Міністерство освіти і науки України

Луцький національний технічний університет

(повне найменування закладу вищої освіти)

Факультет комп'ютерних та інформаційних технологій

(повне найменування факультету)

Кафедра комп'ютерної інженерії та безпеки

(повне найменування кафедри)

КВАЛІФІКАЦІЙНА РОБОТА ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «МАГІСТР»

АДАПТИВНІ НЕЙРОННІ МЕРЕЖІ ДЛЯ ВИЯВЛЕННЯ АНОМАЛІЙ У ВЕЛИКИХ ПОТОКАХ ДАНИХ

ADAPTIVE NEURAL NETWORKS FOR ANOMALY DETECTION IN LARGE DATA STREAMS

спеціальність 123 Комп'ютерна інженерія
(шифр і назва спеціальності)
освітня програма «Комп'ютерна інженерія»
(назва освітньої програми)

Виконав: здобувач вищої освіти
групи КІм-21
Бірук Богдан Валентинович

_____
(підпис)

Керівник: к.т.н., доцент
Гордєєва Дар'я Валеріївна

_____
(підпис)

Кваліфікаційну роботу
допущено до захисту
«___» грудня 2025 р.
Гарант освітньої програми:
к.т.н., доцент Гринюк Сергій Васильович

_____
(підпис)

Луцьк – 2025 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *комп'ютерної інженерії та безпеки*
Ступінь вищої освіти: *магістр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *123 Комп'ютерна інженерія*
Освітня програма: *«Комп'ютерна інженерія»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ доц. Т.ТЕРЛЕЦЬКИЙ
«_____» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Біруку Богдану Валентиновичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *Адаптивні нейронні мережі для виявлення аномалій у великих потоках даних*

Керівник роботи *к.т.н., доцент Гордєєва Дар'я Валеріївна*

затверджені наказом закладу вищої освіти від «17» червня 2025 року № 290/01-02
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи 09.12.2025р.

3. Вихідні дані до роботи *Джерелом розробки є науково-технічна література та публікації в періодичних виданнях з даного питання, опубліковані зарубіжні та вітчизняні роботи в даній області, різні інтернет-ресурси технічного спрямування*

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):
Вступ
Теоретичні основи виявлення аномалій у великих потоках даних
Математична постановка та архітектура системи
Експериментальна оптимізація системи та інтерпретація результатів
Висновки

5. Перелік графічного (ілюстративного) матеріалу:
15 рисунків
11 таблиць

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Теоретичні основи виявлення аномалій у великих потоках даних	*Гордєєва Д.В., доцент*		
Математична постановка та архітектура системи	*Гордєєва Д.В., доцент*		
Експериментальна оптимізація системи та інтерпретація результатів	*Гордєєва Д.В., доцент*		
Нормоконтроль	*Багнюк Н.В., доцент*		
Гарант ОП	*Гринюк С.В., доцент*		
Показник запозичень тексту	***___%***		
Академічна доброчесність	*Міскевич О.І., ст.викладач*		

7. Дата видачі завдання 18.06.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	*Огляд літератури із досліджуваної проблеми*	До 01.08.2025 р.	
2.	*Теоретичні основи виявлення аномалій у великих потоках даних*	До 20.08.2025 р.	
3.	*Математична постановка та архітектура системи*	До 25.09.2025 р.	
4.	*Експериментальна оптимізація системи та інтерпретація результатів*	До 20.10.2025 р.	
5.	*Висновки та пропозиції*	До 25.10.2025 р	
6.	*Формування списку використаних джерел*	До 27.10.2025 р.	
7.	*Формування додатків*	До 30.10.2025 р.	
8.	*Оформлення ілюстративного матеріалу*	До 05.11.2025 р.	
9.	*Представлення остаточного варіанту кваліфікаційної роботи керівникові*	До 11.11.2025 р.	
10.	*Нормоконтроль*	До 29.11.2025 р.	
11.	*Інструментальна перевірка на академічний плагіат*	До 02.12.2025 р.	
12.	*Здача кваліфікаційної роботи та всіх супровідних документів на кафедру*	До 09.12.2025 р.	

Здобувач вищої освіти _____ Бірук Б.В.
(підпис) (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ Гордєєва Д.В.
(підпис) (прізвище, ініціали)

АНОТАЦІЯ

Бірук Б. В. Адаптивні нейронні мережі для виявлення аномалій у великих потоках даних. Рукопис.

Кваліфікаційна робота магістра ОП «Комп'ютерна інженерія» спеціальності 123 Комп'ютерна інженерія. Луцький національний технічний університет. Луцьк, 2025.

Кваліфікаційна робота складається з вступу, трьох розділів, висновків, списку використаних джерел, додатків.

Перший розділ присвячено теоретичним засадам виявлення аномалій у часових рядах великих потоків даних. Наведено класифікацію аномалій, розглянуто наслідки їх виникнення для кіберфізичних та інформаційних систем, проаналізовано класичні статистичні та сучасні методи машинного й глибокого навчання з акцентом на адаптивні нейронні мережі.

У другому розділі подано постановку задачі та описано архітектуру системи виявлення аномалій на основі LSTM-автоенкодера. Обґрунтовано вибір набору даних Numenta Anomaly Benchmark (NAB), наведено основні етапи попередньої обробки часових рядів та сформульовано підхід до прийняття рішень за помилкою реконструкції з урахуванням дрейфу даних.

Третій розділ містить результати експериментальних досліджень та аналіз ефективності розробленого детектора. Порівняно конфігурації моделі в офлайн і потоковому режимах за показниками точності, повноти, F1-міри, ROC/PR-характеристик та NAB-скору, оцінено вплив адаптивних механізмів на стійкість до дрейфу й можливість інтеграції системи у реальні прикладні середовища.

Ключові слова: аномалія, часовий ряд, потокові дані, адаптивна нейронна мережа, LSTM-автоенкодер, дрейф даних, Numenta Anomaly Benchmark.

ANNOTATION

Biruk B. Adaptive neural networks for anomaly detection in large data streams. Manuscript.

Master's qualification work of the study programme «Computer Engineering», specialty 123 Computer Engineering. Lutsk National Technical University. Lutsk, 2025.

The qualification work consists of an introduction, three chapters, conclusions, a list of references and appendices.

The first chapter is devoted to the theoretical foundations of anomaly detection in time series of large data streams. A classification of anomalies is presented, the consequences of their occurrence for cyber-physical and information systems are considered, and classical statistical as well as modern machine-learning and deep-learning methods are analysed, with an emphasis on adaptive neural networks.

The second chapter provides the problem statement and describes the architecture of the anomaly detection system based on an LSTM autoencoder. The choice of the Numenta Anomaly Benchmark (NAB) dataset is justified, the main stages of time series preprocessing are outlined, and a decision-making approach based on reconstruction error with regard to data drift is formulated.

The third chapter contains the results of experimental studies and an analysis of the effectiveness of the developed detector. Model configurations in offline and streaming modes are compared in terms of precision, recall, F1-score, ROC/PR characteristics and NAB score. The impact of adaptive mechanisms on robustness to drift and on the feasibility of integrating the system into real-world application environments is evaluated.

Keywords: anomaly, time series, streaming data, adaptive neural network, LSTM autoencoder, data drift, Numenta Anomaly Benchmark.

ЗМІСТ

ВСТУП

Сучасні інформаційні та кіберфізичні системи генерують величезні обсяги даних у режимі реального часу, що обумовлює необхідність автоматизованих методів аналізу та контролю. Одним із ключових завдань у цьому контексті є виявлення аномалій – подій або станів, що істотно відхиляються від нормальної поведінки системи. Аномалії можуть свідчити про несправності обладнання, вторгнення зловмисників, порушення бізнес-процесів або інші критичні ситуації. Традиційні методи аналізу часто виявляються недостатньо ефективними у високошумних, динамічних та великих потоках даних, що стимулює розвиток підходів на основі глибокого та адаптивного навчання. Останні дослідження демонструють потенціал нейронних мереж до ефективного виявлення аномалій, зокрема автоенкодерів та рекурентних моделей, проте значна частина робіт зосереджена на офлайн-сценаріях і не враховує дрейф даних та експлуатаційні обмеження реальних потокових систем.

Метою роботи є розробка та дослідження системи виявлення аномалій у великих потоках даних на основі адаптивних нейронних мереж.

Об’єктом дослідження є процес виявлення аномалій у потоках часових рядів кіберфізичних, промислових та інформаційних систем.

Предметом дослідження є методи та моделі адаптивних нейронних мереж, що застосовуються для виявлення аномалій у великих потоках даних.

Для досягнення поставленої мети у роботі поставлено наступні завдання:

– проаналізувати поняття аномалій у часових рядах, причини їх виникнення в кіберфізичних, промислових та інформаційних системах та дослідити наслідки появи аномалій у потокових даних для надійності, безпеки та стабільності роботи кіберфізичних і інформаційних систем;

– проаналізувати традиційні підходи та сучасні методи машинного та глибокого навчання для вирішення задач виявлення аномалій, визначити їх переваги й обмеження в умовах потокової обробки;

– дослідити доступні набори даних для задач виявлення аномалій у часових рядах та обґрунтувати вибір і використання набору даних, дослідити структуру й особливості обраного набору даних, провести його попередню обробку;

– формалізувати задачу виявлення аномалій у потоках часових рядів та побудувати модель LSTM-автоенкодера, що реалізує реконструктивний підхід;

– визначити критерії та метрики для оцінювання якості розробленої моделі;

– провести експеримент з симуляції моделі в офлайн та потоковому режимах;

– оптимізувати параметри моделі;

– оцінити можливості інтеграції розробленої системи в реальні кіберфізичні та інформаційні системи, визначити обмеження дослідження та сформулювати рекомендації й напрями подальшого розвитку.

Для розв’язання поставлених у роботі завдань використано комплекс загальнонаукових, математичних та спеціальних методів дослідження:

– методи аналізу та синтезу – для дослідження предметної області, класифікації типів аномалій у часових рядах, узагальнення підходів до роботи з потоковими даними та обґрунтування вибору набору даних;

– системний та структурно-функціональний аналіз – для формалізації процесу моніторингу кіберфізичних та інформаційних систем, виділення ключових підсистем детектора аномалій і побудови узагальненої архітектури системи;

– математичне моделювання та апарат теорії нейронних мереж – для формулювання задачі реконструкції нормальних режимів, побудови моделі LSTM-автоенкодера, задання функції втрат і правила обчислення скору аномальності;

– порівняльний аналіз – для зіставлення ефективності різних конфігурацій моделі й виявлення їхніх переваг і обмежень в офлайн та потоковому режимах;

– експериментальний метод – для проведення серії обчислювальних експериментів з навчання та донавчання моделі на підмножинах набору даних, перевірки гіпотез щодо впливу гіперпараметрів і адаптивних механізмів на якість виявлення аномалій;

– методи математичної статистики – для оцінювання результатів роботи детектора за метриками precision, recall, F1-міри, ROC/PR-кривими, AUC, а також для інтерпретації розподілів помилки реконструкції та скору аномальності;

– емпіричний метод оптимізації та методи програмної інженерії – для налаштування гіперпараметрів моделі та параметрів адаптації (розмір буфера, періодичність донавчання, квантиль порогування), а також для проєктування й реалізації програмного прототипу системи виявлення аномалій та засобів візуалізації результатів.

Інформаційну базу дослідження становлять наукові публікації, присвячені методам виявлення аномалій у потоках даних та кіберфізичних системах, нормативні та рекомендаційні документи щодо моніторингу технічних об’єктів.

Наукова новизна роботи полягає в розробленні адаптивної архітектури LSTM-автоенкодера для виявлення аномалій у потоках часових рядів, яка поєднує каузальне віконування, потокову стандартизацію, обчислення скору аномальності за помилкою реконструкції та динамічне порогування на основі ковзного квантиля. На відміну від відомих підходів із фіксованими порогами та суто офлайн-навчанням, запропонована схема дає змогу підтримувати стабільну якість детектування в умовах дрейфу даних без необхідності повного перенавчання моделі.

Практичне значення одержаних результатів полягає у можливості використання розробленої системи як програмного модуля моніторингу в кіберфізичних, промислових та інформаційних системах. Запропонований детектор забезпечує раннє виявлення відхилень у поведінці об’єктів, підвищує оперативність реагування та сприяє зростанню надійності функціонування сервісів і технологічної інфраструктури.

Апробація результатів. Основні положення та результати кваліфікаційної роботи було апробовано на II Міжнародній науково-практичній конференції «Progressive Approaches in Science and Engineering» (секція Computer engineering), яка проводилася 26-28 листопада 2025 року у м. Копенгаген, Данія, що підтверджує актуальність тематики дослідження та практичну значущість запропонованих підходів для задач потокового виявлення аномалій у кіберфізичних та інформаційних системах [1].

РОЗДІЛ 1
ТЕОРЕТИЧНІ ОСНОВИ ВИЯВЛЕННЯ АНОМАЛІЙ У ВЕЛИКИХ ПОТОКАХ ДАНИХ

1.1 Поняття аномалій та причини її виникнення

Аномалія у даних – це об’єкт, спостереження або підпослідовність, поведінка яких істотно відхиляється від закономірностей більшості даних [2, 3]. У літературі вживають близькі терміни outlier, deviation, novelty залежно від контексту і типу даних [4, 5]. З позиції машинного навчання аномалія – точка або сегмент, що не узгоджується з апріорною або емпірично сформованою моделлю «нормальної» поведінки виявляється за допомогою спеціальних алгоритмів, що дозволяють її ідентифікувати та класифікувати як відхилення [6]. На відміну від шуму, такі відхилення часто сигналізують про структурні збої, зовнішні втручання або ризикові стани й мають практичну цінність для діагностики відмов, виявлення атак чи шахрайства [6, 7].

Класично виділяють три основні типи аномалій. Індивідуальні (точкові) аномалії проявляються як суттєві одиничні відхилення від типового рівня показника, коли за незмінного контексту одна точка різко «вибивається» з загальної картини (наприклад, разовий стрибок температури, тиску чи мережевого трафіку). Контекстуальні аномалії зовні можуть мати звичайні значення, але стають нетиповими лише з урахуванням часу, просторового розташування або логічного оточення спостереження: значення, яке є нормальним удень, може бути аномальним уночі; аналогічно, однаковий рівень навантаження може по-різному інтерпретуватися для робочих та неробочих періодів. Групові (колективні) аномалії виникають тоді, коли сукупність окремих «нормальних» на вигляд точок формує нетиповий шаблон поведінки, наприклад послідовність дрібних, але систематичних змін, що разом утворюють підозрілу тенденцію чи аномальний цикл [8, 9]. У потокових часових рядах усі ці типи можуть з’являтися одночасно, накладатися один на один і

трансформуватися в часі, що зумовлює високу неоднорідність даних та істотно ускладнює їх інтерпретацію й автоматизоване виявлення (рис. 1.1) [6, 10].

