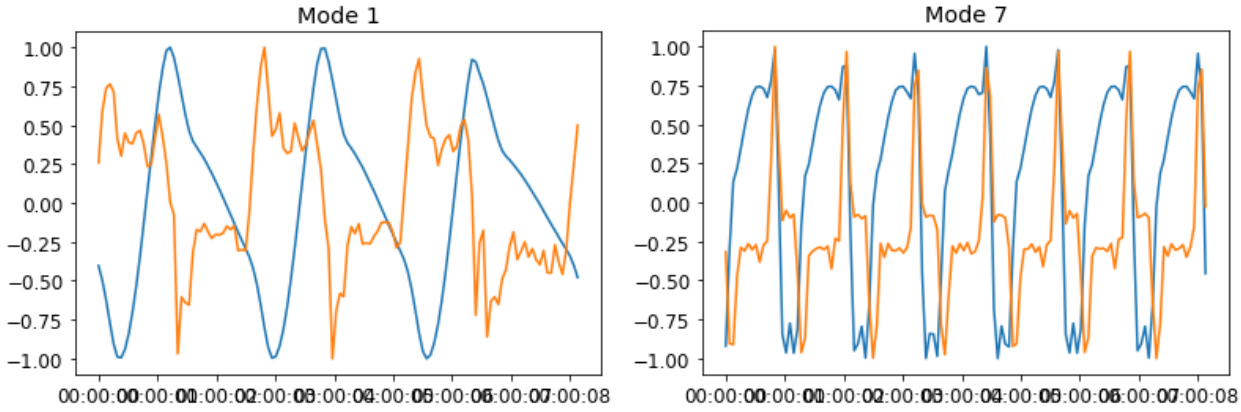


Рис. 2.3 Порівняння різних режимів роботи

У графіках показників датчиків термоусадочної системи присутні повторювані фрагменти, що свідчить про наявність циклічності в її роботі (рис. 2.4). Період приблизно становить 2,5 сек., оскільки в рамках одного експерименту, в залежності від швидкості і режиму, спостерігається близько трьох - чотирьох повторень. Врахування цієї особливості може істотно вплинути на ефективність роботи алгоритму виявлення аномалій.

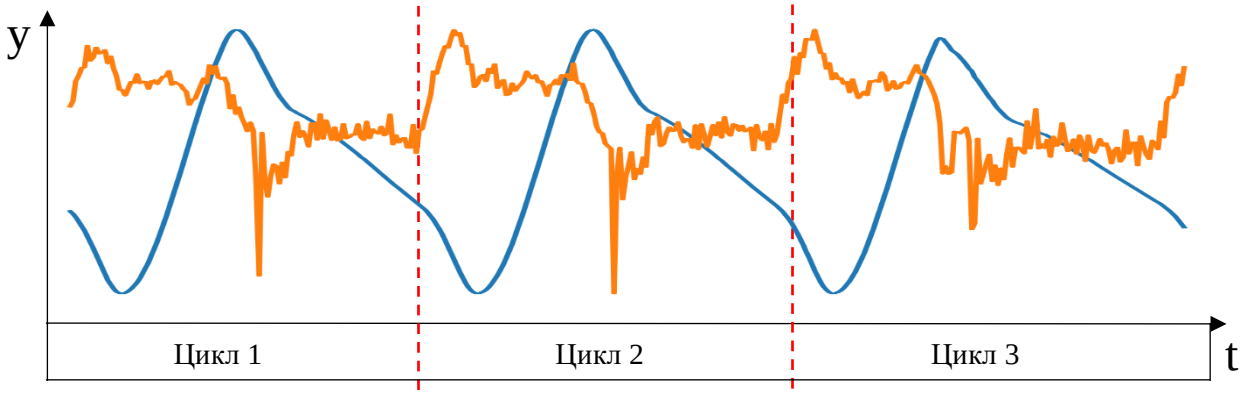


Рис. 2.4 Циклічний характер роботи системи

2.3 Методи дослідження моделі у задачі виявлення аномалій

Для дослідження розробленої моделі глибинного навчання задача виявлення аномалій була сформульована наступним чином:

1. Визначення аномальних циклів роботи. З огляду на відсутність інформації в першоджерелі про предметні відмінності режимів роботи пакувальної машини, після візуальної оцінки графіків циклів досліджуваної системи перший режим приймається як нормальний, а решта - як аномальні. При цьому, як підзадача, крім аномальності, розглядається виявлення належності поданої послідовності до якого-небудь з восьми режимів.
2. Визначення аномальних точок. На відміну від першого завдання, мається на увазі аномальність конкретного спостереження, а не циклу роботи в цілому. З огляду на відсутність явно розмічених аномальних і нормальних даних, автори самостійно генерують аномальні відрізки за допомогою відповідного алгоритму.

Для оцінки обраної моделі набір вихідних даних для вирішення поставлених завдань був розділений на три частини:

- Навчальна вибірка містить випадково вибрані 2/3 нормальних циклів і використовується для тренування автоенкодера.
- Тестова вибірка містить решту нормальних циклів і використовується для тестування перенавчання моделі.

- Перевірочна вибірка містить аномальні цикли і використовується для оцінки ефективності виявлення аномалій і вибору порога прийняття рішення.

В якості критеріїв ефективності виявлення аномалій були обрані матриця помилок і F1-міра. Перевагою останньої метрики є врахування як хибнопозитивних, так і хибнонегативних результатів, що дає адекватну оцінку для незбалансованих наборів даних, таких як термоусадочна система Vega. Крім того, використання цих критеріїв дозволяє провести пряме порівняння з попередніми дослідженнями.

У загальному випадку, в процесі навчання будь-якого алгоритму виявлення аномалій результуюча функція може вирахувати для будь-якого спостереження рівень аномальності. При цьому більшість точок отримають низькі оцінки, а нечисленні аномалії навпаки будуть виділятися більш високими. Таким чином для прийняття рішення відносити конкретне спостереження до аномалій чи ні, необхідно вибрати порогове значення. Якщо помилка відновлення більше ніж заданий поріг, точка є аномалією.

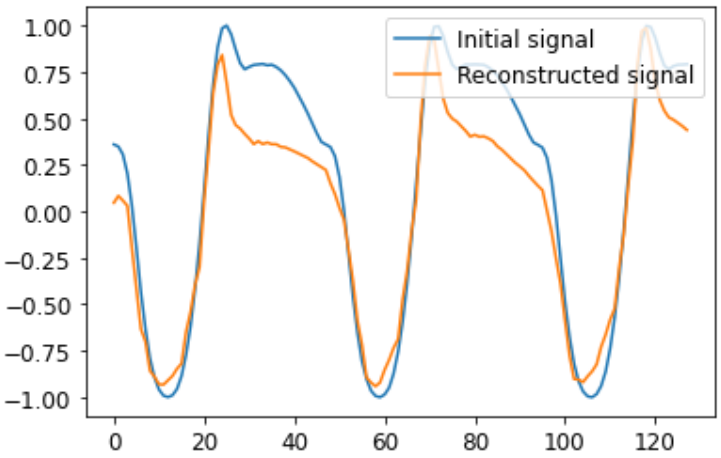


Рис. 2.5 Порівняння початкового і відновленого сигналу

Ефективність відновлення сигналу вимірювалася за допомогою функції втрат MAE. Вона більш надійна і більш стійка до викидів і більше підходить для даних великої розмірності. Нехай y і ŷ - вхідне і відновлене значення сигналу, а N - кількість часових рядів, тоді функція втрат MAE:

$$L(y, \hat{y}) = \frac{1}{N}\sum_{i=0}^{N}|y - \hat{y}_i| \quad (3.1)$$

Після навчання моделі було побудовано розподіл помилки відновлення сигналу для циклів роботи системи в першому і інших режимах, як для кожного спостереження, так і для всього режиму (у другому випадку до уваги береться середня помилка всіх спостережень), для циклів з аномаліями, створеними за допомогою власного алгоритму. (рис. 2.6)

