МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВЕТЕРИНАРНОЇ МЕДИЦИНИ ТА БІОТЕХНОЛОГІЙ ІМ. С.З. ҐЖИЦЬКОГО
ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

другого (магістерського) рівня вищої освіти

на тему:

«ЗАСТОСУВАННЯ МЕТОДІВ МАШИННОГО НАВЧАННЯ ДЛЯ АНАЛІЗУ ТРАФІКУ В МЕРЕЖАХ IoT»

Виконав: здобувач групи Іт-62
спеціальності 126 «Інформаційні системи та технології»

Хом'як Н. Ю.
(прізвище та ініціали)

Керівник: Пташник В. В.
(прізвище та ініціали)

ДУБЛЯНИ-2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ПРИРОДОКОРИСТУВАННЯ
ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Другий (магістерський) рівень вищої освіти
Спеціальність 126 «Інформаційні системи та технології»

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____
(підпис)
д.т.н., професор, Тригуба А. М.
(вч. звання, прізвище, ініціали)
“____” _____202 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Хом'яку Назару Юрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *«Застосування методів машинного навчання для аналізу трафіку в мережах IoT»*
керівник роботи *к. т. н., доцент., Пташник В. В.*
(наук.ступінь, вч. звання, прізвище, ініціали)
затверджені наказом Львівського НУП *від 28.02.2025 року № 140/к-с.*

2. Строк подання студентом роботи *8 грудня 2025 року*
3. Вихідні дані до роботи: *датасети доменних імен для формування вибірок легітимного та DGA-трафіку, зокрема перелік доменів Alexa Top Sites і набори DGA-доменів із відкритих джерел 360 Lab та Bambenek Consulting. Описові матеріали щодо алгоритмів генерації доменів і підходів до їх виявлення методами машинного навчання. Науково-технічна література з кібербезпеки й аналізу трафіку IoT, а також документація до використаних програмних засобів і бібліотек (Google Colab, Python, NumPy, Pandas, Scikit-learn, TensorFlow/Keras, Matplotlib/Seaborn) для реалізації, навчання та оцінювання моделей розпізнавання DGA-доменів.*
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
Вступ
1. Теоретичні засади інтернету речей та безпеки IoT-мереж
2. Алгоритми генерації доменів та методи їх розпізнавання в IoT-мережах
3. Програмна реалізація технології аналізу DGA-трафіку у мережах IoT
4. Охорона праці та безпека в надзвичайних ситуаціях
5. Економічна ефективність
Висновки
Список використаних джерел

5. Перелік графічного матеріалу

Графічний матеріал подається у вигляді презентації

6. Консультанти розділів

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата		Відмітка про виконання
		завдання видав	завдання прийняв	
1, 2, 3, 5	*Пташник В. В., к.т.н., доцент*			
4	*Городецький І. М., к.т.н., доцент*			

7. Дата видачі завдання *3 березня 2025 р.*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Відмітка про виконання
1	*Складання інженерної характеристики систем аналізу трафіку в мережах IoT*	*03.03.2025 – 15.04.2025*	
2	*Вибір принципів, методів та алгоритмів генерації сторонніх доменів та методів їх розпізнавання в IoT-мережах*	*16.04.2025 – 31.05.2025*	
3	*Програмна реалізація технології аналізу DGA-трафіку у мережах IoT*	*01.06.2025 – 30.09.2025*	
4	*Розгляд питань з охорони праці та безпеки у надзвичайних ситуаціях*	*01.10.2025 – 20.10.2025*	
5	*Оцінка економічної ефективності прийнятих рішень*	*21.10.2025 – 15.11.2025*	
6	*Завершення оформлення розрахунково-пояснювальної записки та презентаційного матеріалу*	*16.11.2025 – 30.11.2025*	
7	*Завершення роботи в цілому. Підготовка до захисту кваліфікаційної роботи*	*01.06.2025 – 08.12.2025*	

Здобувач _____ *Хом'як Н. Ю.*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Пташник В. В.*
(підпис) (прізвище та ініціали)

УДК 681.521 / 681.518

Застосування методів машинного навчання для аналізу трафіку в мережах IoT. Хом'як Н. Ю. Кафедра інформаційних технологій – Дубляни, Львівський НУВМБ ім. С.З. Гжицького, 2025.

Кваліфікаційна робота: 75 сторінок текстової частини, 27 рисунків, 5 таблиць, 17 джерел літератури, 4 додатки.

Мета кваліфікаційної роботи полягає у аналізі та розробленні підходу із застосуванням методів машинного (зокрема глибинного) навчання для аналізу мережевого трафіку в IoT-мережах і виявлення аномальної/шкідливої активності, а також у моделюванні роботи IoT-мережі та оцінюванні ефективності запропонованого методу.

Об'єктом дослідження є вузли мережі IoT, побудовані на основі принципів ієрархічної взаємодії, та процеси обміну даними між ними.

Предмет дослідження вивчає методи та моделі машинного навчання для аналізу IoT-трафіку (виділення ознак, класифікація, виявлення аномалій) з урахуванням ресурсних обмежень IoT-пристроїв.

