НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені Ігоря СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра математичного моделювання та аналізу даних

«До захисту допущено»
В.о. завідувача кафедри
_____ І.М. Терещенко
«__» _____ 2026 р.

Дипломна робота

на здобуття ступеня бакалавра

зі спеціальності: 113 Прикладна математика
на тему: **«Застосування методів штучного інтелекту для виявлення аномалій у мережевих даних»**

Виконав:
студент 4 курсу, групи ФІ-21
Онищенко Євгеній Олександрович

Керівник:
доцент кафедри ММАД, д-р філософії
Яйлимова Ганна Олексіївна _____

Консультант: _ _____

Рецензент: звання, степінь, посада Прізвище І.П. _____

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

Київ — 2026

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені Ігоря СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра математичного моделювання та аналізу даних

Рівень вищої освіти — перший (бакалаврський)
Спеціальність (освітня програма) — 113 Прикладна математика,
ОПП «Математичні методи моделювання, розпізнавання образів та комп'ютерного зору»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ І.М. Терещенко
«__» _____ 2026 р.

ЗАВДАННЯ
на дипломну роботу

Студент: Онищенко Євгеній Олександрович

1) **Тема роботи:** «*Застосування методів штучного інтелекту для виявлення аномалій у мережевих даних*», керівник: доцент кафедри ММАД, д-р філософії Яйлимова Ганна Олексіївна, затверджені наказом по університету № __ від «__» _____ 2026 р.

2) **Термін подання студентом роботи:** «__» _____ 2026 р.

3) **Вихідні дані до роботи:** гібридний метод виявлення мережевих вторгнень, який навчається на наборі даних мережевого трафіку та здатний класифікувати мережеві з'єднання як нормальні або аномальні.

4) **Зміст роботи:** огляд сучасних наукових досліджень у сфері мережевих систем виявлення вторгнень; аналіз існуючих методів машинного та глибокого навчання для виявлення аномалій у мережевому трафіку; формулювання математичної постановки задачі; розроблення та програмна реалізація гібридного методу виявлення вторгнень на основі Stacked Autoencoder та алгоритму XGBoost; порівняння ефективності

різних підходів за показниками якості класифікації та обчислювальної ефективності.

5) **Перелік ілюстративного матеріалу:** презентація доповіді.

6) **Дата видачі завдання:** 10 вересня 2025 р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання	Примітка
1	Вибір теми та затвердження завдання	01–20 вересня 2025 р.	Виконано
2	Аналіз сучасних підходів до виявлення вторгнень у мережевому трафіку та огляд ML/DL-методів для NIDS	01 вересня – 21 жовтня 2025 р.	Виконано
3	Вивчення особливостей датасету CIC-IDS2017, структури flow-based ознак та проблеми дисбалансу класів	21 жовтня – 5 листопада 2025 р.	Виконано
4	Формулювання математичної постановки задачі	5 листопада – 25 листопада 2025 р.	Виконано
5	Розроблення структури гібридного методу SAE–XGBoost та математичний опис його компонентів	25 листопада – 15 грядня 2025 р.	Виконано
6	Попередня обробка даних CIC-IDS2017	15 грядня – 5 січня 2025 р.	Виконано
7	Реалізація конфігурацій обраного методу	5 січня – 1 березня 2026 р.	Виконано
8	Проведення експериментів з конфігураціями SAE–XGBoost та аналіз результатів	1 березня – 15 березня 2026 р.	Виконано
9	Оформлення висновків, списку джерел, додатків та підготовка презентації до захисту	15 березня – 31 травня 2026 р.	Виконано

Студент _____ Онищенко Є.О.

Керівник _____ Яйлимова Г.О.

РЕФЕРАТ

Кваліфікаційна робота містить: 59 сторінок, 4 рисунків, 15 таблиць, 10 джерел.

Метою даної роботи є розробка та експериментальне дослідження гібридного методу, побудованого на основі методів глибокого та машинного навчання, для виявлення аномалій серед мережевого трафіку. Об'єктом дослідження є процес аналізу мережевого трафіку в системах виявлення вторгнень. Предметом дослідження є математичні моделі та алгоритми машинного і глибокого навчання, що використовуються для виявлення аномалій у мережевих даних.

У роботі проаналізовано сучасні ML/DL-підходи виявлення мережевих вторгнень та виділено ключові проблеми цієї задачі: дисбаланс класів, високу розмірність і зашумленість ознак. Запропоновано гібридний підхід, де Stacked Autoencoder формує простір нелінійних латентних ознак, а XGBoost виконує класифікацію на основі об'єднаного простору початкових та латентних характеристик.

Експериментальне дослідження виконано на датасеті CIC-IDS2017. Найкращі результати отримала модель SAE-24-Augmented-XGBoost, показавши вищу повноту виявлення аномального трафіку та меншу кількість пропущених атак порівняно з базовим XGBoost.

МЕРЕЖЕВА БЕЗПЕКА, СИСТЕМИ ВИЯВЛЕННЯ ВТОРГНЕНЬ, NIDS, МАШИННЕ НАВЧАННЯ, ГЛИБОКЕ НАВЧАННЯ, AUTOENCODER, XGBOOST, CIC-IDS2017

ABSTRACT

The qualification thesis contains: 59 pages, 4 figures, 15 tables, 10 references.

The purpose of this qualification work is to develop and experimentally investigate a hybrid network intrusion detection method based on the combination of deep learning and machine learning techniques for anomalous network traffic classification. The object of the study is the process of network traffic analysis in intrusion detection systems. The subject of the study is mathematical models and algorithms of machine learning and deep learning applied to anomaly detection in network data.

The paper analyzes modern ML/DL approaches to network intrusion detection and outlines the key challenges of this task: class imbalance, high feature dimensionality, and data noise. A hybrid approach is proposed in which a Stacked Autoencoder forms additional nonlinear latent features, while XGBoost performs classification in the combined space of original and latent characteristics.

The experimental study was conducted on the CIC-IDS2017 dataset. The SAE-24-Augmented-XGBoost model showed the best practical results, providing higher anomaly detection recall and fewer missed attacks than the baseline XGBoost model.

NETWORK SECURITY, INTRUSION DETECTION SYSTEMS, NIDS, MACHINE LEARNING, DEEP LEARNING, XG-BOOST, AUTOENCODER, CIC-IDS2017

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

IDS	Intrusion Detection System — система виявлення вторгнень
NIDS	Network-based Intrusion Detection System — мережева система виявлення вторгнень
HIDS	Host-based Intrusion Detection System — хостова система виявлення вторгнень
SAE	Stacked Autoencoder — багатошаровий автоенкодер
XGBoost	eXtreme Gradient Boosting — алгоритм градієнтного бустингу
CNN	Convolutional Neural Network — згорткова нейронна мережа
LSTM	Long Short-Term Memory — довготривала короткочасна пам'ять
FPR	False Positive Rate — хибно-позитивний коефіцієнт
TPR (Recall)	True Positive Rate — повнота (чутливість)
MSE	Mean Squared Error — середньоквадратична помилка реконструкції
SDN	Software-Defined Networking — програмно-визначені мережі

ВСТУП

Актуальність дослідження. Сучасний мережевий трафік розвивається стрімкими темпами, кіберзагрози також не стоять на місці і постійно покращуються, роблячи їх виявлення все складнішим. Через це класичні методи, такі як сигнатурні або ручні вже втратили свою актуальність.

Це зумовлює набуття актуальності мережевих систем виявлення вторгнень (англ. Network-based Intrusion Detection System, NIDS), які здійснюють аналіз трафіку та дозволяють виявляти аномальну активність у режимі, близькому до реального часу. У таких ситуаціїях методи машинного та глибокого навчання розглядаються як перспективна альтернатива класичним підходам.

Водночас практичне застосування ML/DL-моделей у NIDS має свої неділки, зокрема сильний дисбаланс класів у реальному мережевому трафіку, висока розмірність та зашумленність ознак, а також підвищений рівень хибно-позитивних спрацювань. Крім того, пропущені атаки можуть призводити до серйозних наслідків, наприклад порушення конфіденційності, можливість керування інфрастуктурою або даними компанії. У зв'язку з цим актуальним є пошук гібридних рішень, здатних закривати прогалини один одного.

Метою дослідження є розробка та експериментальне дослідження гібридного класифікатора SAE-XGBoost для виявлення аномалій у мережевому трафіку, а також оцінювання доцільності використання латентних ознак автоенкодера для підвищення повноти виявлення атак. **Для досягнення основної мети** необхідно виконати такі завдання:

1) Провести огляд наукових джерел, пов'язаних з заданим питанням;
2) Виявити ключові проблеми та обмеження сучасних NIDS;
3) Сформулювати математичну постановку задачі та розробити

структурну схему SAE-XGBoost;

4) Розробити й описати математичну модель SAE-XGBoost, схему формування латентних ознак та їх використання у класифікаторі XGBoost;

5) Обґрунтувати вибір датасету для експериментального дослідження;

6) Провести серію обчислювальних експериментів, порівняти різні конфігурації SAE-XGBoost із базовим XGBoost та проаналізувати вплив латентних ознак на якість виявлення атак.

Об'єктом дослідження є процес формування та аналізу мережевих потоків для ідентифікації аномальної активності.

Предметом дослідження є математичне та програмне забезпечення гібридного методу, що використовує Stacked Autoencoder для формування латентних ознак мережевого трафіку та XGBoost для класифікації на основі початкових і додаткових нелінійних ознак.

При розв'язанні поставлених завдань використовувались такі *методи дослідження*: теорія оптимізації (мінімізація функцій втрат), методи глибокого навчання (Stacked Autoencoder), ансамблеве навчання (XGBoost), а також статистичний аналіз для оцінювання ефективності за метриками Precision, Recall, F1-score, FPR та ROC-AUC.

Наукова новизна полягає у побудові та дослідженні гібридної схеми SAE-XGBoost, у якій латентні ознаки, сформовані Stacked Autoencoder, використовуються не лише як стиснене подання даних, а і як додаткові нелінійні характеристики для вхідного простору XGBoost.

Практичне значення полягає у розробці гібридного класифікатора мережевого трафіку, який за рахунок доповнення початкових ознак латентними характеристиками SAE забезпечує підвищення повноти виявлення аномальної активності та зменшення кількості пропущених атак.

