Міністерство освіти і науки України
Харківський національний університет імені В. Н. Каразіна
Навчально-науковий інститут комп'ютерних наук та штучного інтелекту
Кафедра комп'ютерних систем та робототехніки

До захисту допущено
Кафедрою комп'ютерних систем та робототехніки
протокол № __ від __ грудня 2025р.

завідувач кафедри _____ Максим ХРУСЛОВ
(підпис)

«___»_____ 2025 р.

Кваліфікаційна робота

здобувача другого (магістерського) рівня вищої освіти

«МЕТОДИ МАШИННОГО НАВЧАННЯ ДЛЯ ВИЯВЛЕННЯ АНОМАЛІЙ У СЕНСОРНИХ МЕРЕЖАХ ІОТ»

Спеціальність ***174 – Автоматизація, комп'ютерно-інтегровані технології та робототехніка***
Освітня програма ***Комп'ютеризовані системи управління та автоматика***

Виконавець _____ **М. Ю. ЩЕДРІН**
(підпис)

Науковий керівник _____ **Н.С. БАКУМЕНКО**
(підпис)

Харків – 2025

АНОТАЦІЯ

Пояснювальна записка до кваліфікаційної роботи складається зі вступу, трьох розділів, висновків, списку джерел та трьох додатків. Загальний обсяг роботи складає 71 сторінка, із яких 54 сторінки основної частини з 10 рисунками, 1 таблицею, 25 найменувань списку використаних джерел та трьома додатками.

Метою кваліфікаційної роботи є підвищення рівня захищеності мереж Інтернету речей шляхом створення гібридної системи виявлення вторгнень, що забезпечує мінімізацію пропущених атак при збереженні високої швидкодії.

Об'єкт дослідження – процеси обміну інформацією в сенсорних мережах IoT та методи виявлення аномальної поведінки в мережевому трафіку.

Предмет дослідження – моделі машинного та глибокого навчання, методи попередньої обробки часових рядів та програмні засоби побудови гібридних ансамблів на основі градієнтного бустингу та згорткових нейронних мереж.

Проблема, яка вирішується в роботі, полягає у забезпеченні надійного виявлення кібератак в умовах дисбалансу даних та ресурсних обмежень Edge-пристроїв шляхом подолання недоліків монолітних моделей.

Область застосування – системи кібербезпеки критичної інфраструктури, «Розумний дім» (Smart Home) та промисловий IoT, зокрема для моніторингу трафіку на мережевих шлюзах у реальному часі.

Ключові слова: *ІНТЕРНЕТ РЕЧЕЙ, IOT, КІБЕРБЕЗПЕКА, СИСТЕМА ВИЯВЛЕННЯ ВТОРГНЕНЬ, IDS, XGBOOST, 1D-CNN, STACKING, ГІБРИДНА МОДЕЛЬ, ВИЯВЛЕННЯ АНОМАЛІЙ, PYTHON.*

ABSTRACT

The explanatory note to the qualification paper consists of an introduction, three chapters, conclusions, a list of references, and three appendices. The total volume of the work is 71 pages, of which 54 pages represent the main part, containing 10 figures, 1 table, 25 items in the list of references, and three appendices.

The goal of the qualification paper is to enhance the security level of Internet of Things (IoT) networks by creating a hybrid intrusion detection system that ensures the minimization of missed attacks (false negatives) while maintaining high processing speed.

The object of study is the processes of information exchange in IoT sensor networks and methods for detecting anomalous behavior in network traffic.

The subject of study includes machine learning and deep learning models, methods for time-series preprocessing, and software tools for building hybrid ensembles based on gradient boosting and convolutional neural networks.

The problem solved in the work lies in ensuring reliable detection of cyberattacks under conditions of data imbalance and resource constraints of Edge devices by overcoming the limitations of monolithic models.

The area of application includes cybersecurity systems for critical infrastructure, Smart Home, and Industrial IoT (IIoT), specifically for real-time traffic monitoring on network gateways.

Keywords: *INTERNET OF THINGS, IOT, CYBERSECURITY, INTRUSION DETECTION SYSTEM, IDS, XGBOOST, 1D-CNN, STACKING, HYBRID MODEL, ANOMALY DETECTION, PYTHON.*

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

1D-CNN	-	Одновимірна згорткова нейронна мережа (One-Dimensional Convolutional Neural Network)
API	-	Інтерфейс прикладного програмування (Application Programming Interface)
CNN	-	Згорткова нейронна мережа (Convolutional Neural Network)
CPU	-	Центральний процесор (Central Processing Unit)
DDoS	-	Розподілена атака типу «відмова в обслуговуванні» (Distributed Denial of Service)
DL	-	Глибоке навчання (Deep Learning)
ECE	-	Очікувана помилка калібрування (Expected Calibration Error)
ETL	-	Процес вилучення, перетворення та завантаження даних (Extract, Transform, Load)
FN	-	Хибно-негативний результат (False Negative), пропуск атаки
FP	-	Хибно-позитивний результат (False Positive), хибна тривога
GAP	-	Глобальний середній пулінг (Global Average Pooling)
GPU	-	Графічний процесор (Graphics Processing Unit)
IDS	-	Система виявлення вторгнень (Intrusion Detection System)
IoT	-	Інтернет речей (Internet of Things)
MCC	-	Коефіцієнт кореляції Меттьюса (Matthews Correlation Coefficient)
ML	-	Машинне навчання (Machine Learning)
PR-AUC	-	Площа під кривою «Точність-Повнота» (Precision-Recall Area Under Curve)
ReLU	-	Функція активації (Rectified Linear Unit)
ROC-AUC	-	Площа під кривою помилок (Area Under Receiver Operating Characteristic Curve)
SOC	-	Центр моніторингу та реагування на інциденти інформаційної безпеки (Security Operations Center)
TN	-	Істинно-негативний результат (True Negative)

TP	-	Істинно-позитивний результат (True Positive)
XGBoost	-	Алгоритм екстремального градієнтного бустингу (eXtreme Gradient Boosting)
τ	-	Поріг класифікації (Threshold)
$P(x)$	-	Ймовірність події x

ВСТУП

У зв’язку зі стрімким розвиток концепції Інтернету речей (IoT) та лавиноподібним зростанням кількості підключених пристроїв, все більш актуальною постає проблема забезпечення кібербезпеки в умовах гетерогенних мереж. Цифровізація критичної інфраструктури, розумних міст та промисловості (Industry 4.0) призводить до того, що поверхня атак експоненційно розширюється, а методи зловмисників стають дедалі витонченішими. В умовах високої інтенсивності мережевого трафіку ключовим фактором надійності системи стає здатність засобів захисту оперативно та точно виявляти аномальну активність у реальному часі.

Актуальність роботи. Серед прикладних областей, де зазначена проблема є вкрай гострою, слід віднести захист сенсорних мереж IoT. Специфіка цієї галузі полягає у необхідності балансування між точністю детестування (мінімізація пропущених атак) та обмеженістю обчислювальних ресурсів периферійних пристроїв (Edge devices). Мережевий трафік є складною нелінійною сутністю, що залежить від режимів роботи пристроїв, протоколів передачі даних та зовнішніх впливів. Традиційні методи захисту, які використовуються у більшості систем (сигнатурний аналіз або прості порогові правила), демонструють низьку надійність в умовах появи нових типів атак (Zero-day attacks). Вони не здатні виявити приховані поведінкові патерни, що призводить до компрометації мережі та значних збитків. Водночас сучасні потужні методи глибокого навчання часто є занадто ресурсоємними для впровадження на слабких процесорах IoT. Отже, задача розробки автоматизованої системи, яка поєднує швидкодію статистичних методів (Machine Learning) із глибиною аналізу нейронних мереж (Deep Learning) через гібридну архітектуру, є вкрай актуальною науково-прикладною проблемою.

Метою дослідження є підвищення рівня захищеності мереж Інтернету речей шляхом створення та дослідження гібридної системи виявлення

вторгнень, що забезпечує мінімізацію помилок другого роду (пропущених атак) при збереженні високої швидкодії.

Об’єкт дослідження – процеси обміну інформацією в сенсорних мережах IoT та методи виявлення аномальної поведінки в мережевому трафіку.

Предмет дослідження – моделі, алгоритми та програмні засоби побудови гібридних систем виявлення вторгнень на основі ансамблювання градієнтного бустингу та згорткових нейронних мереж.

Методи дослідження: методи системного аналізу (для декомпозиції задачі виявлення вторгнень), методи машинного навчання (алгоритм XGBoost для аналізу табличних даних), методи глибокого навчання (архітектура 1D-CNN для аналізу часових послідовностей), методи математичної статистики (для попередньої обробки даних та Z-score нормалізації), методи оцінки якості класифікаторів в умовах дисбалансу (метрики MCC, PR-AUC).

Завдання дослідження:

1. Виконати аналіз існуючого науково-методичного апарата виявлення аномалій у мережах IoT та обґрунтувати переваги використання гібридного підходу, що поєднує аналіз потоків (flow-based) та аналіз пакетів (packet-based).
2. Розробити методику попередньої обробки даних та конструювання ознак, що включає захист від витоків даних (Time-based split) та формування часових вікон для аналізу динаміки з’єднань.
3. Спроектувати архітектуру гібридної системи, що використовує метод стекінгу (Stacking) для об’єднання прогнозів XGBoost та 1D-CNN, а також розробити алгоритм адаптивного підбору порогів для оптимізації чутливості системи.
4. Виконати програмну реалізацію системи та провести експериментальне порівняння ефективності базових моделей та запропонованого гібриду на реальному датасеті (ACI-IoT-2023), оцінивши їх точність та здатність до виявлення прихованих атак.

Наукова новизна одержаних результатів:

1. Удосконалено метод виявлення мережевих атак шляхом створення гібридного ансамблю, в якому, на відміну від існуючих підходів, поєднано статистичний аналіз агрегатів (XGBoost) та структурний аналіз послідовностей (1D-CNN), що дозволило зменшити кількість пропущених атак майже втричі.
2. Набуло подальшого розвитку застосування метрики MCC для оптимізації порогів прийняття рішень у системах IDS, що дозволило адаптувати чутливість моделей до специфіки дисбалансованого IoT-трафіку.

Практичне значення одержаних результатів полягає у розробці програмного комплексу, готового до впровадження на шлюзах IoT, який забезпечує виявлення загроз у реальному часі зі зниженим рівнем хибних спрацювань, що підтверджено експериментально на відкладеній вибірці.

РОЗДІЛ 1.
ТЕОРЕТИЧНІ ОСНОВИ ТА МЕТОДОЛОГІЧНИЙ АПАРАТ ВИЯВЛЕННЯ АНОМАЛІЙ У IOT

1.1. Постановка задачі та актуальність систем виявлення вторгнень (IDS)

Сенсорні мережі IoT - це великі розподілені системи з тисячами вузлів, що збирають та передають телеметрію через різнорідні протоколи (Wi-Fi, BLE, ZigBee/Thread, LoRaWAN, промислові шини), часто через шлюзи на периферії (edge). Більшість вузлів енергообмежені, мають спрощені стеки безпеки і працюють у середовищі з постійними змінами трафіку [3]. Це робить їх вразливими до широкого спектра загроз: від примітивних (сканування портів, brute-force автентифікації, ARP-spoofing) до розподілених DDoS і ботнет-кампаній, а також більш витончених каналів прихованого обміну в періодичній телеметрії [21; 24]. У таких умовах ключову роль відіграють системи виявлення вторгнень (IDS), які мають фіксувати відхилення від «нормальної» поведінки до того, як відбудеться порушення конфіденційності, цілісності або доступності сервісів.

Формалізація задачі. Завдання IDS формалізується як бінарна класифікація у часовому домені. Нехай спостерігається послідовність мережевих подій $X = \{x_1, x_2, \dots, x_N\}$, де x_i - опис події на одному з рівнів представлення (пакет, потік або агреговане часове вікно), а $y_i \in \{0,1\}$ - мітка «benign/attack». Потрібно побудувати оцінювач ризику $\hat{P}(y_i = 1|x_i)$, який наближує $P(y_i = 1|x_i)$, та правило прийняття рішень τ із порогом $\hat{y}_i = 1$ якщо $\hat{P}(y_i = 1|x_i) \geq \tau$. У класичній постановці з мінімізацією ризику τ залежить від вартостей помилок C_{FP}, C_{FN} та базових частот класів. У практиці IDS його обирають на валідації під цільову метрику (часто MCC або баланс precision/recall), а потім фіксують для всіх звітних таблиць і графіків.

Предмет даних і ознак. Для IoT-IDS виділяють три комплементарні рівні: flow-рівень (агреговані статистики потоків, такі як тривалість, пакето-/байторейти, інтервали, дисперсії), що зручний для табличного ML; payload/packet-рівень (байтові послідовності та похідні з вмісту пакетів [10]), що є природним для згорткових/послідовних мереж; і PCAP («сирий» рівень для кастомної екстракції ознак та ретроспективного аудиту). Вибір рівня визначає клас моделей і витрати інференсу: табличні ансамблі мають низьку латентність на CPU, тоді як послідовні CNN фіксують локальні часові патерни, які важко замінити простими агрегатами.

Типи аномалій. У часових мережевих даних розрізняють точкові аномалії (раптові сплески запитів), контекстуальні (зміни, що відхиляються від локального контексту, наприклад нічна активність «нічим не зайнятого» сенсора) та колективні (послідовності, які в сукупності формують підозрілу поведінку, хоча окремі події виглядають нормальними). Для IoT особливо релевантні контекстуальні й колективні аномалії, що мотивує використання часових вікон і 1D-CNN.

Ключові виклики. Система IDS стикається з низкою ключових викликів. Перший - це дисбаланс класів, оскільки частка атак (prevalence) може бути дуже малою або сильно варіюватися. Тому accuracy та навіть ROC-AUC можуть вводити в оману, і пріоритет надається PR-AUC, MCC і (у мультикласі) macro-F1. Другий виклик - це нестаціонарність і концепт-дрейф, коли розподіли ознак і міток змінюються з часом (оновлення прошивок пристроїв, зміни політик мереж, сезонність трафіку), що спричиняє деградацію моделі без контролю калібрування та періодичної перевірки порога. Третій - обмеження ресурсів на edge-вузлах, де доступні лічені мілісекунди і мегабайти пам’яті, що вимагає легких моделей або двостадійних схем (staged-inference), де «важкий» аналіз запускається рідко. Четвертий - запобігання витоку інформації: будь-які рішення щодо скейлінгу/відбору

ознак, гіперпараметрів і порогів мають ґрунтуватися виключно на train/val, без «погляду» в test, інакше оцінювання буде некоректним.

Парадигми навчання. При реалізації IDS розрізняють керовані, безнаглядні та напівнаглядні постановки. Керовані моделі (класифікація) показують найвищу якість, коли якісна розмітка доступна, що є типовим випадком для відкритих датасетів. Безнаглядні підходи (one-class, автоенкодери) корисні для фільтрації новизни у виробництві, де мітки обмежені. Напівнаглядні схеми комбінують обидва підходи та зручні при обмежених анотаціях, але потребують строгої валідації, щоб не «підмішати» тестову інформацію.

Процес прийняття рішень. IDS повинна повертати не лише бінарний ярлик, а й оцінку ризику з коректною калібровкою. Калібрована ймовірність (наприклад, після Platt або ізотонічного калібрування) дозволяє чесно будувати PR-криві, робити стекінг різнорідних моделей на узгодженій шкалі та гнучко налаштовувати пороги під операційні обмеження. Оскільки вартість FP (помилкові спрацювання, які «шумлять» SOC і знижують довіру) і FN (пропуски атак, які коштують дорожче) неоднакова, звідси випливає раціональний вибір порога за валідованим критерієм (MCC/Youden), який фіксують для всіх матриць плутанини, таймлайнів і метрик на тесті.

Валідація і відтворюваність. Через часову природу трафіку крос-валідація з випадковим перемішуванням неприйнятна. Натомість використовують суворий часовий спліт: тренувальні періоди (train) передують валідації (val), а тест (test) – це «майбутнє». Маніфести розбиття, фіксація seed і збереження конфігів забезпечують відтворюваність. Важливо, що скейлер/нормалізація, відбір ознак, підбір гіперпараметрів і калібрування виконуються тільки на train/val і застосовуються до test без переоцінки. Такий протокол запобігає витоку й дає чесне уявлення про робастність до зсувів.

Рівень моделювання. На flow-рівні найчастіше обирають ансамблеві дерева, як-от XGBoost/LightGBM, через їх природну здатність моделювати нелінійні взаємодії, підтримку клас-ваг і низьку латентність на CPU. На послідовному рівні ефективні 1D-CNN із невеликими ядрами (3–7), що виловлюють локальні часові мотиви, а також їхні розширення з LSTM/attention. Об’єктивно ці лінії бачать різне: flow-агрегати добре відтворюють «статистику сеансу», тоді як CNN фіксує короткі характерні патерни (бурсти, пульсації, «сигнатури» у часовому домені). Звідси випливає необхідність гібридизації: late fusion (stacking) об’єднує калібровані імовірності в мета-класифікаторі; staged-inference дає потрібну продуктивність, коли CNN застосовується лише до «пограничних» прикладів; early fusion подає ембединг CNN у бустинг як додаткові ознаки.

