МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління

БІЛОКУР Владислав Васильович

Інтелектуальна система моніторингу безпеки комп'ютерної мережі на основі глибокого аналізу трафіку / Intelligent System for Monitoring Computer Network Security Based on Deep Traffic Analysis

Спеціальність 122 – Комп'ютерні науки
Освітньо-професійна програма – Комп'ютерні науки

Кваліфікаційна робота

Виконав студент групи КН-43
В.В. Білокур

Науковий керівник:
д.т.н., професор М.П. Комар

Кваліфікаційну роботу допущено до захисту
«___»_____ 2026 р.

Завідувач кафедри
_____ Н.В. Дзюбановська

Тернопіль – 2026

Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «бакалавр»
спеціальність: 122 – Комп'ютерні науки
освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Н.В. Дзюбановська
«____» _____20__ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
БІЛОКУРУ Владиславу Васильовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи
Інтелектуальна система моніторингу безпеки комп'ютерної мережі на основі глибокого аналізу трафіку / Intelligent System for Monitoring Computer Network Security Based on Deep Traffic Analysis
керівник роботи д.т.н., професор М. П. Комар
затверджені наказом по університету від 01 грудня 2025 року № 894.
2. Строк подання студентом закінченої кваліфікаційної роботи 25 травня 2026 р.
3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.
4. Основні питання, які потрібно розробити
– проаналізувати предметну область мережевої безпеки та визначити вимоги до інтелектуальної системи моніторингу вторгнень;
– виконати огляд і порівняльний аналіз сучасних методів виявлення вторгнень на основі глибокого навчання та обґрунтувати вибір підходу;
– визначити джерела даних і сформувати набір інформативних ознак трафіку; організувати підготовку даних для навчання та тестування на еталонних датасетах;
– спроєктувати архітектуру системи, що включає модулі збору трафіку, попередньої обробки, нейромережевого класифікатора, прийняття рішення та представлення результатів;
– реалізувати прототип системи та експериментально оцінити його ефективність з використанням стандартних метрик і порівняльного аналізу;
– виконати інтерпретацію результатів, визначити обмеження запропонованого рішення та сформулювати практичні рекомендації щодо застосування й подальшого вдосконалення.
5. Перелік графічного матеріалу в роботі
– узагальнена схема інтелектуальної системи моніторингу мережевої безпеки;
– конвеєр підготовки даних для інтелектуальної системи моніторингу;

– структурна схема нейромережевої моделі;

– архітектура програмної системи моніторингу мережевої безпеки (компонентна діаграма).

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 01 грудня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Затвердження теми кваліфікаційної роботи, ознайомлення з літературними джерелами та складання плану роботи	до 01.01. 2026 р.	
2	Написання 1 розділу кваліфікаційної роботи	до 01.02. 2026 р.	
3	Написання 2 розділу кваліфікаційної роботи	до 15.03.2026 р.	
4	Написання 3 розділу кваліфікаційної роботи	до 01.05. 2026 р.	
5	Представлення попереднього варіанту кваліфікаційної роботи, перевірка та внесення змін керівником	до 15.05.2026 р.	
6	Опрацювання зауважень та представлення завершеного варіанту кваліфікаційної роботи. Підготовка супроводжуючих документів	до 25.05.2026 р.	
7	Перевірка кваліфікаційної роботи на оригінальність тексту	до 30.05.2026 р.	
8	Оформлення кваліфікаційної роботи та отримання допуску до захисту	до 10.06.2026 р.	
9	Подання кваліфікаційної роботи до захисту на засіданні атестаційної комісії	до 10.06. 2026 р.	

Студент_____ В. В. Білокур

підпис

Керівник роботи_____ д.т.н., професор М. П. Комар

підпис

АНОТАЦІЯ

Кваліфікаційна робота на тему «Інтелектуальна система моніторингу безпеки комп’ютерної мережі на основі глибокого аналізу трафіку» на здобуття освітнього ступеня «бакалавр» зі спеціальності 122 «Комп’ютерні науки» освітньої програми «Комп’ютерні науки» написана обсягом в 85 сторінок і містить 10 ілюстрацій, 5 таблиць, 4 додатки та 40 використаних джерел.

Метою кваліфікаційної роботи є розроблення інтелектуальної системи моніторингу безпеки комп’ютерної мережі на основі глибокого аналізу трафіку, що забезпечує виявлення вторгнень шляхом навчання та застосування моделей глибокого навчання, а також підтримує збір, підготовку та візуалізацію результатів у процесі моніторингу.

Методи досліджень включають аналіз та узагальнення науково-технічної літератури, методи машинного і глибокого навчання, алгоритми рекурентних нейронних мереж, методи попередньої обробки мережевого трафіку, а також методи програмної реалізації та тестування.

Розроблено інтелектуальну систему моніторингу безпеки комп’ютерної мережі на основі глибокого аналізу трафіку, яка забезпечує приймання й підготовку мережевих даних, обробку PCAP- і flow-представлень, формування ознак, навчання та застосування нейромережевої моделі для виявлення вторгнень, журналювання подій і візуалізацію результатів.

Результати дослідження можуть бути використані для створення та вдосконалення програмних засобів виявлення вторгнень, автоматизації аналізу мережевого трафіку та підвищення ефективності систем кіберзахисту інформаційно-телекомунікаційних мереж.

Ключові слова: МЕРЕЖЕВА БЕЗПЕКА, СИСТЕМА ВИЯВЛЕННЯ ВТОРГНЕНЬ, IDS, ГЛИБОКЕ НАВЧАННЯ, МАШИННЕ НАВЧАННЯ, МЕРЕЖЕВИЙ ТРАФІК, BIGRU, MULTIHEAD ATTENTION, КІБЕРБЕЗПЕКА.

ANNOTATION

Qualification work on the topic «Intelligent System for Monitoring Computer Network Security Based on Deep Traffic Analysis» for Bachelor's degree on speciality 122 «Computer Science» educational and professional program «Computer Science» is written on 85 pages and it contains 10 figures, 5 tables, 4 annexes and 40 sources.

The purpose of the qualification work is to develop an intelligent system for monitoring the security of a computer network based on deep traffic analysis, which ensures intrusion detection through the training and application of deep learning models, as well as supports the collection, preparation, and visualization of results during the monitoring process.

The research methods include the analysis and generalization of scientific and technical literature, machine learning and deep learning methods, recurrent neural network algorithms, methods of preliminary processing of network traffic, as well as methods of software implementation and testing.

An intelligent system for monitoring the security of a computer network based on deep traffic analysis has been developed. The system provides the reception and preparation of network data, processing of PCAP and flow representations, feature generation, training and application of a neural network model for intrusion detection, event logging, and visualization of results.

The research results can be used to create and improve software tools for intrusion detection, automate network traffic analysis, and increase the efficiency of cybersecurity systems for information and telecommunication networks.

Keywords: NETWORK SECURITY, INTRUSION DETECTION SYSTEM, IDS, DEEP LEARNING, MACHINE LEARNING, NETWORK TRAFFIC, BIGRU, MULTIHEAD ATTENTION, CYBERSECURITY.

ЗМІСТ

ВСТУП

Сучасні комп’ютерні мережі функціонують в умовах зростання інтенсивності та різноманітності кіберзагроз, зокрема атак на прикладні служби, промислові сегменти, транспортні мережі та канали шифрованого обміну даними. У таких умовах підвищується актуальність систем моніторингу безпеки, здатних не лише фіксувати відомі сигнатури, а й виявляти аномалії та нові сценарії вторгнень на основі аналізу мережевого трафіку. При цьому практичне застосування ускладнюється великими обсягами даних, зміною профілів нормальної поведінки, дисбалансом класів та наявністю зашумлених або неповних ознак, що знижує ефективність традиційних евристичних і статистичних методів.

Перспективним напрямом розв’язання зазначених проблем є використання глибокого навчання та суміжних інтелектуальних підходів, які забезпечують автоматизоване виділення інформативних закономірностей у трафіку та підвищують точність класифікації атак за умови належної підготовки даних. Зокрема, у наукових працях запропоновано моделі на основі рекурентних мереж, механізмів уваги та їхніх гібридних комбінацій для покращення розпізнавання складних часових залежностей у потоках даних [5, 24]. Паралельно розвиваються підходи до виявлення вторгнень у спеціалізованих доменах: бортові мережі транспортних засобів та CAN-шини [2, 7], промисловий інтернет і технологічні процеси [15, 17], SCADA-системи у критичній інфраструктурі [18], а також задачі детекції шкідливого TLS-трафіку із застосуванням мультимодальних ознак [25]. Водночас зберігається потреба у побудові узгодженої архітектури інтелектуальної системи, що поєднує збір трафіку, формування ознак, навчання моделі та інтерпретоване представлення результатів у режимі моніторингу.

Вагомий внесок у розвиток інтелектуальних систем виявлення атак зроблено в роботах М. Комара та співавторів, де розглянуто високопродуктивні та адаптивні системи детекції, у тому числі з використанням паралельних нейромережевих моделей, компресії параметрів трафіку та ідей штучної імунної системи [3, 9, 11–13]. Окремі аспекти зменшення розмірності даних, побудови

ієрархічних класифікаторів і підвищення стійкості інформаційних систем до кібератак висвітлено у публікаціях [29–32]. Наявність таких напрацювань створює методичні передумови для розроблення інтелектуальної системи моніторингу мережевої безпеки на основі глибокого аналізу трафіку з орієнтацією на відтворюваність експериментів і практичну придатність.

Актуальність теми підсилюється нормативним контекстом забезпечення кібербезпеки в Україні, що визначає необхідність розвитку технічних і організаційних заходів протидії кіберзагрозам [33]. У прикладному вимірі важливим є використання стандартних еталонних наборів даних для порівняльної оцінки моделей, зокрема CIC-IDS2017 та UNSW-NB15, які широко застосовуються у дослідженнях з виявлення вторгнень [1, 20]. Практичну основу для формування ознак і первинного аналізу мережевих пакетів забезпечують інструменти захоплення трафіку та його декодування, зокрема Wireshark [22]. Додатково враховуються підходи до інтелектуалізації детекції на основі нейронних мереж, запропоновані у ранніх роботах щодо нейромережевих IDS [19] та застосування DBN для виявлення атак [34], а також патентні розробки у сфері нейромережевих штучних імунних систем [35, 36].

Мета роботи полягає у розробленні інтелектуальної системи моніторингу безпеки комп’ютерної мережі на основі глибокого аналізу трафіку, що забезпечує виявлення вторгнень шляхом навчання та застосування моделей глибокого навчання, а також підтримує збір, підготовку та візуалізацію результатів у процесі моніторингу.

Для досягнення поставленої мети необхідно розв’язати такі завдання:

– проаналізувати предметну область мережевої безпеки та визначити вимоги до інтелектуальної системи моніторингу вторгнень;

– виконати огляд і порівняльний аналіз сучасних методів виявлення вторгнень на основі глибокого навчання та обґрунтувати вибір підходу;

– визначити джерела даних і сформувати набір інформативних ознак трафіку; організувати підготовку даних для навчання та тестування на еталонних датасетах;

– спроєктувати архітектуру системи, що включає модулі збору трафіку, попередньої обробки, нейромережевого класифікатора, прийняття рішення та представлення результатів;

– реалізувати прототип системи та експериментально оцінити його ефективність з використанням стандартних метрик і порівняльного аналізу;

– виконати інтерпретацію результатів, визначити обмеження запропонованого рішення та сформулювати практичні рекомендації щодо застосування й подальшого вдосконалення.

Об’єктом дослідження є мережевий трафік комп’ютерних мереж і процеси виникнення та прояву комп’ютерних атак у інформаційно-телекомунікаційних системах.

Предметом дослідження є методи та алгоритми інтелектуального виявлення вторгнень на основі глибокого навчання, а також принципи побудови програмно-алгоритмічної архітектури системи моніторингу безпеки за даними мережевого трафіку.

Результати кваліфікаційної роботи апробовані та опубліковані у матеріалах IV Всеукраїнської науково-практичної конференції молодих вчених і студентів «Інтелектуальні комп’ютерні системи та мережі» (ІКСМ), м. Тернопіль, ЗУНУ, 20 травня 2026 р. (додаток Г).

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

1 АНАЛІЗ ПРОБЛЕМНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Опис предметної області

Предметна область дослідження охоплює забезпечення мережевої безпеки шляхом безперервного моніторингу трафіку та виявлення ознак комп'ютерних атак в інформаційно-телекомунікаційних системах. Практична потреба у таких рішеннях зумовлена тим, що значна частина критичних інцидентів проявляється на рівні мережевих взаємодій: змінюються профілі обміну даними, порушуються нормальні часові закономірності передачі пакетів, з'являються нетипові комбінації протоколів, портів і служб. У нормативному полі України питання протидії кіберзагрозам визначено як пріоритетне, що підкреслює актуальність розроблення технічних засобів виявлення та реагування на інциденти [33].

Ключовою ланкою мережевого захисту є системи виявлення вторгнень (Intrusion Detection Systems, IDS) та системи запобігання вторгненням (Intrusion Prevention Systems, IPS), які орієнтовані на ідентифікацію підозрілої активності у трафіку та підтримку прийняття рішень щодо реагування. У контексті даної роботи основний акцент робиться на мережевих IDS (NIDS), оскільки саме вони забезпечують спостереження за потоком мережевих подій у режимі, наближеному до реального часу. Разом із тим, вимога до «інтелектуальності» системи передбачає не лише фіксацію відомих сигнатур, а й здатність узагальнювати поведінкові патерни та виявляти складні або модифіковані сценарії атак.

Основним джерелом інформації для NIDS є мережевий трафік, який може аналізуватися на різних рівнях представлення даних: на рівні пакетів (packet-based analysis) або на рівні потоків/сеансів (flow-based analysis). Пакетний підхід дозволяє детально аналізувати заголовки протоколів та вміст корисного навантаження, однак він є ресурсомістким і часто ускладнюється шифруванням. Потоковий підхід агрегує пакетні події у логічні сесії обміну та описує їх набором статистичних і часових характеристик, що підвищує масштабованість моніторингу та спрощує побудову моделей машинного навчання. У практичних дослідженнях детекції вторгнень широко використовуються еталонні набори

даних, сформовані як сукупності потокових/статистичних ознак, зокрема CIC-IDS2017 та UNSW-NB15 [1, 20]. Вибір цих датасетів є доцільним також з огляду на можливість відтворюваного порівняння результатів різних підходів.

Побудова системи моніторингу безпеки на основі глибокого аналізу трафіку передбачає послідовне виконання кількох взаємопов'язаних процесів: захоплення трафіку, його фільтрація та нормалізація, формування ознак, застосування моделі виявлення та інтерпретація результатів для оператора або суміжних підсистем. На етапі збору та первинного аналізу трафіку важливу роль відіграють інструменти мережевої діагностики, серед яких поширеним є Wireshark, що забезпечує захоплення пакетів, декодування протоколів та формування репрезентативних вибірок для подальшої обробки [22]. У межах даної предметної області такий інструментарій використовується як джерело контрольованих даних і як засіб верифікації коректності побудованих ознак.

Суттєвим викликом предметної області є різноманіття доменів, у межах яких виникають атаки, і, відповідно, неоднорідність вимог до методів виявлення. Так, для транспортних мереж (in-vehicle network) та CAN-шини характерними є обмежені ресурси обчислень, вимоги до мінімальної затримки та потреба в легковагових моделях, що підтверджується сучасними дослідженнями [2, 7]. Для промислового інтернету та процесних систем ключовою є залежність сигналів у часі та високі ризики помилкових спрацювань, оскільки інциденти можуть призводити до зупинки технологічних процесів [15, 17]. Для SCADA-систем, зокрема у вітроенергетиці, проблематика ускладнюється специфікою протоколів і необхідністю урахування ризиків критичної інфраструктури [18]. Окремим напрямом є виявлення шкідливого зашифрованого TLS-трафіку, де застосовуються мультимодальні ознаки, що поєднують статистичні характеристики потоку та параметри сеансу шифрування [25]. Зазначене свідчить, що універсальна модель без адаптації до домену часто демонструє обмежену переносимість, а отже під час проєктування інтелектуальної системи доцільно передбачити гнучкість у виборі ознак і конфігурації детектора.

Науково-прикладний інтерес становлять підходи, запропоновані М. Комаром та співавторами, де акцент зроблено на високопродуктивності,

адаптивності та ефективності нейромережевих детекторів для інформаційно-телекомунікаційних систем. Зокрема, розглянуто паралельні глибинні нейромережі для детекції атак, що є важливим для обробки трафіку у великих обсягах [3]. Запропоновано високопродуктивну адаптивну систему виявлення кібератак, орієнтовану на практичне застосування [9], а також підходи до компресії параметрів трафіку з метою підвищення ефективності глибинних моделей [13]. Додатково представлено концепцію інтелектуальної кібероборони на основі поєднання штучних нейронних мереж та технік штучної імунної системи [11, 12]. У сукупності ці результати утворюють методичний фундамент для побудови системи моніторингу, що здатна масштабуватися, адаптуватися та зберігати прийнятну точність за умов варіативності мережевих сценаріїв.

Отже, предметна область дослідження характеризується високою динамікою загроз, великими потоками неоднорідних даних та доменною специфікою атак. Це обґрунтовує доцільність застосування методів глибокого навчання, а також необхідність проєктування архітектури, у якій процеси збору, підготовки та аналізу трафіку узгоджені з вимогами до продуктивності та практичної експлуатації системи моніторингу.

1.2 Аналіз існуючих підходів та методів

Розвиток інтелектуальних систем виявлення вторгнень у мережеву безпеку відбувається в напрямі переходу від правил і сигнатур до моделей, здатних узагальнювати закономірності трафіку та розпізнавати складні або модифіковані атаки. Центральне місце в сучасних дослідженнях займають методи глибокого навчання, оскільки вони забезпечують автоматизоване формування внутрішніх представлень ознак і демонструють високу ефективність у задачах класифікації та детекції аномалій за умови достатнього обсягу навчальних даних. Разом із тим, вибір конкретної архітектури визначається природою трафіку (пакетний чи потоковий), вимогами до затримки, доступністю ресурсів та доменною специфікою мережевого середовища.

Рекурентні нейронні мережі та моделі часових залежностей. Оскільки мережевий трафік є послідовністю подій, доцільним є використання рекурентних архітектур, здатних моделювати часові залежності. Зокрема, BiLSTM застосовується для двонапрямного врахування контексту у часових рядах, що є корисним при аналізі потокових ознак і послідовностей параметрів з'єднань. Практичну реалізацію такого підходу наведено у роботі, де описано проєктування BiLSTM-алгоритму для задачі мережевого IDS та продемонстровано його придатність до класифікації атак [24]. Спорідненим напрямом є використання BiGRU як менш ресурсомісткої альтернативи BiLSTM, що зберігає здатність відтворювати часові залежності. Проте у випадку складних сценаріїв атак, де важливі локальні патерни та взаємодія ознак, рекурентні моделі без додаткових механізмів можуть демонструвати обмежену селективність до ключових фрагментів сигналу.

Механізми уваги та гібридні архітектури. Посилення рекурентних моделей часто реалізується через інтеграцію механізмів уваги (attention), які дозволяють моделі зосереджуватися на найбільш інформативних частинах вхідної послідовності. У роботі, присвяченій інтеграції MultiHead Attention і BiGRU, запропоновано модель, орієнтовану на підвищення якості детекції за рахунок паралельного "перегляду" ознак із різних проєкцій і більш виразного кодування контексту [5]. Такий підхід концептуально узгоджується з вимогами до "глибокого аналізу трафіку", оскільки увага дозволяє компенсувати втрати інформативності при агрегації пакетів у потоки. Водночас у прикладних системах необхідно враховувати обчислювальну складність багатоголової уваги та потенційну чутливість до якості попередньої обробки даних (нормалізації, усунення пропусків, узгодження масштабів ознак).

Підходи на основі глибоких мереж довіри та комбінованих проєкцій. Порівняно з рекурентними та attention-орієнтованими моделями, інший клас методів формує ознаки через багатошарові нелінійні перетворення без явного урахування послідовності, зосереджуючись на структурі вхідного простору. У роботі запропоновано модель, що поєднує deep belief network та local preserving projection для підсилення роздільності класів у зниженому просторі ознак [23].

Такі підходи є корисними, коли трафік представлено табличними ознаками без вираженого порядку, однак вони можуть вимагати ретельного налаштування для уникнення перенавчання і зниження стійкості до зсуву розподілів у реальних мережах.

Легковагові моделі для спеціалізованих доменів: транспортні мережі та CAN. В окремих середовищах (зокрема в автомобільних мережах) домінують суворі обмеження щодо затримки й ресурсів, а також специфічні протоколи та шаблони передачі. Для таких умов запропоновано легковагові підходи, що мінімізують обчислювальні витрати при збереженні прийнятної точності. Зокрема, наведено метод виявлення вторгнень у in-vehicle мережі на основі глибокого навчання, орієнтований на зменшення складності моделі [7]. Інший напрям демонструє застосування MobileViT як легковагової архітектури для детекції вторгнень у CAN-шині, що відображає тенденцію перенесення компактних моделей у кібербезпекові задачі [2]. Перевагою таких підходів є практична придатність для “вбудованих” сценаріїв; обмеженням – можливе зниження узагальнювальної здатності при переході на інші типи атак або інші мережеві середовища.

