

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»**

**Навчально-науковий інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу**

До захисту допущено:
Завідувач кафедри
_____ Оксана ТИМОЩУК
« ____ » _____ 2026 р.

Дипломна робота

на здобуття ступеня бакалавра

**за освітньо-професійною програмою «Системний аналіз і управління»
спеціальності 124 «Системний аналіз» на тему: «Інтелектуальна система
виявлення аномалій у даних технологічних процесів критичної
інфраструктури на основі методів глибинного навчання»**

Виконала:

Студентка IV курсу, групи КА-21

Михальчук Євгенія Богданівна _____

Керівник:

Асистент каф. ММСА Древаль Максим Михайлович _____

Консультант з економічного розділу:

Доцент каф. ЕК, к.е.н. Рощина Надія Василівна _____

Консультант з нормоконтролю:

Асистент каф. ММСА, докт. філос.,

Канцедал Георгій Олегович _____

Рецензент:

Доцент, докт. філос., Гуськова Віра Геннадіївна _____

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.

Студентка _____

Київ – 2026 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Навчально-науковий інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

Рівень вищої освіти – перший (бакалаврський)
 Спеціальність – 124 «Системний аналіз»
 Освітня програма «Системний аналіз і управління»

ЗАТВЕРДЖУЮ
 Завідувач кафедри
 _____ Оксана ТИМОЩУК
 « ____ » _____ 2026 р.

ЗАВДАННЯ
на дипломну роботу студентці
Михальчук Євгенії Богданівні

1. Тема роботи «Інтелектуальна система виявлення аномалій у даних технологічних процесів критичної інфраструктури на основі методів глибинного навчання», керівник роботи Древаль Максим Михайлович, асистент, затверджені наказом по університету від 27.05.2026 р. № 1944-с
2. Термін подання студентом роботи «10» червня 2026 р.
3. Вихідні дані до роботи: Набір телеметричних даних сенсорів та актуаторів промислової кіберфізичної системи (SWaT).
4. Зміст роботи: 1 – Аналіз предметної області; 2 – Математичне моделювання методів виявлення аномалій; 3 – Програмна реалізація моделей та порівняльний аналіз результатів; 4 – Функціонально-вартісний аналіз програмного забезпечення.
5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): Ілюстрації роботи програмного продукту, презентація.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Функціонально-вартісний аналіз	Рощина Н.В., доц., к.е.н.		

7. Дата видачі завдання 19.04.2026

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної області та формулювання завдання	20.04.2026 – 03.05.2026	виконано
2	Вивчення літератури та обробка даних за темою роботи	04.05.2026 – 17.05.2026	виконано
3	Програмна реалізація нейромережових моделей виявлення аномалій та попередня алгоритмічна обробка технологічних даних	18.05.2026 – 24.05.2026	виконано
4	Тестування розробленої інтелектуальної системи, підбір гіперпараметрів детекції та порівняльний аналіз результатів	25.05.2026 – 28.05.2026	виконано
5	Оформлення та узгодження пояснювальної записки і розділів роботи	29.05.2026 – 04.06.2026	виконано
6	Попередній захист дипломної роботи	08.06.2026 – 12.06.2026	виконано
7	Захист дипломної роботи	15.06.2026 – 21.06.2026	виконано

Студентка _____

Євгенія МИХАЛЬЧУК

Керівник _____

Максим ДРЕВАЛЬ

РЕФЕРАТ

Дипломна робота: 115 с., 22 рис., 13 табл., 2 додатки, 18 джерел.

АВТОЕНКОДЕР, ВИЯВЛЕННЯ АНОМАЛІЙ, ГІСТЕРЕЗИСНА ФІЛЬТРАЦІЯ, ГЛИБИННЕ НАВЧАННЯ, КІБЕРФІЗИЧНІ СИСТЕМИ, КРИТИЧНА ІНФРАСТРУКТУРА, ПООЗНАКОВИЙ ПОРІГ, ЧАСОВІ РЯДИ, LSTM.

Об'єкт дослідження — процеси оброблення та аналізу телеметричних даних промислових кіберфізичних систем об'єктів критичної інфраструктури.

Предмет дослідження — методи глибинного навчання на основі автоенкодерів для автоматизованого виявлення аномалій у багатовимірних часових рядах телеметрії кіберфізичної системи.

Мета роботи — створення інтелектуальної системи виявлення аномалій у даних технологічних процесів, яка моделює нормальну поведінку кіберфізичної системи та автоматично фіксує відхилення, спричинені несправностями або кібервтручаннями, без використання розмічених прикладів атак.

Актуальність — конвергенція ІТ та ОТ-технологій розширила поверхню атак на критичну інфраструктуру, а традиційні засоби захисту неефективні в ОТ-мережах через пропріетарні протоколи, обмежені ресурси пристроїв і вимогу безперервної роботи. Це зумовлює потребу в інтелектуальних системах виявлення поведінкових аномалій без втручання в логіку технологічного процесу.

Результати роботи — розроблено програмний продукт мовою Python, що реалізує повний цикл оброблення телеметричних даних стенду SWaT 2015: попередню обробку, стратифіковане leak-free формування вибірок, навчання чотирьох конфігурацій детектора (M1–M4) та калібрування порогів за бюджетом хибних тривог. Запропоновано та експериментально перевірено поознаковий (per-feature) поріг із двопороговою гістерезисною фільтрацією.

ABSTRACT

Thesis: 115 p., 22 fig., 13 tab., 2 appendices, 18 references.

AUTOENCODER, ANOMALY DETECTION, HYSTERESIS FILTERING, DEEP LEARNING, CYBER-PHYSICAL SYSTEMS, CRITICAL INFRASTRUCTURE, PER-FEATURE THRESHOLD, TIME SERIES, LSTM.

Object of research — the processing and analysis of telemetry data of industrial cyber-physical systems in critical-infrastructure facilities.

Subject of research — autoencoder-based deep learning methods for automated anomaly detection in multivariate telemetry time series of a cyber-physical system.

Purpose of the work — the development of an intelligent anomaly-detection system for technological-process data that models the normal behaviour of a cyber-physical system and automatically flags deviations caused by faults or cyber-intrusions, without using labelled attack examples.

Relevance — the convergence of IT and OT technologies has expanded the attack surface of critical infrastructure, while traditional security measures are ineffective in OT networks due to proprietary protocols, limited device resources, and the requirement for continuous operation. This necessitates intelligent systems for detecting behavioural anomalies without interfering with the logic of the technological process.

Results of the work — a software product was developed in Python implementing the full telemetry-data processing cycle for the SWaT 2015 testbed: preprocessing, stratified leak-free dataset construction, training of four detector configurations (M1–M4), and threshold calibration based on a false-alarm budget. A per-feature threshold with two-stage hysteresis filtering was proposed and experimentally validated.

ЗМІСТ

ВСТУП	11
РОЗДІЛ 1 ТЕОРЕТИЧНИЙ АПАРАТ	13
1.1 Опис об'єкта дослідження як кіберфізичної системи	13
1.2 Класифікація аномалій і ризиків у КФС	16
1.3 Огляд підходів до виявлення аномалій у технологічних даних...	17
1.4 Формалізація задачі системного аналізу	21
1.5 Критерії оцінки ефективності інтелектуальної системи.....	22
1.6 Висновки до розділу 1	23
РОЗДІЛ 2 МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ ТА ВИБІР АЛГОРИТМІВ	24
2.1 Математична модель віконної обробки (Sliding Window).....	24
2.2 Архітектури нейромережових моделей	26
2.2.1 Повнозв'язний автоенкодер (Dense-AE, модель M1).....	26
2.2.2 Згортковий автоенкодер (1D-CNN-AE, модель M4).....	27
2.2.3 Рекурентний автоенкодер (LSTM-AE, моделі M2 та M3)	28
2.3 Цільова функція та алгоритм оптимізації	30
2.3.1 Цільова функція MSE	30
2.3.2 Оптимізатор Adam	31
2.4 Алгоритми формування порогу детекції	32
2.4.1 Показник аномальності (Anomaly Score)	32
2.4.2 Глобальний перцентильний поріг	33
2.4.3 Поознаковий поріг (Per-Feature Thresholding)	33
2.4.4 Стратегія leak-free калібрування	35

2.5 Критерії оцінки ефективності	37
2.5.1 Точкові метрики (Point-wise Metrics).....	37
2.5.2 Подієві метрики (Event-based Metrics).....	38
2.5.3 Затримка детекції (Detection Delay)	38
2.6 Висновки до розділу 2	39
РОЗДІЛ 3 РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНА ПЕРЕВІРКА	
МЕТОДІВ ВИЯВЛЕННЯ АНОМАЛІЙ	40
3.1 Програмне середовище та технологічний стек.....	40
3.2 Підготовка та попередня обробка даних	41
3.2.1 Завантаження та кешування.....	41
3.2.2 Очищення даних	42
3.2.3 Відсікання перехідного режиму	42
3.2.4 Нормалізація ознак	42
3.3 Формування навчальної та тестової вибірок.....	43
3.3.1 Параметри датасету та розбивки	43
3.3.2 Виділення інтервалів атак	44
3.3.3 Класифікація атак за масштабом.....	45
3.3.4 Стратифікований leak-free спліт.....	46
3.3.5 Побудова ковзних вікон	48
3.4 Реалізація нейромережових моделей	49
3.5 Обчислення показника аномальності та калібрування порогів ...	50
3.5.1 Помилки реконструкції	50
3.5.2 Калібрування за бюджетом хибних тривог	51
3.5.3 Глобальний та поознаковий порогови.....	51
3.6 Результати експериментів та їх аналіз	52

3.6.1 Точкові та подієві метрики	53
3.6.2 Подієва повнота за типами атак	54
3.6.3 Аналіз кривих ROC та Precision–Recall	55
3.6.4 Матриці помилок	56
3.6.5 Розподіл помилки реконструкції.....	57
3.6.6 Теплова карта поознакової помилки та локалізація аномалій	58
3.6.7 Часова шкала детекції.....	59
3.6.8 Компроміс «повнота — хибні тривоги»	60
3.6.9 Аналіз чутливості параметра гістерезисної фільтрації (Ablation Study)	61
3.7 Обмеження та загрози достовірності результатів	64
3.8 Збереження артефактів і можливість розгортання	65
3.9 Висновки до розділу 3	65

РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Постановка задачі проектування.....	69
4.2 Обґрунтування функцій програмного продукту.....	70
4.3 Обґрунтування системи параметрів програмного продукту	73
4.4 Аналіз експертного оцінювання параметрів	77
4.5 Аналіз рівня якості варіантів реалізації функцій.....	79
4.6 Економічний аналіз варіантів розробки програмного продукту .	81
4.7 Вибір кращого варіанту за техніко-економічним рівнем	85
4.8 Висновки до розділу 4	87

ВИСНОВКИ.....	88
---------------	----

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ	91
ДОДАТОК А ЛІСТИНГ КОДУ	94
ДОДАТОК Б ПРЕЗЕНТАЦІЯ	108

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

KI — критична інфраструктура

КФС — кіберфізична система

OT — Operational Technology (операційні технології)

IT — Information Technology (інформаційні технології)

ICS — Industrial Control System (промислова система управління)

SCADA — Supervisory Control and Data Acquisition

PLC — Programmable Logic Controller

HMI — Human-Machine Interface

LSTM — Long Short-Term Memory

AE — Autoencoder

MAE — Mean Absolute Error

MSE — Mean Square Error

RNN — Recurrent Neural Network

SSSP — Single Stage Single Point

SSMP — Single Stage Multi Point

MSMP — Multi Stage Multi Point

MSSP — Multi Stage Single Point

LOTL — Living-off-the-land

DoS — Denial of Service

SWaT — Secure Water Treatment

ВСТУП

Цифрова трансформація промислового сектору призвела до глибокої інтеграції інформаційних технологій (ІТ) з операційними технологіями (ОТ), що забезпечують функціонування критичної інфраструктури. Сучасні промислові системи управління (ІС) та системи збору даних і диспетчерського управління (SCADA) більше не є ізольованими об'єктами. Вони використовують стандартні мережеві протоколи та хмарні сервіси для оптимізації технологічних процесів. Проте така конвергенція значно розширила поверхню атак, роблячи критичну інфраструктуру (КІ) вразливою до складних кіберфізичних загроз.

Аналіз інформаційних загроз у 2025–2026 роках свідчить про зміну стратегій зловмисників: від банального проникнення в систему до цілеспрямованого вивчення логіки й технологічних процесів підприємства для подальших диверсій. Сучасні загрози, такі як діяльність угруповань ELECTRUM та VOLTZITE, спрямовані на дослідження фізики процесів КІ для подальшого завдання фізичних пошкоджень [1]. Особливу небезпеку становить використання методів «Living-off-the-land» (LOTL) та експлуатація вразливостей периферійних пристроїв, що дозволяє зловмисникам залишатися непоміченими протягом тривалого часу [1].

Традиційні сигнатурні методи захисту є неефективними для ОТ-мереж через використання пропрієтарних протоколів, низьку обчислювальну потужність кінцевих пристроїв та необхідність безперервної роботи систем. Це зумовлює потребу у впровадженні інтелектуальних систем моніторингу, які здатні виявляти аномалії шляхом аналізу поведінкових патернів у промислових даних[2].

Актуальність роботи пов'язана із частими атаками на критичну інфраструктуру та потребою в інструментах моніторингу, які б фіксували відхилення без порушення детермінованих процесів.

Мета роботи полягає у створенні інтелектуальної системи виявлення аномалій у даних технологічних процесів за допомогою методів глибинного навчання. Модель моделює нормальну поведінку об'єкта та виявляє приховані аномалії, спричинені збоями або кібервтручанням.

Для досягнення поставленої мети розв'язуються такі завдання:

- проведення аналізу специфіки промислових часових рядів та класифікація типів аномалій у критичних системах;
- розгляд сучасних архітектур нейронних мереж (Dense-AE, 1D-CNN-AE, LSTM-AE) для аналізу ОТ-трафіку;
- формалізація математичної моделі детекції відхилень через реконструкцію вхідних сигналів;
- програмна реалізація алгоритму виявлення аномалій та оцінка його ефективності за метриками точності, повноти та швидкодії.

Об'єктом дослідження є процеси обробки та передачі даних у промислових мережах об'єктів критичної інфраструктури. Предметом дослідження є використання методів глибинного навчання для автоматизованого виявлення аномалій у цих даних (наприкладі кіберфізичної системи SWaT).

РОЗДІЛ 1 ТЕОРЕТИЧНИЙ АПАРАТ

У першому розділі розглядаються математичні та системні основи задачі виявлення аномалій у даних промислової критичної інфраструктури. Описано об'єкт дослідження, класифіковано типи аномалій та ризиків, проведено порівняльний аналіз сучасних методів детекції, сформульовано математичну модель та визначено критерії оцінки ефективності системи.

1.1 Опис об'єкта дослідження як кіберфізичної системи

Об'єктом дослідження у даній роботі є система Secure Water Treatment (SwaT) — повномасштабна індустріальна установка очищення води, що є еталонною моделлю для дослідження безпеки критичної інфраструктури. SwaT класифікується як кіберфізична система (КФС), оскільки вона поєднує в собі складні фізичні процеси фільтрації та хімічної обробки з розподіленим цифровим керуванням на базі промислових контролерів.

Система спроектована iTrust (Сінгапур) за підтримки Міністерства оборони Сінгапуру як точна копія реального водоочисного заводу [3]. Датасет SwaT 2015 містить 41 сценарій атак, із них 36 мають задокументований фізичний вплив) [4]. Технологічний цикл SwaT розділений на шість взаємопов'язаних підсистем (P1–P6) [5].

1. P1 (Raw Water Storage): збір сирової води. Включає резервуар T101, де рівень контролюється датчиком LIT101, а подача регулюється клапаном MV101 та насосами P101/P102.

2. P2 (Pre-Treatment): хімічне дозування NaCl, HCl та NaOCl для коригування рН та дезінфекції.

3. P3 (Ultrafiltration): ультрафільтрація для видалення твердих частинок та бактерій через систему керамічних мембран.

4. P4 (Dechlorination): дехлорування за допомогою UV-ламп та NaHSO_3 .

5. P5 (Reverse Osmosis): зворотний осмос — основний етап очищення з прецизійним контролем провідності (AIT503/504) та тиску.

6. P6 (Distribution & Backwash): розподіл очищеної води та промивка мембран.

Узагальнену блок-схему фізичних процесів системи SWaT наведено на рисунку 1.1.

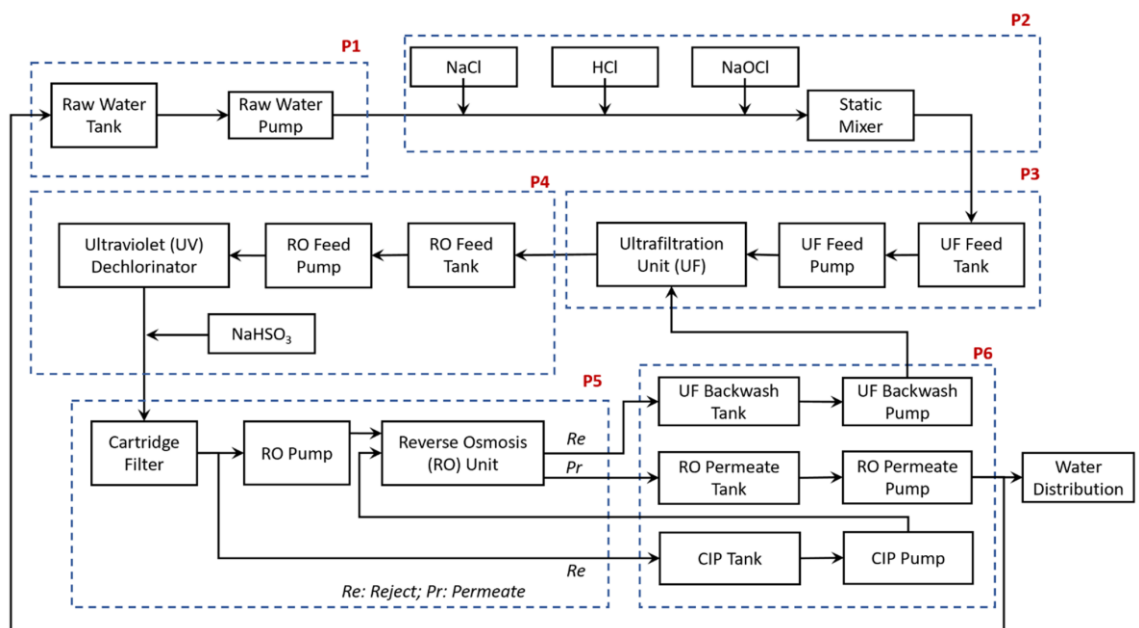


Рисунок 1.1 — Блок-схема фізичних процесів SwaT [5]

Архітектура SwaT побудована відповідно до рівнів моделі Пердью (Purdue Model):

- 1) Рівень 0 — сенсори та актуатори;
- 2) Рівень 1 — PLC Allen-Bradley;
- 3) Рівень 2 — HMI та SCADA;
- 4) Рівень 3 — сервер Historian [4, 5].

Загальну мережеву архітектуру системи SWaT за рівнями моделі Пердью схематично подано на рисунку 1.2

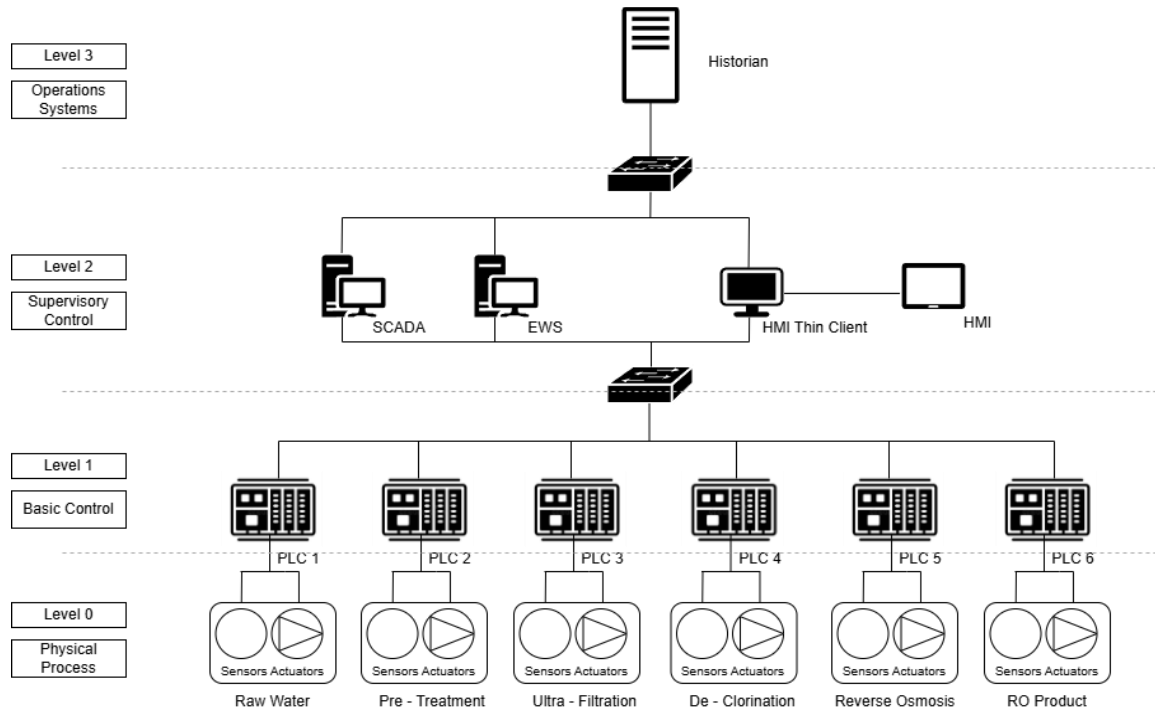


Рисунок 1.2 — Архітектура мережі SwaT [5]

Класифікацію вимірювальних і керуючих пристроїв системи SwaT наведено в таблиці 1.1.

Таблиця 1.1 — Класифікація вимірювальних та керуючих пристроїв системи SwaT [5]

Тип тега	Повна назва	Функціональне призначення	Приклади
LIT	Level Indicating Transmitter	Моніторинг рівня рідини в резервуарах	LIT101, LIT301, LIT401
FIT	Flow Indicating Transmitter	Вимірювання об'ємної витрати води	FIT101, FIT201, FIT503
AIT	Analytical Indicating Transmitter	Вимірювання pH, провідності, ORP	AIT201, AIT402, AIT503
PIT/DPIT	Pressure Indicating Transmitter	Контроль тиску на фільтрах та мембранах	PIT501, DPIT301
MV	Motorized Valve	Відкриття/закриття потоку рідини	MV101, MV201, MV301
P	Pump	Перекачування води; стани On/Off	P101, P203, P501
LS	Level Switch	Дискретний датчик граничного рівня	LS201, LS601, LS603

1.2 Класифікація аномалій і ризиків у КФС

У контексті аналізу критичної інфраструктури аномалія визначається як будь-яке відхилення поведінки системи від її детермінованого або очікуваного статистичного стану. Класифікація аномалій охоплює фізичну та цифрову площини.

Усі аномалії в часових рядах технологічних даних поділяються на три фундаментальні категорії: технічні несправності, випадкові збої та навмисні кібератаки. Технічні несправності (Malfunctions) — це деградаційні процеси фізичного рівня, такі як дрейф характеристик сенсорів, механічний знос актуаторів або засмічення фільтрів; вони проявляються як поступова зміна тренду або збільшення дисперсії сигналу. Випадкові збої (Failures) є короткочасними порушеннями, спричиненими зовнішніми завадами або помилками програмного забезпечення, і генерують поодинокі викиди (outliers). Навмисні кібератаки (Cyber-Physical Attacks) полягають у цілеспрямованій зміні стану системи; вони часто є «стелс-аномаліями», коли зломисник маніпулює даними так, щоб кожен окремий параметр залишався в нормі, але їхня сукупність порушувала фізичну цілісність процесу.

Залежно від об'єкта впливу виокремлюють кілька основних векторів атак на технологічні процеси.

Маніпуляція даними сенсорів (Sensor Spoofing): підміна значень вимірювань хибними. Якщо PLC отримує занижені дані про рівень рідини, він продовжує підвищувати тиск, що призводить до фізичного руйнування резервуара.

Маніпуляція актуаторами (Actuator Manipulation): пряме втручання в команди управління насосами та клапанами.

Атаки на фізичну семантику (Physical-Semantic Attacks): використання знань про фізику процесу для створення критичних станів через каскадні збої.

Мережеві аномалії (DoS та Replay): затримки пакетів, що порушують детермінованість управління, та атаки повтору для приховування поточної аварії.

За масштабом впливу атаки на технологічні процеси поділяють на чотири типи.

Single Stage Single Point (SSSP): вплив на один сенсор в межах одного технологічного етапу.

Single Stage Multi Point (SSMP): координована атака на кілька елементів одного контуру управління.

Multi Stage Single Point (MSSP): вплив на один сенсор, наслідки якого через фізичні зв'язки між підсистемами поширюються на кілька технологічних стадій.

Multi Stage Multi Point (MSMP): найбільш небезпечний тип, що охоплює різні підсистеми КІ. Такі аномалії майже неможливо виявити традиційними пороговими методами.

1.3 Огляд підходів до виявлення аномалій у технологічних даних

Класичні методи виявлення аномалій базуються на статистичному аналізі розподілу даних або геометричному розділенні простору ознак.

Серед класичних підходів широко застосовують статистичні методи та моделі авторегресії. Найбільш поширеним підходом є використання моделей типу ARIMA (AutoRegressive Integrated Moving Average). Стан системи прогнозується як лінійна комбінація попередніх значень та помилок. Аномалією вважається ситуація, коли залишок (residual) перевищує статистично визначений поріг. Такі моделі ефективні лише для одновимірних лінійних рядів і не здатні вловити складні кореляції між 50+ датчиками.

Isolation Forest (Ізоляційний ліс) [6]. Алгоритм використовує ансамбль випадкових дерев для «ізоляції» спостережень. Аномалії потребують меншої кількості випадкових розрізів, що відображається у більш короткому середньому шляху до спостереження. Перевагою є висока швидкість роботи та лінійна обчислювальна складність, що робить метод придатним для великих обсягів даних.

One-Class SVM (OC-SVM) [7]. Метод полягає у пошуку гіперсфери мінімального радіуса у просторі ознак, яка охоплює більшість нормальних точок. Недоліком є обчислювальна складність етапу навчання сягає $O(n^2) - O(n^3)$, що робить класичний OC-SVM непридатним для аналізу довгих промислових часових рядів обсягом у сотні тисяч відліків.

Обмеження класичних методів зумовлюють перехід до методів глибинного навчання, здатних автоматично моделювати нелінійні просторово-часові залежності між десятками сенсорів. Далі розглянуто основні архітектури цього класу.

Autoencoder (AE). Навчається відновлювати власний вхідний сигнал через «пляшкове горло» (bottleneck) — латентний простір нижчої розмірності. Цільова функція — MSE між вхідними та відновленими даними. Недолік: ігнорує часову послідовність, розглядаючи кожен зріз даних незалежно.

LSTM-Autoencoder (LSTM-AE). Вдосконалена архітектура, де нейронні шари замінені на блоки Long Short-Term Memory. LSTM керує потоком інформації через три гейти: гейт забування (f), гейт входу (i) та гейт виходу (o). Дозволяє моделювати інерційність фізичних процесів та виявляти аномалії, які проявляються через поступову зміну динаміки [8]. Формальний опис архітектури наведено у розділі 2.

Variational Autoencoder (VAE) [9]. На відміну від AE, навчає параметри імовірного розподілу в латентному просторі. Цільова функція включає помилку реконструкції та відстань Кульбака–Лейблера (KL divergence). Забезпечує більшу стійкість до шумів у промислових даних.

Transformer-based моделі та USAD [10]. Механізм Self-Attention дозволяє моделі фокусуватися на найбільш корельованих ознаках незалежно від їхнього часового відставання. Алгоритм USAD поєднує принципи AE та GAN, підвищуючи чутливість до прихованих атак. На датасеті SWaT USAD демонструє $F1 \approx 0,846$ з використанням схеми point-adjustment.

Граф-неймережеві методи (GDN) [11]. Graph Deviation Network (GDN) моделює кореляційну структуру між сенсорами у вигляді графа залежностей. Кожен вузол відповідає одному тегу, а ребра відображають виявлені статистичні зв'язки між ними. Підхід демонструє $precision \approx 0,99$ на SWaT також при оцінюванні за point-adjustment.

Гібридні архітектури MTAD-GAT та TranAD [12]. MTAD-GAT (ICDM 2020) поєднує часове прогнозування та реконструкцію з механізмом Graph Attention Networks. TranAD (VLDB 2022) використовує трансформер із двофазною процедурою навчання з масштабуванням, яка підсилює аномальні сигнали. Обидві моделі досягають рекордних результатів ($F1 \approx 0,90-0,93$ з point-adjustment) на наборі даних SWaT.

Згортковий автоенкодер (1D-CNN-AE). Використовує одновимірні згорткові шари для виділення локальних часових ознак із багатовимірного сигналу. Завдяки ковзним ядрам модель здатна ефективно виявляти короточасні (транзйентні) аномалії та різкі стрибки показників. Перевагою є висока швидкість навчання через можливість паралелізації обчислень, проте обмежене рецептивне поле згортки робить цей метод менш ефективним для моделювання довготривалої інерції фізичних процесів порівняно з рекурентними мережами.

Порівняльну характеристику розглянутих підходів до виявлення аномалій узагальнено в табл. 1.2.