У кваліфікаційній роботі проаналізовано багаторівневу архітектуру IoT, досліджено підходи до виявлення аномального трафіку на основі моделей глибокого навчання та систем індикаторів; розроблено метод виявлення аномальної поведінки в сенсорних мережах; змодельовано роботу IoT-мережі та проведено оцінювання запропонованого рішення.

Ключові слова: Інтернет речей (IoT), мережевий трафік, аномальний трафік, виявлення аномалій, кібербезпека IoT, машинне навчання, глибинне навчання, нейронні мережі, DNS, DGA.

ЗМІСТ

ВСТУП

Бездротові сенсорні мережі (БСМ) вважаються однією з найперспективніших технологій для проєктування автоматизованих систем керування, моніторингу та спостереження завдяки таким перевагам, як мобільність, самоорганізація, швидке розгортання та можливість побудови тимчасових мереж у місцях, де традиційну кабельну інфраструктуру важко прокласти або її розгортання є економічно недоцільним. Такі мережі особливо ефективні для роботи на розгалужених об'єктах, у важкодоступних зонах і в умовах, коли конфігурація системи часто змінюється або потребує масштабування без суттєвих витрат на монтаж.

Бездротова сенсорна мережа (БСМ) складається з групи сенсорних пристроїв (СП) та комутаційних вузлів (КВ), з'єднаних бездротовими каналами зв'язку [9]. Джерелом даних у сенсорній мережі є датчики, які фіксують вимірювання параметрів навколишнього середовища, таких як температура, вологість та освітленість, а в прикладних сценаріях також тиск, вібрації, рівень газів чи параметри енергоспоживання. Датчики взаємодіють один з одним і з КВ, наприклад маршрутизаторами або шлюзами, утворюючи розподілену самоорганізовану систему для збору, первинної обробки та передавання отриманих даних до вузлів вищого рівня або серверних компонентів.

Розподіленість у цьому контексті означає просторове розміщення пристроїв БСМ на певній території, що дає змогу покривати великі площі та забезпечувати гнучкість при зміні топології. Самоорганізація означає, що пристрої БСМ можуть самостійно ідентифікувати один одного та знаходити нові шляхи передавання даних у разі виходу з ладу будь-якого вузла, підтримуючи працездатність мережі за рахунок маршрутизації через альтернативні вузли. Така властивість є критичною для систем, де частина елементів може працювати від батарей, періодично переходити у режим сну або бути під впливом перешкод і нестабільного радіоканалу.

Ключовою особливістю БСМ є те, що вони не потребують прямого втручання людини [15, 17], оскільки основні функції виявлення подій, передавання повідомлень та підтримки зв'язності виконуються автоматично на рівні протоколів і логіки вузлів. Методи побудови вимірювальних систем на основі БСМ активно розробляються вченими та інженерами вже понад десять років, що сприяло появі типових архітектур, підходів до енергоощадної взаємодії та стандартних механізмів адресації й маршрутизації. БСМ складаються з простих, компактних пристроїв, призначених для обміну короткими повідомленнями, наприклад пожежних сповіщувачів, сенсорів руху або автономних вимірювачів мікроклімату, однак із розвитком мікроелектроніки такі вузли отримують ширші можливості локальної обробки даних і підтримки додаткових сервісів.

Наступним етапом розвитку БСМ є Інтернет речей (IoT), який формує інтелектуальну систему на основі інтеграції сенсорних вузлів, виконавчих пристроїв, шлюзів та хмарних/периферійних сервісів. «Розумна» функціональність IoT значною мірою залежить від конкретного прикладного завдання, але загалом IoT забезпечує широкий спектр сервісів: контроль параметрів навколишнього середовища, безпеку об'єктів, керування технологічними процесами, моніторинг стану пацієнтів, підтримку «розумних будинків» тощо. Водночас зростання кількості підключених пристроїв і різноманітність протоколів взаємодії підвищують вимоги до надійності передавання, захисту каналів зв'язку та контролю мережевої поведінки вузлів.

Це дослідження спрямоване на аналіз методів виявлення аномального трафіку в IoT з урахуванням обмежених ресурсів пристроїв та користувачів, а також специфіки ієрархічної взаємодії між сенсорними вузлами, шлюзами і сервісами обробки даних. Актуальність такого підходу зумовлена тим, що навіть незначні відхилення в мережевій активності можуть свідчити про компрометацію пристрою, помилки конфігурації або несанкціоновані спроби доступу, а традиційні методи контролю часто є надто ресурсоємними для застосування на периферійних вузлах.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ЗАСАДИ ІНТЕРНЕТУ РЕЧЕЙ ТА БЕЗПЕКИ IoT-МЕРЕЖ

1.1 Архітектура IoT-мережі: сенсорні та виконавчі пристрої, телекомунікаційна ієрархія

Мережа Інтернету речей є динамічною апаратно-програмною екосистемою, у якій сенсорні та виконавчі вузли, канали зв'язку й серверні компоненти спільно забезпечують збір, передавання та обробку даних. Поширення автономних (батарейних) пристроїв робить критично важливим своєчасне виявлення аномального трафіку, що може бути спричинений шкідливим ПЗ та призводить до зайвого енергоспоживання вузлами IoT [2]. Екосистема IoT охоплює сенсори й виконавчі механізми, телекомунікаційну інфраструктуру, обчислювальні/хмарні сервери та програмне забезпечення для протоколів обміну, зберігання й аналізу даних та підтримки алгоритмів прийняття рішень. Вузли IoT зазвичай поділяють на датчики, виконавчі пристрої та вузли з'єднання (агрегатори/хаби, шлюзи, маршрутизатори), які формують цілісну структуру взаємодій.