Промисловий інтернет, процесні системи та SCADA. Сегмент промислових мереж і технологічних процесів висуває підвищені вимоги до надійності та мінімізації помилкових спрацювань. У роботі запропоновано метод для Industrial Internet на основі CapsNet і SRU, що орієнтований на поєднання здатності капсульних мереж фіксувати ієрархічні залежності з ефективним моделюванням послідовностей [15]. Для задач процесної індустрії розглянуто підхід, який поєднує виявлення вторгнень з локалізацією атаки за результатами аналізу процесних даних, що є важливим для оперативного реагування [17]. У випадку SCADA у вітроенергетиці досліджено ризики та застосування методів машинного навчання для детекції вторгнень з урахуванням специфіки таких систем [18]. Загалом, для промислових середовищ характерним є поєднання мережевих та технологічних параметрів, що стимулює розвиток гібридних схем ознак і потребує доменної адаптації моделей.

Детекція зашифрованого трафіку та мультимодальні ознаки. Поширення шифрування трафіку ускладнює використання традиційного контент-аналізу пакетів і зумовлює зростання ролі статистичних, поведінкових і протокольних параметрів сеансу. У роботі запропоновано метод виявлення шкідливого TLS-трафіку на основі інтеграції мультимодальних ознак, що відображає практичний тренд – перенос детекції в площину метаданих та поведінкових характеристик [25]. Перевагою підходу є застосовність у сучасних мережах, де payload недоступний; обмеженням – можливий вплив змін у реалізаціях протоколів і профілях легітимного трафіку, що може спричиняти зростання хибнопозитивних спрацювань.

Методи підвищення ефективності: адаптивність, компресія параметрів, паралельність. Окрему групу підходів становлять методи, спрямовані на масштабування IDS та підвищення швидкодії без критичної втрати якості. У працях М. Комара та співавторів розглянуто високопродуктивну адаптивну систему детекції кібератак, що орієнтована на експлуатаційні вимоги та змінність трафіку [9]. Запропоновано компресію параметрів мережевого трафіку як засіб зменшення розмірності вхідних даних, що є важливим для глибинних моделей в умовах великих потоків подій [13]. Також представлено паралельну глибинну нейромережу для виявлення атак в інформаційно-телекомунікаційних системах, що підкреслює актуальність обчислювального прискорення для задач моніторингу [3]. У контексті системної архітектури ці результати дозволяють обґрунтувати доцільність розподіленої або багатопотокової обробки, а також застосування зменшення розмірності як етапу конвеєра підготовки ознак.

Альтернативні моделі та задачі підсилення даних. Поряд із глибокими мережами застосовуються і відносно простіші підходи, зокрема методи на основі Extreme Learning Machine. У роботі наведено варіант несупервізованої ELM для задачі мережевого IDS, що може бути корисним як базова лінія порівняння або як компонент ансамблевих схем. Також досліджується посилення якості детекції через підходи до “data enhancement” та моделі для промислових вторгнень, що вказує на важливість коректної роботи з дисбалансом і рідкісними класами атак [27]. Практичний висновок полягає в тому, що продуктивність IDS значною

мірою визначається не лише вибором моделі, а й якістю підготовки даних, репрезентативністю вибірок та стратегіями боротьби з дисбалансом.

Для узагальнення розглянутих підходів та порівняння їх придатності до задачі моніторингу мережевої безпеки сформовано таблицю А.1 (додаток А), у якій зіставлено архітектури, типи ознак, області застосування та ресурсні вимоги. У таблиці також відображено ключові переваги й обмеження підходів, що є підставою для обґрунтування вибору моделі в наступних розділах.

Таким чином, аналіз джерел дозволяє виділити кілька стійких тенденцій:

1) зміщення акценту до моделей, що поєднують часові залежності та механізми селекції ознак (BiLSTM/BiGRU + attention) [5, 24];

2) розвиток доменно-орієнтованих легковагових рішень для транспортних і промислових мереж [2, 7, 15, 18];

3) перехід до детекції у шифрованому середовищі через мультимодальні параметри [25];

4) прагнення до високопродуктивної обробки за рахунок компресії ознак, адаптивності та паралельних обчислень [3, 9, 13].

З огляду на тему дослідження, обґрунтованим є вибір архітектури, що підтримує потокове представлення трафіку, забезпечує моделювання часових залежностей і може бути реалізована у вигляді модульної системи моніторингу з можливістю подальшого розширення та адаптації до різних доменів.

1.3 Постановка задачі дослідження

Побудова інтелектуальної системи моніторингу мережевої безпеки на основі глибокого аналізу трафіку передбачає формалізацію задачі виявлення вторгнень як задачі класифікації (або детекції аномалій) за сукупністю параметрів мережевих з’єднань. У практичних умовах дані надходять у вигляді потоку подій, тому рішення має забезпечувати обробку як у режимі offline (навчання/валідація на еталонних наборах), так і в режимі online (оперативний моніторинг і фіксація інцидентів). Для відтворюваного оцінювання результатів обрано відкриті набори даних, що широко застосовуються у дослідженнях IDS:

CIC-IDS2017 та UNSW-NB15 [1, 20]. Крім того, під час формування та перевірки ознак трафіку доцільним є використання інструментарію аналізу пакетів, зокрема Wireshark [22].

Розглянемо формалізацію задачі. Нехай мережевий трафік представлено як послідовність записів $X = \{x_i\}_{i=1}^{N}$, де кожен запис x_i відповідає потоку або з'єднанню та описується вектором ознак $x_i \in \mathbb{R}^d$. Вектор ознак може включати статистичні, часові та протокольні характеристики (наприклад, кількість пакетів/байтів, тривалість з'єднання, напрямки передачі, прапорці TCP, інтервали між пакетами тощо). Множина класів позначається як $Y = \{0,1, \dots, K\}$, де 0 – нормальний трафік, а $1..K$ – типи атак (за наявності багатокласової розмітки) або узагальнений клас "атака" (у випадку бінарної постановки).

Тоді задача виявлення вторгнень формулюється як побудова функції класифікації [37]

$$f: \mathbb{R}^d \to Y, \tag{1.1}$$

яка мінімізує помилку класифікації на тестовій вибірці та є стійкою до змін профілю трафіку в часі. Практично це означає, що для кожного нового запису x_t, отриманого в процесі моніторингу, система має обчислити прогноз $\hat{y}_t = f(x_t)$ та рівень впевненості (або ймовірність), після чого ухвалити рішення про наявність інциденту.

Якщо застосовується модель на основі послідовностей (наприклад, BiLSTM/BiGRU), тоді вхід може бути організований як послідовність векторів ознак [37]:

$$s_j = \{x_{j,1}, x_{j,2}, \dots, x_{j,T}\}, \tag{1.2}$$

де T – довжина вікна спостережень або кількість кроків у часовій агрегації.

Такий підхід узгоджується з моделями, що враховують часові залежності та здатні виділяти інформативні фрагменти через attention [5, 24, 26]. Водночас для системи моніторингу важливо, щоб обрана схема формування

послідовностей була однозначною, відтворюваною та придатною до роботи в online-режимі.

Вихідні дані системи мають включати:

– клас інциденту (атака/норма або тип атаки);
– оцінку впевненості/ймовірність;
– часову мітку події та ключові атрибути потоку;
– запис у журналі подій (лог) для подальшого аналізу.

На якість і практичну придатність інтелектуальної системи моніторингу впливають такі фактори:

1. Продуктивність і масштабованість. У реальних мережах трафік характеризується високою інтенсивністю, тому система має забезпечувати обробку потоків без критичних затримок. Перспективними є підходи, що використовують паралельні глибинні моделі та оптимізовані обчислення [3], а також високопродуктивні адаптивні схеми детекції [9].

2. Зменшення розмірності та інформативність ознак. Високорозмірні ознакові простори ускладнюють навчання та інференс, підвищують ризик перенавчання й збільшують потреби в обчислювальних ресурсах. Доцільним є застосування компресії параметрів трафіку або методів зменшення розмірності як елементу конвеєра підготовки даних [13, 30].

3. Адаптивність і стійкість до дрейфу. Профілі легітимного трафіку змінюються (оновлення сервісів, сезонність, зміни поведінки користувачів), що може погіршувати точність статичних моделей. Тому обґрунтованим є врахування адаптивних механізмів або періодичного донавчання, що підвищує стійкість системи у тривалій експлуатації [9, 21].

4. Доменна специфіка. Застосування IDS у різних середовищах висуває відмінні вимоги: легковаговість у транспортних мережах [2, 7], робота з технологічними параметрами у промислових мережах [15, 17], ризик-орієнтований контекст SCADA [18]. Отже, архітектура системи має бути модульною, щоб допускати заміну або налаштування детектора та набору ознак.

5. Аналіз шифрованого трафіку. У багатьох сегментах мереж payload недоступний, що зумовлює необхідність використання статистичних і

протокольних характеристик сеансів. Це особливо актуально для задач детекції TLS-трафіку із використанням мультимодальних ознак [25].

6. Юридичний та організаційний контекст. Питання кіберзахисту регулюється нормативно, тому результати моніторингу мають бути відтворюваними та придатними для документування інцидентів [33]. Практичним доповненням може бути підтримка фіксації подій і доказової бази, що узгоджується з сучасними підходами IDS/IPS із реєстрацією подій [21].

Отже, мета роботи полягає у розробленні інтелектуальної системи моніторингу безпеки комп'ютерної мережі на основі глибокого аналізу трафіку, що забезпечує виявлення вторгнень шляхом навчання та застосування моделей глибокого навчання, а також підтримує збір, підготовку та візуалізацію результатів у процесі моніторингу.

Для досягнення поставленої мети необхідно розв'язати такі завдання:

– проаналізувати предметну область мережевої безпеки та визначити вимоги до інтелектуальної системи моніторингу вторгнень;

– виконати огляд і порівняльний аналіз сучасних методів виявлення вторгнень на основі глибокого навчання та обґрунтувати вибір підходу;

– визначити джерела даних і сформувати набір інформативних ознак трафіку; організувати підготовку даних для навчання та тестування на еталонних датасетах;

– спроєктувати архітектуру системи, що включає модулі збору трафіку, попередньої обробки, нейромережевого класифікатора, прийняття рішення та представлення результатів;

– реалізувати прототип системи та експериментально оцінити його ефективність з використанням стандартних метрик і порівняльного аналізу;

– виконати інтерпретацію результатів, визначити обмеження запропонованого рішення та сформулювати практичні рекомендації щодо застосування й подальшого вдосконалення.

2 ПРОЄКТУВАННЯ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ МОНІТОРИНГУ

2.1 Вимоги та сценарії використання інтелектуальної системи моніторингу

Проєктування інтелектуальної системи моніторингу мережевої безпеки доцільно розпочинати з формалізації вимог та типових сценаріїв експлуатації. У межах даної роботи система розглядається як програмно-алгоритмічний комплекс, що забезпечує:

1) збір і попередню обробку мережевого трафіку;

2) формування ознак та, за потреби, зменшення розмірності/компресію параметрів;

3) інференс нейромережевої моделі детекції;

4) прийняття рішення щодо інциденту з подальшим журналюванням, оповіщенням і візуалізацією результатів.

Така модульна структура відповідає підходам до високопродуктивних і адаптивних систем виявлення кібератак, зокрема у роботах, де наголошено на необхідності паралельної обробки та адаптації детектора до змін трафіку [3, 9, 13], а також на принципах інтелектуальної кібероборони [11, 12].

На рисунку 2.1 представлено узагальнену схему запропонованої системи, яка відображає основний конвеєр обробки даних (збір → підготовка → ознаки → модель → рішення) та контур зворотного зв’язку (логування → аналітика → адаптація/донавчання), що забезпечує підвищення стійкості й актуальності моделі у часі.

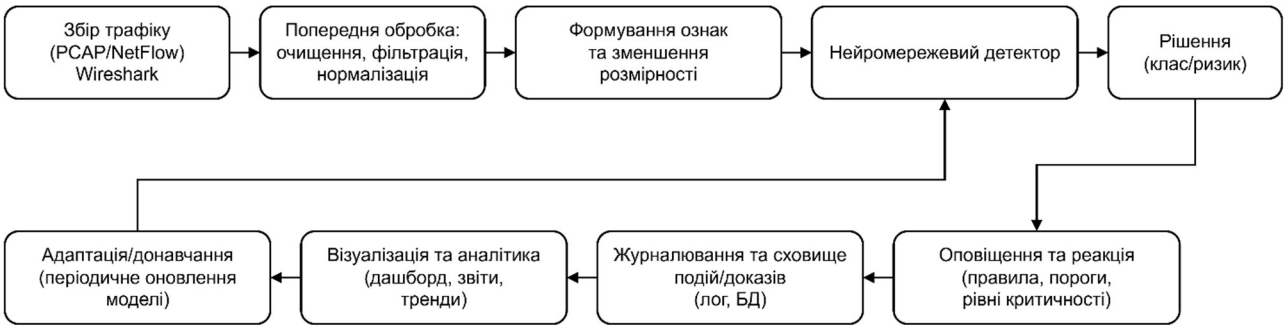

Рисунок 2.1 – Узагальнена схема інтелектуальної системи моніторингу мережевої безпеки

Компоненти збору й первинної верифікації трафіку орієнтовано на застосування типових інструментів аналізу пакетів і потоків, зокрема Wireshark [22], тоді як навчання та тестування моделі передбачено виконувати на еталонних наборах даних CIC-IDS2017 та UNSW-NB15 [1, 20].

Сценарії експлуатації системи визначають вимоги до її модулів та способу інтеграції в мережеву інфраструктуру. Доцільно виокремити два базові режими: offline та online.

1. Offline-сценарій (експериментальний режим). У цьому режимі система використовується для розроблення й перевірки детектора атак на основі готових наборів даних. Дані імпортуються з CIC-IDS2017 і UNSW-NB15, після чого виконується попередня обробка, формування ознак, навчання моделі та оцінювання якості за стандартними метриками (accuracy, precision, recall, F1-score) [1, 20]. Практична цінність offline-режиму полягає у відтворюваності експериментів і можливості порівняння з відомими підходами, включаючи моделі на основі BiLSTM/BiGRU та механізмів уваги [5, 24, 26], а також альтернативні рішення (наприклад, ELM або DBN-подібні схеми) [6, 23]. Результати цього етапу формують підґрунтя для вибору параметрів моделі й конфігурації системи, придатної для практичного моніторингу.

2. Online-сценарій (режим моніторингу). У режимі моніторингу система під’єднується до джерела трафіку (мережевий інтерфейс, дзеркальний порт комутатора, файл PCAP, NetFlow-експортер) і здійснює безперервну обробку даних. На цьому етапі ключовими є вимоги до затримки обробки та надійності: інференс моделі має відбуватися з мінімальною затримкою, а результати повинні фіксуватися у журналі подій з можливістю подальшого аналізу [21]. Застосування інструментів захоплення/аналізу трафіку (зокрема Wireshark) підвищує контрольованість процесу збору та забезпечує можливість ручної верифікації підозрілих потоків [22].

Практично важливо, що online-режим може реалізовуватися у різних доменах. Для корпоративних мереж пріоритетними є масштабованість і стійкість до дрейфу нормальної поведінки; для транспортних мереж – легковаговість моделі та гарантована швидкодія [2, 7]; для промислових сегментів – висока

надійність, мінімізація хибнопозитивних спрацювань і врахування ризиків критичної інфраструктури [15, 17, 18]. Окремий випадок – детекція шкідливого TLS-трафіку, де аналіз здійснюється переважно за метаданими та мультимодальними ознаками [25]. Це зумовлює потребу в модульності рішення: залежно від середовища можуть змінюватися ознаки, детектор і політики прийняття рішень.

До основних функціональних вимог системи віднесено:

– збір трафіку у формах PCAP/NetFlow або агрегованих потоків із підтримкою фільтрації й базової перевірки коректності даних [22];

– попередня обробка (очищення, нормалізація, кодування) та формування набору ознак, сумісного з еталонними датасетами для навчання/тестування [1, 20];

– виявлення вторгнень шляхом застосування нейромережевої моделі; підтримка бінарної та/або багатокласової класифікації залежно від датасету і цілей моніторингу [5, 24];

– прийняття рішення: визначення класу/рівня ризику, застосування порогів, категоризація подій за критичністю, формування оповіщень [21];

– журналювання: збереження подій, метаданих, результатів моделі та контексту для подальшого аналізу й аудиту [21];

– візуалізація та аналітика: подання статистики інцидентів, трендів, частоти атак, розподілів за класами, а також засобів пошуку/фільтрації подій.

Нефункціональні вимоги відображають придатність системи до довготривалої експлуатації та інтеграції:

– продуктивність і масштабованість: забезпечення обробки високих потоків даних потребує оптимізації обчислень і, за можливості, паралельної реалізації детектора [3];

– адаптивність: доцільним є передбачення механізмів оновлення моделі та/або періодичного донавчання з урахуванням дрейфу трафіку та появи нових атак [9, 21];

– стійкість до зменшення розмірності: система має зберігати прийнятну точність при застосуванні компресії параметрів або інших процедур зменшення розмірності, що важливо для швидкодії [13, 30];

– надійність і відтворюваність: процедури підготовки даних, навчання та оцінювання повинні бути формалізованими, щоб результати можна було повторити на тих самих датасетах [1, 20];

– відповідність нормативному контексту: результати моніторингу мають підтримувати документування інцидентів та організаційні процеси реагування, що є важливим у контексті вимог кіберзахисту [33].

Отже, сформульовані сценарії та вимоги визначають технічні рішення наступних підрозділів: вибір даних і конвеєра підготовки, обґрунтування архітектури детектора та проєктування програмної архітектури реалізації. На подальших етапах увагу доцільно зосередити на забезпеченні балансу між точністю детекції, швидкодією та можливістю адаптації системи до різних мережевих доменів.

2.2 Дані та попередня обробка трафіку

Ефективність інтелектуальної системи моніторингу мережевої безпеки значною мірою визначається якістю даних, коректністю їх підготовки та відтворюваністю конвеєра попередньої обробки. У задачах виявлення вторгнень типовою є ситуація, коли різні класи атак відрізняються не окремими ознаками, а їхніми комбінаціями та часовою динамікою. За таких умов неконсистентність ознак, пропуски, шум і дисбаланс класів можуть спричиняти суттєве падіння якості моделі навіть при використанні сучасних архітектур глибокого навчання [5, 24]. Тому в межах даного підрозділу формалізовано вибір наборів даних, окреслено процедури очищення та перетворення ознак, а також визначено підходи до зменшення розмірності/компресії параметрів трафіку, що узгоджується з ідеями підвищення продуктивності IDS [13, 30].

Для експериментального дослідження доцільно використовувати відкриті еталонні датасети, що є де-факто стандартом для порівняння результатів у

задачах IDS. У роботі обрано два джерела даних: CIC-IDS2017 та UNSW-NB15 (таблиця 2.1).

Таблиця 2.1 – Характеристика датасетів, використаних для навчання та оцінювання системи

Датасет	Тип представлення	Призначення в роботі	Коментар щодо використання
CIC-IDS2017 [1]	Переважно потокові/статистичні ознаки мережевих з'єднань	Базова експериментальна перевірка моделі, налаштування гіперпараметрів	Забезпечує зіставність з результатами інших досліджень; придатний для багатокласової/бінарної постановки
UNSW-NB15 [20]	Ознаки мережевого трафіку (з'єднання/потоки), категорії атак	Перевірка переносимості та стійкості моделі на альтернативному наборі	Дозволяє оцінити вплив іншого розподілу даних і профілю атак

Перший набір широко застосовується як база для оцінювання алгоритмів виявлення вторгнень та містить різнорідні типи атак у структурованому представленні ознак потоків [1]. Другий набір є популярним у дослідженнях IDS як приклад більш сучасного трафіку з різними категоріями атак та відповідним набором атрибутів [20]. Використання двох датасетів дозволяє перевірити узагальнювальну здатність підходу та виявити залежність результатів від домену, що є важливим для подальшого розгортання системи в реальному середовищі.

У реальних умовах, окрім використання готових датасетів, система може працювати з трафіком, зібраним у локальній мережі, зокрема у форматі PCAP. Для первинної верифікації пакети й протокольні поля доцільно аналізувати за допомогою Wireshark [22]. Важливо підкреслити, що Wireshark у контексті даної роботи виконує допоміжну функцію: підтвердження коректності захоплення трафіку, перевірка фільтрів, аналіз окремих аномальних потоків та контроль формування ознак.

Попередня обробка даних для IDS має забезпечити узгодженість ознак і придатність даних до навчання нейромережевих моделей. Типовий конвеєр включає такі етапи (рисунок 2.2):

1. Очищення даних. На цьому кроці видаляються дублікати записів, обробляються некоректні значення (наприклад, нескінченності, від'ємні тривалості, неможливі лічильники), узгоджуються типи полів. Доцільним є також вилучення службових колонок, що не несуть інформаційного навантаження або можуть створювати витік мітки (label leakage).

