# *Проєктування поліпшеної архітектури LSTM та XGBoost для контекстно-орієнтованого виявлення аномалій у журналах подій на основі CPU*

https://doi.org/10.31713/MCIT.2025.025

Дмитро Гнатюк

Черкаський національний університет
ім. Богдана Хмельницького
Черкаси, Україна
hnatiuk.dmytro1123@vu.cdu.edu.ua

*Abstract* — Традиційні системи моніторингу IT-інфраструктури не враховують контекстні зв'язки між подіями у журналах подій, що призводить до високого рівня хибних спрацювань (до 60-80%). У роботі пропонується інноваційна гібридна архітектура через поєднання семантичного розуміння послідовностей (LSTM) з точністю табличної класифікації (XGBoost). Основна ідея полягає у створенні "семантичного відбитка" історії подій для кожного сервісу. Очікувані експериментальні результати мають показати покращення оцінки F1 на 15-25% при збереженні низької латентності (<50ms) при роботі виключно на CPU.

**Keywords**— ***виявлення аномалій в журналах подій; LSTM; XGBoost; CPU оптимізація та адаптивні пороги***

## I. ВСТУП

Сучасні розподілені системи генерують величезні обсяги журналів подій, аналіз яких є критично важливим для забезпечення стабільності та безпеки IT-інфраструктури. Традиційні підходи до виявлення аномалій часто не враховують контекстну інформацію та семантичні зв'язки між подіями, що призводить до високого рівня хибних спрацювань.

Представлена робота пропонує архітектурне рішення, яке поєднує переваги рекурентних нейронних мереж для аналізу послідовностей з ефективністю градієнтного бустингу для табличних даних. Система розроблена з урахуванням обмежень реальних production-середовищ, де GPU-ресурси можуть бути недоступними.

## II. АРХІТЕКТУРА СИСТЕМИ

Вдосконалена система виявлення аномалій у журналах подій побудована за принципом багатоетапної обробки даних з гібридним підходом до аналізу. Загальна архітектура складається з двох основних потоків: послідовного аналізу службових повідомлень (метод LSTM) для розуміння семантичного контексту та табличного аналізу вектору отриманої інформації зі службових повідомлень (метод XGBoost) для точної класифікації ознак на основі статистичних даних [1].

### *A. Ключові компоненти:*

1. Log Ingestion & Parsing: обробка потоку журналів подій у різних форматах з підтримкою JSON, syslog та кастомних форматів.
2. Normalization & Masking: заміна змінних елементів на стандартизовані токени (IP → `<IP>`, числа → `<NUMBER>`). Очікується зменшення кількості унікальних токенів на 60-80%.
3. Context Window Management: створення ковзних вікон з 32-64 послідовних записів, адаптивний розмір залежно від частоти генерації журналів подій.
4. LSTM Sequence Encoder: Перетворення послідовностей токенів у векторні представлення з attention pooling та CPU-оптимізацією.
5. Feature Engineering: генерація додаткових ознак у чотирьох категоріях: часові, циклічні, категоріальні, семантичні.
6. XGBoost Classifier: фінальна класифікація на основі об'єднаного вектора [LSTM_embedding та tabular_features]

### *B. Архітектура енкодера Lstm*

LSTM перетворює вікно послідовностей службових повідомлень із журналів подій на компактне векторне представлення (embedding), що утримує ключові семантичні зв'язки між подіями. На вхід подається ковзне вікно з 32–64 записів одного сервісу, кожен запис попередньо нормалізується та токенізується. Далі токени проєктуються в 64-вимірні вектори (embedding layer), після чого одношаровий LSTM (hidden size 64–128) послідовно обробляє їх, зберігаючи часовий

порядок і залежності. На завершальному етапі формується набір шаблонів для виявлення аномалій на основі методу тренування уваги (attention pooling), який виділяє найінформативніші записи у кожному поточному пулі векторів та формує один 64-вимірний вектор інформаційного контексту конкретного сервісу.