Типова структура сенсорного пристрою містить один або кілька датчиків, модулі первинної обробки та інтерфейс взаємодії з іншими вузлами. Як показано на рисунку 1.1, сигнали з датчиків перетворюються з аналогової форми в цифрову за допомогою АЦП, після чого обчислювальний модуль виконує фільтрацію, форматування корисних даних і їх запис у пам'ять. Приймач забезпечує обмін із зовнішнім середовищем і зазвичай має буфери приймання/передавання; за потреби до складу інтегрують GPS-модуль. Сенсорні вузли часто живляться від батарей, тож термін їх служби передусім визначається режимами роботи радіомодуля та обсягом обміну даними [8].

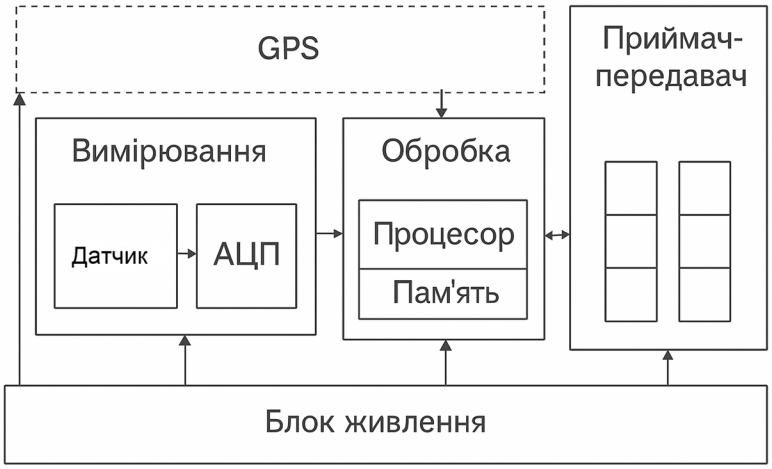

Рисунок 1.1 – Структурна схема сенсорного пристрою Інтернету речей

Мережі моніторингу зазвичай будуються для спостереження за станом об’єктів, екологічними умовами або контрольованими зонами й часто є однорідними за функціями та енергоспоживанням вузлів. Водночас у медичних мережах портативних сенсорних пристроїв (WBAN) поряд із сенсорами використовують виконавчі механізми, здатні впливати на об’єкт вимірювання (рисунок 1.2). Контролери формують команди для виконавчих пристроїв, архітектура яких містить активні елементи взаємодії з середовищем (наприклад, системи дозованого введення ліків). У таких мережах пристрої можуть мати різні задачі та різні енергетичні ресурси, що впливає на організацію обміну даними та політики енергозбереження.

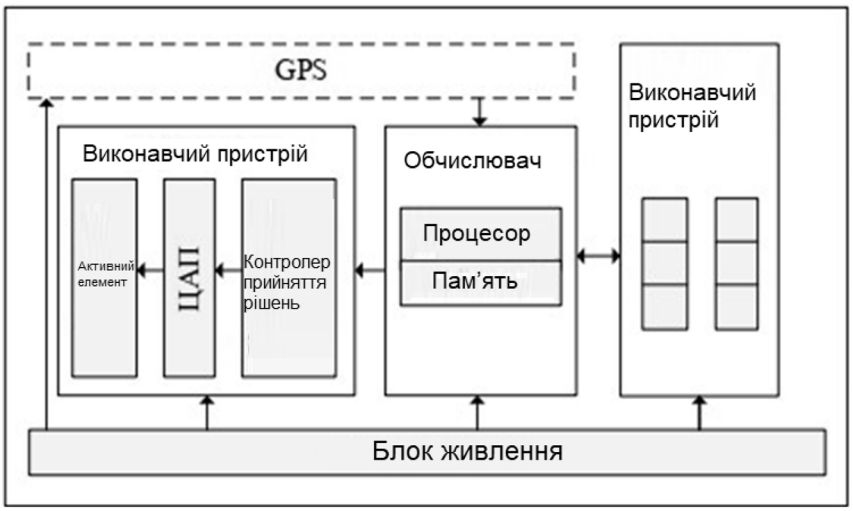

Рисунок 1.2 – Функціональна структура портативного виконавчого пристрою

Організація телекомунікаційної інфраструктури IoT визначається масштабом покриття та складністю задач. На відміну від традиційних мереж Wi-Fi/WiMAX, життєвий цикл сенсорних IoT-мереж значною мірою задається енергоспоживанням під час маршрутизації, передавання/приймання та обробки даних [16]. Тому енергоефективність під час обміну часто досягається зменшенням кількості операцій і застосуванням багатоетапної (ієрархічної) взаємодії, приклад якої наведено на рисунку 1.3.