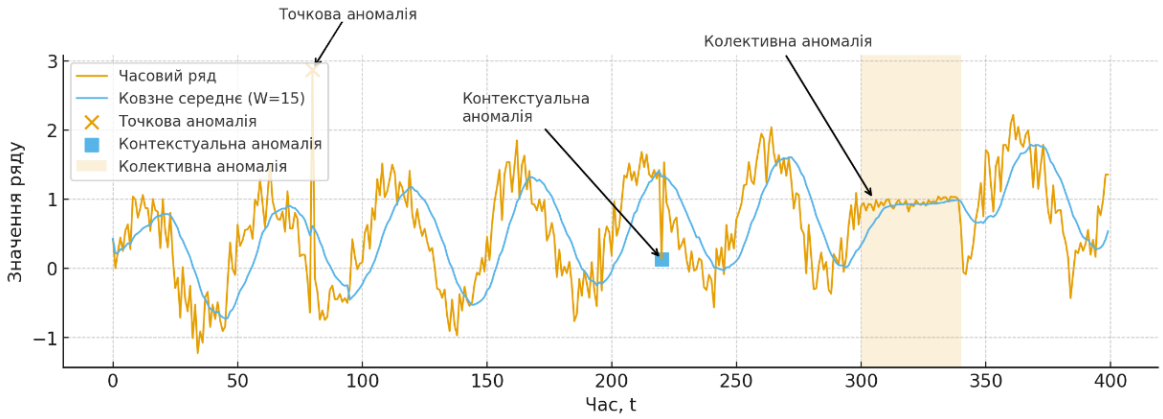

Рисунок 1.1 – Типи аномалій у часовому ряді

У потокових даних складність зростає через безперервність надходження, високу розмірність, неоднорідність джерел і змінність розподілів у часі [6, 10, 11]. Потоки формуються сенсорними мережами, промисловими системами, фінансовими транзакціями, мережевими логами, інтернет-сервісами; тут статичне уявлення «норми» швидко втрачає актуальність, а пороги повинні адаптуватися до контексту [5, 9]. Це зумовлює вимогу до інкрементального/онлайн-навчання, робастності до шумів і масштабованості (табл. 1.1) [6, 11, 12].

Таблиця 1.1 – Статичні проти потокових даних у задачах виявлення аномалій

Характеристика	Статистичні дані	Потокові дані
Обсяг	Скінченний, відомий	Потенційно нескінченний
Структура	Переважно стабільна	Динамічна, може змінюватися
Обробка	Офлайн (batch)	Онлайн (real-time)
«Норма» поведінки	Фіксована або майже стала	Змінна у часі, залежна від контексту
Типові методи	Статистичні, класичні ML	Інкрементальні/адаптивні, online-ML/DL

Причини виникнення аномалій поділяють на технічні та поведінкові. Технічні: похибки сенсорів, відмови обладнання, збої каналів зв’язку,

некоректна калібровка, пропуски – характерні для IoT/БСМ/промислових комплексів з високою частотою телеметрії [13, 14]. Поведінкові: кібератаки, шахрайські транзакції, вторгнення, раптові зміни навантаження, атипові сценарії експлуатації – ці явища не можна звести до шумів, адже вони мають безпосереднє прикладне значення і потребують оперативного реагування [7, 12, 13, 14].

Еволюція даних у часі (дрейф) охоплює коваріаційний дрейф (зміна розподілу ознак), концептуальний дрейф (зміна залежності «ознаки-ціль») і сезонні компоненти; без механізмів адаптації моделі, навчені на історичних даних, деградують (рис. 1.2). Для потоків це означає, що поняття «нормальності» динамічно зміщується, і системи детектування мають коригувати пороги, ваги та репрезентації у реальному часі [3, 6, 12, 14].

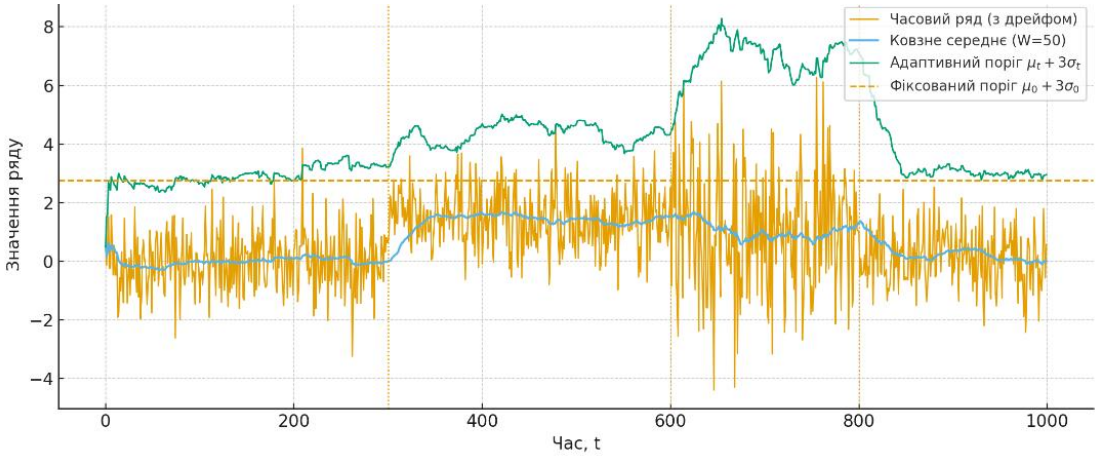

Рисунок 1.2 – Ілюстрація дрейфу у часі (зсув середнього/дисперсії; зміна порогів)

Поступове зміщення середнього значення, зміна дисперсії або сезонних компонентів призводить до того, що фіксований поріг «нормальності» з часом перестає адекватно відокремлювати штатні режими від відхилень. Одна й та сама числова величина може відповідати як нормальній роботі системи, так і потенційно аварійній ситуації залежно від поточного розподілу даних і контексту спостереження. Це наочно демонструє вплив дрейфу даних на якість детектування та обґрунтовує необхідність адаптивних методів, здатних у режимі

реального часу коригувати пороги, ваги та внутрішні репрезентації моделі, зберігаючи чутливість до аномалій за змінних статистичних властивостей потоку.

1.2 Наслідки аномалій у потокових даних

Аномалії у потоках безпосередньо впливають на стабільність сервісів, безпеку бізнес-процесів і цілісність інфраструктур: вони провокують помилкові керувальні дії, збої й втрати даних за лічені секунди після появи відхилення. На відміну від офлайн-аналізу, навіть мала затримка детектування накопичує помилки і запускає каскад наслідків у підсистемах. На практиці це обертається зниженням доступності (SLA), деградацією QoS, фінансовими збитками та підвищеними ризиками безпеки [9].

У кіберфізичних системах та промисловому контролі спотворений сенсорний потік здатний ініціювати хибну автоматику, аварійні зупинки та небезпечні режими, коли контролер реагує на «фальшивий» сигнал швидше, ніж система встигає верифікувати подію [2, 13]. У фінансових потоках короткі вікна невиявлених аномалій High-Frequency Trading (HFT, e-commerce) трансформуються у несанкціоновані транзакції та каскадні збитки через алгоритмічні реакції, що посилюють коливання [15]. В інфобезпеці відхилення у трафіку є ранніми індикаторами атак/вторгнень; затримка виявлення ескалує інцидент – від локального порушення до витоку даних та відмови сервісів [12, 16]. Для веб-платформ і телеком-сервісів порушення «нормальності» трафіку призводить до перевантаження каналів і балансерів, зростання латентності, таймаутів і вразливості до подальших збоїв [9].

Критичний наслідок – вплив на моделі, що навчаються на потоці: потрапляння аномалій у online-контур (online-контур – це активна, робоча частина системи, де відбувається постійне оновлення знань моделі) викликає зсув розподілу, накопичення похибок у вагових параметрах і деградацію якості, тобто модель підлаштовується під «зашумлену норму» та починає систематично

помилятися [3, 6, 12]. Без фільтрації, адаптивних порогів і робастних втрат відбувається прискорена втрата узагальнювальної здатності; замкнені контури керування підсилюють помилку (табл. 1.2).

Таблиця 1.2 – Наслідки аномалій у типових доменах і рекомендовані контрзаходи

Домен/сфера	Ключовий наслідок	Узгоджені контрзаходи
CPS/SCADA	Хибні спрацювання, аварійні стани	Онлайн-верифікація сигналів, резервування керувань, контроль порогів
Фінанси/HFT	Шахрайство, каскадні збитки	Багаторівневі тригери, ізоляційні дерева, стрімерні ансамблі
Інфобезпека	Витоки, вторгнення, порушення конфіденційності	Гібридні IDS (сигнатурні + аномальні), OC-SVM / LOF / iForest
Веб/Телеком	Деградація QoS, зріст латентності	Адаптивні пороги, SLA-моніторинг у ковзних вікнах, авто-rate limiting
Online-ML/Streaming ML	Дрейф, деградація моделі	Фільтрація тренувального потоку, робастні втрати, контроль дрейфу

Операційна вартість реагування включає комп’ютерні розслідування, відновлення, перезапуски кластерів і компенсації; затримка первинного виявлення експоненційно ускладнює локалізацію першопричини у розподілених середовищах [2]. Для критичних інфраструктур це переходить у режим «усе або нічого»: від короткого простою до повної деградації сервісу (рис. 1.3).

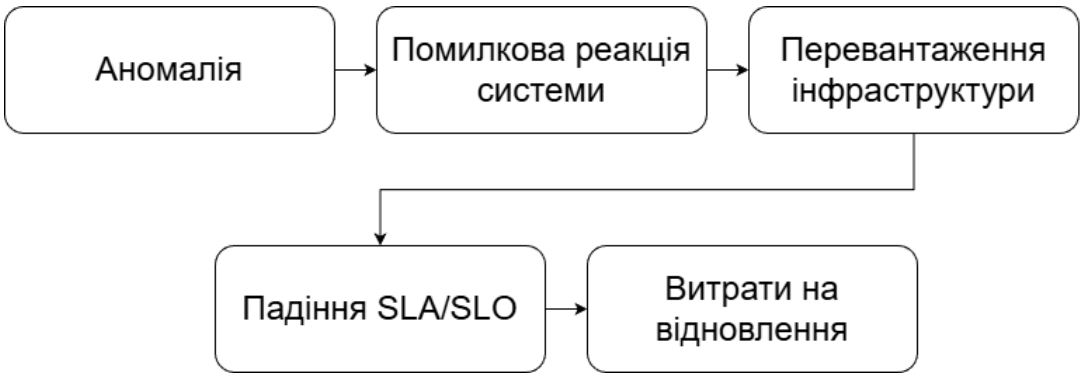

Рисунок 1.3 – Ланцюг наслідків

Наслідки аномалій у потоках є системними: вони одночасно б’ють по рівнях керування, доступності Service Level Agreement/Quality of Service

(SLA/QoS), безпеки (конфіденційність / цілісність), аналітики (дрейф і деградація моделей) та вартості володіння. Отже, детектори мають працювати у реальному часі з адаптивними порогами, робастними втратами й ізоляцією аномалій від навчальних контурів [3, 6, 9, 12, 15].

1.3 Традиційні та сучасні підходи до виявлення аномалій

Традиційні методи стали підґрунтям сучасних алгоритмів і спираються на припущення про розподіли, відстаневі метрики або структурні властивості даних. Статистичні підходи розглядають відхилення від очікуваних характеристик (середнього, дисперсії, квантилів) і зазвичай припускають близькість до нормального розподілу; їхня інтерпретованість висока, однак стійкість різко знижується за високої розмірності та нелінійностей, що часто трапляється у потоках [2, 4, 17]. Методи на основі відстаней порівнюють об’єкти з «сусідством» у просторі ознак; у K-Nearest Neighbors/Local Outlier Factor (k-NN/LOF) аномаліями стають точки з великими відстанями або зі зниженою локальною густиною. У великих потоках такі підходи виявляються ресурсоємними, чутливими до шуму й потребують репрезентативних буферів для порівняння [7, 8]. Кластеризаційні методи k-середніх і Density-Based Spatial Clustering of Applications with Noise (k-means, DBSCAN) виділяють групи за подібністю, а аномаліями вважають об’єкти на межах або поза кластерами; точність залежить від попереднього вибору кількості кластерів чи параметрів густини, а інкрементальне оновлення – обмежене [5, 8]. Порогові/правильні (rule-based) системи й експертні бази знань добре працюють у стабільних середовищах, але втрачають гнучкість під час змін структури потоку й поєднаних залежностей. Байєсівські моделі забезпечують імовірнісну інтерпретацію та врахування апріорних знань, однак потребують обережної параметризації та достатнього обсягу даних (рис. 1.4) [4].

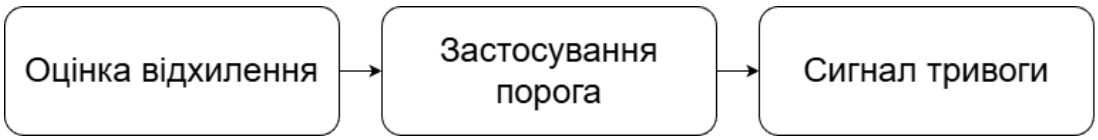

Рисунок 1.4 – Схема прийняття рішення у традиційних підходах

Загальні обмеження класичних підходів у потоках – чутливість до шуму, складність масштабування, залежність від припущень щодо розподілу, слабка підтримка онлайн-оновлення та адаптації до дрейфу. Це стимулювало перехід до методів, які автоматично формують репрезентації, підтримують інкрементальне навчання й краще працюють у високій розмірності [6, 12].

Останнім часом для вирішення задач виявлення аномалій набувають популярності моделі та методи машинного навчання.