Таблиця 1.2 — Порівняння підходів до виявлення аномалій

Метод	Часовий контекст	Тип навчання	Переваги	Недоліки
Statistical (ARIMA)	Так (лінійно)	Статистичне	Простота	Не працює з багатовимірністю
Isolation Forest	Ні	Unsupervised	Швидкість	Ігнорує послідовність
Dense AE	Ні	Unsupervised	Стиснення ознак	Пропускає динамічні атаки
LSTM-AE	Так (нелінійно)	Unsupervised	Захоплює інерцію процесів	Складність навчання
VAE	Ні / Так	Unsupervised	Стійкість до шумів	Ризик перенавчання
USAD / GAN	Так	Unsupervised	Висока Precision	Потребує GPU-ресурсів
GDN	Так (граф)	Unsupervised	Моделює кореляції тегів	Складне побудування графа
MTAD-GAT / TranAD	Так	Unsupervised	SOTA точність	Висока складність
1D-CNN-AE	Так (локально)	Unsupervised	Виявляє локальні швидкі зміни, швидко навчається	Обмежене рецептивне поле, погано захоплює довготривалу інерцію процесів

На основі проведеного аналізу для реалізації системи найбільш підходять три архітектури автоенкодерів для порівняльного дослідження: повнозв'язний Dense-AE як базову лінію (baseline), згортковий 1D-CNN-AE для фіксації локальних швидких змін, та рекурентний LSTM-AE як основну модель, оскільки вона найкраще здатна моделювати часову інерцію фізичних процесів КФС без використання розмічених даних.

Для підвищення чутливості обраного рекурентного підходу в межах цієї роботи передбачено вдосконалення процедури прийняття рішень шляхом переходу від глобального усереднення до поознакового аналізу помилок реконструкції із застосуванням часової фільтрації сигналів тривоги.

1.4 Формалізація задачі системного аналізу

Об'єкт дослідження генерує багатовимірний часовий ряд, що є результатом дискретного зчитування значень F датчиків через рівні проміжки часу Δt . Математично вибірка представляється у вигляді матриці [10]:

$$X \in R^{\{T \times F\}}, x(t) = [x_{1(t)}, x_{2(t)}, \dots, x_{F(t)}]^T$$

де T — загальна кількість часових кроків;

F — кількість ознак (сенсорів та актуаторів);

$x(t)$ — вектор стану системи в момент часу t .

Для врахування часової послідовності застосовується метод «ковзного вікна» (sliding window) з довжиною w кроків. Вхідним сигналом для моделі є підмножина:

$$W(t) = [x(t - w + 1), x(t - w + 2), \dots, x(t)] \in R^{\{w \times F\}}$$

Запропонований підхід ґрунтується на навчанні без учителя: модель засвоює лише нормальний режим роботи системи.

Задача ставиться як пошук відхилень у режимі *unsupervised*. Модель реалізує стиснення та відновлення інформації через дві функції:

$$h = \text{Encoder}(W(t); \theta_e) \in R^d, d \ll w \cdot F$$

$$\hat{W}(t) = \text{Decoder}(h; \theta_d) \in R^{w \times F}$$

де θ_e та θ_d — набори параметрів нейронної мережі, які підлягають оптимізації.

Критерієм навчання є мінімізація середньоквадратичної помилки реконструкції (MSE) на «чистій» вибірці нормального режиму [10]:

$$L(\theta) = \frac{1}{N} \sum_n \frac{1}{w \cdot F} \sum_i \sum_j \left(x_{ij}^{(n)} - \hat{x}_{ij}^{(n)} \right)^2$$

Фінальний крок конвеєра — прийняття рішення про наявність аномалії на основі порогового правила. Для ідентифікації аномалії в момент t

обчислюється показник аномальності (anomaly score). Рішення про наявність аномалії $\hat{y}(t)$ приймається за правилом:

$$\hat{y}(t) = 1[\text{score}(t) > \tau],$$

де τ — поріг детекції.

Вибір τ є критичним оскільки низьке значення призводить до хибних тривог (False Positives), а високе — до пропуску реальних атак (False Negatives).

1.5 Критерії оцінки ефективності інтелектуальної системи

Точкові метрики (Point-wise metrics)

Для оцінки класифікації кожного вектора стану використовуються стандартні метрики на основі матриці помилок (Confusion Matrix) [10, 13]:

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$F1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}},$$

де TP — вірно виявлені аномалії;

FP — хибні тривоги;

FN — пропущені аномалії.

Точкові метрики не враховують, що оператору важливо виявити саму атаку, а не кожен її відлік. Цю властивість оцінюють подієві метрики.

Атака A визначається як неперервний часовий інтервал $[\text{start}(A), \text{end}(A)]$, де $y(t) = 1$. Атака вважається виявленою, якщо система згенерувала хоча б одну тривогу в межах цього інтервалу. Подієва повнота:

$$\text{Event Recall} = \frac{|\{A: \exists t \in A, \hat{y}(t) = 1\}|}{|\{A\}|}$$

Найкритичнішим параметром для КФС є час реакції системи. Затримка детекції $d(A)$ визначається як:

$$d(A) = \min\{t \in [\text{start}(A), \text{end}(A)]: \hat{y}(t) = 1\} - \text{start}(A)$$

У критичній інфраструктурі затримка детекції має прямий зв'язок із ризиком виникнення аварії. Фізичні процеси (інерція потоку, хімічні реакції, накопичення тиску) є незворотними.

1.6 Висновки до розділу 1

У першому розділі описано математичну природу задачі виявлення аномалій у кіберфізичних системах на прикладі системи SWaT. Об'єкт дослідження ідентифіковано як шестиетапну систему очищення води з ієрархічною архітектурою управління на базі PLC Allen-Bradley.

Проведена класифікація аномалій за природою походження (технічні несправності, випадкові збої, навмисні кібератаки) та за масштабом впливу (SSSP, SSMP, MSSP, MSMP). Визначено, що найбільшу небезпеку становлять MSMP-атаки, що маскуються під нормальну роботу окремих вузлів.

Проведено порівняльний аналіз методів виявлення аномалій, розширений до сучасного SOTA 2020–2022 рр. (GDN, MTAD-GAT, TranAD). За його результатами обґрунтовано вибір для практичної реалізації трьох архітектур: Dense-AE (як базової лінії), 1D-CNN-AE (для виявлення локальних транз'єнтів) та LSTM-AE, яка найкраще здатна виявляти контекстні аномалії та моделювати довготривалу інерцію фізичних процесів КФС. Сформульовано математичну модель задачі та визначено систему об'єктивних метрик оцінки як точкових (Precision, Recall, F1) так і подієвих (Event Recall, Detection Delay). Визначено напрямки вдосконалення системи, а саме застосування поозначового порогу з часовою фільтрацією в поєднанні зі строгою *leak-free* стратегією калібрування на нормі тестового домену.

РОЗДІЛ 2 МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ ТА ВИБІР АЛГОРИТМІВ

У другому розділі сформульовано та математично обґрунтовано ключові рішення, покладені в основу реалізованої системи виявлення аномалій. Наведено формальний опис вікнової обробки часових рядів; описано архітектури трьох нейромережових моделей — щільного автоенкодера (Dense-AE), одновимірного згорткового автоенкодера (1D-CNN-AE) та рекурентного автоенкодера (LSTM-AE). Обґрунтовано вибір цільової функції та алгоритму оптимізації; описано два підходи до формування порогу детекції — глобальний та поознаковий (per-feature); визначено систему метрик оцінки ефективності.

2.1 Математична модель віконної обробки (Sliding Window)

Вхідним сигналом системи є багатовимірний часовий ряд, що генерується промисловою установкою SWaT (датасет 2015 року). Кожний запис відповідає одній секунді спостережень і містить значення $F = 51$ вимірювального або керуючого тегу. Повна матриця спостережень записується у вигляді:

$$X \in R^{T \times F}, x(t) = [x_1(t), x_2(t), \dots, x_F(t)]^T \quad (2.1)$$

де T — загальна кількість дискретних часових кроків;

F — кількість ознак (сенсорів та актуаторів).

Безпосередня подача окремих векторів $x(t)$ у нейромережову модель не дозволяє врахувати часову залежність між послідовними станами системи. Для моделювання динаміки процесу застосовується метод ковзного вікна

(sliding window) довжиною w . Вхідним об'єктом для моделі слугує тривимірний тензор:

$$W(t) = [\hat{x}(t - w + 1), \hat{x}(t - w + 2), \dots, \hat{x}(t)] \in R^{w \times F} \quad (2.2)$$

де $\hat{x}(t)$ — нормалізований вектор стану у момент t .

Сусідні вікна зсунуто на один крок ($stride = 1$) на стадії тестування та на $TRAIN_STRIDE = 2$ кроки на стадії навчання для прискорення оброблення. Перший вимір тензору (w) відповідає часовій глибині вікна; другий (F) — простору ознак. Таким чином навчальна вибірка набуває форми:

$$X_{train} \in R^{N_w \times w \times F}, \quad (2.3)$$

де $N_w = \left\lfloor \frac{N_{train} - w}{stride} \right\rfloor + 1$ — кількість навчальних вікон;

N_{train} — кількість рядків нормального режиму.

У реалізованій системі фіксовано $w = 20$. Значення обрано як компроміс між часовою глибиною контексту, необхідною для виявлення повільних атак класу MSMP, та обчислювальними витратами. Довжина вікна визначає лише латентність буферизації, тобто час від появи відхилення до моменту, коли воно повністю потрапляє у вікно аналізу (не більше w секунд); фактична затримка виявлення атаки як події залежить від динаміки показника аномальності та перетину ним порога і оцінюється окремо у підрозділі 3.6 (метрика Detection Delay)

Нормалізація ознак виконується методом Min-Max Scaling за параметрами, обчисленими виключно на навчальних нормальних даних:

$$\hat{x}_j(t) = \frac{x_j(t) - \min_j}{\Delta_j} \quad (2.4)$$

де $\min_j$ та $\max_j$ — мінімум і максимум j -ї ознаки на навчальній вибірці;

$\Delta_j = \max_j - \min_j$ — розмах значень j -ї ознаки.

Для константних ознак, у яких розмах наближається до нуля ($\Delta_j < 10^{-9}$), програмно приймається $\Delta_j = 1$, що автоматично перетворює їхні нормалізовані значення на нулі та виключає хибний вплив на загальну похибку реконструкції.

Параметри масштабування зберігаються та застосовуються до тестових даних без повторного обчислення — це запобігає витоку інформації про аномальні стани та гарантує, що будь-які аномальні відхилення проявляться у відповідних значеннях нормалізованого сигналу, а не будуть поглинуті масштабуванням.

2.2 Архітектури нейромережових моделей

У роботі реалізовано та порівняно три архітектури автоенкодерів, що утворюють послідовно ускладнений ряд за рівнем часового моделювання: Dense-AE (базова лінія без явного часового контексту), 1D-CNN-AE (локальні часові патерни) та LSTM-AE (довготривалі часові залежності).

2.2.1 Повнозв'язний автоенкодер (Dense-AE, модель M1)

Найпростіша архітектура реалізує стиснення через повнозв'язні шари без явного часового моделювання. Вхідне вікно $W \in \mathbb{R}^{w \times F}$ спочатку розгортається у вектор $v \in \mathbb{R}^{w \cdot F}$:

$$v = \text{Flatten}(W) \in \mathbb{R}^{w \cdot F}$$

Після цього застосовуються два шари кодувальника з функцією активації ReLU:

$$h_1 = \text{ReLU}(W_1 v + b_1), h_1 \in \mathbb{R}^{128}$$

$$z = \text{ReLU}(W_2 h_1 + b_2), z \in \mathbb{R}^{32},$$

де z — латентне представлення розмірності 32.

Декодувальник симетрично відновлює вхід:

$$\hat{W} = \text{Reshape}(W_4(\text{ReLU}(W_3 z + b_3)) + b_4), \hat{W} \in \mathbb{R}^{w \times F}$$

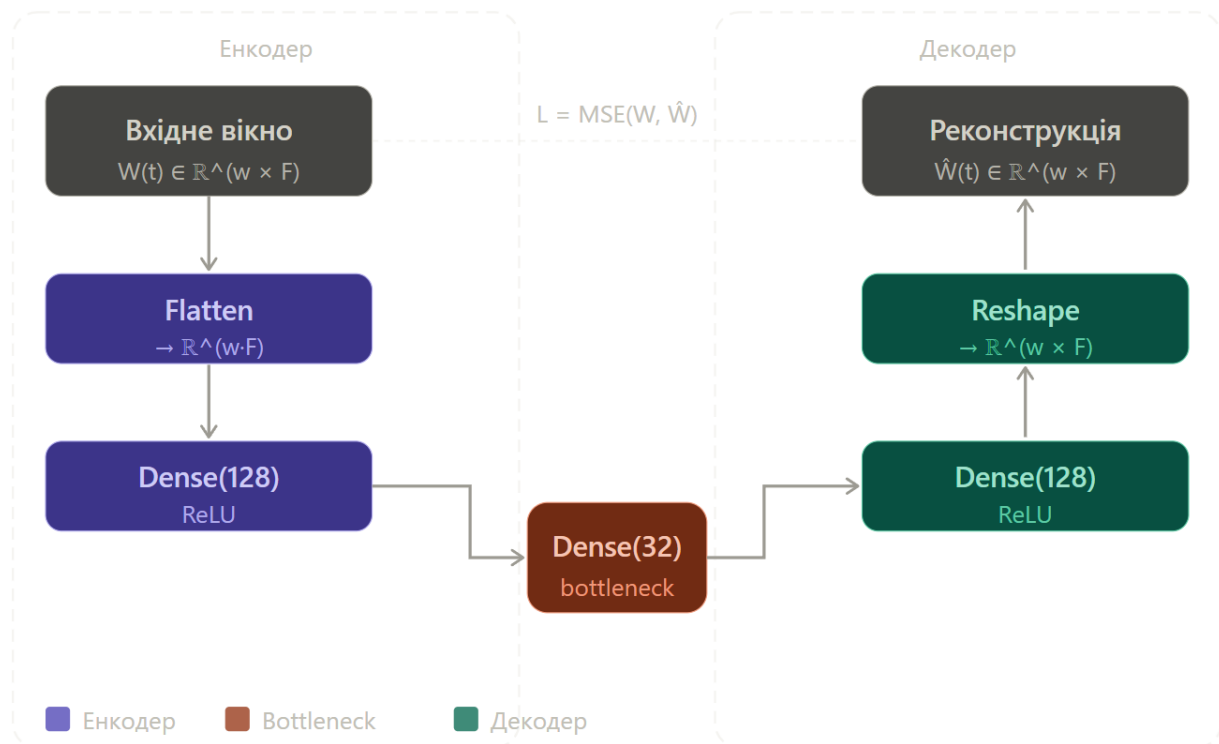


Рисунок 2.1 — Архітектура Dense-AE

2.2.2 Згортковий автоенкодер (1D-CNN-AE, модель M4)

Згортковий автоенкодер виявляє локальні часові патерни за допомогою одновимірних ковзних ядер, що ковзають уздовж часової осі вікна. Кодувальник складається з двох шарів Conv1D[14]:

$$H_1 = \text{ReLU}(\text{Conv1D}(W, K_1, k = 3)), H_1 \in \mathbb{R}^{w \times 64}$$

$$Z = \text{ReLU}(\text{Conv1D}(H_1, K_2, k = 3)), Z \in \mathbb{R}^{w \times 32}$$

де $k = 3$ — розмір ядра; padding = “same” зберігає довжину послідовності.

Декодувальник використовує транспоновані згортки (Conv1DTranspose):

$$H_2 = \text{ReLU}(\text{Conv1DTranspose}(Z, K_3, k = 3)), H_2 \in \mathbb{R}^{w \times 32}$$

$$H_3 = \text{ReLU}(\text{Conv1DTranspose}(H_2, K_4, k = 3)), H_3 \in \mathbb{R}^{w \times 64}$$

$$\hat{W} = \text{Conv1D}(H_3, K_5, k = 3, \text{linear}), \hat{W} \in \mathbb{R}^{w \times F}$$

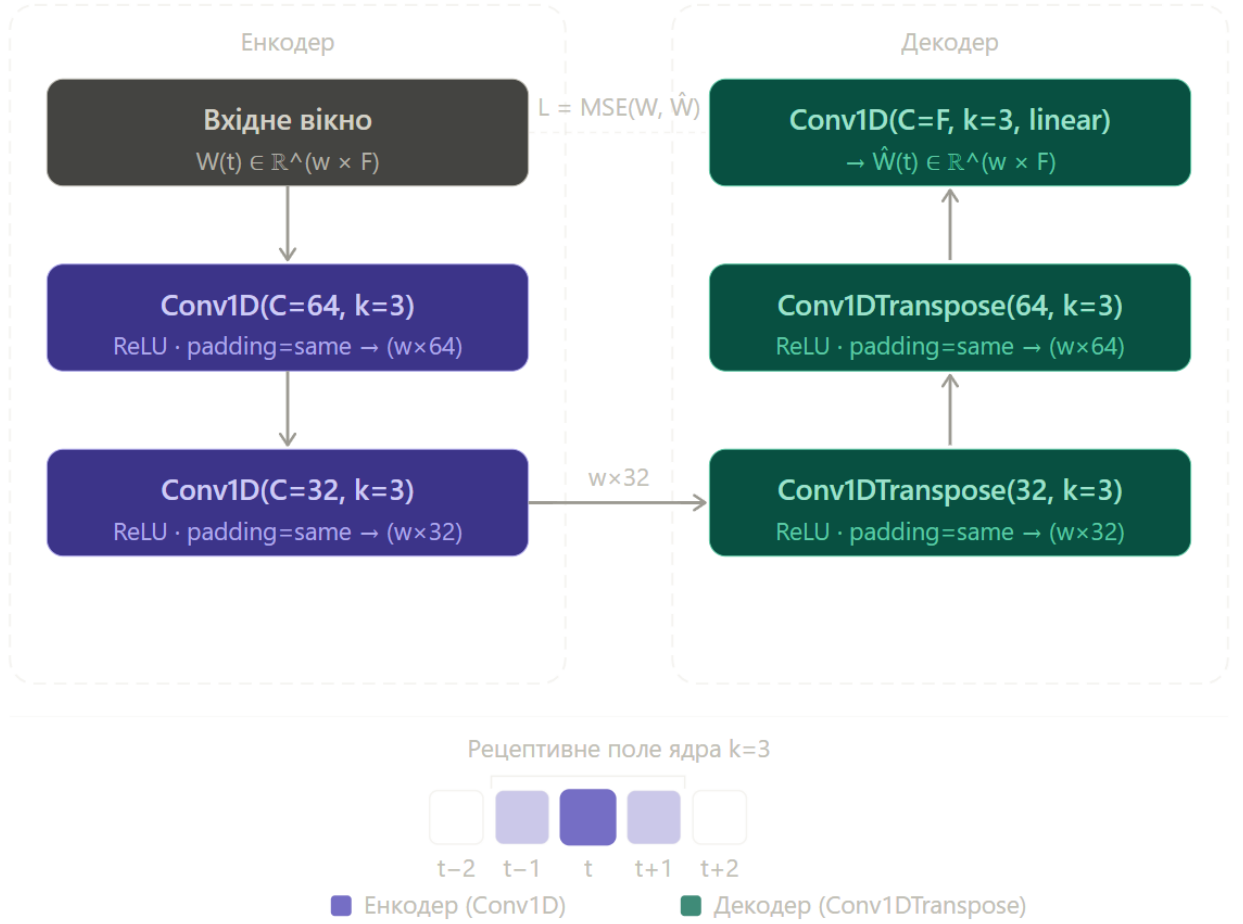


Рисунок 2.2 — Архітектура 1D-CNN-AE

2.2.3 Рекурентний автоенкодер (LSTM-AE, моделі M2 та M3)

Рекурентна архітектура є ключовою в роботі та базується на блоках Long Short-Term Memory [15]. Кожен LSTM-блок управляє потоком інформації через систему трьох гейтів і клітинний стан c_t . Гейт забування (forget gate) визначає, яку частку попереднього стану клітини слід зберегти:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

Гейт входу (input gate) регулює надходження нової інформації:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$

Гейт виходу (output gate) формує прихований стан h_t :

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t \odot \tanh(C_t)$$

Кодувальник LSTM-AE реалізує двошарову рекурентну мережу з поступовим стисненням:

$$H_1 = \text{LSTM}(W, d_h = 64, \text{return_sequences} = \text{True}), H_1 \in \mathbb{R}^{w \times 64}$$

$$z = \text{LSTM}(H_1, d_h = 32, \text{return_sequences} = \text{False}), z \in \mathbb{R}^{32},$$

де z — фінальний прихований стан, що стискає увесь w -кроковий контекст до вектора розмірності 32.

Оскільки декодувальник також є рекурентним і потребує послідовність на вході, прихований стан z тиражується за допомогою шару RepeatVector:

$$\tilde{Z} = \text{RepeatVector}(z, w) \in \mathbb{R}^{w \times 32}$$

$$H_2 = \text{LSTM}(\tilde{Z}, d_h = 32, \text{return_sequences} = \text{True}), H_2 \in \mathbb{R}^{w \times 32}$$

$$H_3 = \text{LSTM}(H_2, d_h = 64, \text{return_sequences} = \text{True}), H_3 \in \mathbb{R}^{w \times 64}$$

$$\hat{W} = \text{TimeDistributed}(\text{Dense}(F))(H_3) \in \mathbb{R}^{w \times F}$$

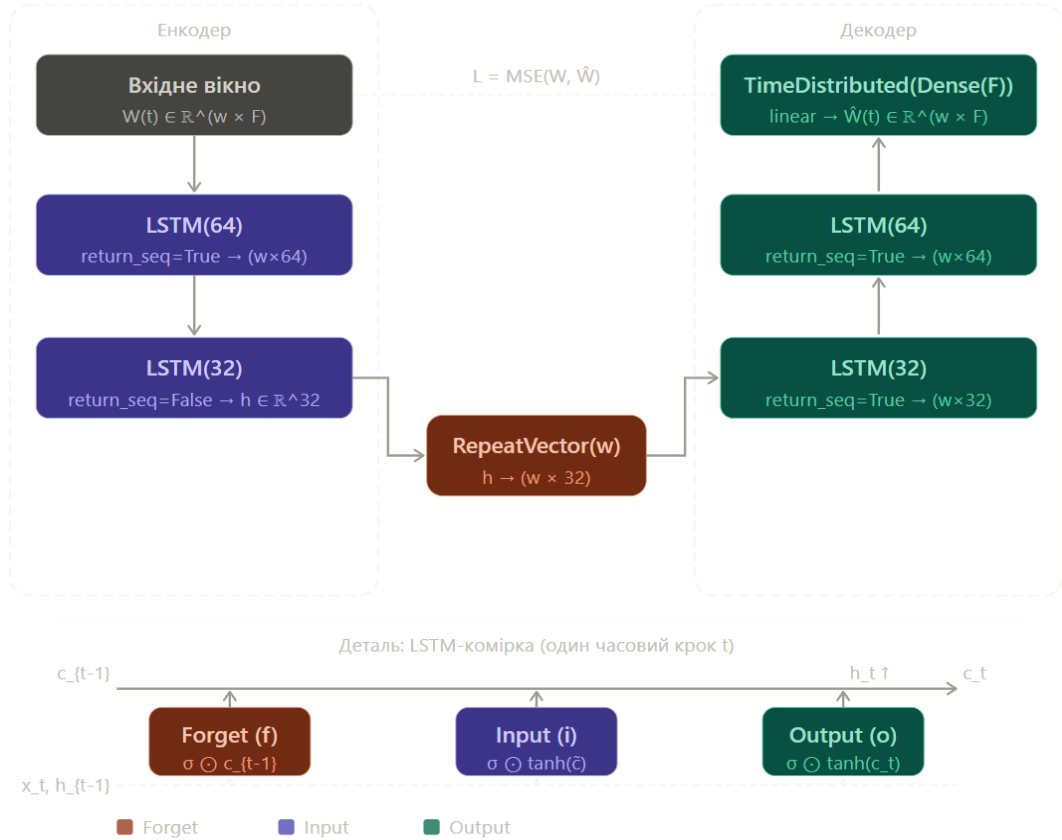


Рисунок 2.3 — Архітектура LSTM-AE

2.3 Цільова функція та алгоритм оптимізації

2.3.1 Цільова функція MSE

Для навчання автоенкодерів усіх трьох архітектур обрано критерій середньоквадратичної помилки реконструкції (Mean Squared Error). Функція втрат на одному навчальному вікні X та його реконструкції $\hat{X}$ визначається як:

$$L(X, \hat{X}) = \frac{1}{w \cdot F} \sum_{i=1}^w \sum_{j=1}^F (x_{ij} - \hat{x}_{ij})^2,$$

де w — довжина вікна;

F — кількість ознак;

x_{ij} — нормалізоване значення j -ї ознаки на i -му кроці вікна;

$\hat{x}_{ij}$ — відновлене значення.

Загальна функція втрат по N_w навчальних вікнах:

$$L(\theta) = \frac{1}{N_w} \sum_{n=1}^{N_w} L(X_n, \hat{X}_n(\theta))$$

Вибір MSE як цільової функції обґрунтований кількома її важливими математичними та практичними властивостями. Зокрема, квадратичний штраф суттєво підсилює великі відхилення порівняно з MAE, що стимулює модель під час навчання до максимально точної реконструкції «важких» вікон у перехідних режимах. Водночас диференційованість MSE у всьому просторі параметрів забезпечує стійке оцінювання градієнтів алгоритмами оптимізації. Також мінімізація цієї функції еквівалентна максимізації правдоподібності за умови гаусівського шуму, що є цілком природним припущенням для метрологічних промислових сенсорів.

На стадії інференсу, безпосереднього формування показника аномальності, замість MSE використовується середня абсолютна похибка (MAE). Оскільки вона є значно менш чутливою до поодиноких різких викидів

у нормальних даних, її застосування дозволяє суттєво підвищити стабільність роботи порогового правила.

2.3.2 Оптимізатор Adam

Для оновлення параметрів $\theta = \{\theta_e, \theta_d\}$ усіх трьох моделей застосовується алгоритм Adam (Adaptive Moment Estimation) [16]. Алгоритм підтримує ковзні оцінки першого (m_t) і другого (v_t) моментів градієнта функції втрат $\nabla L(\theta_t)$:

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot \nabla L(\theta_t)$$

$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot (\nabla L(\theta_t))^2$$

Через ініціалізацію $m_0 = v_0 = 0$ оцінки зміщуються до нуля на початкових ітераціях. Для компенсації цього зміщення обчислюються скориговані оцінки:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

Крок оновлення параметрів:

$$\theta_{t+1} = \theta_t - \frac{\alpha \cdot \hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon},$$

де $\alpha = 0,001$ — початкова швидкість навчання;

$\beta^1 = 0,9, \beta^2 = 0,999$ — коефіцієнти експоненційного згладжування;

$\epsilon = 10^{-8}$ — числова стабілізація.

2.4 Алгоритми формування порогу детекції

Поріг детекції τ є критичним гіперпараметром системи: занижений поріг призводить до хибних тривог (False Positives), а завищений — до пропуску атак (False Negatives). У реалізованій системі запропоновано та порівняно два підходи до формування τ .

2.4.1 Показник аномальності (Anomaly Score)

Після завершення навчання для кожного вікна $W(t)$ у тестовій вибірці обчислюється помилка реконструкції. Глобальний показник аномальності $g(t)$ усереднює MAE по всіх ознаках та часових кроках вікна:

$$g(t) = \frac{1}{w \cdot F} \sum_{i=1}^w \sum_{j=1}^F |x_{ij}(t) - \hat{x}_{ij}(t)|$$

Для зниження чутливості до поодиноких викидів застосовується ковзне середнє з вікном $k = 10$ кроків (лівостороннє, без look-ahead):

$$s(t) = \frac{1}{k} \sum_{i=0}^{k-1} g(t-i)$$

Додатково для кожного вікна $W(t)$ обчислюється поозначковий вектор помилки — середня MAE по часовій осі для кожної ознаки j окремо:

$$pf_j(t) = \frac{1}{w} \sum_{i=1}^w |x_{ij}(t) - \hat{x}_{ij}(t)|$$

Завдяки $pf_j(t)$ система здатна не лише констатувати факт аномалії, але й ідентифікувати конкретний тег, що вносить найбільший вклад у помилку реконструкції:

$$j(t) = \operatorname{argmax}_j pf_j(t)$$

2.4.2 Глобальний перцентильний поріг

Перший підхід ґрунтується на q -му перцентилі розподілу згладженого показника аномальності, обчисленого на нормальних вікнах валідаційної підмножини, тобто на нормі тестового домену:

$$\tau(q) = \text{Quantile}\left(\{s(t): t \in N_{val}\}, \frac{q}{100}\right),$$

де N_{val} — множина нормальних вікон валідаційної підмножини.

Поріг калібрується за стратегією бюджету хибних тривог: з фіксованої сітки перцентилів обирається найчутливіший (найнижчий) квантиль q^* , за якого частка хибних тривог на нормі валідації ще не перевищує заданий бюджет TARGET_FP:

$$q^* = \min\{q \in Q: \text{FPrate}_{cal}(q) \leq \text{TARGET_FP}\}, \quad (2.5)$$

де $Q = 90, 95, 97, 98, 99, 99, 5, 99, 9, 99, 95\%$;

TARGET_FP = 0,03 — бюджет хибних тривог.

Бінарне рішення формується пороговим правилом над згладженим показником:

$$\hat{y}(t) = 1[s(t) > \tau(q^*)]$$

2.4.3 Поознаковий поріг (Per-Feature Thresholding)

Для підвищення чутливості до атак на окремі сенсори запропоновано метод поознакового (per-feature) порогу. Кожній ознаці j призначається власний поріг τ_j , обчислений по розподілу $pf_j(t)$ на калібрувальній підмножині нормальних вікон тестового домену:

$$\tau_j^q = \text{Quantile}\{pf_j(t): t \in N_{cal}\}, \frac{q}{100}, \quad j = 1, \dots, F \quad (2.6)$$

Тривога виникає, якщо хоча б одна ознака перевищила свій поріг (правило OR):

$$a(t) = \begin{cases} 1, & \text{якщо } \exists j: pf_j(t) > \tau_j^q \\ 0, & \text{інакше} \end{cases} \quad (2.7)$$

Сирий прапор $a(t)$, сформований за правилом OR по 51 ознаці, є надмірно чутливим: будь-яке короткочасне перевищення власного порога хоча б однією ознакою, спричинене шумом сенсора, поодиноким викидом або мікродрейфом, миттєво породжує хибну тривогу. Для придушення нестабільності та високочастотних коливань сигналу (chattering) застосовано гістерезисну фільтрацію бінарного прапора за двома часовими параметрами: затримкою ввімкнення (hold-on, h_{on}) та затримкою вимкнення (hold-off, h_{off}).