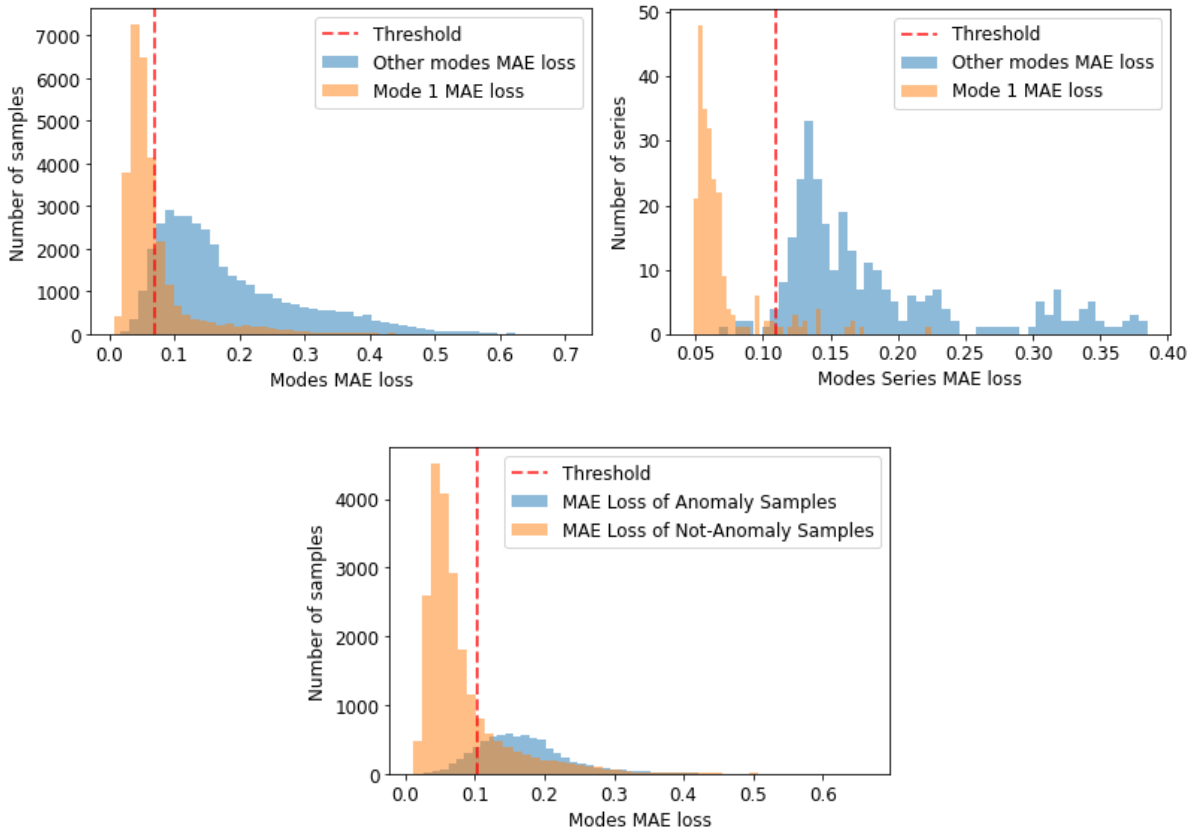


Рис. 2.6 Розподіли помилки реконструкції сигналу

2.4 Умови експериментів

В рамках досліджуваної системи було проведено певну кількість експериментів, використовуючи зазначену архітектуру моделі глибинного навчання. Вони відрізняються параметрами навчання, кількістю шарів нейронної мережі, способами підготовки навчаючої вибірки, обробкою даних тощо.

2.4.1 Підготовка даних

Оскільки моделі глибинного навчання (deep learning) чутливі до розміру вибірки спостережень, шумності сигналу, різним діапазонам їх абсолютних значень, дані термоусадочної системи перед використанням для навчання моделі піддавалися передобробці:

1. Стиснення по часу для зменшення частотності сигналу (downsampling). Це дозволяє значно зменшити кількість даних, що запобігає поганій сходимості навчання через велику кількість параметрів моделі. Сигнал s[n] може бути зменшений на коефіцієнт Q, зберігаючи кожну Q-ту вибірку та відкидаючи решту вибірок. Нова частота дискретизації становить 1/Q від початкової частоти дискретизації. (рис. 2.7)

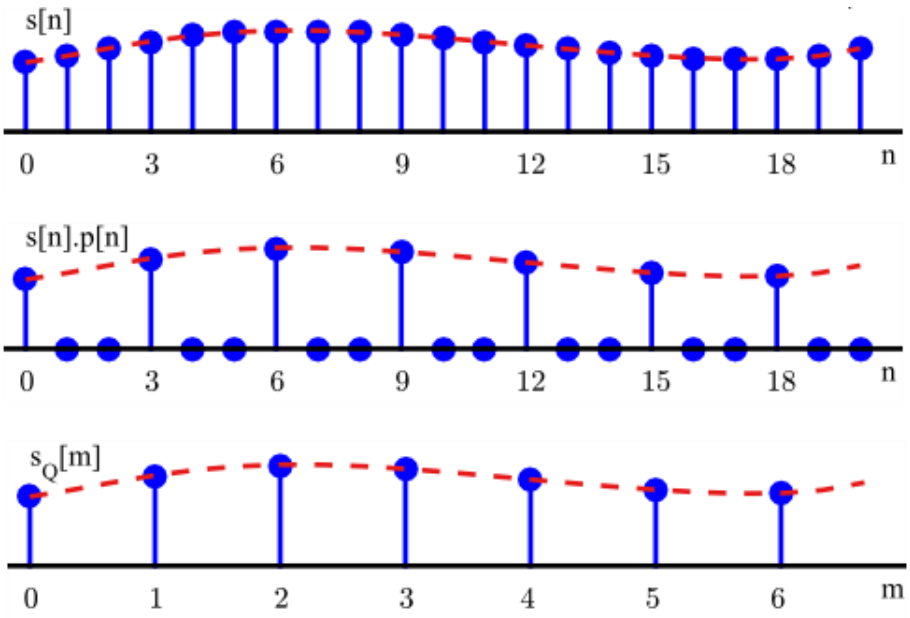


Рис. 2.7 Демонстрація операції downsampling в 3 рази

2. Нормалізація і приведення даних до діапазону [-1; 1]. Необхідна для усунення дисбалансу між даними вхідних сигналів. Будучи різними за фізичним змістом, дані сильно розрізняються між собою за абсолютними величинами. Наприклад, діапазон значень помилки запізнювання

певного експерименту варіюється від -0.668577 до 1.75886, а швидкості ріжучого леза від -4552.46 до 8232.88.

$$n_i = \frac{2 * (x_i - x_{min})}{x_{max} - x_{min}} - 1 \quad (3.2)$$

де n_i = нормалізоване значення ознаки, x_i = початкове значення ознаки, x_{min} = мінімальне значення ознаки x, x_{max} = максимальне значення ознаки x.

3. Згладжування даних за допомогою фільтра Гауса для усунення шуму. Фільтр застосовується до всього обсягу даних за допомогою ковзаючого вікна, тобто до всіх підсерій ширина яких дорівнює ширині фільтра.

$$w(n) = e^{-\frac{1}{2}(\frac{n}{\sigma})^2} \quad (3.3)$$

де n – відстань до центру фільтра, σ – ширина фільтра, e – число Ейлера.

2.4.2 Генерація аномалій з заданими параметрами

Дані пакувальної машини не містять явно зазначених локальних аномалій в циклах роботи або ж повністю аномальних циклів. Тому вони не можуть бути використані для оцінки здатності моделі до точної локалізації аномалії. Для вирішення цієї проблеми була згенерована окрема оціночна вибірка з штучно внесеними аномаліями (рис. 2.8). Такі аномалії мають наступні діапазони властивостей:

- **Length**: тривалість аномального інтервалу від 15% до 50% тривалості одного експерименту.
- **Amplitude**: амплітуда від 0.2 до 0.6
- **Slope**: різкість переходу між нормальним і аномальним режимом від 4 до 6.
- **Start**: час початку аномального інтервалу.

Пропонується алгоритм генерації аномального інтервалу в залежності від заданих параметрів. Для згладжування переходу між нормальним та зміненим сигналом використовується сигмоїда. Алгоритм враховує напрям деформації

сигналу, тобто якщо змінений сигнал виходить за максимальне чи мінімальне значення сигналу, напрям деформації змінюється на протилежний.