2. Обробка пропусків та аномальних значень. Пропуски можуть виникати як через неповні потоки, так і через помилки агрегації. Залежно від типу ознаки застосовуються: заповнення нульовими значеннями (для лічильників), медіаною/середнім (для статистичних ознак), або вилучення записів (якщо пропуски масові та спотворюють дані). Для нейромережевих моделей важливо, щоб стратегія була стабільною й відтворюваною, оскільки інакше змінюється розподіл вхідного простору та порушується коректність порівняння експериментів [5].

3. Нормалізація/масштабування ознак. Більшість потокових характеристик (кількість байтів, пакетів, тривалість) мають різні масштаби. Нормалізація (наприклад, min-max або z-score) зменшує домінування «великих» ознак і стабілізує навчання. Це особливо актуально для рекурентних моделей та attention-компонентів, які чутливо реагують на розбалансованість вхідних значень [5, 24].

4. Кодування категоріальних полів. У мережевих даних можуть бути присутні поля на кшталт протоколу, стану з'єднання, сервісу. Для їх подання застосовується one-hot кодування або індексне кодування (з подальшими embedding-шарами у моделі), залежно від обраної архітектури детектора [5, 26].

5. Формування послідовностей (за потреби). Якщо модель використовує часовий контекст (BiLSTM/BiGRU), то записи групуються у вікна за часом або за ключем потоку/хоста. Важливо, щоб правило формування послідовностей було однозначним: наприклад, вікно фіксованої довжини T з ковзанням або

групування по сесії. Це безпосередньо впливає на якість розпізнавання атак, що проявляються як серії подій, а не одиничні потоки [24].

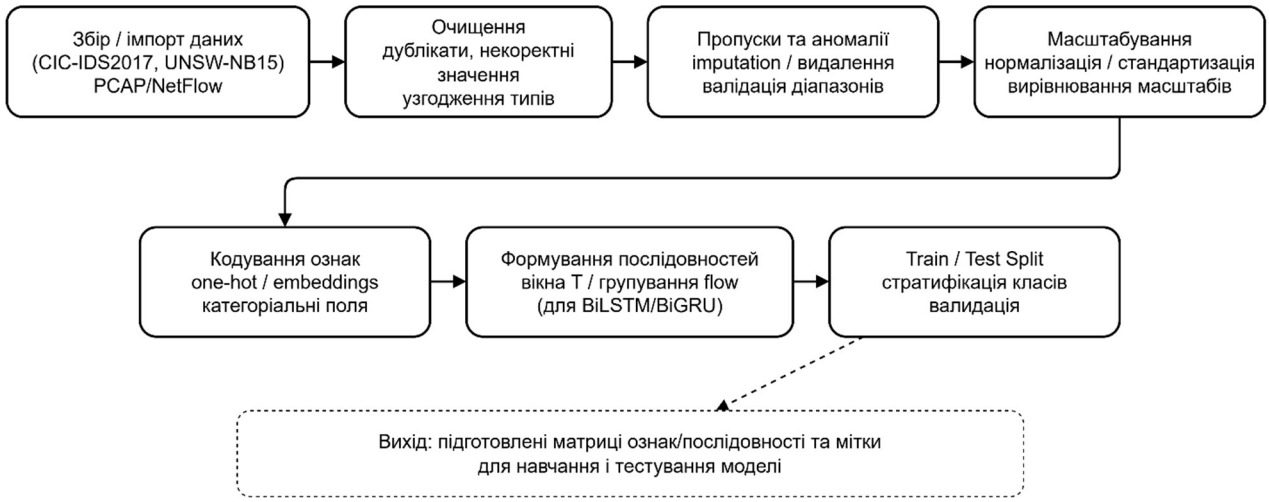

Рисунок 2.2 – Конвеєр підготовки даних для інтелектуальної системи моніторингу

У задачах IDS типово спостерігається дисбаланс класів: нормальний трафік суттєво переважає, а окремі типи атак представлені малими вибірками. Це призводить до «зсуву» моделі в бік домінантного класу та заниження recall для рідкісних атак. Тому доцільно застосовувати комплекс рішень:

– поділ на навчальну та тестову вибірки для збереження пропорцій класів у розбиттях;

– використання ваг класів у функції втрат, що підсилює внесок міноритарних класів;

– підсилення даних або розумні стратегії ресемплінгу, якщо це відповідає природі ознак і не спотворює реальні залежності [27].

У підсумку доцільно вважати основними метриками не лише accuracy, а й precision/recall/F1-score, зокрема у розрізі класів атак, оскільки саме вони відображають практичну цінність системи детекції.

Підвищення продуктивності моніторингу нерідко вимагає оптимізації розмірності вхідних даних. Зменшення розмірності є доцільним у випадках, коли:

– кількість ознак значна;

– ознаки корельовані;

– інференс повинен відбуватися у реальному часі.

В межах даної роботи зменшення розмірності доцільно реалізовувати як опційний модуль конвеєра: модель навчається у двох режимах (без компресії та з компресією/зменшенням розмірності), після чого порівнюються метрики якості та часові витрати на інференс. Такий підхід дозволяє обґрунтовано оцінити компроміс між точністю та продуктивністю з урахуванням вимог до системи моніторингу [3, 13].

Для забезпечення відтворюваності експериментів визначено ключові групи ознак та правила їх обробки. У таблицях Б.1-Б.2 (додаток Б) подано перелік базових полів датасетів CIC-IDS2017 та UNSW-NB15, їх функціональне призначення в задачі IDS і рекомендовані процедури попередньої обробки (очищення, кодування, масштабування, робота з пропусками та вилучення ідентифікаторів).

Таким чином, підготовка даних для інтелектуальної IDS у межах роботи ґрунтується на застосуванні еталонних датасетів і відтворюваного конвеєра попередньої обробки, який включає очищення, нормалізацію, кодування та, за потреби, формування послідовностей для рекурентних моделей. Для забезпечення практичної придатності рішення передбачено врахування дисбалансу класів і оцінювання метрик у розрізі атак, а також опційне застосування компресії/зменшення розмірності як засобу підвищення швидкодії. Використання Wireshark дозволяє підсилити контроль коректності збору та інтерпретацію трафіку на етапах розроблення й тестування.

2.3 Проєктування нейромережевої моделі виявлення вторгнень

Проєктування нейромережевої моделі виявлення вторгнень у межах інтелектуальної системи моніторингу має враховувати дві ключові обставини:

– мережевий трафік характеризується часовою залежністю та контекстом, який не завжди відображається в одиничному записі потоку;

– у практичних умовах важливо забезпечити прийнятний компроміс між точністю детекції та обчислювальними витратами при інференсі.

З огляду на це доцільним є вибір архітектури, здатної моделювати послідовності та акцентувати увагу на найінформативніших фрагментах трафіку.

В якості базового варіант доцільно застосувати гібридну архітектуру BiGRU + MultiHead Attention, оскільки вона поєднує дві важливі властивості (рисунок 2.3) [37]:

– моделювання часових залежностей за рахунок двонапрямної рекурентної мережі (BiGRU/або BiLSTM);

– селективність через механізм уваги, який підсилює внесок найінформативніших кроків послідовності та зменшує вплив шумових фрагментів [5, 24, 26].

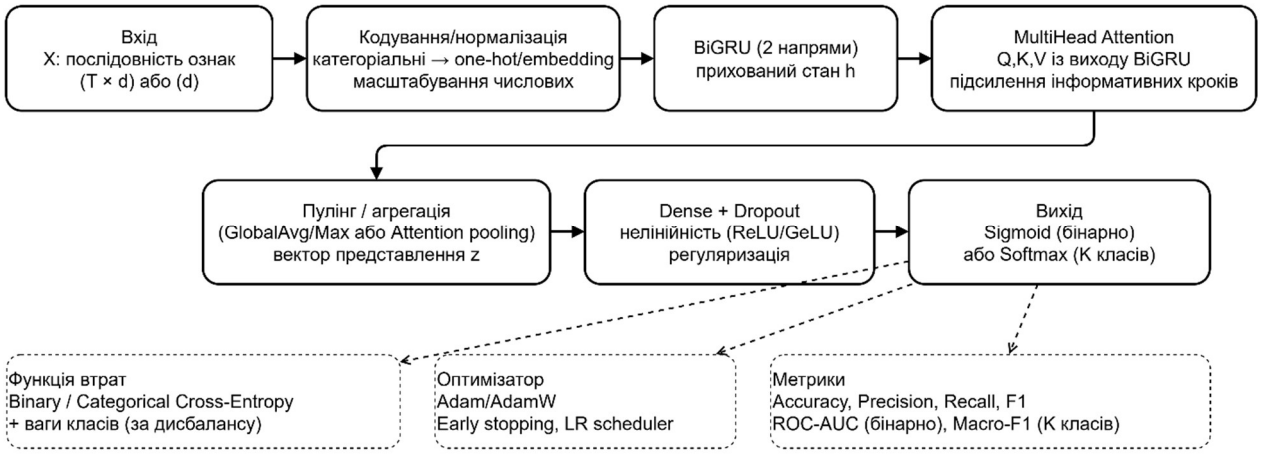

Рисунок 2.3 – Структурна схема нейромережевої моделі [37]

BiGRU у порівнянні з BiLSTM часто розглядається як більш “економний” варіант з меншою кількістю параметрів при збереженні здатності відстежувати контекст, що є корисним для режиму моніторингу. Використання MultiHead Attention дозволяє моделі одночасно аналізувати кілька “проєкцій” залежностей у послідовності, що підвищує чутливість до різних патернів атак (наприклад, короткі імпульсні прояви та довші поведінкові тренди) [5, 37].

У загальному вигляді модель отримує на вхід або:

– табличний вектор ознак потоку $x \in \mathbb{R}^d$ (спрощена постановка), або

– послідовність $x \in \mathbb{R}^{T \times d}$, сформовану з ковзного вікна/групування потоків (послідовнісна постановка), що є пріоритетною для BiGRU/BiLSTM [24].

Вихід моделі реалізується через Sigmoid (для бінарної детекції «норма/атака») або Softmax (для багатокласової класифікації типів атак), що узгоджується з типовими постановками задач IDS [5, 26].

Для обґрунтування вибору базової архітектури доцільно розглянути альтернативи, які застосовуються в IDS у різних доменах [37]:

1. DBN-подібні підходи (Deep Belief Network). Моделі на основі DBN орієнтовані переважно на табличні ознаки та можуть демонструвати конкурентні результати на класичних наборах даних. У працях, де застосовано DBN у поєднанні з проєкційними методами, підкреслюється потенціал такого підходу для виділення прихованих структур у даних [23]. Водночас DBN зазвичай слабше відображає часовий контекст, якщо послідовність не моделюється явно. Додатковим аргументом на користь часових моделей є результати, де глибока довіра застосовується саме для детекції атак, але потребує ретельного підбору ознак та параметрів [34].

2. Легковагові моделі для доменів із обмеженими ресурсами (MobileViT, CapsNet+SRU). У транспортних мережах (CAN/in-vehicle) важливими є компактність і швидкодія, що обґрунтовує використання MobileViT або інших lightweight-архітектур [2]. Для промислового інтернету (IIoT) запропоновано поєднання CapsNet та SRU, яке орієнтується на виявлення структурних залежностей і роботу з послідовностями [15]. Разом з тим такі архітектури часто є домен-специфічними: модель, оптимізована під CAN або IIoT, може втрачати узагальнювальну здатність при перенесенні в класичний NIDS без адаптації ознак та навчальної вибірки.

3. ELM (Extreme Learning Machine), зокрема несупервізовані варіанти. ELM-підхід привабливий швидким навчанням і відносною простотою, що робить його корисним як базову лінію порівняння [6]. Однак для складних багатокласових сценаріїв і задач із часовими залежностями очікувано кращі результати забезпечують гібридні глибинні моделі, які можуть відображати контекст та нелінійні взаємозв’язки між ознаками [5, 24].

Узагальнюючи, BiGRU + MultiHead Attention є збалансованим рішенням для даної теми: воно враховує часову природу трафіку (на відміну від чисто

табличних моделей) і не є жорстко прив’язаним до вузького домену (на відміну від деяких lightweight-рішень), водночас забезпечуючи кращий потенціал точності, ніж спрощені ELM-підходи.

Розглянемо визначення гіперпараметрів, функції втрат, метрик оцінювання.

Функція втрат:

– для бінарної постановки застосовується Binary Cross-Entropy;

– для багатокласової – Categorical Cross-Entropy.

З огляду на типову незбалансованість класів у IDS доцільно передбачити ваги класів у функції втрат, щоб зменшити деградацію recall для міноритарних атак.

В якості стандартного оптимізатора доцільно застосувати Adam/AdamW із планувальником швидкості навчання (LR scheduler) та механізмом ранньої зупинки для обмеження перенавчання.

Метрики оцінювання. Поряд із accuracy доцільно обов’язково аналізувати precision, recall, F1-score, причому у разі багатокласової задачі – macro-F1 (як менш чутливу до дисбалансу). Для бінарної детекції корисним є ROC-AUC, що характеризує якість ранжування рішень при різних порогах.

Обрані гіперпараметри моделі подано в таблиці 2.2.

Таблиця 2.2 – Обрані гіперпараметри моделі та їх обґрунтування

Гіперпараметр	Позначення / приклад значення	Обґрунтування вибору
1	2	3
Довжина вікна послідовності	T=10…50	Достатня для відображення коротких серій подій і без надмірного ускладнення інференсу [24]
Розмір прихованого стану BiGRU	h=64…256	Компроміс між виразністю та ресурсами; збільшення h підвищує точність, але збільшує час інференсу
Кількість шарів BiGRU	1–2	1 шар часто достатній для потокових послідовностей; 2 шар – за складніших патернів, але з ризиком перенавчання

Продовження таблиці 2.2.

1	2	3
Кількість голів уваги	heads = 2–8	Дозволяє моделі “бачити” різні залежності в даних одночасно [5]
Dropout	0.2–0.5	Регуляризація для зменшення перенавчання в задачі IDS з шумними ознаками
Повнозв’язний шар (Dense)	64–256 нейронів	Агрегація представлень та нелінійне відокремлення класів
Швидкість навчання	lr = 1e-4…1e-3	Типовий робочий діапазон для Adam; уточнюється за валідацією
Розмір пакета	batch = 64…256	Баланс стабільності градієнта і швидкодії навчання
Ваги класів	w_c	Компенсація дисбалансу, підвищення recall для рідкісних атак
Поріг рішення (бінарно)	τ=0.5 (з підбором)	Підбір τ за ROC/PR-кривими для зменшення FP/FN залежно від політики безпеки

Також розглянемо механізми адаптивності/оновлення моделі та підвищення продуктивності. У реальному моніторингу модель стикається з дрейфом трафіку (зміна сервісів, поведінки користувачів, появи нових атак). Тому доцільно передбачити контур оновлення, який не порушує стабільності системи:

1. Періодичне перенавчання (наприклад, щотижня/щомісяця) на накопичених верифікованих логах із фіксацією версій моделі та порівнянням метрик. Такий підхід узгоджується з концепціями високопродуктивних адаптивних систем детекції [9].

2. Online/інкрементне оновлення за умов наявності контрольованого механізму валідації, щоб уникнути “зсуву” моделі через шумові або неправильно розмічені дані; це відповідає підходам IDS/IPS із адаптивним навчанням [21].

3. Підвищення продуктивності на рівні моделі: зменшення розмірності ознак (за потреби), обмеження T, застосування більш компактних конфігурацій BiGRU, а також оптимізація інференсу (батчування, прискорені бібліотеки).

Практично важливо, щоб прискорення не призводило до різкого падіння recall критичних атак.

Отже, в даному підрозділі виконано проєктування нейромережевої моделі виявлення вторгнень для інтелектуальної системи моніторингу мережевої безпеки. В якості базової архітектури обґрунтовано використання гібридної моделі BiGRU + MultiHead Attention, яка поєднує здатність рекурентних мереж відображати часовий контекст трафіку та механізм уваги для виділення найбільш інформативних фрагментів послідовності, що є принципово важливим для задач IDS у реальних умовах.

3 РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

3.1 Архітектура програмної реалізації

Архітектура програмної реалізації інтелектуальної системи моніторингу мережевої безпеки має забезпечити безперервний збір трафіку, формування ознак, виконання інференсу нейромережевої моделі та фіксацію результатів у вигляді подій і доказової бази. Враховуючи практичні вимоги до IDS/IPS (відтворюваність, аудит, доказовість, інтеграція з зовнішніми засобами кіберзахисту), доцільно застосувати компонентний підхід із чітким розподілом відповідальностей між модулями [11, 21].

Архітектура програмної системи моніторингу мережевої безпеки включає наступні модулі (рисунок 3.1):

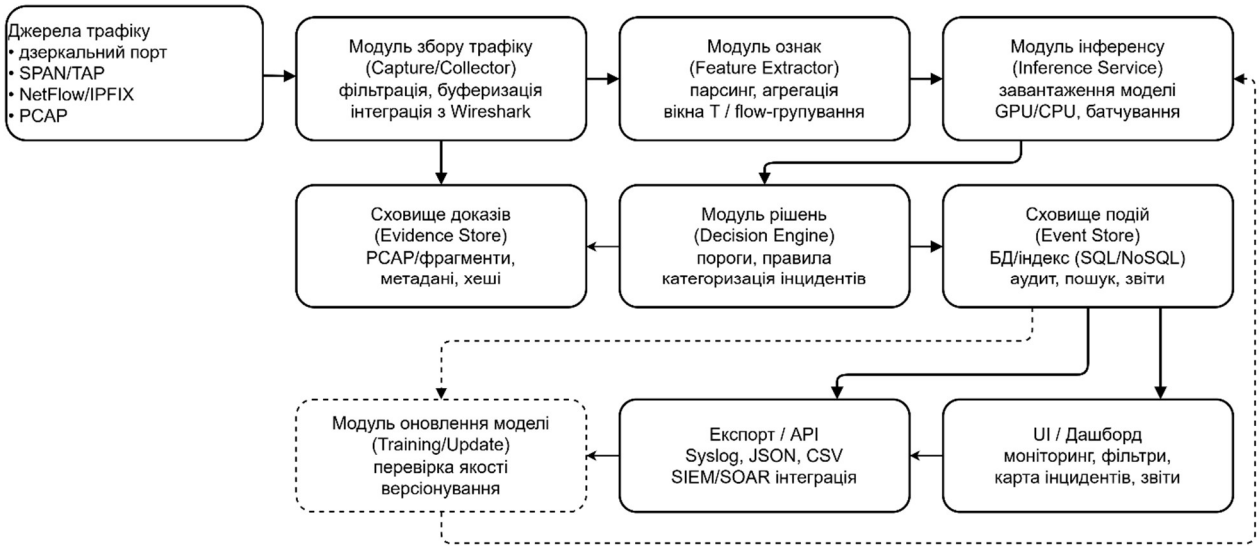

Рисунок 3.1 – Архітектура програмної системи моніторингу мережевої безпеки (компонентна діаграма)

1. Модуль збору трафіку (Capture/Collector). Забезпечує отримання даних з джерел (SPAN/TAP, NetFlow/IPFIX, PCAP), первинну фільтрацію та буферизацію. У режимі налагодження або перевірки коректності захоплення трафіку можливе використання Wireshark як інструмента верифікації на рівні пакетів [22]. Для безперервного моніторингу критичною є стійкість модуля до пікових навантажень і наявність контрольованих механізмів відкидання/дедуплікації.

2. Модуль ознак (Feature Extractor). Виконує парсинг пакетів/потоків, агрегацію статистик і формування вхідного представлення для моделі. Для послідовнісних архітектур (BiGRU/BiLSTM) додатково реалізується групування записів у вікна TTT за часом або за ключем потоку. На цьому рівні також реалізуються типові процедури нормалізації та кодування категоріальних полів відповідно до таблиць Б.1-Б.2.

3. Модуль інференсу (Inference Service). Здійснює завантаження навченої моделі, пакетування (batching) запитів та виконання прогнозу (CPU/GPU). Для забезпечення стабільності якості доцільно жорстко фіксувати версію моделі та конфігурацію препроцесингу (скейлери, словники категорій), оскільки розсинхронізація цих елементів призводить до некоректних результатів навіть при незмінній моделі.

4. Модуль прийняття рішень (Decision Engine). На основі виходу моделі формує подію безпеки: застосовує пороги, правила категоризації інцидентів, збагачує подію контекстом (ідентифікатори потоку, час, сервіс, вузол), визначає рівень критичності та подальші дії (оповіщення, ескалація, додатковий збір доказів).

5. Сховище подій (Event Store). Призначене для накопичення інцидентів, результатів детекції та службової телеметрії (час обробки, версія моделі, ознаки якості). Реалізаційно може базуватися на SQL/NoSQL БД та/або індексі для швидкого пошуку й фільтрації подій. Важливо підтримати незмінність ключових полів і контроль цілісності записів.

6. Сховище доказів (Evidence Store). Зберігає доказові артефакти: фрагменти PCAP, агреговані витяги, “знімки” ключових метаданих, контрольні хеші, а також прив’язку доказів до подій. Такий підхід узгоджується з практиками фіксації доказової бази для IDS/IPS із можливістю подальшої взаємодії з відповідальними підрозділами та правоохоронними органами [21].