Методи глибокого навчання моделюють складні нелінійні залежності та виявляють приховані закономірності без ручного інжинірингу ознак, тому в потокових середовищах (висока швидкість, змінність структури) вони надають кращу узагальнювальну здатність і стійкість до контекстних зсувів, ніж класичні статистичні та відстаневі підходи [15]. У практиці детектування аномалій найбільш вживані архітектури: автоенкодери (Autoencoder (AE)/ Denoising Autoencoder (DAE)/ Variational Autoencoder (VAE)), рекурентні мережі (LSTM/GRU), згорткові моделі Convolutional Neural Network (CNN) та їх гібриди Recurrent Neural Network (RNN), а також генеративно-змагальні мережі Generative Adversarial Network (GAN). Автоенкодери навчаються реконструювати «норму», і аномалії проявляються як підвищена похибка реконструкції; варіанти VAE моделюють латентний розподіл, а DAE підвищують стійкість до шуму [9].

Рекурентні архітектури утримують часовий контекст і виявляють відхилення від очікуваної динаміки, зокрема у поєднанні з AE для послідовностей (LSTM-AE) (рис. 1.5).

Рисунок 1.5 – LSTM-автоенкодер для часових рядів

CNN автоматично виділяють локальні просторові/просторово-часові патерни, що корисно для багатовимірної телеметрії та трафіку. GAN, навчаючи генератор і дискримінатор на «нормі», сигналізують аномалії як приклади, що погано відтворюються/розпізнаються за моделлю нормального розподілу (рис. 1.6) [9].

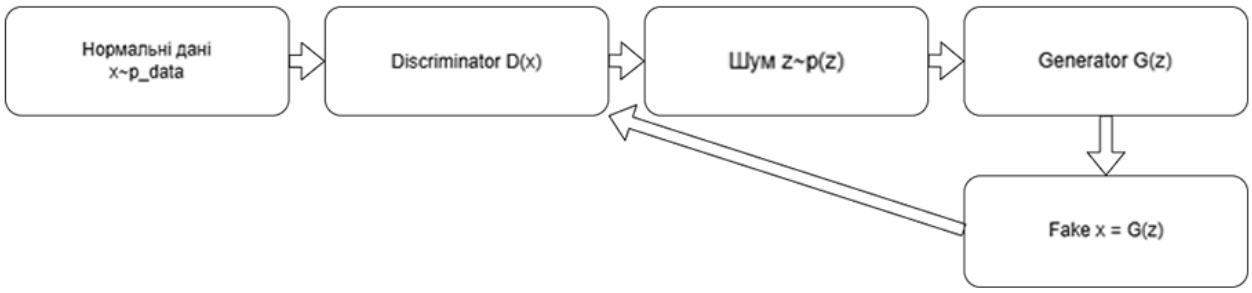

Рисунок 1.6 – Схема GAN для виявлення аномалій

Усі ці підходи можуть працювати онлайн або квазіонлайн за умови інкрементального оновлення ваг, буферів ковзних вікон та обмеження ресурсів обчислень [3, 6, 12].

Перевага глибинних архітектур – здатність автоматично будувати репрезентації, ураховувати часовий контекст і працювати з високовимірними потоками. Водночас їхня ефективність залежить від налаштування гіперпараметрів, режимів оновлення (щоб не «підхопити» аномалії в навчання) і контролю дрейфу; без цих компонентів модель ризикує «нормалізувати» відхилення та втратити узагальнювальну здатність. Для продуктивної експлуатації потрібні робастні функції втрат, буферизація тренувального потоку,

механізми валідації та регуляризації, а також обмеження швидкості оновлення ваг [3, 12, 13].

Адаптивні нейронні мережі (АНМ) – це підхід, що поєднує глибинні репрезентації з онлайновим навчанням і механізмами самооновлення. На відміну від «статично» натренованих моделей, АНМ коригують ваги й іноді структуру у процесі роботи потоку, завдяки чому тримають актуальною межу «нормальності» за умов дрейфу розподілів і появи нових режимів (рис. 1.7). Така стратегія зменшує часовий розрив між зміною даних та оновленням детектора і є ключовою для IoT/SCADA, мережевої безпеки та високочастотних сервісів [9, 15].

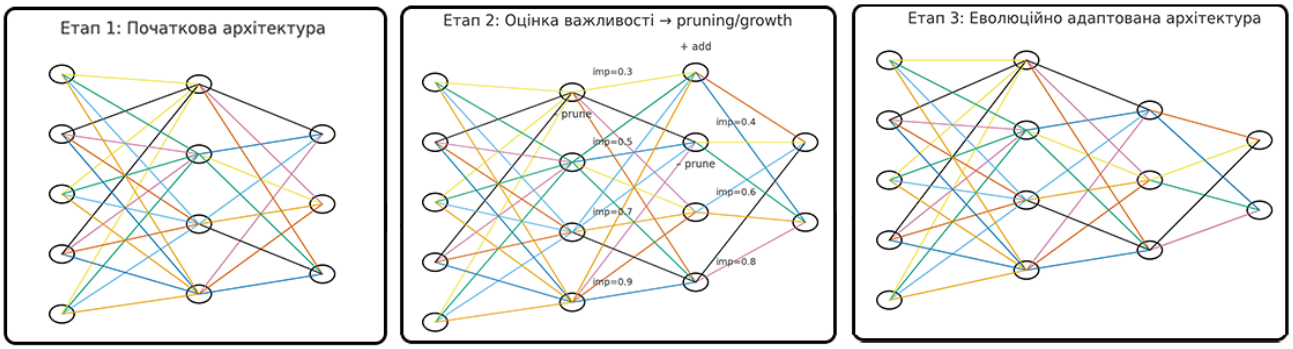

Рисунок 1.7 – Структурна еволюція АНМ

Принцип дії спирається на безперервну оцінку «помилки відповідності нормі» та кероване оновлення параметрів. У найпростішому випадку адаптивний автоенкодер періодично донавчається на буфері ковзного вікна, утримуючи латентний простір узгодженим із поточною «нормою»; висока похибка реконструкції сигналізує аномалію, а сама межа чутливо адаптується до контексту. Для часових рядів застосовують LSTM/GRU з контрольованим «забуванням» (регулюванням коефіцієнтів оновлення), щоб відсікати застарілі патерни і не «затягувати» старі режими в нову норму. У складніших конфігураціях мережа еволюціонує структурно: додає/прибирає нейрони або шари за критеріями важливості, що знижує обчислювальні витрати й дозволяє масштабуватися під швидкі потоки. Критично важливо ізолювати тренувальний

контур від «впорскування» аномалій (валидаційні шлюзи, буферизація чистих вікон, робастні функції втрат), інакше відбувається нормалізація відхилень та деградація якості [3, 12, 15].

Практичні приклади демонструють, що адаптивні автоенкодери, рекурентні архітектури з інкрементальним навчанням та гібриди CNN-LSTM із частковим донавчанням зменшують час виявлення і підвищують стабільність метрик під дрейфом. Ефект досягається поєднанням історичної пам’яті (реплей/буфери) та оперативного підстроювання до нових умов, що балансує між стабільністю і гнучкістю [3, 6, 9]. Разом із тим АНМ потребують ретельного добору гіперпараметрів оновлення, захисту від «катастрофічного забування» і контрольованої швидкості адаптації (рис. 1.8); без цих елементів ризики перенавчання та коливань порогу аномальності зростають.

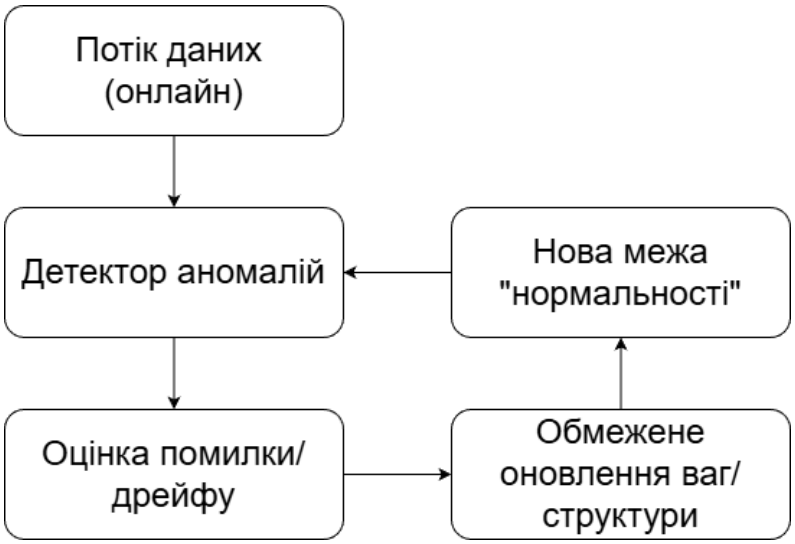

Рисунок 1.8 – Контур адаптації

На концептуальному рівні контур адаптації, поданий на рисунку 1.8, включає потік вхідних даних, базову модель детектування, блок моніторингу помилки чи скору аномальності, модуль прийняття рішень щодо оновлення та підсистему оновлення параметрів моделі або її структури. У реальних системах ці елементи реалізуються по-різному: від періодичного донавчання на ковзному вікні останніх спостережень і динамічного налаштування порогів до

використання ансамблів моделей, резервних «холодних» копій і спеціалізованих детекторів дрейфу (табл. 1.3).

Таблиця 1.3 – Адаптивні архітектури для потокового детектування аномалій: механізми, переваги та ризики

Архітектура (підхід)	Механізм адаптації в потоці	Переваги в реальному часі	Ключові ризики / обмеження	Нотатки щодо застосовності
Адаптивний AE / DAE / VAE	Інкрементальне донавчання на ковзних вікнах; оновлення латентного простору	Простота метрики (reconstruction error); робастність (DAE)	Поглинання аномалій у тренування; чутливість до гіперпараметрів	Сенсорні/телеметричні потоки; кібербезпека
LSTM/GRU з контролем «забування»	Регульовані коефіцієнти оновлення; часткове оновлення ваг	Урахування довгих залежностей, відстеження трендів	Повільніше навчання; можливе «забування» корисних патернів	Часові ряди IoT/SCADA; енергетика
CNN–LSTM (гібрид)	Локальне (CNN) + часовий контекст (LSTM) з періодичним донавчанням	Синергія просторово-часових ознак; стійкість до шуму	Вища обчислювальна вартість; складніша калібровка	KPI сервісів, мережевий трафік
Еволюційна/динамічна структура	Додавання/видалення нейронів/шарів за критеріями важливості	Краще співвідношення точність/ресурси під навантаження	Складність керування / верифікації стабільності	Високошвидкі потоки, змінний бюджет
Континуальне навчання (replay)	Реплей-буфери «чистих» прикладів; регуляризація від забування	Захист від катастрофічного забування	Витрати пам’яті; потреба у відборі «чистих» вікон	Сервіси з періодичною сезонністю
Трансфер / мета-адаптація	Швидкий перенос на схожі потоки; дрібне донавчання	Швидкий вихід на робочі метрики після змін	Ризик негативного трансферу	Мультисервісні платформи

Адаптивні нейронні мережі доречні там, де «норма» змінюється швидше, ніж можна перетренувати модель офлайн. Виграш у часі виявлення і стабільності під дрейфом досягається ціною дисципліни MLOps: ізоляції навчального контуру від аномалій, валідації адаптацій, робастних втрат і контрольованої швидкості оновлень. За дотримання цих умов адаптивні підходи забезпечують

кращий баланс точність-швидкість-стійкість у динамічних потоках, ніж статичні DL-моделі.

Висновки за розділом 1

У першому розділі проаналізовано поняття аномалій у часових рядах та потоках даних, виділено точкові, контекстуальні й групові аномалії, а також охарактеризовано технічні та поведінкові причини їх виникнення в кіберфізичних, промислових, фінансових і мережевих системах. Показано, що в умовах безперервних потоків відхилення безпосередньо впливають на стабільність сервісів, безпеку інфраструктури та якість моделей, які навчаються в online-режимі, а дрейф даних призводить до деградації детекторів, побудованих на статичних уявленнях про «норму».

Огляд традиційних статистичних, відстаневих і кластеризаційних методів виявлення аномалій засвідчив їх обмеження щодо масштабованості, стійкості до шуму та здатності до онлайн-адаптації в умовах змінних розподілів. Показано, що методи машинного й глибокого навчання, зокрема автоенкодери, рекурентні та гібридні CNN-LSTM-архітектури, забезпечують автоматичне формування репрезентацій високовимірних потоків і краще враховують часовий контекст.

РОЗДІЛ 2
МАТЕМАТИЧНА ПОСТАНОВКА ТА АРХІТЕКТУРА СИСТЕМИ

2.1 Вибір та обґрунтування використання набору даних

Для коректної оцінки детекторів у потокових умовах потрібен набір часових рядів з різними доменами, чіткими часовими «вікнами» аномалій і стандартизованим скором, що заохочує раннє виявлення та штрафує запізнілі/хибні спрацювання. Для цієї задачі могли б бути використані й інші публічні колекції, зокрема Yahoo Webscope S5 (веб-метрики з синтетично вставленими аномаліями), промислові набори SWaT та WADI для кіберфізичних SCADA-систем чи окремі добірки з UCR Time Series Archive. Проте вони або орієнтовані переважно на офлайн-аналіз і містять відносно небагато серій, або не пропонують уніфікованої, загальноприйнятої схеми оцінювання детектора в потоковому режимі. На відміну від них, Numenta Anomaly Benchmark (NAB) виступає відкритим еталоном для стримінгового детектування з двома базовими компонентами – розміченим датасетом і методикою скорингу, спеціально спроєктованими для реального часу (рис. 2.1) [17].