Алгоритм 2.1 Згладжування переходу між нормальним і зміненим сигналом

1: $Z \leftarrow$ **function** sigmoid(x, x_0, k)
2: **if** $-k*(x-x_0) > 10$ **then**
3: return 0
4: **else**
5: return $(1 + e^{(-k*(x-x0))})^{-1}$
6: **end if**
7: **end function**

Алгоритм 2.2 Генерація аномалій із заданими властивостями

1: $Y \leftarrow$ **function** Introduce_anomaly(X, *amplitude*, i, l, s)
2: $v_{min} \leftarrow \min(X_{i:i+l})$
3: $v_{max} \leftarrow \max(X_{i:i+l})$
4: **if** $v_{min} + amplitude < -1$ or $v_{max} + amplitude > 1$ **then**
5: return False
6: **end if**
7: **for all** $k \in \overline{i, i+l}$ **do**
8: $S_1 \leftarrow \text{sigmoid}(k, I + s/2, s/6)$
9: $S_2 \leftarrow \text{sigmoid}(-k, -I - l + s/2, s/6)$
10: $X_i \leftarrow X_i + \min(S_1, S_2) * amplitude$
11: **end for**
12: return True
13: **end function**

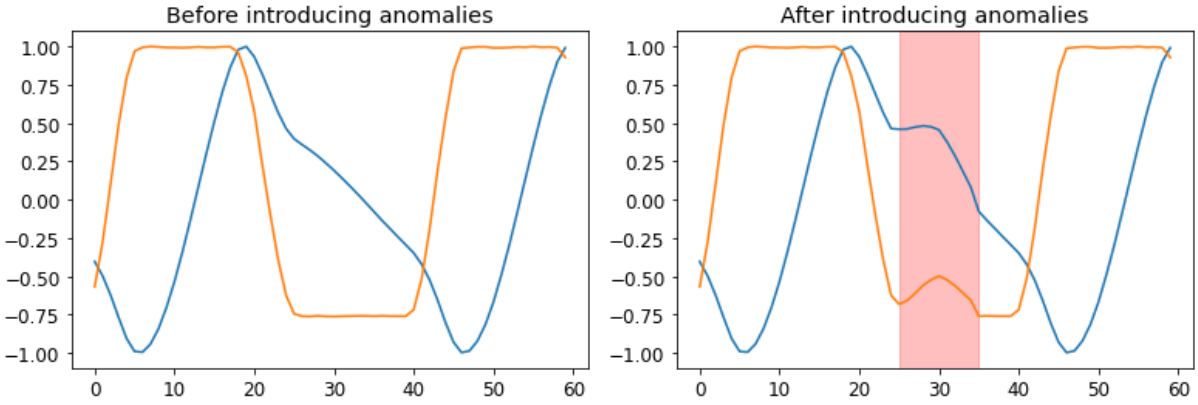


Рис. 2.8 Штучно внесена у сигнал аномалія

2.4.3 Способи підготовки навчальної вибірки

Враховуючи визначену раніше особливість системи, що полягає у наявності повторюваних циклів, доречним визначити декілька способів підготовки навчальної вибірки:

- Довжина навчальної вибірки дорівнює довжині інтервалу одного експеримента (спосіб 1).
- В рамках одного експерименту за допомогою ковзаючого вікна створюється декілька серій, враховуючи періодичність сигналу, як це показано на рис. 2.9 (спосіб 2).

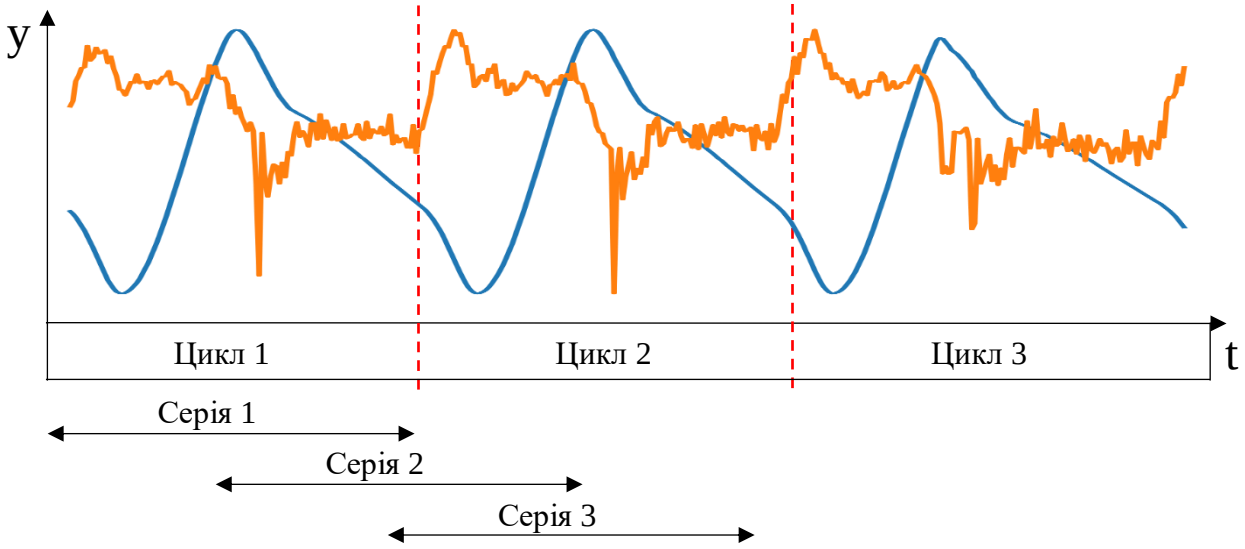


Рис 2.9 Підготовка навчальної вибірки, враховуючи періодичність сигналу

В другому випадку можливо використати прийом, що полягає у частковому перекритті серій. Ковзаюче вікно під час кожної ітерації рухається на певну кількість кроків, меншу від довжини усієї серії (рис. 2.10). Такий спосіб підготовки навчальної вибірки може як покращити ефективність моделі, так і призвести до перенавчання, тому доцільність використання необхідно перевірити.

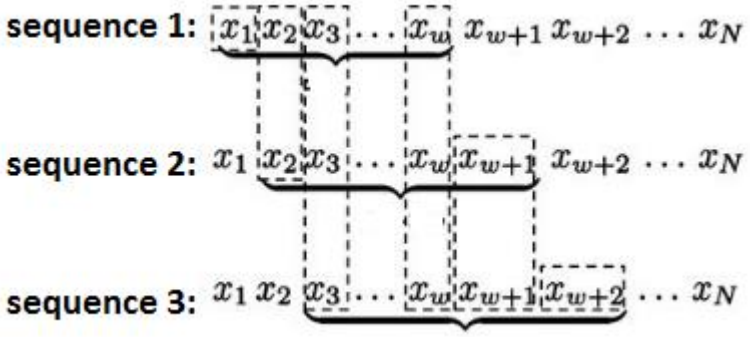


Рис. 2.10 Перекриття серій

2.4.4 Технічні умови

Моделі і алгоритми реалізовані мовою програмування Python, з використанням бібліотек машинного навчання Tensorflow і Keras. Тестування моделей виконано в хмарному сервісі Google Colab на основі Jupyter Notebook.

Для компіляції моделі використаний оптимізаційний алгоритм Adam з learning_rate 0.005, функція втрат MAE. Тестувалася різна кількість шарів LSTM та Conv1D, що підбиралася експериментальним шляхом на основі ефективності результуючої моделі. - по два для кодувальника і декодувальник. Кількість епох навчання - 130.

2.5 Висновки до розділу

У даному розділі сплановано експериментальне дослідження з використанням автоенкодера на основі довгої короткочасної пам’яті, що дозволить вирішити задачу

виявлення аномалій. Для дослідження меж застосування моделі було обрано промислову систему Vega, що являє собою термопакувальну машину.

Задача виявлення аномалій розглядається як в контексті одного спостереження (точки даних), так і в контексті цілого експерименту. Описані методи дослідження ефективності, способи підготовки навчаючої вибірки, алгоритм внесення власних аномалій у сигнал, різні архітектурні варіації обраної моделі.

Сплановане експериментальне дослідження дозволить повноцінно дослідити запропоновану модель глибинного навчання та визначити найбільш ефективну конфігурацію.

РОЗДІЛ 3
ПРОЕКТУВАННЯ І РЕАЛІЗАЦІЯ МОДУЛЯ ВИЯВЛЕННЯ АНОМАЛІЙ

3.1 Загальний опис проектованої системи

Розробляється програмний застосунок, який являє собою модуль виявлення аномалій для часових рядів показів датчиків IIoT. Задача полягає у проектуванні та реалізації програмного забезпечення у вигляді веб-додатку, який дозволяє користувачу наочно спостерігати вхідні сигнали сенсорів, встановлених на промислових системах. За допомогою розробленої моделі глибинного навчання програма має аналізувати дані на наявність аномалій. Для можливості підтримувати та розвивати проект в майбутньому важливо організувати правильну структуру. Компоненти системи повинні бути виділені в окремі модулі для простоти внесення змін.

При цьому застосунок має складатися з таких структурних частин:

- Сторінка авторизації. Користувач вводить свої дані для авторизації. У разі, якщо користувач з такими даними не знайдений, система видає відповідне повідомлення. При успішній авторизаціїї відбувається перенаправлення на інформаційну панель.
- Інформаційна панель. Це головна сторінка додатку, яка фактично є панеллю індикаторів (dashboard), що візуалізує вхідні дані та обраховані моделлю виявлення аномалій показники.
- Сторінка з історією. Дає можливість переглянути результати роботи програми, а саме історичні дані щодо аналізу вхідних сигналів на аномалії.
- Сторінка налаштувань. Дозволяє налаштувати параметри виявлення аномалій, сповіщень, завантажити власну реалізацію моделі глибинного навчання.