7. UI/дашборд. Надає оператору засоби моніторингу (стрічка інцидентів), фільтрації, перегляду деталей події, побудови статистики, формування звітів, а також механізми відмітки “підтверджено/хибне спрацювання” для подальшої аналітики й адаптації моделі.

Опис модулів системи та їхніх функцій подано в таблиці 3.1.

Таблиця 3.1 – Опис модулів системи та їхніх функцій

Модуль	Основні функції	Вхідні дані	Вихідні дані/результат
Capture / Collector	Захоплення трафіку, фільтрація, буферизація, контроль якості	Пакети / потоки (PCAP, NetFlow/IPFIX)	Сировинні записи для обробки, посилання на PCAP-фрагменти
Feature Extractor	Парсинг, агрегація, формування ознак, вікна TTT	Трафік / потоки з Capture	Матриці ознак або послідовності $T \times d$
Inference Service	Завантаження моделі, прогноз, батчування	Ознаки / послідовності	Ймовірності класів / оцінка ризику
Decision Engine	Пороги, правила, категоризація, збагачення контекстом	Вихід моделі + метадані потоку	Подія безпеки (інцидент), рівень критичності, тригери
Event Store	Збереження подій, індексація, аудит	Інциденти, телеметрія	Історія інцидентів, аналітичні вибірки, звіти
Evidence Store	Зберігання доказів, хешування, зв'язок із подіями	PCAP / витяги / метадані	Доказова база (артефакти + контроль цілісності)
UI/Дашборд	Візуалізація, фільтри, аналіз, звіти	Дані з Event Store	Інтерфейс оператора, експорт звітів
Export/API	Інтеграція, обмін даними, сумісність	Події / статистика	Syslog/JSON/CSV, API-відповіді, інтеграція з SIEM/SOAR
Training / Update (опційно)	Оновлення моделі, версіонування, контроль якості	Верифіковані події / розмітка	Нова версія моделі, журнал змін

Логіка прийняття рішення має узгоджуватися з політикою безпеки та допустимим рівнем хибних спрацювань. Для бінарної детекції використовується поріг τ для значення Sigmoid-виходу:

– якщо $p(attack) \geq \tau$, подія класифікується як атака;

– якщо $p(attack) < \tau$, подія вважається нормальною.

Для багатокласового випадку рішення визначається за максимумом імовірності Softmax, а також може доповнюватися “порогом довіри” (наприклад, якщо найбільша імовірність нижча за τ_k, подія переводиться у категорію “невизначено” з підвищеним пріоритетом на ручну перевірку).

Категоризація інцидентів доцільна у вигляді рівнів критичності (наприклад, Low/Medium/High/Critical) на підставі:

– класу атаки (якщо доступно);

– величини ймовірності/ризику;

– повторюваності події (частота за інтервал);

– контексту активу (критичний вузол/сегмент);

– наслідків (ознаки ексфільтрації, відмови сервісу).

Такий підхід дозволяє зменшити навантаження на оператора та сфокусувати увагу на подіях із найбільшим потенційним впливом.

Збереження результатів детекції має забезпечувати відтворюваність і доказовість. Відповідно до практик IDS/IPS з адаптивним навчанням та фіксацією доказів, доцільно застосувати структуроване логування подій і розділення “події” та “доказу” [21].

Рекомендована структура запису інциденту (JSON-подібна):

– event_id, timestamp, sensor_id;

– мережевий контекст: src_ip, dst_ip, src_port, dst_port, proto, service/state (за наявності);

– модель: model_version, preprocess_version, score, predicted_class, threshold;

– категоризація: severity, confidence, rule_hits (якщо застосовано правила);

– посилання на докази: evidence_ref (PCAP-id, діапазон часу/байтів), hash (контроль цілісності).

Доказова база формується як:

– PCAP-фрагмент(и) або витяги з потоків, пов’язані з інцидентом;

– метадані (час, інтерфейси, фільтри захоплення, контрольні суми);

– криптографічний хеш артефактів для контролю незмінності.

Окремо доцільно вести журнал дій оператора (підтвердження/відхилення інциденту, коментарі), оскільки він корисний для подальшого аналізу, а також для керованого оновлення моделі.

Для практичного використання система повинна підтримувати інтеграцію з типовою екосистемою кіберзахисту та суміжними інструментами. До інтеграційних можливостей доцільно віднести:

– експорт подій у форматах JSON/CSV для аналітики та звітності;

– Syslog/CEF/LEEF (за потреби) для підключення до SIEM;

– REST API для отримання подій, статусів, статистики, а також для передачі “міток” оператора (підтверджено/хибне);

– сумісність з пайплайнами ML: версіонування моделей і конфігурацій препроцесингу, можливість “гарячого” оновлення моделі після валідації;

– модульність: можливість заміни детектора (наприклад, перехід на lightweight-модель для іншого домену) без зміни решти компонентів, що узгоджується з принципами побудови інтелектуальних систем кібербезпеки.

3.2 Реалізація програмного прототипу інтелектуальної системи моніторингу

Прототип інтелектуальної системи моніторингу мережевої безпеки реалізовано згідно з компонентною архітектурою, описаною в підрозділі 3.1. Кожен компонент оформлено як окремий модуль із чітко визначеними вхідними/вихідними даними, що спрощує тестування, заміну елементів і подальше масштабування. У цьому підрозділі наведено короткий опис реалізації та фрагменти коду; повні лістинги подано в додатку В.

Модуль Capture/Collector підтримує два сценарії:

– офлайн-аналіз PCAP (імпорт і попередня фільтрація пакетів);

– агрегація у flows (підготовка потокових записів для модулів ознак і інференсу).

Для перевірки коректності захоплення та інтерпретації пакетних даних застосовано Wireshark як інструмент верифікації (фільтри протоколів, аналіз полів, звірка часових міток) [22].

Фрагмент коду – імпорт PCAP та фільтрація:

```
def read_pcap(pcap_path: str, bpf: str | None = None):
    packets = sniff(offline=pcap_path, filter=bpf)  # bpf: "tcp or udp"
    return packets

def packet_filter(pkt) -> bool:
    return (IP in pkt) and (TCP in pkt or UDP in pkt)
```

Результатом роботи модуля є або потік пакетів (для аналізу), або первинні записи, достатні для побудови потокових ознак і подальшої фіксації доказів (PCAP-фрагменти/метадані).

Результат роботи модуля збору трафіку та агрегації пакетів у потоки представлено на рисунку 3.2.

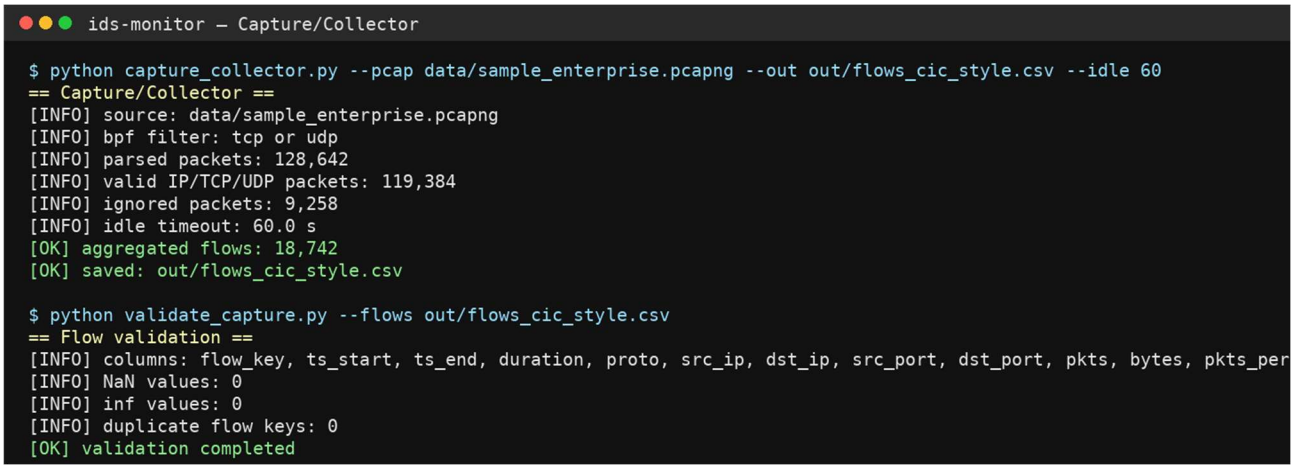

Рисунок 3.2 – Результат роботи модуля збору трафіку та агрегації пакетів у потоки

Модуль Feature Extractor виконує агрегацію ознак за потоком/вікном часу відповідно до підходу, описаного в 2.2 (таблиці Б.1-Б.2). Реалізовано:

– очищення (некоректні значення, дублікати);

– обробку пропусків/аномалій;

– масштабування числових та кодування категоріальних полів;

– формування послідовностей довжини T для рекурентної моделі (BiGRU/BiLSTM).

Попередня обробка мережевих даних і формування послідовностей для моделі представлена на рисунку 3.3.

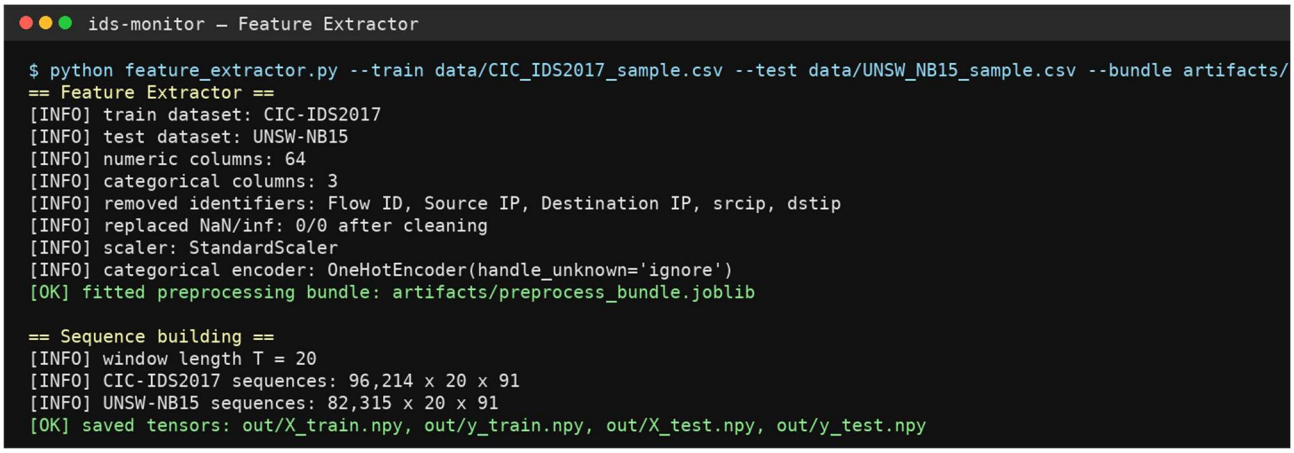

Рисунок 3.3 – Попередня обробка мережевих даних і формування послідовностей для моделі

Фрагмент коду – препроцесинг та формування послідовностей:

```
def preprocess(df, scaler, cat_encoder):
    df = df.replace([np.inf, -np.inf], np.nan)
    df = df.fillna(0.0)  # уточнення правил – у Додатку А
    X_num = scaler.transform(df[num_cols])
    X_cat = cat_encoder.transform(df[cat_cols])
    return np.hstack([X_num, X_cat])

def make_windows(X, T: int):
    return np.stack([X[i:i+T] for i in range(0, len(X) - T + 1)])
```

На виході формується матриця ознак (таблична постановка) або тензор послідовностей $T \times d$ (послідовнісна постановка), який передається до Inference Service.

Inference Service реалізовано як модуль, що завантажує збережені артефакти:

– ваги нейромережі (BiGRU + MultiHead Attention);

– конфігурацію гіперпараметрів (таблиця 2.2);

– параметри препроцесингу (скейлери/енкодери).

Під час інференсу реалізовано батчування та контроль часу обробки, що є важливим для режиму моніторингу.

Скріншот навчання нейромережевої моделі BiGRU + MultiHead Attention подано на рисунку 3.4.

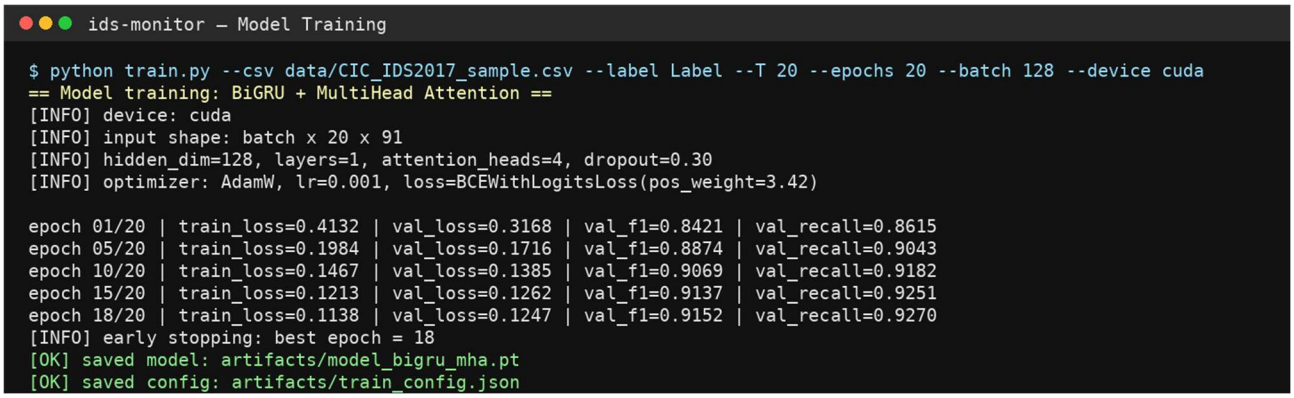

Рисунок 3.4 – Навчання нейромережевої моделі BiGRU + MultiHead Attention

Фрагмент коду – інференс і прогноз:

```
def predict(model, X):
    model.eval()
    with torch.no_grad():
        logits = model(X)               # (batch, 1) або (batch, K)
        probs = torch.sigmoid(logits)   # для бінарного випадку
    return probs.cpu().numpy()
```

Навчання моделі (optimizer, early stopping, фіксація найкращої версії) реалізовано окремим скриптом/модулем; логіка навчання та повні лістинги наведені в додатку В.

Decision Engine перетворює вихід моделі на інцидент, застосовуючи пороги та правила категоризації, описані в 3.1:

– поріг τ для бінарного рішення;

– додаткові правила (повторюваність, контекст активу, тип сервісу) для визначення критичності.

Фрагмент коду – поріг та категоризація:

```
def classify(score: float, tau: float = 0.5):
    label = int(score >= tau)
    if score >= 0.9: severity = "Critical"
    elif score >= 0.7: severity = "High"
    elif score >= 0.5: severity = "Medium"
    else: severity = "Low"
    return label, severity
```

Результатом є структурована подія (event), яка передається в Event Store та може бути доповнена посиланням на доказові артефакти (Evidence Store).

Збереження результатів реалізовано за принципом розділення:

– Event Store – структуровані події, метрики, версія моделі, службові поля аудиту;

– Evidence Store – PCAP-фрагменти/витяги, метадані, контроль цілісності (хеш), прив’язка до event_id.

Такий підхід відповідає практикам ведення доказової бази у системах IDS/IPS, які мають забезпечувати аудит і можливість відтворення інцидентів.

Фрагмент коду – формування події та хеш доказу:

```
def build_event(event_id, meta, score, model_ver):
    return {
        "event_id": event_id,
        "timestamp": meta["ts"],
        "flow": meta["flow_key"],
        "score": float(score),
        "model_version": model_ver
    }

def sha256_file(path: str) -> str:
    h = hashlib.sha256()
    with open(path, "rb") as f:
        for chunk in iter(lambda: f.read(8192), b""):
            h.update(chunk)
    return h.hexdigest()
```

Формування подій безпеки та збереження доказових артефактів представлено на рисунку 3.5.

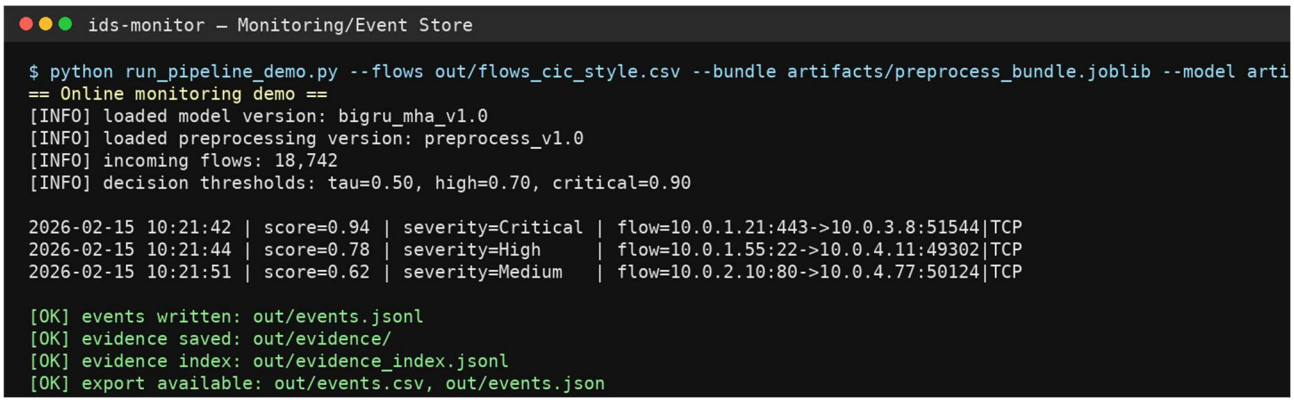

Рисунок 3.5 – Формування подій безпеки та збереження доказових артефактів

UI/дашборд прототипу відображає стрічку інцидентів із фільтрами за часом, критичністю, класом, а також надає перегляд деталей події (метадані потоку, оцінка ризику, посилання на доказ). Інтеграційний модуль підтримує

експорт подій у JSON/CSV та базові API-виклики для зовнішніх систем (SIEM/SOAR або аналітичні інструменти).

Фрагмент коду – експорт подій:

```
def export_events(events, out_path: str):
    with open(out_path, "w", encoding="utf-8") as f:
        for e in events:
            f.write(json.dumps(e, ensure_ascii=False) + "\n")
```

Надійність реалізації забезпечено:

– перевіркою вхідних даних і фіксацією аномалій (NaN/inf/порожні пакети);

– ізоляцією помилок між модулями (збір трафіку не зупиняється через збій інференсу);

– журналюванням подій і службових повідомлень (версія моделі, час інференсу, винятки);

– контролем цілісності доказів (хешування артефактів Evidence Store).

Отже, в даному підрозділі реалізовано програмний прототип інтелектуальної системи моніторингу мережевої безпеки відповідно до компонентної архітектури. Реалізацію структуровано на модулі, що забезпечило логічну відокремленість функцій, керованість і можливість подальшого розширення системи.

3.3 Експериментальні дослідження та оцінювання ефективності прототипу

Експериментальне оцінювання прототипу IDS виконано на двох еталонних наборах мережевих даних – CIC-IDS2017 та UNSW-NB15 [1, 20].

Для інтерпретації результатів використано метрики Accuracy, Recall, Precision, F1-score, оскільки в задачах мережевої безпеки типово спостерігається дисбаланс класів і зведення оцінювання лише до accuracy є методично некоректним. Практичні вимоги до доказовості інцидентів та аудиту подій враховано через підхід до логування/збереження подій і артефактів (Event Store/Evidence Store), описаний у підрозділі 3.1.

Для забезпечення відтворюваності експерименту використано однаковий набір метрик і методику розбиття даних на навчальну та тестову частини у співвідношенні 80/20.

Результати оцінювання запропонованої системи на CIC-IDS2017 та UNSW-NB15 подано у таблиці 3.2.

Таблиця 3.2 – Результати оцінювання системи на CIC-IDS2017 та UNSW-NB15

Датасет	Система	F1-score	Accuracy	Recall
CIC-IDS2017	Відома [37]	84	85	87
	Запропонована	89	90	91
UNSW-NB15	Відома [37]	81	82	84
	Запропонована	86	87	88

Для наочного порівняння наведено графічне зіставлення метрик окремо для кожного набору даних (рисунки 3.6-3.7).