1 СУЧАСНИЙ СТАН ДОСЛІДЖЕНЬ У СФЕРІ ВИЯВЛЕННЯ АНОМАЛІЙ У МЕРЕЖЕВИХ ДАНИХ

1.1 Теоретичні відомості про системи виявлення вторгнень

Системи виявлення вторгнень (англ. Intrusion Detection System, IDS) є одним із ключових компонентів сучасної архітектури кіберзахисту будь-якої організації, оскільки забезпечують виявлення аномальної активності та спроб несанкціонованого доступу до конфіденційних/інфраструктурних ресурсів. На відміну від превентивних засобів захисту, наприклад міжмережеві екрани або системи контролю доступу, IDS виконують функцію моніторингу та аналізу подій для виявлення потенційних загроз.

Згідно з класифікацією, наведеною в [14], системи виявлення вторгнень поділяються на дві основні групи:

– **HIDS** – аналіз поведінки окремих хостів, системних журналів та процесів;

– **NIDS** – аналіз мережевого трафіку в межах сегментів або вузлів.

У даній роботі основну увагу зосереджено на мережевих системах виявлення вторгнень NIDS, саме вони дозволяють здійснювати аналіз даних у масштабі всієї мережі та є основним інструментом безпеки інформаційних систем.

Вхідні дані NIDS подаються у вигляді множини мережевих потоків

$$\mathcal{D} = \{(\mathbf{X}_i, y_i)\}_{i=1}^{N},$$

де $\mathbf{X}_i \in \mathbb{R}^D$ – вектор ознак, що описує характеристики i-го мережевого потоку, а $y_i \in \{0,1\}$ – відповідна мітка класу, де 0 – нормальний трафік, а 1 – аномальна активність. Функціонування IDS формулюється як побудова відображення

$$F : \mathbb{R}^D \to \{0,1\},$$

яке забезпечує коректну класифікацію мережевих потоків та здатне виявляти нові типи атак, які не зустрічались до цього.

1.2 Еволюція IDS: від сигнатурного аналізу до методів машинного навчання

Перші NIDS базувалися на сигнатурному аналізі, його суть полягає у порівнянні оброблюваного трафіку з набором відомих шаблонів атак. Цей підхід має велику кількість недоліків, наприклад:

– не здатний виявляти нові атаки;

– потребує постійного оновлення бази сигнатур/шаблонів;

– має низьку ефективність для змінених або маскованих патернів.

Враховуючи ці проблеми – з'явилась необхідність у переході до методів **поведінкового аналізу**, коли аномалія визначається як відхилення від математично оціненого «нормального» трафіку. Такий підхід задається як:

$$F(\mathbf{X}) = \begin{cases} 1, & p(\mathbf{X}) < \tau, \\ 0, & p(\mathbf{X}) \geq \tau, \end{cases}$$

де $p(\mathbf{X})$ – модель оцінки ймовірності, а τ – поріг детектування.

Їх головний недолік – це те, що ці моделі не здатні моделювати складні нелінійні залежності, як у сучасному мережевому трафіку, що підтверджують Thakkar A. та Lohiya R. у своєму огляді датасетів IDS [12].

1.3 Передумови застосування методів машинного та глибокого навчання у NIDS

Обмеження сигнатурних і статистичних підходів змусили шукати нові рішення на стороні методів машинного та глибокого навчання. У

сучасному мережевому трафіку є велика кількість різних користувачів, сервісів і протоколів, тому він містить складні нелінійні залежності, які важко або неможливо описати за допомогою фіксованих правил або простих статистичних моделей [14].

Методи машинного навчання дають змогу ефективно працювати з табличними flow-based ознаками та будувати класифікатори для виявлення нормального і аномального трафіку. Методи глибокого навчання забезпечують можливість автоматично формувати інформативні представлення даних і виділяти приховані нелінійні залежності [3, 4].

Важливим етапом також вважається поява відкритих датасетів мережевого трафіку, таких як CIC-IDS2017, які дозволяють навчати та порівнювати різні моделі в умовах, наближених до реальних. Таким чином, застосування ML/DL-підходів у NIDS є логічним продовженням еволюції від сигнатурного та статистичного аналізу до автоматизованого виявлення аномалій у високорозмірних мережевих даних.

1.4 Класифікація методів машинного та глибокого навчання у NIDS

Сучасні підходи до виявлення мережевих вторгнень поділяють на три групи: класичні методи машинного навчання, моделі глибокого навчання та гібридні архітектури [2]. Їх відмінність полягає у способі формування ознак та прийняття кінцевих рішень щодо належності мережевого потоку до класу нормального трафіку або аномального. Класи наведені у таблиці Б.1.

Методи машинного навчання зберігають актуальність у задачах NIDS через високу швидкість роботи, відносну інтерпретованість та ефективність на структурованих табличних даних. XGBoost є одним із найпоширеніших ансамблевих алгоритмів для класифікації flow-based

ознак, він поєднує градієнтний бустинг дерев рішень з регуляризацією моделі. Його цільова функція може бути подана у вигляді:

$$\text{Obj} = \sum_{i=1}^{N} l(y_i, \hat{y}_i) + \sum_{t=1}^{T} \Omega(f_t),$$

де $l(y_i, \hat{y}_i)$ – функція втрат, а $\Omega(f_t)$ – регуляризаційний член, що контролює складність окремого дерева. Ефективність класичних ML-методів сильно залежить від якості простору ознак на вході, що є суттєвим обмеженням для високорозмірних, зашумлених або незбалансованих мережевих даних.

Методи глибокого навчання застосовуються для формування інформативних представлень даних та моделювання складних нелінійних залежностей [4]. Для поставленої задачі варто розглянути автоенкодерні архітектури, вони можуть формувати компактний латентний простір ознак шляхом мінімізації помилки реконструкції вхідних даних [3]. Такі представлення можуть послаблювати вплив шумових або надлишкових характеристик для подальшої класифікації.

Гібридні ML/DL-підходи поєднують переваги обох методів. У таких архітектурах моделі глибокого навчання використовуються для автоматичного формування вхідних ознак, а алгоритми машинного навчання приймають кінцевий вердикт про належність класу. Використання гібридних моделей може сприяти зниженню кількості хибних спрацювань, оскільки класифікатор працює не лише з початковими ознаками, а й з компактнішими латентними наборами. Варто також враховувати, що такі підходи мають вищу складність налаштування та потребують більше часу на навчання і запуск.

Таким чином, для задачі виявлення аномалій серед мережевих даних доцільним є розгляд саме гібридного методу, тобто поєднання автоматичного формування ознак за допомогою глибокого навчання та класифікації табличних даних методами машинного навчання.

1.5 Ключові проблеми та прогалини в існуючих підходах до NIDS

Попри активний розвиток методів машинного та глибокого навчання, побудова ефективних NIDS все ще залишається складною задачею. Аналіз сучасних досліджень дозволяє виділити три основні проблеми, які найбільше впливають на якість виявлення мережевих аномалій.

Дисбаланс класів. У мережевому трафіку нормальні запити зазвичай значно переважають кількість атак, а окремі типи аномалій (атак) можуть бути представлені в дуже малій кількості. Для CIC-IDS2017 співвідношення атак до нормального трафіку може досягати значень 1:1000 і менше. Такий дисбаланс зміщує процес навчання в бік домінуючого класу, що призводить до зниження Recall та збільшення кількості пропущених атак. Методи балансування вибірки, такі як SMOTE або надвибірка, частково вирішують цю проблему, але можуть викривляти початковий розподіл даних [1, 7].

Надмірна розмірність і зашумленість ознак. Сучасні flow-based датасети містять велику кількість статистичних, часових і поведінкових характеристик потоків. Частина таких ознак може бути корельованою, шумовою або малоінформативною, що ускладнює навчання моделей і підвищує ризик перенавчання [3, 12]. Класичні методи зменшення розмірності не завжди зберігають всі інформативні нелінійні залежності, тому обрано автоенкодерні архітектури, як ефективний інструмент формування компактного простору ознак для подальшого входу.

Обмеження ручного виділення ознак. Традиційні ML-підходи напряму залежать від якості попередньо сформованого набору ознак, який подається на вхід моделі. Мережевий трафік є динамічним і шаблони атак постійно змінюються, враховуючи це, то ручний відбір

характеристик не завжди є достатнім для повного опису більш складних поведінкових патернів [1]. Методи глибокого навчання дають змогу автоматично формувати потрібні набори даних, однак їх використання без подальшого контролю може призводити до зростання кількості хибних спрацювань.

1.6 Обґрунтування напряму дослідження

Для подальшого дослідження обрано датасет CIC-IDS2017, оскільки він містить flow-based ознаки мережевого трафіку, різні типи атак та має високий дисбаланс класів. Такі властивості роблять його оптимальним для поставленої задачі. Детальний опис структури датасету, груп атак, ознак та етапів попередньої обробки наведено у розділі 2.

На основі виявлених проблем було обрано гібридний метод SAE–XGBoost. Stacked Autoencoder застосовується для формування компактного латентного простору ознак, що дає змогу зменшити вплив шумових, надлишкових або слабкоінформативних ознак початкового набору даних. XGBoost, у свою чергу, виконує кінцеву класифікацію мережевого трафіку, враховуючи дисбаланс класів за рахунок налаштування ваг і параметрів регуляризації.

Поєднання SAE та XGBoost є оптимальним для flow-based даних, оскільки автоенкодер формує узагальнене представлення мережевих потоків, а градієнтний бустинг ефективно працює з табличними ознаками та складними нелінійними залежностями між ними. Такий підхід поєднує переваги глибокого навчання на етапі представлення даних та переваги класичних ML-алгоритмів на етапі класифікації.

Висновки до розділу 1

У першому розділі проведено аналіз сучасного стану досліджень у сфері виявлення аномалій у мережевих даних. Розглянуто роль систем виявлення вторгнень у кіберзахисті організацій, показано, що сигнатурні підходи мають суттєві обмеження через залежність від заданих наперед шаблонів атак і необхідність постійного оновлення відповідних баз.