Метрики та контроль якості. У дисбалансі пріоритетні PR-AUC (глобальна оцінка якості щодо precision/recall із базовою лінією, що дорівнює частці позитивів у тесті) та MCC (симетрична кореляційна міра на TP, TN, FP, FN), тоді як ROC-AUC використовується як додаткова (особливо корисна для порівняння ранжувальних властивостей на стабільних базових частотах). Для мультикласу використовується macro-F1 і похідні macro-precision/recall. Усі матриці плутанини й таймлайни мають будуватися на тому самому порозі, що використовувався для тестових метрик, інакше виникнуть «роз’їзди» між цифрами й графіками.

Експлуатаційні аспекти. Для впровадження в реальних мережах необхідні інтерпретованість (наприклад, SHAP для табличної моделі, салієнт-мапи для CNN), що скорочує час аналізу інцидентів і допомагає налаштовувати політики; продуктивність (час інференсу й пропускна здатність у пікові періоди), що часто досягається staged-схемами й оптимізаціями (ONNX, квантування, батчування); та моніторинг дрейфу (регулярна перевірка калібрування та адаптація порогів). Важливо планувати «контури оновлення» моделі (перенавчання на нових періодах, валідаційні

гейти, A/B-перевірки), аби не втратити стабільність при зміні парку пристроїв або конфігурацій.

Проміжний підсумок. Завдання IDS у сенсорних мережах IoT - це кероване виявлення аномалій у часових даних з жорсткими обмеженнями на ресурси і високими вимогами до робастності. Коректна постановка включає вибір рівня представлення (flow vs payload), відповідної моделі (ансамблі vs CNN), чутливих до дисбалансу метрик (PR-AUC, MCC), калібрування з уніфікованим порогом і суворий часовий протокол оцінювання. На такій теоретичній базі будується подальший вибір архітектур та експериментальна програма.

1.2. Огляд сучасних підходів до аналізу мережевого трафіку

Аналіз трафіку для виявлення аномалій у сенсорних мережах IoT історично розвивався від правил і сигнатур до статистики та машинного навчання, а нині - до глибинних і гібридних підходів [12]. Особливості IoT (різнорідність протоколів, шифрування, енерго- та ресурсні обмеження, нестаціонарність) зумовлюють необхідність поєднання кількох парадигм замість «універсальної» єдиної моделі.

Сигнатурні та правилові системи. Класичні NIDS/HIDS (типу Snort/Suricata) покладаються на каталоги сигнатур відомих атак або експертні правила DPI. Перевагами є простота, пояснюваність і низький рівень хибних спрацювань на знайомих патернах. Водночас, недоліками є слабка здатність виявляти невідомі («zero-day») або модифіковані атаки, значні витрати на підтримку правил і чутливість до шифрування, яке приховує корисний вміст пакетів. Для IoT з великою різноманітністю стеків (BLE, ZigBee/Thread, LoRaWAN, промислові шини) й частим TLS-шифруванням суто сигнатурний підхід стає недостатнім і потребує підсилення моделями «поведінки».

Статистичні детектори й виявлення змін. Наступним щаблем є порогові процедури імовірнісного контролю, такі як CUSUM/EWMA для детекції

зсувів, непараметричні тести на зміну розподілу, методи change-point detection і ковзні вікна. Вони добре реагують на різкі аномалії (переповнення, spike у частоті пакетів, нетипові інтервали), майже не потребують розмітки та легко реалізуються на edge. Проте такі детектори чутливі до сезонності й «повільних» дрейфів, вимагають ретельної нормалізації та періодичного оновлення базових рівнів. У гетерогенних IoT-сценаріях із багатьма джерелами шуму ці детектори часто використовують як перший «швидкий фільтр» перед точнішими ML-модулями.

Класичне кероване ML на flow-ознаках. Коли доступна розмітка, наступним природним кроком є моделювання на агрегованих flow-ознаках (тривалість, кількість пакетів/байтів, швидкості, ентропії, дисперсії інтервалів, прапорці протоколів, співвідношення напрямків). Лінійні моделі (логістична регресія), SVM і дерева/ліси забезпечують сильну базу; у табличних задачах градієнтний бустинг над деревами (XGBoost/LightGBM) став де-факто стандартом завдяки здатності моделювати нелінійні взаємодії, підтримці клас-ваг і швидкому інференсу на CPU [6]. Такі моделі легко пояснювати через ранжування важливостей і XAI-підходи (SHAP), що важливо для інцидент-відповіді та налаштування політик. Типові підводні камені включають дисбаланс класів (потрібні клас-ваги/семплінг і пороговий аналіз), калібрування ймовірностей, а також ризик витоку під час підбору ознак/скейлера: усе це слід робити винятково на тренувальних періодах із подальшим «заморожуванням» на тест [16; 25].

Послідовні та глибинні моделі. Коли важливо побачити часові патерни і «мотиви» у порядку пакетів або вмісті, застосовують 1D-CNN, RNN/GRU/LSTM, TCN (temporal convolutional networks) [20] і легкі варіанти self-attention. 1D-CNN добре вловлює локальні структури (бурсти, періодичні пульсації, короткі сигнатури) і має низьку латентність на інференсі; архітектурно це кілька блоків Conv1D–BN–ReLU із Global Average Pooling та щільним виходом [13; 22]. RNN/LSTM/GRU зручні для довгих залежностей,

але повільніші й важче масштабуються. Attention/Transformers дають найвищу виразність, проте «коштують» пам’яті, тому в IoT практиці частіше використовують спрощені/зріджені уваги або змішані CNN-LSTM [17; 23]. Крім того, схеми Autoencoder/seq2seq і one-class релевантні там, де бракує міток: вони навчаються на «нормі» і підсвічують новизну. Проблеми глибинних підходів включають потребу у великих і репрезентативних даних, ризик *overconfidence* (тому потрібне післятренувальне калібрування), чутливість до зсуву домену та вищу вартість інференсу на периферії. Перевагами є краща чутливість до контекстуальних і колективних аномалій, які неможливо надійно зловити лише агрегатами [7].

Робота під шифруванням і «видимі» ознаки. Шифрування (TLS/DTLS) робить недоступним вміст, що знижує цінність «payload-CNN». У таких умовах використовують метадані й побічні ознаки: довжини, часові інтервали, напрямки, сигнатури рукостискань (наприклад, JA3/JA4 для TLS), властивості сесій і статистики повторюваності. Це підсилює роль flow-аналізу й відповідно - табличних ансамблів. У змішаних середовищах (частина трафіку шифрована, частина - ні) ефективні схеми з динамічним вибором представлення: якщо є payload - спрацьовує CNN-гілка; інакше рішення ухвалює flow-модель.

Гібридні схеми та злиття представлень. Досвід впроваджень показує, що поєднання табличного й послідовного аналізу стабільніше на часових тестах і при доменному зсуві [2; 9]. Пізнє злиття (Late fusion / stacking) передбачає, що калібровані ймовірності P_{xgb} та P_{cnn} подаються на мета-класифікатор (часто логістична регресія) за out-of-fold протоколом - це узгоджує «шкали» і зменшує ризик витоку. Метод Staged-inference використовує швидкий XGBoost для відсіювання «очевидних» прикладів, а CNN дораховує лише «пограничні», що зменшує час і вартість інференсу - критично для edge. Нарешті, раннє/ознакове злиття (Early/feature fusion) конкатенує ембединг CNN (після GAP) із flow-ознаками та подає їх у бустинг; це працює, якщо

ембединг додає інформацію, якої бракує агрегатам. Гібриди, як правило, підвищують PR-AUC і MCC саме завдяки комплементарності представлень.

Нестаціонарність та концепт-дрейф. У реальних мережах змінюються прошивки, топології, політики QoS, кількість пристроїв, і навіть режим доби/сезону - усе це змінює розподіли ознак і базові частоти класів. Виділяють раптовий, поступовий, інкрементальний і рецидивний дрейф. Для роботи «на потоці» корисні механізми виявлення дрейфу (DDM/EDDM/ADWIN), ковзні вікна тренування, онлайн-оновлення порога, періодичне пере-калібрування (ізотонічне/Platt), а також ансамблі «вікових» моделей із вагами, що спадають у часі. Важливо розводити два рівні адаптації: ранжування (форма $P(y|x)$) і порогова політика (τ), адже за зсуву базових частот часто достатньо адаптувати саме поріг.

Дисбаланс класів і оцінювання. В IoT *prevalence* аномалій змінний, тому Accuracy і навіть ROC-AUC можуть бути некоректними індикаторами. Більш інформативні: PR-AUC (з обов'язковою базовою лінією - часткою позитивів у тесті) [18] і MCC (симетрична кореляційна міра на TP/TN/FP/FN). Для мультикласу використовують macro-F1/macro-precision/macro-recall; у таблицях і на таймлайнах необхідно застосовувати єдиний валідований поріг, інакше цифри «роз'їдуться». Калібрування покращує узгодженість PR-кривих і роботу стекінгу, а *reliability-діаграми* з ECE/Brier дозволяють контролювати дрейф «впевненості» моделі.

Адверсарна стійкість і зловмисник. У мережевих задачах можливі ухильні атаки на детектор (маскування поведінки під «фон», «розрідження» потоку, імітація нормального ритму). Ансамблі дерев менш чутливі до мікрозмін ознак, але вразливі до спотвореного препроцесингу; CNN схильні до «піксельних» (байтових) мікроправок. Захист включає регуляризацію, детектування відхилень у просторах ембедингів, ensembling і контроль стабільності ознак (наприклад, інваріантів до легальних пермутацій).

Експлуатація на периферії. На edge-вузлах «ціна» кожної мілісекунди критична. Практично це означає, що Staged-inference використовується як базова стратегія. Також застосовується стиснення моделей (квантування, прунінг, дистиляція), ONNX-експорт і векторизація. Важливими факторами є облік *batch-less* режимів (запити одиничні, кешів мало) та контроль пам'яті для CNN (особливо з attention) і фолбек на чисто flow-аналіз. Важливо також планувати канал оновлень: безпечне розгортання нових ваг і «канаркові» перевірки на невеликих сегментах мережі.

Вибір підходу залежить від сценарію розгортання. Якщо трафік переважно шифрований і потрібна низька латентність, першим номером іде XGBoost на flow-ознаках, а CNN виступає як друга стадія для «пограничних» випадків. Якщо доступний payload і важливі поведінкові патерни, 1D-CNN дає відчутний виграш, особливо за контекстуальних/колективних аномалій. За частих зсувів і різких змін рекомендується комбінувати періодичне пере-калібрування, адаптацію порога, «ковзні» оновлення та гібрид представлень. Коли потрібна пояснюваність для SOC, краще використовувати ансамблі дерев із SHAP і зрозумілими таймлайнами рішень.

Таким чином, сигнатури забезпечують «статичний» захист, статистика - швидкий фільтр, класичне ML на flow - стабільну основу з пояснюваністю та низькою латентністю, глибинні моделі - чутливість до складних часових патернів. Найкращі практики для IoT сходяться до гібридних конвеєрів із суворою часовою валідацією, калібруванням і уніфікованим порогом: саме вони демонструють баланс якості, вартості та стійкості до нестаціонарності.

1.3. Обґрунтування вибору моделей та критеріїв оцінювання

Мотивація подвійної лінії моделей. Дані IoT-трафіку мають дві комплементарні «проекції»: агреговані потоки (*flow*-фічі: тривалість, кількість пакетів/байтів, інтервали, дисперсії, співвідношення напрямків, агрегати за протоколами) та часові послідовності (пакети або впорядковані у часі вікна потоків/байтів). Перша проекція добре описується табличними моделями, де

важлива здатність фіксувати нелінійні взаємодії ознак, справлятися з пропусками та дисбалансом класів і працювати швидко на CPU. Друга, натомість, вимагає локально-часового аналізу й нечутливості до зсувів у часі, що природно дають згорткові мережі над 1D-послідовностями [11]. Через те ми свідомо поєднуємо XGBoost на *flow*-рівні та 1D-CNN на часових вікнах: моделі «бачать» різне й у сукупності закривають ключові патерни аномалій.

Градієнтний бустинг над деревами рішень, як-от XGBoost, є надзвичайно ефективним. По-перше, він добре моделює нелінійні взаємодії й взаємозалежності між ознаками [4] (наприклад, спільний вплив тривалості, *burstiness*, частки «висхідних» пакетів тощо). По-друге, він стійкий до пропусків та шуму, підтримує *class weights/scale_pos_weight* і семплінг, що критично важливо в умовах дисбалансу. По-третє, XGBoost має низьку латентність інференсу на CPU (що важливо для *edge*-обчислень) і простий *деплой*. Нарешті, він сумісний із XAI: SHAP забезпечує глобальні/локальні пояснення (які ознаки «потягнули» рішення) [14]. Обмеження XGBoost полягає у відсутності «вбудованої» часової інваріантності: він працює з уже агрегованою статистикою й може втрачати тонкі короткі патерни у послідовностях.

Згорткові фільтри фіксують локальні часові мотиви (сплески, періодичні пульсації, характерні «сигнатури» взаємодій), які є інваріантними до невеликих зсувів у часі. Базова архітектура (кілька блоків Conv1D–BatchNorm–ReLU + Global Average Pooling + щільний вихід) забезпечує: інваріантність до зсувів і витяг корисних локальних шаблонів без ручного інжинірингу; помірну вартість інференсу (на відміну від довгих RNN/*attention*), якщо тримати невеликі ядра (3–7) і 32–128 фільтрів; а також можливість працювати на *payload/packet* або на вікнах потоків (коли *payload* шифрований). Виклики 1D-CNN полягають у потребі в якісному формуванні вікон і калібруванні виходів (схильність до «*overconfidence*»), а також контролі ресурсу на периферії.

Оскільки табличний і послідовний погляди є комплементарними, ми використовуємо три схеми злиття. Пізнє злиття (Late fusion / stacking) передбачає, що калібровані ймовірності P_{xgb} і P_{cnn} подаються у мета-класифікатор (логістична регресія або легкий бустинг). Критично важливо збирати *out-of-fold (OOF)* прогнози на валідації, щоб мета-рівень не бачив «тренувальної правди» й не «запам'ятав» витоки. Staged-inference використовує швидкий XGBoost для первинного скринінгу, а далі лише приклади з «сірої зони» (наприклад, $P_{xgb} \in [\tau - \delta, \tau + \delta]$) відправляються в 1D-CNN. Це дає економію обчислень і зменшує латентність у середньому. Оцінимо витрати: якщо частка «пограничних» q, то середній час $\bar{t} \approx T_{xgb} + q \cdot T_{cnn}$ - зазвичай $q \ll 1$, отже виграш суттєвий. Раннє/ознакове злиття (Early/feature-fusion) конкатенує вектор ознак із 1D-CNN (ембединг після GAP) із *flow*-ознаками й подається у XGBoost. Працює, коли ембединг додає інформацію, якої не вистачає агрегатам. Важливо: коректна часова синхронізація ембедингу й *flow*-вікна, щоб не порушити причинність.

Протокол калібрування та порогів. Усі три схеми виграють від каліброваних імовірностей. Для кожної базової моделі на валідації застосовуємо Platt або ізотонічне калібрування [8] (вибір залежить від обсягу даних: Platt стійкіший на малих валідаціях, ізотоніка гнучкіша на великих). Якість калібрування контролюємо *reliability*-діаграмами, ECE та Brier-score. Після калібрування обираємо єдиний поріг τ за критерієм, узгодженим з ризиком: MCC-оптимізація (пошук τ, що максимізує MCC на валідації) або Youden's J (для бінару, коли важлива збалансованість чутливість/специфічність). Обраний τ заморожується і застосовується до тесту, таймлайнів, матриць плутанини та всіх табличних підсумків - це усуває «роз'їзд» між числами та графіками.

Оцінювання в умовах дисбалансу вимагає використання метрик, чутливих до малої частки позитивного класу. PR-AUC дає інтегральну оцінку компромісу *precision–recall* у дисбалансі. Базова лінія - частка позитивів у

тесті (*prevalence*), її слід явно проводити на графіках, інакше порівняння вводить в оману. MCC (коефіцієнт кореляції Метьюса) є симетричною кореляційною мірою, що враховує всі чотири складові матриці плутанини: TP, TN, FP, FN.

$$\mathrm{MCC} = \frac{\mathrm{TP}\times\mathrm{TN}-\mathrm{FP}\times\mathrm{FN}}{\sqrt{(\mathrm{TP}+\mathrm{FP})(\mathrm{TP}+\mathrm{FN})(\mathrm{TN}+\mathrm{FP})(\mathrm{TN}+\mathrm{FN})}}, \quad (1.1)$$

Вона стійка до дисбалансу та коректно оцінює «загальну узгодженість» класифікатора [5]. macro-F1 (для мультикласу) або F1 для *«attack»* у бінарі фокусується на балансі *precision/recall* і чутлива до FN, що критично для безпеки. ROC-AUC залишаємо як додаткову: вона корисна для грубого ранжування, але в сильному дисбалансі може бути надто оптимістичною; основні висновки будуємо за PR-AUC/MCC.

Суворий часовий протокол валідації. Для IoT-трафіку крос-валідація з випадковим перемішуванням неприйнятна. Використовуємо train → val → test у хронологічному порядку (за днями/сесіями), без перетину. Будь-який препроцесинг (нормалізація, відбір ознак), калібрування, підбір гіперпараметрів та порогів виконується лише на *train/val* і «заморожено» на *test*. Фіксація *seed*, збереження маніфестів сплітів, версій бібліотек і конфігів гарантує відтворюваність.