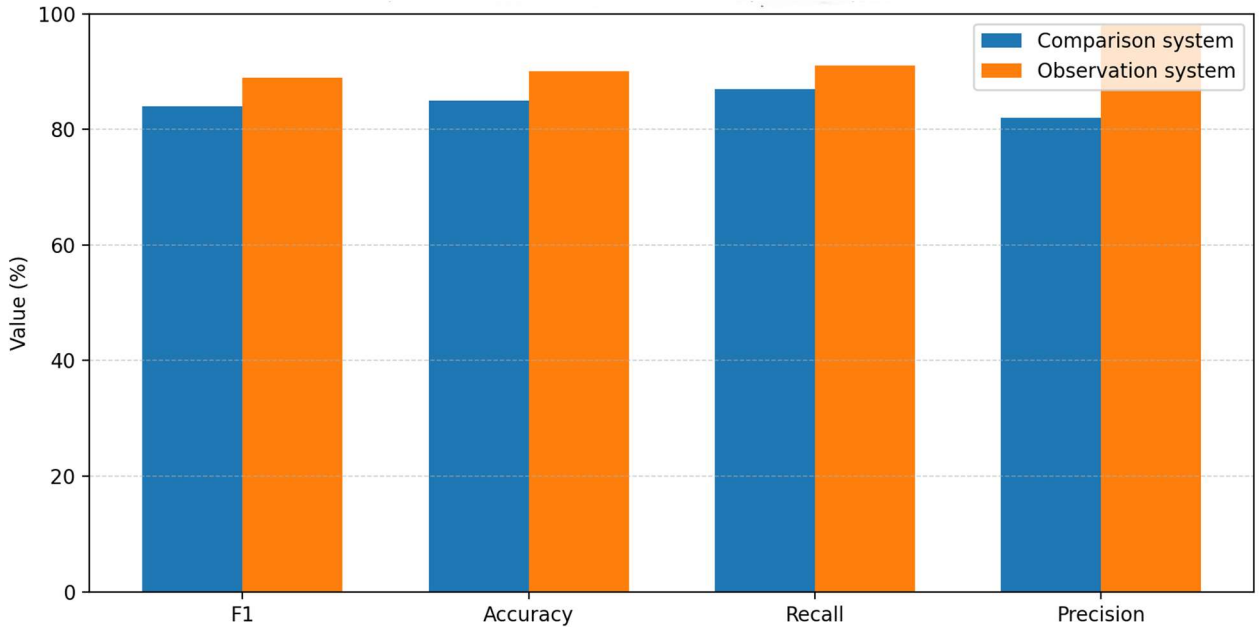

Рисунок 3.6 – Порівняння метрик на CIC-IDS2017

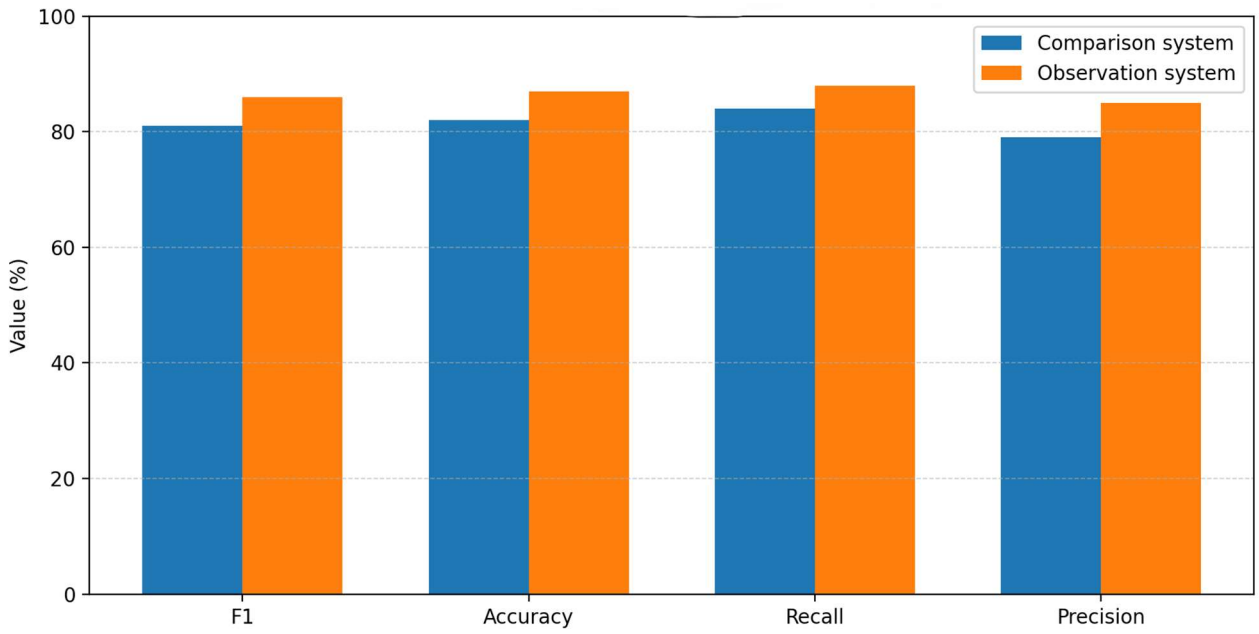

Рисунок 3.7 – Порівняння метрик на UNSW-NB15

Приріст метрик запропонованої системи подано в таблиці 3.3.

Таблиця 3.3 – Приріст метрик запропонованої системи

Метрика	Δ CIC-IDS2017	Δ UNSW-NB15
F1-score	+5	+5
Accuracy	+5	+5
Recall	+4	+4
Precision	+16	+6

Інтерпретаційно найбільш показовими є два аспекти:

1. Стійке зростання F1-score і Recall на обох датасетах (+5 п.п. для F1 та +4 п.п. для Recall), що є важливим для IDS, оскільки безпосередньо пов’язано зі зменшенням імовірності пропуску атак (FN) у контурі моніторингу.

2. Суттєве зростання Precision на CIC-IDS2017 (+16 п.п.), що вказує на зменшення частки хибних спрацювань серед усіх спрацювань, тобто потенційне зниження “шумового” навантаження на оператора SOC/чергового адміністратора.

Отримані числові результати узгоджуються з логікою вибору глибинних архітектур із механізмами “уваги”, які краще відображають часовий контекст мережевих подій (короткі інтенсивні сплески, повторюваність, зміни інтенсивності) та підсилюють внесок інформативних фрагментів трафіку. На цьому тлі:

– табличні підходи (зокрема DBN) можуть демонструвати конкурентну якість лише за умови, що часовий контекст “зашито” у набір ознак; однак це підвищує залежність від ручної інженерії ознак і доменної специфіки;

– легковагові архітектури (MobileViT, CapsNet+SRU) доцільні у спеціалізованих доменах із ресурсними обмеженнями (наприклад, CAN/промислові мережі), де ключовим критерієм стає затримка інференсу, а не максимізація F1;

– ELM-підхід доцільно трактувати як базову лінію швидкодії: у складних сценаріях із різнорідними атаками та часовою структурою очікуваною є перевага глибинних моделей за F1/Recall.

Порівняльний аналіз на CIC-IDS2017 та UNSW-NB15 демонструє перевагу запропонованої системи за всіма ключовими метриками, причому зростання Recall і F1-score має найбільшу практичну вагу для IDS. Зростання Precision (особливо на CIC-IDS2017) додатково свідчить про потенційне зниження кількості хибних тривог, що прямо впливає на керованість системи моніторингу в реальному середовищі.

Узагальнені результати експериментального оцінювання підтверджують, що запропонована система на основі глибокого навчання має перевагу за F1-score, Accuracy, Recall і Precision на CIC-IDS2017 та UNSW-NB15. Практичний сенс отриманого виграшу полягає у зменшенні ймовірності пропуску атак (покращення recall) та зниженні частки хибних спрацювань серед усіх спрацювань (покращення precision), що узгоджується з вимогами до систем IDS/IPS у частині моніторингу, аудиту та доказовості інцидентів.

ВИСНОВКИ

У кваліфікаційній роботі розглянуто та вирішено задачу розроблення інтелектуальної системи моніторингу безпеки комп’ютерної мережі на основі глибокого аналізу трафіку. Актуальність теми обґрунтовано з позицій практичної потреби безперервного виявлення мережевих атак за умов зростання обсягів трафіку, різноманіття сценаріїв вторгнень і доменної специфіки мережевих середовищ. Отримано наступні основні результати:

1. Проаналізовано предметну область мережевих систем виявлення вторгнень і визначено ключові вимоги до інтелектуальної системи: відтворюваність обробки даних, можливість роботи з потоковими ознаками, керованість порогів і підтримка аудиту подій. Окреслено роль еталонних наборів даних у верифікації підходів, а також застосування інструментів аналізу трафіку для валідації та підготовки даних.

2. Виконано огляд і порівняльний аналіз сучасних методів глибокого навчання для IDS, включаючи послідовнісні моделі, механізми уваги, а також альтернативні підходи та доменно-орієнтовані рішення для транспортних і промислових мереж. На підставі аналізу обґрунтовано доцільність використання архітектур, здатних враховувати часовий контекст трафіку та виділяти інформативні фрагменти послідовності.

3. Сформовано підхід до підготовки даних і побудови ознакового опису трафіку. Узагальнено перелік ключових груп ознак і визначено рекомендовані процедури їх обробки (очищення, робота з пропусками/аномаліями, масштабування, кодування полів, формування послідовностей), що забезпечує узгодженість входів моделі та відтворюваність експериментів. Окремо враховано можливість зменшення розмірності/компресії параметрів трафіку як засобу підвищення швидкодії та стабільності навчання.

4. Спроєктовано архітектуру програмної системи як набір взаємодіючих компонентів. Запропонований розподіл модулів забезпечує масштабованість, ізоляцію помилок і можливість інтеграції з зовнішніми системами, а також

відповідає сучасним підходам до інтелектуальної кібербезпеки та практикам документування інцидентів.

5. Реалізовано програмний прототип системи моніторингу з підтримкою обробки PCAP/flow-представлень, навчання та інференсу нейромережевої моделі, порогового прийняття рішень і категоризації інцидентів. Забезпечено структуроване логування подій та збереження доказової бази з контролем цілісності артефактів, що підвищує придатність рішення до аудиту та подальшого аналізу інцидентів.

6. Виконано експериментальне оцінювання та інтерпретацію результатів на двох наборах даних. За експериментальними даними запропонована система на основі глибокого навчання демонструє покращення метрик порівняно з базовою реалізацією: для CIC-IDS2017 підвищено F1-score та Accuracy на 5 п.п., Recall – на 4 п.п., Precision – на 16 п.п.; для UNSW-NB15 підвищено F1-score та Accuracy на 5 п.п., Recall – на 4 п.п., Precision – на 6 п.п. Отримані результати підтверджують доцільність використання глибинних моделей для виявлення вторгнень та їх потенціал для зменшення як пропусків атак, так і частки хибних спрацювань.

7. До перспективних напрямів подальших досліджень віднесено: (1) поглиблений аналіз помилок за окремими типами атак і оптимізацію порогів у контурі моніторингу; (2) реалізацію керованого оновлення моделі на основі верифікованих подій та підходів адаптивного навчання; (3) адаптацію системи для доменів із обмеженими ресурсами (зокрема транспортні мережі) та для сценаріїв аналізу шифрованого трафіку; (4) оцінювання швидкодії та затримки інференсу на потокових навантаженнях у режимі, наближеному до реального часу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Canadian Institute for Cybersecurity (CIC). Intrusion detection evaluation dataset (CIC-IDS2017). URL: https://www.unb.ca/cic/datasets/ids-2017.html (дата звернення: 15.02.2026).

2. Chen H., Zhang L., Jin H., et al. Vehicle CAN Intrusion Detection Method Based on MobileViT Lightweight Network. *Information Security Research*. 2024. Vol. 10, no. 5. P. 411–420.

3. Dorosh V., Komar M., Sachenko A., Golovko V. Parallel Deep Neural Network for Detecting Computer Attacks in Information Telecommunication Systems. *The 38th IEEE International Conference on Electronics and Nanotechnology* : Proceedings (Kyiv, Ukraine, April 24-26, 2018). Kyiv, 2018. P. 675-679.

4. Dubchak L., Komar M. Speedy procesing method of fuzzy data for intelligent systems of intrusion detection. *Projekt interdyscyplinarny projektem XXI wieku: Processing, transmission and security of information*. Bielsko-Biala, 2017. T. 2. P. 65-74.

5. Fan J., Ge L., Zhang H., et al. Intrusion Detection Model Integrating MultiHead Attention and BiGRU. *Computer and Digital Engineering*. 2023. Vol. 51, no. 1. P. 74–80.

6. Fang J., Leng F. Network Security Intrusion Detection System Based on Deep Learning. *Procedia Computer Science*. 2025. Vol. 261. P. 1107–1113.

7. Jiang Y., Xu Y., Li K., et al. A Lightweight In-Vehicle Network Intrusion Detection Method Based on Deep Learning. *Computer Engineering and Applications*. 2023. Vol. 59, no. 22. P. 284–292.

8. Komar M., Dorosh V., Sachenko A., Hladiy G. Deep Neural Network for Detection of Cyber Attacks. *The IEEE First International Conference on System Analysis & Intelligent Computing* : Proceedings (Kyiv, Ukraine October 08-12, 2018). Kyiv, 2018. P. 186-189.

9. Komar M., Kochan V., Dubchak L., Sachenko A., Golovko V., Bezobrazov S., Romanets I. High performance adaptive system for cyber attacks detection. *The 9th IEEE International Conference on Intelligent Data Acquisition and Advanced*

Computing Systems: Technology and Applications : Proceedings (Bucharest, Romania, September 21-23, 2017). Bucharest, 2017. Vol. 2. P. 853-858.

10. Komar M., Kochan V., Sachenko A., Ababii V. Improving of the Security of Intrusion Detection System. *The 13th International Conference on Development and Application Systems* : Proceedings (Suceava, Romania, May 19-21, 2016). Suceava, 2016. Pp. 315–319.

11. Komar M., Sachenko A., Bezobrazov S., Golovko V. Intelligent Cyber Defense System. *CEUR-WS*. 2016. Vol. 1614. P. 534-549.

12. Komar M., Sachenko A., Bezobrazov S., Golovko V. Intelligent Cyber Defense System Using Artificial Neural Network and Immune System Techniques. *Communications in Computer and Information Science*. 2017. Vol. 783. Pp. 36-55.

13. Komar M., Sachenko A., Golovko V., Dorosh V. Compression of Network Traffic Parameters for Detecting Cyber Attacks Based on Deep Learning. *The 9th IEEE International Conference on Dependable Systems, Services and Technologies* : Proceedings (Kyiv, Ukraine, May 24-27, 2018). Kyiv, 2018. P. 44-48.

14. Li C., Yuan Z., Teng G. Research on Spatiotemporal Feature Fusion Network Intrusion Detection Model Based on Deep Learning. *Information Security Research*. 2025. Vol. 11, no. 2. P. 122–129.

15. Li Q., Liu C., Gao G. Industrial Internet Intrusion Detection Method Based on CapsNet and SRU. *Computer Technology and Development*. 2024. Vol. 34, no. 7. P. 93–99.

16. Liu H. Research and Application of Network Intrusion Detection Method Based on Deep Learning. *Computer Programming Techniques and Maintenance*. 2023. No. 10. P. 162–165.

17. Qian J., Jia T., Zeng K., et al. Process Industry Intrusion Detection and Attack Localization for Process Data Analysis. *Modern Electronic Technology*. 2024. Vol. 47, no. 16. P. 117–124.

18. Qian K., Kang L., Wang Q. Research on Network Security Risks and Machine Learning Intrusion Detection Methods of SCADA System in Wind Farm. *Industrial Information Security*. 2024. No. 2. P. 47–52.

19. Sachenko A., Komar M. Intrusion detection system based on neural

networks / A. Sachenko. *Zeszyty naukowe politechniki Śląskiej: Organizacja i zarządzanie*. Gliwice, 2014. №. 68. P. 377–386.

20. UNSW. The UNSW-NB15 Dataset (project page). URL: https://research.unsw.edu.au/projects/unsw-nb15-dataset (дата звернення: 15.02.2026).

21. Vladov S., Vysotska V., Vashchenko S., Bolvinov S., Glubochenko S., Repchonok A., Korniienko M., Nazarkevych M., Herasymchuk R. Neural Network IDS/IPS Intrusion Detection and Prevention System with Adaptive Online Training to Improve Corporate Network Cybersecurity, Evidence Recording, and Interaction with Law Enforcement Agencies. *Big Data and Cognitive Computing*. 2025. Vol. 9, no. 11. P. 267.

22. Wireshark Foundation. Wireshark User's Guide (online documentation). URL: https://www.wireshark.org/docs/ (дата звернення: 15.02.2026).

23. Wu Y., Li W., Chen Y. Intrusion Detection Model Based on Deep Belief Network Fusion and Local Preserving Projection. *Computer Applications and Software*. 2024. Vol. 41, no. 6. P. 62–71.

24. Yu J. Design of BiLSTM Algorithm for Network Security Intrusion Detection. *Microcomputer Applications*. 2024. Vol. 40, no. 11. P. 222–225.

25. Zeng Q., He S., Chai J. Malicious TLS Traffic Detection Method Integrating Multimodal Features. *Information Security Research*. 2025. Vol. 11, no. 2. P. 130–138.

26. Zhao Y., Hu Y. Research and Optimization of Network Intrusion Detection Methods Based on Deep Learning. *Computer Programming Techniques and Maintenance*. 2024. No. 11. P. 161–164.

27. Zong X., Wang Z., He K., et al. Research on Data Enhancement and Detection Model for Industrial Intrusion Detection. *Computer Applications and Softwa*re. 2024. Vol. 41, no. 9. P. 370–376.

28. Комар М., Саченко А., Кочан В. Підвищення безпеки системи виявлення вторгнень на основі використання апаратних засобів. *Сучасні проблеми інформатики в економіці, управлінні, освіті та подоланні наслідків Чорнобильської катастрофи* : матер. XV-го міжнар. наук. семінару, Київ – оз. Світязь, 4-8 липня, 2016. С. 271-275.

29. Комар М.П. Еволюція детекторів атак на інформаційні телекомунікаційні мережі в складі системи колективного інтелекту. *Вимірювальна та обчислювальна техніка в технологічних процесах*. 2014. №4. С. 209–215.

30. Комар М.П. Зменшення розмірності даних для систем виявлення вторгнень на основі нейронної мережі глибокої довіри. *Науковий вісник Чернівецького національного університету: Комп'ютерні системи та компоненти.* 2016. Т. 7, вип. 2. С. 25-31.

31. Комар М.П. Підвищення стійкості комп'ютерних систем до кібератак. *Науковий вісник Чернівецького національного університету: Комп'ютерні системи та компоненти.* 2016. Т. 7, вип. 1. С. 6-12.

32. Комар М.П. Побудова ієрархічного класифікатора комп'ютерних атак на базі багатоканальних нейромережевих детекторів. *Вимірювальна та обчислювальна техніка в технологічних процесах.* 2015. №4. С. 199–124.

33. Про основні засади забезпечення кібербезпеки України : Закон України від 05.10.2017 № 2163-VIII // База даних «Законодавство України» / Верховна Рада України. URL: https://zakon.rada.gov.ua/go/2163-19 (дата звернення: 15.02.2026).

34. Скумін Т., Головко В., Саченко А., Комар М. Застосування нейронних мереж глибокої довіри для виявлення комп'ютерних атак. *Захист інформації і безпека інформаційних систем* : зб. тез міжнар. наук.-техн. конф. Львів, 2-3 червня, 2016. С. 162-164.

35. Спосіб виявлення комп'ютерних атак нейромережевою штучною імунною системою : пат. на винахід 109640 Україна : МПК(2012) H04W 12/08, G06F 21/00, G06F 12/14. № a201205350 ; заявл. 28.04.12 ; опубл. 25.09.15, Бюл. № 18/2015.

36. Спосіб ієрархічної класифікації комп'ютерних атак нейромережевою штучною імунною системою : пат. на кор. модель 127724 Україна : МПК (2006) H04W 12/08, G06F 21/00, G06F 12/14. № u201711238 ; заявл. 17.11.2017 ; опубл. 27.08.2018, Бюл. № 16/2018.

37. Білокур В.В. Проєктування нейромережевої моделі виявлення

вторгнень. Інтелектуальні комп’ютерні системи та мережі» (ІКСМ) : матеріали IV Всеукраїнської науково-практичної конференції молодих вчених і студентів (м. Тернопіль, ЗУНУ, 20 травня 2026 р.). Тернопіль, 2026. С. 129-132.

38. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання. Чинний від 2016-07-01. Вид. офіц. Київ: УкрНДНЦ, 2016. 16 с.

39. Комар М.П., Саченко А.О., Васильків Н.М., Гладій Г.М., Коваль В.С., Ліп’яніна-Гончаренко Х.В. Методичні рекомендації до виконання кваліфікаційної роботи з освітньо-професійної програми «Комп’ютерні науки» спеціальності 122 «Комп’ютерні науки» за першим (бакалаврським) рівнем вищої освіти. Тернопіль: ЗУНУ, 2024. 52 с.

40. Островерхов В. М., Біловус Л. І., Возьний К. З., Луцишин О. О., Монастирський Г. Л., Надвиничний С. А., Питель С. В., Шандрук С. К. Загальні методичні рекомендації з підготовки, оформлення, захисту та оцінювання кваліфікаційних робіт здобувачів вищої освіти першого (бакалаврського) і другого (магістерського) рівнів. Тернопіль: ЗУНУ, 2024. 83 с.