Затримка ввімкнення вимагає, щоб прапор $a(t)$ утримувався у стані «1» неперервно протягом h_{on} кроків, перш ніж тривога активується:

$$b(t) = \begin{cases} 1, & \text{якщо } \sum_{i=0}^{h_{on}-1} a(t-i) = h_{on} \\ 0, & \text{інакше} \end{cases} \quad (2.8)$$

Завдяки умові (2.8) ізольовані сплески тривалістю менше h_{on} секунд відсікаються як шум — це основний механізм зниження кількості хибних тривог поознакового правила.

Затримка вимкнення утримує тривогу активною ще h_{off} кроків після останнього підтвердження, склеюючи близькі за часом спрацювання в одну неперервну подію. Хоча тривога й продовжується вперед у часі, її значення в момент t залежить лише від поточного та минулих підтверджень — без звернення до майбутніх відліків (look-ahead), що зберігає каузальність системи. Це досягається введенням рекурентного таймера залишкової активності $c(t)$:

$$c(t) = \begin{cases} h_{off}, & \text{якщо } b(t) = 1 \\ \max(0, c(t-1) - 1), & \text{якщо } b(t) = 0 \end{cases}$$

Підсумкове рішення про наявність аномалії $y(t)$ приймається, доки таймер не вичерпано:

$$y(t) = \begin{cases} 1, & \text{якщо } c(t) > 0 \\ 0, & \text{інакше} \end{cases} \quad (2.9)$$

Такий механізм є дискретним часовим аналогом гістерезису (тригера Шмітта): подія повинна бути достатньо тривалою, щоб бути зареєстрованою, і лише достатньо довга «тиша» гасить тривогу. На відміну від простого ковзного усереднення бінарного прапора, гістерезисна фільтрація зберігає миттєву реакцію на стійке відхилення і водночас ефективніше фільтрує поодинокі викиди, не розмиваючи межі події. Вплив параметра h_{off} на компроміс «повнота — хибні тривоги» досліджено експериментально у підрозділі 3.6.9.

2.4.4 Стратегія leak-free калібрування

Стандартний датасет SWaT-2015 складається з двох окремих файлів: «нормальний» (SWaT_Dataset_Normal_v1.xlsx, записи роботи за 22–28 грудня) та «атак» (SWaT_Dataset_Attack_v0.xlsx, записи за 28 грудня – 2 січня). Статистика нормальних ділянок у файлі атак систематично відрізняється від статистики тренувального нормального файлу — це відомий ефект дрейфу розподілу (distribution shift), що відзначається в роботах із критичного перегляду оцінювання детекторів аномалій на SWaT[18]. Внаслідок цього поріг τ , відкалібрований за перцентилем помилки на train-нормі, виявляється нижчим за рівень помилки реконструкції на нормальних ділянках тестового файлу, що генерує хибні тривоги на нормі.

Коректним рішенням буде калібрувати поріг по нормальних ділянках тестового домену. У реалізованій системі це досягається за допомогою дворівневої стратегії безвитокового (leak-free) калібрування, завдяки чому рівень хибних тривог (False Positives) контролюється за конструкцією на правильному домені.

На першому рівні конвеєра обробки даних усі зафіксовані часові інтервали атак класифікуються за масштабом впливу (класи SSSP, SSMP, MSSP, MSMP) і стратифіковано розподіляються між валідаційною (val) та тестовою (test) підмножинами за принципом один-через-один (1:1

interleaving). Це гарантує, що підбір порогових гіперпараметрів на валідації здійснюється на одних аномальних сценаріях, а фінальне тестування — на абсолютно інших, раніше не бачених системою атаках.

На другому рівні, з метою запобігання оптимістичному зсуву оцінок (in-sample bias), проміжки нормальної поведінки всередині тестового файлу також підлягають строгому розділенню на два непересічні масиви. Розщеплення реалізується за механізмом часового чергування (parity split): часові вікна штатного режиму з парними індексами утворюють калібрувальну підмножину (calibration), а вікна з непарними індексами — перевірочну підмножину (hold-out / evaluation).

Процедура налаштування та оцінювання системи формалізується наступним чином.

На етапі калібрування оптимальний робочий квантиль сітки GRID та індивідуальні поознакові пороги τ (або глобальний поріг τ) обчислюються виключно на основі розподілу помилок реконструкції у калібрувальній підмножині нормальних вікон, тобто на нормі тестового домену. При цьому вибір квантиля обмежується критерієм задоволення заданого бюджету хибних тривог ($\text{TARGET_FP} = 0,03$), а якість детекції контролюється за допомогою валідаційних атак. Завдяки цьому критерій автоматично штрафує пороги, що перетинають нормальну базову лінію тестового файлу.

На етап оцінювання фінальний розрахунок усіх підсумкових точкових метрик якості (Precision, Recall, F1-score), а також точного обсягу хибних тривог (FP) та пропусків атак (FN) здійснюється виключно на незалежній перевірочній підмножині (hold-out), яка об'єднує нормальні вікна з непарними індексами та тестову групу атак.

Така ізоляція інформації повністю виключає перенавчання порогових правил і гарантує, що зафіксований у роботі рівень ложних спрацювань та точність класифікації є об'єктивними показниками реальної узагальнювальної здатності детектора.

2.5 Критерії оцінки ефективності

Для всебічної оцінки якості системи детекції використовуються дві групи метрик: точкові (поточкові) та подієві. Перша група оцінює класифікацію кожної секунди незалежно, друга — виявлення атак як цілісних подій. Розмежування між групами є принциповим для задач виявлення аномалій у КІ: система може мати посередній $F1$ -score, але стовідсотковий Event Recall — і така конфігурація є прийнятнішою, ніж висока точність при пропуску кількох критичних атак.

Багато публікацій на датасеті SWaT використовують схему point-adjustment (PA): якщо система виявила хоча б одну точку всередині інтервалу атаки, усі точки цього інтервалу вважаються вірно класифікованими ТР. Дослідження Kim et al. (AAAI 2022) математично довели, що за умови використання такої схеми навіть випадковий оцінювач або найпростіший тривіальний детектор здатний досягти $F1 \approx 0,9$ [18].

Для забезпечення строгості та чесності обчислювального експерименту оцінювання ефективності розроблених систем виконується виключно за чесними поточковими метриками (point-wise) без використання процедури point-adjustment: кожен секундний вектор стану об'єкта аналізується та класифікується моделлю повністю незалежно. Фінальний розрахунок здійснюється на ізольованій перевіірчній підмножині (hold-out), яка не брала участі в калібруванні порогових правил.

2.5.1 Точкові метрики (Point-wise Metrics)

Для кожного часового кроку t порівнюється бінарна мітка $\hat{y}(t)$ з істинним значенням $y(t) \in \{0,1\}$. На основі матриці помилок визначаються три основні метрики:

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

2.5.2 Подієві метрики (Event-based Metrics)

Кожна атака A визначається як неперервний часовий інтервал $[start(A), end(A)]$, де $y(t) = 1$. Атака вважається виявленою, якщо система сформувала хоча б одну тривогу в межах цього інтервалу. Подієва повнота (Event Recall):

$$\text{EventRecall} = \frac{|\{A: \exists t \in [start(A), end(A)], \hat{y}(t) = 1\}|}{|\mathcal{A}_{test}|},$$

де $\mathcal{A}_{test}$ — множина атак тестової підмножини.

2.5.3 Затримка детекції (Detection Delay)

Затримка детекції $d(A)$ визначається як кількість кроків між початком атаки та першою тривоною:

$$d(A) = \min\{t \in [start(A), end(A)]: \hat{y}(t) = 1\} - start(A)$$

$$\bar{D} = (1/|A_{det}|) \sum_{A \in A_{det}} d(A),$$

де A_{det} — підмножина атак, що були виявлені.

Чим менше значення $d(A)$, тим раніше система здатна попередити оператора КІ. Стандарт ІЕС 62443-3-3 вимагає, щоб система моніторингу виявляла аномалію до моменту, коли автоматика безпеки більше не зможе запобігти пошкодженню обладнання.

2.6 Висновки до розділу 2

У другому розділі сформульовано та математично обґрунтовано всі ключові складові системи виявлення аномалій.

Метод ковзного вікна формалізовано як перетворення одновимірною часового ряду на тривимірний тензор, що дає змогу нейромережі враховувати динамічний контекст останніх $w = 20$ секунд роботи системи SWaT. Нормалізація Min-Max Scaling параметризована виключно по нормальних тренувальних даних, що гарантує незміщеність відносно аномальних значень.

У роботі досліджено три архітектури автоенкодерів із різними механізмами часового моделювання. Dense-AE реалізує стиснення через повнозв'язні шари без явного часового моделювання і є базовою лінією. 1D-CNN-AE (2.9–2.13) виявляє локальні часові патерни за допомогою ковзних ядер Conv1D та Conv1DTranspose. LSTM-AE використовує систему трьох ґейтів (забування, входу, виходу) для моделювання тривалих залежностей між підсистемами SWaT і є оптимальним вибором для виявлення поступових та координуваних атак.

Цільова функція MSE обрана через її чутливість до великих відхилень та еквівалентність максимізації правдоподібності за гаусівського шуму. Оптимізатор Adam забезпечує адаптивне масштабування кроку для параметрів із різними частотами оновлень.

Запропоновано та обґрунтовано два підходи до формування порогу детекції. Глобальний перцентильний поріг контролює рівень хибних тривог через вибір квантиля на нормальних даних. Поозначковий поріг (per-feature thresholding) усуває ефект «розмивання» локалізованих відхилень; двопорогова гістерезисна фільтрація (hold-on / hold-off) відсікає поодинокі викиди й об'єднує спрацювання в межах однієї атаки.» Описано проблему дрейфу розподілу та обґрунтовано стратегію leak-free калібрування.

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНА ПЕРЕВІРКА МЕТОДІВ ВИЯВЛЕННЯ АНОМАЛІЙ

У третьому розділі описано програмну реалізацію системи виявлення аномалій, теоретичні засади якої сформульовано у розділах 1 та 2, і наведено результати її експериментальної перевірки на еталонному наборі даних SWaT 2015. Послідовно розглянуто технологічний стек і середовище виконання, конвеєр попередньої обробки даних, стратегію формування leak-free вибірок, реалізацію чотирьох нейромережевих моделей (M1–M4), процедуру калібрування порогів детекції за бюджетом хибних тривог, а також усебічний аналіз отриманих метрик. Основну увагу приділено порівнянню точкових і подієвих показників якості та обґрунтуванню переваги поознакового (per-feature) порогового правила для виявлення локалізованих атак на окремі сенсори.

3.1 Програмне середовище та технологічний стек

Систему реалізовано мовою Python 3 у середовищі Google Colab з апаратним прискоренням на графічному процесорі NVIDIA Tesla T4 (16 ГБ відеопам'яті). Вибір хмарного середовища зумовлений потребою у GPU для навчання рекурентних мереж та зручністю відтворення експерименту. Програмний код організовано як єдиний обчислювальний нотбук, що виконується наскрізно (Run All) і поділений на функціональні розділи: завантаження, очищення, нормалізація, формування вибірок, навчання, оцінювання та візуалізація.

Технологічний стек складається з таких бібліотек: TensorFlow / Keras — побудова й навчання автоенкодерів; scikit-learn — масштабування ознак

(MinMaxScaler) та обчислення метрик класифікації; pandas і NumPy — оброблення табличних даних і векторні обчислення; Matplotlib — візуалізація результатів; joblib — серіалізація масштабувальника.

Відтворюваність результатів забезпечується фіксацією початкового значення генератора псевдовипадкових чисел $SEED = 42$ одночасно для NumPy та TensorFlow. Ключові гіперпараметри конвеєра винесено у блок конфігурації: довжина вікна $WINDOW = 20$, тривалість відсікання перехідного режиму $STAB_HOURS = 1,5$, крок вікон при навчанні $TRAIN_STRIDE = 2$, кількість епох $EPOCHS = 30$, розмір міні-батчу $BATCH = 128$.

3.2 Підготовка та попередня обробка даних

Вхідними даними є два файли стандартного датасету SWaT 2015: файл нормального режиму (SWaT_Dataset_Normal_v1.xlsx, період 22–28 грудня 2015 р.) — використовується для навчання, та файл з атаками (SWaT_Dataset_Attack_v0.xlsx, період 28 грудня 2015 р. – 2 січня 2016 р.) — використовується для тестування. Кожен запис відповідає одній секунді спостережень і містить значення 51 вимірювального та керуючого тегу ($F = 51$), а також мітку режиму Normal/Attack [4].

3.2.1 Завантаження та кешування

Функція завантаження враховує особливості формату SWaT: заголовок таблиці розміщено у другому рядку файла, а назви колонок містять зайві пробіли, які видаляються. Оскільки повторне читання великих xlsx-файлів є повільним, після першого завантаження дані кешуються у бінарний формат Parquet; за наявності кешу подальші запуски читають саме його. Часові

позначки приводяться до типу `datetime` за форматом «%d/%m/%Y %I:%M:%S %p», а рядки з некоректними позначками відкидаються.

3.2.2 Очищення даних

Сенсорні та актуаторні сигнали приводяться до числового типу. Дискретні стани актуаторів кодуються: «Active» $\rightarrow$ 1, «Inactive» $\rightarrow$ 0; службові маркери «Bad Input» замінюються пропусками. Пропущені значення заповнюються методом `forward-fill`, а залишкові пропуски на початку рядів — нулями. Мітка атаки перетворюється на бінарний вектор $y(t) \in \{0,1\}$.

3.2.3 Відсікання перехідного режиму

Реальна установка SWaT на початку запису нормального файлу перебуває у стані стабілізації технологічного процесу. Такі перехідні дані не відображають усталеної «нормальної поведінки» і спотворюють статистику масштабування та навчання. Тому перші `STAB_HOURS` = 1,5 години нормального режиму відрізаються від навчальної вибірки.

3.2.4 Нормалізація ознак

Усі ознаки масштабуються методом `Min-Max` у діапазон $[0, 1]$ за формулою (2.4). Принципово важливо, що параметри масштабування обчислюються виключно на навчальних нормальних даних і незмінно застосовуються до тестового файлу. Окремо обробляються константні на навчальній вибірці ознаки (розмах значень $\Delta_j < 10^{-9}$): для уникнення помилки ділення на нуль їхній розмах програмно прирівнюється до одиниці, завдяки чому після нормалізації вони гарантовано набувають нульових значень і не вносять хибного внеску в загальну помилку реконструкції.

3.3 Формування навчальної та тестової вибірок

3.3.1 Параметри датасету та розбивки

Основні параметри обчислювального експерименту наведено в табл. 3.1.

Таблиця 3.1 — Параметри експерименту

Параметр	Значення
Датасет	SWaT 2015 (iTrust, SUTD)
Записів у нормальному файлі (після відсікання 1,5 год)	489 600
Записів у файлі атак	449 919
Частота дискретизації	1 Гц (1 запис / с)
Кількість тегів (F)	51
Кількість сценаріїв атак у файлі	41 (36 із фізичним впливом)
Розмір train (нормальні вікна, stride=2)	244 791 вікон
Усього тестових вікон (stride=1)	449 900 (норма 395 316 + атаки 54 584)
Калібрування порога (норма, parity-половина)	197 658 вікон
Hold-out для оцінки FP (норма, друга parity-половина)	197 658 вікон
Атаки: val / test (поділ за подіями, стратифіковано)	10 920 / 43 664 вікон
Фінальна вибірка оцінювання (hold-out норма + test-атаки)	241 322 вікон
Параметри моделі Dense-AE (M1)	270 620 параметрів
Параметри моделі LSTM-AE (M2/M3)	78 579 параметрів
Параметри моделі 1D-CNN-AE (M4)	35 187 параметрів
Час навчання кожної моделі (T4 GPU)	≈ 5–15 хв
Апаратне забезпечення	NVIDIA Tesla T4, Google Colab

3.3.2 Виділення інтервалів атак

Бінарний вектор міток $y(t)$ тестового файла перетворюється на множину неперервних інтервалів атак. Кожна атака A подається як суцільний проміжок $[start(A), end(A)]$, у межах якого $y(t) = 1$. Виділені інтервали зіставляються з офіційним списком атак (List_of_attacks_Final.xlsx) за часом початку (з допуском ± 3 хв). Записи з позначкою «No Physical Impact» виключаються.

Для попереднього аналізу поведінки технологічного процесу на рисунок 3.1 наведено динаміку шести репрезентативних сенсорів різних стадій (MV-101, P-102, LIT-101, AIT-202, LIT-301, DPIT-301) протягом тестового періоду. Інтервали реальних атак виділено рожевим тлом. Видно, що під час окремих атак сигнали різко відхиляються від штатного циклічного патерну (зокрема стрибки рівнів LIT-101 та LIT-301, «полиці» сталих значень), тоді як інші атаки візуально майже не відрізняються від норми — це і є «стелс»-сценарії, виявлення яких становить основну складність задачі.

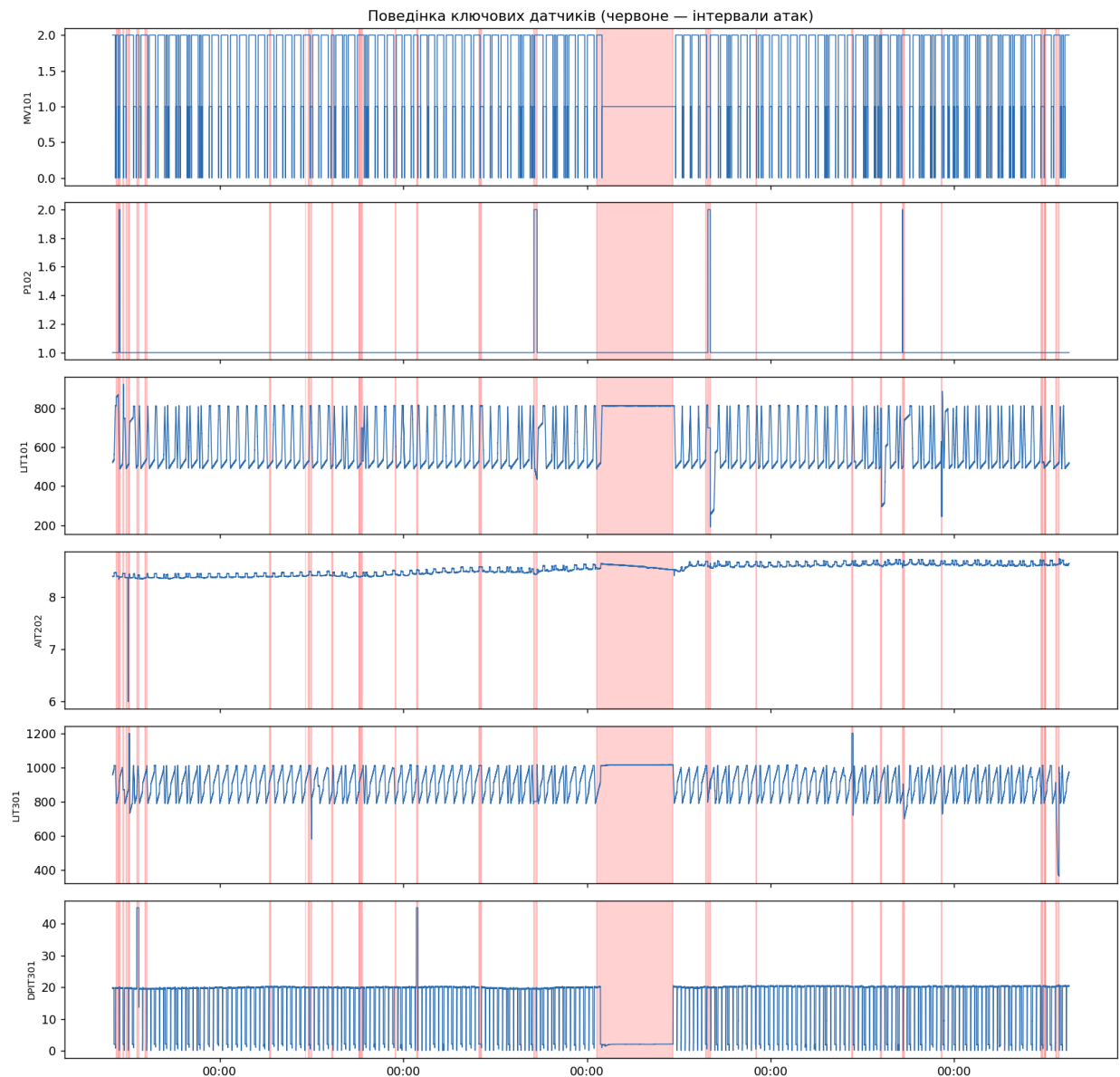


Рисунок 3.1 — Поведінка ключових сенсорів SWaT у тестовому періоді (рожевим позначено інтервали атак)

3.3.3 Класифікація атак за масштабом

Тип кожної атаки за масштабом узято з офіційного розмічування SWaT додатку до датасету, що виокремлює чотири класи: SSSP, SSMP, MSSP та MSMP відповідно до моделі підрозділу 1.2.3. Зокрема, до класу MSSP належать атаки a26, a27, a38 та a39.

3.3.4 Стратифікований leak-free спліт

Для коректного підбору гіперпараметрів без витоку інформації множина атак поділяється на дві непересічні підмножини — валідаційну (val) і тестову (test). Поділ виконується стратифіковано: у межах кожного класу атак (SSSP / SSMP / MSSP / MSMP) події по черзі розподіляються між val і test. Завдяки цьому обидві підмножини містять збалансований мікс типів атак, а підбір порогу на валідації не «бачить» тих самих атак, на яких обчислюються підсумкові метрики.

Унаслідок виявлення інтервалів у тестовому файлі та зіставлення з офіційним списком отримано 35 подій-інтервалів атак (суміжні інтервали a10+a11 та a27+a28 зливаються в один кожен). Тип кожної події взято з офіційного розмічування (підрозділ 3.3.3) та наведено у табл. 3.2. Підсумковий розподіл за підмножинами: валідаційна — 18 атак (SSSP=11, SSMP=3, MSSP=2, MSMP=2), тестова — 17 атак (SSSP=11, SSMP=3, MSSP=2, MSMP=1). Саме ці 17 тестових подій є знаменником подієвих метрик у підрозділі 3.6. Скорочення: Трив. — тривалість (хв:с, для тривалих атак — год:хв:с).

Таблиця 3.2 — Перелік атак SWaT, їх класифікація та розподіл між валідаційною і тестовою підмножинами

№	Атака	Атаковані вузли	Клас	Часовий інтервал	Трив.	Підмножина	Опис атаки (фізичний наслідок)
1	a1	MV-101	SSSP	10:29:14 – 10:44:53	15 хв 39 с	val	Відкриття клапана → переповнення резервуара T-101
2	a2	P-102	SSSP	10:51:08 – 10:58:30	7 хв 22 с	test	Увімкнення насоса → ризик розриву труби
3	a3	LIT-101	SSSP	11:22:00 – 11:28:22	6 хв 22 с	val	Підміна рівня → хибне осушення T-101
4	a4	MV-504	SSSP	11:47:39 – 11:54:08	6 хв 29 с	test	Відкриття клапана → зрив зупинки зворотного осмосу
5	a6	AIT-202	SSSP	12:00:55 – 12:04:10	3 хв 15 с	val	Заниження показника → вимкнення дозування P-203
6	a7	LIT-301	SSSP	12:08:25 – 12:15:33	7 хв 08 с	test	Завищення рівня → зупинка притоку, осушення T-301

Продовження таблиці 3.2

№	Атака	Атаковані вузли	Клас	Часовий інтервал	Трив.	Підмножина	Опис атаки (фізичний наслідок)
7	a8	DPIT-301	SSSP	13:10:10 – 13:26:13	16 хв 03 с	val	Завищення тиску → безперервний зворотний промив
8	a10	FIT-401	SSSP	14:16:20 – 14:28:20	12 хв 00 с	test	Заниження витрати → вимкнення системи дехлорування
9	a?	Невідомо	SSSP	06:30:00 – 06:42:00	12 хв 00 с	val	Недокументована атака, зафіксована у логах датасету
10	a13	MV-304	SSSP	11:11:25 – 11:15:17	3 хв 52 с	test	Закриття клапана → зупинка стадії ультрафільтрації
11	a14	MV-303	SSSP	11:35:40 – 11:42:50	7 хв 10 с	val	Блокування клапана → зупинка стадії ультрафільтрації
12	a16	LIT-301	SSSP	11:57:25 – 12:02:00	4 хв 35 с	test	Заниження рівня → переповнення Т-301
13	a17	MV-303	SSSP	14:38:12 – 14:50:08	11 хв 56 с	val	Блокування клапана → зупинка стадії ультрафільтрації
14	a19	AIT-504	SSSP	18:10:43 – 18:15:01	4 хв 18 с	test	Завищення провідності → хибна зупинка осмосу
15	a20	AIT-504	SSSP	18:15:43 – 18:22:17	6 хв 34 с	val	Критичне завищення провідності → зупинка осмосу
16	a21	MV-101, LIT-101	SSMP	18:30:00 – 18:42:00	12 хв 00 с	val	Утримання клапана + підміна рівня → переповнення
17	a22	UV-401, AIT-502, P-501	MSM P	22:55:18 – 23:03:00	7 хв 42 с	val	Складна атака на дехлорування та осмос → деградація
18	a23	P-602, DPIT-301, MV-302	MSM P	01:42:34 – 01:54:10	11 хв 36 с	test	Підміна тиску + утримання актуаторів → зависання системи
19	a24	P-203, P-205	SSMP	09:51:45 – 09:56:28	4 хв 43 с	test	Вимкнення дозувальних насосів → порушення хімії
20	a25	LIT-401, P-401	SSMP	10:01:50 – 10:12:01	10 хв 11 с	val	Підміна рівня + утримання насоса → осушення Т-401
21	a26	P-101, LIT-301	MSSP	17:04:56 – 17:29:00	24 хв 04 с	val	Утримання насоса + підміна рівня на різних стадіях
22	a27	P-302, LIT-401	MSSP	01:17:08 – 11:15:27	9 год 58 хв	test	Тривала зупинка притоку через утримання P-302 та LIT-401
23	a29	P-201, P-203, P-205	SSMP	15:32:00 – 15:34:00	2 хв 00 с	test	Одночасне увімкнення всіх дозувальних насосів
24	a30	LIT-101, P-101, MV-201	MSM P	15:47:40 – 16:07:10	19 хв 30 с	val	Утримання актуаторів + підміна рівня → збій притоку

Продовження таблиці 3.2

№	Атака	Атаковані вузли	Клас	Часовий інтервал	Трив.	Підмножина	Опис атаки (фізичний наслідок)
25	a31	LIT-401	SSSP	22:05:34 – 22:11:40	6 хв 06 с	test	Зниження рівня → переповнення резервуара T-401
26	a32	LIT-301	SSSP	10:36:00 – 10:46:00	10 хв 00 с	val	Завищення рівня → осушення та знос насоса P-302
27	a33	LIT-101	SSSP	14:21:12 – 14:28:35	7 хв 23 с	test	Завищення рівня → осушення та знос насоса P-101
28	a34	P-101	SSSP	17:12:40 – 17:14:20	1 хв 40 с	val	Вимкнення насоса → зупинка відтоку
29	a35	P-101, P-102	SSMP	17:18:56 – 17:26:56	8 хв 00 с	val	Вимкнення P-101 + утримання резервного P-102
30	a36	LIT-101	SSSP	22:16:01 – 22:25:00	8 хв 59 с	test	Зниження рівня → переповнення резервуара T-101
31	a37	P-501, FIT-502	SSMP	11:17:02 – 11:24:50	7 хв 48 с	test	Закриття насоса + підміна витратоміра → зниження виходу
32	a38	AIT-402, AIT-502	MSSP	11:31:38 – 11:36:18	4 хв 40 с	val	Завищення показників → хімічне передозування та скид
33	a39	FIT-401, AIT-502	MSSP	11:43:48 – 11:50:28	6 хв 40 с	test	Спроба вимкнення ультрафіолетової лампи
34	a40	FIT-401	SSSP	11:51:42 – 11:56:38	4 хв 56 с	val	Зниження витрати до нуля → спроба вимкнення UV
35	a41	LIT-301	SSSP	13:13:02 – 13:41:11	28 хв 09 с	test	Повільне зниження рівня → переповнення T-301

3.3.5 Побудова ковзних вікон

Згідно з формулами (2.1)–(2.3) одновимірні часові ряди перетворюються на тривимірні тензори ковзних вікон довжиною $WINDOW = 20$. Навчальні вікна будуються з кроком $TRAIN_STRIDE = 2$ (прорідження для прискорення), а тестові — з кроком 1, що забезпечує посекундну роздільну здатність детекції. Маски val_mask та $test_mask$ відносять кожне тестове вікно до нормального, валідаційного або тестового сегмента за індексом його останнього відліку.

3.4 Реалізація нейромережевих моделей

Реалізовано та порівняно чотири конфігурації детектора. Моделі M1, M2 і M4 використовують різні нейромережеві архітектури зі спільним глобальним пороговим правилом, а модель M3 застосовує ту саму навчену мережу, що й M2 (LSTM-AE), але з поознаковим порогом. Таке планування дозволяє оцінити вплив архітектури (M1 vs M2 vs M4) та вплив запропонованого правила детекції загалом (M2 vs M3).

M1 — Dense-AE повнозв'язний автоенкодер; вхідне вікно розгортається у вектор, стискається до латентної розмірності 32 й відновлюється. Базова лінія без явного часового моделювання.

M2 — LSTM-AE рекурентний автоенкодер. Кодувальник складається з двох рекурентних шарів LSTM (64 та 32 нейрони), які стискають вхідне вікно у латентний вектор розмірності 32. Цей вектор повторюється на всю довжину вікна та подається на декодувальник із двох шарів LSTM (32 і 64 нейрони) з вихідним повношаровим перетворенням, що відновлює послідовність вихідної розмірності F. Архітектура моделює часові залежності фізичних процесів.

M3 — LSTM-AE (per-feature) та сама мережа, що й M2, але з поознаковим порогом (формули 2.6 — 2.7) та двопороговим гістерезисним фільтром (hold-on / hold-off формули 2.8 — 2.9), що дозволяє реагувати на відхилення в окремому сенсорі й водночас придушувати шумові спрацювання.

M4 — 1D-CNN-AE згортковий автоенкодер із шарами Conv1D / Conv1Dtranspose. Виявляє локальні транзйєнтні патерни вздовж часової осі.