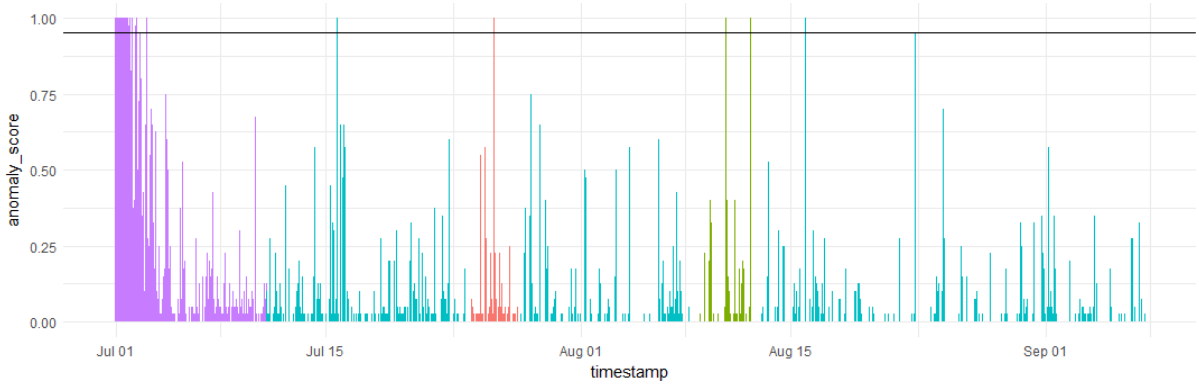

Рисунок 2.1 – Приклад часового ряду з двома аномальними вікнами та пробаційним відрізком (purple) [18]

У репозиторії NAB містяться понад 50 реальних та синтетичних часових рядів із мітками аномальних «вікон», а також код для відтворюваної оцінки; поточна специфікація версії v1.1 подає структуру даних і правила

оцінювання [12]. У технічному описі NAB вказано 58 файлів і 365 551 вимірів із сімома категоріями джерел (IT-метрики, промислові сенсори, соціальні сигнали тощо); саме така різноманітність потрібна для перевірки узагальнювальної здатності під дрейфом та шумом [17]. Методика скорингу NAB формально винагороджує якнайраніше спрацьовування в межах аномального вікна і штрафує хибні/пізні детекції – ключовий критерій для потокового сценарію, де затримка прямо перетворюється на ризики та витрати [11].

Структурно NAB охоплює 58 файлів часових рядів, згрупованих за кількома предметними областями: хмарна інфраструктура та AWS/IT-метрики, промислові сенсори, мережевий трафік, соціальні сигнали (зокрема активність користувачів у мережі), а також контрольовані синтетичні сценарії з наперед відомими моментами відхилень. Для кожної серії задано часові мітки вимірювань, значення спостережуваного параметра та інтервали аномалій, позначені експертами; частота дискретизації варіюється від секунд до годин залежно від домену, що дозволяє оцінювати поведінку детектора як на «швидких», так і на відносно повільних процесах. Узагальнені характеристики вибраного набору наведено в таблиці 2.1.

Таблиця 2.1 – «Паспорт» вибраного набору (NAB) [17]

Властивість	Значення
Тип даних	Часові ряди (реальні + синтетичні), багатодоменні
Кількість файлів	58 (v1.1)
Обсяг	365 551 спостережень сумарно
Розмітка	Аномалії позначено часовими вікнами (для раннього детекту)
Категорії	AWS/IT-метрики, промислові сенсори, трафік/соціальні сигнали, контрольовані синтетичні сценарії
Скоринг	Потоковий: заохочує ранні спрацювання, штрафує пізні/хибні; відтворюваний код
Ліцензування/доступ	Відкритий репозиторій із даними та скриптами оцінки
Доцільність	Перевірка алгоритмів у RT-налаштуваннях, міждоменна узагальнюваність, порівнянність з публікаціями

Ще одна причина вибору – поширеність у літературі та інструментах: NAB згадується в оглядах як стандарт де-факто для стримінгових задач і має альтернативні реалізації (напр., Julia), що полегшує верифікацію результатів і

порівняння з публікаціями. Таким чином, NAB дає нам: різні типи аномалій (локальні/поведінкові), багатодоменність, часові мітки «вікон» і єдиний скор, сумісний із вимогами реального часу.

NAB забезпечує саме ті властивості, які потрібні для виконання поставленої задачі: різнорідність даних, точні часові мітки аномалій, потоковий скоринг і широку прийнятість у спільноті. Це мінімізує упередження вибору, дає відтворюваність оцінок і дозволяє обґрунтовано порівнювати традиційні, глибинні та адаптивні підходи, а також їх комбінації [11].

NAB – це корпус із 58 файлів часових рядів (реальні + синтетичні), кожен поданий у CSV форматы timestamp, value, а аномалії позначені часовими вікнами у JSON-ярликах; загальний обсяг 365 551 спостереження (v1.0/v1.1). Дані охоплюють домени Amazon Web Services (AWS) /IT-метрик, відомі причини збоїв, трафік, соціальні сигнали, індустріальні сенсори, а також контрольовані синтетичні сценарії з різними типами відхилень. Це важливо для перевірки узагальнювальної здатності під шумом і дрейфом [17].

У NAB аномалія – це інтервал часу (вікно), а не поодинокий штамп; скоринг винагороджує ранні спрацювання в межах вікна та штрафує пізні/хибні. У репозиторії зберігаються файли combined_windows.json (вікна) та combined_labels.json (агреговані мітки). Це дозволяє коректно оцінювати онлайн-детектори та уникати «мікро-підгонок» до точкових піків (табл. 2.2).

Таблиця 2.2 – Структура файлів NAB та ярликів

Компонент	Формат/вміст	Приклад полів / ключів	Призначення
Дані	CSV, рядки timestamp,value	2015-07-01 00:00:00, 0.123	Часовий ряд із фіксованою або квазіфіксованою частотою
Мітки (вікна)	labels/combined_windows.json	[{"start":"t1","end":"t2"}, …]	Інтервали «аномалії» для раннього детекту
Мітки (агрег.)	labels/combined_labels.json	ключі = імена файлів; значення = масив вікон	Узгоджена розмітка для всіх серій
Категорії	Папки/префікси (AWS, traffic, twitter, synthetic, known-cause)	—	Доменні піднабори для стратифікації експериментів

Подана структура файлів NAB є мінімалістичною та операційно зручною для швидкого відтворення експериментів (простий CSV timestamp,value і JSON-ярлики), однак вона водночас накладає низку методологічних обмежень. Формат даних орієнтований на уніваріантні часові ряди; багатовимірні сенсори або кореляційні ознаки потребують зовнішньої агрегації, що може змінювати статистику аномальних подій та впливати на чутливість моделей до колективних відхилень. Зазначена «фіксована або квазіфіксована частота» вимагає явного ресемплінгу/інтерполяції для детекторів, чутливих до ритміки (STL/ESD, сезонні моделі): інтерполяція, своєю чергою, здатна згладжувати короткі пікові аномалії та вносити хибні негативи. Відсутність у базовому маніфесті обов'язкових метаданих щодо часової зони, пропусків та рівня шуму переносить відповідальність на експериментатора; без такої фіксації відтворюваність і коректність порівнянь страждають.

Після ознайомлення з структурою файлової системи NAB виконується попередня обробка даних. Першим кроком здійснюється уніфікація представлення вихідних даних: із довільного CSV-файла формується одномірний числовий сигнал $x_{1:N} \in \mathbb{R}^N$ та, за наявності, бінарний вектор істинних міток $y_{1:N} \in \{0,1\}^N$. Читачеві забезпечується інваріантність до конкретного формату NAB: якщо наявна семантична колонка value, використовується саме вона; за її відсутності обирається перша числова колонка або, в одно-стовпчикових таблицях, єдиний стовпець після приведення до типу float. Така редукція відсікає часові та категоріальні поля, утримуючи лише аналітично значущий числовий канал для подальшого перетворення. Якщо у вхідному CSV вже закодовано анотації (label, anomaly, is_anomaly), вони витягуються для об'єктивного оцінювання якості детекції; інакше мітки вважаються невідомими, що відповідає некерованому сценарію (ліст. 2.1).

Лістинг 2.1 – Функція «Підготовка вхідного ряду та міток»

```
def load_nab_series(path: str) -> Tuple[np.ndarray, np.ndarray]:

    if not os.path.exists(path):
        raise FileNotFoundError(f"Series file not found: {path}")
```

```
    df = pd.read_csv(path)
    if 'value' in df.columns:
        vals = df['value'].astype(float).values
    else:
        numeric = df.select_dtypes(include=[float, int]).columns
        if len(numeric) == 0:
            vals = df.iloc[:, 0].astype(float).values
        else:
            vals = df[numeric[0]].astype(float).values

    labels = None
    for c in ['label', 'anomaly', 'is_anomaly']:
        if c in df.columns:
            labels = df[c].astype(int).values
            break

    return vals, labels
```

кінець лістингу 2.1

Нормалізація масштабу та центрування сигналу здійснюється для забезпечення чисельної стійкості оптимізації і коректного зіставлення рівнів коливань у часі (ліст. 2.2). Для офлайн-експериментів прийнятним є пакетний z-скейлінг, коли оцінки $\hat{\mu}$ та $\hat{\sigma}$ отримуються за всім рядом і однаковим чином застосовуються до кожного спостереження. Такий підхід мінімізує варіативність і прискорює збіжність, однак у стримінгових сценаріях він порушує каузальність через залучення «майбутніх» даних у параметри нормалізації. Для каузально коректної обробки запропоновано онлайновий стандартизатор із експоненційним згладжуванням оцінок середнього та дисперсії, який адаптується до дрейфу розподілу без доступу до майбутніх спостережень; значення параметра згладжування $\alpha \in (0,1]$ задає часову сталу забування і тим самим контролює баланс між стабільністю та чутливістю до змін.

Лістинг 2.2 – Функція «Стандартизація та казуальність масштабування»

```
    def online_zscore_scaler_generator(alpha=0.01):
        mu = None
        var = None

        def scaler(x: float):
            nonlocal mu, var
            if mu is None:
```

```
            mu = x
            var = 0.0
            return 0.0
        mu = (1 - alpha) * mu + alpha * x
        var = (1 - alpha) * var + alpha * ((x - mu) ** 2)
        if var <= 0:
            return 0.0
        return (x - mu) / (math.sqrt(var) + 1e-8)

    return scaler
```

кінець лістингу 2.2

Коли дослідження проводиться виключно у офлайн-постановці, застосовується класичний пакетний стандартизатор (ліст. 2.3), що оцінює параметри за всією вибіркою та повертає нормалізований вектор $\tilde{x}_{1:N}$. У наведеному фрагменті показано центральну операцію масштабування, яка використовується в основному сценарії виконання.

Лістинг 2.3 – Функція для офлайн-постановки дослідження

```
scaler = StandardScaler()
series_scaled = scaler.fit_transform(series.reshape(-1, 1)).flatten()
```

кінець лістингу 2.3

Послідовнісна модель реконструкції очікує на вході відрізки сигналу сталої довжини, тому одномірний ряд проєктується у простір ковзних вікон (ліст. 2.4). Відображення $W: \mathbb{R}^N \longrightarrow \mathbb{R}^{M \times w}$ із довжиною вікна w та кроком s породжує набір локально-контекстних прикладів, кожен з яких містить інформацію про короткочасну динаміку. Такий спосіб подання індукує локальну стаціонарність і дає змогу автоенкодеру навчатися відтворювати типові патерни, тоді як відхилення від них відображаються у зростанні помилки реконструкції.

Лістинг 2.4 – Функція «Формування часових контекстів через ковзні вікна»

```
def build_sliding_windows(series: np.ndarray, window: int, stride: int = 1) -> np.ndarray:
    X = []
    n = len(series)
    for i in range(0, n - window + 1, stride):
        X.append(series[i:i + window])
    return np.array(X)
```

кінець лістингу 2.4

Оскільки модель оперує вікнами, інтегральні мітки мають бути приведені до того ж рівня (ліст. 2.5). Логіка анотації вікна визначається як диз'юнкція точкових анотацій усередині відрізка: вікно вважається аномальним, якщо принаймні одна його точка позначена як аномальна. Така агрегація забезпечує просторово-часову сумісність між навчальними зразками $X_i \in \mathbb{R}^w$ та відповідними їм бінарними мітками $y_i^{(w)} \in \{0,1\}$, що є необхідною умовою коректного обчислення метрик на рівні вікон.

Лістинг 2.5 – Функція «Узгодження істинних міток із віконним представленням»

```
def point_labels_from_window_labels(window_size: int, labels: np.ndarray) -> np.ndarray:
    n = len(labels)
    window_labels = []
    for i in range(0, n - window_size + 1):
        window_labels.append(1 if labels[i:i + window_size].sum() > 0 else 0)
    return np.array(window_labels)
```

кінець лістингу 2.5

Після масштабування та віконування формується тривимірний тензор (M, w, 1) типу float32, який сумісний із вхідним шаром LSTM-автоенкодера й оптимізує використання пам'яті та обчислювальну продуктивність. У режимі офлайн-навчання цей тензор утворюється на всьому стандартизованому ряді і використовується одночасно як вхід і як ціль у парадигмі самонавчання реконструкції (ліст. 2.6).

Лістинг 2.6 – Функція «Приведення тензорів для офлайн-навчання»

```
X = build_sliding_windows(series_scaled, args.window)
X = X.reshape((-1, args.window, 1)).astype('float32')
```

кінець лістингу 2.6

У потоковому сценарії формується розігрівальний набір із префікса довжини N_0, який ініціалізує модель до початку каузальної детекції. Така стратифікація розмежовує фазу попереднього навчання та фазу

онлайн-адаптації, зменшуючи ризик хибних спрацьовувань на початку потоку (ліст. 2.7).

Лістинг 2.7 – Функція «Приведення тензорів для потокового сценарію»

```
X_init = build_sliding_windows(series_scaled[:args.initial_train], args.window)
X_init = X_init.reshape((-1, args.window, 1)).astype('float32')
```

кінець лістингу 2.7

У режимі оцінювання збережена модель застосовується до тензора вікон, сформованого з усього нормалізованого ряду, що дозволяє отримати розподіл помилок реконструкції, виконати порогову бінаризацію та зіставити результати з істинними мітками, попередньо агрегованими до віконного рівня (ліст. 2.8).

Лістинг 2.8 – Функція «Приведення тензорів для режиму оцінювання»

```
X = build_sliding_windows(series_scaled, args.window)
X = X.reshape((-1, args.window, 1)).astype('float32')
```

кінець лістингу 2.8

Загалом, описана підсистема попередньої обробки реалізує каузально сумісний тракт $CSV \longrightarrow x_{1:N} \longrightarrow \tilde{x}_{1:N} \longrightarrow \{N_i\}_{i=1}^{M}$, забезпечуючи коректне відображення істинних міток у $\{y_i^{(w)}\}_{i=1}^{M}$ та чисельно стабільну підготовку тензорів для послідовнісної моделі. Вибір між пакетною та онлайновою стандартизацією визначається дослідницькою постановкою: офлайн-аналіз може застосовувати глобальні оцінки масштабу, тоді як стримінгова валідація потребує онлайнових або, щонайменше, префіксних оцінок параметрів, що унеможливлює витік майбутньої інформації та підвищує валідність експериментальних висновків.