Додаток А

Результати порівняльного аналізу відомих підходів

Таблиця А.1 – Порівняльний аналіз підходів глибокого навчання для виявлення вторгнень

Підхід / джерело	Архітектура	Тип ознак (вхідні дані)	Область застосування	Переваги	Обмеження	Вимоги до ресурсів
1	2	3	4	5	6	7
MultiHead Attention + BiGRU [5]	Гібрид: механізм уваги + двонапрямна рекурентна мережа (BiGRU)	Послідовні/ потокові ознаки (flow features), часові вікна	Загальні мережі (NIDS), аналіз послідовностей	Краще врахування контексту та «важливих» фрагментів через attention; придатність до складних часових залежностей	Вища складність; чутливість до якості нормалізації та формування послідовностей	Середні/високі (особливо при MultiHead Attention)
BiLSTM для IDS [24]	Двонапрямна LSTM	Послідовності ознак трафіку (вікна/часові ряди)	Загальні мережі (NIDS)	Моделювання довготривалих залежностей; стійкість до шуму порівняно з простішими RNN	Повільніший інференс; потребує ретельного налаштування, можливе перенавчання	Середні/високі
Оптимізація DL-методів IDS [26]	Узагальнено: DL-моделі (рекурентні/гібридні)	Потокові або табличні ознаки	Загальні мережі	Показано напрями покращення (налаштування, оптимізація), узгодження метрик	Узагальнений характер; залежність результатів від датасету та препроцесингу	Залежить від моделі
DBN + Local Preserving Projection [23]	Глибока мережа довіри + проєкція зі збереженням локальної структури	Табличні/статистичні ознаки (flow-таблиці)	Загальні мережі (NIDS)	Добре працює з табличними ознаками; зменшення розмірності зі збереженням структури даних	Може втрачати часовий контекст; складність налаштування; ризик деградації при дрейфі трафіку	Середні

Продовження таблиці А.1

1	2	3	4	5	6	7
Mobile ViT для CAN IDS [2]	Легковагова мережа (MobileViT)	Ознаки CAN-кадрів/послідовностей	Транспортні мережі, CAN	Низька затримка й компактність; придатність до вбудованих середовищ	Обмежена переносимість на інші домени; залежність від специфіки CAN-повідомлень	Низькі/середні
Lightweight DL для in-vehicle IDS [7]	Полегшена DL-модель (орієнтація на швидкодію)	Ознаки транспортної мережі (in-vehicle traffic)	Транспортні мережі	Врахування обмежених ресурсів; практична придатність у реальному часі	Можливе зниження точності на складних атаках; доменна залежність	Низькі/середні
CapsNet + SRU для IIoT [15]	Капсульна мережа + SRU (послідовні блоки)	Потокові та/або часові ознаки IIoT	Промисловий інтернет (IIoT)	Здатність уловлювати ієрархічні залежності; ефективна обробка послідовностей	Складніша реалізація; потреба у якісній розмітці та стабільних ознаках	Середні/високі
Process Industry IDS + локалізація [17]	Модель детекції + модуль локалізації атаки	Процесні/технологічні дані + мережеві параметри	Процесна індустрія	Додає локалізацію (корисно для реагування); фокус на практичній інтерпретації	Вузька спеціалізація; залежить від структури процесних даних	Середні
SCADA IDS для вітропарку [18]	ML/IDS підхід для SCADA (ризик-орієнт.)	Ознаки SCADA/мережеві журнали	SCADA критичної інфраструктури	Урахування специфіки SCADA; акцент на ризиках	Обмежена узагальнюваність; складність збирання реальних даних	Середні
TLS detection (мультимодальні ознаки) [25]	Детектор з інтеграцією мультимодальних ознак	Метадані TLS-сесій + статистика потоків	Зашифрований трафік	Працює без payload; актуально для сучасних мереж	Чутливість до змін протоколів і легітимних профілів; ризик FP	Середні

Продовження таблиці А.1

1	2	3	4	5	6	7
Паралельна DNN для атак [3]	Паралельна глибинна нейромережа	Ознаки трафіку ITC (мережеві параметри)	Інфотелекомунікаційні системи	Прискорення обробки; придатність для великих потоків даних	Потреба у паралельній інфраструктурі; складність розгортання	Високі (але компенсуються паралельністю)
Адаптивна high-performance IDS [9]	Адаптивна система детекції (нейромережевий компонент)	Ознаки трафіку, можливі агреговані параметри	Загальні мережі/корпоративні сегменти	Фокус на продуктивності та адаптації; придатність до реальної експлуатації	Складність реалізації контурів адаптації; потреба в якісному логуванні	Середні/високі
Компресія параметрів трафіку + DL [13]	Компресія/зменшення розмірності + DL-детектор	Стиснуті параметри потоків	Загальні мережі (NIDS)	Зменшує обчислювальні витрати; прискорює навчання/інференс	Ризик втрати інформативності; потребує підбору рівня компресії	Низькі/середні
Adaptive Online Training IDS/IPS [21]	IDS/IPS з адаптивним online-тренуванням	Ознаки трафіку + журнали подій	Корпоративні мережі	Підвищення стійкості через online-оновлення; інтеграція з реєстрацією подій	Ризик «псування» моделі при некоректних даних; потребує контролю якості	Середні/високі
Unsupervised ELM [6]	Extreme Learning Machine (несупервіз.)	Табличні/статистичні ознаки	Загальні мережі	Швидке навчання; проста базова лінія для порівняння	Зазвичай нижча якість за DL на складних сценаріях; обмежена виразність	Низькі

Додаток Б

Перелік ключових ознак трафіку та їх призначення

Таблиця Б.1 – Перелік ключових ознак трафіку та їх призначення (CIC-IDS2017)

Назва ознаки	Група	Призначення	Коментар для IDS	Рекомендована обробка
1	2	3	4	5
Flow ID	Ідентифікатор	Унікальний ідентифікатор потоку	Технічний ключ, не несе поведінкової інформації	Видалити з навчання (ризик leakage/переднавчання)
Source IP, Destination IP	Ідентифікаційні	Адреси учасників обміну	Корисно для кореляції/логування, але небезпечно для навчання	Для моделі: видалити або анонімізувати/хешувати; для журналу: залишити
Source Port, Destination Port	Ідентифікаційні	Порти з’єднання	Можуть бути інформативними (сервіси), але є ризик “запам’ятовування”	За потреби: кодування (категоріальне/бінінг) або залишити як числове з нормалізацією
Protocol	Категоріальні	Протокол (TCP/UDP/ICMP)	Важливо для відмінностей профілів	One-hot або embedding; пропуски – рідкі
Timestamp	Часові	Мітка часу	Потрібна для побудови вікон/послідовностей	Для навчання (таблично): видалити; для послідовностей: використати для сортування/вікон
Flow Duration	Часові	Тривалість потоку	Відображає характер сесії	Нормалізація/стандартизація, перевірка на аномальні значення
Total Fwd Packets, Total Backward Packets	Об’ємні	К-сть пакетів у напрямках	Асиметрія важлива для сканувань/DoS	Нормалізація; нулі допустимі
Total Length of Fwd Packets, Total Length of Bwd Packets	Об’ємні	Байти у напрямках	Ознака інтенсивності/ексфільтрації	Нормалізація, лог-перетворення (за потреби)
Flow Bytes/s, Flow Packets/s	Інтенсивність	Швидкість потоку	Чутливі до помилок обчислення	Перевірка/очистка (inf/NaN), далі нормалізація

Продовження таблиці Б.1

1	2	3	4	5
Fwd Packet Length Max / Min / Mean/Std	Статистичні	Розподіл довжин forward-пакетів	Відображає “форму” трафіку	Нормалізація, контроль NaN
Bwd Packet Length Max/Min/Mean/Std	Статистичні	Розподіл довжин backward-пакетів	Аналогічно для зворотного напрямку	Нормалізація, контроль NaN
Flow IAT Mean/Std/Max/Min	Часові	Інтервали між пакетами в потоці	Важливо для повільних атак/ритміки	Нормалізація, контроль крайніх значень
Fwd IAT Total/Mean/Std/Max/Min	Часові	IAT у forward-напрямку	Додатковий часовий контекст	Нормалізація
Bwd IAT Total/Mean/Std/Max/Min	Часові	IAT у backward-напрямку	Показує відповіді/затримки	Нормалізація
FIN/SYN/RST/PSH/ACK/URG/ECE/CWE Flag Count	TCP/сеансові	Лічильники TCP-прапорців	Критичні для SYN-flood, аномальних завершень	Нормалізація (або залишити цілі), пропуски - заповнити 0
Down/Up Ratio	Напрямні	Асиметрія трафіку	Корисно для відсіву нетипових профілів	Нормалізація, контроль ділення на нуль
Average Packet Size	Статистичні	Середній розмір пакета	Уточнює профіль сесії	Нормалізація
Avg Fwd Segment Size, Avg Bwd Segment Size	Напрямні	Середні сегменти у напрямах	Відображає структуру обміну	Нормалізація
Active Mean/Std/Max/Min	Поведінкові	Активні фази потоку	Для “пульсуючих” патернів	Нормалізація, контроль NaN
Idle Mean/Std/Max/Min	Поведінкові	Фази простою	Актуально для beaconing/перерв	Нормалізація, контроль NaN
Label	Цільова	Мітка класу (норма/атака або тип)	Не використовується як ознака	Використати як у, перевірити баланс класів

Таблиця Б.2 – Перелік ключових ознак трафіку та їх призначення (UNSW-NB15)

Назва ознаки	Група	Призначення	Коментар для IDS	Рекомендована обробка
1	2	3	4	5
srcip, dstip	Ідентифікаційні	IP адреси джерела/призначення	Корисно для кореляції подій, але шкідливо для узагальнення	Для моделі: видалити або хеш/анонімізація; для логів: залишити
sport, dsport	Ідентифікаційні	Порти	Пов’язані з сервісами, але можуть спричинити “memorization”	Бінінг/кодування або залишити числовими + нормалізація
proto	Категоріальні	Протокол	Важливий контекст обміну	One-hot/embedding, пропуски – заповнення модою/окремою категорією
state	Категоріальні	Стан з’єднання	Дає ознаки розривів/аномалій	One-hot/embedding, пропуски → окрема категорія
service	Категоріальні	Сервіс (http, ftp, ssh тощо)	Підсилює відокремлення профілів	One-hot/embedding, “-” → окрема категорія
stime, ltime	Часові	Початок/кінець події	Потрібно для сортування/вікон	Для табличного навчання: видалити; для послідовностей: використати
dur	Часові	Тривалість	Аналог Flow Duration	Нормалізація/стандартизація, контроль крайніх
spkts, dpkts	Об’ємні	К-сть пакетів у напрямах	Асиметрія потоку	Нормалізація, нулі допустимі
sbytes, dbytes	Об’ємні	Байти у напрямах	Ознака інтенсивності/ексфільтрації	Нормалізація, за потреби лог-перетворення
sload, dload	Інтенсивність	Навантаження у напрямах	Чутливі до масштабу	Нормалізація, контроль inf/NaN
sintpkt, dintpkt	Часові	Інтервали між пакетами	Дає ритм обміну	Нормалізація, контроль крайніх
sjit, djit	Часові	Джитер у напрямах	Ознака нестабільності/тунелювання	Нормалізація, заповнення пропусків (0 або медіана)
sttl, dttl	IP-рівень	TTL у напрямах	Слабка, але інколи корисна ознака	Нормалізація (або залишити як цілі)
sloss, dloss	Якість передачі	Втрати/перевідправлення	Непрямий індикатор проблем каналу/аномалій	Нормалізація, пропуски → 0 (якщо семантично допустимо)

Продовження таблиці Б.2

1	2	3	4	5
tcprtt	TCP/сеансові	Round-trip time	Відрізняє мережеві стани/сканування	Нормалізація, контроль пропусків
synack, ackdat	TCP/сеансові	Фази рукостискання TCP	Інформативно для підозрілих встановлень	Нормалізація, пропуски → 0/медіана
attack_cat	Цільова	Категорія атаки	Для багатокласової постановки	у; за потреби кодування класів
Label	Цільова	Бінарна мітка (0/1)	Для задачі “норма/атака”	у; контроль дисбалансу

Додаток В

Лістинги програмної реалізації прототипу інтелектуальної системи

Лістинг В.1 – Модуль збору/підготовки трафіку

Файл: `capture_collector.py`

```
"""
capture_collector.py
Модуль Capture/Collector: імпорт PCAP, первинна фільтрація, базова агрегація
пакетів у потоки (flows).

Залежності:
  pip install scapy pandas

Примітка:
  Для повноцінного NetFlow/IPFIX режиму потрібні окремі колектори.
"""

from __future__ import annotations

from dataclasses import dataclass, asdict
from typing import Iterable, Iterator, Optional, Tuple, Dict, Any
import time
import math

import pandas as pd
from scapy.all import PcapReader, IP, TCP, UDP  # type: ignore


@dataclass(frozen=True)
class PacketRecord:
    ts: float
    src_ip: str
    dst_ip: str
    proto: str
    src_port: int
    dst_port: int
    length: int


@dataclass
class FlowAgg:
    """Базові агрегати flow-рівня (мінімальний набір для демонстрації)."""
    flow_key: str
    ts_start: float
    ts_end: float
    proto: str
    src_ip: str
    dst_ip: str
    src_port: int
    dst_port: int

    pkts: int = 0
    bytes: int = 0

    # прості статистики інтервалів (IAT)
    last_ts: Optional[float] = None
    iat_sum: float = 0.0
    iat_sq_sum: float = 0.0
```

```
    iat_min: float = math.inf
    iat_max: float = 0.0

    def update(self, pkt: PacketRecord) -> None:
        self.pkts += 1
        self.bytes += pkt.length
        self.ts_end = pkt.ts

        if self.last_ts is not None:
            dt = max(0.0, pkt.ts - self.last_ts)
            self.iat_sum += dt
            self.iat_sq_sum += dt * dt
            self.iat_min = min(self.iat_min, dt)
            self.iat_max = max(self.iat_max, dt)

        self.last_ts = pkt.ts

    def finalize(self) -> Dict[str, Any]:
        dur = max(0.0, self.ts_end - self.ts_start)
        iat_count = max(0, self.pkts - 1)
        iat_mean = (self.iat_sum / iat_count) if iat_count > 0 else 0.0
        iat_var = (self.iat_sq_sum / iat_count) - (iat_mean * iat_mean) if
iat_count > 0 else 0.0
        iat_std = math.sqrt(max(0.0, iat_var))

        return {
            "flow_key": self.flow_key,
            "ts_start": self.ts_start,
            "ts_end": self.ts_end,
            "duration": dur,
            "proto": self.proto,
            "src_ip": self.src_ip,
            "dst_ip": self.dst_ip,
            "src_port": self.src_port,
            "dst_port": self.dst_port,
            "pkts": self.pkts,
            "bytes": self.bytes,
            "pkts_per_s": (self.pkts / dur) if dur > 0 else 0.0,
            "bytes_per_s": (self.bytes / dur) if dur > 0 else 0.0,
            "iat_mean": iat_mean,
            "iat_std": iat_std,
            "iat_min": (self.iat_min if self.iat_min != math.inf else 0.0),
            "iat_max": self.iat_max,
        }


def _proto_name(ip_layer) -> str:
    # IP.proto: 6 TCP, 17 UDP, інше – як число
    p = int(getattr(ip_layer, "proto", -1))
    if p == 6:
        return "TCP"
    if p == 17:
        return "UDP"
    return str(p)


def iter_pcap_records(pcap_path: str, max_packets: Optional[int] = None) ->
Iterator[PacketRecord]:
    """
    Ітератор по PacketRecord з PCAP.
    Повертає лише IP+TCP/UDP пакети.
    """
    count = 0
    with PcapReader(pcap_path) as reader:
        for pkt in reader:
            if IP not in pkt:
```

```
                continue

            ip = pkt[IP]
            proto = _proto_name(ip)

            if TCP in pkt:
                l4 = pkt[TCP]
            elif UDP in pkt:
                l4 = pkt[UDP]
            else:
                continue

            try:
                rec = PacketRecord(
                    ts=float(pkt.time),
                    src_ip=str(ip.src),
                    dst_ip=str(ip.dst),
                    proto=proto,
                    src_port=int(l4.sport),
                    dst_port=int(l4.dport),
                    length=int(len(pkt)),
                )
            except Exception:
                # некоректний пакет – пропускається
                continue

            yield rec
            count += 1
            if max_packets is not None and count >= max_packets:
                break


def make_flow_key(rec: PacketRecord) -> str:
    """
    Ключ потоку у вигляді 5-tuple.
    Для демонстрації використано напрямний ключ.
    """
    return f"{rec.src_ip}:{rec.src_port}-
>{rec.dst_ip}:{rec.dst_port}|{rec.proto}"


def aggregate_to_flows(
    records: Iterable[PacketRecord],
    idle_timeout_s: float = 60.0
) -> pd.DataFrame:
    """
    Базова агрегація пакетів у flows.
    idle_timeout_s: якщо між пакетами потоку пауза більше порога – потік
закривається.
    """
    active: Dict[str, FlowAgg] = {}
    closed_rows: list[Dict[str, Any]] = []

    for rec in records:
        key = make_flow_key(rec)

        if key not in active:
            active[key] = FlowAgg(
                flow_key=key,
                ts_start=rec.ts,
                ts_end=rec.ts,
                proto=rec.proto,
                src_ip=rec.src_ip,
                dst_ip=rec.dst_ip,
                src_port=rec.src_port,
                dst_port=rec.dst_port,
```

```
            )

        flow = active[key]

        # idle timeout: якщо великий розрив – закривається попередній flow і
створюється новий
        if flow.last_ts is not None and (rec.ts - flow.last_ts) >
idle_timeout_s:
            closed_rows.append(flow.finalize())
            active[key] = FlowAgg(
                flow_key=key,
                ts_start=rec.ts,
                ts_end=rec.ts,
                proto=rec.proto,
                src_ip=rec.src_ip,
                dst_ip=rec.dst_ip,
                src_port=rec.src_port,
                dst_port=rec.dst_port,
            )
            flow = active[key]

        flow.update(rec)

    # закриття активних потоків
    for flow in active.values():
        closed_rows.append(flow.finalize())

    return pd.DataFrame(closed_rows)


def run_pcap_to_flows(pcap_path: str, out_csv: str, bpf_hint: Optional[str] =
None) -> None:
    """
    Демонстраційний запуск: PCAP → flows.csv.
    bpf_hint не застосовується в scapy PcapReader як BPF-фільтр (лише довідково
для користувача).
    """
    t0 = time.time()
    recs = iter_pcap_records(pcap_path)
    df = aggregate_to_flows(recs)
    df.to_csv(out_csv, index=False)
    dt = time.time() - t0
    print(f"[capture] saved flows: {len(df)} rows to {out_csv} in {dt:.2f}s")


if __name__ == "__main__":
    import argparse

    ap = argparse.ArgumentParser()
    ap.add_argument("--pcap", required=True, help="Path to .pcap/.pcapng")
    ap.add_argument("--out", required=True, help="Output CSV for flows")
    ap.add_argument("--idle", type=float, default=60.0)
    args = ap.parse_args()

    df = aggregate_to_flows(iter_pcap_records(args.pcap),
idle_timeout_s=args.idle)
    df.to_csv(args.out, index=False)
    print(f"OK: {args.out} ({len(df)} flows)")
```