Отриманий «семантичний відбиток» вікна журналів подій поєднується з табличними ознаками та передається у XGBoost для остаточного рішення. Така побудова дає змогу поєднати сильні сторони обох підходів: LSTM фіксує контекст і послідовність подій у журналах подій, а XGBoost додає точність та інтерпретованість на рівні табличних ознак [2]. Для production планується використовувати INT8-квантизацію а саме 8-бітні ваги, активації з калібруванням та динамічне батчування з таймаутом ~10 мс і оптимізовані бібліотеки лінійної алгебри оптимізовані під CPU. Це зменшує пам'ять і прискорює інференс, забезпечуючи стабільно низьку затримку для більшості запитів.

### *C. Інженерія ознак та інтеграція з XGBoost*

При обробці пропонується доповнити семантичний вектор від LSTM статистичними ознаками з журналів подій і подати об'єднаний вектор до XGBoost для остаточного рішення. Планується використовувати 4 групи ознак: часові (наприклад, error rate за 1h/24h, інтенсивність повідомлень, різноманітність шаблонів), циклічні (sin/cos часу доби та дня тижня для обліку періодичності), категоріальні (історичний anomaly rate для host_id/service_id, target encoding рівнів), а також семантичні (ентропія шаблонів у вікні та прапорець «новий шаблон»). Такий набір стисло описує динаміку, періодичність і контекст у журналах подій, зберігаючи інтерпретованість і стабільність на production-даних.

64-вимірний вектор LSTM об'єднується з табличними ознаками (часовими, циклічними, історичними профілями, а також семантичними). Об'єднаний вектор подається до XGBoost із CPU-оптимізованими налаштуваннями (tree_method='hist', помірна глибина, невелике eta, min_child_weight, subsample, colsample_bytree) для балансу точності та швидкодії.

Очікується, що запропонована гібридна модель перевершить «чистий» deep learning за точністю, поєднуючи контекст логів зі статистичними ознаками, водночас залишаючись достатньо швидкою: щонайменше 95% запитів обробляються швидше ніж за 50 мс на CPU.

## III. АДАПТИВНЕ НАЛАШТУВАННЯ ПОРОГІВ ТА ПРОФІЛЮВАННЯ

Статичні пороги однаково поводяться для всіх сервісів і часто генерують зайві сповіщення. Пропонується фільтрувати кожен сервіс окремо: підтримувати ковзні статистики (середнє, стандартне відхилення, історичний коефіцієнт помилок) за його сигналами та показниками з журналів подій. На основі цих профілів формується «нормальна зона» поведінки сервісу, від якої зручно відштовхуватися при визначенні аномалій.

Пороги планується визначати адаптивно: базовий рівень задається як межа, яку не перевищує приблизно 95% нормальних значень, із додатковим запасом, що є пропорційним типовому розкиду оцінок, далі він коригується за допомогою експоненційного згладжування з обмеженням швидкості зниження, щоб уникати надмірної чутливості. Для нових або малонавантажених сервісів застосовується накопичення власної статистики. Такий підхід персоналізує аналіз та зменшує рівень хибнопозитивних результатів на 30–50% без втрати чутливості до справжніх інцидентів.

## IV. ПОЯСНЮВАЛЬНІСТЬ ТА ПЛАН ЕКСПЕРИМЕНТІВ

Для пояснюваності пропонується легкий, практичний набір механізмів: для XGBoost — feature importance (gain/cover) та SHAP-значення; для LSTM — ваги attention, що локалізують у вікні журналів подій записи, які найбільше вплинули на рішення. В сповіщеннях планується відображати оцінки моделі, 3–5 ключових ознак із їх внеском, а також приклади шаблонів із відповідного інформаційного пулу, щоб оператор швидко відтворив контекст. Така комбінація має забезпечити достатню інтерпретованість без істотних накладних витрат на CPU і зробити причини спрацювань прозорими для експлуатаційних команд.

Цільові метрики практичні: очікується приріст F1 (гармонічне середнє між точністю й чутливістю) на 15–25% відносно baseline, переважна більшість запитів (95%) обробляється швидше за 50 мс на CPU, рівень хибнопозитивних результатів <5% на сервіс, recall критичних аномалій >90% і лінійна масштабованість до ~1000 сервісів [3].