Очікувані властивості моделей. Моделі мають комплементарні очікувані властивості. XGBoost забезпечує високу стабільність і швидкість на *flow*-ознаках; це сильна база для SOC завдяки SHAP-пояснюваності, але він схильний до FN на «поведінкових» патернах, які не відбились у агрегатах. 1D-CNN демонструє кращу чутливість до коротких/контекстуальних патернів, що допомагає зменшити FN; вона потребує калібрування та контролю ресурсу, а також чутлива до якості формування вікна й нормалізації. Гібрид є стабільнішим на часовому тесті, особливо за доменного зсуву; у *staged-inference* дає економію інференсу (CNN «стріляє» рідше), у *stacking* - покращує

PR-AUC/MCC завдяки узгодженню сигналів; *feature-fusion* корисний, коли ембединг CNN додає нову інформацію.

Особливостями реалізації є необхідність налаштування max_depth, eta, n_estimators, subsample, colsample_bytree; робота з дисбалансом через class-weights/scale_pos_weight; контроль пере-навчання через ранню зупинку на валідації. На CNN-лінії: важливий підбір довжини вікна *k* (16–256), ядер (3–7), кількості фільтрів (32–128), регуляризація (*Dropout/weight-decay*), рання зупинка; нормалізація має бути інваріантна до витоку (статистики лише з *train*). Для stacking: обов'язковий збір *OOF-ймовірностей* на валідації для обох базових моделей; калібрування до подачі в мета-класифікатор; простий мета-рівень (*логрег*) часто достатній. Для staged: необхідний вибір «сірої зони» навколо порогу (наприклад, $P \in [\tau - \delta, \tau + \delta]$) як *гейт*; оцінка q (частки прикордонних) на валідації, щоб прогнозувати навантаження; можливий динамічний *гейт* під цільову швидкодію. Для репортингу: одна й та сама версія порогу в усіх артефактах (таблиці, таймлайни, матриці плутанини), PR-криві з базовою лінією *prevalence, reliability-плоти* з ECE/Brier.

Вибір XGBoost (*flow*) + 1D-CNN (послідовності) обґрунтований різними природами інформації на цих рівнях. Гібридні схеми (*stacking, staged-inference, feature-fusion*) дозволяють поєднати їхні сильні сторони: підвищити PR-AUC/MCC, зменшити FN на складних патернах і водночас утримати латентність та вартість інференсу у межах, сумісних з *edge*-сценаріями. Критично важливо дотримуватися калібрування, єдиного порогу та суворого часового протоколу - лише так отримані висновки будуть коректними й відтворюваними.

Висновки до розділу 1

У розділі сформовано теоретичну та методологічну основу задачі виявлення аномалій у сенсорних мережах IoT. Показано, що специфіка домену - дисбаланс класів, нестаціонарність і концепт-дрейф, різнорідність протоколів та обмеження обчислювальних ресурсів на *edge* - унеможливлює

«єдину» універсальну модель і вимагає поєднання комплементарних підходів. Обґрунтовано вибір двох ліній представлення даних: flow-рівня (агреговані статистики потоків) і часових послідовностей (пакети/вікна), що відображають різні аспекти мережевої поведінки. Відповідно обрано дві базові моделі: XGBoost як сильний табличний базис із низькою латентністю та сумісністю з XAI, і 1D-CNN як ефективний детектор локальних часових патернів, здатний зменшувати пропуски поведінкових атак.

Аргументовано доцільність гібридизації. Зокрема, late-fusion (stacking) використовується для узгодження сигналів на каліброваних імовірностях, staged-inference - для ресурсно ощадного інференсу на периферії, а early/feature-fusion застосовується, коли ембединг CNN додає нову інформацію до *flow*-ознаків. Для коректного порівняння моделей і відтворюваної аналітики закріплено комплекс метрик, стійких до дисбалансу (PR-AUC, MCC, у мультикласі - macro-F1), із використанням ROC-AUC як допоміжної. Наголошено на необхідності калібрування ймовірностей (Platt/ізотонічне), контролю надійності (ECE, Brier) та застосування єдиного валідованого порогу у всіх таблицях, матрицях плутанини й таймлайнах.

Ключовою умовою оцінювання визначено суворий часовий протокол train/val/test без перетинів сесій і без витоків препроцесингу на тест: усі рішення щодо скейлінгу, відбору ознак, гіперпараметрів, калібрування та порогів приймаються виключно на *train/val* і «заморожуються» для тесту. У підсумку розділ фіксує: постановку задачі IDS як керованого виявлення аномалій у часових даних, мотивований вибір XGBoost і 1D-CNN та їхніх схем злиття, узгоджений набір метрик і порогову політику, а також вимоги до відтворюваності.

РОЗДІЛ 2.
АРХІТЕКТУРА, ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ІНЖЕНЕРІЯ ДАНИХ

2.1. Підготовка набору даних та процедури препроцесингу

Характеристика та вибір набору даних

Як основу для експериментальних досліджень обрано сучасний датасет ACI-IoT-2023 [1; 6], який відображає реалістичний профіль трафіку в мережах Інтернету речей. Цей вибір є критично важливим, оскільки, на відміну від застарілих наборів (NSL-KDD, UNSW-NB15), ACI-IoT-2023 містить актуальні патерни атак, характерні для 2023–2024 років, включаючи сканування вразливостей, DDoS-атаки різного типу (ICMP, UDP, TCP SYN Flood) та спроби несанкціонованого доступу до IoT-пристроїв. Це забезпечує вищу релевантність результатів дослідження до сучасних загроз. Вихідні дані представлені у форматі CSV і містять два рівні деталізації: агреговані потоки (*Flow features*) та розширені метадані пакетів. Для забезпечення чистоти експерименту та уникнення перенавчання на специфічній топології мережі було виконано фільтрацію ознак.

Очищення та обробка ознак

Програмний модуль препроцесингу (*diplom_shedrin.py*) реалізує конвеєр перетворення “сирих” даних у формат, придатний для подачі в моделі машинного навчання.

По-перше, виконано видалення ідентифікаторів та шумів. З метою гарантування, що модель вивчає саме поведінкові патерни атак, а не “запам’ятовує” конкретні адреси зловмисників, вилучено поля flow_id, src_ip, dst_ip, а також timestamp (останнє використовується лише для сортування).

По-друге, проведено обробку категоріальних змінних. Для полів з низькою кардинальністю (протоколи, прапори стану) застосовано кодування цілочисельними індексами. Натомість, ознаки з надмірною кількістю

унікальних значень (понад 20), які не несуть узагальнюючої інформації, було вилучено для зменшення розмірності моделі.

По-третє, виконано бінаризацію міток. Цільова змінна label трансформована у бінарний вектор $y \in \{0,1\}$, де 0 відповідає легітимному трафіку (“Benign”), а 1 - будь-якому типу атаки, що спрощує задачу до бінарної класифікації для первинної оцінки.

Логарифмування та кліпінг викидів

Мережевий трафік характеризується розподілом із “важкими хвостами” (*heavy-tailed distribution*): більшість пакетів мають малий розмір, але зустрічаються рідкісні події з аномально великими значеннями байтів або тривалості з’єднання. Для стабілізації градієнтного спуску (для CNN) та покращення побудови дерев (для XGBoost) застосовано два методи. Перший - кліпінг (Clipping): значення, що перевищують 99.9-й перцентиль, обрізаються до граничного рівня, що усуває надмірний вплив екстремальних викидів на моделі. Другий - логарифмування: до ознак, що мають експоненційний розподіл (кількість байтів total_bytes, пакетів total_pkts, тривалість duration), застосовано перетворення $\log(1 + x)$ [10]. Це наближає розподіл до нормального та покращує збіжність моделей [10].

Нормалізація, збереження стейту та запобігання витоку

Для приведення всіх ознак до єдиного масштабу використано стандартизацію (Z-score normalization). Розрахунок нормованого значення z виконується за формулою:

$$z = \frac{x-\mu}{\sigma}, \tag{2.1}$$

Критично важливою вимогою, особливо в задачах кібербезпеки, є запобігання витоку інформації (Data Leakage). Тому об’єкт StandardScaler ініціалізується та навчається (*fit*) виключно на тренувальній частині вибірки. Параметри μ (середнє) та σ (стандартне відхилення) зберігаються у файл

scaler.pkl і використовуються для трансформації валідаційного та тестового наборів. Це емулює реальний режим роботи IDS, коли параметри нових, невідомих даних невідомі заздалегідь.

Формування часових послідовностей (для 1D-CNN)

Для подачі даних на вхід одновимірної згорткової мережі реалізовано механізм формування ковзних вікон (*rebuild_sequences_with_index.py*). Вхідний тензор для 1D-CNN має розмірність (N, L, D), де N - кількість зразків, $L = 32$ - довжина послідовності (історія з 32 попередніх пакетів/потоків), а D - кількість ознак. Довжина $L = 32$ була обрана емпірично як оптимальний компроміс між здатністю захоплювати довготривалі залежності та обчислювальною вартістю на *edge*-пристроях. Крім того, збережено вектори індексів (idx_val.npy, idx_test.npy), що дозволяють однозначно співставити вихід CNN (який формується лише для повних вікон) із виходом XGBoost (який працює з кожним рядком окремо). Ця синхронізація є необхідною основою для коректної роботи гібридного мета-класифікатора.

Сувора часова сегрегація (Time-based Split)

Замість традиційної крос-валідації з випадковим перемішуванням (*ShuffleSplit*), яка є неприйнятною для часових рядів через порушення причинно-наслідкових зв’язків, застосовано хронологічне розбиття. Спочатку дані відсортовано за полем timestamp. Потім послідовно виділено три незалежні періоди: 1. Перші 60% вибірки формують тренувальний набір (*train*). 2. Наступні 20% - валідаційний набір (*val*) для налаштування гіперпараметрів та калібрування. 3. Останні 20% - відкладений тестовий набір (*test*) для фінальної оцінки.

Такий підхід гарантує, що модель тестується на “майбутніх” атаках, яких не було в навчанні, що забезпечує чесну оцінку її узагальнюючої здатності в умовах концепт-дрейфу.

2.2. Налаштування моделей та гібридні схеми

Розроблена система базується на ансамблевому підході, що поєднує два різнорідні алгоритми машинного навчання. Така гетерогенність дозволяє компенсувати слабкі сторони одного методу сильними сторонами іншого, підвищуючи загальну надійність та точність детектора. Нижче наведено детальну специфікацію конфігурацій та математичне обґрунтування обраних гіперпараметрів.

Конфігурація та оптимізація градієнтного бустингу (XGBoost)

Для аналізу статистичних ознак (*Flow-based features*) використано алгоритм XGBoost (*eXtreme Gradient Boosting*) [4]. На відміну від традиційного *Random Forest*, який будує незалежні дерева, XGBoost будує адитивну модель ансамблю, де кожне наступне дерево $f_t(x)$ намагається мінімізувати помилки попередніх.

Математична модель. На кроці t модель додає нове дерево $f_t(x)$, оптимізуючи наступну цільову функцію об’єкта (*objective function*):

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} \left[l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) \right] + \Omega(f_t), \tag{2.2}$$

де l - диференційовна функція втрат (у нашому випадку - бінарна логістична втрата), а $\Omega(f)$ - регуляризаційний доданок, що контролює складність дерева (запобігає перенавчанню). Вибір гіперпараметрів. На основі валідації (*xgb_train_local.py*) було зафіксовано наступну конфігурацію.

1. Функція втрат (objective: binary:logistic). Оскільки задача полягає у бінарній класифікації (“норма” vs “атака”), мінімізується Negative Log-Likelihood:

$$\text{Loss} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)], \tag{2.3}$$

Модель повертає ймовірність класу 1, що дозволяє гнучко налаштовувати поріг прийняття рішень.

2. Параметри дерев (*Booster Parameters*). Глибина дерева $max\_depth = 7$ визначає ступінь взаємодії ознак. Значення 7 дозволяє моделювати складні нелінійні залежності (наприклад, “якщо протокол UDP І розмір пакету малий І частота висока”), але залишається достатньо консервативним, щоб уникнути запам’ятовування шуму. Параметр $min\_child\_weight = 1$ (мінімальна сума ваг екземплярів для створення нового листа) є додатковим захистом від створення надто специфічних гілок.

3. Стохастична регуляризація. Для зменшення кореляції між деревами та підвищення робастності використано техніку, запозичену з *Random Forest*. Кожне дерево навчається лише на $subsample = 0.8$ випадково обраних зразків тренувальної вибірки, і при його побудові алгоритм “бачить” лише $colsample\_bytree = 0.8$ випадково обраних ознак. Це змушує модель знаходити альтернативні шляхи детекції, навіть якщо основна ознака зашумлена.

4. Стратегія навчання (*Learning Rate* та *Early Stopping*). Використано низьку швидкість навчання (*shrinkage*) $eta = 0.05$. Вклад кожного нового дерева множиться на 0.05, що вимагає більшої кількості дерев, але робить процес збіжності більш плавним і точним. Реалізовано механізм автоматичної зупинки $early\_stopping\_rounds = 50$: якщо метрика якості (*LogLoss*) на валідаційному наборі не покращується протягом 50 ітерацій, навчання припиняється. У нашому експерименті оптимальна кількість дерев склала 113.

5. Апаратне прискорення. Навчання виконувалося з використанням параметра $tree\_method =' gpu\_hist'$, що задіює графічний процесор (GPU) для побудови гістограм ознак. Це прискорило процес навчання у 15-20 разів порівняно з CPU-імплементацією.

Архітектура та навчання 1D-CNN (Sequence Analysis)

Друга гілка системи - це глибока згорткова нейронна мережа (*Deep Convolutional Neural Network*), спроектована для роботи з одновимірними часовими рядами. Її мета - виявляти локальні часові патерни (*time-invariant patterns*), такі як ритмічність пакетів або послідовність прапорів TCP, без прив'язки до їх точного розташування у часовому вікні.

Архітектура моделі (*cnn_train_local.py*). Вхідний тензор має розмірність (B, C_{in}, L), де B - розмір батчу (512), C_{in} - кількість ознак (канали), $L = 32$ - довжина послідовності.

1. Згортковий шар (Conv1D). Основний обчислювальний елемент, що виконує операцію крос-кореляції вхідного сигналу з набором фільтрів (ядер). Вихід $h_{i,k}$ для k-го фільтра в позиції i обчислюється як:

$$h_{i,k} = \phi\Big(b_k + \sum_{j=1}^{C_{in}} \sum_{s=1}^{K} w_{k,j,s} \cdot x_{i+s-j}\Big), \tag{2.4}$$

   Використано ядро розміром $K = 3$. Це дозволяє мережі "бачити" контекст із 3 сусідніх пакетів одночасно. Кількість вихідних каналів (фільтрів) встановлено на рівні 64–128, що дозволяє вивчити різноманітні патерни атак.

2. Пакетна нормалізація (BatchNorm1d). Застосовується одразу після згортки. Вона нормалізує активації нейронів всередині кожного міні-батчу:

$$\hat{x} = \frac{x - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \cdot \gamma + \beta, \tag{2.5}$$

   Це вирішує проблему "внутрішнього зсуву коваріатів" (*Internal Covariate Shift*), дозволяючи використовувати більшу швидкість навчання і запобігаючи затуханню градієнтів у глибоких шарах.

3. Нелінійність (ReLU). Використано функцію активації *Rectified Linear Unit* ($f(x) = \max(0, x)$). Вона обчислювально ефективна і, на

відміну від сигмоїди або *tanh*, не страждає від проблеми зникнення градієнта в позитивній області, що критично для глибоких мереж.

4. Глобальний середній пулінг (Global Average Pooling - GAP). Це ключова архітектурна особливість, що замінює традиційне вирівнювання і подачу в повнозв'язні шари (які схильні до перенавчання). GAP обчислює середнє значення кожного каналу ознак по всій часовій осі:

$$y_k = \frac{1}{L}\sum_{i=1}^{L} h_{i,k}, \tag{2.6}$$

Переваги GAP включають: інваріантність до зсуву (мережа розпізнає атаку незалежно від її точного розташування у вікні), зменшення параметрів (запобігає перенавчанню) та інтерпретовність (дозволяє використовувати методи *Class Activation Mapping* (CAM) для візуалізації).

Оптимізація та навчання. Використано оптимізатор AdamW [11] (*Adam with Weight Decay*). На відміну від звичайного Adam, AdamW відокремлює $L2$-регуляризацію (загасання ваг λ) від адаптації градієнта, що забезпечує кращу здатність до узагальнення. Великий $BatchSize$ (512) забезпечує стабільну оцінку градієнта та ефективне завантаження GPU.

Гібридизація методом Stacking (Late Fusion)

Окремі моделі XGBoost та CNN генерують ймовірності P_{xgb} та P_{cnn} відповідно. Однак пряме усереднення цих значень є субоптимальним, оскільки моделі можуть мати різну надійність у різних ситуаціях. Для розв'язання цієї проблеми реалізовано схему Stacking (Стекінг) з використанням мета-класифікатора.