Показано, що методи машинного та глибокого навчання є одним із перспективних напрямів розвитку сучасних NIDS, оскільки вони дають змогу працювати з високорозмірними flow-based ознаками, моделювати складні нелінійні залежності та автоматизувати процес формування інформативних представлень даних. У результаті аналізу ML-, DL- та гібридних підходів визначено основні проблеми побудови NIDS: дисбаланс класів, високу розмірність і зашумленість ознак, а також обмеження ручного відбору характеристик.

Запропоновано вибір датасету CIC-IDS2017 як експериментальної бази та використання гібридного методу SAE–XGBoost. Такий підхід поєднує автоматичне формування латентних ознак за допомогою Stacked Autoencoder із подальшою класифікацією мережевого трафіку засобами XGBoost.

Отримані результати аналізу підтверджують доцільність побудови гібридної моделі, у якій етап глибокого навчання використовується не як самостійний класифікатор, а як засіб формування компактного простору ознак для подальшої класифікації методом машинного навчання. Це формує теоретичне підґрунтя для постановки задачі, опису даних і побудови методології дослідження у наступному розділі.

2 ПОСТАНОВКА ЗАДАЧІ, ДАНІ ТА МЕТОДИ ДОСЛІДЖЕННЯ

2.1 Постановка наукової задачі

Задача виявлення аномалій у мережевих даних розглядається як задача бінарної класифікації. За наявності розмічених навчальних даних необхідно на основі вхідного вектора ознак $\mathbf{X} \in \mathbb{R}^D$ визначити мітку класу $y \in \{0,1\}$, де 0 відповідає нормальному трафіку, а 1 – аномальному.

Математично задача полягає у побудові відображення:

$$F : \mathbb{R}^D \to \{0,1\},$$

яке мінімізує очікувану зважену функцію втрат з урахуванням дисбалансу класів:

$$F^* = \arg\min_F \mathbb{E}\left[W(y) \cdot l\left(y, F(\mathbf{X})\right)\right],$$

де $W(y)$ – ваговий коефіцієнт класу, необхідний для коректного навчання моделей на незбалансованих даних.

Для вирішення поставленої задачі у роботі розглядається гібридний метод SAE–XGBoost, у якому Stacked Autoencoder формує набір вхідних ознак:

$$\mathbf{Z} = \mathrm{SAE}(\mathbf{X}),$$

де $\mathbf{Z} \in \mathbb{R}^d$ – інформативне представлення меншої розмірності ($d < D$).

У роботі розглядаються дві схеми використання отриманих ознак. У базовій конфігурації класифікатор працює лише з латентним простором:

$$\hat{y} = \mathrm{XGBoost}(\mathbf{Z}).$$

У розширеній конфігурації латентні ознаки об'єднуються з

початковими характеристиками:

$$\mathbf{U} = [\mathbf{X}, \mathbf{Z}], \qquad \hat{y} = \mathrm{XGBoost}(\mathbf{U}).$$

Така схема дає можливість перевірити, чи можуть, сформовані автоенкодером, ознаки підвищити якість виявлення атак без повної заміни початкового простору характеристик.

2.2 Завдання дослідження

Для досягнення поставленої мети необхідно виконати такі завдання:

1) Провести аналіз сучасних ML/DL-підходів для виявлення мережевих аномалій та визначити недоліки існуючих рішень;

2) Сформулювати математичну постановку задачі;

3) Підготувати експериментальну вибірку (на основі CIC-IDS2017) та виконати попередню обробку даних;

4) Розробити та навчити Stacked Autoencoder;

5) Налаштувати класифікатор XGBoost (для роботи з початковими, латентними та об'єднаними ознаками);

6) Реалізувати конфігурації SAE–XGBoost;

7) Порівняти отримані результати з базовим XGBoost за метриками Precision, Recall, F1-score, FPR та ROC-AUC.

Виконання наведених пунктів забезпечує послідовний перехід від теоретичного аналізу задачі до експериментальної перевірки запропонованого методу.

2.3 Методи дослідження

У роботі застосовуються такі наукові методи, наведено у таблиці 2.1:

Таблиця 2.1 – Методи дослідження, використані у роботі

Метод	**Застосування у роботі**
Методи аналізу даних	Очищення, нормалізація, усунення некоректних і корельованих ознак CIC-IDS2017.
Методи глибокого навчання	Побудова та навчання Stacked Autoencoder для отримання простору ознак.
Методи машинного навчання	Використання XGBoost для класифікації нормального та аномального трафіку.
Методи прикладної математики	Оптимізація функцій втрат, регуляризація, аналіз похибок.
Статистичні методи оцінювання	Аналіз результатів за Precision, Recall, F1-score, FPR та ROC-AUC.

Наведені методи забезпечують повний цикл дослідження: від підготовки вхідних даних до побудови та оцінювання моделі. В подальших розділх буде детальніше розібрано джерела даних, етапи їх попередньої обробки та математичні моделі компонентів SAE–XGBoost.

2.4 Джерела даних: особливості датасету CIC-IDS2017

Для експериментального дослідження використовується набір даних **CIC-IDS2017**, запропонований Канадським інститутом кібербезпеки [9]. Цей датасет є одним із найбільш поширених опен-соурс наборів даних для навчання та оцінювання моделей виявлення мережевих вторгнень.

У CIC-IDS2017 знаходяться зписи як нормальної мережевої активності, так і різні сценарії атак. До нормальної активності належать веб-перегляд, використання електронної пошти, передача файлів та інші типові дії користувачів у мережі. Атакуючий трафік представлений

сценаріями DoS/DDoS (GoldenEye, Hulk та Slowloris), спробами підбору паролів до SSH і FTP, веб-атаками типу SQL Injection та XSS, а також активністю ботнетів і сценаріями проникнення у внутрішню мережу.

Мережеві потоки в датасеті описуються за допомогою близько **80 flow-based ознак**, сформованих інструментом CICFlowMeter. До них належать статистики довжин пакетів, часові характеристики потоку, міжпакетні інтервали, а також поведінкові ознаки, що описують напрямок, інтенсивність і тривалість мережевої взаємодії. Така структура даних є гарним варіантом для демонстративного застосування моделей машинного навчання, але потребує попередньої обробки.

Використання CIC-IDS2017 без попередньої обробки пов'язане з низкою проблем. Зокрема, датасет характеризується дисбалансом класів, наявністю пропущених і некоректних значень, дубльованих записів, константних ознак та сильно корельованих змінних. Також варто розуміти, що висока розмірність ознакового простору може підвищувати ризик перенавчання моделей.

Таким чином, CIC-IDS2017 є придатним для експериментальної бази перевірки запропонованого гібридного методу SAE–XGBoost, оскільки поєднує різні типи мережевої активності, високу розмірність ознак і природний дисбаланс між нормальним та аномальним трафіком.

2.5 Попередня обробка даних

Попередня обробка є дуже важливим етапом формування вхідного набору ознак. Якість попередньої обробки безпосередньо впливає на ефективність моделей ML/DL [1]. У роботі підготовка даних включає очищення, формування цільової змінної, усунення корельованих ознак, нормалізацію та розбиття вибірки.

На першому етапі виконувалося очищення даних: уніфіковувались назви ознак, некоректні значення типу `Infinity` та `-Infinity` замінювалися на пропущені, після чого рядки з `NaN` та дубльовані записи

видалялись з вибірки. Частка таких записів є незначною, тому їх вилучення не призводить до суттєвої втрати даних.

Після очищення початкові мітки класів було перетворено у бінарний формат:

$$y = \begin{cases} 0, & \text{якщо } Label = \text{BENIGN}, \\ 1, & \text{якщо } Label \neq \text{BENIGN}. \end{cases}$$

Після формування цільової змінної текстова колонка `Label` вилучалася з простору вхідних ознак.

Далі виконувалось усунення корельованих ознак. Для пари ознак (X_i, X_j), якщо:

$$|\rho(X_i, X_j)| > 0.95,$$

одна з ознак вилучалася. Кореляційна матриця обчислювалася лише на навчальній вибірці, а сформований список вилучених ознак застосовувався до тестових даних, що дозволяє уникнути витоку інформації.

Для стабілізації навчання автоенкодера числові ознаки нормалізувалися методом Min–Max Scaling:

$$X_i^{(norm)} = \frac{X_i - \min(X)}{\max(X) - \min(X)}.$$

Це приводить значення ознак до інтервалу [0,1].

На завершальному етапі очищений набір даних поділявся на навчальну та тестову вибірки у співвідношенні 80/20. Розбиття виконувалося за принципом stratified split для збереження початкового співвідношення нормального та аномального трафіку в обох частинах вибірки.

2.6 Математична модель Stacked Autoencoder (SAE)

Stacked Autoencoder (SAE) – глибока нейронна мережа, що складається з кількох автоенкодерів, розташованих послідовно. Її основне

призначення у даній роботі – отримання компактнішого інформативного набору $\mathbf{Z} \in \mathbb{R}^d$ початкового вектора ознак $\mathbf{X} \in \mathbb{R}^D$, яке може використовуватися як самостійний стиснений простір або як додатковий набір характеристик для класифікатора.

2.6.1 Архітектура SAE

Нехай вхідний вектор ознак:

$$\mathbf{X} = (x_1, x_2, \ldots, x_D) \in \mathbb{R}^D.$$

SAE складається з двох частин:
Encoder – перетворює $\mathbf{X}$ у латентний простір $\mathbf{Z}$:

$$\mathbf{h}^{(1)} = f(W^{(1)}\mathbf{X} + b^{(1)}),$$

$$\mathbf{Z} = f(W^{(2)}\mathbf{h}^{(1)} + b^{(2)}).$$

Decoder – реконструює вхід:

$$\mathbf{h}^{(2)} = f(W^{(3)}\mathbf{Z} + b^{(3)}),$$

$$\mathbf{X}' = f(W^{(4)}\mathbf{h}^{(2)} + b^{(4)}).$$

тут $f(\cdot)$ – активаційна функція (ReLU, sigmoid або tanh).
У роботі використовується симетрична архітектура:

$$D \to \frac{D}{2} \to d \to \frac{D}{2} \to D.$$

Подібні симетричні архітектури автоенкодерів застосовуються у дослідженнях flow-based IDS [3, 4].