Усі моделі компілюються з оптимізатором Adam і цільовою функцією MSE (підрозділ 2.3). Навчання здійснюється виключно на нормальних даних із валідаційним розщепленням 10% і механізмом ранньої зупинки (EarlyStopping, моніторинг val_loss, терпіння 5 епох, відновлення найкращих wag).

Таке планування дозволяє оцінити вплив архітектури (M1 vs M2 vs M4) та вплив запропонованого правила детекції (M2 vs M3). Оскільки M3 відрізняється від M2 одночасно поозначковим порогом і гістерезисною фільтрацією, внесок кожного з цих двох чинників окремо ізолювано в абляційному дослідженні за схемою 2×2 (підрозділ 3.6.9), де всі чотири правила застосовано до однієї навченої мережі M2, таким чином єдиним змінним фактором лишається саме правило детекції, а не архітектура.

Динаміку навчання моделей ілюструють криві функції втрат (MSE) на навчальній і валідаційній вибірках, наведені на рисунку 3.2. Монотонне спадання обох кривих до виходу на плато без розходження свідчить про відсутність перенавчання.

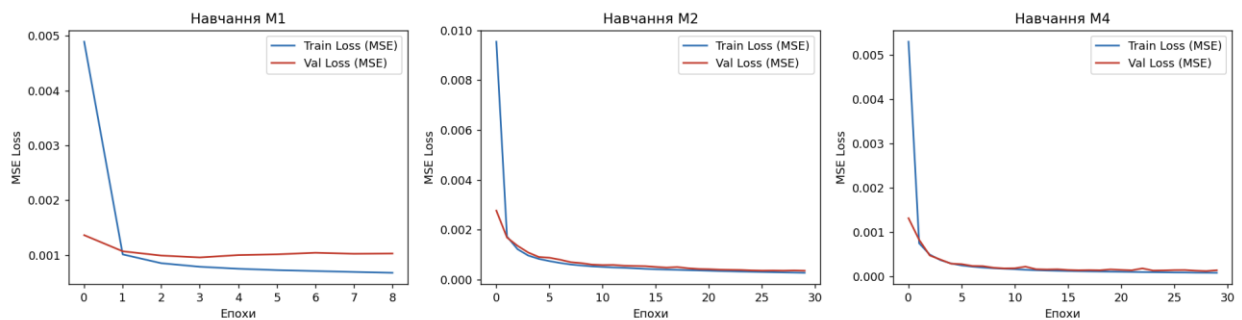


Рисунок 3.2 — Криві функції втрат (MSE) при навчанні моделей M1, M2 і M4: навчальна та валідаційна похибка за епохами

3.5 Обчислення показника аномальності та калібрування порогів

3.5.1 Помилки реконструкції

Для кожної навченої моделі обчислюються помилки реконструкції на навчальних і тестових вікнах. Тестові вікна обробляються порціями (predict із розміром порції 1024). Для кожного вікна обчислюються: глобальний

показник $g(t)$ — середня MAE по всіх ознаках і часових кроках вікна, та поознаковий вектор $pf_j(t)$ — середня MAE по часовій осі окремо для кожної ознаки. Глобальний показник згладжується ковзним середнім з вікном $k = 10$ без випередження.

3.5.2 Калібрування за бюджетом хибних тривог

У реалізованій системі поріг детекції калібрується за стратегією бюджету хибних тривог (false-alarm budget), що є практичним еквівалентом критерію (2.5). Перебирається сітка перцентилів $GRID = \{90, 95, 97, 98, 99, 99,5, 99,9, 99,95\}$. Для кожного значення обчислюються подієва повнота на валідаційних атаках та частка хибних тривог на калібрувальній підмножині нормальних вікон (перша parity-половина норми, див. підрозділ 2.4.4); обирається найнижчий перцентиль, за якого $FPR_{cal} \leq TARGET_FP = 0,03$.

3.5.3 Глобальний та поознаковий пороги

Для моделей $M1$, $M2$, $M4$ застосовується глобальний перцентильний поріг τ над згладженим показником $s(t)$, який фіксується на рівні квантилів $q \approx 98\%$ для задоволення заданого бюджету хибних тривог ($TARGET_FP = 0.03$). Для моделі $M3$ реалізовано поознакове правило: кожній ознаці j призначається власний поріг τ_j , і тривога-кандидат виникає, якщо хоча б одна ознака перевищила свій поріг (правило логічного «АБО», формула 2.7).

Варто наголосити, що агрегація за правилом «АБО» для 51 ознаки робить поознаковий підхід критично схильним до накопичення хибних спрацювань: експериментально встановлено, що при базовому квантілі $q=90\%$ частка хибних тривог на валідації сягає $FPR_{cal} = 0.877$. Внаслідок цього модель $M3$ змушена функціонувати в суттєво жорсткіших умовах: цільовий бюджет для неї досягається лише на гранично строгому квантілі $q=99.95\%$. Ця асиметрія

умов лише підкреслює структурну перевагу запропонованого методу: працюючи за значно консервативнішим бар'єром детекції ($q=99.95\%$ проти $\approx 98\%$), модель *M3* все одно забезпечує на тестовій вибірці вищу подієву повноту за одночасного зниження абсолютної кількості хибних тривог порівняно з глобальними аналогами.

Для придушення високочастотного коливання (chattering) сирого бінарного прапора, спричиненого високою чутливістю правила OR, його пропускають через двопороговий гістерезисний фільтр. Затримка ввімкнення $\text{hold_on} = 5$ с вимагає п'яти послідовних секунд безперервного підтвердження аномалії для активації тривоги (формула 2.8), що ефективно відсікає ізольовані шумові сплески. Затримка вимкнення $\text{hold_off} = 15$ с утримує тривогу активною, склеюючи близькі спрацювання в одну подію та запобігаючи дефрагментації сигналу (формула 2.9). За того самого гранично строгого квантиля $q = 99.95\%$ двопороговий фільтр знижує кількість хибних тривог із 5301 до 3034 (підрозділ 3.6.9), вилучаючи коливання (chattering), тоді як подієва повнота зменшується лише незначно. Гістерезисну фільтрацію реалізовано ковзними операціями rolling-sum та rolling-max без використання майбутніх відліків (look-ahead), що повністю зберігає каузальність системи та дозволяє її розгортання в режимі реального часу.

3.6 Результати експериментів та їх аналіз

Усі підсумкові метрики обчислено на незалежній тестовій підмножині атак, що не використовувалася під час калібрування порогів. На відміну від багатьох публікацій на датасеті SWaT, у цій роботі застосовано чесні точкові метрики без схеми point-adjustment (PA), яка, як зазначено у підрозділі 2.5, систематично завищує F1.

За такого оцінювання поознакова модель М3 із гістерезисною фільтрацією досягає точкового $F1 \approx 0,876$, тоді як моделі з глобальним порогом дають $F1 \approx 0,83$. Додатково, окрім точкових метрик, система вимірює затримку детекції та подієву повноту в розрізі типів атак — параметри, критичні для систем безпеки критичної інфраструктури.

3.6.1 Точкові та подієві метрики

Підсумкові значення метрик для всіх моделей на тестовій підмножині подано в табл. 3.3.

Таблиця 3.3 — Підсумкові метрики моделей на тестовій підмножині

Model	Precision	Recall	F1	EventRecall	DetectionDelay	FP	FN
M1 Dense-AE	0.895	0.769	0.827	0.353	680	3947	10107
M2 LSTM-AE	0.894	0.769	0.827	0.412	586	3960	10107
M3 LSTM-AE perfeat	0.923	0.834	0.876	0.588	306	3034	7242
M4 CNN-AE	0.895	0.767	0.826	0.353	685	3950	10160

Три моделі з глобальним порогом (M1, M2, M4) практично нерозрізнявані: їхній F1 зосереджений на рівні $\approx 0,827$. Натомість модель M3 із поознаковим порогом та фільтром гістерезису вийшла вперед за всіма точковими показниками одночасно — $precision = 0,923$, $recall = 0,834$ та $F1 = 0,876$ проти $\approx 0,827$ у глобальних моделей.

Подієві метрики підкреслюють розрив ще виразніше. Подієва повнота M3 (0,588) у 1,4 — 1,7 раза перевищує показники моделей із глобальним порогом (0,353–0,412), а середня затримка детекції M3 (≈ 306 с) приблизно вдвічі менша, ніж у решти (586–685 с). Принципово, що цей виграш у повноті досягнуто не ціною точності: завдяки гістерезисній фільтрації кількість хибних тривог M3 ($FP = 3034$) виявилася навіть нижчою, ніж у глобальних моделей. Двопорогова гістерезисна фільтрація усуває компроміс «повнота

проти точності» і зробить модель M3 домінантною за всіма метриками без винятку.

Висока точкова F1 усіх моделей пояснюється структурою тестового набору: вона формується переважно за рахунок однієї тривалої масштабної атаки, що дає тисячі вірно класифікованих аномальних секунд (TP). Натомість численні короткі атаки на окремі сенсори дають мало секунд кожна, тому їх пропуск майже не впливає на точкову F1 глобальних моделей, але різко знижує їхню подієву повноту.

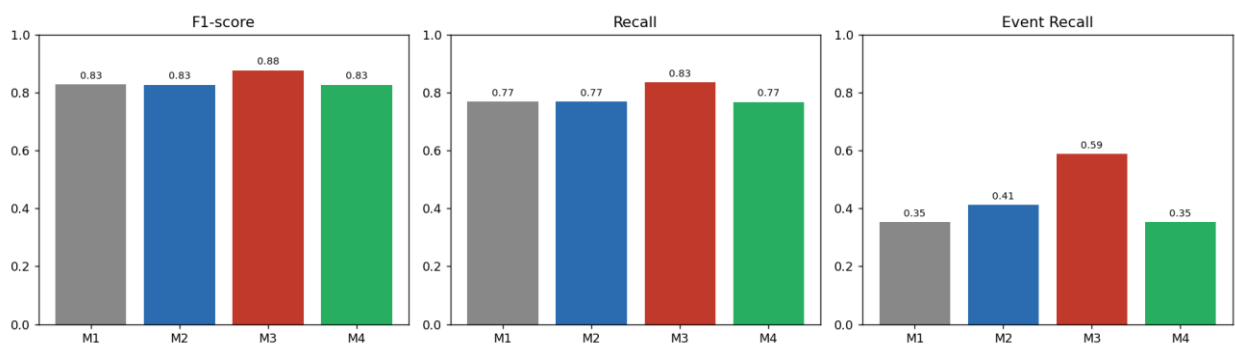


Рисунок 3.3 — Порівняння моделей за F1, точковою повнотою (Recall) та подієвою повнотою (Event Recall)

3.6.2 Подієва повнота за типами атак

Подієву повноту (Event Recall) у розрізі типів атак наведено в табл. 3.4.

Таблиця 3.4 — Event Recall у розрізі типів атак (тестова підмножина)

Model	MSMP	MSSP	SSMP	SSSP
M1 Dense-AE	0.0	1.0	0.333	0.273
M2 LSTM-AE	0.0	1.0	0.333	0.364
M3 LSTM-AE perfeat	1.0	1.0	0.667	0.455
M4 CNN-AE	0.0	1.0	0.333	0.273

Закономірність однозначна. Моделі з глобальним порогом (M1, M2, M4) повністю пропускають масштабні атаки класу MSMP (Event Recall = 0,0),

виявляють лише третину координованих атак SSMP (0,333) і слабо реагують на точкові атаки SSSP (0,27–0,36). Натомість модель M3 із поозначковим порогом виявляє всі MSMP-атаки (1,0), дві третини SSMP (0,667) і майже половину SSSP (0,455). Атаки класу MSSP виявляються всіма моделями (1,0): у тестовій підмножині вони представлені двома подіями (a27 та a39), наслідки яких через фізичні зв'язки поширюються на кілька стадій і тому надійно фіксуються навіть глобальним порогом.

Переважа M3 на класах MSMP та SSMP є прямим підтвердженням теоретичного аргументу підрозділу 2.4.3: індивідуальний поріг τ_j реагує на локалізоване відхилення окремого сенсора, яке глобальне усереднення розмиває до непомітності.

Порівняння подієвої повноти всіх чотирьох моделей у розрізі типів атак за масштабом наведено на рисунку 3.4

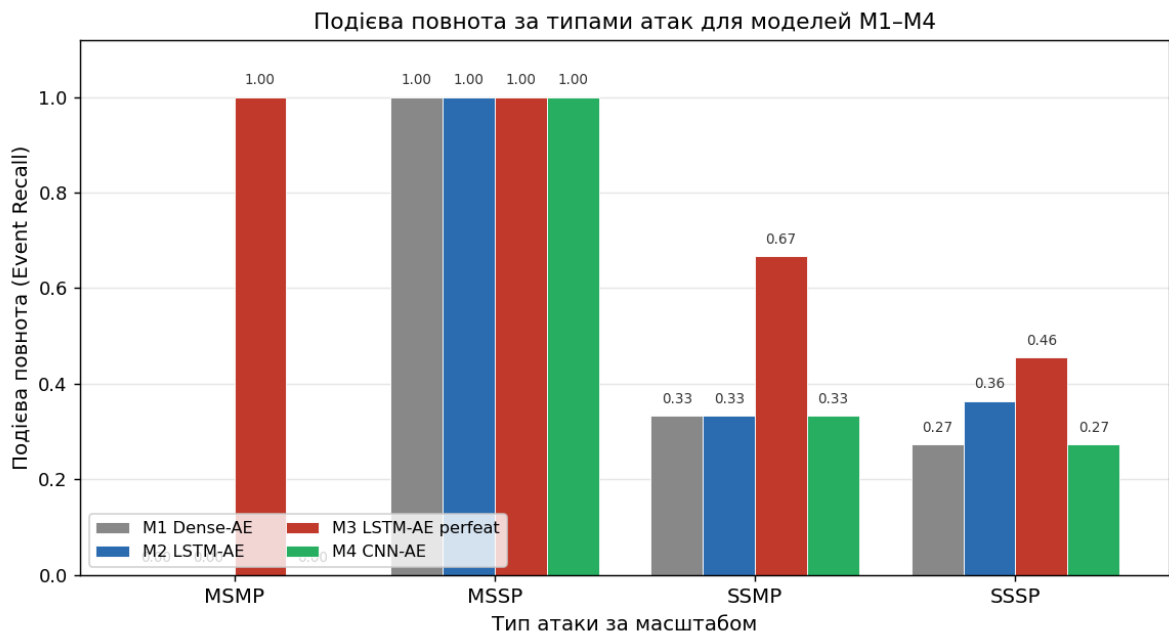


Рисунок 3.4 — Подієва повнота за типами атак для моделей M1–M4

3.6.3 Аналіз кривих ROC та Precision–Recall

Криві ROC і Precision–Recall (рисунку 3.5) побудовано для моделей із глобальним показником (M1, M2, M4) за неперервним згладженим MAE без

фіксованого порога. Площа під ROC-кривою становить 0,92 для M1, 0,91 для M2 та 0,91 для M4. Характерна особливість PR-кривої — високе плато точності (близько одиниці) до значення повноти $\approx 0,77$ із подальшим різким спадом: приблизно три чверті аномальних секунд впевнено відокремлюються від норми, а решта — «стелс»-сегменти, помилка реконструкції яких перекривається з нормальним діапазоном.

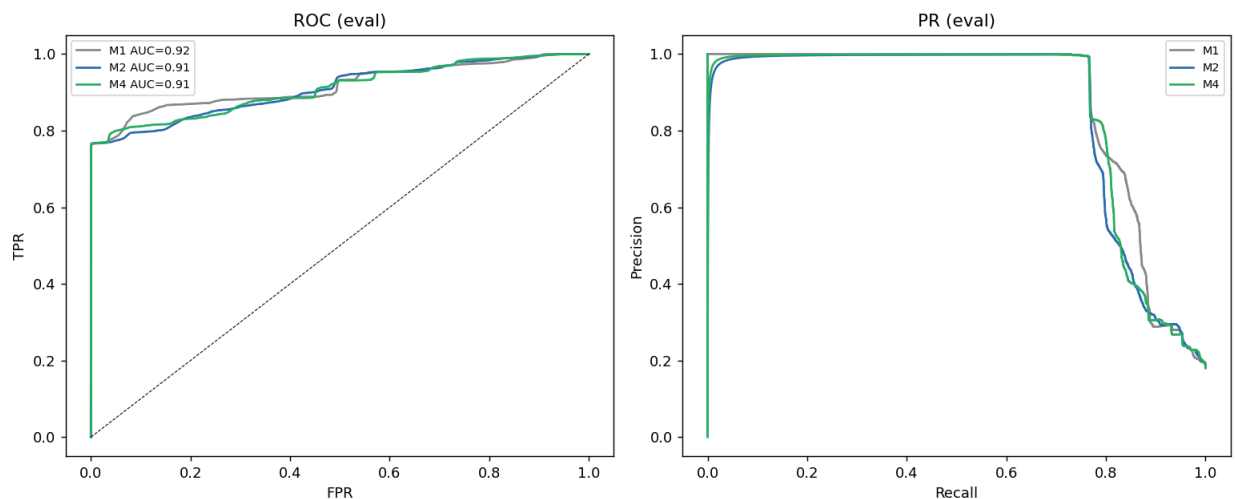


Рисунок 3.5 — Криві ROC (ліворуч) та Precision–Recall (праворуч) для тестової підмножини

3.6.4 Матриці помилок

Матриці помилок (рисунок 3.6) деталізують структуру похибок на рівні секунд. Моделі M1, M2 і M4 мають близьку кількість хибних тривог ($FP \approx 3947\text{--}3960$), оскільки їхні глобальні пороги калібровано за однаковим бюджетом $TARGET_FP = 0,03$ на спільній калібрувальній підмножині норми (усі три обрали квантиль $q = 98\%$), а оцінено — на спільній hold-out підмножині; невеликі розбіжності зумовлені відмінностями архітектур у ранжуванні нормальних вікон. Кількість пропусків у них також близька ($FN \approx 10,1$ тис.). Модель M3 демонструє одночасно найменшу кількість хибних тривог ($FP = 3034$) і найменшу кількість пропусків ($FN = 7242$)

серед усіх конфігурацій. Гістерезисна фільтрація відсікає поодинокі хибні спрацювання поознакового правила (зниження FP), а склеювання спрацювань у межах однієї атаки разом з індивідуальними порогами τ_j підвищує число вірно виявлених аномальних секунд ($TP \approx 36,4$ тис. проти $\approx 33,5$ тис. у глобальних моделей). Таким чином поознакова модель з гістерезисною фільтрацією не жертвує точністю заради повноти, а покращує обидва аспекти водночас — компроміс, оптимальний для систем безпеки КІ.

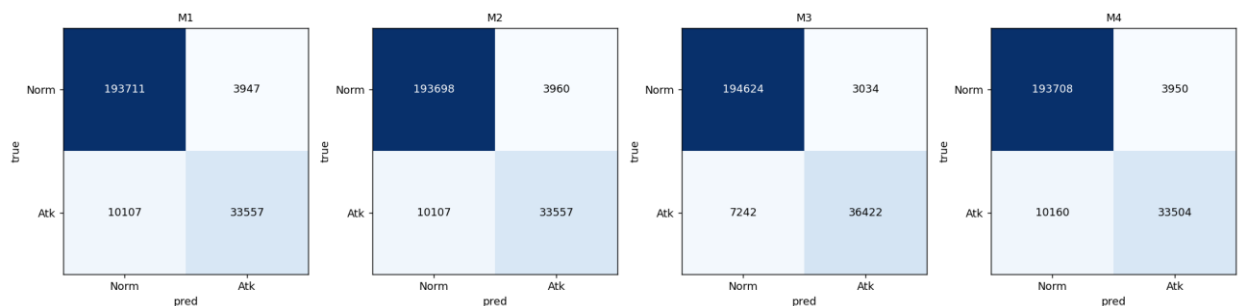


Рисунок 3.6 — Матриці помилок моделей M1–M4 на тестовій підмножині

3.6.5 Розподіл помилки реконструкції

Гістограма розподілу згладженого показника MAE для нормальних і аномальних вікон моделі M2 (рисунок 3.7) візуалізує фундаментальну складність задачі. Розподіл є істотно бімодальним. Основна маса і нормальних, і значної частини аномальних вікон зосереджена в області малих помилок ($MAE \approx 0-0,3$), де класи суттєво перекриваються — це «стелс»-атаки. Водночас виразно виокремлюється вузький пік аномальних вікон в області $MAE \approx 3,0$ — домінантна масштабна атака.

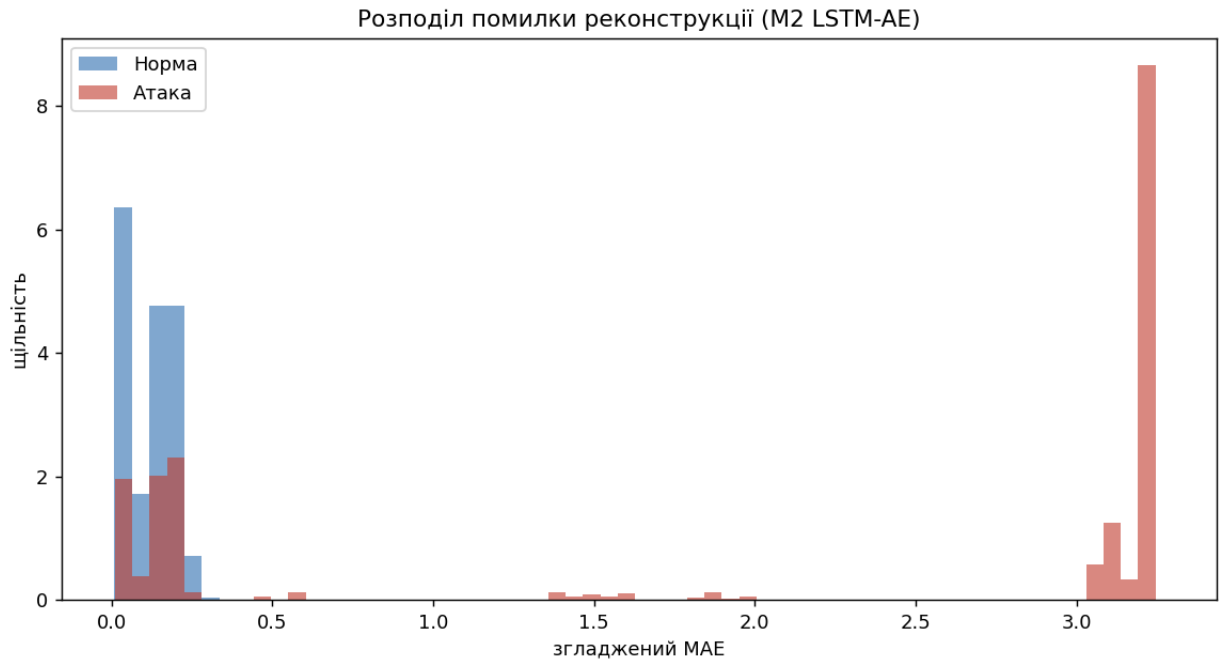


Рисунок 3.7 — Розподіл згладженої помилки реконструкції (M2 LSTM-AE): норма та атака

3.6.6 Теплова карта поознакової помилки та локалізація аномалій

Теплова карта поознакової помилки реконструкції M3 (рисунок 3.8) відображає нормовану помилку кожної з 51 ознаки (рядки) уздовж усіх тестових вікон (стовпці); інтервали реальних атак позначено маркерами зверху. Яскрава суцільна вертикальна смуга в центральній частині відповідає масштабній MSMP-атаці, що одночасно «підсвічує» багато ознак різних підсистем. Помітні горизонтальні «гарячі» рядки окремих сенсорів (AIT-202, AIT-402, PIT-501), помилка яких зростає локально — саме такі ізольовані відхилення глобальний поріг усереднює до непомітності, а поознаковий поріг M3 фіксує завдяки індивідуальному порогу τ_j .

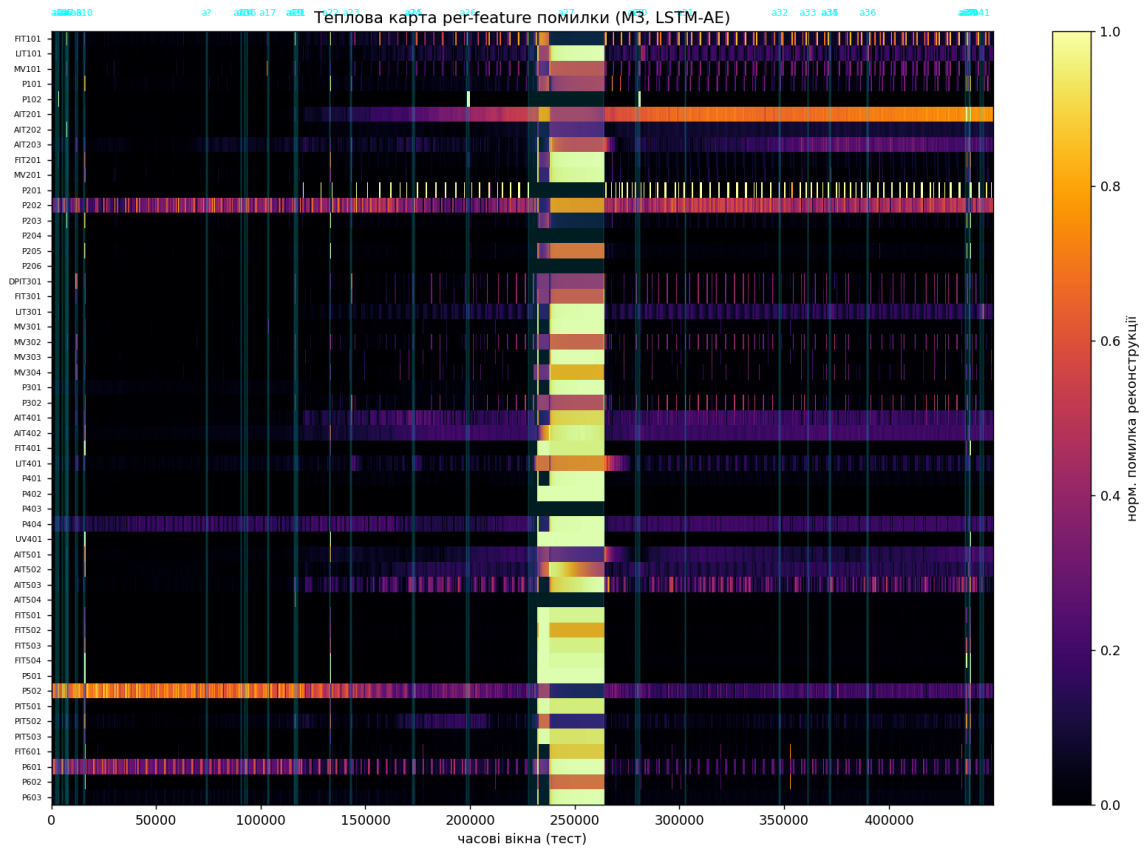


Рисунок 3.8 — Теплова карта поознакової помилки реконструкції (M3, LSTM-AE)

3.6.7 Часова шкала детекції

Часова шкала детекції (рисунок 3.9) зіставляє моменти спрацювання чотирьох моделей (рядки) з реальними інтервалами атак (затінені смуги). Рядок M3 містить найщільніше та найрівномірніше розподілені спрацювання, що покривають більшість затіненних інтервалів атак по всьому тестовому періоду. Рядки M1, M2 і M4 спрацьовують переважно в зоні домінантної масштабної атаки, залишаючи багато коротких атак невиявленими.

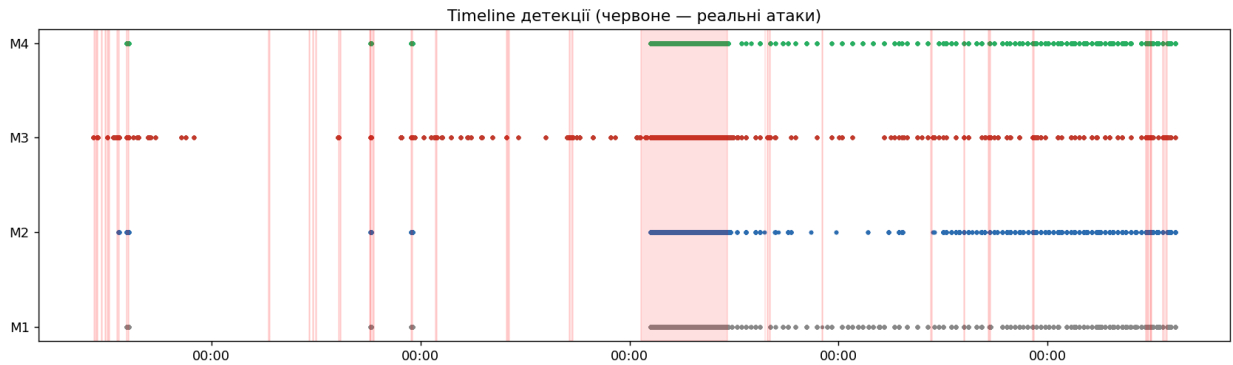


Рисунок 3.9 — Часова шкала детекції моделей M1–M4 (затінено — реальні інтервали атак)

3.6.8 Компроміс «повнота — хибні тривоги»

Для переконання у тому, що перевага M3 не є наслідком конкретного вибору бюджету TARGET_FP, побудовано криву компромісу між подієвою повнотою та кількістю хибних тривог для серії бюджетів {0,01; 0,02; 0,03; 0,05; 0,10} (рисунок 3.10). Крива моделі M3 лежить вище й лівіше за криві всіх інших моделей у всьому досліджуваному діапазоні: за будь-якої однакової кількості хибних тривог поознаковий поріг забезпечує вищу подієву повноту. Це означає, що перевага M3 є структурною й не зводиться до вдалого вибору бюджету хибних тривог: вона зберігається в усьому досліджуваному діапазоні TARGET_FP. Цю перевагу забезпечує поєднання поознакового порога з двопороговою гістерезисною фільтрацією. Як зазначено у підрозділ 3.6.9, поознаковий поріг підвищує подієву повноту (з 0,412 до 0,647), але ціною зростання хибних тривог (FP = 5301), тоді як гістерезис знижує їх до 3034 за незначного зниження повноти. Саме тому в моделі M3 обидва компоненти застосовано спільно: поознаковий поріг відповідає за чутливість, гістерезис — за придушення хибних спрацювань.