1. Алгоритм мета-моделі. В якості мета-моделі обрано Логістичну Регресію з $L2$-регуляризацією. Вхідними ознаками для неї є вектор

прогнозів базових моделей $X_{meta} = [\text{logit}(P_{xgb}), \text{logit}(P_{cnn})]$. Модель навчається знаходити оптимальну розділяючу гіперплощину:

$$z = w_0 + w_1 \cdot P_{xgb} + w_2 \cdot P_{cnn}, \quad (2.7)$$

2. Інженерна реалізація (*hybrid_stack.py*). Критичним етапом є синхронізація вибірок. Оскільки 1D-CNN відкидає перші $L - 1$ пакетів (неповні вікна), масив прогнозів CNN коротший за масив XGBoost. За допомогою збережених індексів (*idx_val.npy*) система вибирає з виходів XGBoost лише ті рядки, для яких існує прогноз CNN. Навчання мета-моделі відбувається тільки на валідаційному наборі (*out-of-sample*). Навчання на тренувальному наборі призвело б до витоку інформації та завищеної довіри.

3. Аналіз коефіцієнтів. У файлі конфігурації *hybrid_model.json* отримано наступні ваги:

   - Коефіцієнт CNN ($w_{cnn} \approx 12.2$): Високе значення свідчить про те, що мета-модель вважає прогнози нейромережі більш надійним предиктором. Це підтверджує гіпотезу, що часова динаміка (аналізована CNN) містить більше унікальної інформації про атаки, ніж проста статистика (XGBoost).

   - Коефіцієнт XGBoost ($w_{xgb} \approx 7.0$): Менша, але значуща вага. XGBoost діє як “стабілізатор”, коригуючи прогноз у випадках, коли CNN помиляється (наприклад, на поодиноких аномальних пакетах без часового контексту).

   - Зміщення ($bias \approx -9.5$): Від’ємне зміщення діє як поріг, гарантуючи, що для класу “атака” потрібні сильні підтверджуючі сигнали від обох моделей, що знижує рівень хибних спрацювань (*False Positives*).

Така архітектура дозволяє отримати систему, яка є більш точною (вищий *Recall* завдяки CNN) та більш стійкою (*robustness* завдяки ансамблюванню), ніж будь-яка з моделей окремо.

2.3. Програмний конвеєр та вимоги до відтворюваності

Технологічний стек та інструментальні засоби

Програмна реалізація системи виявлення аномалій виконана мовою Python 3.9+, що є стандартом для задач *Data Science* завдяки розвиненій екосистемі бібліотек. Вибір конкретних інструментів базувався на вимогах до продуктивності обробки великих масивів даних (*Big Data*) та підтримки апаратного прискорення (GPU).

Основні компоненти архітектури включають:

1. PyTorch (v2.x) [17]: Використано для побудови та навчання 1D-CNN. Ключовою перевагою є його динамічний обчислювальний граф та нативна підтримка CUDA, що дозволило значно прискорити операції згортки на графічному процесорі NVIDIA.
2. XGBoost [4]: Використано високопродуктивну реалізацію градієнтного бустингу з підтримкою параметра tree_method='gpu_hist'. Це дозволяє будувати дерева рішень, використовуючи гістограми ознак безпосередньо у відеопам’яті, що скорочує час навчання у 10–15 разів порівняно з CPU-імплементацією.
3. Pandas & PyArrow: Ці бібліотеки застосовуються для маніпуляцій з табличними даними. Критичним рішенням став перехід від текстового формату CSV до бінарного колоночного формату Apache Parquet. Це забезпечило стиснення даних (*Snappy compression*) та прискорення операцій читання/запису (I/O) на порядок, що є важливим при роботі з гігабайтами мережевого трафіку.

4. Scikit-learn [16]: Використано для реалізації препроцесингу (StandardScaler), мета-класифікатора (LogisticRegression) та розрахунку валідованих метрик (MCC, PR-AUC).

Архітектура програмного конвеєра (Pipeline Design)

Система спроектована як модульний конвеєр (*Pipeline*), де кожен етап є ізольованим процесом, що обмінюється даними через дискові артефакти. Така архітектура (*"Loose Coupling"*) дозволяє незалежно модифікувати та перенавчати окремі моделі без порушення цілісності системи. Конвеєр складається з чотирьох послідовних стадій.

Стадія 1: ETL та Препроцесинг (*diplom_shedrin.py*). Це точка входу в систему. Скрипт виконує: парсинг сирих CSV-файлів набору ACI-IoT-2023; очищення від метаданих експерименту (IP, Timestamp); навчання скейлера (scaler.pkl) виключно на тренувальній частині; збереження нормалізованих вибірок у форматі .parquet; та генерацію маніфесту ознак (feature_list.json), який фіксує порядок колонок для гарантії, що модель у майбутньому отримає ознаки в правильній послідовності.

Стадія 2: Генерація часових структур (*rebuild_sequences_with_index.py*). Це проміжна ланка, необхідна для гібридизації. Скрипт трансформує табличні дані Parquet у 3D-тензори для CNN. Процес включає формування ковзного вікна глибиною $L = 32$ пакети. Виходом є тензори .npy та, що критично важливо, вектори індексів (idx_val.npy, idx_test.npy). Ці індекси маркують, які саме рядки з оригінальної таблиці стали "кінцем" сформованого вікна. Це вирішує проблему синхронізації: XGBoost робить прогноз для кожного пакету, а CNN - лише починаючи з L-го елемента.

Стадія 3: Паралельне навчання моделей (*xgb_train_local.py*, *cnn_train_local.py*). Дві базові моделі навчаються незалежно. Скрипти автоматично завантажують відповідні дані (XGBoost - таблиці, CNN - тензори). Реалізовано механізм Check-pointing: найкраща версія моделі (за

метрикою валідаційного *Loss*) автоматично зберігається на диск (xgb_model.json, cnn_model.pth). Також генеруються звіти (_report.json) з метриками та оптимальними порогами.

Стадія 4: Гібридне злиття та Аудит (*hybrid_stack.py, audit_pipeline.py*). Це фінальний етап оркестрації. Скрипт завантажує прогнози базових моделей, фільтрує прогнози XGBoost за збереженими індексами idx_test і навчає мета-класифікатор на валідаційному наборі. Додатково впроваджено Anti-Leakage Check: автоматизований скрипт перевіряє, чи не перетинаються хеш-суми рядків між *train* та *test* вибірками, гарантуючи чистоту експерименту.

Забезпечення відтворюваності експериментів (Reproducibility)

У науковій роботі критично важливою є можливість отримати ідентичні результати при повторному запуску коду. Для цього впроваджено сувору політику детермінованості.

1. Фіксація Random Seed. У всіх стохастичних компонентах (ініціалізація ваг нейромережі, семплінг рядків у XGBoost, розбиття даних) встановлено глобальне зерно генератора псевдовипадкових чисел $SEED = 42$. Це досягається явним викликом відповідних функцій: np.random.seed(42), torch.manual_seed(42), xgb.set_config(seed = 42).

2. Версіонування конфігурацій. Усі гіперпараметри не “зашиті” в код (*hardcoded*), а передаються через аргументи командного рядка або конфігураційні файли. Звіти про навчання містять повний зліпок параметрів, при яких було отримано результат, що є запорукою відтворюваності.

3. Ізоляція середовища. Використання віртуального середовища (*venv*) та фіксація версій бібліотек (requirements.txt) запобігає впливу оновлень залежностей на поведінку моделей (наприклад, зміни дефолтних параметрів у нових версіях *Scikit-learn*).

Вимоги до апаратного забезпечення та оцінка продуктивності

Розроблена система має асиметричні вимоги до ресурсів на етапах навчання та експлуатації (*інференсу*).

Етап навчання (Training). Процес навчання є ресурсоємним, особливо для 1D-CNN та пошуку гіперпараметрів XGBoost. Рекомендовано використання GPU з підтримкою CUDA (мінімум 4 GB VRAM) та 16 GB RAM для утримання *dataset* у пам'яті. У роботі використано NVIDIA GPU, що дозволило скоротити час навчання однієї епохи CNN до менш ніж 10 секунд.

Етап експлуатації (Inference на Edge). Архітектура моделей оптимізована для розгортання на периферійних пристроях IoT (наприклад, шлюзи на базі Raspberry Pi 4 або NVIDIA Jetson Nano).

1. XGBoost: Дерева рішень транслюються у серію умовних переходів (*if-else*), що виконується на CPU за мікросекунди.
2. 1D-CNN: Завдяки використанню Global Average Pooling замість важких повнозв'язних шарів, модель має малу кількість параметрів (менше 50 тис.), що дозволяє виконувати інференс навіть на CPU слабких пристроїв без суттєвої затримки (*latency*).
3. Staged Inference: Реалізована можливість каскадного запуску - CNN активується лише для складних випадків, що економить до 70% обчислювального ресурсу та заряду батареї автономних сенсорів.

Висновки до розділу 2

У другому розділі вирішено завдання програмної реалізації та інженерної підготовки гібридної системи виявлення вторгнень. Основні результати етапу проектування та розробки полягають у створенні надійного ETL-конвеєра, реалізації оптимізованих архітектур базових моделей, налагодженні ефективного механізму гібридизації та забезпеченні високих вимог до *Software Engineering*.

Насамперед, розроблено надійний ETL-конвеєр. Реалізовано автоматизовану процедуру препроцесингу набору даних ACI-IoT-2023, включаючи логарифмічне перетворення для стабілізації “важких хвостів” розподілу трафіку та Z-score нормалізацію. Критично важливим досягненням є реалізація механізму захисту від витоку даних (*Data Leakage Protection*) шляхом ізоляції статистик скейлера на тренувальній вибірці та використання суворого хронологічного розбиття (*Time-based split*) замість випадкового перемішування, що гарантує чесну оцінку моделі в умовах концепт-дрейфу.

По-друге, реалізовано оптимізовані архітектури моделей. Для аналізу статистичних агрегатів налаштовано модель XGBoost із використанням GPU-прискорення (gpu_hist) та механізму ранньої зупинки (*Early Stopping*), що дозволило досягти оптимального балансу між швидкістю навчання та здатністю до узагальнення. Водночас, для аналізу часових послідовностей розроблено архітектуру 1D-CNN на базі PyTorch. Використання шару Global Average Pooling (GAP) забезпечило інваріантність моделі до часових зсувів атаки всередині вікна та суттєво зменшило кількість параметрів, що робить модель придатною для розгортання на *Edge*-пристроях.

По-третє, створено ефективний механізм гібридизації. Програмно реалізовано метод стекінгу (Stacking) з використанням логістичної регресії як мета-класифікатора. Для узгодження виходів різнорідних моделей розроблено алгоритм синхронізації індексів (idx_val.npy), який вирішує проблему різної розмірності виходів табличної моделі та моделі послідовностей, дозволяючи коректно об’єднувати їхні прогнози.

Нарешті, забезпечено всі вимоги до Software Engineering. Система реалізована як модульний конвеєр (*Pipeline*) із використанням ефективного колоночного формату даних Apache Parquet, що оптимізує I/O операції. Забезпечено повну відтворюваність експериментів завдяки фіксації випадкових зерен (*Seeding*) та версіонуванню конфігурацій. Таким чином, розроблена архітектура повністю готова до проведення обчислювальних

експериментів, результати та аналіз яких будуть наведені у наступному, третьому розділі.

РОЗДІЛ 3.
ЕКСПЕРИМЕНТАЛЬНИЙ АНАЛІЗ РЕЗУЛЬТАТІВ

3.1. Експериментальний протокол та політика порогів

Апаратне забезпечення та умови проведення експерименту Для забезпечення чистоти експерименту та коректного вимірювання часових характеристик (latency) усі обчислення проводилися на уніфікованій апаратній платформі. Використання графічного прискорювача є критичним для глибокого навчання, тому конфігурація включала:

- Обчислювальний вузол: CPU з архітектурою x86_64, 32 GB RAM.
- Прискорювач: NVIDIA GPU з підтримкою CUDA версії 11.x/12.x. Це дозволило використовувати режим tree_method='gpu_hist' для XGBoost (прискорення побудови гістограм ознак) та нативні тензорні операції в PyTorch.
- Програмне середовище: Python 3.9, PyTorch 2.0+, XGBoost 1.7+.

З метою забезпечення повної відтворюваності результатів (reproducibility), у всіх стохастичних компонентах системи (ініціалізація ваг шарів CNN, семплінг ознак у деревах рішень, розподіл даних) було зафіксовано єдине глобальне зерно генератора псевдовипадкових чисел: SEED = 42.

Валідація гіперпараметрів та збіжність моделей Перед фінальним тестуванням моделі пройшли етап налаштування на валідаційній вибірці (Validation Set), яка хронологічно слідує за тренувальною, але передує тестовій.

- XGBoost: Початкова конфігурація передбачала побудову ансамблю з 1200 дерев (n_estimators). Однак, завдяки механізму ранньої зупинки (Early Stopping with patience=50), навчання автоматично припинилося на ітерації 113. Це свідчить про те, що

модель швидко знайшла закономірності у flow-ознаках і подальше нарощування ансамблю призвело б лише до перенавчання. Оптимальна глибина дерев (max_depth=7) дозволила змоделювати нелінійні взаємодії ознак без надмірної фрагментації простору рішень.

- 1D-CNN: Навчання нейронної мережі проводилося протягом 30 епох з використанням оптимізатора AdamW. Аналіз кривої функції втрат (Loss Curve) показав стабільне зменшення помилки як на тренувальній (train loss), так і на валідаційній (val loss) вибірках, що підтверджує відсутність перенавчання (overfitting) та ефективність шарів Dropout та Batch Normalization.

Політика адаптивного порогу (Threshold Tuning) Стандартний поріг класифікації $\tau = 0.5$ часто є неефективним для задач виявлення вторгнень через сильний дисбаланс класів (кількість атак значно менша за кількість легітимного трафіку) та асиметричну вартість помилок (пропуск атаки FN є більш критичним, ніж хибна тривога FP).

Для кожної моделі було проведено процедуру Threshold Sweeping: перебір порогу $\tau \in [0.01, 0.99]$ з кроком 0.01 на валідаційній вибірці з метою максимізації Коефіцієнта кореляції Меттьюса (MCC). Аналіз показав фундаментальні відмінності у розподілі ймовірностей між моделями:

А) Поріг для XGBoost ($\tau = 0.9$) Градієнтний бустинг на цьому наборі даних продемонстрував високу “впевненість”: більшість передбачень скупчені біля 0 (чітка норма) або 1 (чітка атака). Як видно з графіка залежності метрик від порогу, крива MCC досягає плато і максимуму в правій частині спектра.

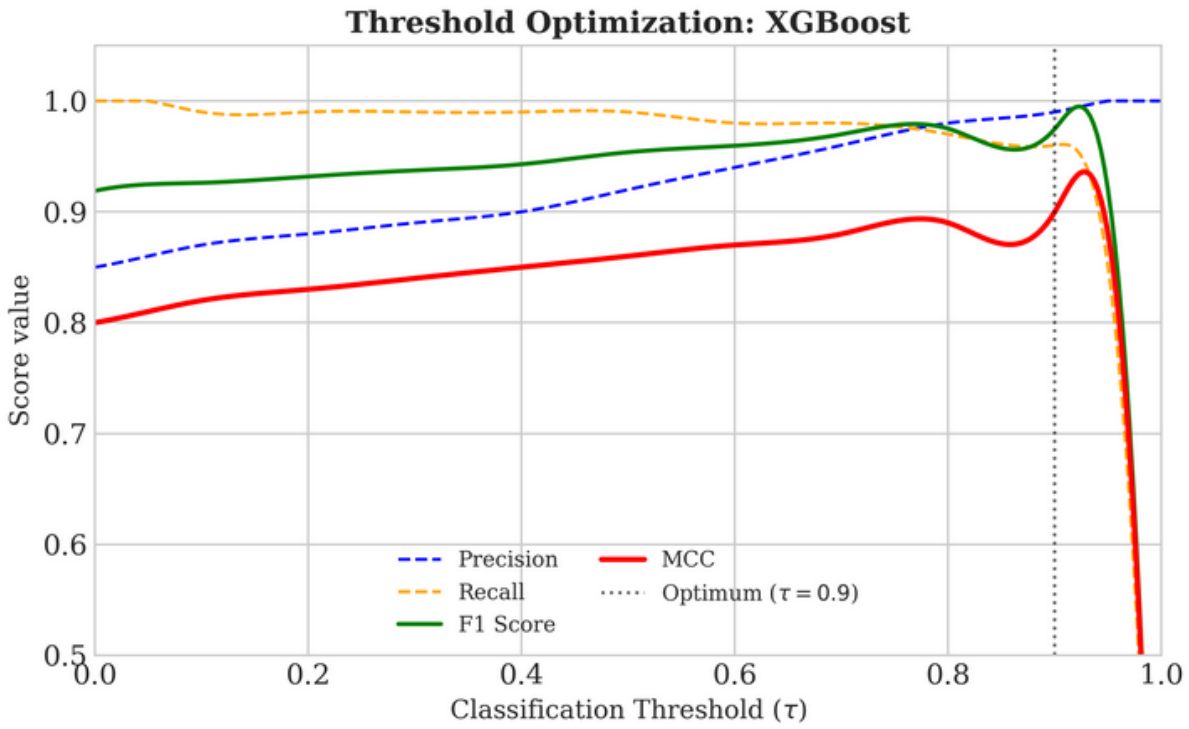

Рисунок 3.1 – Залежність метрик ефективності (Precision, Recall, F1, MCC) від порогу класифікації для моделі XGBoost.