3.2 Моделювання архітектури системи за допомогою UML

Для опису архітектури модуля виявлення аномалій була використана мова UML. Вона являє собою систему позначень, яку можна застосовувати для об'єктно-орієнтованого аналізу і проектування. Її можна використовувати для візуалізації, специфікації, конструювання та документування програмних систем.

Діаграма варіантів використання (діаграма прецедентів) описує дійові особи и прецеденти, а також специфікації, що являють собою текстовий опис конкретних послідовностей дій (потока подій), які виконує користувач при роботі з системою.

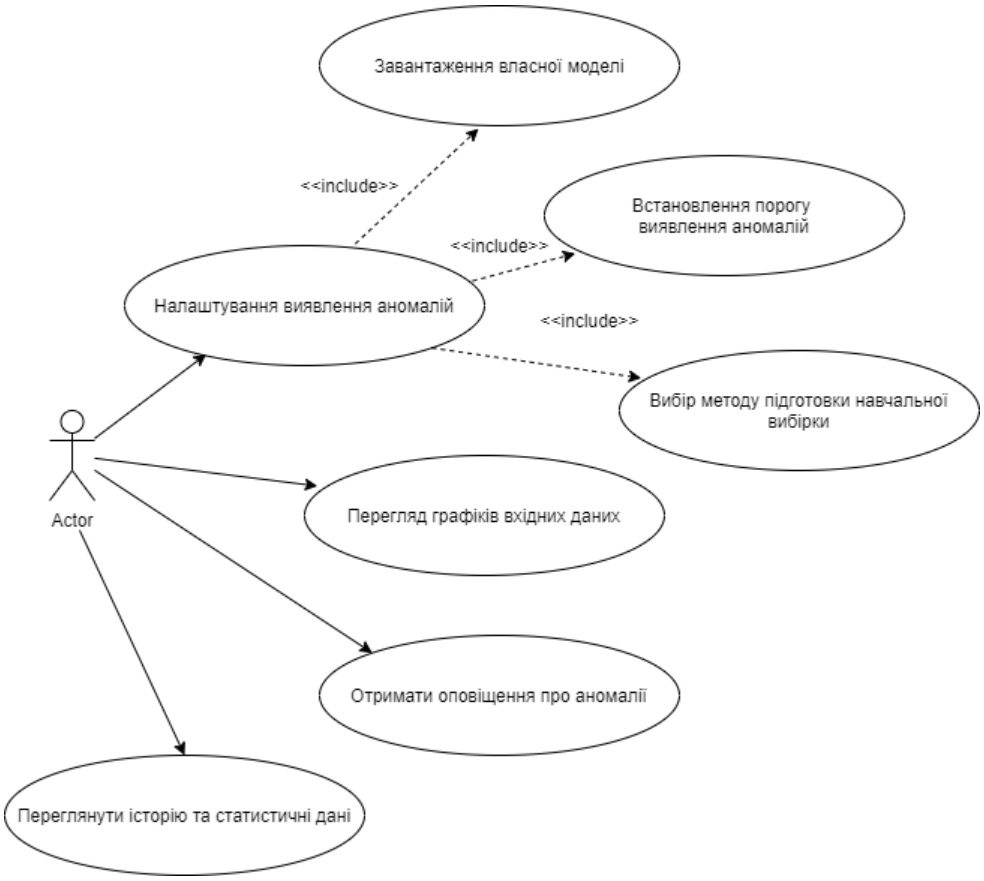


Рис. 3.1 Діаграма прецедентів

На рис. 3.1 зображений один актор, що представляє користувача модуля виявлення аномалій. Передбачені такі варіанти використання: перегляд графіків даних, історії, статистичних даних, налаштування параметрів виявлення аномалій, отримання оповіщень.

Діаграма розгортання служить для моделювання фізичної конфігурації системи, а саме фізичні вузли та зв’язки між ними. Діаграма дозволяє визначити розподіл компонентів системи за її фізичними вузлами, показати зв'язки між всіма вузлами реалізації системи на етапі її виконання, виявити та усунути проблемні місця для підвищення продуктивності.

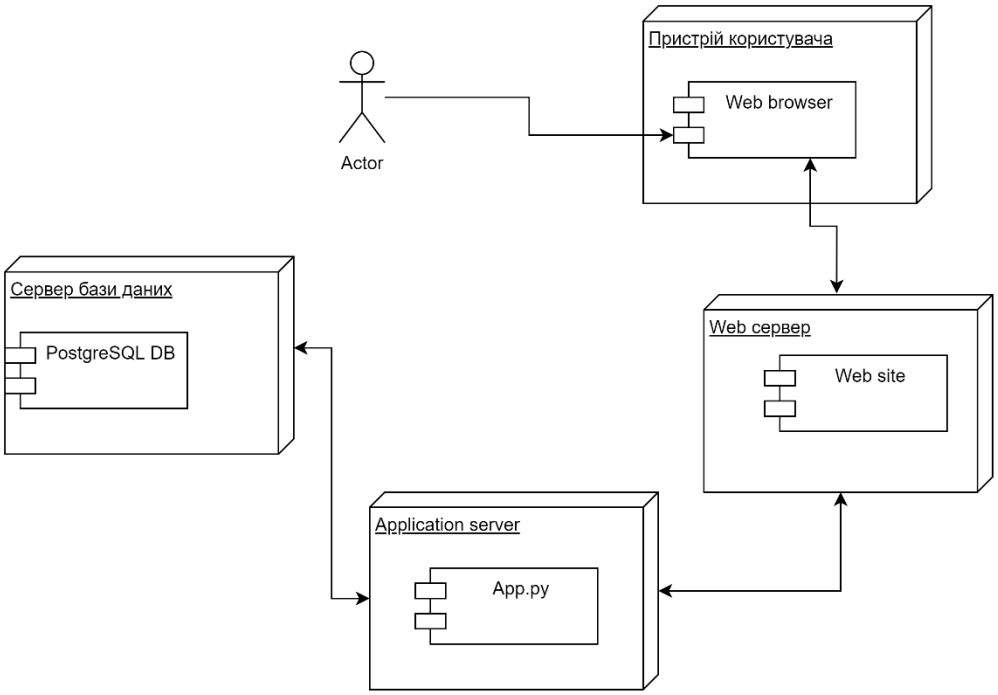


Рис. 3.2 Діаграма розгортання

На рис. 3.2 зображені наступні вузли:

- Пристрій користувача – це може бути персональний комп’ютер або мобільний телефон тощо. За допомогою веб-браузера користувач здійснює HTTP запити до веб-серверу, який повертає веб-сторінку.
- Web сервер відповідає на HTTP запити користувача та повертає HTTP відповіді.
- Application server – містить вихідний код програми та призначений для ефективного виконання процедур, на яких побудований додаток. Артефактом є програмне забезпечення.
- Сервер бази даних – необхідний для управління базою даних та виконання обслуговування, відповідає за цілісність і збереження даних, а

також забезпечує операції вводу-виводу при доступі користувача до інформації. Артефактом є СУБД PostgreSQL.

Діаграма послідовності дозволяє зобразити взаємодію об’єктів системи, впорядковану на проміжку за часом виконання. Розглянемо послідовність дій для основної функції системи, а саме виявлення аномалій у вхідному сигналі.