## V. ОБМЕЖЕННЯ ТА ВИКЛИКИ

Запропонований підхід має низку ризиків, які планується враховувати під час експериментів і прототипування. По-перше, можливе зміщення даних між синтетичними, напівсинтетичними та production-журналами подій: розподіли шаблонів і частот можуть відрізнятися, що вплине на узагальнюваність. Для зниження ризику передбачаються регулярні перевірки стабільності (drift-моніторинг ознак та score), періодична перекалібровка та перенавчання за часово стратифікованими зрізами. По-друге, значний дисбаланс класів (1–5% аномалій) ускладнює оптимізацію — планується фокус на AUCPR (якість на дисбалансі класів), та частці коректних серед топ-N спрацювань і зваженому F1 з урахуванням дисбалансу. Також використовуємо ваги класів і тюнінг порога, щоб керовано обмінюватися між вищим recall і меншою кількістю сповіщень [4].

Окремий виклик — сталість послідовнісної моделі у розподіленому середовищі (кешування hidden-станів, відновлення після перезапусків, консистентність між інстансами) та жорсткі обмеження на затримку в CPU-середовищі. Це планується пом'якшувати через чіткий життєвий цикл станів (час життя, гарячий запуск, захист від розсинхронізації), динамічне батчування з таймаутом, стискання нейромережі до 8-бітних цілих чисел замість 32-бітних чисел з плаваючою комою, зі збереженням масштабів і нульових зсувів

та профілювання вузьких місць. Також враховується компроміс затримки проти точності: для сервісів із високими вимогами до затримки передбачається спрощена конфігурація (менші вікна, легші моделі), тоді як для критичних детекцій — більш «важкі» налаштування з пріоритетом якості.

## ВИСНОВКИ

У роботі пропонується гібридна архітектура LSTM + XGBoost для контекстно-орієнтованого виявлення аномалій у журналах подій із фокусом на CPU-середовища. Ключові ідеї включають поєднання семантичного відбитка послідовності службових повідомлень з їх табличними ознаками, адаптивні пороги для виявлення аномалій та запобігання хибних спрацювань з урахуванням профілю сервісу та практичної пояснюваності (attention для LSTM і SHAP/feature importance для XGBoost). Архітектура проєктується під низьку латентність на CPU (INT8, динамічне батчування, BLAS) і очікувано підвищує F1 на 15–25% проти baseline, забезпечуючи час відповіді < 50 мс для переважної більшості запитів (не менше 95%) на CPU та зберігаючи контрольовану кількість сповіщень.

Очікуване практичне значення полягає в економічній доцільності (робота без GPU), масштабованості до ~1000 сервісів, зменшенні операційного навантаження через адаптивні пороги та прозорі сповіщення, а також у простій інтеграції через стандартні інтерфейси. Водночас визнаються обмеження: можливе зміщення даних, дисбаланс класів, сталість у розподілених розгортаннях і компроміс між затримкою та якістю. Як напрями розвитку розглядаються federated learning для multi-tenant-сценаріїв, адаптація до дрейфу та інтеграція з існуючими моніторинг-системами.

## ПОСИЛАННЯ


[1] D. Zhan, K. Tan, L. Ye, X. Yu, H. Zhang, and Z. He, "An Adversarial Robust Behavior Sequence Anomaly Detection Approach Based on Critical Behavior Unit Learning," arXiv preprint arXiv:2509.15756, September 2025 https://arxiv.org/abs/2509.15756.

[2] K. Ozaki, Y. Uchino, and T. Imamura, "Ozaki Scheme II: A GEMM-oriented emulation of floating-point matrix multiplication using an integer modular technique," arXiv preprint arXiv:2504.08009, April 2025 https://arxiv.org/abs/2504.08009 .

[3] Y. Jin, K. Okamoto, K. Harada, A. Shibata, and K. Karube, "Similarity-Based Supervised User Session Segmentation Method for Behavior Logs," arXiv preprint arXiv:2508.16106, August 2025 https://www.arxiv.org/abs/2508.16106 .

[4] I. Popova and H. A. A. Gardi, "Credit Card Fraud Detection," arXiv preprint arXiv:2509.15044, September 2025 https://arxiv.org/abs/2509.15044 .