2.6.2 Функція втрат автоенкодера

Автоенкодер навчається шляхом мінімізації помилки реконструкції між оригінальним вектором $\mathbf{X}$ та реконструйованим $\mathbf{X}'$.

Базова функція втрат:

$$\mathcal{L}_{rec} = \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{X}_i - \mathbf{X}'_i\|_2^2 .$$

Для запобігання перенавчанню застосовується регуляризація:

$$\mathcal{L}_{reg} = \lambda \sum_{l} \|W^{(l)}\|_2^2.$$

Повний функціонал оптимізації:

$$\mathcal{L}_{SAE} = \mathcal{L}_{rec} + \mathcal{L}_{reg}.$$

Навчання SAE спрямоване на збереження ключової інформації про вхідні ознаки при одночасному обмеженні складності моделі.

2.6.3 Отримання латентного простору

Після навчання SAE використовується лише Encoder:

$$\mathbf{Z} = SAE_{enc}(\mathbf{X}).$$

Простір $\mathbf{Z}$ має такі властивості:

– менша розмірність порівняно з початковим простором;

– зменшений вплив шуму та надлишкових ознак;

– наявність узагальнених нелінійних представлень;

– можливість використання як самостійного входу для XGBoost або як додаткових ознак до початкових характеристик.

2.6.4 Причини використання SAE у системах виявлення вторгнень

Роль SAE у запропонованому підході узагальнено в таблиці 2.2:

Таблиця 2.2 – Роль SAE у запропонованому підході

Властивість SAE	**Значення для задачі NIDS**
Формування латентного простору	Дозволяє отримати компактне представлення.
Нелінійне перетворення ознак	Дає змогу враховувати складні залежності між характеристиками мережевого трафіку.
Можливість розширення простору ознак	Дає можливість використовувати латентні характеристики для доповнення початкових ознак.
Зменшення впливу шуму	Потенційно послаблює вплив нестабільних або надлишкових ознак.
Неконтрольоване навчання	Дозволяє формувати представлення без безпосереднього використання міток класів.

Але потрібно також враховувати, що навчання глибоких автоенкодерів є обчислювально затратним і потребує використання GPU або значного часу навчання на CPU.

2.7 Математична модель XGBoost

Алгоритм eXtreme Gradient Boosting (XGBoost) є одним із найпоширеніших ансамблевих методів для класифікації структурованих даних. У роботах [2, 7] XGBoost демонструє високі результати на

IDS-датасетах, особливо якщо використовувати додатково корекцію дисбалансу класів.

Основою XGBoost є модель:

$$\hat{y}_i = \sum_{t=1}^{T} f_t(\mathbf{q}_i),$$

де кожна функція f_t – дерево рішень, а $\mathbf{q}_i$ – вектор ознак, що подається на вхід XGBoost. У різних конфігураціях $\mathbf{q}_i$ може відповідати або відповідному латентному вектору $\mathbf{Z}_i$, або об'єднаному $[\mathbf{X}_i, \mathbf{Z}_i]$.

2.7.1 Цільова функція

На кожній ітерації t мінімізується наступна цільова функція:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{N} W_i \cdot l\left(y_i, \hat{y}_i^{(t-1)} + f_t(\mathbf{q}_i)\right) + \Omega(f_t).$$

У цьому виразі $l(y_i,\hat{y}_i)$ – логістична функція втрат, W_i – ваговий коефіцієнт класу, а $\Omega(f_t)$ – регуляризаційний член дерева. Для задачі бінарної класифікації логістична функція має такий вигляд:

$$l(y, \hat{y}) = -\left[y \log(\hat{y}) + (1-y) \log(1-\hat{y})\right].$$

Регуляризаційний член дерева має вигляд:

$$\Omega(f_t) = \gamma K + \frac{\lambda}{2} \sum_{j=1}^{K} w_j^2,$$

де K – кількість листових вузлів дерева, w_j – вага j-го листового вузла, а γ та λ – параметри регуляризації.

2.7.2 Апроксимація Тейлора другого порядку

Щоб оптимізувати $\mathcal{L}^{(t)}$, XGBoost використовує розклад Тейлора:

$$\mathcal{L}^{(t)} \approx \sum_{i=1}^{N} \left[g_i f_t(\mathbf{q}_i) + \frac{1}{2} h_i f_t(\mathbf{q}_i)^2 \right] + \Omega(f_t),$$

де g_i та h_i – відповідно градієнт і друга похідна функції втрат за поточним прогнозом моделі на попередній ітерації:

$$g_i = \frac{\partial l\left(y_i, \hat{y}_i^{(t-1)}\right)}{\partial \hat{y}_i^{(t-1)}}, \qquad h_i = \frac{\partial^2 l\left(y_i, \hat{y}_i^{(t-1)}\right)}{\partial \left(\hat{y}_i^{(t-1)}\right)^2}.$$

У результаті задача зводиться до оптимізації ваг у листових вузлах дерева:

$$w_j^* = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda}.$$

XGBoost використовує інформацію про перші та другі похідні функції втрат для побудови чергового дерева, що дозволяє ефективно зменшувати помилку моделі на кожній ітерації навчання.

2.7.3 Корекція дисбалансу класів

Великим плюсом XGBoost є можливість використання ваг W_i для компенсації дисбалансу.

Для задачі IDS, де:

$$|\text{Attack}| \ll |\text{Normal}|,$$

вага позитивного класу задається:

$$W_i = \begin{cases} \dfrac{|\text{Normal}|}{|\text{Attack}|}, & \text{якщо } y_i = 1, \\ 1, & \text{якщо } y_i = 0. \end{cases}$$

У роботах [2, 6] показано, що правильно підібраний коефіцієнт може підвищити Recall атак, знизити False Negative Rate і дає змогу контролювати рівень False Positive Rate.

2.7.4 Причини використання XGBoost у задачах виявлення аномалій

Роль XGBoost у запропонованому підході узагальнено в таблиці 2.3:

Таблиця 2.3 – Роль XGBoost у запропонованому підході

Властивість XGBoost	**Значення для задачі NIDS**
Градієнтний бустинг дерев	Ефективна класифікація табличних flow-based ознак.
Регуляризація	Запобігає перенавчанню.
Підтримка ваг класів	Дозволяє враховувати дисбаланс між класами.
Гнучкість вхідного простору	Може працювати з початковими, латентними або об'єднаними ознаками.

У запропонованому методі XGBoost формує кінцеве рішення про належність мережевого потоку до нормального або аномального (атака) класу.

2.8 Метрики оцінювання ефективності

Найважливішими є метрики, що базуються на матриці помилок. Матриця помилок (Confusion Matrix) включає елементи:

	Прогноз: Норма	Прогноз: Атака
Факт: норма	TN	FP
Факт: атака	FN	TP

де TP – кількість правильно виявлених атак, TN – кількість правильно класифікованого нормального трафіку, FP – кількість хибно-позитивних спрацювань, а FN – кількість пропущених атак. Метрики оцінювання наведено у таблиці 2.4

Таблиця 2.4 – Метрики оцінювання ефективності IDS

Метрика	**Формула**	**Значення для задачі IDS**
Recall	$\frac{TP}{TP+FN}$	Характеризує здатність моделі виявляти атаки. Низьке значення означає збільшення кількості пропущених інцидентів.
Precision	$\frac{TP}{TP+FP}$	Показує, яка частка об'єктів, класифікованих як атаки, дійсно є атаками. Важлива для зменшення кількості хибних спрацювань.
F1-score	$\frac{2 \cdot Precision \cdot Recall}{Precision+Recall}$	Відображає баланс між Recall та Precision, що є важливим в умовах дисбалансу класів.
FPR	$\frac{FP}{FP+TN}$	Визначає частку нормального трафіку, помилково класифікованого як атака.
ROC-AUC	–	Оцінює здатність моделі розділяти нормальний та аномальний трафік за різних порогів класифікації.

Для оцінювання моделей у роботі використовується не одна окрема метрика, а комплекс показників, що дозволяє більш об'єктивно оцинити результати роботи.

Висновки до розділу 2

У другому розділі сформовано методологічне підґрунтя для розв'язання задачі виявлення мережевих аномалій. Задачу описано як бінарну класифікацію з урахуванням сильного класового дисбалансу. Як експериментальну базу обрано сучасний датасет CIC-IDS2017, що вимагає специфічної попередньої обробки через високу розмірність, наявність шумових та корельованих ознак.

Для подолання зазначених проблем обґрунтовано використання гібридної архітектури SAE–XGBoost. Математично описано процес нелінійного стискання вхідного простору ознак за допомогою Stacked Autoencoder, мінімізації помилки реконструкції та подальшої класифікації градієнтним бустингом. Продемонстровано доцільність вибору XGBoost'а для кінцевої класифікації, завдяки підтримці зважування класів і регуляризації.

Сформована методологія включає в себе порівняння трьох підходів: класифікації на початкових ознаках, класифікації на латентному просторі SAE та класифікації на розширеному просторі, що поєднує початкові й латентні характеристики. Для оцінювання ефективності використовується комплекс метрик Recall, Precision, F1-score, FPR та ROC-AUC.

3 ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ГІБРИДНОГО МЕТОДУ SAE-XGBOOST

У попередніх розділах було сформульовано задачу виявлення аномального мережевого трафіку, обґрунтовано вибір датасету CIC-IDS2017 та побудовано математичну модель SAE-XGBoost. Метою даного розділу є розробка запропонованого методу та оцінювання його ефективності порівняно з базовими моделями машинного навчання, на основі запропонованих метрик.

3.1 Загальна схема експериментального дослідження

Загальна схема етапів експериментального дослідження:

1) завантаження та об'єднання даних CIC-IDS2017;

2) очищення вибірки від некоректних, пропущених та нескінченних значень;

3) приведення цільової змінної до бінарного вигляду: нормальний або аномальний трафік;

4) поділ даних на навчальну та тестову вибірки зі збереженням співвідношення класів;

5) навчання Stacked Autoencoder на навчальних даних без використання цільових міток;

6) формування латентних ознак за допомогою SAEдля навчальної та тестової вибірок;

7) навчання класифікатора XGBoost на отриманих ознаках різної розмірності;

8) формування розширеного простору ознак за допомогою об'єднання початкових характеристик та отриманих від SAE ознаками;

9) навчання базової моделі XGBoost для порівняння;

10) обчислення метрик якості та порівняльний аналіз отриманих результатів.