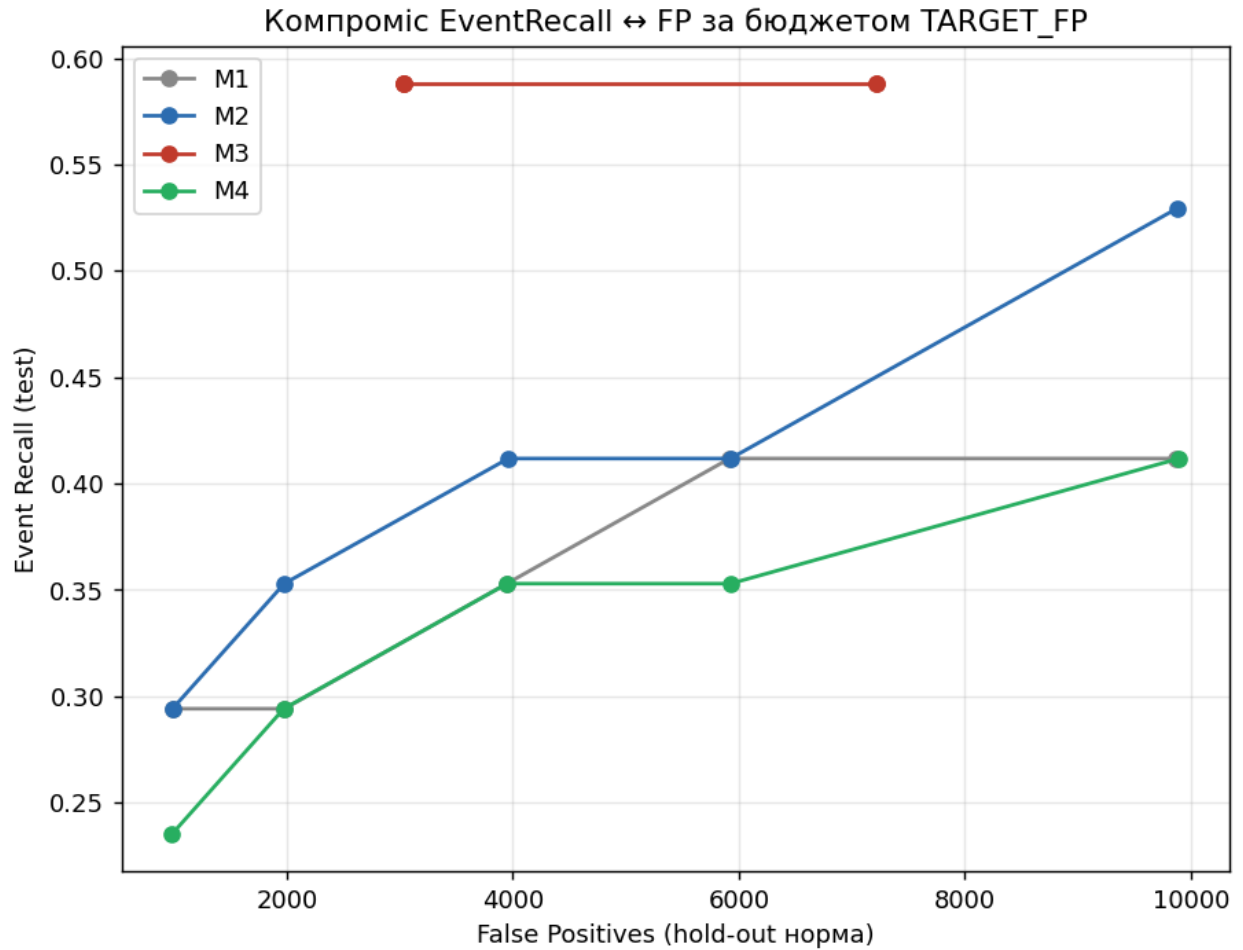


Рисунок 3.10 — Компроміс «Event Recall — кількість хибних тривог» за різних бюджетів TARGET_FP

3.6.9 Аналіз чутливості параметра гістерезисної фільтрації (Ablation Study)

Запропоноване правило детекції моделі M3 поєднує два незалежні механізми — поознаковий поріг і двопорогову гістерезисну фільтрацію. Щоб ізолювати внесок кожного з них, проведено абляційне дослідження за схемою 2×2: усі чотири комбінації (глобальний/поознаковий поріг × без/з гістерезисом) застосовано до однієї навченої мережі M2 (LSTM-AE), тож єдиним змінним фактором є саме правило детекції, а не архітектура. Результати наведено в таблиці 3.5.

Таблиця 3.5 — Внесок поознакового порога та гістерезису (мережа M2)

Rule	F1	Recall	EventRecall	FP
Global, без гістерезису	0.827	0.769	0.412	3960
Global, з гістерезисом	0.822	0.770	0.412	4509
Per-feature, без гістерезису	0.860	0.847	0.647	5301
Per-feature, з гістерезисом	0.876	0.834	0.588	3034

Перехід від глобального до поознакового порога (без гістерезису) підвищує подієву повноту з 0,412 до 0,647 і точковий F1 з 0,827 до 0,861 — отже, чутливість до локалізованих атак на окремі сенсори забезпечує саме поознаковий поріг, проте ціною зростання хибних тривог (3960 → 5301). Додавання гістерезису до поознакового правила знижує FP з 5301 до 3034 (43 %) і піднімає F1 до 0,876, жертвуючи незначною часткою подієвої повноти (0,647 → 0,588). На глобальному порозі гістерезис майже не впливає на метрики (F1 0,827 проти 0,822), оскільки згладжений глобальний показник уже стійкий до коливань. Отже, два компоненти є взаємодоповнювальними: поознаковий поріг відповідає за повноту, гістерезис — за точність, і саме їхнє поєднання дає найвищий F1 за найнижчого FP серед усіх конфігурацій.

Узагальнені результати ілюструє рисунок 3.11, де для кожної з чотирьох конфігурацій правила детекції зіставлено точкову F1-міру та кількість хибних тривог.

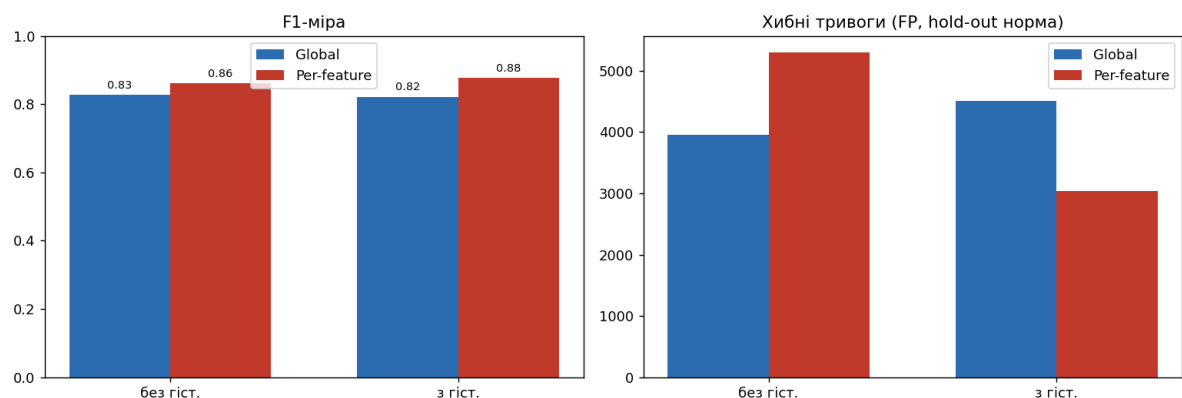


Рисунок 3.11 — Вплив поознакового порога та гістерезисної фільтрації на F1-міру та кількість хибних тривог: абляційна сітка 2×2 на мережі M2

Окремо перевірено стійкість моделі M3 до вибору параметра гістерезису. При фіксованій затримці ввімкнення $\text{hold_on} = 5$ с параметр hold_off послідовно набував значень $\{15, 30, 60\}$ с; для кожного значення поріг калібрувався заново за тим самим бюджетом $\text{TARGET_FP} = 0,03$, після чого обчислювалися F1 та кількість хибних тривог (рисунок 3.12).

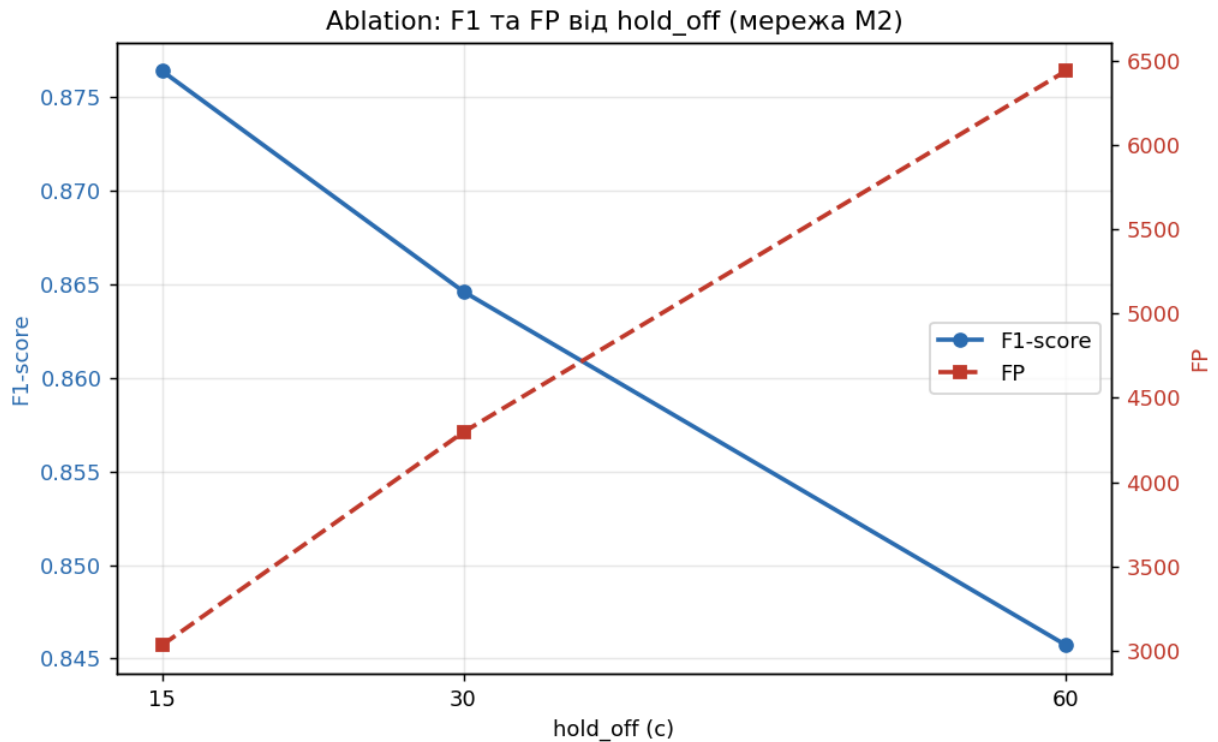


Рисунок 3.12 — Аналіз чутливості (ablation study) фільтра: залежність F1-міри та кількості хибних тривог (FP) від параметра hold_off

Збільшення hold_off подовжує «шлейф» утримання тривоги після згасання аномального сигналу. З одного боку, це склеює сусідні спрацювання, з іншого — тривога частіше «затікає» на нормальні ділянки одразу після завершення атаки, що збільшує кількість хибних тривог і знижує точкову F1 ($F1 = 0,876$ при $\text{hold_off} = 15$ с; $0,865$ при 30 с; $0,846$ при 60 с, тоді як FP зростає з 3034 до 6439). Обране робоче значення $\text{hold_off} = 15$ с забезпечує найкращий компроміс: подальше зростання параметра погіршує точність швидше, ніж покращує повноту. Монотонний і плавний характер залежності (рисунок 3.12)

підтверджує, що результати моделі M3 є стійкими щодо вибору цього гіперпараметра, а не артефактом єдиної вдалої конфігурації.

3.7 Обмеження та загрози достовірності результатів

Отримані результати слід інтерпретувати з урахуванням наступних обмежень. По-перше, дослідження проведено на одному датасеті (SWaT 2015); узагальнення на інші кіберфізичні системи (наприклад, BATADAL, HAI, SWaT 2021) потребує додаткової перевірки. По-друге, чутливість результатів до вибору бюджету хибних тривог TARGET_FP частково компенсована аналізом компромісної кривої (рисунок 3.10), а чутливість до параметра гістерезисної фільтрації — аналізом чутливості (рисунок 3.12), однак у виробничому середовищі оптимальні значення залежать від вартості FP у конкретному технологічному процесі. По-третє, рідкісні класи атак представлені у тестовій підмножині малою кількістю подій (1 подія MSMP, 2 події MSSP), тому оцінки подієвої повноти для цих класів (зокрема значення 1,0) спираються на малу вибірку й потребують обережної інтерпретації. По-четверте, введена гістерезисна фільтрація знижує чутливість до найкоротших точкових атак SSSP (умова hold-on відсікає спрацювання тривалістю менше п'яти секунд) — це свідомо плата за зниження рівня хибних тривог. По-п'яте, вартість обробки FP (час оператора, ризик «alarm fatigue») у роботі не оцінювалася. По-шосте, підсумкові метрики отримано за статичним порогом, відкаліброваним на репрезентативній (parity) підмножині нормальних вікон. Окрема проба з часовим поділом норми (калібрування на першій половині нормальних вікон тест-домену та застосування до другої) показала, що за такого сценарію частка хибних тривог зростає з 0,02 до 0,76 — прояв внутрішньофайлового дрейфу розподілу нормального режиму SWaT. Це означає, що у виробничому розгортанні статичний поріг потребує

періодичного онлайн-рекалібрування на свіжих нормальних даних; розроблення адаптивного механізму рекалібрування є напрямом подальших досліджень.

3.8 Збереження артефактів і можливість розгортання

Для практичного застосування детектора без повторного навчання реалізовано серіалізацію всіх артефактів системи. Зберігаються: ваги навчених мереж у форматі Keras (M1.keras, M2.keras, M4.keras; модель M3 використовує мережу M2 і окремого збереження не потребує), об'єкт масштабувальника (scaler.joblib) та конфігурація детектора у форматі JSON (detector_config.json), що містить довжину вікна, перелік ознак, глобальні пороги моделей, квантиль і поознакові пороги для M3 та параметр згладжування глобального показника (smooth_k = 10), параметри двопорогової гістерезисної фільтрації (hold-on = 5 с, hold-off = 15 с), бюджет хибних тривог (TARGET_FP = 0,03) і тип поділу норми. Завдяки цьому всі гіперпараметри правила детекції відтворюються із конфігурації без повторного калібрування. Навчену систему можна завантажити й застосувати до нового потоку даних SWaT в експлуатаційному режимі, обчислюючи показник аномальності у реальному часі з латентністю буферизації, що не перевищує довжину вікна (20 с) за конструкцією.

3.9 Висновки до розділу 3

У третьому розділі здійснено програмну реалізацію та повну експериментальну перевірку системи виявлення аномалій на еталонному наборі даних SWaT 2015.

Реалізовано наскрізний конвеєр обробки даних. Сформовано стратифікований leak-free поділ: атаки роз'єднано на валідаційну та тестову підмножини за типом, а нормальні вікна тест-домену поділено на калібрувальну та незалежну (hold-out) частини, тож пороги детекції калібруються й оцінюються на непересічних даних. Навчено та порівняно чотири конфігурації детектора (Dense-AE, LSTM-AE, LSTM-AE per-feature, 1D-CNN-AE).

Експериментально встановлено, що точкові метрики є недостатнім критерієм оцінювання детекторів аномалій у критичній інфраструктурі: моделі з глобальним порогом за пристойного $F1 \approx 0,83$ майже не реагують на численні короткі атаки (подієва повнота лише 0,353–0,412). Запропоноване правило детекції моделі M3 — поознаковий поріг у поєднанні з двопороговим гістерезисом ($\text{hold-on} = 5$ с, $\text{hold-off} = 15$ с) — стало ключовим удосконаленням: серед усіх чотирьох конфігурацій M3 одночасно досягла найвищого точкового $F1$ (0,876, що перевищує опубліковані для USAD значення 0,846 за чесного оцінювання без point-adjustment), найвищої подієвої повноти (0,588) та найнижчої кількості хибних тривог ($FP = 3034$), скоротивши середню затримку детекції приблизно вдвічі (≈ 306 с проти 586–685 с у глобальних моделях).

Абляційне дослідження за схемою 2×2 розділило внесок двох компонентів правила: поознаковий поріг відповідає за чутливість (підвищує подієву повноту), а гістерезис — за придушення хибних спрацювань; саме їхнє поєднання забезпечує найкращий баланс точності та повноти. Усі результати отримано без схеми point-adjustment, що робить їх об'єктивними та порівнянними з літературою [18]. Окрім точкових метрик, запропонований підхід вимірює затримку детекції та подієву повноту в розрізі типів атак — параметри, критичні для систем безпеки критичної інфраструктури.

Виявлено обмеження: одиночний датасет; мала кількість подій рідкісних класів (1 подія MSMP, 2 події MSSP) у тестовій підмножині; зниження чутливості до найкоротших атак SSSP як плата за гістерезисну фільтрацію;

зростання частки хибних тривог за часового дрейфу нормального режиму (з 0,02 до 0,76 у разі калібрування лише на ранніх даних), що мотивує онлайн-рекалібрування порога; невизначена вартість FP у виробничому середовищі. Реалізовано серіалізацію артефактів для розгортання системи в режимі реального часу.

РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У цьому розділі виконано техніко-економічне обґрунтування розробки інтелектуальної системи виявлення аномалій у телеметричних даних промислових кіберфізичних систем (КФС). Якщо попередні розділи відповідали на запитання «чи здатна система виявляти аномалії та наскільки якісно», то завдання цього розділу — відповісти на запитання «якою ціною досягається ця якість і чи є обрана сукупність технічних рішень економічно доцільною». Для об'єктивної відповіді застосовано метод функціонально-вартісного аналізу (ФВА).

Функціонально-вартісний аналіз — це метод системного дослідження об'єкта (виробу, процесу, програмного продукту), спрямований на оптимізацію співвідношення між споживчою якістю об'єкта та витратами на його створення. На відміну від суто технічної оцінки, ФВА розглядає продукт як сукупність функцій, кожна з яких має кілька можливих варіантів реалізації з різними характеристиками якості та вартості. Це дозволяє виявити та усунути зайві витрати, що не створюють додаткової споживчої цінності, і обрати таку комбінацію технічних рішень, яка забезпечує максимальний техніко-економічний рівень.

Аналіз проводиться поетапно: спочатку визначаються основні функції продукту та можливі варіанти їх виконання; далі формується система техніко-економічних параметрів і за допомогою експертного оцінювання встановлюється відносна вагомість кожного параметра; після цього розраховується коефіцієнт технічного рівня кожного варіанта реалізації; нарешті, виконується економічний розрахунок вартості розробки, і за інтегральним критерієм техніко-економічного рівня обирається оптимальний варіант.

4.1 Постановка задачі проектування

Метою проектування є створення програмного продукту для автоматизованого виявлення аномалій у телеметричних даних промислової кіберфізичної системи на прикладі еталонного стенду водоочищення SWaT 2015. Продукт призначений для застосування у складі систем моніторингу безпеки об'єктів критичної інфраструктури, де він має виконувати роль додаткового рівня захисту, здатного фіксувати відхилення технологічного процесу від штатної поведінки незалежно від сигнатур відомих атак.

Кінцевими користувачами продукту є оператори SCADA-систем, інженери з промислової кібербезпеки та аналітики центрів моніторингу (SOC), які потребують інструмента раннього попередження про потенційні кібератаки або несправності обладнання. Ключовою споживчою цінністю продукту є не стільки точність класифікації окремих секунд, скільки здатність своєчасно виявити атаку як цілісну подію та локалізувати її джерело — конкретний сенсор чи підсистему.

Основними функціональними вимогами до програмного продукту є:

- реалізація методів глибинного навчання (Dense-AE, LSTM-AE, 1D-CNN-AE) для виявлення аномалій без використання розмічених прикладів атак (навчання лише на нормальній поведінці);
- підтримка стратифікованого leak-free поділу даних на навчальну, валідаційну та тестову підмножини, що унеможливорює витік інформації між етапами;
- реалізація двох стратегій порогування, а саме глобального перцентильного та поознакового (per-feature) з гістерезисною фільтрацією;
- обчислення як точкових метрик якості (Precision, Recall, F1), так і подієвих (Event Recall, затримка детекції), що є критичними для систем безпеки;

- забезпечення відтворюваності результатів (фіксація початкового значення генератора випадкових чисел, автоматичне кешування даних);
- серіалізація навчених моделей та порогових параметрів для подальшого розгортання детектора в режимі реального часу без повторного навчання.

Сформульовані вимоги визначають набір технічних рішень, кожне з яких допускає кілька альтернатив. Саме ці альтернативи й становлять предмет подальшого функціонально-вартісного аналізу.

4.2 Обґрунтування функцій програмного продукту

На основі аналізу вимог до системи виявлення аномалій побудовано функціональну модель продукту. Головна (зовнішня) функція F0 формулюється як розробка програмного інструментарію для виявлення аномалій у часових рядах промислових КФС. Для її реалізації виділено чотири основні внутрішні підфункції, кожна з яких визначає окремий аспект технічної реалізації та безпосередньо впливає як на якість, так і на вартість продукту:

F1 — вибір обчислювального середовища для розробки та навчання моделей.

1. Використання спеціалізованих хмарних GPU-інстансів (наприклад, Google Colab / AWS). Забезпечує необхідну потужність для навчання складних рекурентних мереж, але потребує стабільного зв'язку та оренди потужностей(А).

2. Локальна розробка на стандартних корпоративних CPU-серверах. Усуває залежність від хмарних провайдерів, проте жорстко обмежує складність моделей, які можна навчити за адекватний час(Б).

F2 — вибір архітектури нейромережевої моделі (математичне ядро).

1. Глибока рекурентна архітектура (LSTM-AE). Здатна моделювати інерційність та часові залежності фізичних процесів водоочищення, виявляючи складні MSMP-атаки(А).

2. Базова архітектура (baseline) на основі повнозв'язного автоенкодера (Dense-AE). Використовується як швидкий і дешевий прототип (MVP) для відпрацювання конвеєра даних, але ігнорує часовий контекст системи(Б).

F3 — стратегія калібрування порогу детекції.

1. Поозначковий (per-feature) поріг із гістерезисною фільтрацією ($\text{hold_on} = 5 \text{ с}$, $\text{hold_off} = 15 \text{ с}$)(А).

2. Глобальний перцентильний поріг(Б).

F4 — оцінка якості та візуалізація результатів.

1. Повний набір точкових і подієвих метрик із тепловими картами, ROC/PR-кривими та часовою шкалою детекції(А).

2. Лише точкові метрики без візуалізації(Б).

Сукупність підфункцій F1–F4 та альтернативних варіантів їх виконання утворює морфологічну множину можливих конфігурацій продукту, яку подано у вигляді морфологічної карти на рисунку 4.1.

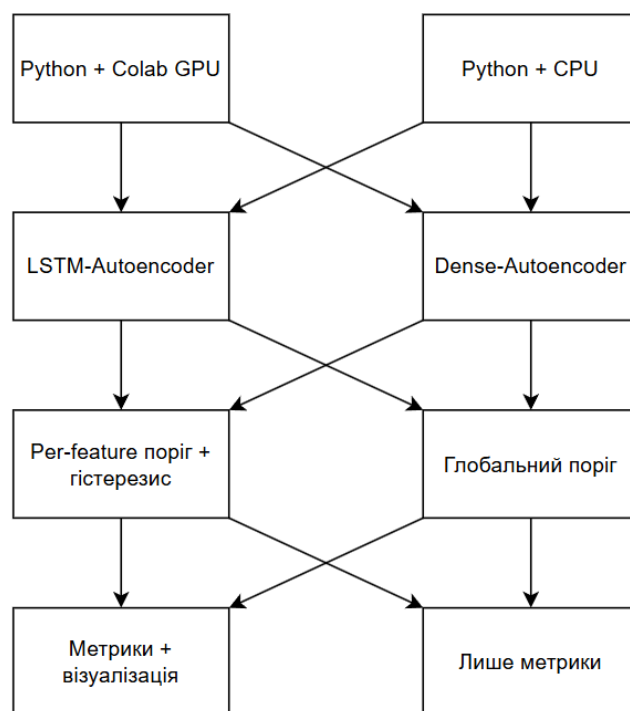


Рисунок 4.1 — Морфологічна карта

Для кожної підфункції існує по два варіанти виконання, що утворює морфологічну множину можливих конфігурацій продукту. Щоб скоротити цю множину до раціональних альтернатив, проведено якісний аналіз переваг і недоліків кожного варіанта за допомогою позитивно-негативної матриці (табл. 4.1).

Таблиця 4.1 — Позитивно-негативна матриця варіантів реалізації функцій

Функція	Варіант	Переваги	Недоліки
F1	А — Python + Colab GPU	Готова GPU-інфраструктура, хмарне зберігання, швидке навчання рекурентних мереж	Залежність від інтернет-з'єднання, ліміт часу GPU-сесії у безкоштовному режимі
	Б — Python + CPU	Автономність, відсутність залежності від хмари	Навчання рекурентних мереж займає занадто багато часу, що обмежує розробку лише найпростішими моделями.
F2	А — LSTM-AE	Моделює часові залежності між підсистемами, виявляє контекстні аномалії	Більший час навчання, складніша структура
	Б — Dense-AE	Простіша реалізація, швидше навчання	Придатний лише як baseline. Ігнорує часову послідовність процесів КІ, що призводить до пропуску критичних атак.
F3	А — Per-feature + гістерезисна фільтрація	Висока подієва повнота, виявлення атак на окремі сенсори, мала затримка	Складніша логіка калібрування
	Б — Глобальний поріг	Простіша реалізація	Низька подієва повнота; «розмиває» локалізовані відхилення
F4	А — Метрики + візуалізація	Повний аналіз поведінки системи, наочна локалізація аномалій	Додаткові витрати часу на модулі візуалізації
	Б — Лише метрики	Менший обсяг коду, простіше тестування	Відсутність наочного аналізу помилок

За результатами якісного аналізу для кожної підфункції обрано конкретне технічне рішення. Для F1 перевагу надано Python 3 у середовищі Google Colab з GPU T4: апаратне прискорення скорочує час навчання рекурентних мереж у десятки разів, а хмарне сховище забезпечує відтворюваність. Для F2 обрано LSTM-AE, оскільки рекурентна архітектура здатна моделювати часові залежності між 51 тегом системи й виявляти контекстні аномалії, недосяжні для повнозв'язної мережі. Для F3 обрано поознаковий поріг із гістерезисною фільтрацією — саме він, як показано в розділі 3, забезпечує найвищу подієву повноту за найменшої кількості хибних тривог. Для F4 обрано повний набір метрик із візуалізацією, оскільки теплові карти та часові шкали є критично важливими для інтерпретації роботи детектора оператором.

На основі проведеного аналізу для подальшого кількісного порівняння сформовано два цілісні варіанти реалізації, що відповідають двом стратегіям проектування — максимальної якості та мінімальної складності:

Варіант 1: F1(a) – F2(a) – F3(a) – F4(a) — Python + Colab GPU, LSTM-AE, поознаковий поріг з гістерезисною фільтрацією, повна візуалізація.

Варіант 2: F1(б) – F2(б) – F3(б) – F4(б) — Python + CPU, Dense-AE, глобальний поріг, лише метрики.

4.3 Обґрунтування системи параметрів програмного продукту

Для кількісного порівняння варіантів реалізації встановлено систему техніко-економічних параметрів, які найповніше характеризують споживчу якість продукту й водночас піддаються вимірюванню. Обрано чотири параметри.

X1 — подієва повнота (Event Recall, частки одиниці). Відображає здатність системи виявляти атаки як цілісні події, а не лише окремі секунди. Це головний цільовий показник для систем безпеки КІ: пропуск події означає невиявлену атаку. Бажаним є максимальне значення.

X2 — середня затримка детекції (с). Час між початком атаки та першою тривоною. Чим менша затримка, тим раніше оператор може відреагувати й тим менші потенційні наслідки. Бажаним є мінімальне значення.

X3 — обсяг оперативної пам'яті (МБ). Характеризує ресурсомісткість продукту під час роботи з матрицею ковзних вікон та проміжними тензорами помилок реконструкції. Впливає на вимоги до апаратного забезпечення. Бажаним є мінімальне значення.

X4 — час розробки (люд-год, відносна оцінка трудомісткості). Інтегрально відображає складність реалізації відповідної конфігурації. Бажаним є мінімальне значення.

Для кожного параметра встановлено три рівні значень — «гірше», «середнє» та «краще», які задають шкалу для подальшого бального оцінювання (табл. 4.2). Межі шкали для подієвої повноти (X1) визначено з урахуванням реалістично досяжних значень на датасеті SWaT за умови чесного оцінювання без схеми point-adjustment: верхня межа 0,65 відповідає практичному «стелі» сучасних методів, а не теоретичній одиниці.

Таблиця 4.2 — Основні техніко-економічні параметри продукт

Назва параметра	Позначення	Одиниці	Гірше	Середнє	Краще
Подієва повнота	X1	частки	0,25	0,45	0,65
Затримка детекції	X2	с	900	600	300
Витрати оперативної пам'яті	X3	МБ	3000	1800	800
Час розробки	X4	люд-год	1000	850	700

За даними таблиці 4.2 побудовано графічні залежності бальної оцінки кожного параметра від його абсолютного значення (рисунок 4.2 – рисунок 4.5). На кожному графіку краще значення параметра відповідає вищій бальній оцінці.