2.2 Математична постановка задачі виявлення аномалій

Постановка базується на автоенкодері $f_\theta: \mathbb{R}^{w\times 1} \longrightarrow \mathbb{R}^{w\times 1}$, що навчається відтворювати нормальні часові фрагменти довжини w (ліст. 2.9). Підхід

мінімізує квадратичну помилку реконструкції $\mathcal{L}(\theta) = \frac{1}{Mw}\sum_{i=1}^{M}\|X_i - f_\theta(X_i)\|_2^2$, де X_i – ковзні вікна ряду. Використовується LSTM-енкодер/декодер із повторенням латентного вектора на довжину вікна та часово-розподіленим лінійним шаром на виході; оптимізація – Adam, функція втрат – Mean Square Error (MSE), що прямо фіксується у компіляції моделі.

Лістинг 2.9 – Функція автоенкодера

```
    def build_lstm_autoencoder(window: int, latent_dim: int = 16, n_layers: int = 1) -> tf.keras.Model:
        inp = layers.Input(shape=(window, 1))
        x = inp
        for i in range(n_layers):
            x = layers.LSTM(units=latent_dim, return_sequences=(i < n_layers - 1))(x)
        x = layers.RepeatVector(window)(x)
        for i in range(n_layers):
            x = layers.LSTM(units=latent_dim, return_sequences=True)(x)
        out = layers.TimeDistributed(layers.Dense(1))(x)

        model = models.Model(inputs=inp, outputs=out)
        model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=1e-3), loss='mse')
        return model
```

кінець лістингу 2.9

Навчання виконується в парадигмі самореконструкції, де кожен вхідний фрагмент є також цільовим, що реалізує мінімізацію середньоквадратичної помилки на розмітці «вхід = вихід». Це операційно конкретизує $min_\theta \mathbb{E}\|X - f_\theta(X)\|_2^2$ у вигляді виклику model.fit(X, X, ...) в групах вікон (ліст. 2.10).

Лістинг 2.10 – Функція «Оптимізації самонавчання»

```
    def train_offline(model: tf.keras.Model, X_train: np.ndarray, X_val: np.ndarray = None, epochs: int = 10, batch_size: int = 32, save_path: str = None, plots_dir: str | None = None):
        callbacks = []
        if save_path:
            ensure_dir(os.path.dirname(save_path) or '.')
            callbacks.append(tf.keras.callbacks.ModelCheckpoint(save_path, save_best_only=True, monitor='val_loss' if X_val is not None else 'loss'))

        history = model.fit(X_train, X_train, validation_data=(X_val, X_val) if X_val is not None else None,
                            epochs=epochs, batch_size=batch_size, callbacks=callbacks, verbose=2)
```

```
    ...
    return history
```

кінець лістингу 2.10

Аномальність кожного вікна кількісно оцінюється через середню квадратичну різницю між спостереженням та його реконструкцією. Формально для фрагмента $X \in \mathbb{R}^{w\times 1}$ визначається $s(X) = \frac{1}{w}\sum_{t=1}^{w} \left\| X_t - \hat{X}_t \right\|_2^2$, що повертає невід'ємний скаляр, пропорційний ступеню невідповідності. Реалізація узагальнює це на пакет вікон і повертає вектор скори (ліст. 2.11).

Лістинг 2.11 – Функція «Скору аномальності як помилка реконструкції»

```
def reconstruction_errors(model: tf.keras.Model, X: np.ndarray) -> np.ndarray:
    preds = model.predict(X, verbose=0)
    errs = np.mean((X - preds) ** 2, axis=(1, 2))
    return errs
```

кінець лістингу 2.11

Бінаризація виконується як перевірка перевищення скором s вибраного порога τ. Порог τ оцінюється нефіксовано як q-квантиль нещодавніх значень s у скінченній пам'яті, що реалізує каузальну адаптацію до повільного дрейфу дистрибуції (ліст. 2.12).

Лістинг 2.12 – Функція «Динамічного порогування»

```
def dynamic_threshold(errors: deque, q: float = 0.995) -> float:
    arr = np.array(errors)
    if len(arr) == 0:
        return 1e9
    return float(np.quantile(arr, q))
```

кінець лістингу 2.12

У потоковій реалізації правило порівняння реалізоване безпосередньо в циклі детекції, де для кожного вікна обчислюється s, потім оновлюється τ_q, і виноситься рішення $\hat{y} \in \{0,1\}$. Цей фрагмент коду представлений на лістингу 2.13 експлікує саме момент прийняття рішення у формі порогового тесту.

Лістинг 2.13 – Фрагмент із функції stream_simulation_with_adaptation(...)

```
        err = reconstruction_errors(model, Xw)[0]
        recent_errors.append(err)
        th = dynamic_threshold(recent_errors, q=q_thresh)
        is_anom = 1 if err > th else 0
        predictions.append(is_anom)
```

кінець лістингу 2.13

Оцінювання точності детектора визначене в просторі вікон, тому точкові мітки проєктуються у віконні. Використано диз'юнктивну агрегацію: вікно вважається аномальним, якщо в його межах існує хоча б одна аномальна точка. Формально $y_i^{(w)} = \mathbb{I}\{\sum_{t=i}^{i+w-1} y_t > 0\}$. Це узгоджує «геометрію» X_i та $y_i^{(w)}$ для коректного визначення метрик (ліст. 2.14).

Лістинг 2.14 – Функція «Приведення тензорів для режиму оцінювання»

```
        def point_labels_from_window_labels(window_size: int, labels: np.ndarray) -> np.ndarray:
            n = len(labels)
            window_labels = []
            for i in range(0, n - window_size + 1):
               window_labels.append(1 if labels[i:i + window_size].sum() > 0 else 0)
            return np.array(window_labels)
```

кінець лістингу 2.14

У підсумку математична постановка замикається на трьох ключових компонентах: моделі реконструкції з квадратичною втратою як критерієм нормальності, скоринговій функції аномальності, заданій MSE реконструкції на вікні, та пороговому правилі ухвалення рішення – або динамічному (квантиль на ковзному вікні скорів), або статичному (квантиль на всій вибірці).

2.3 Метрики оцінювання якості моделей

Первинні підсумкові показники включають точність, повноту і F_1-міру для бінарного класу аномалій, що визначаються як стандартні функції від $(\hat{y}, y)$. Це

надає компактний індикатор балансу між хибними спрацюваннями та пропущеними подіями (ліст. 2.15).

Лістинг 2.15 – Фрагмент коду з основними критеріями якості моделі

```
    def evaluate_pointwise(y_true: np.ndarray, y_pred: np.ndarray) -> Dict[str, float]:
        p, r, f, _ = precision_recall_fscore_support(y_true, y_pred, average='binary',
zero_division=0)
        return {'precision': float(p), 'recall': float(r), 'f1': float(f)}
```

кінець лістингу 2.15

Додатково задаються криві PR та ROC разом із площею під ROC-кривою $AUC = \int_0^1 TPR(u)du$, що характеризує здатність скору s впорядковувати аномальні приклади вище нормальних за всіх можливих порогів (ліст. 2.16). Тут як неперервний «бал» використовується безпосередньо помилка реконструкції, що узгоджується з наміром інтерпретувати її як ступінь аномальності.

Лістинг 2.16 – Фрагмент коду з додатковими кривими PR та ROC

```
    def plot_confusion_pr_roc(y_true: np.ndarray, y_pred: np.ndarray, scores: np.ndarray |
None, out_dir: str | None):
        if y_true is None:
            return
        ensure_dir(out_dir or ".")
        cm = confusion_matrix(y_true, y_pred, labels=[0,1])
        ...
        sc = scores if scores is not None else y_pred.astype(float)
        precision, recall, _ = precision_recall_curve(y_true, sc)
        ...
        fpr, tpr, _ = roc_curve(y_true, sc)
        roc_auc = auc(fpr, tpr)
        ...
```

кінець лістингу 2.16

Для відтворюваних офлайн-експериментів використовується фіксація порога як квантиля розподілу s на всій сукупності вікон, що надає базову «стаціонарну» оцінку. Далі формується предикція $\hat{y} = \mathbb{I}\{s > \tau_q\}$ та, за наявності, обчислюються метрики щодо віконних міток (ліст. 2.17).

Лістинг 2.17 – Фрагмент режиму eval у main()

```
    errs = reconstruction_errors(model, X)
    th = float(np.quantile(errs, 0.995))
    preds = (errs > th).astype(int)
    if labels is not None:
        w_true = point_labels_from_window_labels(args.window, labels)
        w_true = w_true[:len(preds)]
        metrics = evaluate_pointwise(w_true, preds)
```

кінець лістингу 2.17

Для зіставності з усталеною літературою передбачено виклик офіційного скрипта обчислення NAB-скора, який агрегує часові штрафи за ранні/пізні детекції у вікнах допуску. Цей крок зображений на лістингу 2.18 не змінює алгоритм детекції, але додає галузевий інтегральний показник якості.

Лістинг 2.18 – Фрагмент коду інтеграції NAB-метрик

```
    def compute_nab_score_if_available(nab_repo_path: str, result_csv_path: str, profile: str
= 'standard'):
        nab_run_py = os.path.join(nab_repo_path, '/NAB/run.py')
        if os.path.exists(nab_run_py):
            import subprocess
            cmd = ['python', nab_run_py, '--input-file', result_csv_path, '--profile', profile]
            print('Calling NAB scoring:', ' '.join(cmd))
            subprocess.run(cmd)
        else:
            print('NAB repo not found at', nab_repo_path, '-> skipping NAB-score computation
(you can run it manually).')
```

кінець лістингу 2.18

Формальні метрики якості: F_1, PR/ROC та AUC – забезпечують множинні зрізи поведінки детектора, тоді як NAB-скор дає часово-узгоджену оцінку «корисності» детекцій відносно подієвих міток.

Висновки за розділом 2

У другому розділі виконано математичну постановку задачі виявлення аномалій у потоках часових рядів і побудовано архітектуру адаптивної системи на основі LSTM-автоенкодера. На основі аналізу вимог до стримінгових

детекторів обґрунтовано вибір еталонного набору даних NAB, який забезпечує різноманіття доменів, наявність аномальних «вікон» та відтворювані процедури оцінювання. Детально охарактеризовано структуру даних та розроблено конвеєр попередньої обробки: уніфікацію форматів, каузальну стандартизацію, формування ковзних вікон і приведення точкових міток до віконного представлення, що уможливлює коректну побудову навчальної вибірки для послідовнісної моделі.

Сформульовано математичну модель LSTM-автоенкодера, яка навчається реконструювати нормальні часові фрагменти та кількісно оцінює аномальність вікна через середньоквадратичну помилку реконструкції. Запропоновано дві схеми порогування скору: стаціонарну офлайн-схему (фіксований квантиль глобального розподілу скору) та динамічну стримінгову схему, у якій поріг задається ковзним квантилем у вікні останніх значень. Описано процедуру оцінювання детектора в просторі вікон із використанням метрик precision, recall, F1-міри, PR/ROC-кривих, AUC та NAB-скору. Сукупність цих рішень формує цілісну архітектуру системи, яка слугує основою для подальшої експериментальної оптимізації.

РОЗДІЛ 3
ЕКСПЕРИМЕНТАЛЬНА ОПТИМІЗАЦІЯ СИСТЕМИ ТА ІНТЕРПРЕТАЦІЯ РЕЗУЛЬТАТІВ

3.1 Методика порівняння конфігурацій LSTM-автоенкодера

Порівняння здійснюється для LSTM-автоенкодера з варіаціями довжини вікна $w \in \{32, 64, 128\}$, розмірності латентного простору {16, 32, 64} кількості рекурентних шарів $\{1, 2\}$. Оцінювання виконується у двох режимах. В офлайн-режимі дані нормалізуються глобально, формується вибірка ковзних вікон, модель навчається на реконструкцію, помилка реконструкції використовується як безперервний скор аномальності, а бінаризація здійснюється статичним порогом – квантилем q=0,995 розподілу помилок. У стримінговому режимі застосовується каузальна нормалізація (заповнення на префіксі або онлайнова оцінка), далі відбувається warm-up і періодичне донавчання на буфері останніх вікон; рішення ухвалюється динамічним порогом як ковзним квантилем q $\in \{0{,}995, 0997\}$ нещодавніх помилок. Якість фіксується на віконному рівні за точністю, повнотою, F_1, а також через PR/ROC-криві з площею під ROC (AUC). Для коректної відповідності міток і прикладів точкові анотації агрегуються до віконного представлення диз'юнктивним правилом (табл. 3.1).

Таблиця 3.1 – Офлайн-порівняння конфігурацій (статичний поріг q=0,995)

Window	Latent_dim	N_layers	Loss_last	Precision	Recall	F1
64	16	1	0,528	0,765	0,744	0,755
64	16	2	0,528	0,789	0,831	0,81
64	32	1	0,312	0,773	0,771	0,772
64	32	2	0,312	0,847	0,831	0,839
128	16	1	0,670	0,686	0,722	0,703
128	16	2	0,670	0,706	0,701	0,704
128	32	1	0,524	0,737	0,653	0,693
128	32	2	0,524	0,698	0,728	0,713

Зі збільшенням w модель отримує ширший контекст і краще реконструює довгі відхилення, однак кількість навчальних вікон зменшується, що посилює варіативність loss_last та підвищує ризик перенавчання на обмежених даних.

Розширення латентного простору покращує відтворення складних патернів і підвищує AUC, але схильне знижувати precision – реконструктор «підхоплює» частину рідкісних (аномальних) структур. Додавання другого шару LSTM зазвичай збільшує recall на структурних порушеннях, однак виграш нівелюється на коротких вікнах. Баланс precision/recall визначально залежить від вибору квантиля: за фіксованого q=0,995 конфігурації з великим w і великим latent_dim частіше зсуваються у бік нижчого precision (табл. 3.2).