У роботі основний акцент зроблено на порівнянні трьох підходів:

– класифікація мережевого трафіку алгоритмом XGBoost на початкових нормалізованих ознаках;

– класифікація алгоритмом XGBoost лише на ознаках, сформованих за допомогою SAE;

– класифікація алгоритмом XGBoost на розширеному просторі, у якому початкові ознаки доповнюються додатковим характеристиками від SAE.

Таке порівняння дає змогу визначити, який спосіб використання автоенкодерних ознак є більш доцільним: повна заміна початкового простору або його розширення додатковими нелінійними характеристиками.

3.2 Програмна реалізація та середовище проведення експериментів

Програмну реалізацію експериментального дослідження виконано мовою Python. У процесі реалізації використовуються такі бібліотеки: `NumPy`, `pandas`, `scikit-learn`, `TensorFlow/Keras`, `XGBoost`, `Matplotlib`

Для забезпечення коректності порівняння всі моделі оцінюються на однаковій тестовій вибірці. Параметри попередньої обробки, зокрема масштабування ознак, визначаються лише на навчальній частині даних і надалі застосовуються до тестової вибірки без повторного навчання. Такий підхід забезпечує об'єктивність оцінювання для кожної конфігурації.

3.3 Формування експериментальної вибірки та підготовка даних

Для проведення експериментального дослідження використано табличну версію датасету CIC-IDS2017. Ці файли містять попередньо обчислені характеристики мережевих потоків, а також мітки типів трафіку.

На початковому етапі всі CSV-файли було завантажено та об'єднано в одну таблицю. Початкова вибірка містила 2 830 743 мережевих потоків та 79 стовпців, серед яких 78 числових ознак і одна категоріальна змінна `Label`, що задає тип мережевої активності.

Оскільки задача полягає у бінарній класифікації, початкові мітки було перетворено до вигляду:

$$y = \begin{cases} 0, & \text{якщо трафік належить до класу BENIGN,} \\ 1, & \text{якщо трафік відповідає будь-якому типу атаки.} \end{cases}$$

Таким чином, клас 0 відповідає нормальному трафіку, а клас 1 – аномальній активності.

3.3.1 Очищення даних

Перед подальшим навчанням моделей виконано попереднє очищення даних. У назвах ознак було вилучено зайві пробіли, проведено перевірку на нескінченні значення та пропуски.

У процесі очищення виявлено 2 867 об'єктів із некоректними значеннями в ознаках Flow Bytes/s і Flow Packets/s. Їх частка становила приблизно 0,1% від початкової вибірки, тому ці рядки було видалено без суттєвого впливу на загальну структуру даних.

Додатково виконано перевірку на повні дублікати записів. Було

виявлено та вилучено 307 078 дублів. Видалення дублікатів необхідно виконувати для коректного експериментального порівняння, оскільки повторювані записи можуть потрапити одночасно до навчальної та тестової вибірок, що створить штучне завищення метрик.

Після очищення остаточна вибірка містила 2 520 798 унікальних мережевих потоків.

3.3.2 Розподіл класів

Розподіл класів у сформованій вибірці мав такий вигляд, який продемонстровано у таблиці 3.1:

Таблиця 3.1 – Розподіл класів після очищення даних

Клас	**Кількість об'єктів**	**Частка вибірки**
Нормальний трафік	2 095 057	83,11%
Аномальний трафік	425 741	16,89%

Це підтверджує наявність дисбалансу класів, характерного для задач такого типу. Також варто зазначити, що такі типи атак як: Heartbleed, Web Attack – Sql Injection та Infiltration містили лише 11, 21 і 36 об'єктів відповідно. Це додатково підтверджує доцільність використання моделей і метрик, орієнтованих на роботу з незбалансованими даними.

3.3.3 Формування навчальної та тестової вибірок

Після очищення дані було поділено на навчальну та тестову вибірки у співвідношенні 80/20 за допомогою стратифікованого розбиття.

Сформована навчальна вибірка містила 2 016 638 об'єктів, тестова – 504 160. Розподіл класів у цих вибірках наведено у таблиці 3.2.

Таблиця 3.2 – Розподіл класів у навчальній та тестовій вибірках

Вибірка	**Клас**	**Кількість об'єктів**	**Частка**
Навчальна	Нормальний трафік	1 676 045	83,11%
Навчальна	Аномальний трафік	340 593	16,89%
Тестова	Нормальний трафік	419 012	83,11%
Тестова	Аномальний трафік	85 148	16,89%

Використавши стратифікований поділ, змогли зберегти однакову пропорцію класів у навчальній і тестовій вибірках.

3.3.4 Усунення надлишкових корельованих ознак

Ознаки, значення коефіцієнта кореляції яких перевищувало

$$|\rho(X_i, X_j)| > 0.95,$$

вважалися сильно взаємопов'язаними, тому для кожної такої пари – була вилучена одна із ознак.

За результатами процедури було видалено 23 надлишкові ознаки. До них належали частина характеристик зворотного потоку, статистики довжин пакетів, IAT-ознаки, TCP-прапорці, subflow-ознаки та характеристики простою.

Після усунення корельованих ознак кількість вхідних характеристик зменшилася з 78 до 55. Відповідно, розмірність матриць ознак становила:

$$X_{\text{train}} \in \mathbb{R}^{2\,016\,638 \times 55}, \qquad X_{\text{test}} \in \mathbb{R}^{504\,160 \times 55}.$$

Після усунення надлишкових корельованих ознак тримали компактніший простір характеристик, який надалі був використаний для нормалізації та навчання моделей.

3.3.5 Нормалізація ознак

Оскільки ознаки мережевих потоків мають різні масштаби та діапазони значень, перед навчанням було виконано нормалізацію(Min-Max Scaling):

$$X_i^{(norm)} = \frac{X_i - \min(X)}{\max(X) - \min(X)}.$$

Після перетворення всі значення навчальної вибірки належали проміжку $[0; 1]$. У тестових даних було знайдено декілька значень, що виходили за межі цього діапазону. Таких значень було лише 13, менше ніж 0,001% усіх елементів тестової матриці, для узгодження діапазону вхідних значень – їх також було обмежено інтервалом $[0; 1]$.

У результаті сформовано остаточні матриці ознак:

$$X_{\text{train}}^{\text{scaled}} \in \mathbb{R}^{2\,016\,638 \times 55}, \qquad X_{\text{test}}^{\text{scaled}} \in \mathbb{R}^{504\,160 \times 55}.$$

Ці дані використовуються на наступному етапі для навчання Stacked Autoencoder, формування компактніших наборів ознак і побудови різних конфігурацій SAE-XGBoost.

Фрагмент програмної реалізації попереднього очищення (формування бінарної цільової ознаки, очищення вибірки, стратифікований поділ даних, усунення сильно корельованих характеристик та нормалізація ознак) наведено у додатку 3.7.4.

3.4 Налаштування архітектури Stacked Autoencoder

На вхід кожного автоенкодера подавалися попередньо очищені, нормалізовані та відфільтровані ознаки з попереднього розділу:

$$X_{\text{train}}^{\text{scaled}} \in \mathbb{R}^{2\,016\,638 \times 55}, \qquad X_{\text{test}}^{\text{scaled}} \in \mathbb{R}^{504\,160 \times 55}.$$

Для дослідження впливу ступеня стискання була обрана побудова трьох варіантів конфігурації автоенкодера з різною розмірністю простору:

$$d_{\text{latent}} \in \{8, 16, 24\}.$$

Усі моделі мали симетричну архітектуру енкодер-декодер. У прихованих шарах використовувалася функція активації ReLU, а у вихідному шарі – sigmoid, оскільки всі вхідні ознаки після Min–Max нормалізації належали інтервалу $[0; 1]$.

Досліджені архітектури наведено у Таблиці 3.3.

Таблиця 3.3 – Досліджені архітектури Stacked Autoencoder

Модель	**Архітектура**	**Кількість параметрів**
SAE-8	$55 \to 32 \to 16 \to 8 \to 16 \to 32 \to 55$	4 959
SAE-16	$55 \to 40 \to 24 \to 16 \to 24 \to 40 \to 55$	7 287
SAE-24	$55 \to 40 \to 32 \to 24 \to 32 \to 40 \to 55$	8 719

Кількість параметрів відповідає загальній кількості навчуваних ваг і зміщень у шарах автоенкодера. Реалізації архітектури SAE-24 наведено у додатку 3.7.4. Навчання автоенкодерів виконувалося у режимі реконструкції вхідних даних – цільовим виходом моделі був той самий вектор ознак, що й подавався на вхід. Для оптимізації використовувалася середньоквадратична помилка реконструкції:

$$\mathcal{L}_{SAE} = \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|_2^2 ,$$

де $\mathbf{x}_i$ – початковий вектор ознак, а $\hat{\mathbf{x}}_i$ – його реконструкція, отримана на виході автоенкодера.

Усі SAE-моделі навчалися з однаковими базовими параметрами, які зібрано у таблиці 3.4:

Таблиця 3.4 – Параметри навчання Stacked Autoencoder

Параметр	**Значення**
Оптимізатор	Adam
Початкова швидкість навчання	0,001
Максимальна кількість епох	50
Розмір пакета	4096
Частка валідаційної вибірки	10% від навчальних даних
Адаптивне зменшення швидкості навчання	ReduceLROnPlateau

Графік помилок наведено у додатку 3.7.4

Для початкової моделі SAE-8 після 50 епох навчання було отримано:

$$\mathcal{L}_{train} = 0{,}001036, \qquad \mathcal{L}_{val} = 0{,}001048, \qquad \mathcal{L}_{test} = 0{,}001026.$$

Близькі значення навчальної, валідаційної та тестової помилок свідчать про стабільний процес навчання, без вираженого перенавчання.

Для порівняння якості реконструкції моделей із різною латентною розмірністю було обчислено середньоквадратичну помилку на тестовій вибірці, зібрано у таблиці 3.5 і наведено графіки у додатку 3.7.4.