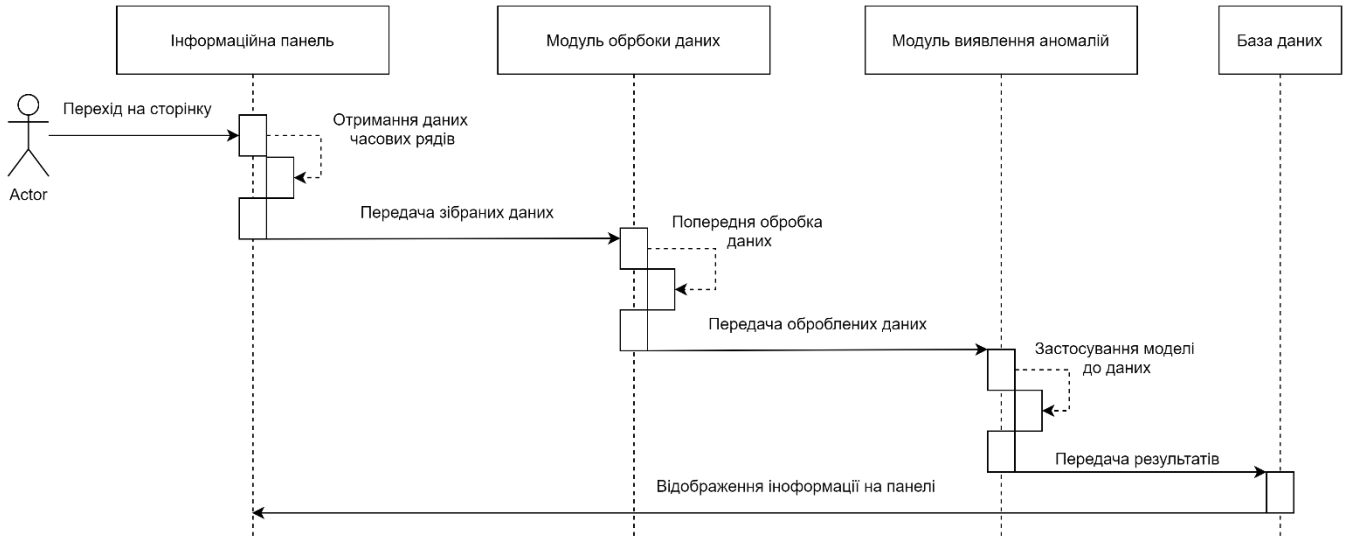


Рис. 3.3 Діаграма послідовності

Як видно з рис. 2.4, діаграма розбита на чотири окремі часові проміжки, кожен з яких характеризує певну складову частину системи. Все починається з переходу користувача на головну сторінку – інформаційну панель. Отримавши вхідні дані, програма візуалізує їх для користувача і передає в модуль обробки, де відбуваються різні операції для приведення даних у відповідність з вхідною розмірністю моделі глибинного навчання. Після застосування моделі до даних, статистичні результати експерименту записуються у базу даних, виводяться на інформаційну панель. У разі якщо було виявлено аномалії, користувач отримає повідомлення.

3.3 Функціональні та нефункціональні вимоги

Система повинна задовольняти наступним функціональним вимогам:

- Прийняття та обробка даних з датчиків IoT.

- Виявлення аномалій у вхідних сигналах.
- Відображення графіків сигналів, аномальних точок на них, гістограм розподілу помилки реконструкції та інших статистичних даних.
- Оповіщення користувачів про аномалії, як у вигляді нотифікацій в додатку, так і у вигляді електронного листа або повідомлення в Telegram.
- Налаштування параметрів виявлення аномалій, наприклад задання порогових значень, вибір способу підготовки навчаючої вибірки, вибір різних архітектурних модифікацій вбудованих моделей.
- Можливість завантаження в додаток власної моделі.

До нефункціональних вимог відносяться:

- Надійність – забезпечення надійного обміну даними під час здійснення запитів у мережі.
- Цілісність – забезпечення стабільного доступу до всіх елементів системи.
- Масштабованість – можливість розширити систему без потреби її зупинки.
- Ефективність – забезпечення швидкого обміну даними та відсуності навантажешь.
- Швидкість – швидке виконання всіх процесів між складовими компонентами системи.
- Зручність – зрозумілість всіх елементів інтерфейсу при роботі з системою управління.
- Функціонально повнота – надання користувачуві всіх доступих функцій для зручної роботи.

3.4 Проектування бази даних

Веб-застосунок підключений до головної бази даних, що містить інформацію про кожного користувача, його налаштування, історичні сенсорні дані, виявлені аномалії. Розглянемо кожну таблицю та її атрибути.

Таблиця user представляє список доступних користувачів системи, які можуть входити у інформаційну панель. Таблиця містить наступні атрибути:

- id – унікальний ідентифікатор користувача;
- firstName – ім’я користувача;
- lastName – прізвище користувача;
- email – поштова адреса користувача;
- password – пароль користувача (у зашифрованому вигляді).

Таблиця user_settings представляє список доступних користувачів системи, які можуть входити у інформаційну панель. Таблиця містить наступні атрибути:

- user_id – ідентифікатор користувача, якому належать налаштування;
- type1_threshold – встановлений поріг для виявлення аномалій 1-го типу;
- type2_threshold – встановлений поріг для виявлення аномалій 2-го типу;
- model – модель виявлення аномалій (за замовчуванням або власна);
- notifications – чи бажає отримувати повідомлення та яким способом;
- slicingType – спосіб підготовки навчальної вибірки.

Таблиця experiments містить набір всіх проведених експериментів та її оцінки моделью виявлення аномалій. Таблиця містить наступні атрибути:

- id – унікальний ідентифікатор експеримента;
- anomalyScores – рівні аномальності, обраховані моделлю;
- experimentData – дані часових рядів сенсорів промислової системи;
- user_id – ідентифікатор користувача, якому належить експеримент.

База даних представлена у реляційній моделі, структури зберігання даних у яких приведені до нормальних форм, відповідно до перших трьох правил нормалізації. Дані не мають надлишкових залежностей, підтримують повну атомарність та володіють первинними і зовнішніми ключами.

3.5 Реалізація основних функцій

3.5.1 Вибір технологій та засобів для реалізації програмного забезпечення

Для реалізації експерименту, описаного у попередньому розділі, була обрана мова програмування Python. Широкий асортимент бібліотек і фреймворків для машинного навчання сильно спрощує процес розробки та дозволяє економити час. Python має простий синтаксис та читабельність, що сприяє легшій розробці та швидкому тестуванню складних алгоритмів.

Безпосередньо для розробки моделі глибинного навчання були використані бібліотеки Keras та Tensorflow. Keras є одним з провідних високорівневих API для нейронних мереж, що підтримує багато движків глибинного навчання, зокрема Tensorflow від компанії Google. Серед переваг Keras варто відзначити вбудовані інструменти для препроцесингу даних, візуалізації моделей, різноманітні функції помилки, оптимізаторів, метрик, використання колбеків (callbacks) тощо. Keras дозволяє гнучко задавати архітектуру моделей глибинного навчання за допомогою Functional API (Додаток А).

Для обробки та аналізу вхідних даних використана потужна бібліотека Pandas. Перевагою є легка робота з csv файлами за допомогою вбудованої структури даних DataFrame.

Для візуалізації графіків сигналів системи, втрат тренування та валідації, побудови гістограм розподілу втрат реконструкції сигналу використана бібліотека matplotlib.

Нормалізація даних здійснена за допомогою класу MinMaxScaler з бібліотеки sklearn.

Для розробки веб-застосунку була обрана бібліотека Flask. Це мікрофреймворк, який майже не має залежностей від зовнішніх бібліотек. Особливістю є легкість, відсутність надлишкових модулів, адже додаток складається саме з тих частин, які безпосередньо потрібні. Flask найкраще працює, коли розробник користується її гнучкістю та можливостями кастомізації. Таким чином це гарний вибір для простих

веб-програм, які мають обмежений функціонал. Flask використовується для розробки веб-застосунків такими великими компаніями, як Airbnb, Uber, Netflix тощо. Flask базується на Werkzeug, службовій бібліотеці WSGI, та шаблонізаторі Jinja2 [5].

Основні переваги мікрофреймворку Flask:

- Flask має легкий і модульний дизайн. Розробникам необхідно декілька розширень для побудови повноцінного веб-додатку без зайвого перевантаження.
- ORM: розробники можуть підключати улюблений ORM, наприклад SQLAlchemy.
- Основний API фундаменту має гарну форму та цілісність.
- Документація є цілісною, добре структурованою та має велику кількість прикладів.
- Надзвичайно просто розгорнути Flask-додатки на production (на 100% відповідає стандарту WSGI 1.0).
- Функціональність обробки HTTP запитів.
- Висока гнучкість.

Автором використовувалось для розробки інтегроване середовище PyCharm, а також Postman для створення та тестування API.

3.5.2 Реалізація бази даних

Для реалізації бази даних була використана бібліотека SQLAlchemy [10] та розширення для веб-фрейморвку Flask-SQLAlchemy. Вона реалізує технологію ORM (Object-Relational Mapping), яка дозволяє оперувати таблицями баз даних у об’єктно-орієнтованому стилі. Цей інтерфейс є універсальним і не прив’язаним до жодної СУБД, що дозволяє працювати з будь-якою з них не модифікуючи вихідний код програми веб-додатку.

При використанні ORM процес конфігурації починається з опису таблиць бази даних, а потім із визначення власних класів, які будуть зіставлені з цими таблицями.

У SQLAlchemy ці дві завдання зазвичай виконуються разом, використовуючи систему, відому як декларативні розширення (Declarative Extensions), що дозволяє нам створювати класи, що включають директиви для опису фактичної таблиці бази даних, з якою вони будуть зіставлені.