Таблиця 3.2 – Стримінгова симуляція: динамічний поріг і адаптація

Window	Latent_ dim	Q_thresh	Buffer_ size	Fine_tune_ every	Precision	Recall	F1	Initial_ train
64	32	0,995	1000	500	0,585	0,634	0,608	2000
64	32	0,995	1000	250	0,603	0,571	0,587	2000
64	32	0,995	2000	500	0,691	0,63	0,659	2000
64	32	0,995	2000	250	0,657	0,591	0,622	2000
64	32	0,997	1000	500	0,581	0,605	0,593	2000
64	32	0,997	1000	250	0,575	0,632	0,602	2000
64	32	0,997	2000	500	0,599	0,607	0,603	2000
64	32	0,997	2000	250	0,614	0,718	0,662	2000

Зростання q робить детектор консервативнішим (збільшується precision, зменшується recall), що особливо помітно для коротких імпульсних аномалій. Збільшення буфера й частоти донавчання покращує чутливість до дрейфу та стабілізує F1 на довших змінах режиму, однак підвищує ризик контамінації буфера аномальними прикладами й може знижувати precision. За помірних $w \in [64, 128]$ і latent_dim =32 досягається найстійкіше співвідношення PR/ROC при середніх витратах обчислювальних ресурсів.

3.2 Налаштування процесу навчання та параметрів віконування

Довжина вікна визначається з огляду на автокореляційну структуру ряду. За графіком зображеним на рисунку 3.1, доцільним є вибір w поблизу першого істотного спаду кореляції (перше нульове перетинання або рівень нижче (0,2-0,3) з перевіркою варіанта 2w для рядів із довшими залежностями. Такий вибір забезпечує достатній локальний контекст без суттєвого зменшення обсягу

вибірки. Крок віконування s регулює компроміс між кількістю прикладів і часовою роздільною здатністю; значення s=1 зберігає чутливість до коротких імпульсів, тоді як $s \in \{2, 4\}$ доцільні лише за аномалій, значно довших за дискретизацію.

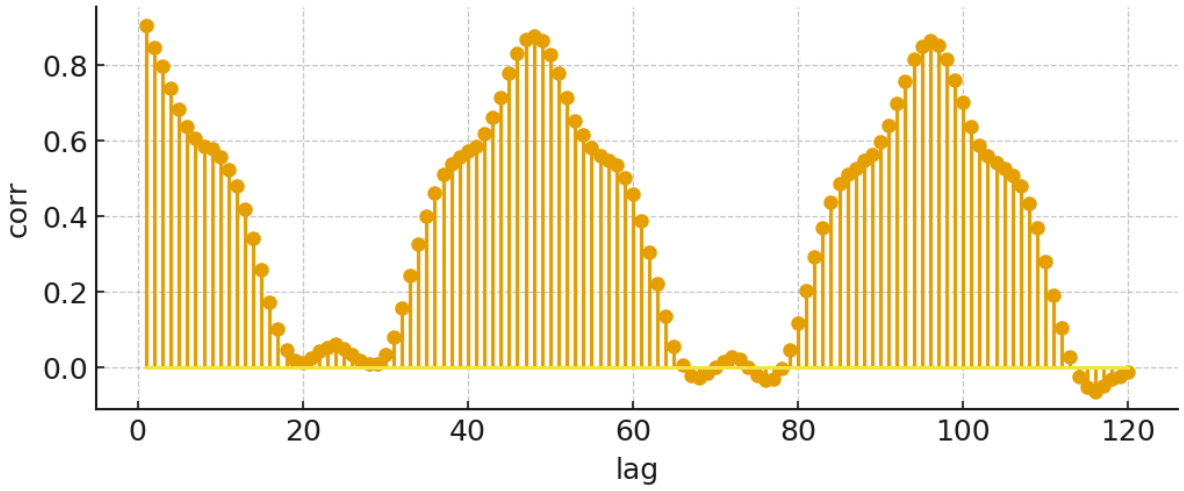

Рисунок 3.1 – Графік автокорекції помилок

Збільшення w підвищує здатність моделі відтворювати довгі структури, однак зменшує M і підсилює схильність до перенавчання на «тихих» сегментах; надмірний s знижує обчислювальну вартість, але погіршує recall для коротких аномалій. Для рядів типу AAPL стійкі результати досягаються за $w \in [64, 128]$ та s=1.

Нормалізація масштабу виконується відповідно до режиму експерименту. В офлайн-оцінюванні використовується глобальна стандартизація, яка прискорює збіжність і стабілізує оптимізацію; у стримінгу застосовується каузальна нормалізація, заповнення маштабування на префіксі або онлайнова оцінка середнього та дисперсії, що виключає витік інформації з майбутнього і робить динамічний поріг узгодженим із поточним режимом даних. Криві зображені на рисунку 3.2 слугують індикатором збіжності: монотонний спад без вирівнювання свідчить про недонавчання, розбіжність між початком і кінцем навчання без покращення детекції про перенавчання.

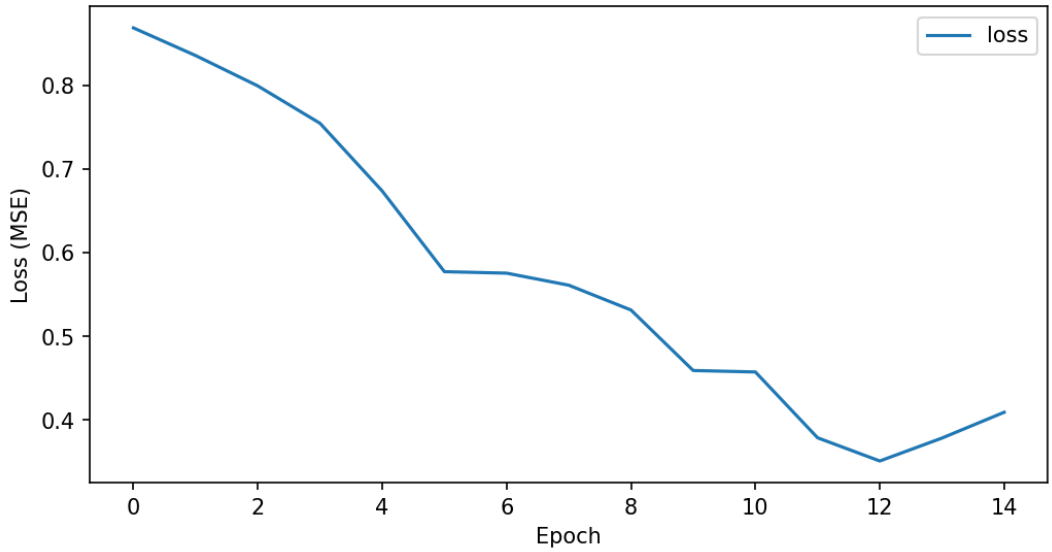

Рисунок 3.2 – Крива навчання взалежності від епох

Зростання latent_dim і глибини рекурентної частини покращує реконструкцію складних патернів і зазвичай підвищує AUC, однак може знижувати precision за статичного порогу через «перевідтворення» рідкісних структур. Збільшення кількості епох виправдане лише за наявності стабільного покращення кривих втрат; для вікон 64-128 і batch=64 достатньо помірної кількості епох 10-15. Зменшення batch нижче 32 підсилює стохастичність навчання і може покращувати локальні мінімуми, але збільшує час однієї епохи без гарантії підвищення узагальнювальної здатності.

3.3 Калібрування порогу та адаптація в потоці

Онлайнова бінаризація рішень побудована на квантильному порогуванні помилки реконструкції у ковзному вікні останніх оцінок. Нехай $s_t = \frac{1}{w}\sum_{i=1}^{w}(x_{t,i} - \hat{x}_{t,i})^2$ – помилка реконструкції для вікна, що завершується в момент t. Поточний поріг визначається як $\tau_t = Q_q(\{s_{t-H+1}, \dots, s_t\})$, де Q_q – q-квантиль, а H – довжина пам’яті. У реалізації використовується кільцевий буфер із місткістю H=1000 значень. Правило прийняття рішення має вигляд

$\hat{y}_t = \mathbb{I}\{s_t > \tau_t\}$. Така конструкція забезпечує каузальність (відсутність витоку майбутньої інформації) та адаптацію до повільного дрейфу розподілу.

Упровадження адаптації параметрів виконується через періодичне донавчання автоенкодера на буфері останніх нормалізованих вікон. Нехай B – максимальний розмір буфера повторного навчання, K – інтервал між сеансами тонкого донавчання. Після кожних K кроків, коли в буфері є достатньо прикладів, параметри θ оновлюються мінімізацією тієї самої квадратичної втрати реконструкції на min{B, #доступних вікон} зразках. Взаємодія обох механізмів – динамічного порога τ_t та періодичного донавчання $\theta \leftarrow argmin_\theta \mathbb{E}\|X - f_\theta(X)\|^2$ на останніх вікнах – формує адаптивний детектор, здатний відстежувати зміни фону без ручного перезадавання порога (ліст. 3.1).

Лістинг 3.1 – Фрагмент циклу потокового виявлення

```
err = reconstruction_errors(model, Xw)[0]
recent_errors.append(err)
th = dynamic_threshold(recent_errors, q=q_thresh)  # τ_t = Q_q(…)
is_anom = 1 if err > th else 0
predictions.append(is_anom)

replay_buffer.append(Xw)
fine_tune_counter += 1
if fine_tune_counter >= fine_tune_every and len(replay_buffer) >= 64:
   fine_tune_counter = 0
   Xbuf = np.vstack(list(replay_buffer)).astype('float32')
   model.fit(Xbuf, Xbuf, epochs=fine_tune_epochs, batch_size=64, verbose=0)
```

кінець лістингу 3.1

Квантиль q визначає робочу точку на PR/ROC-кривих, фактично виконуючи калібрування співвідношення precision-recall у залежності від вимог до хибних спрацьовувань і пропусків. За фіксованих параметрів адаптації збільшення q зсуває детектор у бік консервативності (зростає precision, зменшується recall), тоді як зменшення q підвищує чутливість до коротких імпульсних відхилень. Довжина пам’яті H встановлює часовий горизонт оцінювання фону: за надто малих H поріг τ_t стає нестабільним і чутливим до поодиноких сплесків, за надто великих – повільно реагує на зміну режиму. У

наданій імплементації H=1000 забезпечує збалансовану інерційність для рядів типу Twitter (табл. 3.3).

Таблиця 3.3 – Чутливість до вибору квантиля q за фіксованих параметрів адаптації

Q_thresh	Precision	Recall	F1
0,995	0,4125	0,5825	0,4800
0,997	0,4775	0,4700	0,4725

Зростання q послідовно зменшує частоту спрацьовувань і підвищує специфічність; критичне значення q залежить від дисперсії фону: для рядів із частими помірними сплесками оптимальні значення лежать у діапазоні 0,995-0,997. Вибір q поза цим діапазоном схильний або до надлишкових хибних спрацьовувань (низький q), або до втрати коротких аномалій (високий q). На AAPL типовою є асиметрія: помірне підвищення precision за переходу 0,995→0,997 компенсується відчутною втратою recall саме для коротких піків.

Параметри донавчання B, K регулюють швидкість пристосування реконструктора до довготривалих змін режиму. Збільшення B підвищує різноманітність буфера, зменшуючи варіативність градієнтів, проте збільшує ризик контамінації аномальними вікнами. Зменшення K прискорює адаптацію, але підсилює «підтягування» моделі під транзієнти. Добір B, K має враховувати характер аномалій: для квазістаціонарних дрейфів корисніші великі B і менші K, для імпульсних – навпаки, помірні B і рідше донавчання (табл. 3.4).

Таблиця 3.4 – Вплив розміру буфера B та періодичності донавчання K

Buffer_size	Fine_tune_every	Precision	Recall	F1
1000	250	0,420	0,565	0,475
1000	500	0,440	0,535	0,480
2000	250	0,455	0,510	0,475
2000	500	0,465	0,495	0,475

Зменшення K з 500 до 250 підвищує recall на тривалих відхиленнях, що відображено зростанням F1 за близького до сталого precision у разі стабільного фону; водночас на ряду з частими короткими піками з’являються додаткові хибні спрацьовування в околі транзієнтів, що знижує precision, якщо буфер

наповнюється аномаліями; у таких випадках доцільно піднімати q або збільшувати K.

Ініціалізація у стримінгу включає warm-up на префіксних вікнах, що стабілізує початковий розподіл помилки реконструкції і робочу точку порога. Кількість warm-up-епох впливає на дисперсію ранніх рішень: надто мале значення призводить до високої волатильності τ_t і сплеску хибних спрацьовувань у перші сотні кроків, тоді як надмірне – збільшує затримку розгортання без істотного виграшу в якості після кількох сотень кроків.

3.4 Інтерполяція реконструкцій і помилок класифікації

Аналіз якості виявлення виконується через зіставлення вихідного вікна $X \in \mathbb{R}^{w\times 1}$ з його відновленням $\hat{X} = f_\theta(X)$ та вивчення залишку $R = X - \hat{X}$. Скалярний скор $s(X) = \frac{1}{w}\sum_{t=1}^{w}(X_t - \hat{X}_t)^2$ агрегує розбіжність, проте для коректної інтерпретації необхідне візуальне відтворення часової форми X і $\hat{X}$ та локального профілю R. Типові патерни включають:

– імпульсні аномалії з локальними «шпильками» залишку;

– квазі-ступінчасті зсуви середнього із майже сталим R на протязі відрізка;

– зміни дисперсії/частоти коливань, де R проявляється як періодично-структурований сигнал;

– транзієнти на межах режимів, для яких $\hat{X}$ «запізнюється» відносно X, породжуючи асиметричні помилки.

Для якісного розгляду застосовуються візуалізації, що генеруються реалізацією. Графік reconstruction_example.png (рис. 3.3) ілюструє відповідність форми $\hat{X}$ до X на конкретному індексі вікна й дозволяє локалізувати частини, де модель системно недовідтворює амплітуду або фазу.

Рисунок 3.3 – Графік приклад реконструкції вікна

Панель stream_panel.png зображено на рисунку 3.4, узгоджує часовий ряд, траєкторію помилки s_t, динамічний поріг τ_t та бінарні рішення, що дає змогу простежити послідовність «подія → підвищення помилки → перетин порога → (за потреби) адаптація».