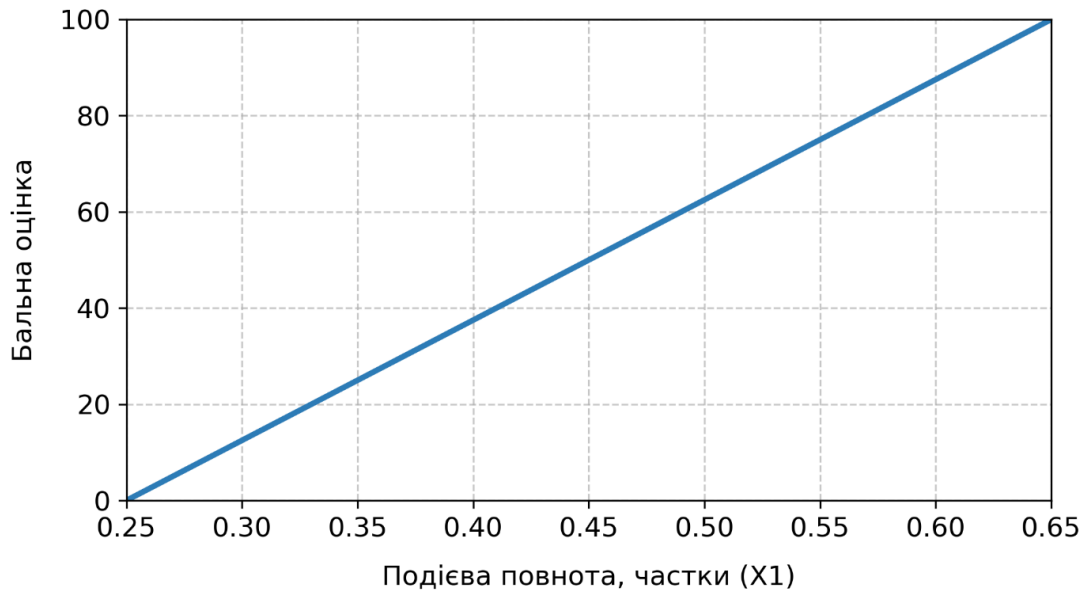


Рисунок 4.2 — X1, Подієва повнота

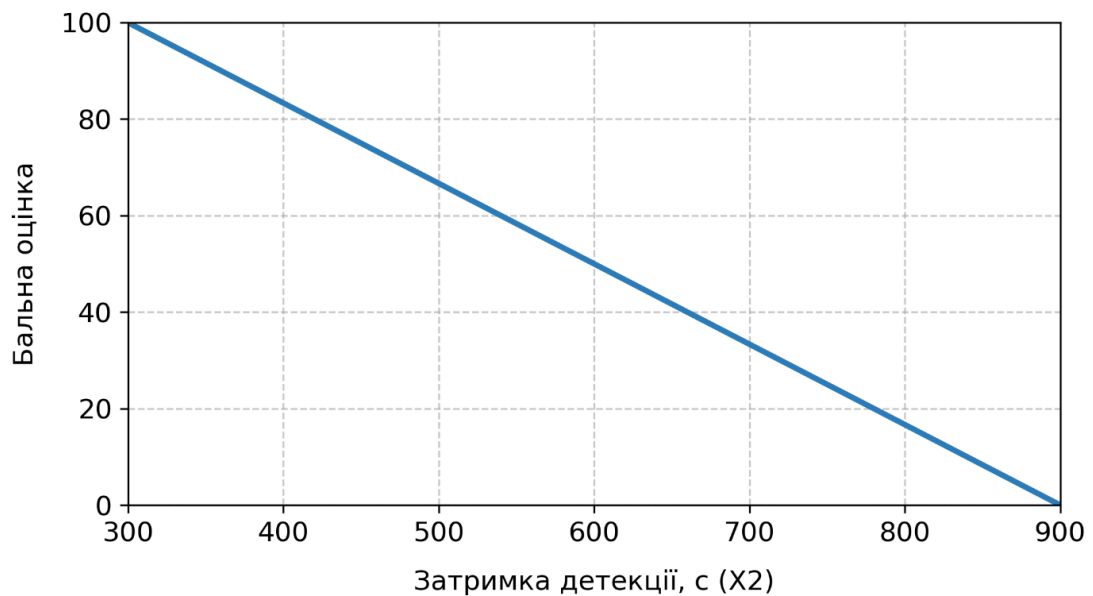


Рисунок 4.3 — X2, Затримка детекції

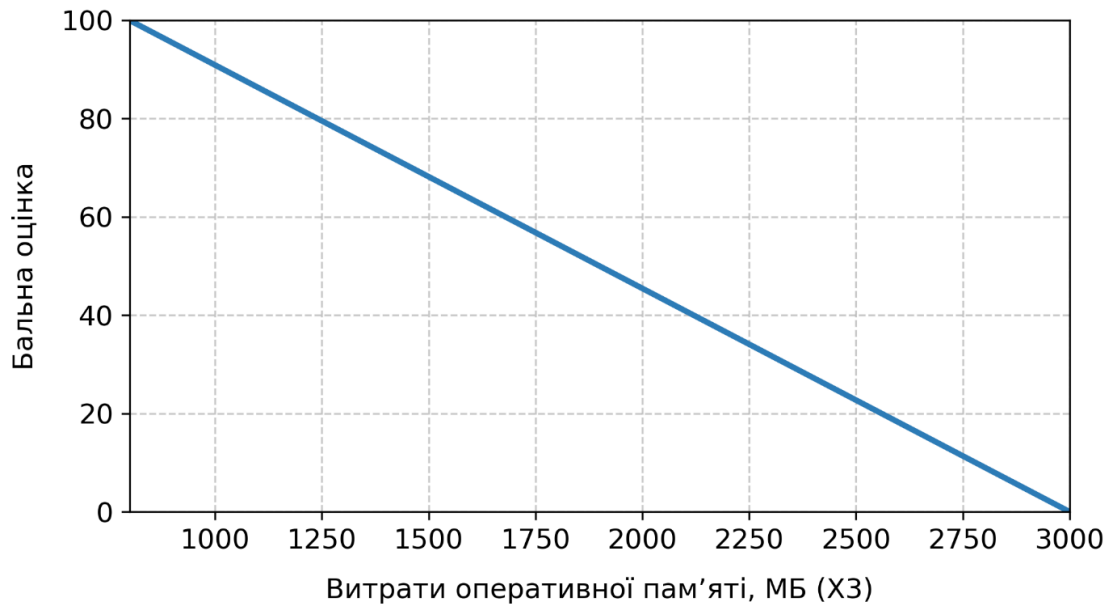


Рисунок 4.4 — X3, Витрати оперативної пам'яті

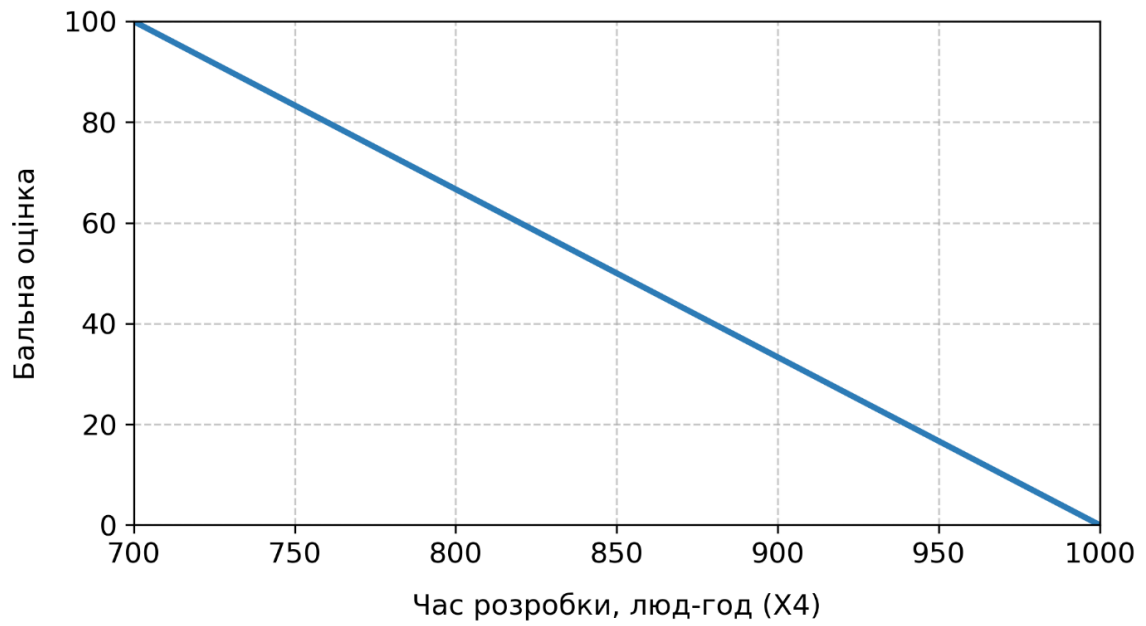


Рисунок 4.5 — X4, Час розробки

4.4 Аналіз експертного оцінювання параметрів

Оскільки обрані параметри мають різну важливість для кінцевого користувача, необхідно визначити їхню відносну вагомість. Для цього застосовано метод експертного ранжування. До оцінювання залучено експертну групу з $N = 7$ фахівців у галузі кібербезпеки промислових систем та методів машинного навчання. Кожен експерт незалежно проранжував чотири параметри за ступенем важливості (ранг 1 — найважливіший). Результати ранжування та проміжні розрахунки наведено в табл. 4.3.

Таблиця 4.3 — Результати ранжування параметрів експертами

Поз.	Параметр	1	2	3	4	5	6	7	ΣR_i	Δi	Δi^2
X1	Подієва повнота	1	1	2	1	1	2	1	9	-8,5	72,25
X2	Затримка детекції	2	2	1	2	2	1	2	12	-5,5	30,25
X3	Витрати пам'яті	3	4	4	4	3	4	4	26	8,5	72,25
X4	Час розробки	4	3	3	3	4	3	3	23	5,5	30,25
	Разом	10	10	10	10	10	10	10	70	0	205

Достовірність експертних оцінок перевіримо за допомогою коефіцієнта узгодженості (конкордації). Загальна сума рангів усіх експертів:

$$\sum R_i = \frac{N \cdot n \cdot (n + 1)}{2} = \frac{7 \cdot 4 \cdot 5}{2} = 70,$$

де $N = 7$ — кількість експертів;

$n = 4$ — кількість параметрів.

Середня сума рангів одного параметра:

$$T = \frac{1}{n} \cdot \sum R_i = \frac{70}{4} = 17,5.$$

Відхилення суми рангів кожного параметра від середньої:

$$\Delta_i = R_i - T.$$

$$\Delta_1 = 9 - 17,5 = -8,5; \quad \Delta_2 = 12 - 17,5 = -5,5;$$

$$\Delta_3 = 26 - 17,5 = 8,5; \quad \Delta_4 = 23 - 17,5 = 5,5.$$

Сума відхилень дорівнює нулю, що підтверджує коректність розрахунку. Загальна сума квадратів відхилень:

$$S = \sum \Delta_i^2 = 72,25 + 30,25 + 72,25 + 30,25 = 205.$$

Коефіцієнт узгодженості (конкордації) Кендалла:

$$W = \frac{12 \cdot S}{N^2 \cdot (n^3 - n)} = \frac{12 \cdot 205}{49 \cdot 60} = \frac{2460}{2940} \approx 0,837.$$

Отримане значення $W = 0,837$ перевищує нормативне $W_k = 0,67$, тому ступінь узгодженості думок експертів вважається достатнім, а результати ранжування — достовірними.

На основі узгоджених рангів проведено попарне порівняння параметрів. Числове значення переваги a_{ij} визначається за правилом:

$$a_{ij} = 1,5 \text{ при } X_i > X_j; \quad a_{ij} = 1,0 \text{ при } X_i = X_j; \quad a_{ij} = 0,5 \text{ при } X_i < X_j.$$

Результати попарного порівняння за оцінками експертів зведено в табл. 4.4.

Таблиця 4.4 — Попарне порівняння параметрів

Параметри	1	2	3	4	5	6	7	Оцінка	a_{ij}
X1 і X2	>	>	>	>	>	>	>	>	1,5
X1 і X3	>	>	>	>	>	>	>	>	1,5
X1 і X4	>	>	>	>	>	>	>	>	1,5
X2 і X3	>	>	>	>	>	>	>	>	1,5
X2 і X4	>	>	>	>	>	>	>	>	1,5
X3 і X4	<	>	>	>	<	>	>	>	1,5

За результатами попарного порівняння складено матрицю переваг $A = |a_{ij}|$ та обчислено вагомість кожного параметра. Вагомість визначається ітераційно за формулами:

$$K_{bi} = \frac{b_i}{\sum b_i}, \quad b_i = \sum a_{ij},$$

причому розрахунок повторюється доти, доки різниця коефіцієнтів вагомості між сусідніми ітераціями не стане меншою за 2 %. Результати трьох ітерацій наведено в табл. 4.5.

Таблиця 4.5 — Розрахунок коефіцієнтів вагомості параметрів

xi	X1	X2	X3	X4	bi	Kbi	bi¹	Kbi¹	bi²	Kbi²
X1	1	1,5	1,5	1,5	5,5	0,344	21,25	0,36	77,88	0,361
X2	0,5	1	1,5	1,5	4,5	0,281	16,25	0,275	59,13	0,274
X3	0,5	0,5	1	1,5	3,5	0,219	12,25	0,208	44,88	0,208
X4	0,5	0,5	0,5	1	2,5	0,156	9,25	0,157	34,13	0,157
Σ					16	1,0	59	1,0	216	1,0

Різниця між другою та третьою ітераціями не перевищує 2 %, тому остаточними приймаються значення третьої ітерації: $K_{bX1} = 0,361$; $K_{bX2} = 0,274$; $K_{bX3} = 0,208$; $K_{bX4} = 0,157$. Як і очікувалося, найвищу вагомість має подієва повнота (X1), що повністю відповідає пріоритетам систем безпеки критичної інфраструктури, де ціна пропущеної атаки є визначальною.

4.5 Аналіз рівня якості варіантів реалізації функцій

Інтегральний коефіцієнт технічного рівня (рівня якості) кожного варіанта реалізації обчислюється як зважена сума бальних оцінок параметрів:

$$K_{Kj} = \sum K_{Vi} \cdot B_i,$$

де K_{Vi} — коефіцієнт вагомості i -го параметра;

B_i — бальна оцінка i -го параметра за стобальною шкалою, отримана лінійною інтерполяцією абсолютного значення параметра між рівнями «гірше» (0 балів) та «краще» (100 балів) з табл. 4.2.

Абсолютні значення параметрів для обох варіантів узято з результатів експериментальної перевірки (розділ 3): для Варіанта 1 (поознаковий поріг з гістерезисною фільтрацією) подієва повнота становить 0,588, середня затримка детекції — 306 с; для Варіанта 2 (глобальний поріг) — 0,353 та 680 с відповідно. Розрахунок бальних оцінок і коефіцієнтів технічного рівня наведено в табл. 4.6.

Таблиця 4.6 — Розрахунок коефіцієнтів технічного рівня варіантів

Варіант	Параметр	Абс. знач.	Бал B_i	K_{Vi}	$K_{Vi} \cdot B_i$
Варіант 1	X1	0,588	85	0,361	30,69
	X2	306 с	99	0,274	27,13
	X3	1800 МБ	55	0,208	11,44
	X4	935 люд-год	22	0,157	3,45
Варіант 2	X1	0,353	26	0,361	9,39
	X2	680 с	37	0,274	10,14
	X3	800 МБ	100	0,208	20,8
	X4	725 люд-год	92	0,157	14,44

Підсумовуючи внески всіх параметрів, отримуємо коефіцієнти технічного рівня:

$$K_{K1} = 30,69 + 27,13 + 11,44 + 3,45 = 72,71$$

$$K_{K2} = 9,39 + 10,14 + 20,80 + 14,44 = 54,77$$

Варіант 1 має суттєво вищий коефіцієнт технічного рівня (72,71 проти 54,77). Хоча Варіант 2 переважає за ресурсомісткістю та простотою

(параметри X_3 , X_4), ці параметри мають низьку сумарну вагомість (0,365). Натомість Варіант 1 домінує за двома найвагомішими параметрами — подієвою повнотою та затримкою детекції (сумарна вагомість 0,635), які безпосередньо визначають захисну цінність системи. Отже, за технічним рівнем Варіант 1 переважає Варіант 2 у 1,33 рази.

4.6 Економічний аналіз варіантів розробки програмного продукту

Для визначення вартості розробки спочатку розраховується трудомісткість реалізації кожного варіанта. Розробка обох варіантів охоплює чотири завдання:

- 1) підготовка та обробка даних SWaT — завантаження, очищення, нормалізація, формування ковзних вікон;
- 2) реалізація нейромережових моделей та їх навчання — побудова архітектур, функція втрат, оптимізатор;
- 3) калібрування порогів та обчислення метрик — leak-free стратегія, поріг детекції, гістерезисна фільтрація;
- 4) візуалізація результатів і тестування — побудова графіків, теплових карт, часових шкал.

Принципово важливо, що завдання 1 (підготовка даних) виконується ідентично для обох варіантів, оскільки обидва опрацьовують той самий датасет SWaT за однаковим конвеєром попередньої обробки. Тому трудомісткість цього завдання для Варіанта 2 не зменшується. Відмінності в трудомісткості виникають лише у завданнях 2–4: Варіант 2 використовує простішу архітектуру Dense-AE, глобальний поріг без гістерезису та не містить модулів візуалізації.

Загальна трудомісткість кожного завдання обчислюється за формулою:

$$TO = TP \cdot KP \cdot KCK \cdot KM \cdot KCT \cdot KCTM,$$

де TP — базова трудомісткість;

KP — поправочний коефіцієнт новизни та складності;

$KCK = 1$ — коефіцієнт складності контролю вхідної інформації;

$KM = 0,85$ — коефіцієнт рівня мови програмування (для Python);

$KCT = 0,8$ — коефіцієнт використання стандартних модулів;

$KCTM = 1$ — коефіцієнт стандартного математичного забезпечення.

Для Варіанта 1 розрахунок трудомісткості за завданнями:

$$T_1 = 30 \cdot 1,2 \cdot 1 \cdot 0,85 \cdot 0,8 \cdot 1 = 24,48 \text{ люд-дні}$$

$$T_2 = 46 \cdot 1,4 \cdot 1 \cdot 0,85 \cdot 0,8 \cdot 1 = 43,79 \text{ люд-дні}$$

$$T_3 = 38 \cdot 1,3 \cdot 1 \cdot 0,85 \cdot 0,8 \cdot 1 = 33,59 \text{ люд-дні}$$

$$T_4 = 22 \cdot 1,0 \cdot 1 \cdot 0,85 \cdot 0,8 \cdot 1 = 14,96 \text{ люд-дні}$$

$$T_I = (24,48 + 43,79 + 33,59 + 14,96) \cdot 8 = 116,82 \cdot 8 = 934,56 \text{ люд-год.}$$

Для Варіанта 2 завдання 1 збігається з Варіантом 1 (24,48 люд-дні), а завдання 2–4 мають меншу базову трудомісткість через спрощену архітектуру, глобальний поріг і відсутність візуалізації:

$$T_1 = 24,48 \text{ люд-дні (ідентично Варіанту 1)}$$

$$T_2 = 42 \cdot 1,3 \cdot 1 \cdot 0,85 \cdot 0,8 \cdot 1 = 37,13 \text{ люд-дні}$$

$$T_3 = 26 \cdot 1,1 \cdot 1 \cdot 0,85 \cdot 0,8 \cdot 1 = 19,45 \text{ люд-дні}$$

$$T_4 = 14 \cdot 1,0 \cdot 1 \cdot 0,85 \cdot 0,8 \cdot 1 = 9,52 \text{ люд-дні}$$

$$T_{II} = (24,48 + 37,13 + 19,45 + 9,52) \cdot 8 = 90,58 \cdot 8 = 724,64 \text{ люд-год.}$$

Розробка програмного продукту виконується Data Science інженером (аналітиком даних) із середньомісячним окладом 25 000 грн. Середня погодинна ставка визначається за формулою:

$$СЧ = \frac{M}{T_m \cdot t},$$

де M — середньомісячний оклад розробника;

$T_m = 21$ — кількість робочих днів на місяць;

$t = 8$ — кількість робочих годин на день.

Отже:

$$СЧ = \frac{25000}{21 \cdot 8} = \frac{25000}{168} = 148,80 \text{ грн/год.}$$

Основна заробітна плата з урахуванням додаткової (коефіцієнт $КД = 1,2$) розраховується як:

$$СЗП = СЧ \cdot T_i \cdot КД.$$

$$\text{Варіант 1: } СЗП = 148,80 \cdot 934,56 \cdot 1,2 = 166875,03 \text{ грн}$$

$$\text{Варіант 2: } СЗП = 148,80 \cdot 724,64 \cdot 1,2 = 129391,72 \text{ грн}$$

Відрахування на єдиний соціальний внесок ($ЄСВ = 22\%$ від заробітної плати):

$$\text{Варіант 1: } СВІД = 166875,03 \cdot 0,22 = 36712,51 \text{ грн}$$

$$\text{Варіант 2: } СВІД = 129391,72 \cdot 0,22 = 28466,18 \text{ грн}$$

Для написання коду, попередньої обробки даних та підключення до хмарних інстансів Google Colab використовується локальна робоча станція (ноутбук) розробника. Тому до розрахунку включаються витрати на її амортизацію та електроенергію. Безкоштовного рівня доступу Google Colab виявилось достатньо для виконання навчальних ітерацій, тому витрати на підписку не включалися.

Далі визначимо собівартість однієї машино-години роботи обчислювальної техніки. Інфраструктуру (робочу станцію та доступ до

хмарних сервісів) обслуговує системний адміністратор з окладом 15 000 грн за коефіцієнта зайнятості $KЗ = 0,2$. Річний фонд оплати обслуговування:

$$CG = 12 \cdot 15000 \cdot 0,2 = 36000 \text{ грн}$$

$$CЗП_{ЕОМ} = 36000 \cdot 1,2 = 43200 \text{ грн}$$

$$CВІД_{ЕОМ} = 43200 \cdot 0,22 = 9504 \text{ грн}$$

Амортизаційні відрахування за норми амортизації 25 %:

$$CA = KTM \cdot KA \cdot ЦПР = 1,15 \cdot 0,25 \cdot 40000 = 11500 \text{ грн}$$

де $KTM = 1,15$ — коефіцієнт транспортування та монтажу;

$KA = 0,25$ — річна норма амортизації;

$ЦПР = 40\,000$ грн — ціна обладнання.

Витрати на ремонт і профілактику:

$$CP = KTM \cdot ЦПР \cdot КР = 1,15 \cdot 40000 \cdot 0,05 = 2300 \text{ грн.}$$

Ефективний річний фонд часу роботи ПК:

$$TEФ = (365 - 104 - 11 - 8) \cdot 8 \cdot 0,85 = 242 \cdot 8 \cdot 0,85 = 1645,6 \text{ год.}$$

Витрати на електроенергію (тариф 13,64 грн/кВт·год, потужність 0,075 кВт):

$$СЕЛ = 1645,6 \cdot 0,075 \cdot 0,2 \cdot 13,64 = 336,69 \text{ грн.}$$

Накладні витрати на експлуатацію ЕОМ:

$$CH(EOM) = 40000 \cdot 0,67 = 26800 \text{ грн.}$$

Сумарні річні експлуатаційні витрати:

$$C_{EKC} = 43200 + 9504 + 11500 + 2300 + 336,69 + 26800 = 93640,69 \text{ грн.}$$

Собівартість однієї машино-години:

$$C_{M-Г} = \frac{C_{EKC}}{TEФ} = \frac{93640,69}{1645,6} = 56,90 \text{ грн/год.}$$

Витрати на машинний час для кожного варіанта ($CM = CM-Г \cdot Ti$):

$$\text{Варіант 1: } CM = 56,90 \cdot 934,56 = 53176,46 \text{ грн}$$

$$\text{Варіант 2: } CM = 56,90 \cdot 724,64 = 41232,02 \text{ грн}$$

Накладні витрати на розробку складають 67 % від основної заробітної плати $CH = CZП \cdot 0,67$:

$$\text{Варіант 1: } CH = 166875,03 \cdot 0,67 = 111806,27 \text{ грн}$$

$$\text{Варіант 2: } CH = 129391,72 \cdot 0,67 = 86692,45 \text{ грн}$$

Повна вартість розробки програмного продукту визначається як сума заробітної плати, відрахувань, витрат на машинний час і накладних витрат:

$$C = CZП + СВІД + CM + CH$$

$$\text{Варіант 1: } C_1 = 166875,03 + 36712,51 + 53176,46 + 111806,27 = 368570,27 \text{ грн;}$$

$$\text{Варіант 2: } C_2 = 129391,72 + 28466,18 + 41232,02 + 86692,45 = 285782,37 \text{ грн}$$

Таким чином, повна вартість розробки Варіанта 1 становить 368 570,27 грн, Варіанта 2 — 285 782,37 грн. Варіант 1 дорожчий приблизно на 29 %, що зумовлено вищою трудомісткістю реалізації поознакового порогу з гістерезисною фільтрацією та модулів візуалізації.

4.7 Вибір кращого варіанту за техніко-економічним рівнем

Остаточний вибір оптимального варіанта здійснюється за інтегральним коефіцієнтом техніко-економічного рівня, що поєднує технічну якість і вартість розробки:

$$K_{\text{ТЕР}j} = \frac{KK_j}{C_j},$$

де KK_j — коефіцієнт технічного рівня;

C_j — повна вартість розробки j -го варіанта.

Підставляючи отримані значення:

$$K_{\text{ТЕР}1} = \frac{72,71}{368570,27} = 1,973 \cdot 10^{-4}$$

$$K_{\text{ТЕР}2} = \frac{54,77}{285782,37} = 1,916 \cdot 10^{-4}$$

Попри те, що Варіант 1 дорожчий у розробці, його коефіцієнт техніко-економічного рівня вищий ($1,973 \cdot 10^{-4}$ проти $1,916 \cdot 10^{-4}$, перевага близько 3 %). Це означає, що кожна гривня, вкладена в розробку Варіанта 1, забезпечує більший приріст технічної якості, ніж у Варіанті 2: вища вартість повністю виправдовується істотно вищою подієвою повнотою та меншою затримкою детекції — характеристиками, що мають вирішальне значення для систем безпеки критичної інфраструктури.

Слід окремо підкреслити, що навіть формальна перевага за коефіцієнтом КТЕР узгоджується з якісними міркуваннями. У задачах захисту критичної інфраструктури вартість невиявленої атаки (пошкодження обладнання, екологічні та техногенні наслідки, загроза життю) на порядки перевищує різницю у вартості розробки (близько 83 тис. грн). Тому Варіант 1, що виявляє в 1,7 раза більше атак-подій, є беззаперечно доцільним вибором.

Таким чином, оптимальним для реалізації є Варіант 1 — система на основі Python 3 та Google Colab (GPU T4), архітектури LSTM-Autoencoder, поознакового порогу з гістерезисною фільтрацією і повного набору метрик із візуалізацією. Саме цю конфігурацію й реалізовано в дипломній роботі.

4.8 Висновки до розділу 4

Виконано функціонально-вартісний аналіз програмного забезпечення для виявлення аномалій у телеметричних даних промислових КФС.

Побудовано функціональну модель продукту: визначено головну функцію та чотири підфункції (вибір середовища, архітектури моделі, стратегії порогування та способу оцінювання якості), для кожної з яких сформовано по два варіанти виконання. За допомогою позитивно-негативної матриці відібрано раціональні рішення та сформовано два цілісні варіанти реалізації — конфігурацію максимальної якості (Варіант 1) і конфігурацію мінімальної складності (Варіант 2).

Установлено систему з чотирьох техніко-економічних параметрів та методом експертного ранжування визначено їхню вагомість. Коефіцієнт узгодженості думок експертів $W = 0,837$ перевищує нормативний, що підтверджує достовірність оцінок. Найвагомішим параметром визнано подієву повноту ($K_v = 0,361$), що відповідає пріоритетам систем безпеки критичної інфраструктури.

Розраховано коефіцієнти технічного рівня варіантів на основі фактичних результатів експериментальної перевірки (розділ 3): $KK_1 = 72,71$ для Варіанта 1 та $KK_2 = 54,77$ для Варіанта 2. Перевага Варіанта 1 у 1,33 раза зумовлена його домінуванням за двома найвагомішими параметрами — подієвою повнотою (0,588 проти 0,353) та затримкою детекції (306 с проти 680 с).

Виконано економічний розрахунок повної вартості розробки: 368 570,27 грн для Варіанта 1 та 285 782,37 грн для Варіанта 2. Враховано заробітну плату виконавців, відрахування на ЄСВ, витрати на машинний час та накладні витрати.

За інтегральним критерієм техніко-економічного рівня оптимальним визнано Варіант 1 ($K_{TEP1} = 1,973 \cdot 10^{-4}$ проти $K_{TEP2} = 1,916 \cdot 10^{-4}$).

ВИСНОВКИ

У дипломній роботі розв'язано актуальну науково-прикладну задачу — розроблено інтелектуальну систему виявлення аномалій у телеметричних даних технологічних процесів критичної інфраструктури на основі методів глибинного навчання. Мету роботи досягнуто: створено програмний інструментарій, який моделює нормальну поведінку кіберфізичної системи й автоматично виявляє відхилення, спричинені несправностями або кібервтручаннями, без використання розмічених прикладів атак.

У теоретичній частині проаналізовано об'єкт дослідження — еталонну установку водоочищення SWaT 2015 — як шестиетапну кіберфізичну систему з ієрархічною архітектурою керування. Класифіковано аномалії за природою походження (технічні несправності, випадкові збої, навмисні кібератаки) та за масштабом впливу (SSSP, SSMP, MSSP, MSMP); встановлено, що найбільшу складність для виявлення становлять координовані та «стелс»-атаки, які маскуються під нормальну роботу окремих вузлів. За результатами порівняльного аналізу класичних і нейромережових методів обґрунтовано вибір архітектури автоенкодера як такої, що не потребує розмічених даних і здатна моделювати складні залежності між десятками сенсорів.

У другому розділі сформульовано математичну модель системи: формалізовано віконну обробку часових рядів, описано три архітектури автоенкодерів (Dense-AE, 1D-CNN-AE, LSTM-AE), обґрунтовано вибір функції втрат MSE та оптимізатора Adam, а також запропоновано два підходи до формування порогу детекції — глобальний перцентильний і поознаковий (per-feature). Ключовим методологічним рішенням стала стратегія leak-free калібрування: атаки роз'єднано на валідаційну й тестову підмножини, а нормальні вікна тест-домену поділено на калібрувальну та незалежну (hold-out) частини, тож пороги детекції калібруються й оцінюються на непересічних даних, що забезпечує об'єктивність метрик точності та хибних тривог.