Рисунок 1.3 – Взаємодія різних типів IoT-пристроїв з фізичними об’єктами

На нижньому рівні сенсорні пристрої об’єднуються в кластери; головний вузол агрегує дані, формує пакети та передає їх до маршрутизаторів, які можуть будувати сітчасту топологію й підтримувати багаторівневу доставку до шлюзу. Шлюз забезпечує вихідний інтерфейс до глобальної мережі (зокрема хмарних обчислень). У межах такої архітектури виділяють типові моделі взаємодії: «пристрій–пристрій» (D2D) усередині сенсорної мережі, «пристрій–сервер»

(D2S) у парадигмі клієнт/сервер та «сервер–сервер» (S2S) для обміну між елементами серверної інфраструктури.

Кластеризація є ключовим механізмом масштабування: кількість кластерів не обмежена, а вибір головного вузла ґрунтується на балансі енергоспоживання. Для цього застосовують алгоритми LEACH, PEGASIS, TEEN тощо [2]. Зокрема, LEACH виконує періодичну (раундову) ротацію головних вузлів і організовує сеанси зв'язку за TDMA; PEGASIS формує ланцюги вузлів із передаванням узагальнених даних до вузлів вищого рівня; TEEN передає дані лише за досягнення порогових значень. У межах раунду головний вузол приймає дані від членів кластера, обробляє та пересилає їх іншим головним вузлам і/або шлюзам, що вимагає підвищених енергоресурсів. Для формування кластера та оцінювання зв'язності може використовуватися інформація про рівень прийнятого сигналу (RSS): вона допомагає орієнтовно оцінити «віддаленість» вузлів і надалі налаштувати параметри мережі та розклад передавання даних

1.2 Протоколи та архітектурні моделі IoT

Для забезпечення зв'язку між IoT-пристроями, серверною інфраструктурою та користувачами застосовують спеціалізовані прикладні протоколи. За сферою взаємодії вузлів виділяють: DDS (D2D), CoAP і MQTT (D2S), XMPP та STOMP (переважно D2S), AMQP (S2S) [2]. Ці протоколи орієнтовані на роботу в режимі, наближеному до реального часу, і підтримують механізми публікації–підписки, що дає змогу масштабувати мережу до тисяч підключених пристроїв.

DDS (Data Distribution Service) використовує модель publish/subscribe для передавання даних, станів і команд між кінцевими вузлами. Дані організовуються за темами (наприклад, температура чи місцезнаходження), а взаємодія відбувається без жорсткої прив'язки «хто кому» (анонімна модель), що

спрощує самоорганізацію та масштабування мережі без необхідності змінювати ПЗ при зміні складу пристроїв або користувачів. CoAP (Constrained Application Protocol) призначений для ресурсно-обмежених пристроїв; він концептуально близький до HTTP, але використовує двійковий формат і UDP, що зменшує службовий трафік. У CoAP розділяють рівень транзакцій (обмін повідомленнями) та рівень запит–відповідь (клієнт/сервер), а сам протокол може поєднуватися з іншими компонентами мережевого стеку. MQTT (Message Queuing Telemetry Transport) застосовується для збирання телеметрії та віддаленого моніторингу, працює поверх TCP і базується на архітектурі «клієнти–брокер» із publish/subscribe, де брокер керує чергами й доставкою повідомлень за топіками. Надійність обміну визначається рівнем QoS: від безпідтверджувальної доставки (QoS 0) до режиму з гарантією одноразової доставки (QoS 2). XMPP є розширюваним протоколом обміну повідомленнями на основі XML (TCP) і частіше застосовується в менших мережах, зокрема в задачах внутрішнього освітлення та клімат-контролю; на практиці для взаємодії «видавець–брокер» найпоширеніші CoAP, MQTT і XMPP. Вибір конкретного протоколу визначається призначенням мережі та умовами експлуатації: MQTT часто використовують для енергомоніторингу, контролю параметрів середовища та систем керування освітленням, тоді як CoAP є типовим для сценаріїв «розумного будинку».

Для узгодження обміну текстовими повідомленнями між різноплатформними компонентами застосовують STOMP (Simple Text Oriented Messaging Protocol). Його перевага полягає в простоті та наявності реалізацій для багатьох мов і брокерів, що полегшує інтеграцію клієнтського та серверного ПЗ. На відміну від MQTT, який переважно орієнтований на взаємодію «пристрої/сервер–брокер», STOMP часто використовують як універсальний «клей» між сервером і брокером. Для взаємодій типу «сервер–сервер» (S2S) поширений AMQP (Advanced Message Queuing Protocol) – асинхронний протокол черг повідомлень на TCP, що забезпечує надійну доставку. Його базові сутності: повідомлення (як одиниця даних), брокер (приймає та маршрутизує

повідомлення) і черга (накопичує дані до моменту отримання абонентом). Маршрутизація може виконуватися за схемами fanout, direct або topic, а клієнти можуть читати дані з кількох черг, що зручно для серверної обробки та інтеграції бізнес-повідомлень.

Паралельно з вибором протоколів важливим аспектом є архітектура IoT. Єдиної еталонної архітектури не існує через різноманіття технологій (хмарні, туманні, периферійні обчислення; дротові й бездротові мережі; локальні й глобальні ІКТ-ресурси). Водночас окремі організації стандартизації (OpenFog Alliance, Консорціум периферійних обчислень, проєкт ЄС H2020 mF2C) пропонують власні референтні моделі, які слугують шаблонами для розроблення систем під конкретні вимоги. Класичною вважають тришарову архітектуру: фізичний рівень (сенсори/виконавчі пристрої), мережевий рівень (шлюзи та глобальна мережа для доставки даних до інфраструктури обробки) і прикладний рівень (сервіси на кшталт «розумного дому», транспорту чи енергетики), який переважно реалізується засобами хмарних обчислень (рисунок 1.4). Така модель забезпечує наскрізну взаємодію, але за зростання обсягів даних і вимог до затримок потребує значних ресурсів хмари та пропускної здатності мережі, а також не оптимізує використання фізичних ресурсів IoT.