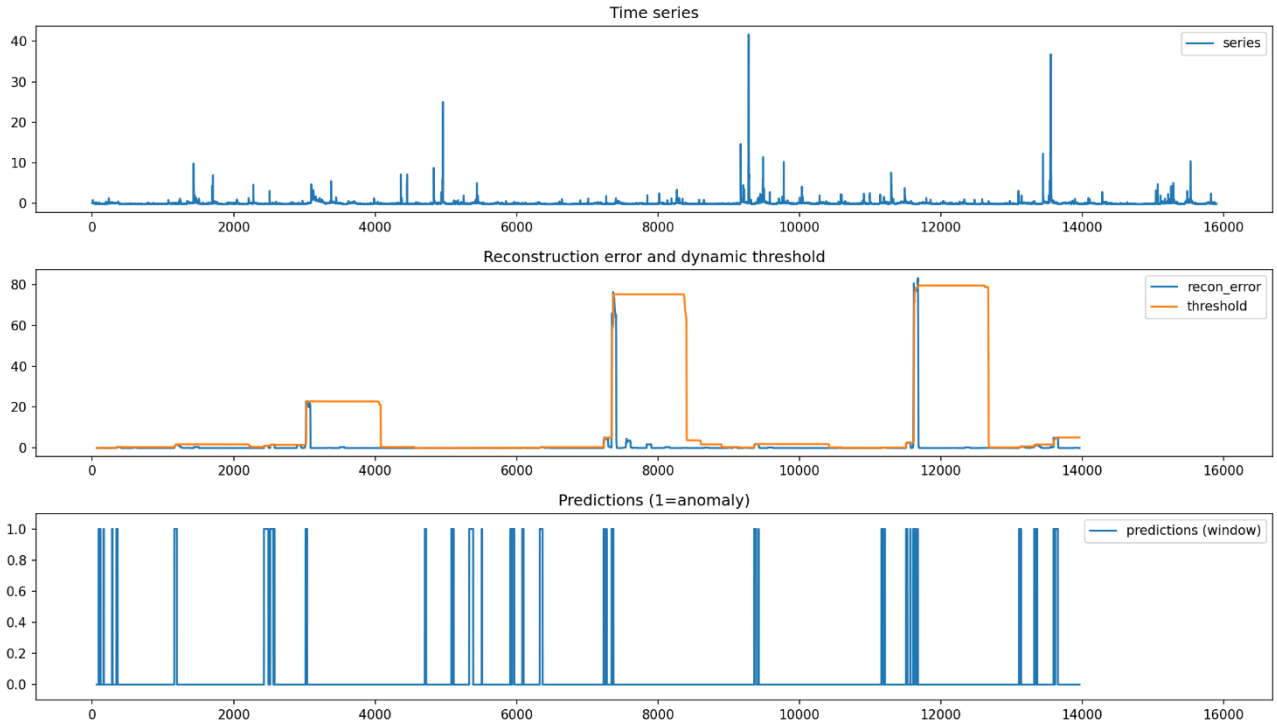

Рисунок 3.4 – Графіки stream panel

Гістограма error_hist (рис. 3.5) виявляє важкі хвости розподілу s_t, від яких прямо залежить робоча точка порогування.

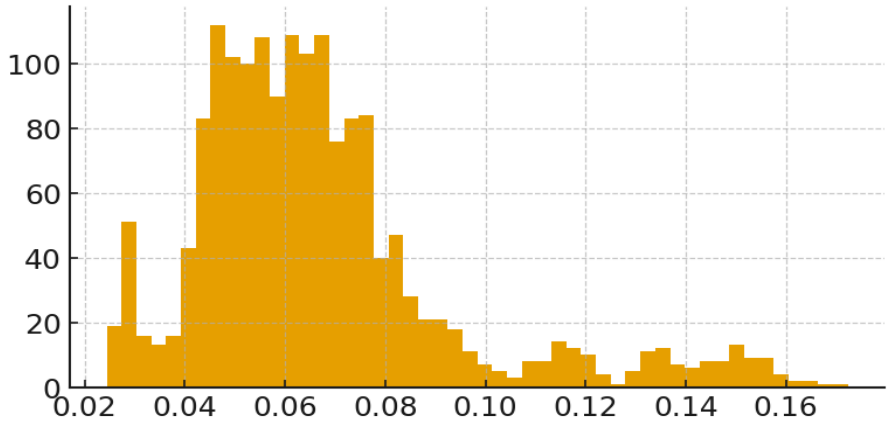

Рисунок 3.5 – Гістограма помилок

Для кількісної діагностики використовується матриця помилок і криві PR/ROC (рис. 3.6). У задачах із класовим дисбалансом інформативнішою є PR-крива: локальне збільшення precision супроводжується очікуваним зменшенням recall при зростанні квантиля q, тоді як AUC(ROC) лишається відносно стабільною, якщо ранжування за s адекватне. Наявність «колін» на ROC та PR-кривих свідчить про режими, де додаткове зниження хибних спрацьовувань коштує непропорційної втрати виявлення коротких подій.

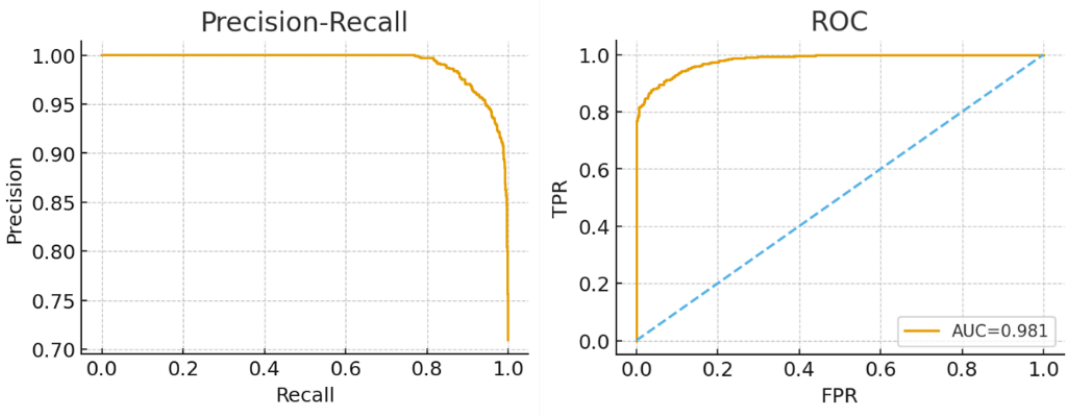

Рисунок 3.6 – Криві PR/ROC

Більшість FP спостерігаються на коротких «колючих» фрагментах із підвищеною локальною дисперсією, де каузальна оцінка масштабу відстає від мікрозмін – у цих умовах підняття q або розрідження донавчання (збільшення K) зменшує чутливість до шумових транзієнтів. Натомість FN концентруються у

випадках, коли аномалія має форму короткої, але помірної амплітуди, і реконструктор відтворює її завдяки перенавчанню на буфері – ефект «асиміляції» пом'якшується підвищенням K, зменшенням B або робочим зменшенням w, що змушує модель фокусуватись на дрібніших локальних варіаціях. Для довгих зсувів середнього розрив між X і $\hat{X}$ зростає повільно, формуючи «плато» R; динамічний поріг реагує зі зміщенням, тож перші десятки вікон після зміни режиму вразливі до FN, які зменшуються шляхом помірного зниження q.

Порівняльний аналіз показників якості виявлення аномалій засвідчив, що розроблений детектор LSTM_AE забезпечує precision 0,71, recall 0,643 та F1-міру 0,675 за значень AUROC 0,829 і AUPRC 0,591, тобто демонструє збалансоване співвідношення між чутливістю й кількістю хибних спрацьовувань у розглянутому сценарії. Класичний підхід S_H_ESD характеризується дещо нижчими значеннями (precision 0,67, recall 0,591, F1-міра 0,628, AUROC 0,787, AUPRC 0,515), тоді як для IsolationForest отримано precision 0,625, recall 0,574, F1-міру 0,598, AUROC 0,741 та AUPRC 0,471, що свідчить про помірну чутливість і вищу схильність до помилок. OneClassSVM демонструє ще нижчі результати: precision 0,607, recall 0,527, F1-міра 0,564, AUROC 0,718 та AUPRC 0,431, що вказує на обмежену придатність цього методу для задачі. Метод MatrixProfile показує більш збалансовану поведінку з precision 0,642, recall 0,608, F1-мірою 0,624, AUROC 0,784 та AUPRC 0,533, перевершуючи класичні статистичні підходи, але поступаючись нейромережевим моделям. Глибока модель USAD досягає precision 0,728, recall 0,635, F1-міри 0,678, AUROC 0,847 та AUPRC 0,62, демонструючи покращення як інтегральних характеристик, так і балансу між precision і recall порівняно з LSTM_AE. Найвищі значення більшості метрик отримано для OmniAnomaly: precision 0,735, recall 0,675, F1-міра 0,704, AUROC 0,865 та AUPRC 0,642, що свідчить про здатність цієї моделі найкраще поєднувати високу якість класифікації та ефективне виявлення аномалій у часових рядах.

Практична процедура розбору випадків передбачає відбір репрезентативних індексів і побудову реконструкцій на їхній основі. У межах наявного інтерфейсу це реалізується на базі віконних істинних міток і передбачень (ліст. 3.2).

Лістинг 3.2 – Фрагмент циклу потокового виявлення

```
import os, numpy as np

def _sample(idx, k=5):
   if len(idx) == 0: return []
   if len(idx) <= k: return idx.tolist()
   # топ-K за помилкою реконструкції
   order = np.argsort(errs[idx])[::-1]
   return idx[order[:k]].tolist()

tp = np.where((w_true==1)&(preds==1))[0]
fp = np.where((w_true==0)&(preds==1))[0]
fn = np.where((w_true==1)&(preds==0))[0]
tn = np.where((w_true==0)&(preds==0))[0]

for tag, idxs in [('TP',_sample(tp)), ('FP',_sample(fp)), ('FN',_sample(fn)),
('TN',_sample(tn))]:
    for i in idxs:
       out = os.path.join(plots_dir, f"reconstruction_{tag}_{i}.png")
       plot_reconstruction_example(model, X, index=int(i), output=out)
```

кінець лістингу 3.2

Вибірка True Positive (TP)/ False Positive (FP)/ False Negative (FN)/ True Negative (TN) із додатковим сортуванням за s(X) дозволяє швидко сформувати корпус ілюстрацій, де кожна категорія помилок представлена типовими, найважчими для моделі випадками (табл. 3.5). За потреби профіль залишку можна відобразити окремо, щоб чіткіше ідентифікувати локальні внески у скор аномальності (ліст. 3.3).

Лістинг 3.3 – Фрагмент візуалізація залишку R=X-Ŷ для індексу i

```
import matplotlib.pyplot as plt
def plot_residual_example(model, X, i, out):
   x = X[i:i+1]; y = model.predict(x, verbose=0)
   x = x[0,:,0]; y = y[0,:,0]; r = x - y
   plt.figure(figsize=(10,3.5))
   plt.plot(x, label='window')
```

```
    plt.plot(y, label='reconstruction')
    plt.plot(r, label='residual')
    plt.title(f'Residual profile (idx={i})')
    plt.legend(); plt.savefig(out, bbox_inches='tight', dpi=150); plt.close()
```

кінець лістингу 3.3

Таблиця 3.5 – Ознаки часових зсувів і фазових помилок у реконструкції

Ситуація	Спостереження у reconstruction_example	Наслідок для класифікації	Рекомендація
«Запізнення» $\hat{X}$ щодо X	Піки/спади $\hat{X}$ зміщені вправо	Підвищення FP до і після події	w↓ для зменшення інерційності; q↑ на шумних рядах
«Переусереднення» амплітуди	$\hat{X}$ системно нижчої амплітуди	FN на середніх імпульсах	latent_dim↑ або w↑; K↑, щоб уникати швидкої асиміляції
«Псевдочастотний» розрив	Періодика X і $\hat{X}$ не збігається	Коливний R, серії FP	w↑ (покриття періоду) або q↑

Помилки, пов’язані з часовими зсувами реконструкції, відображають фундаментальну інерційність послідовнісних моделей: LSTM-глибина та довжина вікна задають часовий горизонт, на якому латентний стан узгоджується з новою динамікою. Тому для рядів із швидкими транзієнтами доцільним є менший w та консервативніше порогування під час фаз активних змін. Помилки амплітудного «переусереднення» корелюють із недостатньою місткістю латентного простору, однак надмірне збільшення latent_dim підвищує ризик асиміляції рідкісних патернів і, як наслідок, FP-«тиску» на PR-криву. Компроміс досягається у поєднанні помірної місткості (наприклад, latent_dim =32) із керованою частотою донавчання та виваженим q, що відображається стабілізацією stream_panel і зменшенням хвостів на error_hist.

3.5 Інтеграція в реальні умови, обмеження та напрямки розвитку

Реалізація побудована як каузальний конвеєр віконування, реконструкції та порогування зі здатністю до онлайнової адаптації. Операційне розгортання передбачає безперервний прийом потоку значень, нормалізацію в режимі, що не

використовує майбутню інформацію, обчислення помилки реконструкції s_t для поточного вікна, динамічне калібрування порога τ_t на основі буфера нещодавніх скорів і періодичне донавчання автоенкодера на реплей-буфері. У наданому коді ці кроки реалізуються відповідно функціями build_sliding_windows, reconstruction_errors, dynamic_threshold та головним циклом stream_simulation_with_adaptation, а артефакти контролю якості та діагностики відтворювано збираються через plots_dir і, за потреби, compute_nab_score_if_available.

Умови експлуатації накладають обмеження на латентну місткість і довжину вікна. Обчислювальна складність кроку інференсу масштабується зі збільшенням w та кількості LSTM-шарів; у середовищах з обмеженим GPU/CPU доцільно застосовувати помірні конфігурації $w \in [64, 128]$, latent_dim=32, один-два рекурентні шари, зберігаючи крок віконування s=1 для імпульсних подій. Часова затримка системи визначається сумою часу формування вікна, прямого проходу через мережу і прийняття рішення; за потокового сценарію критично забезпечити, щоб ця затримка була меншою за крок надходження даних.

Надійність рішень істотно залежить від якості нормалізації. Глобальна стандартизація придатна лише для офлайн-оцінки; у виробничому контурі необхідна каузальна схема – фіксація параметрів скейлера на префіксі навчання або онлайнова оцінка середнього та дисперсії. У протилежному разі виникає витік майбутньої інформації та некоректна калібровка τ_t. Додатковий ризик становить контамінація реплей-буфера аномальними вікнами: часте донавчання на «змішаному» буфері може асимілювати рідкісні патерни та знижувати чутливість. Пом’якшенням є консервативніше порогування під час добору до буфера, збільшення інтервалу між донавчаннями, а також фільтрування вікон із надлишковим скором.