Лістинг В.2 – Модуль формування ознак

Файл: `feature_extractor.py`

```
"""
feature_extractor.py
Модуль Feature Extractor: очищення, обробка пропусків/аномалій, масштабування,
кодування категоріальних,
формування послідовностей довжини T.

Залежності:
  pip install pandas numpy scikit-learn joblib
"""

from __future__ import annotations

from dataclasses import dataclass
from typing import List, Dict, Any, Tuple, Optional

import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
import joblib


@dataclass
class PreprocessBundle:
    pipeline: Pipeline
    feature_names: List[str]


def _sanitize_df(df: pd.DataFrame) -> pd.DataFrame:
    df = df.copy()
    df = df.replace([np.inf, -np.inf], np.nan)
    # Базове правило: пропуски → 0; за потреби замінюється на медіану/моду
    df = df.fillna(0.0)
    return df


def build_preprocess_pipeline(
    numeric_cols: List[str],
    categorical_cols: List[str],
) -> PreprocessBundle:
    """
    Формує препроцесор:
      - числові: StandardScaler
      - категоріальні: OneHotEncoder(handle_unknown="ignore")
    """
    num_pipe = Pipeline(steps=[
        ("scaler", StandardScaler(with_mean=True, with_std=True)),
    ])

    cat_pipe = Pipeline(steps=[
        ("ohe", OneHotEncoder(handle_unknown="ignore", sparse_output=False)),
    ])

    ct = ColumnTransformer(
        transformers=[
            ("num", num_pipe, numeric_cols),
            ("cat", cat_pipe, categorical_cols),
        ],
        remainder="drop"
    )

    pipe = Pipeline(steps=[
        ("ct", ct),
    ])
```

```
    # імена ознак будуть відомі після fit
    return PreprocessBundle(pipeline=pipe, feature_names=[])


def fit_transform(
    df: pd.DataFrame,
    bundle: PreprocessBundle,
    label_col: Optional[str] = None,
    drop_cols: Optional[List[str]] = None,
) -> Tuple[np.ndarray, Optional[np.ndarray], PreprocessBundle]:
    """
    Fit+Transform для train.
    """
    df = _sanitize_df(df)

    if drop_cols:
        for c in drop_cols:
            if c in df.columns:
                df = df.drop(columns=[c])

    y = None
    if label_col is not None and label_col in df.columns:
        y = df[label_col].to_numpy()
        df = df.drop(columns=[label_col])

    X = bundle.pipeline.fit_transform(df)

    # формування назв ознак після fit
    ct: ColumnTransformer = bundle.pipeline.named_steps["ct"]
    num_names =
ct.named_transformers_["num"].named_steps["scaler"].get_feature_names_out(ct.tra
nsformers_[0][2])
    ohe: OneHotEncoder = ct.named_transformers_["cat"].named_steps["ohe"]
    cat_names = ohe.get_feature_names_out(ct.transformers_[1][2])
    feature_names = list(num_names) + list(cat_names)

    bundle.feature_names = feature_names
    return X.astype(np.float32), y, bundle


def transform(
    df: pd.DataFrame,
    bundle: PreprocessBundle,
    label_col: Optional[str] = None,
    drop_cols: Optional[List[str]] = None,
) -> Tuple[np.ndarray, Optional[np.ndarray]]:
    """
    Transform для val/test/online.
    """
    df = _sanitize_df(df)

    if drop_cols:
        for c in drop_cols:
            if c in df.columns:
                df = df.drop(columns=[c])

    y = None
    if label_col is not None and label_col in df.columns:
        y = df[label_col].to_numpy()
        df = df.drop(columns=[label_col])

    X = bundle.pipeline.transform(df)
    return X.astype(np.float32), y


def make_windows(X: np.ndarray, T: int) -> np.ndarray:
```

```
    """
    Формування вікон для рекурентної моделі.
    Вхід: X (N, d) → Вихід: (N-T+1, T, d)
    """
    if T <= 1:
        return X.reshape(X.shape[0], 1, X.shape[1])

    n, d = X.shape
    if n < T:
        raise ValueError(f"Not enough rows to form windows: N={n}, T={T}")

    out = np.zeros((n - T + 1, T, d), dtype=np.float32)
    for i in range(n - T + 1):
        out[i] = X[i:i+T]
    return out


def save_bundle(bundle: PreprocessBundle, path: str) -> None:
    joblib.dump({"pipeline": bundle.pipeline, "feature_names":
bundle.feature_names}, path)


def load_bundle(path: str) -> PreprocessBundle:
    obj = joblib.load(path)
    return PreprocessBundle(pipeline=obj["pipeline"],
feature_names=obj.get("feature_names", []))
```

Лістинг В.3 – Нейромережева модель

Файл: `model_bigru_mha.py`

```
"""
model_bigru_mha.py
PyTorch модель: BiGRU + MultiHead Attention + pooling + classifier.

Залежності:
  pip install torch
"""

from __future__ import annotations

import torch
import torch.nn as nn


class BiGRU_MHA(nn.Module):
    def __init__(
        self,
        input_dim: int,
        hidden_dim: int = 128,
        num_layers: int = 1,
        attn_heads: int = 4,
        dropout: float = 0.3,
        num_classes: int = 1,   # 1 для бінарної; K для багатокласової
    ):
        super().__init__()

        self.gru = nn.GRU(
            input_size=input_dim,
            hidden_size=hidden_dim,
            num_layers=num_layers,
            batch_first=True,
            bidirectional=True,
```

```
            dropout=dropout if num_layers > 1 else 0.0,
        )

        # BiGRU вихід: 2*hidden_dim
        self.attn = nn.MultiheadAttention(
            embed_dim=2 * hidden_dim,
            num_heads=attn_heads,
            dropout=dropout,
            batch_first=True
        )

        self.dropout = nn.Dropout(dropout)
        self.ff = nn.Sequential(
            nn.Linear(2 * hidden_dim, 256),
            nn.ReLU(),
            nn.Dropout(dropout),
            nn.Linear(256, num_classes),
        )

    def forward(self, x: torch.Tensor) -> torch.Tensor:
        """
        x: (batch, T, d)
        """
        h, _ = self.gru(x)  # (batch, T, 2H)

        # self-attention: Q=K=V=h
        attn_out, _ = self.attn(h, h, h)  # (batch, T, 2H)

        # pooling (mean)
        z = attn_out.mean(dim=1)  # (batch, 2H)
        z = self.dropout(z)

        logits = self.ff(z)       # (batch, 1) або (batch, K)
        return logits
```

Лістинг В.4 – Навчання моделі та збереження артефактів

Файл: `train.py`

```
"""
train.py
Навчання BiGRU_MHA на підготовлених таблицях (CSV) з формуванням вікон T.
Збереження:
  - preprocess_bundle.joblib
  - model.pt
  - train_config.json

Залежності:
  pip install pandas numpy scikit-learn joblib torch
"""

from __future__ import annotations

import json
from dataclasses import dataclass, asdict
from typing import List, Optional, Tuple

import numpy as np
import pandas as pd
import torch
import torch.nn as nn
from torch.utils.data import TensorDataset, DataLoader
```

```
from feature_extractor import build_preprocess_pipeline, fit_transform,
make_windows, save_bundle
from model_bigru_mha import BiGRU_MHA


@dataclass
class TrainConfig:
    csv_path: str
    label_col: str = "Label"
    drop_cols: Optional[List[str]] = None

    T: int = 20
    batch: int = 128
    epochs: int = 20
    lr: float = 1e-3
    weight_decay: float = 0.0

    hidden_dim: int = 128
    num_layers: int = 1
    attn_heads: int = 4
    dropout: float = 0.3

    num_classes: int = 1  # 1: binary
    device: str = "cpu"

    out_bundle: str = "preprocess_bundle.joblib"
    out_model: str = "model.pt"
    out_config: str = "train_config.json"


def _split_train_val(X: np.ndarray, y: np.ndarray, val_ratio: float = 0.2, seed:
int = 42):
    rng = np.random.default_rng(seed)
    idx = np.arange(len(X))
    rng.shuffle(idx)
    n_val = int(len(X) * val_ratio)
    val_idx = idx[:n_val]
    tr_idx = idx[n_val:]
    return X[tr_idx], y[tr_idx], X[val_idx], y[val_idx]


def _bce_loss(logits: torch.Tensor, y: torch.Tensor, pos_weight: Optional[float]
= None) -> nn.Module:
    if pos_weight is None:
        return nn.BCEWithLogitsLoss()
    w = torch.tensor([pos_weight], dtype=torch.float32, device=logits.device)
    return nn.BCEWithLogitsLoss(pos_weight=w)


def main(cfg: TrainConfig) -> None:
    df = pd.read_csv(cfg.csv_path)

    # Визначення колонок (приклад для flow-таблиці прототипу).
    # Для CIC/UNSW перелік підбирається окремо згідно таблиць 2.2а–2.2б.
    if cfg.drop_cols is None:
        cfg.drop_cols = ["flow_key", "src_ip", "dst_ip"]  # ідентифікатори →
виключаються

    numeric_cols = [c for c in df.columns if c not in (cfg.drop_cols +
[cfg.label_col]) and df[c].dtype != "object"]
    categorical_cols = [c for c in df.columns if c not in (cfg.drop_cols +
[cfg.label_col]) and df[c].dtype == "object"]

    bundle = build_preprocess_pipeline(numeric_cols, categorical_cols)
    X_tab, y_raw, bundle = fit_transform(df, bundle, label_col=cfg.label_col,
drop_cols=cfg.drop_cols)
```

```
    if y_raw is None:
        raise ValueError("Label column not found")

    # Бінаризація міток: очікується 0/1 або текст (Benign/Attack)
    if y_raw.dtype.kind in ("U", "S", "O"):
        y = np.array([0 if str(v).lower() in ("benign", "normal", "0") else 1
for v in y_raw], dtype=np.float32)
    else:
        y = y_raw.astype(np.float32)

    X_seq = make_windows(X_tab, cfg.T)
    y_seq = y[cfg.T - 1:]  # вирівнювання міток з кінцем вікна

    Xtr, ytr, Xva, yva = _split_train_val(X_seq, y_seq)

    device = torch.device(cfg.device if torch.cuda.is_available() or cfg.device
== "cpu" else "cpu")

    model = BiGRU_MHA(
        input_dim=X_seq.shape[-1],
        hidden_dim=cfg.hidden_dim,
        num_layers=cfg.num_layers,
        attn_heads=cfg.attn_heads,
        dropout=cfg.dropout,
        num_classes=cfg.num_classes
    ).to(device)

    # pos_weight для дисбалансу (простий варіант)
    pos = float((ytr == 1).sum())
    neg = float((ytr == 0).sum())
    pos_weight = (neg / pos) if pos > 0 else None

    optimizer = torch.optim.AdamW(model.parameters(), lr=cfg.lr,
weight_decay=cfg.weight_decay)
    best_val = float("inf")
    best_state = None

    def make_loader(Xn, yn, shuffle: bool):
        ds = TensorDataset(torch.tensor(Xn), torch.tensor(yn).unsqueeze(1))
        return DataLoader(ds, batch_size=cfg.batch, shuffle=shuffle,
drop_last=False)

    tr_loader = make_loader(Xtr, ytr, shuffle=True)
    va_loader = make_loader(Xva, yva, shuffle=False)

    for ep in range(1, cfg.epochs + 1):
        model.train()
        tr_loss = 0.0
        for xb, yb in tr_loader:
            xb = xb.to(device)
            yb = yb.to(device)

            optimizer.zero_grad()
            logits = model(xb)
            loss_fn = _bce_loss(logits, yb, pos_weight=pos_weight)
            loss = loss_fn(logits, yb)
            loss.backward()
            optimizer.step()
            tr_loss += float(loss.item()) * xb.size(0)

        tr_loss /= len(tr_loader.dataset)

        model.eval()
        va_loss = 0.0
        with torch.no_grad():
```

```
            for xb, yb in va_loader:
                xb = xb.to(device)
                yb = yb.to(device)
                logits = model(xb)
                loss_fn = _bce_loss(logits, yb, pos_weight=pos_weight)
                loss = loss_fn(logits, yb)
                va_loss += float(loss.item()) * xb.size(0)
        va_loss /= len(va_loader.dataset)

        print(f"[train] epoch={ep:02d} train_loss={tr_loss:.4f}
val_loss={va_loss:.4f}")

        if va_loss < best_val:
            best_val = va_loss
            best_state = {k: v.detach().cpu().clone() for k, v in
model.state_dict().items()}

    if best_state is not None:
        model.load_state_dict(best_state)

    # збереження артефактів
    save_bundle(bundle, cfg.out_bundle)
    torch.save(model.state_dict(), cfg.out_model)
    with open(cfg.out_config, "w", encoding="utf-8") as f:
        json.dump(asdict(cfg), f, ensure_ascii=False, indent=2)

    print(f"[train] saved: {cfg.out_bundle}, {cfg.out_model}, {cfg.out_config}")


if __name__ == "__main__":
    import argparse

    ap = argparse.ArgumentParser()
    ap.add_argument("--csv", required=True)
    ap.add_argument("--label", default="Label")
    ap.add_argument("--T", type=int, default=20)
    ap.add_argument("--epochs", type=int, default=20)
    ap.add_argument("--batch", type=int, default=128)
    ap.add_argument("--lr", type=float, default=1e-3)
    ap.add_argument("--device", default="cpu")
    args = ap.parse_args()

    cfg = TrainConfig(
        csv_path=args.csv,
        label_col=args.label,
        T=args.T,
        epochs=args.epochs,
        batch=args.batch,
        lr=args.lr,
        device=args.device
    )
    main(cfg)
```

Лістинг В.5 – Модуль інференсу

Файл: `inference_service.py`

```
"""
inference_service.py
Модуль інференсу: завантаження preprocess+model, підготовка ознак, формування
вікон, прогноз.

Залежності:
```

```
  pip install pandas numpy torch joblib scikit-learn
"""

from __future__ import annotations

from dataclasses import dataclass
from typing import Optional, List, Dict, Any

import numpy as np
import pandas as pd
import torch

from feature_extractor import load_bundle, transform, make_windows
from model_bigru_mha import BiGRU_MHA


@dataclass
class InferenceArtifacts:
    bundle_path: str
    model_path: str
    T: int
    hidden_dim: int = 128
    num_layers: int = 1
    attn_heads: int = 4
    dropout: float = 0.3
    num_classes: int = 1
    device: str = "cpu"


class InferenceService:
    def __init__(self, art: InferenceArtifacts, drop_cols: Optional[List[str]] =
None):
        self.art = art
        self.drop_cols = drop_cols or ["flow_key", "src_ip", "dst_ip"]

        self.bundle = load_bundle(art.bundle_path)
        self.device = torch.device(art.device if torch.cuda.is_available() or
art.device == "cpu" else "cpu")

        # input_dim визначається після transform, тому модель ініціалізується
"пізніше"
        self.model = None

        self._load_state = torch.load(art.model_path, map_location="cpu")

    def _ensure_model(self, input_dim: int) -> None:
        if self.model is not None:
            return
        self.model = BiGRU_MHA(
            input_dim=input_dim,
            hidden_dim=self.art.hidden_dim,
            num_layers=self.art.num_layers,
            attn_heads=self.art.attn_heads,
            dropout=self.art.dropout,
            num_classes=self.art.num_classes
        )
        self.model.load_state_dict(self._load_state)
        self.model.to(self.device)
        self.model.eval()

    def predict_from_df(self, df: pd.DataFrame) -> np.ndarray:
        X_tab, _ = transform(df, self.bundle, label_col=None,
drop_cols=self.drop_cols)
        X_seq = make_windows(X_tab, self.art.T)
        self._ensure_model(X_seq.shape[-1])
```

```
        with torch.no_grad():
            xb = torch.tensor(X_seq, dtype=torch.float32, device=self.device)
            logits = self.model(xb)
            probs = torch.sigmoid(logits).squeeze(1).detach().cpu().numpy()
        return probs


if __name__ == "__main__":
    import argparse

    ap = argparse.ArgumentParser()
    ap.add_argument("--csv", required=True)
    ap.add_argument("--bundle", required=True)
    ap.add_argument("--model", required=True)
    ap.add_argument("--T", type=int, default=20)
    ap.add_argument("--device", default="cpu")
    args = ap.parse_args()

    svc = InferenceService(InferenceArtifacts(
        bundle_path=args.bundle,
        model_path=args.model,
        T=args.T,
        device=args.device
    ))

    df = pd.read_csv(args.csv)
    probs = svc.predict_from_df(df)

    print(f"Predictions: n={len(probs)}, min={probs.min():.4f},
max={probs.max():.4f}")
```

Лістинг А.6 – Модуль прийняття рішення та категоризації інцидентів

Файл: `decision_engine.py`

```
"""
decision_engine.py
Decision Engine: порогове рішення, категоризація критичності, формування
структури інциденту.
"""

from __future__ import annotations

from dataclasses import dataclass
from typing import Dict, Any, Tuple
import time
import uuid


@dataclass
class ThresholdPolicy:
    tau: float = 0.5
    high: float = 0.7
    critical: float = 0.9


def classify(score: float, policy: ThresholdPolicy) -> Tuple[int, str]:
    label = int(score >= policy.tau)

    if score >= policy.critical:
        sev = "Critical"
    elif score >= policy.high:
        sev = "High"
```

```
    elif score >= policy.tau:
        sev = "Medium"
    else:
        sev = "Low"

    return label, sev


def build_event(meta: Dict[str, Any], score: float, model_version: str, policy:
ThresholdPolicy) -> Dict[str, Any]:
    """
    meta: контекст потоку/події (flow_key, ts, src/dst тощо)
    """
    label, severity = classify(score, policy)
    event_id = str(uuid.uuid4())

    return {
        "event_id": event_id,
        "timestamp": float(meta.get("ts", time.time())),
        "flow_key": meta.get("flow_key"),
        "src_ip": meta.get("src_ip"),
        "dst_ip": meta.get("dst_ip"),
        "src_port": meta.get("src_port"),
        "dst_port": meta.get("dst_port"),
        "proto": meta.get("proto"),
        "score": float(score),
        "label": int(label),
        "severity": severity,
        "model_version": model_version,
        "thresholds": {
            "tau": policy.tau,
            "high": policy.high,
            "critical": policy.critical
        }
    }
```

Лістинг А.7 – Сховище подій та доказів

Файл: `storage.py`

```
"""
storage.py
Event Store / Evidence Store: збереження подій (JSONL) та доказів
(PCAP/фрагменти) з контролем цілісності.

Залежності:
  стандартна бібліотека Python
"""

from __future__ import annotations

import os
import json
import hashlib
import shutil
from dataclasses import dataclass
from typing import Dict, Any, Optional


def sha256_file(path: str) -> str:
    h = hashlib.sha256()
    with open(path, "rb") as f:
        for chunk in iter(lambda: f.read(8192), b""):
```

```
            h.update(chunk)
    return h.hexdigest()


@dataclass
class StorageConfig:
    event_jsonl: str = "events.jsonl"
    evidence_dir: str = "evidence"
    index_jsonl: str = "evidence_index.jsonl"


class EventStore:
    def __init__(self, cfg: StorageConfig):
        self.cfg = cfg
        os.makedirs(os.path.dirname(cfg.event_jsonl) or ".", exist_ok=True)

    def append(self, event: Dict[str, Any]) -> None:
        with open(self.cfg.event_jsonl, "a", encoding="utf-8") as f:
            f.write(json.dumps(event, ensure_ascii=False) + "\n")


class EvidenceStore:
    def __init__(self, cfg: StorageConfig):
        self.cfg = cfg
        os.makedirs(cfg.evidence_dir, exist_ok=True)
        os.makedirs(os.path.dirname(cfg.index_jsonl) or ".", exist_ok=True)

    def save_artifact(self, event_id: str, src_path: str, note: Optional[str] =
None) -> Dict[str, Any]:
        """
        Копіює артефакт у evidence_dir, рахує хеш, додає індексний запис.
        """
        base = os.path.basename(src_path)
        dst_name = f"{event_id}__{base}"
        dst_path = os.path.join(self.cfg.evidence_dir, dst_name)

        shutil.copy2(src_path, dst_path)
        digest = sha256_file(dst_path)

        rec = {
            "event_id": event_id,
            "path": dst_path,
            "sha256": digest,
            "note": note
        }

        with open(self.cfg.index_jsonl, "a", encoding="utf-8") as f:
            f.write(json.dumps(rec, ensure_ascii=False) + "\n")

        return rec
```

Лістинг А.8 – Експорт подій та API-інтерфейс

Файл: `export_api.py`

```
"""
export_api.py
Експорт подій з JSONL у CSV/JSON, мінімальний API-ендпойнт (демо).

Залежності:
  pip install pandas flask
"""
```

```
from __future__ import annotations

import json
from typing import List, Dict, Any

import pandas as pd
from flask import Flask, jsonify, request  # type: ignore


def read_jsonl(path: str) -> List[Dict[str, Any]]:
    out = []
    with open(path, "r", encoding="utf-8") as f:
        for line in f:
            line = line.strip()
            if not line:
                continue
            out.append(json.loads(line))
    return out


def export_csv(events: List[Dict[str, Any]], out_csv: str) -> None:
    df = pd.DataFrame(events)
    df.to_csv(out_csv, index=False)


app = Flask(__name__)
EVENTS: List[Dict[str, Any]] = []


@app.get("/events")
def api_events():
    severity = request.args.get("severity")
    if severity:
        filtered = [e for e in EVENTS if str(e.get("severity")) == severity]
        return jsonify(filtered)
    return jsonify(EVENTS)


@app.get("/stats")
def api_stats():
    total = len(EVENTS)
    by_sev = {}
    for e in EVENTS:
        s = str(e.get("severity", "NA"))
        by_sev[s] = by_sev.get(s, 0) + 1
    return jsonify({"total": total, "by_severity": by_sev})


if __name__ == "__main__":
    import argparse

    ap = argparse.ArgumentParser()
    ap.add_argument("--jsonl", required=True, help="events.jsonl")
    ap.add_argument("--export_csv", default=None)
    ap.add_argument("--run_api", action="store_true")
    args = ap.parse_args()

    EVENTS = read_jsonl(args.jsonl)

    if args.export_csv:
        export_csv(EVENTS, args.export_csv)
        print(f"Exported CSV: {args.export_csv}")

    if args.run_api:
        app.run(host="127.0.0.1", port=8080, debug=False)
```