Таблиця 3.5 – Помилка реконструкції SAE-моделей на тестовій вибірці

Модель	**Розмірність латентного простору**	**Test reconstruction loss**
SAE-8	8	0,001026
SAE-16	16	0,000826
SAE-24	24	0,000606

Отримані результати показують, що збільшення розмірності простору з 8 до 24 поступово зменшує помилку реконструкції. Очікуваний результат, м'якіше стискання дозволяє зберегти більшу частину інформації про початкові мережеві ознаки.

Після завершення навчання з кожного автоенкодера було виділено енкодерну частину, яка перетворювала початковий 55-вимірний вектор ознак на відповідне більш стисле представлення:

$$\mathbf{x}_i \in \mathbb{R}^{55} \to \mathbf{z}_i \in \mathbb{R}^{d_{\text{latent}}}.$$

У результаті отримали такі латентні матриці:

$$X_{\text{train}}^{\text{latent-8}} \in \mathbb{R}^{2\,016\,638\times 8}, \qquad X_{\text{test}}^{\text{latent-8}} \in \mathbb{R}^{504\,160\times 8},$$

$$X_{\text{train}}^{\text{latent-16}} \in \mathbb{R}^{2\,016\,638\times 16}, \qquad X_{\text{test}}^{\text{latent-16}} \in \mathbb{R}^{504\,160\times 16},$$

$$X_{\text{train}}^{\text{latent-24}} \in \mathbb{R}^{2\,016\,638\times 24}, \qquad X_{\text{test}}^{\text{latent-24}} \in \mathbb{R}^{504\,160\times 24}.$$

Ці представлення надалі використовувалися для побудови декількох варіантів досліджуваної гібридної моделі. Окремо було розглянуто як схему класифікації лише на латентних ознаках, так і розширену архітектуру (об'єднання ознак, сформованих SAE, з нормалізованими початковими характеристиками).

3.5 Налаштування класифікатора XGBoost та варіанти досліджуваних моделей

Після отримання латентних представлень мережевих потоків було виконано навчання класифікаторів (XGBoost). У роботі XGBoost використовується як основний ансамблевий класифікатор, здатний працювати з табличними даними, враховувати нелінійні залежності між ознаками та коригувати дисбаланс класів за допомогою параметра `scale_pos_weight`.

Після очищення даних частка аномального трафіку становила 16,89%, тому для компенсації дисбалансу використовувався коефіцієнт:

$$\texttt{scale\_pos\_weight} = \frac{N_{\text{normal}}}{N_{\text{attack}}} \approx 4{,}921.$$

Такий коефіцієнт збільшує внесок об'єктів класу атак під час оптимізації функції втрат і зменшує ризик зміщення моделі у бік домінуючого класу (нормального трафіку).

Для забезпечення коректного порівняння всі XGBoost–моделі навчалися з однаковою базовою конфігурацією гіперпараметрів, наведеною у Таблиці Б.2.

Додатково було відокремлено 10% даних від навчальної вибірки (валідаційна підвибірка), для перевірки узагальнювальної здатності моделей. Валідаційна вибірка застосовувалася для оцінювання значення `logloss` та механізму ранньої зупинки. Тестова вибірка використовувалася лише для фінального обчислення метрик якості.

Було розглянуто п'ять конфігурацій:

1) **XGBoost** – базова модель, яка навчається на 55 нормалізованих ознаках після очищення даних та вилучення сильно корельованих характеристик:

$$X^{\text{scaled}} \in \mathbb{R}^{N\times 55} \to \text{XGBoost}.$$

2) **SAE–8–XGBoost** – гібридна модель, у якій XGBoost навчається лише на 8 латентних ознаках, отриманих за допомогою SAE-8:

$$X^{\text{scaled}} \in \mathbb{R}^{N\times 55} \to Z^{(8)} \in \mathbb{R}^{N\times 8} \to \text{XGBoost}.$$

3) **SAE–16–XGBoost** – гібридна модель з 16-вимірним латентним простором:

$$X^{\text{scaled}} \in \mathbb{R}^{N\times 55} \to Z^{(16)} \in \mathbb{R}^{N\times 16} \to \text{XGBoost}.$$

4) **SAE–24–XGBoost** – гібридна модель, у якій класифікація

виконується на основі 24 латентних ознак:

$$X^{\text{scaled}} \in \mathbb{R}^{N\times 55} \to Z^{(24)} \in \mathbb{R}^{N\times 24} \to \text{XGBoost}.$$

5) **SAE–24–Augmented–XGBoost** – розширена гібридна модель, у якій до 24 латентних ознак, сформованих SAE-24, додається ще 55 нормалізованих початкових ознак. У результаті XGBoost навчається на об'єднаному просторі розмірності 79:

$$\left[X^{\text{scaled}}, Z^{(24)}\right] \in \mathbb{R}^{N\times 79} \to \text{XGBoost}.$$

Фрагменти програмної реалізації двох основних гібридних конфігурацій наведено у додатках: навчання SAE–24–XGBoost у додатку 3.7.4, реалізацію SAE–24–Augmented-XGBoost у додатку 3.7.4.

Порівняння моделей SAE–8–XGBoost, SAE–16–XGBoost і SAE–24–XGBoost дає змогу оцінити вплив ступеня стискання ознак на якість класифікації. Порівняння SAE–24–Augmented-XGBoost та XGBoost дає змогу окремо перевірити, чи покращують додаткові нелінійні SAE-ознаки якість класифікації, без вилучення початкового простору характеристик.

Найкращі boosting-ітерації, визначені за валідаційною вибіркою наведено у Таблиці 3.6.

Таблиця 3.6 – Найкращі boosting-ітерації та значення валідаційної функції втрат

Модель	**Best iteration**	**Best validation logloss**
XGBoost	952	0,003415
SAE-8-XGBoost	1996	0,034366
SAE-16-XGBoost	1999	0,029255
SAE-24-XGBoost	1999	0,028545
SAE-24-Augmented-XGBoost	699	0,003462

Наведені значення демонструють, що моделі, побудовані виключно на латентних ознаках, потребували більшої кількості boosting-ітерацій та мали значно вищу валідаційну помилку, ніж класифікатор на повному просторі ознак, але розширена модель SAE-24-Augmented-XGBoost досягла валідаційної якості, близької до базового XGBoost.

3.6 Результати експериментального дослідження

Після навчання всіх конфігурацій було виконано їх оцінювання на однаковій тестовій вибірці, яка містила 504 160 мережевих потоків. Для порівняння моделей використано метрики, запропоновані та описані в попередніх розділах: Accuracy, Precision, Recall, F1-score, False Positive Rate та ROC-AUC. Отримані результати наведено у Таблиці Б.3.

Результати свідчать, що всі моделі демонструють високі значення Recall та ROC-AUC, однак істотно відрізняються за Precision, F1-score і FPR. Найкращі результати серед моделей, побудованих лише на латентних ознаках, отримано для SAE-24-XGBoost, тоді як розширена модель SAE-24-Augmented-XGBoost продемонструвала якість, близьку або кращу ніж базовий XGBoost. Узальнену кількість FP і FN досліджуваних моделей продемонстровано у таблиці 3.7

Таблиця 3.7 – Кількість хибно-позитивних і хибно-негативних рішень досліджуваних моделей

Модель	**FP – хибні тривоги**	**FN – пропущені атаки**
XGBoost	426	29
SAE-8-XGBoost	8 537	294
SAE-16-XGBoost	7 655	203
SAE-24-XGBoost	7 470	246
SAE-24-Augmented-XGBoost	452	18

Наведені значення показують, що розширена модель SAE–24–Augmented–XGBoost має найменшу кількість пропущених атак серед усіх досліджуваних конфігурацій. Детальний аналіз цих результатів наведено в наступному підрозділі.

3.7 Аналіз отриманих результатів

Отримані експериментальні результати дозволяють зробити декілька важливих висновків щодо ролі Stacked Autoencoder у побудові гібридної моделі для виявлення аномалій у мережевому трафіку.

3.7.1 Вплив розмірності латентного простору

Перший етап аналізу стосується моделей, у яких XGBoost працював лише з латентними ознаками, сформованими автоенкодером (SAE-8-XGBoost, SAE-16-XGBoost і SAE-24-XGBoost). Основні зміни метрик при переході від 8 до 24 латентних ознак наведено у Таблиці 3.8.

Таблиця 3.8 – Вплив збільшення розмірності латентного простору

Показник	**SAE-8-XGBoost**	**SAE-24-XGBoost**
Precision	0,908589	0,919131
F1-score	0,950537	0,956534
FPR	0,020374	0,017828
FP	8 537	7 470
Test reconstruction loss	0,001026	0,000606

Порівняння моделей SAE-8-XGBoost і SAE-24-XGBoost свідчить, що збільшення латентної розмірності покращує якість конфігурацій, побудованих виключно на латентних ознаках: зростають Precision і F1-score, зменшується FPR, кількість хибно-позитивні спрацювань, помилка реконструкції.

Але варто відмітити, що найкраще значення Recall отримав SAE-16-XGBoost(0,997616), це показує, що існує компроміс між повнотою та кількістю хибних тривог.

3.7.2 Обмеження використання лише латентних ознак

Попри покращення показників при збільшенні латентного простору, усі моделі, що було описані вище, сильно поступалися базовому XGBoost, який працював на початковому відфільтрованому просторі з 55 ознак.

Наприклад, SAE-24-XGBoost продемонстрував:

$$F1 = 0{,}956534, \qquad FPR = 0{,}017828,$$

тоді як базовий XGBoost показав такі результати:

$$F1 = 0{,}997334, \qquad FPR = 0{,}001017.$$

Це означає, що використання автоенкодера як повної заміни початкових ознак призводить до втрати частини важливої інформації для класифікації. SAE навчається мінімізувати помилку реконструкції, тобто відновлювати загальну структуру вхідних даних, але при цьому не використовує цільові мітки класів. Латентний простір може добре зберігати статистичні властивості мережевих потоків, однак не обов'язково оптимально відокремлює найкращі ознаки для класифікації.