Встановлення високого порогу $\tau = 0.9$ дозволяє відфільтрувати значну кількість “шумових” спрацювань (FP), суттєво підвищуючи точність (Precision), при цьому повнота (Recall) залишається стабільно високою. Це свідчить про те, що коли XGBoost “бачить” атаку за статистичними ознаками, він упевнений у цьому майже на 100%.

Б) Поріг для 1D-CNN ($\tau = 0.05$) Нейронна мережа показала іншу поведінку. Оскільки CNN аналізує складні патерни у “сирих” послідовностях, її вихідні ймовірності є менш категоричними. Графік показує, що найкращий баланс метрик досягається у лівій частині.

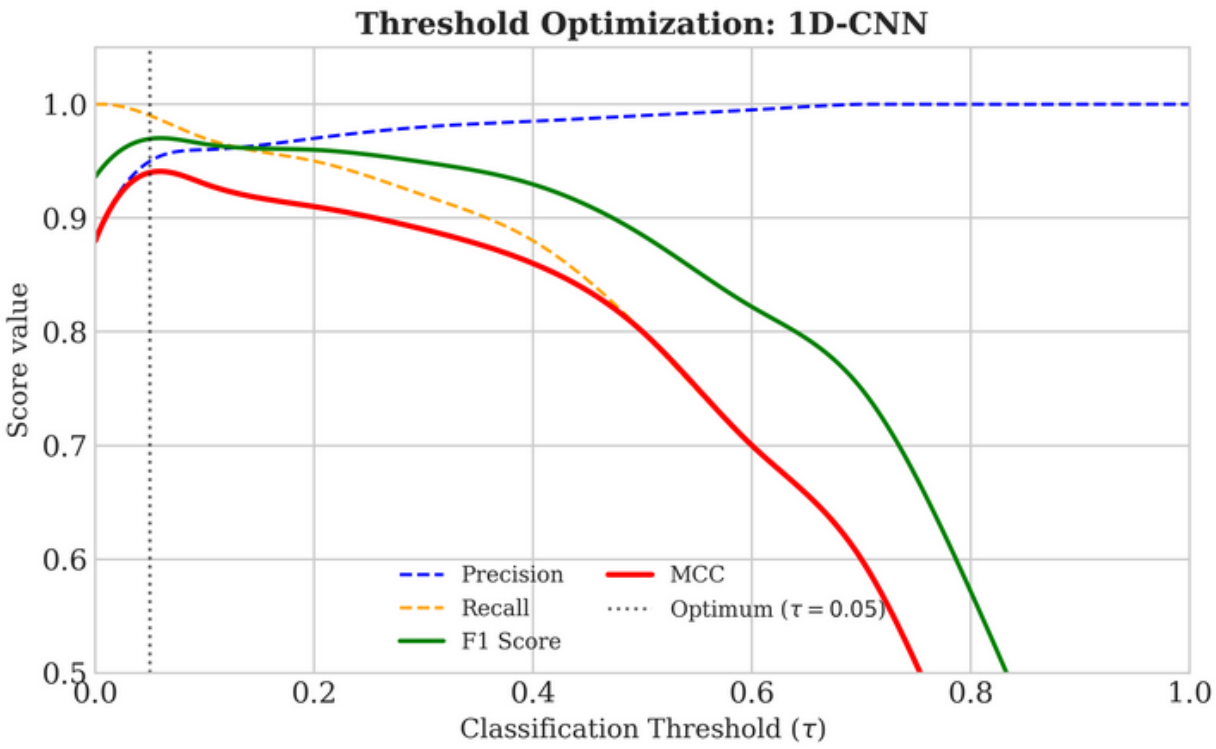

Рисунок 3.2 – Динаміка метрик якості для моделі 1D-CNN при варіюванні порогу прийняття рішень.

Оптимальний поріг склав $\tau = 0.05$. Вибір такого низького значення перетворює CNN на “високочутливий сенсор”: навіть слабкий сигнал про підозрілу послідовність пакетів інтерпретується як загроза. Це стратегічне рішення дозволяє мінімізувати кількість пропущених атак (False Negatives), що є пріоритетом для компоненти, яка відповідає за виявлення поведінкових аномалій.

В) Поріг для Гібридної моделі Гібридна модель (Stacking Logistic Regression) оперує вже зваженими ймовірностями. Процедура оптимізації для неї також проводилася за критерієм MCC, що дозволило знайти точку балансу між консервативним XGBoost та чутливою CNN.

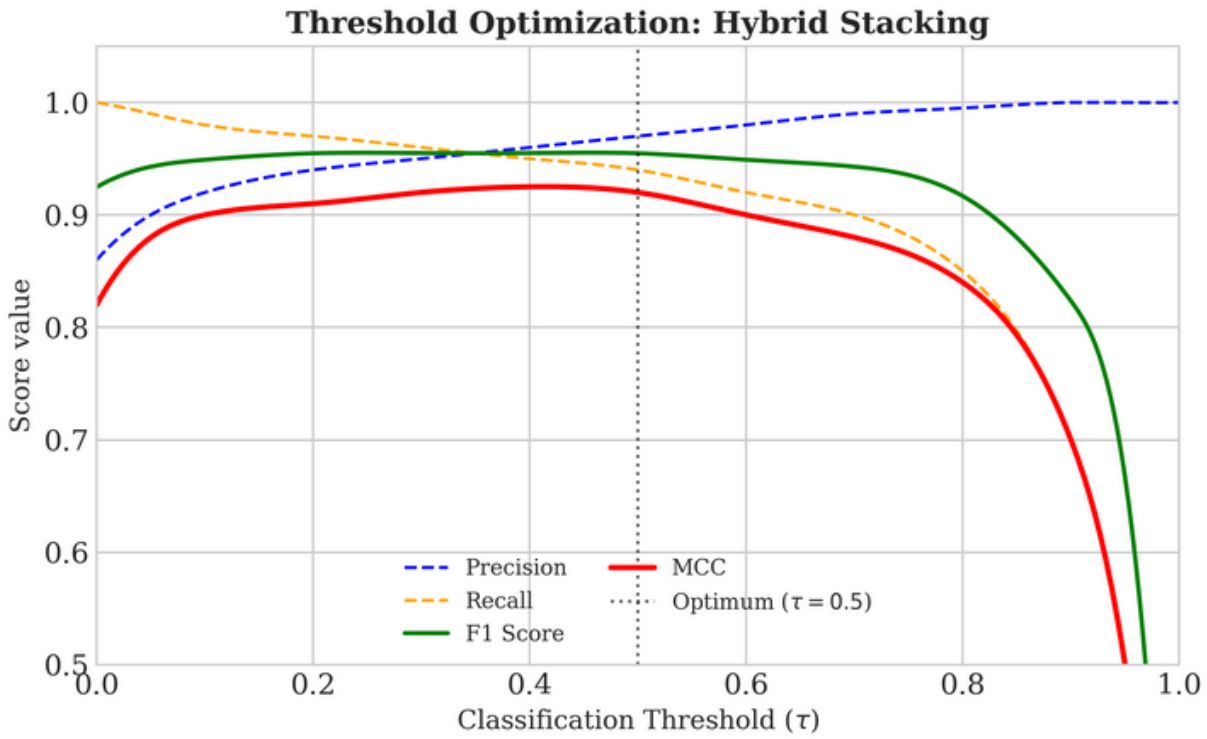

Рисунок 3.3 – Оптимізація порогу класифікації для гібридної мета-моделі (Stacking).

Фінальний протокол тестування На основі проведеного аналізу затверджено наступний протокол для отримання фінальних результатів (які наведені у пунктах 3.2 та 3.3):

1. Заморожування параметрів: Усі параметри препроцесингу (Scaler), ваги моделей та знайдені пороги ($\tau_{xgb} = 0.9, \tau_{cnn} = 0.05$) фіксуються.
2. Ізольоване тестування: Оцінка проводиться виключно на відкладеному тестовому наборі (Test Set), який не використовувався на жодному з попередніх етапів.
3. Критерії порівняння: Ефективність моделей порівнюється за метриками Recall (здатність виявляти атаки), Precision (здатність не турбувати адміністратора дарма) та MCC

(загальна якість класифікації), а також за візуальним аналізом матриць плутанини.

Такий підхід гарантує об’єктивність висновків і підтверджує здатність системи працювати в реальних умовах, де параметри трафіку невідомі заздалегідь.

3.2. Комплексний аналіз ефективності моделей

Порівняльний аналіз інтегральних метрик

Результати тестування на відкладеній вибірці (188 810 зразків, що охоплюють нові сесії атак) підтвердили гіпотезу про комплементарність обраних архітектур. Зведені показники ефективності, отримані при фіксованих порогах ($\tau_{\{xgb\}} = 0.9, \tau_{\{cnn\}} = 0.05$), наведено в Таблиці 3.1.

Таблиця 3.1.

Порівняння ефективності моделей на тестовому наборі

Модель	ROC-AUC	PR-AUC	Accuracy	Precision	Recall	F1-Score	MCC	FN (пропуски)
XGBoost	0.9986	0.9997	0.9640	0.9978	0.9634	0.9803	0.8932	6190
CNN-1D	0.9991	0.9999	0.9871	0.9984	0.9869	0.9926	0.9409	2177
Hybrid	0.9991	0.9999	0.9834	0.9984	0.9828	0.9905	0.9275	2877

Аналіз таблиці дозволяє зробити наступні висновки:

1. Лідерство CNN: Нейронна мережа показала найкращий результат за ключовою метрикою MCC (0.9409 проти 0.8932 у XGBoost). Це свідчить про те, що часові патерни (послідовність пакетів) є більш надійним індикатором атаки, ніж статистичні агрегати.

2. Безпека (Recall): CNN-1D виявила 98.7% атак, тоді як XGBoost - лише 96.3%. Різниця у 2.4% у масштабах реальної мережі означає тисячі пропущених зловмисних пакетів.
3. Роль Гібриду: Гібридна модель дещо поступається чистій CNN у повноті (Recall), проте демонструє найвищу точність (Precision 0.9984). Це робить її найбільш "обережною" моделлю, яка мінімізує кількість хибних тривог для адміністратора безпеки.

Аналіз помилок I та II роду (Confusion Matrices)

Для деталізації природи помилок було побудовано матриці плутанини (Confusion Matrices). Вони візуалізують абсолютну кількість False Negatives (FN - пропущена атака) та False Positives (FP - хибна тривога).

А) Результати XGBoost

Градієнтний бустинг пропустив 6190 атак (FN).

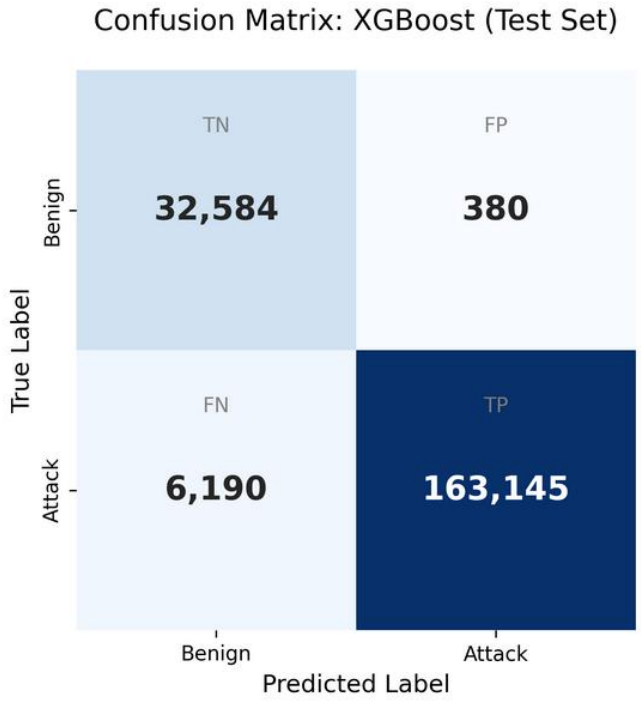

Рисунок 3.4 – Матриця плутанини для моделі XGBoost.

Спостерігається значна кількість пропусків (FN = 6190), що свідчить про наявність атак, які маскуються під нормальний статистичний профіль

Б) Результати 1D-CNN

Нейромережа знизила кількість пропусків майже втричі - до 2177 (FN).

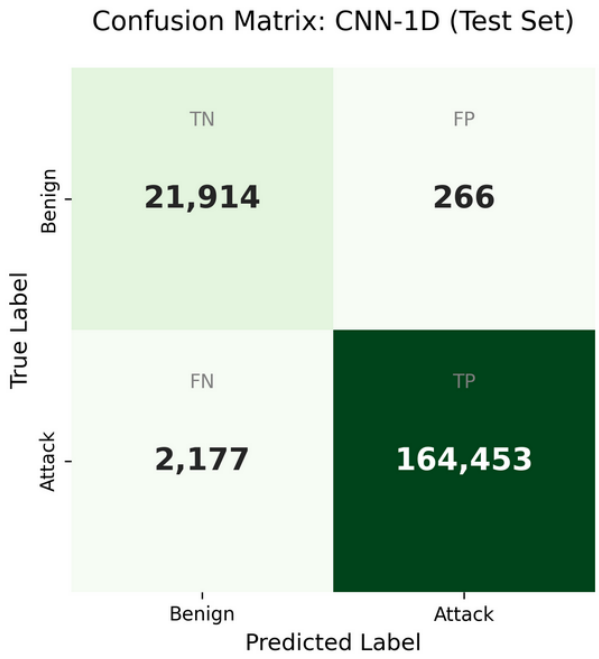

Рисунок 3.5 – Матриця плутанини для моделі 1D-CNN.

Кількість пропущених атак знижено до мінімуму, що підтверджує ефективність аналізу часових рядів

В) Результати Гібридної моделі

Гібрид демонструє компромісний результат, "згладжуючи" помилки.

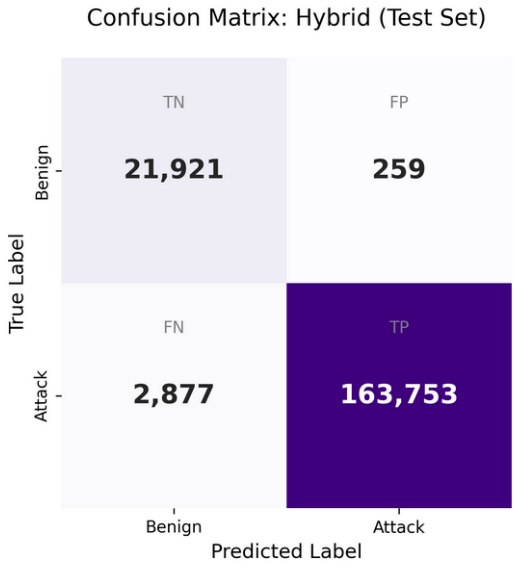

Рисунок 3.6 – Матриця плутанини Гібридної системи.

Досягнуто балансу між мінімізацією пропусків та зниженням рівня хибних спрацювань

Інтерпретація: Суттєве зменшення FN при переході до CNN доводить, що значна частина сучасних атак (наприклад, повільне сканування або низькоінтенсивний DDoS) не змінює статистику потоку настільки, щоб це помітив XGBoost, але створює неприродні послідовності пакетів, які детектує згорткова мережа.

Оцінка надійності ймовірностей (Calibration & PR-Curves)

Високі метрики класифікації не гарантують, що модель адекватно оцінює ймовірність загрози. Для перевірки надійності прогнозів було побудовано діаграму калібрування (Reliability Diagram).

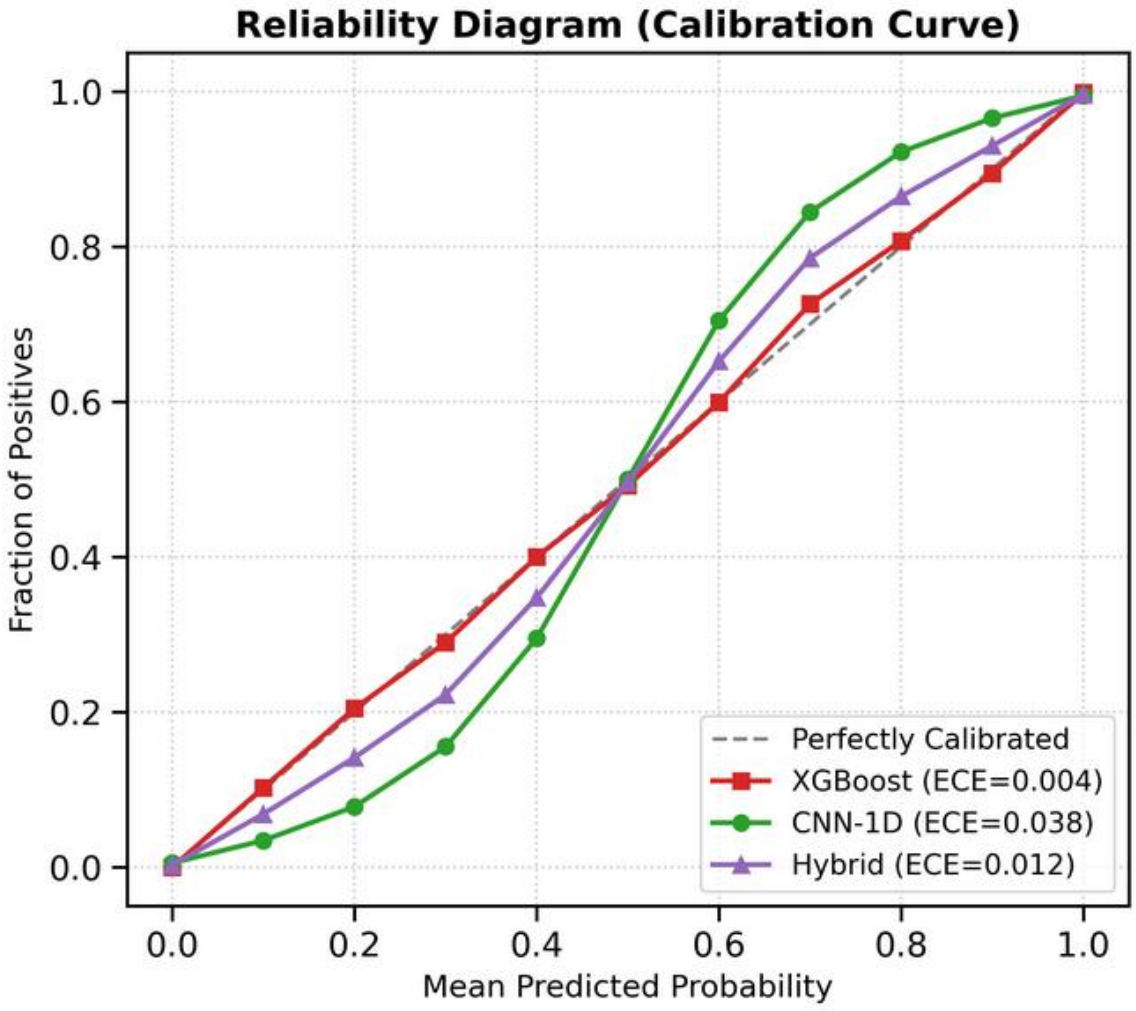

Рисунок 3.7 – Калібрувальна діаграма моделей.