Код моделі таблиці користувачів:

```
class User(db.Model):
    __tablename__ = 'users'
    id = Column(Integer, primary_key=True)
    firstName = Column(String(50), nullable=False)
    lastName = Column(String(50), nullable=False)
    email = Column(String(255), nullable=False, unique=True)
    password = Column(String(255), nullable=False)

    def __init__(self, firstName=None, email=None, lastName=None, password=None):
        self.firstName = name
        self.email = email
        self.lastName = lastName
        self.password = password
```

У якості СУБД була обрана PostgreSQL [9], оскільки вона використовується на хмарній PaaS-платформі Heroku, на який було завантажено проект. Для локальної розробки використона компактна вбудовувана SQLite, що зберігає усю базу даних в єдиному файлі на комп’ютері, де виконується програма.

3.5.3 Реалізація статистичних графіків

Оскільки головна сторінка являє собою інформаційну панель (dashboard) важливо використати правильні засоби для найбільш наочного представлення інформації. Для цього була використана Chart.js - проста і в той же час досить гнучка

бібліотека JavaScript для візуалізації даних, популярна серед веб-дизайнерів та розробників. Вона представляє собою прекрасне базове рішення для тих, кому не потрібне велике різноманіття типових графіків та індивідуальних настроїв, але кому достатньо, щоб графіки виглядали аккуратно, наглядно та інформативно.

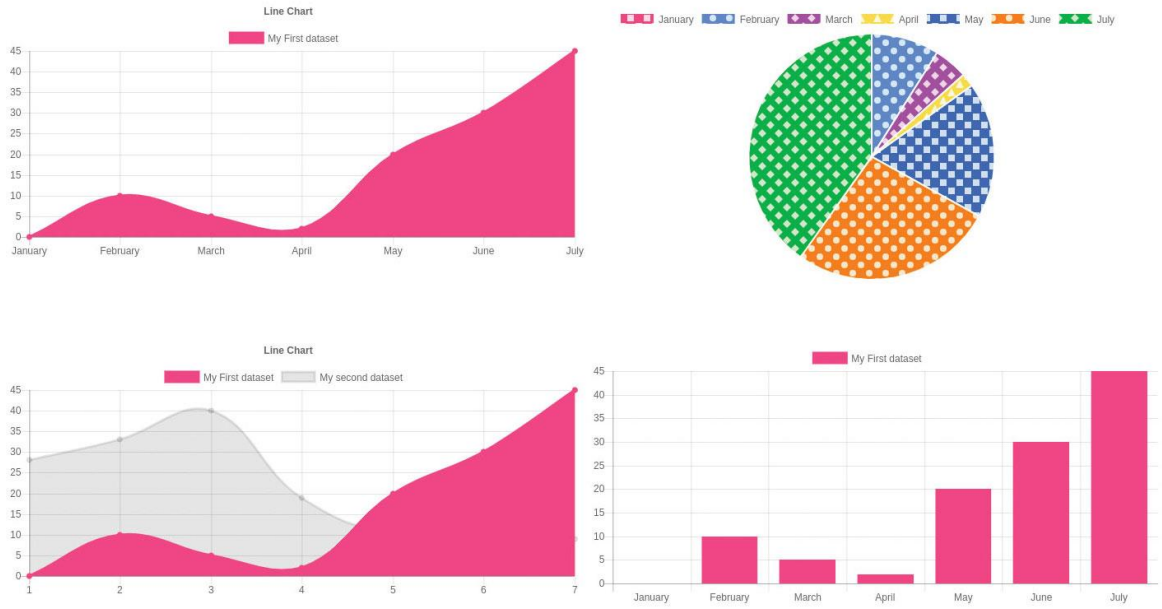


Рис. 3.4 Приклад графіків Chart.js

3.5.4 Реалізація API

За допомогою обраного фреймворку Flask, можливо легко та швидко реалізувати Web API. Додаток має набір кінцевих точок доступу, що продемонстровані у таблиці 3.1.

Таблиця 3.1

№	Назва	Метод	Запит
1	Вибірка всіх користувачів	GET	/api/users
2	Створення нового користувача	POST	/api/users
3	Вибірка налаштувань	GET	/api/user/settings
4	Оновлення налаштувань	PUT	/api/user/settings
5	Видалення даних	DELETE	/api/experiments

7	Виявлення аномалій	POST	/api/predict

Кожна кінцева точка також має набір додаткових параметрів, що передаються у самому запиті або у його тілі. Отже, усі запити, що пов’язані з роботою з користувачами, мають отримати параметр id, що ідентифікує з яким користувачем буде відбуватись взаємодія. Якщо ми виконуємо операції створення чи оновлення, програма має отримати модель даних у вигляді об’єкту зі всіма атрибутами.

Приклад реалізації кінцевої точки /api/predict:

```
@app.route('/api/predict', methods=['POST'])
def makeprediction():
    data = request.get_json()
    prediction = model.predict(data)

    return jsonify(prediction)
```

3.6 Висновки до розділу

Спроектований модуль для виявлення аномалій у показах датчиків систем IIoT. Описані компоненти системи та продемонстрована їх архітектура за допомогою UML [6] діаграм, а саме прецедентів, розгортання та послідовності. Модель передбачає розподіл застосунку на такі складові компоненти: клієнтська частина, модуль виявлення аномалій, база даних, модуль обробки даних.

У розділі описані використані інструменти розробки та реалізації програмного забезпечення, наведені їх основні переваги для мотивацію їх вибору. Автор ставить функціональні та нефункціональні вимоги для системи, що реалізовувалася. Також розглядається структура бази даних, яка необхідна для повноцінного функціонування веб-додатку, кінцеві точки доступу API.

Отже, було успішно проведено розробку власного програмного застосунку і реалізовано усі складові частини системи. У результаті отримано повноцінно функціонуючий засіб, що працює відповідно до спроектованої архітектури і дозволяє вирішити задачу виявлення аномалій для промислових систем.

РОЗДІЛ 4
РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

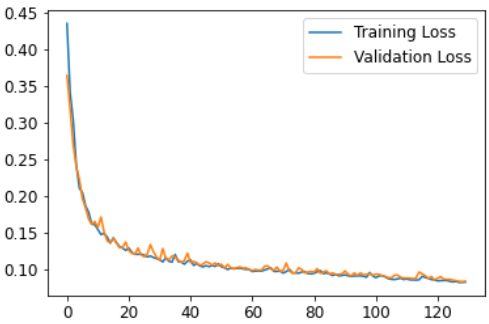
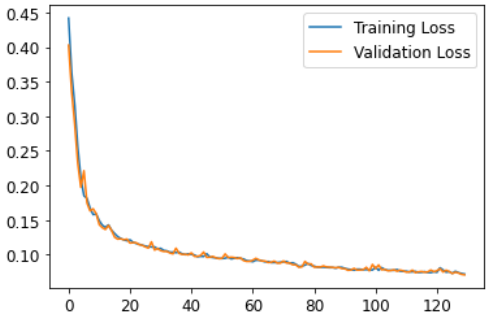
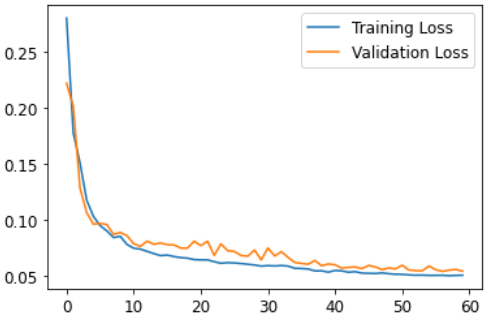
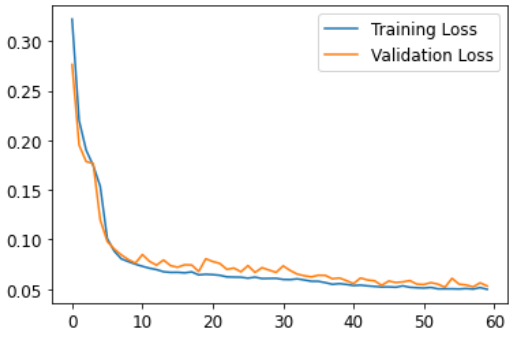
4.1 Результати експериментального дослідження системи

У результаті виконання дослідження було проведено декілька експериментів з використанням двох типів моделей:

- LSTM Autoencoder без використання Conv1D шару (130 епох навчання моделі).
- LSTM Autoencoder з використанням Conv1D шару, вихідна розмірність – 6, ширина вікна – 3 (60 епох навчання моделі).

Враховуючи два різних способа підготовки навчальної вибірки (нарізки даних) отримаємо чотири різні моделі автоенкодерів. В таблиці 4.1 представлений графік помилки навчання і валідації.