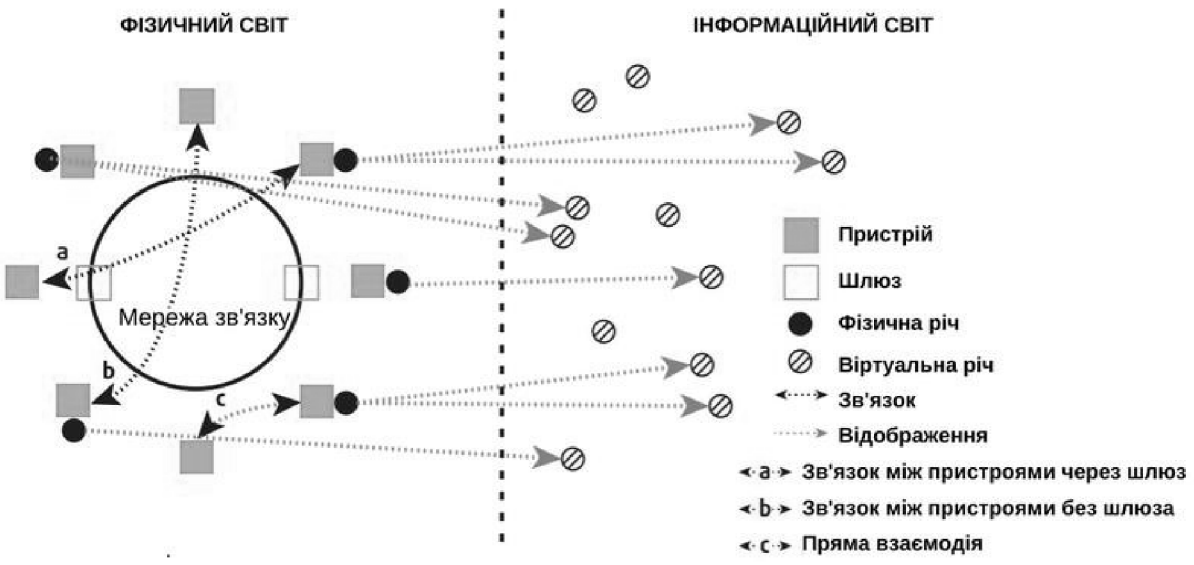

Рисунок 1.4 – Концепція архітектури систем Інтернету речей

Тому поширення набули архітектури з проміжними рівнями, де частина функцій обробки та зберігання переноситься ближче до джерела даних. До них належать платформи проміжного ПЗ, туманні та периферійні обчислення. У хмарній моделі обробка відбувається у віддалених ЦОД, що забезпечує масштабованість і продуктивність, але збільшує затримки. Туманні обчислення реалізують децентралізований підхід: дані обробляються ближче до користувача (наприклад, на локальних вузлах), а сама інфраструктура може включати велику кількість малих вузлів (рисунок 1.5). Периферійні обчислення ще більше зменшують затримку, виконуючи обробку безпосередньо поблизу джерела даних, що забезпечує швидшу реакцію системи.