Лістинг А.9 – Демонстраційний сценарій конвеєра

Файл: `run_pipeline_demo.py`

```
"""
run_pipeline_demo.py
Демо-конвеєр:
  1) читає flows.csv (або іншу таблицю ознак)
  2) робить інференс
  3) формує події та записує в Event Store
  4) (опційно) зберігає артефакт доказу

Залежності:
  pip install pandas numpy torch joblib scikit-learn
"""

from __future__ import annotations

import time
import pandas as pd

from inference_service import InferenceService, InferenceArtifacts
from decision_engine import build_event, ThresholdPolicy
from storage import StorageConfig, EventStore, EvidenceStore


def main(
    flows_csv: str,
    bundle_path: str,
    model_path: str,
    T: int,
    model_version: str = "v1.0",
    evidence_file: str | None = None
) -> None:
    df = pd.read_csv(flows_csv)

    svc = InferenceService(InferenceArtifacts(
        bundle_path=bundle_path,
        model_path=model_path,
        T=T,
        device="cpu"
    ))

    probs = svc.predict_from_df(df)

    cfg = StorageConfig(event_jsonl="out/events.jsonl",
evidence_dir="out/evidence", index_jsonl="out/evidence_index.jsonl")
    es = EventStore(cfg)
    evs = EvidenceStore(cfg)

    policy = ThresholdPolicy(tau=0.5, high=0.7, critical=0.9)

    # вирівнювання: probs відповідає зразкам з кінця вікон
    base_rows = df.iloc[T-1:].reset_index(drop=True)

    for i, score in enumerate(probs):
        row = base_rows.iloc[i].to_dict()
        meta = {
            "ts": float(row.get("ts_end", time.time())),
            "flow_key": row.get("flow_key"),
            "src_ip": row.get("src_ip"),
            "dst_ip": row.get("dst_ip"),
```

```
            "src_port": row.get("src_port"),
            "dst_port": row.get("dst_port"),
            "proto": row.get("proto"),
        }

        event = build_event(meta, float(score), model_version=model_version,
policy=policy)
        es.append(event)

        # якщо критичний інцидент – можна додати доказовий артефакт
        if evidence_file and event["severity"] in ("High", "Critical"):
            evs.save_artifact(event["event_id"], evidence_file, note="Demo
evidence artifact")

    print(f"OK: saved events to {cfg.event_jsonl}")


if __name__ == "__main__":
    import argparse

    ap = argparse.ArgumentParser()
    ap.add_argument("--flows", required=True, help="flows.csv")
    ap.add_argument("--bundle", required=True)
    ap.add_argument("--model", required=True)
    ap.add_argument("--T", type=int, default=20)
    ap.add_argument("--evidence", default=None)
    args = ap.parse_args()

    main(args.flows, args.bundle, args.model, args.T,
evidence_file=args.evidence)
```

Додаток Г

Апробація результатів роботи

Почесні співголови конференції:

Десятнюк О.М., ректор Західноукраїнського національного університету, д-р економічних наук, професор;

Дивак М.П., д-р технічних наук, професор, проректор з наукової роботи ЗУНУ;

Голова конференції:

Березький О.М., д-р технічних наук, професор, професор кафедри комп'ютерної інженерії Західноукраїнський національний університет;

Заступники голови конференції:

Мельник Г.М., к.т.н, доцент, Західноукраїнський національний університет;

Піцун О.Й., к.т.н. доцент, Західноукраїнський національний університет;

ПРОГРАМНИЙ КОМІТЕТ КОНФЕРЕНЦІЇ

Склад програмного комітету:

Семанюк В.З., д-р економічних наук, професор, начальник науково-дослідної частини Західноукраїнського національного університету

Антощук С.Г., д.т.н, професор, Національний університет «Одеська політехніка»;

Баловсяк С.В., д.т.н, професор, Чернівецький національний університет

Бармак О.В., д.т.н., професор, Хмельницький національний університет

Батько Ю.М., к.т.н., доцент, Західноукраїнський національний університет

Березька К.М.,к.т.н.,доцент, Західноукраїнський національний університет

Винокурова О.А., д.т.н., професор, Львівський національний університет імені Івана Франка

Возна Н.Я., д.т.н., професор, Західноукраїнський національний університет;

Говорущенко Т.О., д.т.н., професор, Хмельницький національний університет;

Дубчак Л.О., к.т.н., доцент, Західноукраїнський національний університет;

Дунець Р.Б., д.т.н., професор, НУ "Львівська політехніка";

Ізонін І.В., д.т.н., доцент, НУ "Львівська політехніка";

Комар М.П., д.т.н, професор, Західноукраїнський національний університет;

Литвиненко В.І., д.н.т, професор, Національний технічний університет України "Київський політехнічний інститут";

Лупенко С.А., д.т.н., професор, Опольський технологічний університет, Польща;

Ляцинський П.Б., доктор філософії з комп'ютерних наук, НУ "Львівська політехніка" ;

Мельникова Н.І., д.т.н., професор, НУ "Львівська політехніка" ;

Мулеса О.Ю., д.т.н., професор, Пряшівський університет, Словаччина

Пелешко Д.Д., д.т.н., професор, Львівський національний університет імені Івана Франка;

Сельський П.Р., д.м.н., професор, Тернопільський національний медичний університет імені І. Я. Горбачевського ;
Субботін С.О., д.т.н., професор, Національний університет «Запорізька політехніка» ;
Теслюк В.М., д.т.н., професор, НУ "Львівська політехніка" ;
Тимченко Л.І., д.т.н., професор, Державний університет інфраструктури та технологій;
Цмоць І.Г., д.т.н., професор, НУ "Львівська політехніка" ;
Якименко І.З., к.т.н., доцент, Західноукраїнський національний університет;
Яровий А.А., д.т.н., професор, Вінницький національний технічний університет ;
Яцків В.В., д.т.н., професор, Західноукраїнський національний університет.

ОРГАНІЗАЦІЙНИЙ КОМІТЕТ КОНФЕРЕНЦІЇ

Склад організаційного комітету:

Мельник Г.М. к.т.н., доцент, Західноукраїнський національний університет
Піцун О.Й. к.т.н., доцент, Західноукраїнський національний університет

Технічна підтримка:

Зінькевич О. В. студентка, Західноукраїнський національний університет
Кіт М. О. студентка, Західноукраїнський національний університет
Лебединський Т.В. студент,Західноукраїнський національний університет;
Кришталь Е.Р., студент, Західноукраїнський національний університет;

Метою конференції є представлення та обговорення наукових і практичних результатів, сприяння активізації творчої і інноваційної діяльності студентів, аспірантів і молодих вчених.
Конференція проводиться із залученням Ради Молодих Вчених та Студентського Наукового Товариства ФКІТ ЗУНУ.

IV Всеукраїнська науково-практична конференція студентів, аспірантів та молодих вчених «Інтелектуальні комп'ютерні системи та мережі» (ІКСМ весна 2026), м. Тернопіль, ЗУНУ, 29 травня 2026 р. Тернопіль, 2026

Адреса:
Вул. О. Теліги, 8, корпус №6, Тернопіль
URL: https://ki.wunu.edu.ua/conference/
e-mail: kistudconference@gmail.com

Тези доповідей подаються за оригіналом рукопису

Білокур В.В.
студент 4 курсу ФКІТ ЗУНУ
Науковий керівник д.т.н., професор Комар М.П., кафедра ІОСУ ЗУНУ

ПРОЄКТУВАННЯ НЕЙРОМЕРЕЖЕВОЇ МОДЕЛІ ВИЯВЛЕННЯ ВТОРГНЕНЬ

Вступ. Сучасні комп'ютерні мережі функціонують в умовах зростання інтенсивності та різноманітності кіберзагроз, зокрема атак на прикладні служби, промислові сегменти, транспортні мережі та канали шифрованого обміну даними. У таких умовах підвищується актуальність систем моніторингу безпеки, здатних не лише фіксувати відомі сигнатури, а й виявляти аномалії та нові сценарії вторгнень на основі аналізу мережевого трафіку. При цьому практичне застосування ускладнюється великими обсягами даних, зміною профілів нормальної поведінки, дисбалансом класів та наявністю зашумлених або неповних ознак, що знижує ефективність традиційних евристичних і статистичних методів.

Перспективним напрямом розв'язання зазначених проблем є використання глибокого навчання та суміжних інтелектуальних підходів, які забезпечують автоматизоване виділення інформативних закономірностей у трафіку та підвищують точність класифікації атак за умови належної підготовки даних. Зокрема, у наукових працях запропоновано моделі на основі рекурентних мереж, механізмів уваги та їхніх гібридних комбінацій для покращення розпізнавання складних часових залежностей у потоках даних [1, 2].

Постановка задачі. Побудова інтелектуальної системи моніторингу мережевої безпеки на основі глибокого аналізу трафіку передбачає формалізацію задачі виявлення вторгнень як задачі класифікації (або детекції аномалій) за сукупністю параметрів мережевих з'єднань. У практичних умовах дані надходять у вигляді потоку подій, тому рішення має забезпечувати обробку як у режимі offline (навчання/валідація на еталонних наборах), так і в режимі online (оперативний моніторинг і фіксація інцидентів). Для відтворюваного оцінювання результатів обрано відкриті набори даних, що широко застосовуються у дослідженнях IDS: CIC-IDS2017 та UNSW-NB15 [3, 4]. Крім того, під час формування та перевірки ознак трафіку доцільним є використання інструментарію аналізу пакетів, зокрема Wireshark [5].

Розглянемо формалізацію задачі. Нехай мережевий трафік представлено як послідовність записів $X = \{x_i\}_{i=1}^{N}$, де кожен запис x_i відповідає потоку або з'єднанню та описується вектором ознак $x_i \in \mathbb{R}^d$. Вектор ознак може включати статистичні, часові та протокольні характеристики (наприклад, кількість пакетів/байтів, тривалість з'єднання, напрямки передачі, прапорці TCP, інтервали між пакетами тощо). Множина класів позначається як $Y = \{0,1,\dots,K\}$, де 0 – нормальний трафік, а $1..K$ – типи атак (за наявності багатокласової розмітки) або узагальнений клас "атака" (у випадку бінарної постановки).

Тоді задача виявлення вторгнень формулюється як побудова функції класифікації

$$f: \mathbb{R}^d \to Y, \tag{1}$$

яка мінімізує помилку класифікації на тестовій вибірці та є стійкою до змін профілю трафіку в часі. Практично це означає, що для кожного нового запису x_t, отриманого в процесі моніторингу, система має обчислити прогноз $\hat{y}_t = f(x_t)$ та рівень впевненості (або ймовірність), після чого ухвалити рішення про наявність інциденту.

Якщо застосовується модель на основі послідовностей (наприклад, BiLSTM/BiGRU), тоді вхід може бути організований як послідовність векторів ознак:

$$s_j = \{x_{j,1}, x_{j,2}, \dots, x_{j,T}\}, \tag{2}$$

де T – довжина вікна спостережень або кількість кроків у часовій агрегації.

Такий підхід узгоджується з моделями, що враховують часові залежності та здатні виділяти інформативні фрагменти через attention [1, 2, 6]. Водночас для системи моніторингу важливо, щоб обрана схема формування послідовностей була однозначною, відтворюваною та придатною до роботи в online-режимі.

Об'єктом дослідження є мережевий трафік комп'ютерних мереж і процеси виникнення та прояву комп'ютерних атак у інформаційно-телекомунікаційних системах.

Предметом дослідження є методи та алгоритми інтелектуального виявлення вторгнень на основі глибокого навчання, а також принципи побудови програмно-алгоритмічної архітектури системи моніторингу безпеки за даними мережевого трафіку.

Мета роботи полягає у проєктуванні нейромережевої моделі виявлення вторгнень для системи моніторингу безпеки комп'ютерної мережі на основі глибокого аналізу трафіку, що забезпечує виявлення вторгнень шляхом навчання та застосування моделей глибокого навчання, а також підтримує збір, підготовку та візуалізацію результатів у процесі моніторингу.

Основний матеріал. Проєктування нейромережевої моделі виявлення вторгнень у межах інтелектуальної системи моніторингу має враховувати дві ключові обставини:

–мережевий трафік характеризується часовою залежністю та контекстом, який не завжди відображається в одиничному записі потоку;

–у практичних умовах важливо забезпечити прийнятний компроміс між точністю детекції та обчислювальними витратами при інференсі.

З огляду на це доцільним є вибір архітектури, здатної моделювати послідовності та акцентувати увагу на найінформативніших фрагментах трафіку.

В якості базового варіант доцільно застосувати гібридну архітектуру BiGRU + MultiHead Attention, оскільки вона поєднує дві важливі властивості (рисунок 1):

– моделювання часових залежностей за рахунок двонапрямної рекурентної мережі (BiGRU/або BiLSTM);

– селективність через механізм уваги, який підсилює внесок найінформативніших кроків послідовності та зменшує вплив шумових фрагментів [1, 2, 6].

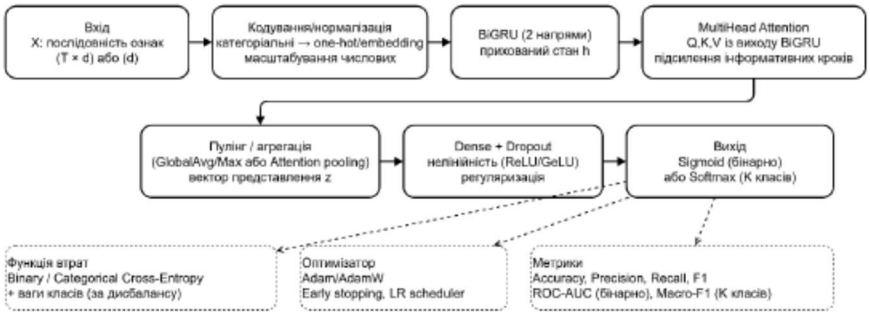

Рисунок 1 – Структурна схема нейромережевої моделі

BiGRU у порівнянні з BiLSTM часто розглядається як більш "економний" варіант з меншою кількістю параметрів при збереженні здатності відстежувати контекст, що є корисним для режиму моніторингу. Використання MultiHead Attention дозволяє моделі одночасно аналізувати кілька "проєкцій" залежностей у послідовності, що підвищує чутливість до різних патернів атак (наприклад, короткі імпульсні прояви та довші поведінкові тренди) [1].

У загальному вигляді модель отримує на вхід або:

– табличний вектор ознак потоку $x \in \mathbb{R}^d$ (спрощена постановка), або

– послідовність $x \in \mathbb{R}^{T \times d}$, сформовану з ковзного вікна/групування потоків (послідовнісна постановка), що є пріоритетною для BiGRU/BiLSTM [2].

Вихід моделі реалізується через Sigmoid (для бінарної детекції «норма/атака») або Softmax (для багатокласової класифікації типів атак), що узгоджується з типовими постановками задач IDS [1, 6].

Для обґрунтування вибору базової архітектури доцільно розглянути альтернативи, які застосовуються в IDS у різних доменах:

1. DBN-подібні підходи (Deep Belief Network). Моделі на основі DBN орієнтовані переважно на табличні ознаки та можуть демонструвати конкурентні результати на класичних наборах даних. У працях, де застосовано DBN у поєднанні з проєкційними методами, підкреслюється потенціал такого підходу для виділення прихованих структур у даних [7]. Водночас DBN зазвичай слабше відображає часовий контекст, якщо послідовність не моделюється явно. Додатковим аргументом на користь часових моделей є результати, де глибока довіра застосовується саме для детекції атак, але потребує ретельного підбору ознак та параметрів [8].

2. Легковагові моделі для доменів із обмеженими ресурсами (MobileViT, CapsNet+SRU). У транспортних мережах (CAN/in-vehicle) важливими є компактність і швидкодія, що обґрунтовує використання MobileViT або інших lightweight-архітектур [9]. Для промислового інтернету (IIoT) запропоновано поєднання CapsNet та SRU, яке орієнтується на виявлення структурних залежностей і роботу з послідовностями [10]. Разом з тим такі архітектури часто є домен-специфічними: модель, оптимізована під CAN або IIoT, може втрачати узагальнювальну здатність при перенесенні в класичний NIDS без адаптації ознак та навчальної вибірки.

3. ELM (Extreme Learning Machine), зокрема несупервізовані варіанти. ELM-підхід привабливий швидким навчанням і відносною простотою, що робить його корисним як базову лінію порівняння [11]. Однак для складних багатокласових сценаріїв і задач із часовими залежностями очікувано кращі результати забезпечують гібридні глибинні моделі, які можуть відображати контекст та нелінійні взаємозв'язки між ознаками [1, 2].

Узагальнюючи, BiGRU + MultiHead Attention є збалансованим рішенням для даної теми: воно враховує часову природу трафіку (на відміну від чисто табличних моделей) і не є жорстко прив'язаним до вузького домену (на відміну від деяких lightweight-рішень), водночас забезпечуючи кращий потенціал точності, ніж спрощені ELM-підходи.

Розглянемо визначення гіперпараметрів, функції втрат, метрик оцінювання.

Функція втрат:

– для бінарної постановки застосовується Binary Cross-Entropy;
– для багатокласової – Categorical Cross-Entropy.

З огляду на типову незбалансованість класів у IDS доцільно передбачити ваги класів у функції втрат, щоб зменшити деградацію recall для міноритарних атак.

В якості стандартного оптимізатора доцільно застосувати Adam/AdamW із планувальником швидкості навчання (LR scheduler) та механізмом ранньої зупинки для обмеження перенавчання.

Метрики оцінювання. Поряд із accuracy доцільно обов'язково аналізувати precision, recall, F1-score, причому у разі багатокласової задачі – macro-F1 (як менш чутливу до дисбалансу). Для бінарної детекції корисним є ROC-AUC, що характеризує якість ранжування рішень при різних порогах.

Також розглянемо механізми адаптивності/оновлення моделі та підвищення продуктивності. У реальному моніторингу модель стикається з дрейфом трафіку (зміна сервісів, поведінки користувачів, появи нових атак). Тому доцільно передбачити контур оновлення, який не порушує стабільності системи:

1. Періодичне перенавчання (наприклад, щотижня/щомісяця) на накопичених верифікованих логах із фіксацією версій моделі та порівнянням метрик. Такий підхід узгоджується з концепціями високопродуктивних адаптивних систем детекції [12].

2. Online/інкрементне оновлення за умов наявності контрольованого механізму валідації, щоб уникнути "зсуву" моделі через шумові або неправильно розмічені дані; це відповідає підходам IDS/IPS із адаптивним навчанням [13].

3. Підвищення продуктивності на рівні моделі: зменшення розмірності ознак (за потреби), обмеження T, застосування більш компактних конфігурацій BiGRU, а також оптимізація інференсу (батчування, прискорені бібліотеки). Практично важливо, щоб прискорення не призводило до різкого падіння recall критичних атак.

Висновки. Отже, в даному дослідженні виконано проєктування нейромережевої моделі виявлення вторгнень для інтелектуальної системи моніторингу мережевої безпеки. В якості базової архітектури обґрунтовано використання гібридної моделі BiGRU + MultiHead Attention, яка поєднує здатність рекурентних мереж відображати часовий контекст трафіку та механізм уваги для виділення найбільш інформативних фрагментів послідовності, що є принципово важливим для задач IDS у реальних умовах.

Список літератури

1. Fan J., Ge L., Zhang H., et al. Intrusion Detection Model Integrating MultiHead Attention and BiGRU. Computer and Digital Engineering. 2023. Vol. 51, no. 1. P. 74–80.
2. Yu J. Design of BiLSTM Algorithm for Network Security Intrusion Detection. Microcomputer Applications. 2024. Vol. 40, no. 11. P. 222–225.
3. Canadian Institute for Cybersecurity (CIC). Intrusion detection evaluation dataset (CIC-IDS2017). URL: https://www.unb.ca/cic/datasets/ids-2017.html (дата звернення: 15.02.2026).
4. UNSW. The UNSW-NB15 Dataset (project page). URL: https://research.unsw.edu.au/projects/unsw-nb15-dataset (дата звернення: 15.02.2026).
5. Wireshark Foundation. Wireshark User's Guide (online documentation). URL: https://www.wireshark.org/docs/ (дата звернення: 15.02.2026).
6. Zhao Y., Hu Y. Research and Optimization of Network Intrusion Detection Methods Based on Deep Learning. Computer Programming Techniques and Maintenance. 2024. No. 11. P. 161–164.
7. Wu Y., Li W., Chen Y. Intrusion Detection Model Based on Deep Belief Network Fusion and Local Preserving Projection. Computer Applications and Software. 2024. Vol. 41, no. 6. P. 62–71.
8. Скумін Т., Головко В., Саченко А., Комар М. Застосування нейронних мереж глибокої довіри для виявлення комп'ютерних атак. Захист інформації і безпека інформаційних систем : зб. тез міжнар. наук.-техн. конф. Львів, 2-3 червня, 2016. С. 162-164.
9. Chen H., Zhang L., Jin H., et al. Vehicle CAN Intrusion Detection Method Based on MobileViT Lightweight Network. Information Security Research. 2024. Vol. 10, no. 5. P. 411–420.
10. Li Q., Liu C., Gao G. Industrial Internet Intrusion Detection Method Based on CapsNet and SRU. Computer Technology and Development. 2024. Vol. 34, no. 7. P. 93–99.
11. Fang J., Leng F. Network Security Intrusion Detection System Based on Deep Learning. Procedia Computer Science. 2025. Vol. 261. P. 1107–1113.
12. Komar M., Kochan V., Dubchak L., Sachenko A., Golovko V., Bezobrazov S., Romanets I. High performance adaptive system for cyber attacks detection. The 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications : Proceedings (Bucharest, Romania, September 21-23, 2017). Bucharest, 2017. Vol. 2. P. 853-858.
13. Vladov S., Vysotska V., Vashchenko S., Bolvinov S., Glubochenko S., Repchonok A., Korniienko M., Nazarkevych M., Herasymchuk R. Neural Network IDS/IPS Intrusion Detection and Prevention System with Adaptive Online Training to Improve Corporate Network Cybersecurity, Evidence Recording, and Interaction with Law Enforcement Agencies. Big Data and Cognitive Computing. 2025. Vol. 9, no. 11. P. 267.