У третьому розділі здійснено програмну реалізацію та повну експериментальну перевірку чотирьох конфігурацій детектора. Експериментально встановлено, що точкові метрики є недостатнім критерієм оцінювання детекторів у критичній інфраструктурі: моделі з глобальним порогом за прийнятного $F1 \approx 0,83$ пропускають більшість коротких атак (подієва повнота лише 0,35–0,41). Натомість поознакова модель з гістерезисною фільтрацією (M3) одночасно досягла найвищого точкового $F1$ (0,876), найвищої подієвої повноти (0,588), найнижчої кількості хибних тривог ($FP = 3034$) та вдвічі меншої середньої затримки детекції (≈ 306 с проти 586–685 с). Вона виявила всі атаки класу MSMP, дві третини SSMP та майже половину точкових SSSP, тоді як глобальні моделі повністю пропускали MSMP. Усі результати отримано без схеми point-adjustment, що робить їх об'єктивними та порівнянними з опублікованими роботами.

Наукова новизна роботи полягає у поєднанні поознакового (per-feature) порогового правила з гістерезисною (двопороговою часовою) фільтрацією та leak-free калібруванням, поширеним і на нормальні дані тестового домену. Абляційним дослідженням за схемою 2×2 вперше розділено внесок цих двох компонентів: поознаковий поріг забезпечує чутливість до локалізованих атак на окремі сенсори (підвищує подієву повноту), які глобальне усереднення помилки реконструкції «розмиває» до непомітності, а гістерезис придушує хибні спрацювання; їхнє поєднання дає найкращий баланс точності й повноти. Стійкість запропонованого рішення підтверджено аналізом чутливості до бюджету хибних тривог та до параметрів часової фільтрації.

У четвертому розділі методом функціонально-вартісного аналізу доведено, що обрана сукупність технічних рішень є не лише технічно ефективною, а й економічно доцільною: за інтегральним критерієм техніко-економічного рівня конфігурація на основі LSTM-автоенкодера з поознаковим порогом і гістерезисною фільтрацією переважає спрощену альтернативу, попри вищу вартість розробки.

Практична цінність роботи полягає у можливості розгортання навченого детектора в режимі реального часу: реалізовано серіалізацію всіх артефактів системи, а затримка обчислення показника аномальності за конструкцією не перевищує довжину вікна (20 с). Систему можна застосовувати як додатковий поведінковий рівень захисту в SCADA-системах об'єктів критичної інфраструктури.

Серед обмежень дослідження — використання єдиного датасету (SWaT 2015), мала кількість подій рідкісних класів атак у тестовій підмножині, зниження чутливості до найкоротших точкових атак як свідомо плата за зниження рівня хибних тривог, а також зростання частки хибних тривог за часового дрейфу нормального режиму (з 0,02 до 0,76 у разі калібрування лише на ранніх даних). Перспективними напрямками подальших досліджень є розроблення адаптивного онлайн-рекалібрування порога для протидії дрейфу, перевірка узагальнювальної здатності системи на інших наборах даних (BATADAL, HAI, SWaT 2021), дослідження гібридних архітектур (graph- та transformer-based) і врахування вартості хибних тривог у функції калібрування порогу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Dragos Inc. OT Cybersecurity Year in Review 2026. Dragos Inc., 2026. URL: <https://www.dragos.com/ot-cybersecurity-year-in-review> (Last accessed: 15.05.2026).
2. Лисов Б. С., Гуськова В. Г., Просянкіна-Жарова Т. І., Халигов А. А. Інформаційна технологія класифікації даних моніторингу систем і мереж передачі даних для виявлення кіберзагроз. *Вісник Національного технічного університету «ХПІ»*. Серія: Інформатика та моделювання. 2025. № 1 (13). С. 129–147. DOI: 10.20998/2411-0558.2025.01.09.
3. Mathur A. P., Tippenhauer N. O. SWaT: A Water Treatment Testbed for Research and Training on ICS Security. *Proceedings of the 2016 International Workshop on Cyber-Physical Systems for Smart Water Networks (CySWater)*. New York: ACM, 2016. P. 31–36. DOI: 10.1145/2831347.2831351.
4. Goh J., Adepu S., Junejo K. N., Mathur A. A Dataset to Support Research in the Design of Secure Water Treatment Systems. *Proceedings of the 11th International Conference on Critical Information Infrastructures Security (CRITIS 2016)*. Cham: Springer, 2016. P. 88–99. DOI: 10.1007/978-3-319-71368-7_8.
5. iTrust Labs. SWaT Dataset. Singapore University of Technology and Design, 2016. URL: https://itrust.sutd.edu.sg/itrust-labs_datasets/dataset_info/ (Last accessed: 12.05.2026).
6. Liu F. T., Ting K. M., Zhou Z.-H. Isolation Forest. *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008)*. Los Alamitos, CA: IEEE Computer Society, 2008. P. 413–422. DOI: 10.1109/ICDM.2008.17.
7. Schölkopf B. et al. Estimating the Support of a High-Dimensional Distribution / B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, R. C.

- Williamson. *Neural Computation*. 2001. Vol. 13, No. 7. P. 1443–1471. DOI: 10.1162/089976601750264965.
8. Nedashkovskaya N. I., Androsov D. V. Generative Time Series Model Based on Encoder-Decoder Architecture. *System Research and Information Technologies*. 2022. № 1. P. 97–109. DOI: 10.20535/SRIT.2308-8893.2022.1.08.
 9. Kingma D. P., Welling M. Auto-Encoding Variational Bayes. *2nd International Conference on Learning Representations (ICLR 2014)*. 2014. arXiv:1312.6114. URL: <https://arxiv.org/abs/1312.6114> (Last accessed: 16.05.2026).
 10. Audibert J. et al. USAD: UnSupervised Anomaly Detection on Multivariate Time Series / J. Audibert, P. Michiardi, F. Guyard, S. Marti, M. A. Zuluaga. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD 2020)*. New York: ACM, 2020. P. 3395–3404. DOI: 10.1145/3394486.3403392.
 11. Deng A., Hooi B. Graph Neural Network-Based Anomaly Detection in Multivariate Time Series. *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI 2021)*. 2021. Vol. 35, No. 5. P. 4132–4140. DOI: 10.1609/aaai.v35i5.16523.
 12. Tuli S., Casale G., Jennings N. R. TranAD: Deep Transformer Networks for Anomaly Detection in Multivariate Time Series Data. *Proceedings of the VLDB Endowment*. 2022. Vol. 15, No. 6. P. 1201–1214. DOI: 10.14778/3514221.3514232.
 13. Hundman K. et al. Detecting Spacecraft Anomalies Using LSTMs and Nonparametric Dynamic Thresholding / K. Hundman, V. Constantinou, C. Laporte, I. Colwell, T. Soderstrom. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD 2018)*. New York: ACM, 2018. P. 387–395. DOI: 10.1145/3219819.3219845.
 14. Kravchik M., Shabtai A. Detecting Cyber Attacks in Industrial Control Systems Using Convolutional Neural Networks. *Proceedings of the 2018*

- ACM Workshop on Cyber-Physical Systems Security and Privacy (CPS-SPC)*.
New York: ACM, 2018. P. 72–83. DOI: 10.1145/3264888.3264896.
- 15.Hochreiter S., Schmidhuber J. Long Short-Term Memory. *Neural Computation*. 1997. Vol. 9, No. 8. P. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.
 - 16.Kingma D. P., Ba J. Adam: A Method for Stochastic Optimization. *3rd International Conference on Learning Representations (ICLR 2015)*. 2015. arXiv:1412.6980. URL: <https://arxiv.org/abs/1412.6980> (Last accessed: 15.05.2026).
 - 17.IEC 62443-3-3:2013. Industrial communication networks — Network and system security — Part 3-3: System security requirements and security levels. Geneva: IEC, 2013.
 - 18.Kim S. et al. Towards a Rigorous Evaluation of Time-Series Anomaly Detection / S. Kim, K. Choi, H.-S. Choi, B. Lee, S. Yoon. *Proceedings of the 36th AAAI Conference on Artificial Intelligence (AAAI 2022)*. 2022. Vol. 36, No. 7. P. 7194–7201. DOI: 10.1609/aaai.v36i7.20680.

ДОДАТОК А ЛІСТИНГ КОДУ

```

import os, warnings, numpy as np, pandas as pd
warnings.filterwarnings('ignore')
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import (precision_recall_fscore_support,
                             confusion_matrix,
                             roc_curve, auc, precision_recall_curve)

import tensorflow as tf
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.layers import (Input, Dense, LSTM, RepeatVector,
                                     TimeDistributed, Conv1D,
                                     Conv1DTranspose,
                                     Flatten, Reshape)
from tensorflow.keras.callbacks import EarlyStopping

for _g in tf.config.list_physical_devices('GPU'):
    try: tf.config.experimental.set_memory_growth(_g, True)
    except Exception: pass

try:
    import google.colab
    IN_COLAB = True
except ImportError:
    IN_COLAB = False

NORMAL_FN = 'SWaT_Dataset_Normal_v1.xlsx' # train (нормальний режим)
ATTACK_FN = 'SWaT_Dataset_Attack_v0.xlsx' # test (з атаками)
ATTACKLIST = 'List_of_attacks_Final.xlsx'

if IN_COLAB:
    from google.colab import drive
    drive.mount('/content/drive')
    DATA_DIR = '/content/drive/MyDrive/swat_data'
    OUT_DIR = '/content/drive/MyDrive/swat_outputs'
else:
    DATA_DIR = os.path.abspath('.')
    OUT_DIR = os.path.abspath('swat_outputs')

FIG_DIR = os.path.join(OUT_DIR, 'figures')
MODEL_DIR = os.path.join(OUT_DIR, 'saved_models')
os.makedirs(FIG_DIR, exist_ok=True)
print('Середовище :', 'Google Colab' if IN_COLAB else 'локальний kernel')
print('Вхідні дані:', DATA_DIR)
print('Файли в DATA_DIR:', os.listdir(DATA_DIR))

# --- конфіг ---
WINDOW = 20
STAB_HOURS = 1.5
TRAIN_STRIDE = 2
EPOCHS = 30

```

```

BATCH          = 128
PRED_BATCH     = 1024
SEED           = 42

# параметри детекції
TARGET_FP = 0.03      # бюджет хибних тривог (частка) для вибору порога
HOLD_ON      = 5       # затримка ввімкнення (сек)
HOLD_OFF     = 15      # затримка вимкнення (сек)

# ваги вже збережені - завантажуюмо. True = переучити з нуля
TRAIN_FROM_SCRATCH = False

np.random.seed(SEED); tf.random.set_seed(SEED)
TIME_FMT = '%d/%m/%Y %I:%M:%S %p'

def load_swat(fn):
    pq = os.path.join(DATA_DIR, fn.replace('.xlsx', '.parquet'))
    if os.path.exists(pq):
        df = pd.read_parquet(pq)
    else:
        df = pd.read_excel(os.path.join(DATA_DIR, fn), header=1)
        df.columns = df.columns.str.strip()
        try:
            df.to_parquet(pq)
            print(f' (кеш збережено: {os.path.basename(pq)}) ')
        except Exception as e:
            print(' parquet-кеш не вдалось зберегти:', e)
    df['Timestamp'] =
pd.to_datetime(df['Timestamp'].astype(str).str.strip(),
               format=TIME_FMT, errors='coerce')
    return df.dropna(subset=['Timestamp']).reset_index(drop=True)

print('Завантаження даних...')
df_norm = load_swat(NORMAL_FN)
df_att  = load_swat(ATTACK_FN)

LABEL = 'Normal/Attack'
FEATURES = [c for c in df_norm.columns if c not in ('Timestamp', LABEL)]
print(f' Норма : {df_norm.shape}, {df_norm.Timestamp.min()} ->
{df_norm.Timestamp.max()}')
print(f' Атаки : {df_att.shape}, {df_att.Timestamp.min()} ->
{df_att.Timestamp.max()}')
print(f' Ознак : {len(FEATURES)}')

def clean(df):
    df = df.copy()
    for c in FEATURES:
        df[c] = pd.to_numeric(df[c].replace({'Active':1, 'Inactive':0, 'Bad
Input':np.nan}),
                               errors='coerce')
    df[FEATURES] = df[FEATURES].ffill().fillna(0)
    return df

df_norm = clean(df_norm)
df_att  = clean(df_att)
y_att   = (df_att[LABEL].astype(str).str.strip() ==
'Attack').astype(int).values

t0 = df_norm.Timestamp.min()

```

```

mask_stab = df_norm.Timestamp >= (t0 + pd.Timedelta(hours=STAB_HOURS))
df_norm = df_norm[mask_stab].reset_index(drop=True)
print(f' Після відсікання {STAB_HOURS} год стабілізації:
{df_norm.shape[0]} рядків норми')

Xn = df_norm[FEATURES].values.astype('float32')
scaler = MinMaxScaler()
scaler.fit(Xn)
rng = (scaler.data_max_ - scaler.data_min_)
const_feats = [FEATURES[i] for i in range(len(FEATURES)) if rng[i] < 1e-9]
print(f' Константних ознак на train-normal: {len(const_feats)} ')
Xn_s = scaler.transform(Xn)
Xa_s = scaler.transform(df_att[FEATURES].values.astype('float32'))

def contiguous(labels, times):
    iv, i = [], 0
    while i < len(labels):
        if labels[i] == 1:
            j = i
            while j+1 < len(labels) and labels[j+1] == 1: j += 1
            iv.append((i, j, times[i], times[j])); i = j+1
        else: i += 1
    return iv

intervals = contiguous(y_att, df_att.Timestamp.values)
print(f' Знайдено інтервалів атак: {len(intervals)}')

al = pd.read_excel(os.path.join(DATA_DIR, ATTACKLIST), header=0)
al.columns = [str(c).strip() for c in al.columns]

def stage_of(p):
    import re
    m = re.search(r'(\d)', str(p))
    return m.group(1) if m else '?'

al_valid = al.dropna(subset=['Start Time']).copy()
al_valid['st'] = pd.to_datetime(al_valid['Start Time'], errors='coerce')
_jan2015 = (al_valid['st'].dt.year == 2015) & (al_valid['st'].dt.month ==
1)
if _jan2015.any():
    al_valid.loc[_jan2015, 'st'] = al_valid.loc[_jan2015, 'st'] +
pd.DateOffset(years=1)
    print(f' Виправлено рік (2015->2016) для {_jan2015.sum()} записів
(a37-a41)')
al_valid = al_valid[~al_valid['Attack Point'].astype(str)
.str.contains('No Physical', case=False, na=False)]

attack_meta = []
for k, (i0, i1, ts, te) in enumerate(intervals, 1):
    ts_pd = pd.Timestamp(ts)
    cand = al_valid[(al_valid['st'] - ts_pd).abs() <
pd.Timedelta(minutes=3)]
    if len(cand):
        row = cand.iloc[0]
        point = str(row['Attack Point']); num = row['Attack #']
    else:
        point, num = '?', '?'
    attack_meta.append(dict(idx=k, num=num, point=point, i0=i0, i1=i1,
start=ts_pd, end=pd.Timestamp(te), n=i1-i0+1))

```

```

OFFICIAL_TYPE = {
    1:'SSSP', 2:'SSSP', 3:'SSSP', 4:'SSSP', 6:'SSSP', 7:'SSSP', 8:'S
SSP',
    10:'SSSP', 11:'SSSP', 13:'SSSP', 14:'SSSP', 16:'SSSP', 17:'SSSP',
19:'SSSP',
    20:'SSSP', 21:'SSMP', 22:'MSMP', 23:'MSMP', 24:'SSMP', 25:'SSMP',
26:'MSSP',
    27:'MSSP', 28:'SSSP', 29:'SSMP', 30:'MSMP', 31:'SSSP', 32:'SSSP',
33:'SSSP',
    34:'SSSP', 35:'SSMP', 36:'SSSP', 37:'SSMP', 38:'MSSP', 39:'MSSP',
40:'SSSP',
    41:'SSSP',
}

def classify(point):
    pts = [p.strip() for p in str(point).replace(' and
', ',').replace(':', ',').split(',')]
    if p.strip() not in ('', 'nan', '?')]
    stages = set(stage_of(p) for p in pts); stages.discard('?')
    multi_pt, multi_st = len(pts) > 1, len(stages) > 1
    if not multi_pt and not multi_st: return 'SSSP'
    if multi_pt and not multi_st: return 'SSMP'
    if not multi_pt and multi_st: return 'MSSP'
    return 'MSMP'

def cls_of(m):
    try:
        n = int(m['num'])
        if n in OFFICIAL_TYPE: return OFFICIAL_TYPE[n]
    except (ValueError, TypeError):
        pass
    return classify(m['point'])
for m in attack_meta:
    m['cls'] = cls_of(m)

from collections import defaultdict, Counter
buckets = defaultdict(list)
for m in sorted(attack_meta, key=lambda m: m['start']):
    buckets[m['cls']].append(m)
for cls, lst in buckets.items():
    for k, m in enumerate(lst):
        m['subset'] = 'val' if k % 2 == 0 else 'test'

print('\n Атаки (leak-free, стратифіковано за офіційним типом):')
for m in sorted(attack_meta, key=lambda m: m['start']):
    print(f"    #{m['idx']} a{str(m['num']):<3} {m['point']:<24}
{m['cls']:<5} "
          f"{m['start'].time()-m['end'].time()} [{m['subset']}]")
print('  val :', dict(Counter(m['cls'] for m in attack_meta if
m['subset']=='val')))
print('  test:', dict(Counter(m['cls'] for m in attack_meta if
m['subset']=='test')))

def make_windows(X, w, stride=1):
    idx = range(0, len(X)-w+1, stride)
    return np.stack([X[i:i+w] for i in idx]).astype('float32'),
np.array(list(idx))

```

```

Xtr, _ = make_windows(Xn_s, WINDOW, TRAIN_STRIDE)

te_beg = np.arange(0, len(Xa_s) - WINDOW + 1)
te_end = te_beg + WINDOW - 1
yte_win = y_att[te_end]
F = Xtr.shape[2]
n_test = len(te_beg)
print(f' Вікна: train={Xtr.shape}, test_вікон={n_test}, ознак={F}')

# приналежність кожного тестового вікна: 'normal' | 'val' | 'test' (за
останнім відліком)
def attack_subset_of_window(end_idx):
    for m in attack_meta:
        if m['i0'] <= end_idx <= m['i1']:
            return m['subset']
    return 'normal'
win_subset = np.array([attack_subset_of_window(e) for e in te_end])

# leak-free поділ норми
NORM_SPLIT = 'parity' # 'parity' (рекоменд.) | 'time' (строгіша проба
для розд. 3.7)
norm_idx = np.where(win_subset == 'normal')[0] # індекси
відсортовані за часом
if NORM_SPLIT == 'time':
    half = len(norm_idx) // 2
    cal_sel, ev_sel = norm_idx[:half], norm_idx[half:]
else: # parity: чергуємо вікна - обидві половини покривають увесь
таймлайн
    cal_sel, ev_sel = norm_idx[0::2], norm_idx[1::2]
norm_cal_mask = np.zeros(n_test, bool); norm_cal_mask[cal_sel] =
True # калібрування
norm_eval_mask = np.zeros(n_test, bool); norm_eval_mask[ev_sel] =
True # hold-out (FP)

# маска фінального оцінювання: hold-out норма + ТЕСТОВІ атаки
eval_mask = norm_eval_mask | (win_subset == 'test')
print(f' Поділ норми: {NORM_SPLIT}. Нормальних тест-вікон:
{len(norm_idx)} -> '
      f'калібрування={int(norm_cal_mask.sum())}, hold-
out(FP)={int(norm_eval_mask.sum())}')
print(f' Вікон у фінальному eval (hold-out норма + test-атаки):
{int(eval_mask.sum())}')

import gc
es = EarlyStopping(monitor='val_loss', patience=5,
restore_best_weights=True)

def build_dense_ae(w, f):
    inp = Input((w, f)); x = Flatten()(inp)
    x = Dense(128, activation='relu')(x); z = Dense(32,
activation='relu')(x)
    x = Dense(128, activation='relu')(z); x = Dense(w*f,
activation='linear')(x)
    out = Reshape((w, f))(x)
    mdl = Model(inp, out); mdl.compile('adam', 'mse'); return mdl

def build_lstm_ae(w, f):
    inp = Input((w, f))

```

```

x = LSTM(64, return_sequences=True)(inp)
x = LSTM(32, return_sequences=False)(x)
x = RepeatVector(w)(x)
x = LSTM(32, return_sequences=True)(x)
x = LSTM(64, return_sequences=True)(x)
out = TimeDistributed(Dense(f))(x)
mdl = Model(inp, out); mdl.compile('adam', 'mse'); return mdl

def build_cnn_ae(w, f):
    inp = Input((w, f))
    x = Conv1D(64, 3, activation='relu', padding='same')(inp)
    x = Conv1D(32, 3, activation='relu', padding='same')(x)
    x = Conv1DTranspose(32, 3, activation='relu', padding='same')(x)
    x = Conv1DTranspose(64, 3, activation='relu', padding='same')(x)
    out = Conv1D(f, 3, activation='linear', padding='same')(x)
    mdl = Model(inp, out); mdl.compile('adam', 'mse'); return mdl

BUILDERS = {'M1 Dense-AE': build_dense_ae, 'M2 LSTM-AE': build_lstm_ae,
'M4 CNN-AE': build_cnn_ae}
os.makedirs(MODEL_DIR, exist_ok=True)
_paths = {nm: os.path.join(MODEL_DIR, nm.split()[0] + '.keras') for nm in
BUILDERS}
_have_all = all(os.path.exists(p) for p in _paths.values())

histories = {}
models = {}
if TRAIN_FROM_SCRATCH or not _have_all:
    print('Навчання з нуля...' if TRAIN_FROM_SCRATCH else 'Ваги не
знайдено -> навчання...')
    for nm, build in BUILDERS.items():
        mdl = build(WINDOW, F)
        print(f' Навчання {nm} ...', flush=True)
        h = mdl.fit(Xtr, Xtr, epochs=EPOCHS, batch_size=BATCH,
                    validation_split=0.1, callbacks=[es], verbose=2)
        mdl.save(_paths[nm]) # зберігаємо ОДРАЗУ (щоб load
був валідним)
        models[nm] = mdl; histories[nm] = h.history
        gc.collect()
    print(' навчання завершено, ваги збережено')
else:
    print('Завантаження збережених ваг (без навчання)...')
    for nm in BUILDERS:
        models[nm] = load_model(_paths[nm])
    print(' моделі завантажено. ')

for nm, mdl in models.items():
    print(f" {nm}: {mdl.count_params()} параметрів")
n_test = len(te_beg)

def recon_errors(mdl):
    rec_tr = mdl.predict(Xtr, batch_size=256, verbose=0)
    g_tr = np.mean(np.abs(Xtr - rec_tr), axis=(1, 2))
    pf_tr = np.mean(np.abs(Xtr - rec_tr), axis=1)
    del rec_tr; gc.collect()
    g_te = np.empty(n_test, dtype='float32')
    pf_te = np.empty((n_test, F), dtype='float32')
    for s in range(0, n_test, PRED_BATCH):
        e = min(s + PRED_BATCH, n_test)
        batch = np.stack([Xa_s[te_beg[i]:te_beg[i] + WINDOW
for i in range(s, e)]).astype('float32')

```

```

        rec = mdl.predict(batch, batch_size=256, verbose=0)
        diff = np.abs(batch - rec)
        g_te[s:e] = diff.mean(axis=(1, 2))
        pf_te[s:e] = diff.mean(axis=1)
        del batch, rec, diff
    gc.collect()
    return g_tr, g_te, pf_tr, pf_te

errs = {nm: recon_errors(mdl) for nm, mdl in models.items()}

def smooth(a, k=10):
    return pd.Series(a).rolling(k, min_periods=1).mean().values

print('Помилки пораховано для:', list(errs.keys()))

val_atks = [m for m in attack_meta if m['subset'] == 'val']

def _event_recall_val(yp):
    if not val_atks: return 0.0
    return sum(1 for m in val_atks
               if (yp[(te_end>=m['i0']) & (te_end<=m['i1'])] == 1).any()) /
    len(val_atks)

def _fp_rate_cal(yp):
    n = int(norm_cal_mask.sum())
    return 0.0 if n == 0 else float(np.sum(yp[norm_cal_mask] == 1)) / n

def apply_hysteresis(flag, hold_on=None, hold_off=None):
    # hold_on=1, hold_off=0 - прапор без змін (вироджений випадок = без
    # гістерезису)
    hold_on = HOLD_ON if hold_on is None else hold_on
    hold_off = HOLD_OFF if hold_off is None else hold_off
    s = pd.Series(np.asarray(flag).astype(int))
    held_on = (s.rolling(hold_on, min_periods=1).sum() >=
    hold_on).astype(int)
    held_off = held_on.rolling(hold_off + 1,
    min_periods=1).max().astype(int)
    return held_off.values

GRID = [90, 95, 97, 98, 99, 99.5, 99.9, 99.95]

def _select(builders, label, verbose=True):
    metr = [(q, _event_recall_val(yp), _fp_rate_cal(yp), yp)
             for q, yp in ((q, fn()) for q, fn in builders)]
    ok = [r for r in metr if r[2] <= TARGET_FP]
    chosen = ok[0] if ok else min(metr, key=lambda r: r[2]) # перший =
    найнижчий q
    if verbose:
        print(f' [{label}]')
    q*={chosen[0]} ER_val={chosen[1]:.2f} FPR_cal={chosen[2]:.3f}'
        + (' ' if ok else ' (бюджет недосяжний -> мін. FPR)')
    for q, er, fpr, _ in metr:
        print(f' q={q:<6} ER={er:.2f} FPR_cal={fpr:.3f}'
              + (' <-' if q == chosen[0] else ''))
    return chosen[3], chosen[0]

# генератори прапора
def mk_global(sm, q, base, hyst):
    flag = (sm > np.percentile(base, q)).astype(int)

```

```

    return apply_hysteresis(flag) if hyst else flag

def mk_perfeat(pf, q, base, hyst):
    thr = np.percentile(base, q, axis=0)
    flag = (pf > thr).any(axis=1).astype(int)
    return apply_hysteresis(flag) if hyst else flag

def tune_global(g_te, label, hyst=False, verbose=True):
    sm = smooth(g_te); base = sm[norm_cal_mask]
    builders = [(q, (lambda q=q: mk_global(sm, q, base, hyst))) for q in
GRID]
    yp, q = _select(builders, label, verbose)
    return yp, float(np.percentile(base, q)), sm, q

def tune_perfeat(pf_te, label, hyst=True, verbose=True):
    base = pf_te[norm_cal_mask]
    builders = [(q, (lambda q=q: mk_perfeat(pf_te, q, base, hyst))) for q
in GRID]
    yp, q = _select(builders, label, verbose)
    return yp, float(q), np.percentile(base, q, axis=0)

# основні моделі: M1/M2/M4 = global без гістерезису; M3 = per-feature +
гістерезис на M2
print(f'Підбір порогів (TARGET_FP={TARGET_FP}, HOLD_ON={HOLD_ON},
HOLD_OFF={HOLD_OFF}):')
preds = {}; g_te_store = {}; thr_store = {}; q_store = {}
for nm in models:
    g_tr, g_te, pf_tr, pf_te = errs[nm]
    yp, thr, sm, q = tune_global(g_te, nm.split()[0], hyst=False)
    preds[nm] = yp; g_te_store[nm] = sm; thr_store[nm] = thr; q_store[nm]
= q

g_tr, g_te, pf_tr, pf_te = errs['M2 LSTM-AE']
yp3, q3, thr3 = tune_perfeat(pf_te, 'M3 per-feature+гістерезис',
hyst=True)
preds['M3 LSTM-AE perfeat'] = yp3
print('preds готово:', list(preds.keys()))

def attack_indices(subset):
    out = []
    for m in attack_meta:
        if m['subset'] != subset: continue
        sel = np.where((te_end>=m['i0']) & (te_end<=m['i1']))[0]
        out.append((m, sel))
    return out
test_atk = attack_indices('test')

def evaluate(yp):
    yt = yte_win[eval_mask]; yp_t = yp[eval_mask] # hold-out норма +
test-атаки
    p, r, f1, _ = precision_recall_fscore_support(yt, yp_t, average=None,
zero_division=0)
    P = p[1] if len(p)>1 else 0; R = r[1] if len(r)>1 else 0; F1 = f1[1]
if len(f1)>1 else 0
    FP = int(np.sum((yp_t==1) & (yt==0))) # FP на hold-out нормі
    FN = int(np.sum((yp_t==0) & (yt==1)))
    hit, delays = 0, []
    for m, sel in test_atk:
        if len(sel)==0: delays.append(None); continue

```

```

        h = np.where(yp[sel]==1)[0]
        if len(h): hit += 1; delays.append(int(h[0]))
        else: delays.append(None)
    er = hit/len(test_atk) if test_atk else 0
    dd = np.mean([d for d in delays if d is not None]) if any(d is not
None for d in delays) else None
    return dict(P=P, R=R, F1=F1, FP=FP, FN=FN, ER=er, DD=dd,
delays=delays)

results = {nm: evaluate(yp) for nm, yp in preds.items()}
order = ['M1 Dense-AE', 'M2 LSTM-AE', 'M3 LSTM-AE perfeat', 'M4 CNN-AE']

print('\n===== ТАБЛИЦЯ test, hold-out нопма =====')
print(f"{'Model':<22}{'Prec':>6}{'Rec':>6}{'F1':>6}{'EventR':>8}{'Delay':>
7}{'FP':>6}{'FN':>6}")
for nm in order:
    r = results[nm]; dd = f"{r['DD']:.0f}" if r['DD'] is not None else '-'
    print(f"{'nm':<22}{r['P']:>6.3f}{r['R']:>6.3f}{r['F1']:>6.3f}{r['ER']:>8
.3f}{dd:>7}{r['FP']:>6}{r['FN']:>6}")

tbl =
pd.DataFrame([{'Model':nm, 'Precision':results[nm]['P'], 'Recall':results[nm]
['R'],
                'F1':results[nm]['F1'], 'EventRecall':results[nm]['ER']
],
                'DetectionDelay':results[nm]['DD'], 'FP':results[nm]['
FP'],
                'FN':results[nm]['FN']} for nm in order])
tbl.to_csv(os.path.join(OUT_DIR, 'model_comparison.csv'), index=False)

# Event Recall у поспізі типів атак
test_classes = sorted([m['cls'] for m in attack_meta if
m['subset']=='test'])
print('\n--- Event Recall за типом атаки (test) ---')
print(f"{'Model':<22}" + ''.join(f"{c:>8}" for c in test_classes))
perclass_rows = []
for nm in order:
    yp = preds[nm]; row = {'Model': nm}; line = f"{nm:<22}"
    for c in test_classes:
        atk_c = [m for m in attack_meta if m['subset']=='test' and
m['cls']==c]
        hit = sum(1 for m in atk_c
                    if len(np.where((te_end>=m['i0'])&(te_end<=m['i1'])))[0])
                    and
                    (yp[np.where((te_end>=m['i0'])&(te_end<=m['i1']))][0]==1).any())
        er = hit/len(atk_c) if atk_c else float('nan')
        row[c] = er; line += f"{er:>8.2f}"
    perclass_rows.append(row); print(line)
pd.DataFrame(perclass_rows).to_csv(os.path.join(OUT_DIR, 'event_recall_by_c
lass.csv'), index=False)

g_tr, g_te, pf_tr, pf_te = errs['M2 LSTM-AE']

print('Калібрування 4 правил (на M2):')
cells22 = {}
cells22['Global, без рістєрезису'] = tune_global (g_te, 'G -hyst',
hyst=False, verbose=False)[0]
cells22['Global, з рістєрезисом'] = tune_global (g_te, 'G +hyst',
hyst=True, verbose=False)[0]