Таблиця 4.1

	LSTM	LSTM + Conv1D
Способ нарізки даних 1		
Способ нарізки даних 2		

Використовуючи оптимальні значення порогу були отримані наступні результати (Таблиця 4.2):

Таблиця 4.2

Тип моделі	Тип аномалій	Способ підготовки даних	Оптимальний поріг	F1	Accuracy	Precision	Recall
LSTM	1	Спосіб 1	0,098	0,857	0,837	0,924	0,878
	2		0,13	0,963	0,958	0,938	0,99
	3		0,124	0,643	0,762	0,52	0,843
	1	Спосіб 2	0,058	0,862	0,842	0,843	0,881
	2		0,068	0,952	0,946	0,949	0,955
	3		0,065	0,661	0,781	0,572	0,783
LSTM + Conv1D	1	Спосіб 1	0,081	**0,88**	0,86	0,839	0,926
	2		0,108	**0,967**	0,961	0,938	0,997
	3		0,106	**0,7**	0,816	0,6	0,842
	1	Спосіб 2	0,058	0,863	0,843	0,842	0,884
	2		0,073	0,952	0,946	0,955	0,948
	3		0,067	0,662	0,782	0,575	0,78

Таким чином найбільш ефективною виявилася архітектура моделі з використанням шарів LSTM і нарізкою навчальної серії без урахування циклічності роботи системи. Показник F1-міри склав 0,88 / 0,967 / 0,7 для виявлення аномалій 1/2/3 типів відповідно. Середній показник F1-міри - 0,849.

Підготовка даних з урахуванням циклічності роботи системи не має істотного впливу на ефективність моделі, але дозволяє підвищити швидкість збіжності навчання (з 130 до 60 епох) і істотно знизити варіацію помилки автоенкодера в нормальному режимі, як видно. Це вказує на більш низьку чутливість до перешкод і більш високу робастність моделі.

4.2 Висновки до розділу

При моделюванні термоусадочної системи Vega вказаний метод забезпечує досить високу ефективність виявлення аномалій (F1-міра 86 - 89%). Досліджувався вплив на ефективність виявлення аномалій способу підготовки даних і архітектурних модифікацій моделі. Найкращі показники забезпечує модель LSTM автоенкодера без використання Conv1D шарів з результатами середньої F1-міри до 84.9% на основі оцінки 3 різних завдань. У першому розглядається аномальність окремих точок даних, у другому – аномальність усього експерименту, а у третому – аномальність окремих точок в модифікованому сигналі за допомогою власного алгоритму. Ефективність роботи моделі у вигляді F1-міри склала 88%, 96.7% та 70% відповідно.

Отримані результати вказують на широкі перспективи застосування моделей на основі архітектури автоенкодера для вирішення задачі виявлення аномалій у часових рядах промислових систем.

ВИСНОВКИ

В даній роботі вирішувалася задача виявлення аномалій для систем Industrial IoT. Здійснено короткий огляд сучасних промислових систем в контексті четвертої промислової революції, однією з важливих проблем яких є своєчасне виявлення аномалій для запобігання виходу з ладу певних компонентів. Проаналізовані найпоширеніші підходи до вирішення вказаної задачі, проте існуючі методи мають суттєві обмеження.

Серед інших автором розглядається підхід на основі моделювання з використанням архітектури автоенкодерів довгої короткочасної пам'яті. Для оцінки ефективності даного методу була відібрана промислова система, дані сенсорів якої були використані під час експериментального дослідження.

Спроектовано і реалізовано програмне забезпечення, що емулює роботу реальної промислової системи. Розроблений модуль дозволяє аналізувати вхідні дані датчиків виробництв (часові ряди) на наявність аномалій. При цьому підтримується використання як вбудованих, так і власних моделей глибинного навчання. Застосунок, виконаний у вигляді інформаційної панелі, дозволяє наочно зобразити статистичні дані щодо аналізу вхідних даних системи.

Тестування різних архітектурних модифікацій автоенкодерів дозволило визначати найбільш вдалу конфігурацію. Середній показник F1-міри виявлення різних типів аномалій склав приблизно 85%, тому запропонований метод можна вважати ефективним. В інших дослідженнях, присвячених промисловій системі, що розглядається, альтернативні підходи вирішення задачі демонструють гірші результати.

Отже, враховуючи отримані у дослідженні результати, метод виявлення аномалій на основі архітектурних модифікацій автоенкодерів можна вважати робастним та таким, що має широкі перспективи застосування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Индустрия 4.0 [Електронний ресурс] – Режим доступу до ресурсу: https://www.it.ua/ru/knowledge-base/technology-innovation/industry-4.

2. M. Cerliani. Predictive maintenance with LSTM siamese network [Електронний ресурс] / Cerliani // 2019 – Режим доступу до ресурсу: https://towardsdatascience.com/predictive-maintenance-with-lstm-siamese-network51ee7df29767.

3. K. Kadomskyi. Evaluating Deep Learning Models for Anomaly Detection in an Industrial Transporting System [Електронний ресурс] / Kadomskyi Kyrylo. – 2020. – Режим доступу до ресурсу: http://ceur-ws.org/Vol-2845/Paper_2.pdf.

4. Vega shrink-wrapper component degradation [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://www.kaggle.com/inIT-OWL/vega-shrinkwrapper-runtofailure-data.

5. Flask’s documentation [Електронний ресурс] – Режим доступу до ресурсу: https://flask.palletsprojects.com/en/2.0.x/.

6. Буч Г., Рамбо Дж., Джекобсон А. Язык UML. Руководство пользователя. – 2006. – 29-33 с., 120-123 с.

7. A. Sagheer, K. Mostafa Unsupervised pre-training of a deep LSTM-based stacked autoencoder for multivariate time series forecasting problems / A. Sagheer, M. Kotb. – 2019. – №19038.

8. Susan Li Time Series of Price Anomaly Detection with LSTM [Електронний ресурс] / Susan Li. – 2020. – Режим доступу до ресурсу: https://towardsdatascience.com/time-series-of-price-anomaly-detection-with-lstm-11a12ba4f6d9.

9. PostgreSQL 13.3 Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://www.postgresql.org/docs/current/.

10. SQLAlchemy [Електронний ресурс] – Режим доступу до ресурсу: https://www.sqlalchemy.org/.

ДОДАТКИ

Додаток А

Код моделі архітектури автоенкодера

```
timesteps = X_train.shape[1]  # довжина часового ряду
features = X_train.shape[2]  # розмірність вхідного сигналу

input_layer = Input(shape=(timesteps, features))

x = LSTM(20, activation='tanh', recurrent_activation='sigmoid', return_sequences=True)(
    input_layer)  # Output shape (timesteps, nodes)
x = MaxPooling1D(2, padding='same')(x)  # Output shape (downsampled_steps, features)
x = LSTM(4, activation='tanh', recurrent_activation='sigmoid', return_sequences=True)(x)
x = MaxPooling1D(2, padding='same')(x)
encoded = x

x = Conv1D(filters = 6, kernel_size=3, activation='tanh', padding='same')(encoded)
x = UpSampling1D(2)(encoded)
x = LSTM(4, activation='tanh', recurrent_activation='sigmoid', return_sequences=True)(x)
x = UpSampling1D(2)(x)
x = LSTM(20, activation='tanh', recurrent_activation='sigmoid', return_sequences=True)(x
)
decoded = TimeDistributed(Dense(features))(x)

lstm_autoencoder_conv_bn = Model(input_layer, decoded)
lstm_autoencoder_conv_bn.compile(optimizer=Adam(learning_rate=0.005),
                                 loss='mae')
lstm_autoencoder_conv_bn.summary()
```

Додаток Б

Software Architecture Document (SAD)

Taras Shevchenko National University of Kyiv

Research of architecture and management of systems with distributed databases

Software Architecture Document (SAD)

CONTENT OWNER: Ernest Midoian

DOCUMENT NUMBER:	RELEASE/REVISION:	RELEASE/REVISION DATE:
1	1.0	June 2021

Table of Contents

1. Documentation Roadmap

1.1. Document Management and Configuration Control Information

- Revision Number: 1
- Revision Release Date: 04/06/2021
- Purpose of Revision: check the correctness and completeness of the created document

1.2. Purpose and Scope

Purpose: explore the limits of the applicability of anomaly detection methods in the Industrial IoT systems.

Scope: the tool is used to receive IIoT sensors data. Users can visualize incoming signals from industrial systems, identify anomalies using built-in pre-trained models or upload their own. Based on the results of previous neural network research, the application may well prove itself as a web service or platform for detecting anomalies.