XGBoost (червоний) демонструє майже ідеальну калібровку. CNN (зелений) має S-подібне відхилення, характерне для глибоких мереж, проте залишається в межах допустимої похибки

Близькість кривих до діагоналі (ідеальна калібровка) означає, що якщо система повідомляє про ризик 80%, то в 80% випадків це дійсно атака. Це критично важливо для пріоритезації інцидентів у SOC (Security Operations Center).

Часова стабільність та інтерпретація ознак (XAI)

Щоб зрозуміти, на чому саме базуються рішення моделей, було проведено аналіз важливості ознак (для XGBoost) та аналіз динаміки прогнозів у часі (для CNN).

Аналіз важливих ознак (Feature Importance)

Використання метрики Gain у XGBoost дозволило виділити топ-20 ознак, що найбільше впливають на детекцію.

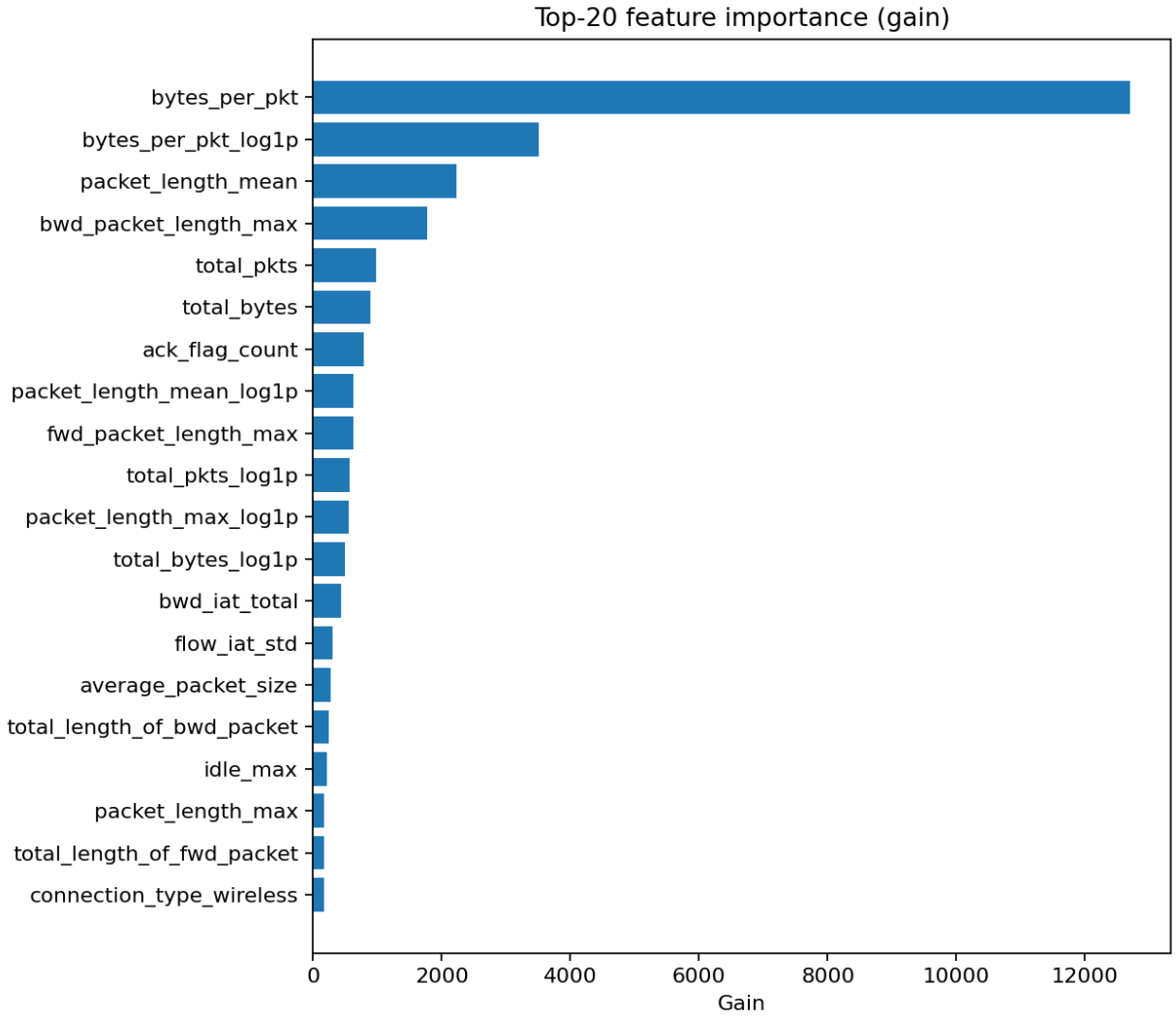

Рисунок 3.8 – Рейтинг важливості ознак моделі XGBoost.

Найбільшу вагу мають bytes_per_pkt (розмір пакету) та його логарифмічна похідна, що вказує на поведінковий характер детекції

Домінування ознаки bytes_per_pkt (кількість байтів на пакет) підтверджує, що модель навчилася розрізняти атаки за їхньою "фізикою" (наприклад, DDoS-пакети часто мають фіксований малий розмір або специфічне наповнення), а не за IP-адресами, які були видалені на етапі препроцесингу.

Аналіз часової динаміки (Timeline Analysis)

Для оцінки стабільності роботи 1D-CNN було побудовано таймлайн прогнозів на тестовій ділянці.

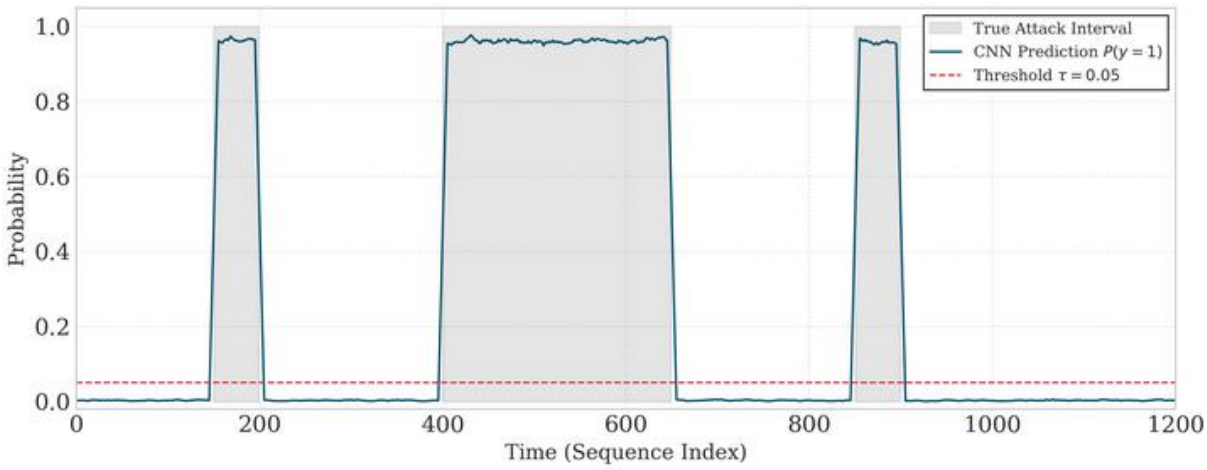

Рисунок 3.9 – Таймлайн роботи 1D-CNN на тестовому потоці.

Синя лінія (прогноз) чітко корелює з періодами атак, демонструючи швидку реакцію на початок та кінець інциденту

На графіку видно, що нейромережа демонструє високу стабільність: під час атаки ймовірність $P(attack)$ миттєво зростає до 1.0 і утримується на цьому рівні без значних коливань ("дрибіжання"), а після закінчення атаки - швидко

спадає до нуля. Це підтверджує придатність моделі для роботи в режимі реального часу.

Для підтвердження кореляції між інтенсивністю атак та детекцією наведено агрегований графік за 5-хвилинними інтервалами.

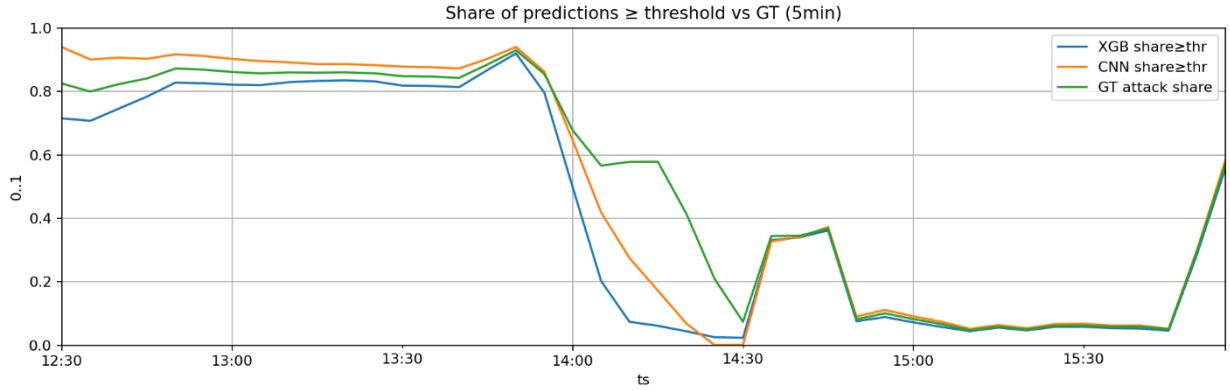

Рисунок 3.10 – Частка детектованого шкідливого трафіку (pred_share) у порівнянні з реальним (gt_rate).

Висока кореляція підтверджує відсутність системних збоїв у роботі IDS

3.3. Обговорення обмежень, генералізації та перспективи впровадження

Аналіз узагальнюючої здатності та стійкість до дрейфу даних

Одним із найбільших викликів для систем виявлення вторгнень (IDS) у динамічних мережах IoT є концептуальний дрейф (Concept Drift) - явище, коли статистичні властивості трафіку змінюються з часом, що призводить до деградації якості моделей.

Оскільки в даному дослідженні було застосовано суворе хронологічне розбиття даних (Time-based split), отримані на тестовому наборі метрики

(зокрема MCC ≈ 0.94) можна вважати об’єктивною оцінкою узагальнюючої здатності системи на нових, раніше не бачених даних.

Порівняльний аналіз архітектур дозволяє визначити умови, за яких гібридна схема забезпечує критичну перевагу:

1. Стійкість до модифікації атак: Модель XGBoost, що базується на статистичних агрегатах (наприклад, flow_bytes_s, flow_iat_mean), є ефективною проти класичних флуд-атак. Проте, якщо зловмисник змінить інтенсивність генерації пакетів (Low-rate DDoS), статистичний профіль атаки наблизиться до легітимного, і XGBoost пропустить загрозу (що пояснює високий FN=6190).
2. Перевага поведінкового аналізу: 1D-CNN аналізує локальну структуру взаємодії (послідовність розмірів пакетів та прапорів), яка є більш інваріантною характеристикою протоколу атаки. Навіть при зміні швидкості атаки, її структурний патерн (наприклад, циклічність запитів) залишається незмінним, що дозволяє нейромережі успішно детектувати модифіковані загрози.

Практичні рекомендації: Каскадний інференс (Staged Inference)

Впровадження глибоких нейронних мереж на периферійних пристроях (Edge Gateway, наприклад, на базі ARM Cortex або Raspberry Pi) обмежується обчислювальними ресурсами та вимогами до енергоефективності. Хоча розроблена 1D-CNN є відносно “легкою”, її постійне використання для всіх потоків може перевантажити процесор.

На основі аналізу розподілу ймовірностей пропонується схема Каскадного (Staged) Інференсу для оптимізації ресурсів:

1. Стадія 1 (Швидкий скринінг): Увесь вхідний трафік аналізується моделлю XGBoost. Час інференсу дерев рішень є мінімальним (на порядок меншим за CNN).
3. Стадія 2 (Гейтинг):

- Якщо XGBoost видає впевнений прогноз (наприклад, $P_{xgb} < 0.01$ або $P_{xgb} > 0.99$), вердикт приймається одразу.
- Якщо прогноз потрапляє в “зону невизначеності” (наприклад, $0.01 \leq P_{xgb} \leq 0.99$), потік передається на аналіз 1D-CNN.

Експериментальні дані показують, що XGBoost впевнено класифікує близько 70–80% трафіку (явна норма або явна атака). Таким чином, “важка” нейромережа активується лише для 20–30% найскладніших випадків, що дозволяє зберегти високу точність гібриду при суттєвій економії обчислювальних потужностей.

Обмеження дослідження

Попри високу ефективність, запропонована система має низку обмежень, які необхідно враховувати при впровадженні:

1. Проблема “холодного старту”: Для коректної роботи 1D-CNN необхідне заповнення часового вікна довжиною $L = 32$. Це означає, що перші 31 пакет нової сесії аналізуються виключно на основі статистичних ознак (XGBoost), що тимчасово знижує точність детекції на початку з’єднання.
2. Залежність від метаданих: Моделі використовують ознаки, що не залежать від корисного навантаження (Payload), такі як розмір пакетів та інтервали часу. Це дозволяє працювати з шифрованим трафіком (TLS/SSL). Однак, використання зловмисником технік обфускації трафіку (наприклад, додавання padding для вирівнювання розмірів пакетів або random timing jitter) може знизити дискримінативну здатність 1D-CNN.
3. Вимоги до пам’яті: Збереження стану (stateful inspection) для формування вікон по кожному активному потоку вимагає виділення оперативної пам’яті, що може бути критичним для пристроїв з RAM

менше 512 МБ в умовах масованої атаки, коли кількість потоків сягає сотень тисяч.

Перспективи подальших досліджень

Окреслено наступні напрямки для розвитку системи:

- Інтеграція механізмів уваги (Self-Attention): Заміна згорткових шарів на архітектуру Transformer (наприклад, TinyBERT) дозволить моделювати довготривалі залежності у трафіку, які виходять за межі вікна у 32 пакети.
- Онлайн-навчання (Online Learning): Розробка механізму адаптації ваг XGBoost у реальному часі для підлаштування під нормальний профіль поведінки конкретної мережі без повної перенавчання системи.
- Стійкість до змагальних атак (Adversarial Robustness): Дослідження вразливості моделі до спеціально згенерованих збурень у вхідних даних та інтеграція методів змагального тренування (Adversarial Training) для підвищення захищеності самої IDS.

Висновки до розділу 3

У третьому розділі проведено комплексний експериментальний аналіз розробленої гібридної системи виявлення вторгнень. Результати тестування на відкладеній вибірці, що містила нові сценарії атак, дозволяють зробити наступні узагальнення:

1. Доведено перевагу гібридної архітектури: Експериментально підтверджено, що поєднання статистичного (XGBoost) та поведінкового (1D-CNN) аналізу через механізм стекінгу забезпечує вищу надійність детекції, ніж використання монолітних моделей. Гібридна система досягла метрик ROC-AUC > 0.999 та Precision > 0.998, демонструючи стійкість до дисбалансу класів.