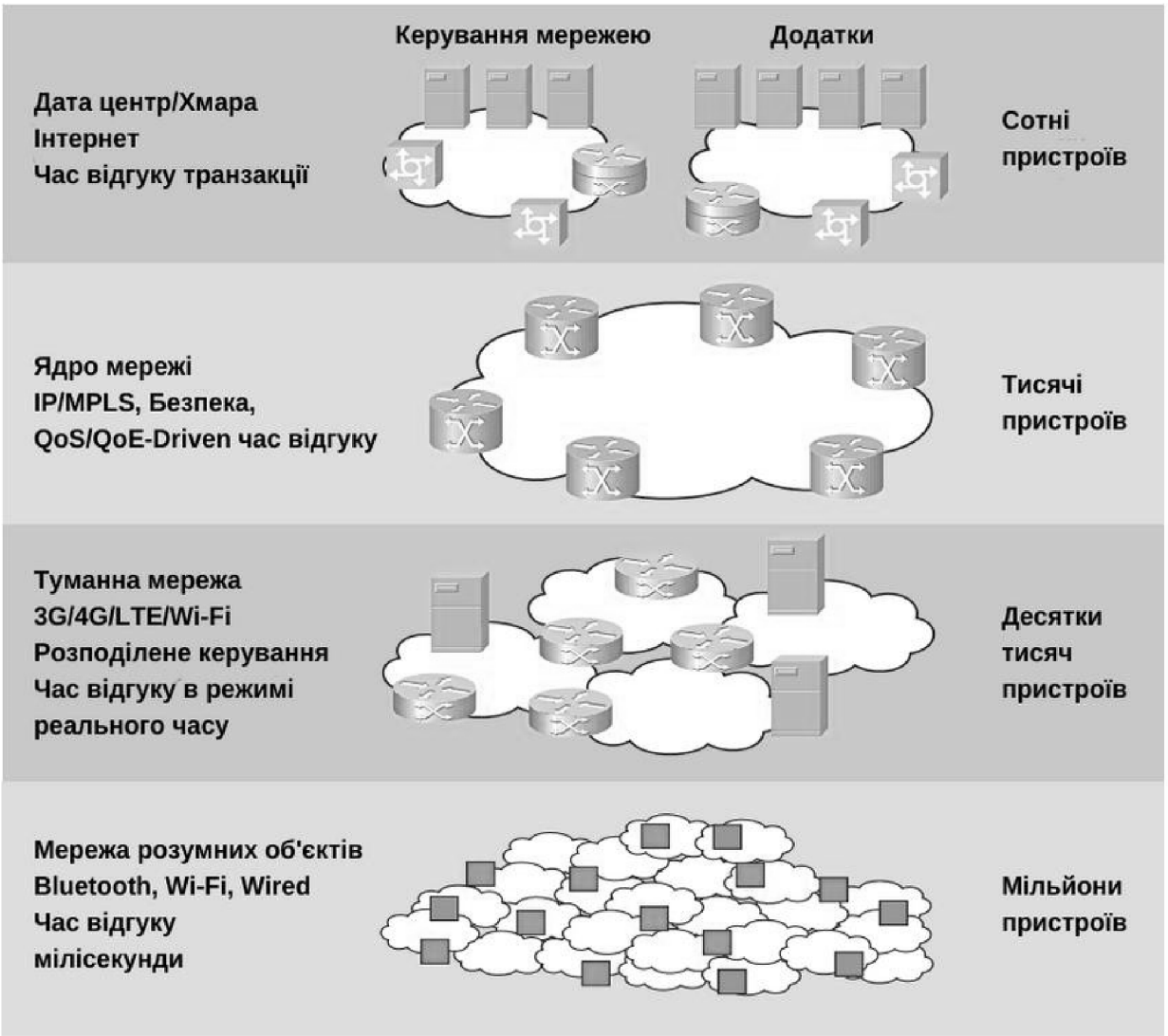

Рисунок 1.5 – Схематична модель архітектури туманних обчислень

1.3 Архітектури IoT на основі проміжних платформ і туманних обчислень

Архітектури IoT із проміжними рівнями застосовують для підвищення ефективності обробки даних і раціонального використання ресурсів мережі. Одним із підходів є використання платформ проміжного програмного забезпечення (рис. 1.6), які виконують роль програмного інтегратора: поєднують різні технології, протоколи та пристрої й забезпечують їх узгоджену роботу в межах єдиної IoT-екосистеми. Така платформа формує «процедурну» інфраструктуру, що спрощує масштабування, керованість і інтеграцію з зовнішніми сервісами.

Функціональність проміжних платформ зазвичай охоплює підключення та нормалізацію даних (ідентифікацію пристроїв, підтримку протоколів, безпечний обмін, зменшення шумів), керування пристроями (моніторинг станів, надсилання команд, оновлення ПЗ), а також обробку подій і реагування через сценарії та процесори правил (наприклад, оповіщення користувача або запуск виконавчих механізмів). Додатково платформи надають засоби візуалізації для графічного подання показників і аналітичні інструменти, включно з алгоритмами прогнозування (зокрема для прогнозного обслуговування в промисловому IoT). Важливою складовою є зовнішні інтерфейси інтеграції, які забезпечують взаємодію з іншими інформаційними системами та постачальниками послуг.

Іншим поширеним підходом є архітектури туманних обчислень, у яких обробка даних частково переноситься на межу мережі, зменшуючи навантаження на центральну хмару та скорочуючи затримки. Туманна модель зберігає переваги сучасної інфраструктури (віртуалізація, контейнеризація, керованість), але розгортає обчислення ближче до джерела даних, що є критичним для задач реального часу. Типово виділяють багаторівневу структуру: хмарний рівень із потужними ЦОД для складних обчислень, рівень

розподілених туманних контролерів із «інтелектом» локальної обробки та рівень мільйонів периферійних IoT-пристроїв. Основна роль туману полягає в прийманні запитів, зборі та аналізі релевантних даних і наданні швидкої відповіді з урахуванням вимог безпеки під час передавання й зберігання інформації.