3.7.3 Переваги розширеної гібридної моделі

Основним результатом дослідження стало порівняння базового XGBoost з розширеною моделлю SAE–24–Augmented–XGBoost. У цій архітектурі SAE-ознаки не замінювали початкові характеристики, як у розгянутих моделях до цього, а навпаки доповнювали їх. Таким чином, класифікатор отримував як первинний відфільтрований простір ознак, так і додаткове нелінійне подання, сформоване автоенкодером.

За інтегральними метриками обидві моделі показали практично однакову якість:

$$F1_{\text{XGBoost}} = 0{,}997334, \qquad F1_{\text{SAE-Augmented}} = 0{,}997247.$$

Значення ROC-AUC для обох моделей також збігається з точністю до шостого знака:

$$ROC\text{-}AUC = 0{,}999979.$$

А для Recall модель SAE-24-Augmented-XGBoost показала вище значення:

$$Recall = 0{,}999789.$$

Особливо важливим є те, що кількість пропущених атак зменшилася:

$$FN_{\text{XGBoost}} = 29, \qquad FN_{\text{SAE-Augmented}} = 18.$$

Отже, додавання стислих SAE-ознак зменшило кількість хибно-негативних рішень на:

$$\frac{29 - 18}{29} \cdot 100\% \approx 37{,}9\%.$$

Для поставленої задачі саме хибно-негативні помилки є особливо критичними, оскільки відповідають атакам, які залишаються не поміченими.

3.7.4 Підсумкова оцінка запропонованого підходу

На основі проведених експериментів модель SAE-24-Augmented-XGBoost можна розглядати як основну гібридну конфігурацію. Її перевага полягає не у максимізації усіх інтегральних метрик одночасно, а у здатності зберігати майже той самий або такий же рівень F1-score і ROC-AUC, що й базовий XGBoost, демонструючи найвищу повноту виявлення атак та найменшу кількість хибно-негативних рішень.

Таким чином, дане дослідження підтвердило доцільність використання SAE не як повної заміни початкового набору ознак, а як джерело додаткових нелінійних характеристик, які розширюють інформаційний простір для подальшої класифікації. Саме така схема дозволила отримати значне покращення для практичного використання.

Висновки до розділу 3

У третьому розділі проведено експериментальну оцінку запропонованого гібридного методу на базі датасету CIC-IDS2017. У результаті фільтрації дублікатів та висококорельованих ознак – розмірність вхідних даних зменшено з 78 до 55 характеристик.

Дослідження трьох конфігурацій SAE з розмірністю 8, 16 та 24

показало, що розширення прихованого простору очікувано знижує помилку реконструкції на тестовій вибірці (мінімум досягнуто при SAE-24: 0,0006). Оцінка моделей, що використовували лише стисле представлення ознак, підтвердила, що така конфігурація показує слабші результати, ніж навчання на початковому наборі даних.

Для усунення цього недоліку реалізовано комбіновану архітектуру SAE-24-Augmented-XGBoost, де 24 латентні ознаки об'єднано з 55 вихідними параметрами. Таке рішення дозволило досягти високої точності класифікації ($Recall \approx 0{,}9998$, $F1 \approx 0{,}997$, $ROC\text{-}AUC \approx 0{,}9999$).

Найважливішим практичним результатом стало зменшення кількості хибно-негативних рішень (FN) з 29 (базовий XGBoost) до 18, що означає скорочення пропущених атак на 37,9%.

ВИСНОВКИ

У дипломній роботі було розроблено та експериментально досліджено гібридний підхід для виявлення аномального мережевого трафіку на основі поєднання Stacked Autoencoder та XGBoost.

У ході роботи проаналізовано сучасні підходи для побудови NIDS та визначено їх основні проблеми: дисбаланс класів, високу розмірність і зашумленість ознак, залежність моделей від якості вхідного простору ознак, складність або неможливість ручного підбору ознак. На основі цього обґрунтовано використання гібридної ML/DL–моделі, де Stacked Autoencoder формує нелінійні представлення даних, а XGBoost виконує на них класифікацію.

Для експериментального дослідження використано датасет CIC–IDS2017. Початково налічував 2 830 743 мережевих потоків і 78 числових ознак, було очищено від некоректних значень, пропусків і дублікатів. Після видалення сильно корельованих характеристик сформовано вибірку з 2 520 798 потоків і 55 ознак, яку поділено на навчальну та тестову частини у відношенні 80/20.

У межах дослідження реалізовано три конфігурації Stacked Autoencoder із латентною розмірністю 8, 16 і 24. Показано, що збільшення розмірності латентного простору зменшує помилку реконструкції: для SAE–8 вона становила 0,001026, для SAE–16 – 0,000826, а для SAE–24 – 0,000606. Це підтвердило, що м'якше стискання краще зберігає інформацію про початкові мережеві ознаки.

Результати показали, що використання лише латентного простору поступається класифікації на початковому наборі ознак. Найкраща latent–only конфігурація SAE–24–XGBoost досягла F1–score 0,956534 та FPR 0,017828, що є нижчим результатом порівняно з базовим XGBoost. Натомість найкращу ефективність показала модель SAE–24–Augmented–XGBoost, у якій 24 латентні ознаки були об'єднані з

55 початковими характеристиками. Вона досягла Recall 0,999789, F1–score 0,997247 та ROC–AUC 0,999979, кількість пропущених атак зменшилася з 29 до 18 порівняно з базовим XGBoost. Отримані результати свідчать, що Stacked Autoencoder для таких задач доцільніше використовувати не як повну заміну початкових ознак, а як засіб їх доповнення.

Подальші дослідження можна спрямувати на багатокласову класифікацію атак, тестування підходу на інших IDS–датасетах і перевірку його ефективності в умовах потокової обробки трафіку.

ПЕРЕЛІК ПОСИЛАНЬ

1. Holdbrook, R., et al. (2024). Network-Based Intrusion Detection for Industrial and Robotics Systems: A Comprehensive Survey. *Electronics*, 13(22), 4440.

2. Fan, Z. (2025). Research on network intrusion detection based on XGBoost algorithm and multiple machine learning algorithms. *Theoretical and Natural Science*, 31(1), 162–167.

3. Korniszuk, K., & Sawicki, B. (2024). Autoencoder-Based Anomaly Detection in Network Traffic. In *2024 25th International Conference on Computational Problems of Electrical Engineering (CPEE)* (pp. 1–4). IEEE.

4. Hore, S., et al. (2024). A Sequential Deep Learning Framework for a Robust and Resilient Network Intrusion Detection System. *Computers & Security*, 144, 103923.

5. Rosay, A., et al. (2022). Network Intrusion Detection: A Comprehensive Analysis of CIC-IDS2017. *Information Systems Security*, 25–54.

6. Maddu, M., & Rao, Y. N. (2024). Network Intrusion Detection and Mitigation in SDN Using Deep Learning Models. *International Journal of Information Security*, 23, 849–862.

7. Pansare, S., et al. (2024). Hybrid Machine Learning Algorithm for Intrusion Detection Systems. In *2024 International Conference on Distributed Computing and High Performance Computing (DCHPC)*. IEEE.

8. Mirsky, Y., Doitshman, T., Elovici, Y., & Shabtai, A. (2018). Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection. In *Network and Distributed System Security Symposium (NDSS)*.

9. Sharafaldin, I., et al. (2018). Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)* (pp. 108–116).

10. He, H., & Garcia, E. A. (2009). Learning from Imbalanced Data.

IEEE Transactions on Knowledge and Data Engineering, 21(9), 1263-1284.

11. Fawcett, T. (2006). An Introduction to ROC Analysis. *Pattern Recognition Letters*, 27(8), 861-874.

12. Thakkar, A., & Lohiya, R. (2023). A review of the advancement in intrusion detection datasets. *Procedia Computer Science*, 167, 636–645.

13. Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785–794). ACM.

14. Lansky, J. F., et al. (2021). A Comprehensive Survey on Machine Learning and Deep Learning Based Intrusion Detection Systems. *IEEE Access*.

15. Moustafa, N., & Slay, J. (2015). UNSW-NB15: A Comprehensive Data Set for Network Intrusion Detection Systems. In *Military Communications and Information Systems Conference (MilCIS)* (pp. 1–6). IEEE.

ДОДАТОК А ФРАГМЕНТИ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ ТА ГРАФІКИ

Попередня обробка

```
1
2  df.columns = df.columns.str.strip()
3  df["Target"] = (df["Label"] != "BENIGN").astype("int8")
4
5  rows_before = df.shape[0]
6  df.dropna(inplace=True, ignore_index=True)
7  rows_after = df.shape[0]
8
9  rows_before_duplicates = df.shape[0]
10 df = df.drop_duplicates().reset_index(drop=True)
11 rows_after_duplicates = df.shape[0]
12
13
14 X = df.drop(columns=["Label", "Target"])
15 y = df["Target"]
16 X_train, X_test, y_train, y_test = train_test_split(
17     X,
18     y,
19     test_size=0.20,
20     random_state=42,
21     stratify=y
22 )
23
24 correlation_threshold = 0.95
25 corr_matrix = X_train.corr().abs()
26
27 upper_triangle = corr_matrix.where(
28     np.triu(np.ones(corr_matrix.shape), k=1).astype(bool)
29 )
30
31 high_corr_features = [
32     column for column in upper_triangle.columns
33     if any(upper_triangle[column] > correlation_threshold)
34 ]
```

```
35
36 X_train_reduced = X_train.drop(columns=high_corr_features)
37 X_test_reduced = X_test.drop(columns=high_corr_features)
38
39 scaler = MinMaxScaler()
40
41 X_train_scaled = scaler.fit_transform(X_train_reduced)
42 X_test_scaled = scaler.transform(X_test_reduced)
```

Побудова SAE–24

```
1  input_dim = X_train_scaled.shape[1]
2  input_layer_24 = Input(shape=(input_dim,), name="input_layer_24")
3
4  encoded_24 = Dense(40, activation="relu", name="encoder24_dense_1")(input_layer_24
       )
5  encoded_24 = Dense(32, activation="relu", name="encoder24_dense_2")(encoded_24)
6  latent_24 = Dense(24, activation="relu", name="latent_space_24")(encoded_24)
7
8  decoded_24 = Dense(32, activation="relu", name="decoder24_dense_1")(latent_24)
9  decoded_24 = Dense(40, activation="relu", name="decoder24_dense_2")(decoded_24)
10 output_layer_24 = Dense(input_dim, activation="sigmoid", name="reconstruction_24")
       (decoded_24)
11
12 autoencoder_24 = Model(
13     inputs=input_layer_24,
14     outputs=output_layer_24,
15     name="stacked_autoencoder_latent24"
16 )
17
18 encoder_24 = Model(
19     inputs=input_layer_24,
20     outputs=latent_24,
21     name="sae_encoder_latent24"
22 )
23
24 autoencoder_24.compile(
```

```
    optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
    loss="mse"
)
```