1.3. Viewpoint Definitions

1.3.1. Management system model Viewpoint Definitions

1.3.1.1. Abstract

Views on the model and processes of control systems are considered. These include entry points, action processes, event handlers, components, modules, and exit points. Views characterize the point of view on a particular object of the system or sequence of actions.

1.3.1.2. Stakeholders and Their Concerns Addressed

Stakeholders and their concerns include:

- Customers who are interested in the correct operation of the system as a whole.
- Developers who are interested in this topic and carry out activities related to it.
- Lifecycle and process management managers who need to know how the software is designed.
- Testers that test components and identify system performance issues.
- Organizers and leaders, for whom it is important to control all components of the software package.

1.3.1.3. Elements, Relations, Properties, and Constraints

Elements are part of a software package that are created to solve a specific problem. The elements in the system are interconnected and perform a number of actions at different levels of software implementation. They define consecutive and parallel types of bonds that can be described in different constructions. Such constructions include cyclic, conditional and program-defined processes. The properties of the elements depend on the goals of the component in which the solutions and problems of this element are written.

Therefore, the model of the anomaly detection application has its own set of elements defined within the components in which they are implemented. The main components of the system include: user interface, anomaly detection module, data processing module, database.

Non-functional requirements:

- Integrity. The system model should provide stable access to all elements of the system, which provides the ability to access any module without problems.
- Reliability. Ensuring reliable data exchange during network requests, which involves the implementation of modules and error handling processes.
- Scalability. The model should be able to expand the system without the need to stop it, which will add new nodes during the operation of the system.
- Efficiency. The system module should provide fast data exchange and no load.

- Convenience. Provides clarity of all elements of the interface when working with the control system.
- Functionally complete. The model needs to provide the user with all available functions for convenient operation.

1.3.1.4. Language(s) to Model/Represent Conforming Views

Views can be formed using such representations:

- Plain text that can be presented in a variety of formats, such as lists, forms, or regular paragraphs.
- Representation in the form of uml diagrams representing a unified modeling language is used in the paradigm of object-oriented programming, and allow you to correctly describe the data in graphical form.
- Software based on the representation of the system model in the form of code, algorithmic sequences and constructions of programming languages.

2. Architecture Background

2.1. Problem Background

2.1.1. System Overview

The system is a fully functional tool that works according to the designed model and has the ability to distribute and interact with data in the network.

The main purposes of the system include:

- Anomaly detection. The main purpose of the system is to efficiently detect anomalies in sensor data.
- Data storage. The system allows you to store large amounts of information about the results of the neural network operation on certain data.
- Customization. The user should have ample opportunity to configure the parameters for anomaly detection, including setting a threshold, choosing various models of deep learning, etc.
- Visualization. The main page of the system should display various data, such as incoming signals, the number of anomalies, histograms of the reconstruction error distribution.

2.1.2. Goals and Context

The system is based on a client-server architecture, which can be later scaled up without additional changes in the program code for the collective work of users. This architecture requires that yhe servers are independent of each other. Customers also operate in parallel and independently of each other via network requests.

The advantages of this architecture are:

- the ability to distribute the functions of a computer system between several independent servers.

- save all data on a separate secure database server.

- the possibility of simultaneous work of a group of users.

- guarantee of data integrity.

The system development life cycle determines the processes, starting from the stage of formation of the design model and ending with the presentation of a fully functioning anomaly detection application. In this case, the design of the model, based on the architecture, is a key step that determines the main goals and objectives of the system. Different approaches and methods of building elements of the software complex, parts of the abstract model and algorithmic solutions are determined, which requires a detailed analysis and research of this area.

2.1.3. Significant Driving Requirements

The Architecture Tradeoff Analysis Method is a method for evaluating software architectures relative to quality attribute goals. Method evaluations expose architectural risks that potentially inhibit the achievement of an organization’s business goals.

Attributes define a set of characteristics that the system must have. Considering this topic, you need to focus on non-functional requirements for the system and the quality of the system as a result. On the other hand, if we consider the scenarios, then it is worth looking at the functional requirements and compare with the results of the components of each component of the system.

Depending on the stage of the life cycle and the tasks set on it, the attributes can be placed in order of importance. For example, if we organize a model of receiving sensor data from devices, it will be appropriate to consider the attributes of reliability, integrity, communication and organization of data exchange.

Quality attribute requirements are the means by which a system is designed to achieve its business objectives. These requirements must be recognized and taken into account early in the life cycle, and the architecture of the system and software must be designed to match their performance characteristics.

2.2. Solution Background

2.2.1. Architectural Approaches

Software architecture refers to the basic structure of a software system and the discipline to create

such a structure. Each structure consists of program objects, the properties they possess, and the relationships.

The software model implements several different software design patterns that facilitate work with the system itself and optimize the software elements of the application. In software engineering, software schemes are a common solution for reusable problems that typically arise in a particular software design context. This is not a complete design that can be converted directly to source code or machine code. Design templates are formalized best practices that programmers can use to solve common problems in developing a program or system. In a anomaly detection system, several design patterns are implemented such as template inheritance, streaming contents, package usage to structure the application. Web application implements an architectural template (pattern) MVC, which allows you to divide the related program logic into three interconnected elements.

Web user (Actor) is a system user who wants to use the anomaly detection module. This is a list of use-cases that represent major functionality of the final system:

- anomaly detection settings.
- download your own model.
- setting the threshold for detecting anomalies.
- choice of method of training sample preparation.
- view graphs of input data.
- receiving notification of anomalies.
- view history and statistics.

2.2.2. Analysis Results

After analyzing and researching the subject area and theoretical information, we designed our own model of the system, consisting of the following main components: user interface, anomaly detection module, data processing module, database. Each component plays its role and has limited functionality within its capabilities. The components organize a clear connection for the exchange of data between the components of the system.

Based on the designed model of the system, which implements the client-server, a software application for anomaly detection was developed. It is a web application that can be accessed from any device that has access to the network. Meets all functional and non-functional requirements and provides interaction with sensor data. Has the ability to scale depending on the user's wishes.

The design solution of the system model demonstrates how easy it is to organize a tool for anomaly detection, which implements all the necessary functionality, including receiving and preprocessing data, applying deep learning model to it.

2.2.3. Requirements Coverage

An important condition for working with the system is the availability of access to the network and the ability to open a web application.

Development Requirements:

- PyCharm IDEA or other development environment;
- Programing language Python;
- Flask;
- PostgreSQL;

3. Referenced Materials

IEEE 1471	ANSI/IEEE-1471-2000, IEEE Recommended Practice for Architectural Description of Software-Intensive Systems, 21 September 2000.
Barbacci 2003	Barbacci, M.; Ellison, R.; Lattanze, A.; Stafford, J.; Weinstock, C.; & Wood, W. Quality Attribute Workshops (QAWs), Third Edition (CMU/SEI-2003-TR-016). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003. <http://www.sei.cmu.edu/publications/documents/03.reports/03tr016.html>.
Bass 2003	Bass, Clements, Kazman, Software Architecture in Practice, second edition, Addison Wesley Longman, 2003.
Kyrylo Kadomskyi 2020	Evaluating Deep Learning Models for Anomaly Detection in an Industrial Transporting System, 2020.
Yuan Luo 2021	Deep Learning-based Anomaly Detection in Cyber-physical Systems, ACM Computing Surveys, 20 May 2021.

4. Directory

4.1. Glossary

Term	Definition
Software architecture	The structure or structures of that system, which comprise software elements, the externally visible properties of those elements, and the relationships among them [Bass 2003]. "Externally visible" properties refer to those assumptions other elements can make of an element, such as its provided services, performance characteristics, fault handling, shared resource usage, and so on.
Software design pattern	In software engineering, a software design pattern is a general, reusable solution to a commonly occurring problem within a given context in software design. It is not a finished design that can be transformed directly into source or machine code. Rather, it is a description or template for how to solve a problem that can be used in many different situations. Design patterns are formalized best practices that the programmer can use to solve common problems when designing an application or system.
Anomaly detection	Anomaly detection is a step in data mining that identifies data points, events, and/or observations that deviate from a dataset's normal behavior. Anomalous data can indicate critical incidents, such as a technical glitch, or potential opportunities, for instance a change in consumer behavior.
Flask	Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.
PostgreSQL	PostgreSQL, also known as Postgres, is a free and open-source relational database management system emphasizing extensibility and SQL compliance. It was originally named POSTGRES, referring to its origins as a successor to the Ingres database developed at the University of California, Berkeley.

4.2. Acronym List

SAD	Software Architecture Document
MVC	Model-View-Controller
DB	Database
UML	Unified Modeling Language
IEEE	Institute of Electrical and Electronics Engineers