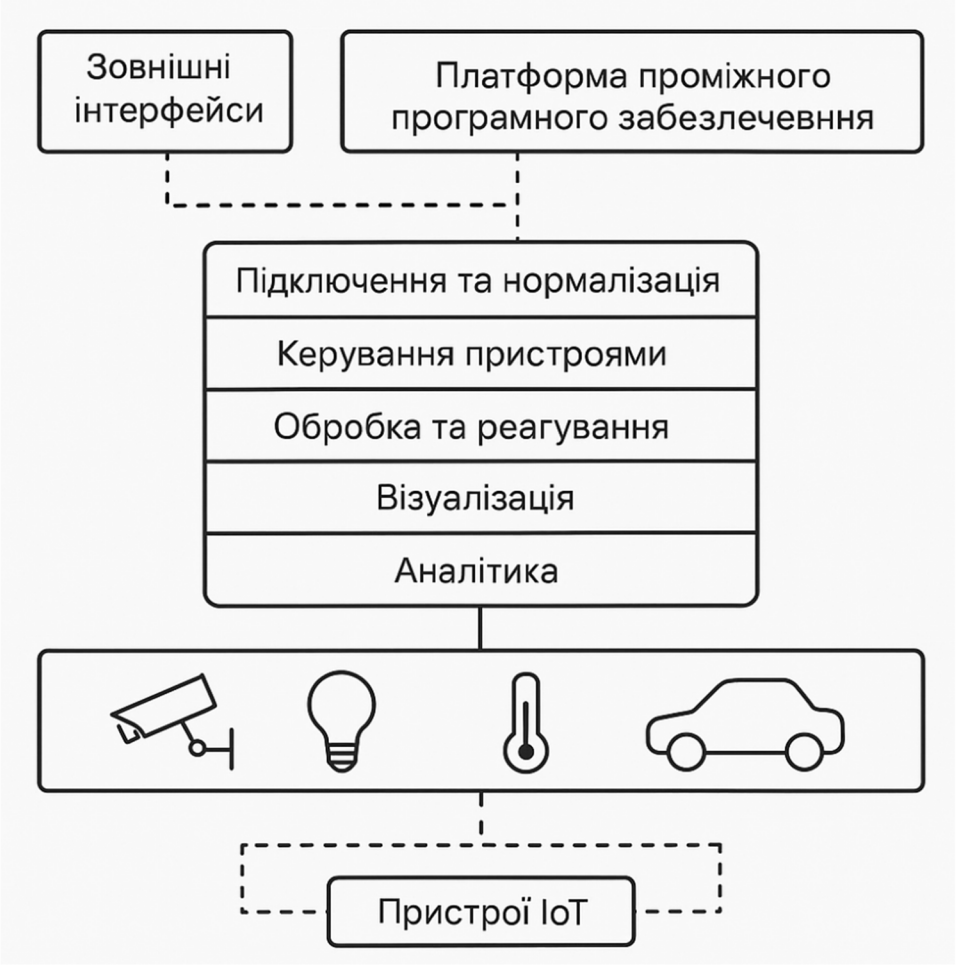

Рисунок 1.6 – Архітектура IoT на основі платформ проміжного програмного забезпечення

Стандартизація туманних обчислень пов’язана з діяльністю консорціуму OpenFog (створеного у 2015 р.), який запропонував OpenFog Reference Architecture: еталонну архітектуру, прийняту як стандарт у 2018 р. Така архітектура орієнтована на обробку великих обсягів даних для IoT та систем штучного інтелекту й ґрунтується на принципах безпеки, масштабованості,

відкритості, автономності, програмованості, надійності, адаптивності та ієрархічності. Вимоги безпеки в тумані залежать від застосування та типу вузлів і охоплюють конфіденційність, анонімність, цілісність, довіру, автентифікацію та верифікацію. Практично це реалізується через ланцюг довіри від апаратних компонентів, керування правами доступу, шифрування, ізоляцію та контроль цілісності контексту на прикордонних вузлах. Масштабованість і відкритість забезпечують можливість змінювати кількість вузлів і сервісів, додавати програми та використовувати різні платформи й протоколи. Автономність і надійність дозволяють туманним вузлам зберігати працездатність за відсутності зовнішніх сервісів, що особливо важливо для промислових та важкодоступних об'єктів. Ієрархічність відображає логічну побудову системи від фізичного рівня (датчики, виконавчі механізми, мобільні пристрої) через рівні доступу й мережі (Wi-Fi, ZigBee, шлюзи, глобальні мережі) до рівня застосунків і сервісів.

1.4 Характеристики IoT-мереж і класифікація загроз

Ключові характеристики мереж Інтернету речей визначають умови їх побудови та впливають на надійність передавання даних і стійкість до атак. Зона відповідальності описує просторовий діапазон, який покривається чутливою зоною одного або кількох датчиків; під полем датчика розуміють простір, що контролюється сенсорною мережею [5, 7]. Гетерогенність відображає різноманітність інформаційно-комунікаційних технологій у межах однієї IoT-мережі: одночасне використання кількох стандартів бездротового зв'язку, наявність вузлів із різними функціями, швидкостями, протоколами та, за потреби, мобільністю. Зв'язність характеризує ймовірність того, що активний сенсорний вузол має шлях до центру обробки/хмари через логічні канали та проміжні ретранслятори; вона залежить від довжини маршруту (кількості передач) і надійності з'єднань між вузлами. Кластеризація є кількісною

метрикою локальної зв'язності: для обраного вузла вона визначається як імовірність того, що два його сусідні вузли є сусідами також між собою.

Окремо виділяють трафік IoT, тобто властивості потоків даних, що генеруються сенсорами. Трафік описують розміром пакета, інтенсивністю надходження та законом розподілу інтервалів між пакетами [15]. Розмір і формат пакета залежать від стандарту й топології, а в IoT він зазвичай менший, ніж у традиційних мережах. Передавані пакети поділяються на керувальні (запити, підтвердження, команди) та інформаційні (корисні дані), а структура пакета включає службові поля (адресація, контроль помилок, послідовність) і корисне навантаження; керувальні пакети можуть не містити поля даних. Інтенсивність трафіку задають частотою генерації пакетів (λ, пак/с), а дослідження показують, що реальний мережевий трафік часто має самоподібну (фрактальну) природу з довгостроковою залежністю та «пульсуючими» піками, які не згладжуються простим усередненням.