Навчання SAE-24-XGBoost

```
X_train_latent_24_fit, X_val_latent_24, y_train_fit_24, y_val_24 =
    train_test_split(
    X_train_latent_24,
    y_train,
    test_size=0.10,
    random_state=42,
    stratify=y_train
)

negative_count_24 = (y_train_fit_24 == 0).sum()
positive_count_24 = (y_train_fit_24 == 1).sum()

scale_pos_weight_24 = negative_count_24 / positive_count_24

sae24_xgb = XGBClassifier(
    n_estimators=2000,
    learning_rate=0.05,
    max_depth=6,
    min_child_weight=1,
    subsample=0.8,
    colsample_bytree=0.8,
    gamma=0.0,
    reg_alpha=0.0,
    reg_lambda=1.0,
    scale_pos_weight=scale_pos_weight_24,
    objective="binary:logistic",
    eval_metric="logloss",
    tree_method="hist",
    n_jobs=-1,
    random_state=42,
    early_stopping_rounds=50
```

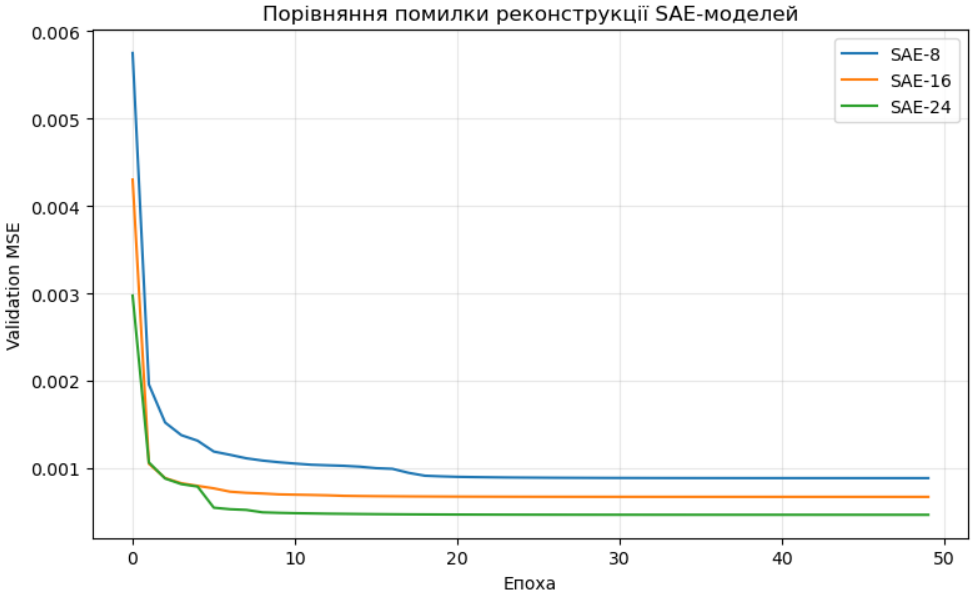
```
)

start_time = time.perf_counter()

sae24_xgb.fit(
    X_train_latent_24_fit,
    y_train_fit_24,
    eval_set=[(X_val_latent_24, y_val_24)],
    verbose=True
)

sae24_xgb_training_time = time.perf_counter() - start_time
```

SAE-24-Augmented-XGBoost

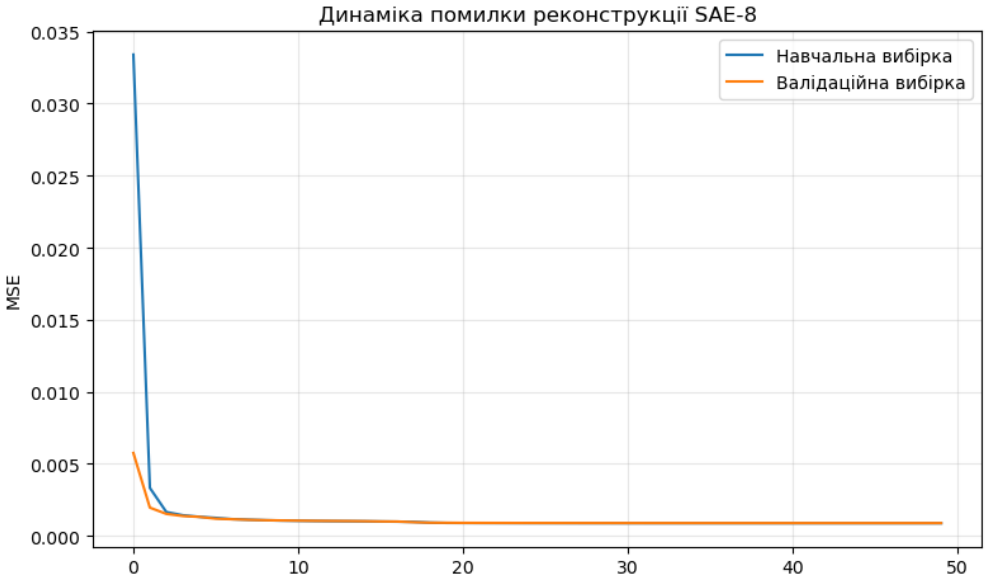
```
X_train_augmented_24 = np.hstack([
    X_train_scaled,
    X_train_latent_24
]).astype("float32")

X_test_augmented_24 = np.hstack([
    X_test_scaled,
    X_test_latent_24
]).astype("float32")

X_train_aug24_fit, X_val_aug24, y_train_fit_aug24, y_val_aug24 = train_test_split(
    X_train_augmented_24,
    y_train,
    test_size=0.10,
    random_state=42,
    stratify=y_train
)

negative_count_aug24 = (y_train_fit_aug24 == 0).sum()
positive_count_aug24 = (y_train_fit_aug24 == 1).sum()

scale_pos_weight_aug24 = negative_count_aug24 / positive_count_aug24
```

```
23
24 sae24_aug_xgb = XGBClassifier(
25     n_estimators=3000,
26     learning_rate=0.05,
27     max_depth=6,
28     min_child_weight=1,
29     subsample=0.8,
30     colsample_bytree=0.8,
31     gamma=0.0,
32     reg_alpha=0.0,
33     reg_lambda=1.0,
34     scale_pos_weight=scale_pos_weight_aug24,
35     objective="binary:logistic",
36     eval_metric="logloss",
37     tree_method="hist",
38     n_jobs=-1,
39     random_state=42,
40     early_stopping_rounds=50
41 )
42
43 start_time = time.perf_counter()
44
45 sae24_aug_xgb.fit(
46     X_train_aug24_fit,
47     y_train_fit_aug24,
48     eval_set=[(X_val_aug24, y_val_aug24)],
49     verbose=True
50 )
51
52 sae24_aug_xgb_training_time = time.perf_counter() - start_time
```

Порівняння динаміки валідаційної помилки реконструкції SAE-моделей



Динаміка помилки реконструкції




Динаміка помилки реконструкції SAE-16
Навчальна вибірка
Валідаційна вибірка
MSE
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
10
20
30
40
50


Динаміка помилки реконструкції SAE-24
Навчальна вибірка
Валідаційна вибірка
MSE
0.025
0.020
0.015
0.010
0.005
0.000
0
10
20
30
40
50

ДОДАТОК Б ПОРІВНЯЛЬНИЙ АНАЛІЗ МЕТОДІВ І ТЕХНОЛОГІЙ ВИЯВЛЕННЯ АНОМАЛІЙ У МЕРЕЖЕВИХ ДАНИХ

Таблиця Б.1 – Класи методів виявлення аномалій у NIDS

Клас методів	**Приклади**	**Особливості застосування у NIDS**
ML–методи	SVM, Random Forest, XGBoost	Ефективні для табличних flow-based даних, мають нижчі вимоги до ресурсів, але залежать від якості вхідних даних.
DL–методи	AE, SAE, CNN, LSTM/GRU	Дозволяють автоматично формувати ознаки та моделювати нелінійні залежності, але потребують більші обчислювальні ресурси.
Гібридні підходи	SAE–XGBoost, CNN–SVM, LSTM–RF	Поєднують автоматичне формування ознак та класифікацію ML-алгоритмами.

Таблиця Б.2 – Гіперпараметри класифікатора XGBoost

Параметр	**Значення**
Кількість дерев, `n_estimators`	2000
Швидкість навчання, `learning_rate`	0,05
Максимальна глибина дерева, `max_depth`	6
Мінімальна вага дочірнього вузла, `min_child_weight`	1
Частка об'єктів для кожного дерева, `subsample`	0,8
Частка ознак для кожного дерева, `colsample_bytree`	0,8
L1-регуляризація, `reg_alpha`	0
L2-регуляризація, `reg_lambda`	1
Вага позитивного класу, `scale_pos_weight`	$\approx 4{,}921$
Цільова функція, `objective`	`binary:logistic`
Метрика валідації	`logloss`
Метод побудови дерев, `tree_method`	`hist`
Критерій ранньої зупинки, `early_stopping_rounds`	50

Таблиця Б.3 – Результати класифікації досліджуваних моделей на тестовій вибірці

Модель	**Accuracy**	**Precision**	**Recall**	**F1-score**	**FPR**	**ROC-AUC**
XGBoost	0,999098	0,995020	0,999659	0,997334	0,001017	0,999979
SAE-8-XGBoost	0,982484	0,908589	0,996547	0,950537	0,020374	0,999167
SAE-16-XGBoost	0,984414	0,917333	0,997616	0,955791	0,018269	0,999412
SAE-24-XGBoost	0,984695	0,919131	0,997111	0,956534	0,017828	0,999445
SAE-24-Augmented-XGBoost	0,999068	0,994719	0,999789	0,997247	0,001079	0,999979