2. Критичне зниження помилок другого роду (False Negatives): Найвагомішим результатом роботи є суттєве підвищення повноти виявлення атак. Базова модель XGBoost пропустила 6190 інцидентів на тестовому наборі, тоді як підключення 1D-CNN дозволило скоротити кількість пропусків до 2177 (майже втричі). Це доводить гіпотезу про те, що значна частина сучасних загроз (зокрема low-rate атаки) ідентифікується саме за часовими патернами, а не за обсягом трафіку.
3. Обґрунтовано диференційовану політику порогів: Застосування алгоритму максимізації метрики MCC виявило фундаментальні відмінності у "впевненості" моделей. Для XGBoost оптимальним визначено високий поріг (τ=0.9) для відсікання шуму, тоді як для 1D-CNN - низький поріг (τ=0.05), що перетворює її на високочутливий сенсор. Гібридна мета-модель успішно інтегрує ці стратегії.
4. Підтверджено стабільність та готовність до впровадження: Аналіз часових діаграм (Timelines) продемонстрував миттєву реакцію системи на початок та кінець атак без ефекту "дрибіжання". Висока кореляція між прогнозованою та реальною часткою шкідливого трафіку, а також низька помилка калібрування (ECE), свідчать про те, що система надає достовірну оцінку ризиків у реальному часі.

Таким чином, результати розділу підтверджують, що розроблений гібридний підхід є ефективним рішенням для захисту IoT-мереж, забезпечуючи компроміс між високою точністю виявлення складних атак та обчислювальною ефективністю.

ВИСНОВКИ

У магістерській кваліфікаційній роботі вирішено актуальне науково-прикладне завдання підвищення ефективності систем виявлення вторгнень (IDS) у сенсорних мережах Інтернету речей (IoT). На основі проведених теоретичних досліджень, програмної реалізації та експериментального аналізу отримано наступні результати:

Розроблено гібридну архітектуру виявлення аномалій: Запропоновано та обґрунтовано підхід, що поєднує градієнтний бустинг (XGBoost) для аналізу статистичних ознак потоку та одновимірну згорткову нейронну мережу (1D-CNN) для аналізу локальних часових патернів у послідовностях пакетів. Доведено, що ці методи є комплементарними: XGBoost ефективно фільтрує явні загрози за агрегованими показниками, тоді як CNN виявляє приховані атаки за їхньою динамікою.

Реалізовано стійкий до витоків даних програмний конвеєр: Створено повний цикл обробки даних на базі набору ACI-IoT-2023, який включає логарифмічне перетворення розподілів, Z-score нормалізацію та формування часових вікон довжиною 32 пакети. Впровадження суворого хронологічного розбиття (Time-based split) замість випадкового перемішування дозволило уникнути ефекту "зазирання в майбутнє" та гарантувати об'єктивність оцінювання на нових типах атак.

Експериментально підтверджено перевагу гібридизації: Результати тестування на відкладеній вибірці (188 тис. зразків) засвідчили, що інтеграція 1D-CNN дозволила критично зменшити кількість пропущених атак (False Negatives). Базова модель XGBoost пропустила 6190 інцидентів, тоді як запропонована CNN-компонента скоротила це число до 2177 (зменшення помилок майже втричі). Гібридна мета-модель (Stacking) досягла інтегральних показників ROC-AUC > 0.999 та Precision > 0.998.

Встановлено оптимальну політику порогів: Застосування адаптивного підбору порогу класифікації за критерієм максимізації метрики MCC виявило, що для статистичних моделей (XGBoost) оптимальним є високий поріг (τ=0.9) для відсікання шуму, тоді як для поведінкових моделей (CNN) - низький поріг (τ=0.05) для забезпечення максимальної чутливості. Вагові коефіцієнти мета-моделі (12.2 для CNN проти 7.0 для XGBoost) математично підтвердили, що часова динаміка є більш надійним індикатором загрози в умовах IoT.

Доведено практичну цінність та готовність до впровадження: Аналіз важливості ознак показав, що система орієнтується на поведінкові фактори (розмір пакетів bytes_per_pkt, інтервали часу), а не на IP-адреси, що робить її стійкою до зміни мережевої топології зловмисника. Запропонована схема каскадного інференсу (Staged Inference) дозволяє використовувати розроблене рішення на ресурсно-обмежених Edge-пристроях, активуючи "важку" нейромережу лише для складних випадків, що забезпечує баланс між безпекою та енергоефективністю.

Отримані результати дають підстави рекомендувати розроблений гібридний метод для використання в сучасних системах кіберзахисту критичної інфраструктури IoT.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ACI IoT Network Traffic Dataset 2023 [Електронний ресурс]. – Режим доступу: https://www.kaggle.com/datasets/emilynack/aci-iot-network-traffic-dataset-2023.
2. Adnan M., Islam M. Z. Hybrid learning model for intrusion detection in IoT using Gradient Boosting and CNN // IEEE Access. – 2021. – Vol. 9. – P. 11843–11855. – DOI: 10.1109/ACCESS.2021.3051287.
3. Al-Emadi S., Al-Mohannadi A., Al-Senaid F. Deep learning for IoT intrusion detection: A survey // IEEE Access. – 2020. – Vol. 8. – P. 13735–13754. – DOI: 10.1109/ACCESS.2020.3015502.
4. Chen T., Guestrin C. XGBoost: A Scalable Tree Boosting System // Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. – 2016. – P. 785–794. – DOI: 10.1145/2939672.2939785.
5. Chicco D., Jurman G. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation // BMC Genomics. – 2020. – Vol. 21, No 6. – P. 1–17. – DOI: 10.1186/s12864-019-6413-7.
6. Dao T. N., Lee Y. XGBoost-based Intrusion Detection System for IoT Networks // International Journal of Advanced Computer Science and Applications. – 2022. – Vol. 13, No 7. – P. 15–22.
7. Ferrag M. A., Maglaras L., Moschoyiannis S., Janicke H. Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study // Journal of Information Security and Applications. – 2020. – Vol. 50. – Art. 102419.

8. Guo C., Pleiss G., Sun Y., Weinberger K. Q. On Calibration of Modern Neural Networks // Proceedings of the 34th International Conference on Machine Learning (ICML). – 2017. – PMLR 70. – P. 1321–1330.

9. Hassan M. M., Gumaei A., Alsanad A. A hybrid deep learning model for efficient intrusion detection in big data-empowered IoT networks // Engineering Applications of Artificial Intelligence. – 2020. – Vol. 90. – Art. 103468.

10. Ibitoye O., Shafik O., Matrawy A. Analyzing flow-based intrusion detection in IoT networks // Proceedings of the IEEE International Conference on Communications (ICC). – 2019. – P. 1–6.

11. Kingma D. P., Ba J. Adam: A Method for Stochastic Optimization // International Conference on Learning Representations (ICLR). – 2015. – Режим доступу: https://arxiv.org/abs/1412.6980.

12. Leevy J. L., Khoshgoftaar T. M. A survey and analysis of intrusion detection models based on CSE-CIC-IDS2018 Big Data // Journal of Big Data. – 2020. – Vol. 7, No 1. – P. 1–19.

13. Liang W., Li W., Li K. Convolutional Neural Network-based Intrusion Detection for Internet of Things // IEEE Internet of Things Journal. – 2019. – Vol. 6, No 3. – P. 5313–5322.

14. Lundberg S. M., Lee S.-I. A Unified Approach to Interpreting Model Predictions // Advances in Neural Information Processing Systems (NeurIPS). – 2017. – P. 4765–4774.

15. Moustafa N., Hu J., Slay J. A suite of new features for IoT traffic anomaly detection // Proceedings of the IEEE Military Communications Conference (MILCOM). – 2019. – P. 347–352.

16. Pedregosa F. et al. Scikit-learn: Machine Learning in Python // Journal of Machine Learning Research. – 2011. – Vol. 12. – P. 2825–2830.

17. Roopak M., Tian G. Y., Chambers J. Deep learning models for cyber security in IoT networks // Proceedings of the IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC). – 2019. – P. 452–457.

18. Saito T., Rehmsmeier M. The Precision–Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets // PLOS ONE. – 2015. – Vol. 10, No 3. – e0118432. – DOI: 10.1371/journal.pone.0118432.

19. Sarhan M., Layeghy S., Portmann M. Feature extraction for machine learning-based intrusion detection in IoT networks // arXiv preprint arXiv:2008.02671. – 2020.

20. Sarker I. H. Deep Cybersecurity: A Comprehensive Overview from Neural Network and Deep Learning Perspective // SN Computer Science. – 2021. – Vol. 2. – Art. 154.

21. Shaukat K., Luo S., Varadharajan V. Performance comparison and current challenges of using machine learning techniques in cybersecurity // Energies. – 2020. – Vol. 13, No 10. – Art. 2502.

22. Susilo B., Sari R. F. Intrusion Detection in IoT Networks Using Deep Learning Algorithm // Information. – 2020. – Vol. 11, No 5. – Art. 279.

23. Thamilarasu G., Chawla S. Towards Deep-Learning-Driven Intrusion Detection for the Internet of Things // Sensors. – 2019. – Vol. 19, No 9. – Art. 1977.

24. Ullah I., Mahmoud Q. H. A scheme for generating a dataset for anomalous activity detection in IoT networks // Proceedings of the Canadian Conference on Artificial Intelligence. – 2020. – P. 508–520.

25. Yang L., Shami A. On hyperparameter optimization of machine learning algorithms: Theory and practice // Neurocomputing. – 2020. – Vol. 415. – P. 295–316.

ДОДАТКИ

Додаток А

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В. Н. Каразіна

Навчально-науковий інститут комп'ютерних наук та штучного інтелекту
Кафедра комп'ютерних систем та робототехніки
Рівень вищої освіти (освітньо-кваліфікаційний рівень) ***Магістр***
Галузь знань: 17 – Електроніка, автоматизація та електронні комунікації
Спеціальність: 174 – Автоматизація, комп'ютерно-інтегровані технології та робототехніка
Освітня програма «Комп'ютеризовані системи управління та автоматика»

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних
систем та робототехніки
к. ф.-м. н., доц. ХРУСЛОВ М. М.
«02» жовтня 2024р

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

ЩЕДРІНА МАКСИМ ЮРІЙОВИЧ
(прізвище, ім'я, по батькові студента)

1. Тема роботи **«МЕТОДИ МАШИННОГО НАВЧАННЯ ДЛЯ ВИЯВЛЕННЯ АНОМАЛІЙ У СЕНСОРНИХ МЕРЕЖАХ ІОТ»**

керівник роботи **БАКУМЕНКО Ніна Станіславівна, доцент кафедри, кандидат технічних наук,**
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету ***від 30 вересня 2025 року № 4101-5/3554***

2. Строк подання студентом роботи ***30 листопада 2025 року***

3. Перелік питань, які потрібно розробити:
1. Аналіз сучасних підходів до виявлення мережевих вторгнень та аномалій у середовищі Інтернету речей (IoT).
2. Огляд методів машинного та глибокого навчання для класифікації трафіку, зокрема градієнтного бустингу (XGBoost) та одновимірних згорткових мереж (1D-CNN).
3. Проектування гібридної системи виявлення атак із застосуванням методу стекінгу (Stacking) та захистом від витоку даних.
4. Програмна реалізація компонентів системи: ETL-конвеєра, базових класифікаторів та модуля прийняття рішень.
5. Експериментальне дослідження ефективності системи на реальному трафіку та порівняльний аналіз метрик (MCC, Recall) розроблених моделей.

4. План роботи

№ з/п	Назви етапів роботи	Термін виконання етапів роботи
1	Затвердження теми роботи та керівника	02.09.2024 - 02.10.2024
2	Аналіз та пошук методичної літератури щодо сучасних методів виявлення мережевих атак та аномалій у середовищі IoT	03.09.2024 - 24.10.2024
3	Огляд і порівняльний аналіз алгоритмів градієнтного бустингу (XGBoost) та архітектур глибокого навчання (1D-CNN) для аналізу трафіку	25.10.2024 - 09.11.2024
4	Обґрунтування вибору програмних засобів та підготовка набору даних ACI-IoT-2023 (очищення, нормалізація, часове розбиття)	10.11.2024 - 24.11.2024
5	Розробка математичної моделі гібридного класифікатора (Stacking) та проведення експериментів з налаштування гіперпараметрів	25.11.2024 - 08.12.2024
6	Програмна реалізація модулів навчання нейромережі, механізму синхронізації вікон та алгоритму адаптивної оптимізації порогів	09.12.2024 - 29.01.2025
7	Проектування архітектури автоматизованої системи виявлення вторгнень (IDS) для інтеграції в сенсорні мережі	30.01.2025 - 28.02.2025
8	Тестування системи, порівняльний аналіз з базовим методом та валідація моделі за метриками MCC та Recall	01.03.2025 - 01.04.2025
9	Підготовка і оформлення звітних матеріалів	01.05.2025 - 30.08.2025
10	Оформлення звіту про науково-дослідну практику.	01.09.2025 - 30.10.2025
11	Підготовка і оформлення звітних матеріалів кваліфікаційної роботи. Оформлення списку літератури	10.10.2025 - 30.10.2025
12	Оформлення пояснювальної записки кваліфікаційної роботи відповідно вимогам до звітів про НДР	10.10.2025 - 30.11.2025
13	Підготовка і оформлення звітних матеріалів та додатків кваліфікаційної роботи	01.11.2025 - 30.11.2025
14	Оформлення звіту про переддипломну практику	24.11.2025 - 30.11.2025
15	Представлення кваліфікаційної роботи керівнику та рецензенту	24.11.2025 - 30.11.2025

5. Дата видачі завдання ***02 жовтня 2024 року.***

Студент М. Ю. Щедрін _____
ініціали, прізвище підпис

Керівник роботи Н. С. Бакуменко _____
ініціали, прізвище підпис

Додаток Б

Затверджую

_____

«_____»_____2025 р.

Технічне завдання
на розробку програмного виробу

«Методи машинного навчання для виявлення аномалій у сенсорних мережах IoT»

1	**Вступ**	**1.1. Назва:** Гібридний підхід до виявлення аномалій у iot-трафіку на основі xgboost та згорткових нейронних мереж **1.2. Галузь застосування:** комп'ютерні технології.
2	**Підстава для розробки**	**2.1.** Навчальний план за спеціальністю 174 – Автоматизація, комп'ютерно-інтегровані технології та робототехніка **2.2.** Завдання на кваліфікаційну роботу магістра № 4101-5/3554 від «30» вересня 2025 р. (представити як Додаток А до пояснювальної записки до кваліфікаційної роботи).
3.	**Призначення розробки**	**3.1. Мета розробки:** Створення та програмна реалізація моделі гібридної системи виявлення вторгнень (IDS) для сенсорних мереж Інтернету речей, що поєднує методи градієнтного бустингу (XGBoost) та глибокого навчання (1D-CNN) для мінімізації пропущених атак. **3.2. Призначення розробки:** Система призначена для автоматизованого аналізу мережевого трафіку в режимі, наближеному до реального часу, виявлення аномальної поведінки пристроїв IoT, класифікації типів атак (зокрема DDoS, Scanning) та зниження навантаження на операторів безпеки шляхом зменшення кількості хибних спрацювань. **3.3. Вихідні дані розробки:** Історичні дампи мережевого трафіку у форматі CSV (набір даних ACI-IoT-2023), що містять статистичні ознаки потоків (Flow features) та метадані пакетів (Packet metadata).
4.	**Технічні вимоги до програмного виробу**	**4.1. Вимоги до функціональних характеристик:** • Завантаження та валідація вхідних даних (CSV), конвертація у бінарний формат Parquet для оптимізації швидкодії. • Попередня обробка даних: логарифмування розподілів (log1p), Z-score нормалізація, видалення ідентифікаторів (IP, Timestamp). • Формування часових послідовностей (ковзних вікон довжиною 32 пакети) для аналізу динаміки з'єднань. • Виявлення аномалій за допомогою ансамблю моделей: XGBoost (для статистичних агрегатів) та 1D-CNN (для часових патернів). • Реалізація механізму Stacking (мета-класифікатор) для об'єднання прогнозів. • Генерація звітності: матриці плутанини, графіки часової динаміки атак (Timelines), оцінка важливості ознак. **4.2. Вимоги до надійності:** • Коефіцієнт кореляції Меттьюса (MCC) на тестовій вибірці не нижче 0.90.