```

```

cells22['Per-feature, без гістерезису'] = tune_perfeat(pf_te, 'PF -hyst',
hyst=False, verbose=False)[0]
cells22['Per-feature, з гістерезисом'] = tune_perfeat(pf_te, 'PF +hyst',
hyst=True, verbose=False)[0]

res22 = {k: evaluate(v) for k, v in cells22.items()}

print('\n==== мережа M2 ====')
print(f"{'Правило  
детекції':<30}{'F1':>7}{'Recall':>8}{'EventR':>8}{'FP':>7}")
for k in cells22:
    r = res22[k]
    print(f"{k:<30}{r['F1']:>7.3f}{r['R']:>8.3f}{r['ER']:>8.3f}{r['FP']:>7  
}")
pd.DataFrame([{'Rule':k, 'F1':res22[k]['F1'], 'Recall':res22[k]['R'],
'EventRecall':res22[k]['ER'], 'FP':res22[k]['FP']} for k in
cells22]
).to_csv(os.path.join(OUT_DIR, 'ablation_2x2.csv'),
index=False)

# PF+hyst має збігатися з M3 основної таблиці
_chk = abs(res22['Per-feature, з гістерезисом']['F1'] - results['M3 LSTM-
AE perfeat']['F1'])
print(f'\n [перевірка] |F1(PF+hyst) - F1(M3)| = {_chk:.4f} (має бути
~0)')

# F1 та FP
labels22 = ['без гіст.', 'з гіст.']
g_f1 = [res22['Global, без гістерезису']['F1'], res22['Global, з
гістерезисом']['F1']]
pf_f1 = [res22['Per-feature, без гістерезису']['F1'], res22['Per-feature,
з гістерезисом']['F1']]
g_fp = [res22['Global, без гістерезису']['FP'], res22['Global, з
гістерезисом']['FP']]
pf_fp = [res22['Per-feature, без гістерезису']['FP'], res22['Per-feature,
з гістерезисом']['FP']]
x = np.arange(2); wd = 0.35
fig, ax = plt.subplots(1, 2, figsize=(12, 4.5))
ax[0].bar(x-wd/2, g_f1, wd, label='Global', color='#2b6cb0')
ax[0].bar(x+wd/2, pf_f1, wd, label='Per-feature', color='#c0392b')
ax[0].set_xticks(x); ax[0].set_xticklabels(labels22); ax[0].set_ylim(0,1)
ax[0].set_title('F1-mipa'); ax[0].legend()
for i,v in enumerate(g_f1): ax[0].text(i-wd/2, v+0.02, f'{v:.2f}',
ha='center', fontsize=8)
for i,v in enumerate(pf_f1): ax[0].text(i+wd/2, v+0.02, f'{v:.2f}',
ha='center', fontsize=8)
ax[1].bar(x-wd/2, g_fp, wd, label='Global', color='#2b6cb0')
ax[1].bar(x+wd/2, pf_fp, wd, label='Per-feature', color='#c0392b')
ax[1].set_xticks(x); ax[1].set_xticklabels(labels22)
ax[1].set_title('Хибні тривоги (FP, hold-out норма)'); ax[1].legend()
plt.suptitle('Внесок поознакового порога та гістерезису (мережа M2)')
plt.tight_layout(); plt.savefig(f'{FIG_DIR}/10_ablation_2x2.png',
dpi=130); plt.close()
print('Збережено: 10_ablation_2x2.png')

COL = {'M1 Dense-AE':'#888', 'M2 LSTM-AE':'#2b6cb0',
'M3 LSTM-AE perfeat':'#c0392b', 'M4 CNN-AE':'#27ae60'}

# Loss
if histories:

```

```

fig, ax = plt.subplots(1, 3, figsize=(15, 4))
for i, nm in enumerate(['M1 Dense-AE', 'M2 LSTM-AE', 'M4 CNN-AE']):
    h = histories[nm]
    ax[i].plot(h['loss'], label='Train (MSE)', color='#2b6cb0')
    ax[i].plot(h['val_loss'], label='Val (MSE)', color='#c0392b')
    ax[i].set_title(f'Навчання {nm.split()[0]}');
ax[i].set_xlabel('Епохи'); ax[i].legend()
plt.tight_layout(); plt.savefig(f'{FIG_DIR}/00_training_loss.png',
dpi=130); plt.close()
print('Збережено: 00_training_loss.png')
else:
    print('histories порожні ')

# Порівняння моделей
fig, ax = plt.subplots(1, 3, figsize=(14, 4))
for a, (metr, lab) in zip(ax, [('F1', 'F1-score'), ('R', 'Recall'), ('ER', 'Event
Recall')]):
    vals = [results[nm][metr] for nm in order]
    a.bar(range(len(order)), vals, color=[COL[n] for n in order])
    a.set_xticks(range(len(order))); a.set_xticklabels([n.split()[0] for n
in order])
    a.set_title(lab); a.set_ylim(0,1)
    for i,v in enumerate(vals): a.text(i, v+0.02, f'{v:.2f}', ha='center',
fontsize=8)
plt.tight_layout(); plt.savefig(f'{FIG_DIR}/03_model_comparison.png',
dpi=130); plt.close()

# ROC + PR на eval-вибірці
yt = yte_win[eval_mask]
plt.figure(figsize=(12,5))
plt.subplot(1,2,1)
for nm in ['M1 Dense-AE', 'M2 LSTM-AE', 'M4 CNN-AE']:
    g_tr, g_te, _, _ = errs[nm]; raw = smooth(g_te)[eval_mask]
    fpr, tpr, _ = roc_curve(yt, raw); plt.plot(fpr, tpr, color=COL[nm],
label=f'{nm.split()[0]} AUC={auc(fpr, tpr):.2f}')
plt.plot([0,1], [0,1], 'k--', lw=.6); plt.xlabel('FPR'); plt.ylabel('TPR');
plt.title('ROC (eval)'); plt.legend(fontsize=8)
plt.subplot(1,2,2)
for nm in ['M1 Dense-AE', 'M2 LSTM-AE', 'M4 CNN-AE']:
    g_tr, g_te, _, _ = errs[nm]; raw = smooth(g_te)[eval_mask]
    pr, rc, _ = precision_recall_curve(yt, raw);
plt.plot(rc, pr, color=COL[nm], label=nm.split()[0])
plt.xlabel('Recall'); plt.ylabel('Precision'); plt.title('PR (eval)');
plt.legend(fontsize=8)
plt.tight_layout(); plt.savefig(f'{FIG_DIR}/04_roc_pr.png', dpi=130);
plt.close()

# Матриці помилок на eval
fig, ax = plt.subplots(1, 4, figsize=(16, 4))
for a, nm in zip(ax, order):
    cm = confusion_matrix(yte_win[eval_mask], preds[nm][eval_mask],
labels=[0,1])
    a.imshow(cm, cmap='Blues'); a.set_title(nm.split()[0], fontsize=10)
    a.set_xticks([0,1]); a.set_yticks([0,1])
    a.set_xticklabels(['Norm', 'Atk']); a.set_yticklabels(['Norm', 'Atk'])
    for (i,j), v in np.ndenumerate(cm):
        a.text(j,i, int(v), ha='center', va='center', color='white' if
v>cm.max()/2 else 'black')
    a.set_ylabel('true'); a.set_xlabel('pred')

```

```

plt.tight_layout(); plt.savefig(f'{FIG_DIR}/05_confusion.png', dpi=130);
plt.close()

# Розподіл помилки (M2)
g_tr, g_te, _, _ = errs['M2 LSTM-AE']; sm = smooth(g_te)
plt.figure(figsize=(9,5))
plt.hist(sm[yte_win==0], bins=60, alpha=.6, label='Норма',
color='#2b6cb0', density=True)
plt.hist(sm[yte_win==1], bins=60, alpha=.6, label='Атака',
color='#c0392b', density=True)
plt.xlabel('згладжений MAE'); plt.ylabel('щільність')
plt.title('Розподіл помилки реконструкції (M2 LSTM-AE)'); plt.legend()
plt.tight_layout(); plt.savefig(f'{FIG_DIR}/06_error_dist.png', dpi=130);
plt.close()

# Timeline
fig, ax = plt.subplots(figsize=(13,4))
for yi, nm in enumerate(order):
    det = te_end[(preds[nm]==1)]
    ax.scatter(df_att.Timestamp.values[det], [yi]*len(det), s=4,
color=COL[nm])
for m in attack_meta:
    ax.axvspan(m['start'], m['end'], color='red', alpha=0.12)
ax.set_yticks(range(len(order))); ax.set_yticklabels([n.split()[0] for n
in order])
ax.xaxis.set_major_formatter(mdates.DateFormatter('%H:%M'))
ax.set_title('Timeline детекції (червоне – реальні атаки)')
plt.tight_layout(); plt.savefig(f'{FIG_DIR}/07_timeline.png', dpi=130);
plt.close()
print('Збережено у', FIG_DIR)

budgets = [0.01, 0.02, 0.03, 0.05, 0.10]
curve = {nm: {'fp': [], 'er': []} for nm in order}
_saved_tf = TARGET_FP
for tf_ in budgets:
    TARGET_FP = tf_
    pr = {}
    for nm in models:
        g_tr, g_te, pf_tr, pf_te = errs[nm]
        pr[nm] = tune_global(g_te, nm.split()[0], hyst=False,
verbose=False)[0]
        g_tr, g_te, pf_tr, pf_te = errs['M2 LSTM-AE']
        pr['M3 LSTM-AE perfeat'] = tune_perfeat(pf_te, 'M3', hyst=True,
verbose=False)[0]
    for nm in order:
        r = evaluate(pr[nm]); curve[nm]['fp'].append(r['FP']);
curve[nm]['er'].append(r['ER'])
TARGET_FP = _saved_tf

plt.figure(figsize=(8,6))
for nm in order:
    plt.plot(curve[nm]['fp'], curve[nm]['er'], 'o-', label=nm.split()[0],
color=COL[nm])
plt.xlabel('False Positives (hold-out норма)'); plt.ylabel('Event Recall
(test)')
plt.title('Компроміс EventRecall ↔ FP за бюджетом TARGET_FP');
plt.legend(); plt.grid(alpha=.3)
plt.savefig(f'{FIG_DIR}/08_recall_fp_tradeoff.png', dpi=130,
bbox_inches='tight'); plt.close()
print('Збережено: 08_recall_fp_tradeoff.png')

```

```

TARGET_FP = _saved_tf
print(f'Ablation hold_off (15,30,60) при TARGET_FP={TARGET_FP},
HOLD_ON={HOLD_ON}...')
hold_off_values = [15, 30, 60]; f1_scores = []; fp_counts = []
g_tr, g_te, pf_tr, pf_te = errs['M2 LSTM-AE']
for ho in hold_off_values:
    base = pf_te[norm_cal_mask]
    builders = [(q, (lambda q=q, ho=ho: apply_hysteresis(
        (pf_te > np.percentile(base, q,
axis=0)).any(axis=1).astype(int),
        hold_on=HOLD_ON, hold_off=ho))) for q in GRID]
    yp_abl, _ = _select(builders, f'hold_off={ho}', verbose=False)
    res = evaluate(yp_abl); f1_scores.append(res['F1']);
fp_counts.append(res['FP'])
    print(f' hold_off={ho:>2} -> F1={res["F1"]:.3f}, FP={res["FP"]}')

fig, ax1 = plt.subplots(figsize=(8,5))
ax1.set_xlabel('hold_off (c)'); ax1.set_ylabel('F1-score',
color='#2b6cb0')
l1 = ax1.plot(hold_off_values, f1_scores, 'o-', color='#2b6cb0', lw=2,
label='F1-score')
ax1.tick_params(axis='y', labelcolor='#2b6cb0');
ax1.set_xticks(hold_off_values); ax1.grid(alpha=.3)
ax2 = ax1.twinx(); ax2.set_ylabel('FP', color='#c0392b')
l2 = ax2.plot(hold_off_values, fp_counts, 's--', color='#c0392b', lw=2,
label='FP')
ax2.tick_params(axis='y', labelcolor='#c0392b')
ax1.legend(l1+l2, [x.get_label() for x in l1+l2], loc='center right')
plt.title('Ablation: F1 та FP від hold_off (мережа M2)')
fig.tight_layout(); plt.savefig(f'{FIG_DIR}/09_ablation_hold_off.png',
dpi=130); plt.close()
print('Збережено: 09_ablation_hold_off.png')

# незалежно від NORM_SPLIT будуємо часовий поділ норми для проби
_t_half = len(norm_idx) // 2
_t_cal = np.zeros(n_test, bool); _t_cal[norm_idx[:_t_half]] = True
_t_eval = np.zeros(n_test, bool); _t_eval[norm_idx[_t_half:]] = True

g_tr, g_te, pf_tr, pf_te = errs['M2 LSTM-AE']
sm = smooth(g_te)
# поріг за бюджетом на часовій калібрувальній нормі (1-а половина)
q_star = None
for q in GRID:
    thr_q = np.percentile(sm[_t_cal], q)
    fpr_cal = float(np.mean((sm[_t_cal] > thr_q)))
    if fpr_cal <= TARGET_FP:
        q_star = q; break
if q_star is None: q_star = GRID[-1]
thr_t = np.percentile(sm[_t_cal], q_star)
fpr_cal = float(np.mean(sm[_t_cal] > thr_t)) # на 1-й половині
(калібрування)
fpr_eval = float(np.mean(sm[_t_eval] > thr_t)) # на 2-й половині
(майбутнє)
print('--- часовий holdout (M2) ---')
print(f' q*={q_star}, поріг={thr_t:.4f}')
print(f' FPR на калібрувальній (ранній) нормі : {fpr_cal:.3f}')
print(f' FPR на hold-out (пізній) нормі : {fpr_eval:.3f} <--
дрейф')

```

```

import json, joblib
for nm, mdl in models.items():
    mdl.save(os.path.join(MODEL_DIR, nm.split()[0] + '.keras'))
joblib.dump(scaler, os.path.join(MODEL_DIR, 'scaler.joblib'))
artifacts = {
    'window': WINDOW, 'features': FEATURES,
    'global_thresholds': thr_store, 'global_q': q_store,
    'perfeature_q': float(q3), 'perfeature_thresholds': thr3.tolist(),
    'smooth_k': 10, 'hold_on': HOLD_ON, 'hold_off': HOLD_OFF, 'target_fp':
TARGET_FP,
    'norm_split': NORM_SPLIT,
    'ablation_2x2': {k:
{'F1':res22[k]['F1'], 'FP':res22[k]['FP'], 'ER':res22[k]['ER']} for k in
cells22},
}
with open(os.path.join(MODEL_DIR, 'detector_config.json'), 'w',
encoding='utf-8') as f:
    json.dump(artifacts, f, ensure_ascii=False, indent=2)
print('Збережено у', MODEL_DIR)

print('='*64)
print('Метрики:')
print(f"{'Model':<22}{'Prec':>7}{'Rec':>7}{'F1':>7}{'EventR':>8}{'Delay':>
7}{'FP':>7}{'FN':>7}")
for nm in order:
    r = results[nm]; dd = f"{r['DD']:.0f}" if r['DD'] is not None else '-'
    print(f"{nm:<22}{r['P']:>7.3f}{r['R']:>7.3f}{r['F1']:>7.3f}{r['ER']:>8
.3f}{dd:>7}{r['FP']:>7}{r['FN']:>7}")
print('\nEvent Recall за типами event_recall_by_class.csv')
print('\nМежею M2')
for k in cells22:
    r = res22[k]; print(f"    {k:<30}
F1={r['F1']:.3f}    Recall={r['R']:.3f}    EventR={r['ER']:.3f}    FP={r['FP']}"
)

```

ДОДАТОК Б ПРЕЗЕНТАЦІЯ

Інтелектуальна система виявлення аномалій у даних технологічних процесів критичної інфраструктури на основі методів глибинного навчання

Виконала: Студентка IV курсу, групи КА-21 Михальчук Євгенія. Богданівна

Керівник: Асистент каф. ММСА Древаль Максим Михайлович

АКТУАЛЬНІСТЬ РОБОТИ

Критична інфраструктура стала вразливою. Конвергенція ІТ та ОТ-технологій розширила поверхню атак. Сучасні ICS/SCADA більше не ізольовані — і традиційний сигнатурний захист у ОТ-мережах не працює.



Нові стратегії атак

Угрупування ELECTRUM, VOLTZITE вивчають фізику процесів КІ заради фізичних руйнувань; техніки LOTL роблять їх непомітними.



Сигнатури безсилі

Пропрієтарні протоколи, слабкі кінцеві пристрої та вимога безперервної роботи унеможливають класичний захист ОТ.



Потрібен поведінковий рівень

Система, що фіксує відхилення процесу від норми без втручання в його детерміновану логіку.

ОБ'ЄКТ, ПРЕДМЕТ ТА МЕТА ДОСЛІДЖЕННЯ

ОБ'ЄКТ

Процеси оброблення та аналізу телеметричних даних промислових кіберфізичних систем об'єктів критичної інфраструктури

ПРЕДМЕТ

Методи глибинного навчання на основі автоенкодерів для автоматизованого виявлення аномалій у багатовимірних часових рядах телеметрії кіберфізичної системи.

МЕТА

Створення системи виявлення аномалій, що моделює нормальну поведінку КФС та автоматично фіксує відхилення, спричинені несправностями або кібервтручаннями, без використання розмічених прикладів атак.

03

ДАНИ · DATASET SWAT 2015

Вхід — багатовимірний часовий ряд із 51 сенсора

51

вимірювальних
та керуючих тегів

~496 тис.

записів
нормального режиму

~449 тис.

записів
у режимі атак

41

сценарій атак
(36 з фіз. впливом)

Неперервні сенсори

рівень, витрата, рН/провідність, тиск — дійсні величини

Дискретні актуатори

клапани, насоси, граничні датчики — стани On / Off

Технологічний цикл — шість фізично взаємопов'язаних стадій:

P1

Збір сирової води

P2

Хім. дозування

P3

Ультрафільтрація

P4

Деслювання

P5

Зворотний осмос

P6

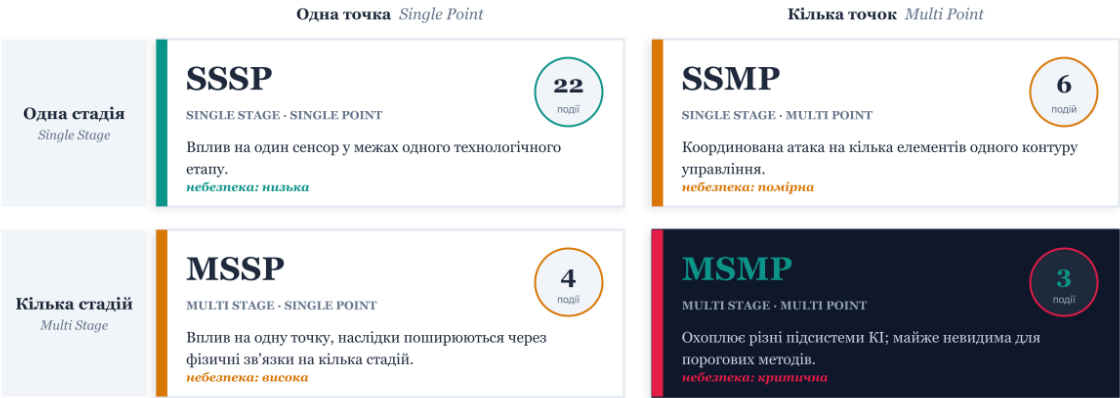
Розподіл / промивка

Стадії каскадно пов'язані: атака на клапан стадії P1 неминуче змінює стан на P5. Саме ця складна просторово-часова кореляція між десятками сенсорів зумовила вибір **рекурентних мереж (LSTM)** у роботі.

04

Модель масштабу атак: Stage × Point

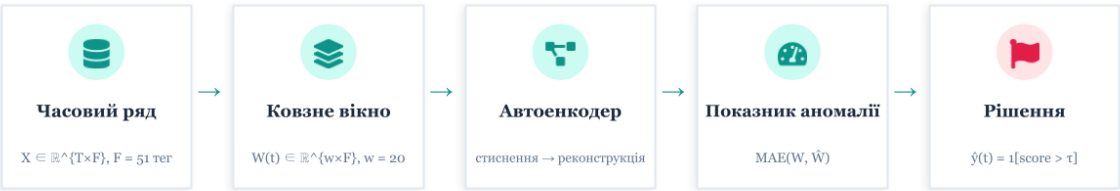
Атаки класифікують за двома осями — скільки технологічних стадій уражено та скільки точок (сенсорів/актуаторів) маніпульовано.



05

Виявлення без розмічених атак: реконструкція норми

Автоенкодер навчається відновлювати лише нормальну поведінку. Велика помилка реконструкції = аномалія.



Навчання — режим unsupervised: модель бачить лише норму. **Критерій** — мінімізація MSE реконструкції. **Виклик** — вибір порога τ : низький → хибні тривоги, високий → пропуск атак.

06

Дві пастки в оцінюванні детекторів



Point-adjustment завищує F1

Якщо зараховувати весь інтервал атаки за одну вгадану точку, навіть випадковий детектор дає $F1 \approx 0,9$ (Kim et al., AAAI 2022).



Дрейф розподілу (distribution shift)

Норма у файлі атак статистично відрізняється від тренувальної. Поріг, відкалібрований на train-нормі, генерує хибні тривоги на тесті.

Рішення — *leak-free* калібрування на нормі тестового домену

07

ОБРАНІ АРХІТЕКТУРИ

Три автоенкодери зі зростаючим часовим контекстом

M1

БАЗОВА ЛІНІЯ

Dense-AE

Повнозв'язні шари, без явного часового моделювання. Швидкий прототип конвеєра.

270 620

параметрів

M2

КЛЮЧОВА МОДЕЛЬ

LSTM-AE

Рекурентні гейти моделюють інерцію фізичних процесів і довгі залежності між підсистемами.

78 579

параметрів

M4

ЛОКАЛЬНІ ПАТЕРНИ

1D-CNN-AE

Згорткові ядра вздовж часу ловлять короткі транзйєнти; обмежене рецептивне поле.

35 187

параметрів

08

Модель М3: поознаковий поріг + гістерезис

Та сама навчена мережа LSTM-AE (М2), але інше правило детекції

**1 · Поознаковий поріг (per-feature)**

Кожен із 51 сенсора отримує власний поріг τ_i
Тривога-кандидат, якщо хоч одна ознака перевищила свій τ_i (правило «АБО»)
Ловить локальні відхилення, які глобальне усереднення «розмиває» до непомітності

**2 · Гістерезис (тригер Шмітта)**

hold-on = 5 c — тривога лише після 5 послідовних секунд підтвердження. Відсікає поодинокі сплески
hold-off = 15 c — утримує тривогу, склеюючи спрацювання в одну подію.
Каузально (rolling-sum / rolling-max, без look-ahead) працює в реальному часі.

09

МЕТОДОЛОГІЯ

Leak-free калібрування: пороги й оцінка — на різних даних





Стратифікація зберігає баланс класів у val і test, а пороги калібруються й оцінюються на непересічних даних. Бюджет TARGET_FP = 0,03 контролюється за конструкцією — тому зафіксований FP об’єктивний.

10

РЕЗУЛЬТАТИ

Підсумкові метрики: М3 виграє за всіма одразу

Модель	Precision	Recall	F1	Event Recall	Delay, c	FP	FN
M1 Dense-AE	0,895	0,769	0,827	0,353	680	3947	10107
M2 LSTM-AE	0,894	0,769	0,827	0,412	586	3960	10107
M3 LSTM-AE per-feat	0,923	0,834	0,876	0,588	306	3034	7242
M4 CNN-AE	0,895	0,767	0,826	0,353	685	3950	10160

F1 = 0,876

проти ≈0,827 у глобальних

Event Recall ×1,4–1,7

0,588 проти 0,35–0,41

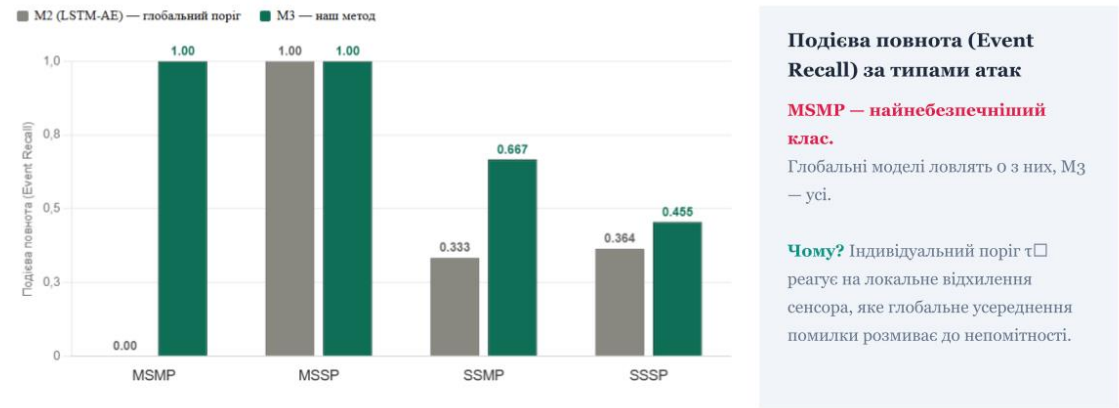
Затримка вдвічі менша

306 с проти 586–685 с

Найменший FP = 3034

11

Глобальний поріг повністю пропускає клас MSMP



* Не лише М2: за глобального порога **усі три моделі (М1, М2, М4) пропускають клас MSMP** (Event Recall = 0). М2 показано як основу М3 — обидві мають однакову мережу, відрізняється лише правило детекції.

12

Чому потрібні обидва компоненти (схема 2×2)

Усі чотири правила — на одній навченій мережі M2, тож єдиний змінний фактор — саме правило детекції.

Правило детекції	F1	Event Recall	FP
Глобальний, без гістерезису	0,827	0,412	3960
Глобальний, з гістерезисом	0,822	0,412	4509
Per-feature, без гістерезису	0,860	0,647	5301
Per-feature, з гістерезисом (M3)	0,876	0,588	3034

Per-feature → повнота

Event Recall 0,412 → 0,647, але FP зростає до 5301.

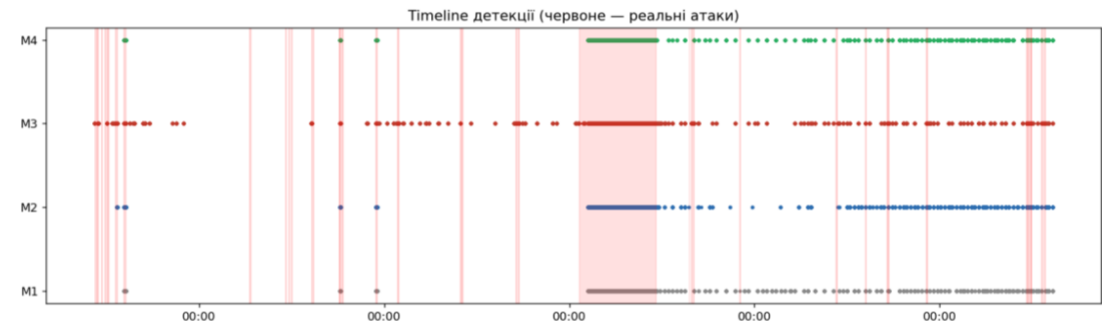
Гістерезис → точність

FP 5301 → 3034 (−43 %), F1 росте до 0,876.

Компоненти взаємодоповнювальні: поозначковий поріг дає чутливість, гістерезис — точність. Разом — найвищий F1 за найнижчого FP.

13

Локалізація аномалій і часова шкала детекції



Рядок M3 — найщільніше покриття інтервалів атак; M1/M2/M4 спрацьовують лише на домінантній атаці.

14

Обмеження та напрями подальшої роботи



Один датасет

Результати на SWaT 2015; узагальнення на HAI, BATADAL потребує перевірки.



Дрейф норми

За часового спліту частка FP зростає 0,02 → 0,76 — статичний поріг потребує онлайн-рекалібрування.



Рідкісні класи

MSMP представлений 1 подією, MSSP — 2; значення Event Recall = 1,0 — на малій вибірці.



Короткі атаки SSSP

Умова hold-on відсікає сплески < 5 с — свідомо плата за зниження хибних тривог.

Перспективи: адаптивне рекалібрування проти дрейфу · гібридні graph/transformer-архітектури · врахування вартості FP у функції калібрування.

15

ВИСНОВКИ

Що зроблено

- 1 Розроблено систему виявлення аномалій у телеметрії КФС на основі автоенкодерів — без розмічених атак.
- 2 Запропоновано поозначковий поріг + двопорогову гістерезисну фільтрацію зі строгим leak-free калібруванням.
- 3 Абляцією 2×2 розділено внески: per-feature дає повноту, гістерезис — точність.
- 4 МЗ переважає за всіма метриками: F1 = 0,876, Event Recall = 0,588, FP = 3034, затримка ≈ 306 с; усі MSMP-атаки виявлено.
- 5 Доведено придатність до роботи в реальному часі (real-time, латентність ≤ 20 с).

Дякую за увагу!