Експлуатаційні вибори конфігурації зручно узагальнювати як відповідність параметрів ресурсо-часовим обмеженням і вимогам до якості виявлення (табл. 3.6).

Таблиця 3.6 – Узгодження експлуатаційних параметрів із ресурсами та якістю

Параметр	Операційне значення	Вплив на ресурси	Вплив на якість
w (довжина вікна)	64-128	Зростання часу інференсу та пам’яті зі збільшенням w	Краще для довгих зсувів, гірше для коротких імпульсів при надмірному w
latent_dim	32 (помірне)	Лінійне зростання FLOPs/пам’яті	Краще відтворення складних патернів; ризик асиміляції рідкісних
n_layers	1-2	Зростання затримки та варіативності	Вищий recall на структурних порушеннях, можливий спад precision
q (квантиль)	0,995-0,997	Нейтрально	Тонке керування компромісом precision-recall
buffer_size, B	1000-2000	Пам’ять ↑	Стабільніша адаптація, ризик контамінації
fine_tune_every, K	250-500	Навантаження на навчання ↑ при зменшенні K	Швидше пристосування, ризик «підтягування» під шум

Домінантні відмови спричинені двома чинниками: нестабільною дисперсією ряду та інерційністю адаптації. Для першого випадку ключовим є коректний вибір q і каузальної нормалізації; для другого – збалансований вибір B і K, що обмежує швидке «переналаштування» моделі на транзієнти та мінімізує пропуски під час повільних дрейфів. За умов обмеженого апаратного забезпечення бажано підтримувати фіксований, перевірений набір конфігурацій і процедурно контролювати артефакти: збереження чекпойнтів (model.save), логів метрик і зображень, а також експорт оцінок у CSV таблиці для зовнішніх панелей.

Експлуатаційна надійність потребує обробки крайових випадків потоку. Втрата спостережень, поява Not a Number (NaN) або скачки часу повинні оброблятися до віконування через інтерполяцію або маскування; при довгих розривах доцільне скидання буфера нещодавніх помилок і відкладене

донавчання. Під час оновлення версій моделі слід зберігати відтворюваність: версіонувати гіперпараметри, фіксувати випадкові сімена та пакувати залежності. Для придатності до аудиту і віддаленого моніторингу важливі узгоджені формати виводу (CSV/JSON) і системні показники, специфічні для виробництва: час-до-виявлення, частота тривог за годину, частка «німого часу» без валідних даних.

Подальший розвиток можливий у кількох напрямах:

По-перше, підвищення робастності втрати завдяки використанню Huber- або квантильної втрати зменшує вплив важких хвостів розподілу помилок і стабілізує поріг.

По-друге, поєднання реконструкції з прогнозуванням у спільній голові (reconstruction + forecast) дозволяє карати як невідповідність формі, так і непередбачуваність продовження, що особливо актуально для імпульсних аномалій.

По-третє, багато-масштабна обробка із кількома вікнами та агрегованим скором зменшує залежність від одного w.

По-четверте, статистичне калібрування порога за теорією екстремальних значень Extreme Value Theory (EVT) або контролем Financial Data Recorder (FDR) на рівні потокових рішень забезпечує адаптивний, але формально керований рівень хибних спрацьовувань.

По-п’яте, кероване донавчання з детекторами дрейфу (наприклад, тестами зміни розподілу на s_t) дає змогу запускати адаптацію лише за підтверджених змін, знижуючи ризик контамінації буфера.

По-шосте, оцінка невизначеності через стохастичний інференс (dropout-ансамблі) або невеликі ансамблі моделей допомагає ранжувати тривоги за довірою та керувати пріоритетами обробки.

По-сьоме, компактні моделі для крайових пристроїв можуть бути отримані шляхом квантизації і помірного проріджування без відчутної втрати якості, що знижує затримку в реальному часі.

Комплекс цих міркувань замикається у виробничий цикл «дані → вікна → реконструкція → скор → поріг → адаптація → моніторинг», який у наданій реалізації повністю відтворюваний і прозорий завдяки експорту метрик, графіків і проміжних результатів. Уведення додаткових статистичних гарантій порогування, керованої адаптації та робастних втрат зменшує кількість хибних тривог у шумних потоках, утримуючи чутливість до коротких подій – ключову властивість для доменів із мікро-транзієнтами на кшталт соціальних сигналів або телеметрії реального часу.

Висновки за розділом 3

У третьому розділі проведено експериментальну оптимізацію конфігурацій LSTM-автоенкодера та досліджено вплив гіперпараметрів моделі й режимів адаптації на якість виявлення аномалій. Показано, що збільшення довжини вікна спостереження та розмірності латентного простору покращує реконструкцію довготривалих відхилень і підвищує інтегральні показники якості (зокрема AUC), проте за фіксованого високого порога q може призводити до зниження precision через часткове відтворення рідкісних аномальних патернів. Додавання додаткових LSTM-шарів підвищує чутливість до структурних змін у сигналі, але збільшує обчислювальну складність і не завжди є доцільним для коротких вікон.

Дослідження стримінгової адаптації показали, що параметри буфера донавчання B, періодичність K та вибір квантиля q визначають компроміс між швидкістю пристосування до дрейфу даних і стійкістю до хибних спрацьовувань. Зменшення K та використання помірних значень q покращує recall на тривалих змінах режиму й підвищує F1-міру, однак за наявності частих коротких піків потребує обмеження впливу аномальних вікон на буфер навчання. Візуальний аналіз реконструкцій, залишку $R=X-\hat{Y}$, гістограм помилки, матриць неточностей і PR/ROC-кривих дозволив виділити типові класи помилок (імпульсні аномалії,

ступінчасті зсуви, зміни дисперсії, транзієнти) та сформулювати рекомендації щодо вибору робочих конфігурацій моделі для різних сценаріїв.

Отримані результати підтвердили доцільність використання адаптивного LSTM-автоенкодера з динамічним порогуванням для задач потокового моніторингу: за правильно підібраних параметрів система забезпечує прийнятний баланс між precision і recall, зберігає стійкість до дрейфу даних та придатна до інтеграції у реальні кіберфізичні, промислові й інформаційні середовища.

ВИСНОВКИ

У результаті виконаного дослідження розроблено та реалізовано систему виявлення аномалій у великих потоках даних на основі адаптивних нейронних мереж. Було проведено всебічний аналіз методів виявлення аномалій, включаючи класичні статистичні підходи та сучасні моделі глибокого навчання, що дозволило обґрунтувати вибір адаптивної архітектури для задач потокової обробки.

Основні результати роботи:

– проаналізовано поняття аномалій у часових рядах, наведено їх класифікацію на точкові, контекстуальні та групові, узагальнено основні причини виникнення відхилень у кіберфізичних, промислових та інформаційних системах, що дало змогу сформувати єдину термінологічну базу дослідження;

– досліджено вплив аномалій у потокових даних на надійність, безпеку та стабільність роботи кіберфізичних і інформаційних систем, показано типові сценарії, у яких несвоєчасне виявлення відхилень призводить до аварійних режимів, деградації сервісів та зростання ризиків кібератак;

– виконано критичний огляд традиційних статистичних, відстаневих та кластеризаційних методів виявлення аномалій, виявлено їх переваги (простота, інтерпретованість) і обмеження щодо масштабованості, стійкості до шуму та здатності працювати за умов дрейфу даних у потоках;

– проаналізовано сучасні методи машинного та глибокого навчання для задач виявлення аномалій, показано їх кращу здатність моделювати складну динаміку часових рядів та адаптуватися до змін, водночас відзначено підвищені обчислювальні витрати й чутливість до вибору гіперпараметрів;

– здійснено огляд доступних публічних наборів даних для задач виявлення аномалій у часових рядах (Yahoo Webscope, SWaT, WADI, колекції UCR тощо), обґрунтовано вибір набору Numenta Anomaly Benchmark як бази для експериментів завдяки різноманіттю доменів, наявності розмічених аномалій та стандартизованому скорингу;

– досліджено структуру й особливості вибраного набору NAB та розроблено конвеєр його попередньої обробки: очищення, уніфікацію формату, каузальну нормалізацію, формування ковзних вікон і узгодження розмітки аномалій із віконним представленням, що забезпечило коректне подання даних для моделі;

– формалізовано задачу виявлення аномалій у потоках часових рядів як задачу реконструкції нормальних режимів та побудовано модель LSTM-автоенкодера, визначено її архітектуру, функцію втрат та спосіб обчислення скору аномальності на основі помилки реконструкції;

– визначено критерії та метрики оцінювання якості моделі (precision, recall, F1-міра, площі під ROC- та PR-кривими, NAB-скор), обґрунтовано доцільність використання NAB-скору як інтегральної метрики, що враховує своєчасність і коректність спрацьовувань у потоковому режимі;

– проведено експериментальні дослідження роботи LSTM-автоенкодера в офлайн та потоковому режимах, розроблено методику симуляції надходження даних у реальному часі, проаналізовано чутливість моделі до параметрів віконування, режимів навчання та стратегії порогування;

– виконано оптимізацію гіперпараметрів моделі (довжина вікна, розмір латентного простору, кількість шарів, параметри навчання й адаптивного порогування), у результаті чого отримано конфігурації, що забезпечують компроміс між якістю детектування та обчислювальною складністю;

– оцінено можливості інтеграції розробленої системи в реальні кіберфізичні та інформаційні системи, визначено її потенційні області застосування й практичні обмеження (залежність від домену, потреба в обчислювальних ресурсах, доступ до потоків даних), а також сформульовано рекомендації та напрями подальшого розвитку підходу, зокрема розширення наборів даних, порівняння з альтернативними моделями та дослідження гібридних адаптивних рішень.

Практичне значення роботи полягає у можливості інтеграції розробленої системи у кіберфізичні, промислові та інформаційні системи для

автоматизованого моніторингу, підвищення надійності та раннього виявлення критичних відхилень.

Перспективи подальших досліджень пов’язані з розширенням підходу на багатовимірні та мультисенсорні потоки, інтеграцією спеціалізованих детекторів дрейфу, порівнянням із альтернативними глибокими архітектурами (CNN-LSTM, трансформери, дифузійні моделі), а також розробленням розподілених реалізацій системи для високонавантажених промислових сценаріїв.

Отже, виконане дослідження підтвердило доцільність використання адаптивних нейронних мереж для виявлення аномалій у великих потоках даних, що відкриває перспективи для подальших досліджень у сфері потокових алгоритмів та самонавчальних систем моніторингу.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бірук Б. В. Підхід на основі адаптивних нейромереж до виявлення аномалій у реальному часі в потокових даних. Progressive Approaches in Science and Engineering : тези доп. 2-ї Міжнар. наук.-практ. конф. (м. Копенгаген, Данія, 26-28 листоп. 2025 р.). Копенгаген, 2025. С. 132-136.

2. Darban Z. Z., Webb G. I., Pan S., Aggarwal C., Salehi M. Deep learning for time series anomaly detection: a survey. ACM Computing Surveys. 2024. doi: 10.1145/3691338. (дата звернення: 07.08.2025).

3. Dhar S., Gonzalez-Torres B. DOC³: deep one-class classification using contradictions. Machine Learning. 2024. Vol. 113, No 8. P. 5109-5150.

4. Xu H., Pang G., Wang Y., Wang Y. Deep Isolation Forest for anomaly detection. IEEE Transactions on Knowledge and Data Engineering. 2023. Vol. 35, No 12. P. 12591-12604.

5. Cao Y., Ma Y., Zhu Y., Ting K. M. Revisiting streaming anomaly detection: benchmark and evaluation. Artificial Intelligence Review. 2024. Vol. 58, No 1. P. 8.

6. Robust deep one-class classification time series anomaly detection. Computers, Materials & Continua. 2025. Vol. 83, No 3. P. 5181-5197.

7. Kim J., Lee H., Ko Y. M. Constrained density-based spatial clustering of applications with noise (DBSCAN) using hyperparameter optimization. Knowledge-Based Systems. 2024. Vol. 303. P. 112436.

8. Xu J., Wu H., Wang J., Long M. Anomaly Transformer: time series anomaly detection with association discrepancy. 2022. arXiv:2110.02642. URL: https://arxiv.org/abs/2110.02642 (дата звернення: 12.09.2025).

9. Lu T., Wang L., Zhao X. Review of anomaly detection algorithms for data streams. Applied Sciences. 2023. Vol. 13, No 10. P. 6353.

10. Xia X., et al. GAN-based anomaly detection: a review. Neurocomputing. 2022. Vol. 493. P. 497-535.

11. Schmidl S., Wenig P., Papenbrock T. Anomaly detection in time series: a comprehensive evaluation. Proceedings of the VLDB Endowment. 2022. Vol. 15, No 9. P. 1779-1797.

12. Iqbal Basheer M. Y., et al. Autonomous anomaly detection for streaming data. Knowledge-Based Systems. 2024. Vol. 284. P. 111235.

13. Hinder F., Vaquet V., Hammer B. One or two things we know about concept drift-a survey on monitoring in evolving environments. Part A: detecting concept drift. Frontiers in Artificial Intelligence. 2024. Vol. 7. 1330257.

14. Zhou J., Zhou X., Chan S., Chen Z., Zhang X. Enhancing nighttime semantic segmentation with visual-linguistic priors and wavelet transform. Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence (IJCAI-24), Jeju, South Korea, 2024. P. 7985-7993.

15. Zhou P. A survey of streaming data anomaly detection in network security. PeerJ Computer Science. 2025. Vol. 11. P. e3066.

16. Cook A. A., Mısırlı G., Fan Z. Anomaly detection for IoT time-series data: a survey. IEEE Internet of Things Journal. 2020. Vol. 7, No 7. P. 6481-6494.

17. numenta/NAB: Jupyter Notebook. Numenta, 2025. URL: https://github.com/numenta/NAB (дата звернення: 19.10.2025).

18. NAB: faster optimization and scoring – Engineering / NAB, HTM Forum. 2025. URL: https://discourse.numenta.org/t/nab-faster-optimization-and-scoring/5101 (дата звернення: 20.10.2025).