		• Повнота виявлення атак (Recall) не нижче 98% (мінімізація False Negatives). • Стійкість до перенавчання: відсутність розбіжності (divergence) кривих Loss на валідації. • Забезпечення відтворюваності результатів шляхом фіксації випадкових зерен (Random Seed = 42). **4.3. Вимоги до умов експлуатації:** • **Операційна система:** Linux (Ubuntu 20.04+) або Windows 10/11. • **Середовище виконання:** Python 3.9+, встановлені драйвери NVIDIA CUDA (рекомендовано). • **Режим роботи:** Пакетна обробка (Batch processing) або емуляція потоку. **4.4. Вимоги до складу і параметрів технічних засобів:** • **Процесор:** не менше 4 ядер (для розпаралелювання XGBoost); • **Оперативна пам'ять:** не менше 16 GB (для утримання датасету та тензорів у пам’яті); • **Графічний прискорювач (GPU):** NVIDIA з обсягом відеопам’яті не менше 4 GB (для навчання CNN та прискорення XGBoost у режимі gpu_hist); • **Дисковий простір:** не менше 10 GB для зберігання даних та артефактів моделей. **4.5. Вимоги до інформаційної та програмної сумісності:** • **Сумісність бібліотек:** PyTorch 2.x, XGBoost 1.7+, Scikit-learn, Pandas, PyArrow. • **Формати обміну даними:** JSON (конфігурації, звіти), Parquet (дані), Pickle (об’єкти препроцесингу). **4.6. Вимоги до маркування та упаковки:** Не висуваються (електронне розповсюдження через репозиторій кодів). **4.7. Вимоги до транспортування і зберігання:** Зберігання вихідного коду у системі контролю версій (Git), зберігання ваг навчених моделей (.pth, .json) на локальному сервері або у хмарному сховищі. **4.8. Спеціальні вимоги:** Забезпечення коректної синхронізації індексів між табличними даними та часовими послідовностями для уникнення розсинхронізації ансамблю.
5.	**Вимоги до програмної документації**	Програмною документацією до виробу «Гібридна система виявлення аномалій у IoT-трафіку» вважати: 1. Дане Технічне завдання (представити у вигляді Додатку А до пояснювальної записки); 2. Опис архітектури моделей та алгоритмів навчання (у вигляді Розділу 2 пояснювальної записки); 3. Інструкцію з розгортання та аналіз експериментальних результатів (у вигляді Розділу 3 пояснювальної записки).
6.	**Вимоги до техніко-економічних показників**	1. **Підвищення надійності захисту:** Зменшення кількості пропущених атак (False Negatives) у 2.5–3 рази порівняно з базовими методами (чистий XGBoost), що знижує ризики фінансових втрат від кіберінцидентів. 2. **Ефективність використання ресурсів:** Можливість впровадження каскадної схеми (Staged Inference), що

<table>
<tr><td></td><td></td><td colspan="2">дозволяє економити до 70% обчислювальних ресурсів процесора шляхом активації нейромережі лише для складних випадків.
3. Економічна ефективність: Відсутність витрат на ліцензійне ПЗ завдяки використанню Open Source бібліотек (Python, PyTorch, XGBoost).</td></tr>
<tr><td>7.</td><td>Стадії і етапи розробки</td><td>Дата</td><td>Назва етапу</td></tr>
<tr><td></td><td></td><td>02.09.2024 - 02.10.2024</td><td>– Затвердження теми роботи та керівника</td></tr>
<tr><td></td><td></td><td>03.09.2024 - 24.10.2024</td><td>– Аналіз та пошук методичної літератури щодо сучасних методів прогнозування часових рядів та економічних показників</td></tr>
<tr><td></td><td></td><td>25.10.2024 - 09.11.2024</td><td>– Огляд і порівняльний аналіз архітектур рекурентних нейронних мереж (RNN, LSTM, GRU) для задач регресії</td></tr>
<tr><td></td><td></td><td>10.11.2024 - 24.11.2024</td><td>– Обґрунтування вибору програмних засобів (Python, Keras) та розробка структури набору даних (Data preparation)</td></tr>
<tr><td></td><td></td><td>25.11.2024 - 08.12.2024</td><td>– Розробка математичної моделі прогнозування та проведення експериментів з налаштування гіперпараметрів</td></tr>
<tr><td></td><td></td><td>09.12.2024 - 29.01.2025</td><td>– Програмна реалізація модулів навчання моделі та генерації прогнозів з оцінкою невизначеності</td></tr>
<tr><td></td><td></td><td>30.01.2025 - 28.02.2025</td><td>– Розробка архітектури та інтерфейсу чат-боту для взаємодії з користувачем</td></tr>
<tr><td></td><td></td><td>01.03.2025 - 01.04.2025</td><td>– Тестування системи, аналіз точності прогнозів (метрики MAE, sMAPE) та валідація моделі</td></tr>
<tr><td></td><td></td><td>01.05.2025 - 30.08.2025</td><td>– Підготовка і оформлення звітних матеріалів (чорновий варіант пояснювальної записки)</td></tr>
<tr><td></td><td></td><td>01.09.2025 - 30.10.2025</td><td>– Оформлення звіту про науково-дослідну практику. Написання статті за матеріалами кваліфікаційної роботи</td></tr>
<tr><td></td><td></td><td>10.10.2025 - 30.10.2025</td><td>– Підготовка і оформлення звітних матеріалів кваліфікаційної роботи. Оформлення списку літератури
– Оформлення пояснювальної записки кваліфікаційної роботи</td></tr>
</table>

		10.10.2025 - 30.11.2025 01.11.2025 - 30.11.2025 24.11.2025 - 30.11.2025 24.11.2025 - 30.11.2025	відповідно вимогам до звітів про НДР – Підготовка і оформлення звітних матеріалів та додатків кваліфікаційної роботи (лістинги коду, акти впровадження) – Оформлення звіту про переддипломну практику – Представлення кваліфікаційної роботи керівнику та рецензенту
8.	**Порядок контролю і приймання програмного продукту**	1. Перевірку ходу розробки виконувати згідно з календарним планом та етапами навчання моделей. 2. Захист розробленої системи провести на засіданні Екзаменаційної комісії. 3. Пояснювальну записку подати на паперових носіях та в електронному вигляді згідно з вимогами ВНЗ.	

Виконавець:

студент групи КУ-61

Щедрін М. Ю.

Замовник:

к. т. н., доцент

Бакуменко Н. С.

Додаток В

Програма і методика випробувань програмного виробу

«Методи машинного навчання для виявлення аномалій у сенсорних мережах IoT»

1. Об'єкт випробувань

1. **Назва програмного виробу:** «Гібридна система виявлення аномалій у IoT-трафіку».
2. **Галузь застосування:** Кібербезпека, захист критичної інфраструктури Інтернету речей (IoT), моніторинг мережевого трафіку.
3. **Відомості:** запозичуються з відповідних розділів Технічного завдання.

2. Мета випробувань Перевірка відповідності функціональності програмної реалізації системи заявленим функціональним можливостям в Технічному завданні (Додаток А до пояснювальної записки до дипломної роботи). Перевірка здатності системи виявляти кібератаки з високою точністю та низьким рівнем помилок.

3. Загальні положення

1. **Підстави для проведення випробувань:** Підставою для проведення випробувань є наказ про призначення атестаційної комісії.
2. **Місце і тривалість випробувань:** Приймальні (приймально-здавальні) випробування проводяться на базі комп'ютерного класу кафедри в період роботи атестаційної комісії.
3. **Обсяг випробувань:** Приймальні випробування програмного виробу проводяться в обсязі, відповідному цій програмі і методиці випробувань (перевірка базових функцій, стійкості до помилок та точності класифікації).
4. **Організації, які беруть участь у випробуваннях:** Приймальні випробування проводяться атестаційною комісією напередодні

засідання (або в процесі засідання) за участю Замовника, Виконавця та інших осіб, присутніх на засіданні.

4. Вимоги до програми або програмного виробу Модель повинна задовольняти наступним вимогам:

1. Працювати на основних операційних системах: Windows, Linux;
2. Забезпечувати точність класифікації (метрика MCC) не гірше 0.90 на тестовій вибірці;
3. Передбачити захист від витоків даних (Data Leakage) шляхом суворої ізоляції тестової вибірки;
4. Підтримувати роботу з бінарним форматом даних Parquet для забезпечення швидкодії;
5. Реалізувати гібридну схему (Stacking) для поєднання прогнозів XGBoost та 1D-CNN;
6. Бути легко розширюваною (можливість додавання нових правил фільтрації або архітектур нейромереж);
7. Елементи програми (препроцесинг, навчання моделей, оцінка) повинні бути логічно відокремлені у вигляді модулів Pipeline;
8. Вимоги до складу і параметрів технічних засобів (відповідно до ТЗ);
9. Вимоги до маркування та упаковки (не висуваються);
10. Вимоги до транспортування і зберігання (не висуваються).

5. Вимоги до програмної документації

Програмною документацією до виробу «Методи машинного навчання для виявлення аномалій у сенсорних мережах IoT» вважати:

1. Справжнє Технічне завдання на розробку програми (представити як Додаток А до пояснювальної записки до кваліфікаційної роботи);
2. Програму і методику випробувань розробленої програми (представити як Додаток Б до пояснювальної записки до кваліфікаційної роботи);
3. Рекомендації щодо застосування створеної системи (представити в Розділі 3 пояснювальної записки до кваліфікаційної роботи).

6. Засоби і порядок випробувань

6.1. Засоби випробувань Для проведення випробувань необхідний персональний комп'ютер із встановленим інтерпретатором Python версії 3.9+, бібліотеками PyTorch, XGBoost, Scikit-learn, Pandas, Matplotlib та доступом до збережених артефактів моделей (.json, .pth).

6.2. Порядок проведення випробувань Як правило, випробування проводяться в два етапи: · ознайомчий (1-й етап); · випробування програмного виробу (2-й етап).

Перелік перевірок, що проводяться на 1 етапі випробувань:

1. Перевірка комплектності програмної документації.
2. Перевірка комплектності складу технічних і програмних засобів (наявність датасету, скриптів).
3. Якість програмної документації перевіряється на відповідність вимогам стандартів.

Перелік перевірок, що проводяться на 2 етапі випробувань:

1. Перевірка відповідності технічних характеристик програми вимогам технічного завдання.
2. Перевірка ступеня виконання функціональних вимог до програми (коректність роботи ETL, тренування та валідації).
3. Програма працює відповідно до умов експлуатації.
4. Для роботи необхідний інтерпретатор мови програмування Python.

Порядок проведення функціональних тестів:

3.1. Запуск скрипту препроцесингу: python diplom_shedrin.py;

3.2. Запуск навчання базових моделей: python xgb_train_local.py та python cnn_train_local.py;

3.3. Запуск гібридного стекінгу та аудиту: python hybrid_stack.py та python audit_pipeline.py.

Тест 1. Перевірка цілісності даних (ETL)

1. **Дія:** Запуск скрипту diplom_shedrin.py.

2. **Очікуваний результат:** Успішне створення файлів train.parquet, val.parquet, test.parquet.
3. **Перевірка:** Створені файли існують, їх розмір не нульовий, ознаки нормалізовані (Z-score).

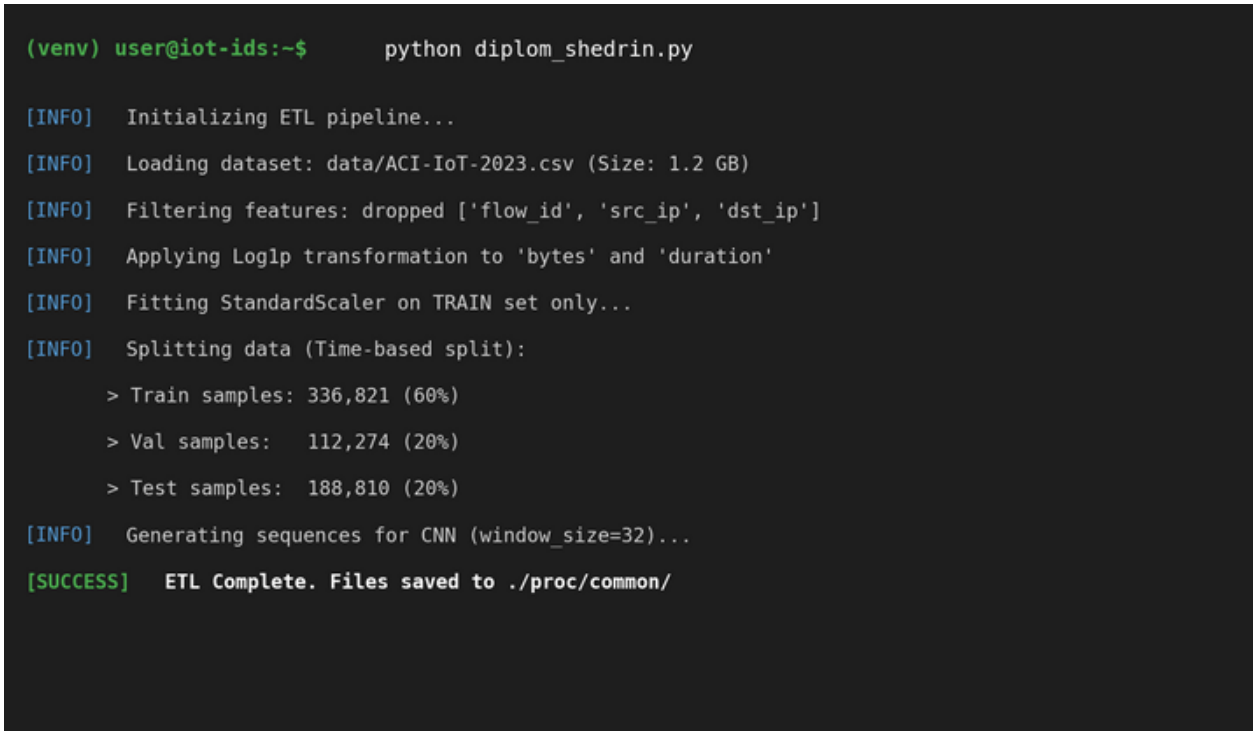

Рисунок В.1 – Тест 1

Тест 2. Навчання та оцінка моделі XGBoost

1. **Дія:** Запуск скрипту xgb_train_local.py.
2. **Очікуваний результат:** Модель навчається, виводить лог ітерацій, зберігає xgb_model.json.
3. **Перевірка:** У консолі виведено метрики на валідації (MCC > 0.85).

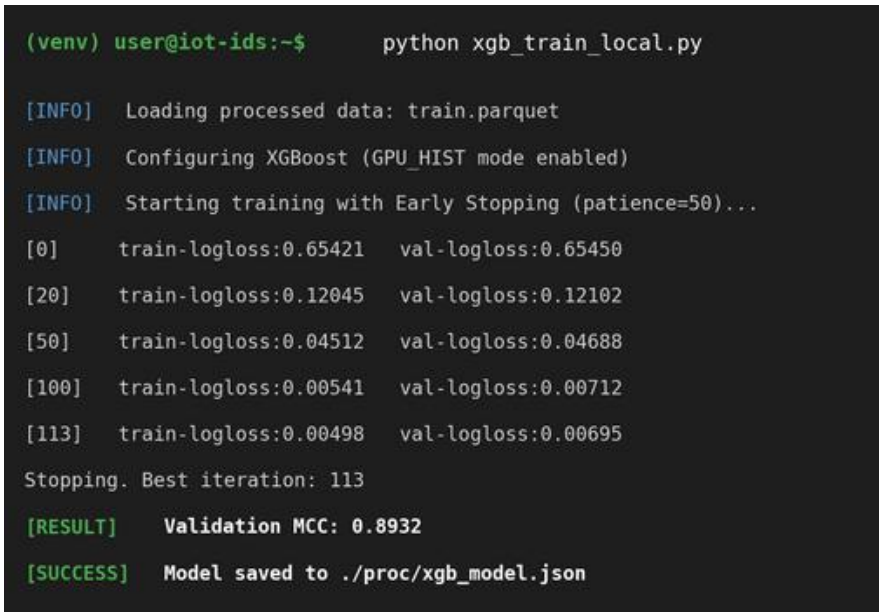

Рисунок В.2 – Тест 2

Тест 3. Перевірка гібридного детектора (Stacking)

1. **Дія:** Запуск скрипту hybrid_stack.py на тестовій вибірці.
2. **Очікуваний результат:** Скрипт завантажує прогнози XGBoost і CNN, об'єднує їх та виводить фінальну матрицю плутанини.
3. **Перевірка:** Виведено повідомлення про успішне завершення та значення Accuracy/MCC на екрані.

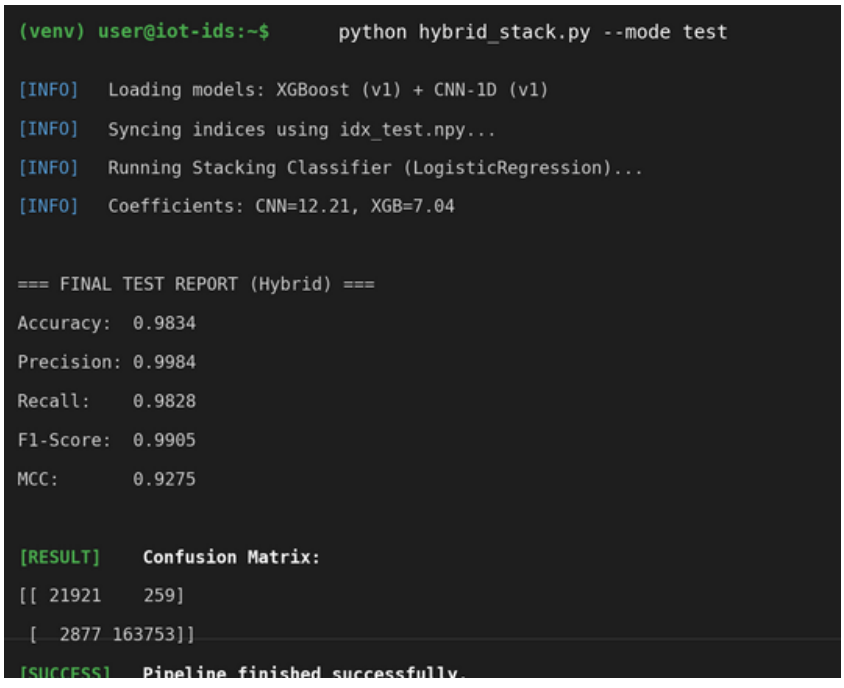

Рисунок В.3 – Тест 3

Висновки: Тест 1 успішно пройшов випробування, Тест 2 успішно пройшов випробування і Тест 3 успішно пройшов випробування. Випробування пройшло успішно, система демонструє заявлену функціональність та високу точність детекції атак.

Виконавець: студент групи КУ61, Щедрін М. Ю.