Через відкритість бездротового середовища, IoT-мережі є вразливими до широкого спектра загроз [11]. На рисунку 1.7 наведено узагальнену класифікацію атак на IoT. Фізичні атаки пов'язані з прямим впливом на вузли в неконтрольованих умовах і можуть спричиняти незворотні пошкодження. DoS/DDoS спрямовані на виснаження ресурсів через масовий потік «порожніх» запитів; у DDoS джерелом трафіку зазвичай виступає ботнет із заражених вузлів, що призводить до переповнення пам'яті, зростання черг і деградації або паралічу сервісів. Атаки на цілісність і достовірність охоплюють спотворення повідомлень, надсилання неправдивих даних скомпрометованими вузлами та введення підроблених вузлів, які розповсюджують шкідливий код під виглядом нормальних даних; компрометація вузлів комутації (головного вузла кластера, маршрутизатора, шлюзу) особливо небезпечна через системний ефект. Захоплення вузла націлене на отримання конфіденційної інформації (зокрема ключів), а клонування використовує ідентифікаційні дані захопленого вузла для інтеграції клонів у мережу й подальшого контролю її сегментів.

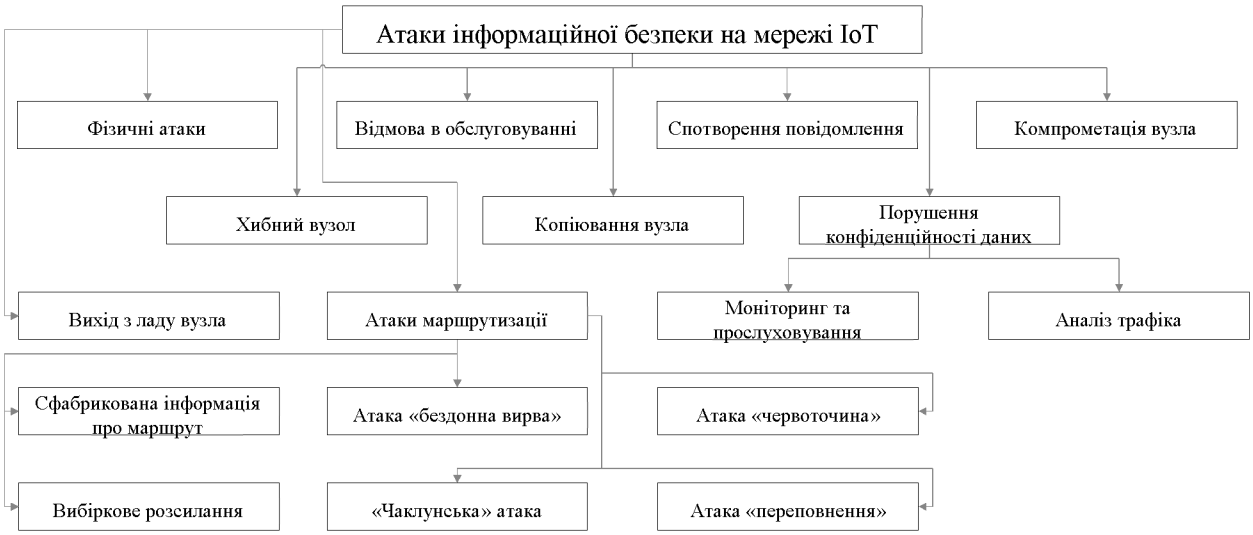

Рисунок 1.7 – Ієрархічна класифікація атак на мережі Інтернету речей

До витоків даних належать підслуховування/перехоплення (особливо критичне для команд керування) та аналіз трафіку, який збирає статистику активності вузлів і дає змогу будувати шаблони зв'язку; при цьому шифрування не гарантує захисту саме від аналізу трафіку. Окремий клас становлять маршрутизаційні атаки: модифікація маршрутної інформації (петлі, неефективні маршрути, відведення трафіку через клоновані вузли), вибіркове видалення пакетів, «бездонна яма», «червоточина», широкомовне «переповнення» непотрібними повідомленнями, атаки «точки зустрічі», дамп пакетів («чорна/сіра діра»), Sybil (множинні фальшиві ідентифікатори), створення циклів і атаки з надмірними службовими повідомленнями, що знижують пропускну здатність і прискорюють енергетичне виснаження мережі.

1.5 Механізми забезпечення безпеки та захисту IoT-мереж

Механізми безпеки в інформаційно-комунікаційних мережах, зокрема в IoT, спрямовані на запобігання атакам і своєчасне виявлення аномального трафіку. Загалом їх розглядають як сукупність рішень високого і низького рівнів

(рис. 1.8), які охоплюють криптографічний захист, безпечну маршрутизацію, протидію DoS/DDoS, захист вузлів та інструменти моніторингу. Для бездротових сенсорних мереж, що мають обмежені ресурси (насамперед енергію), критичною є оптимізація витрат на безпеку без втрати базових властивостей конфіденційності, цілісності та доступності.

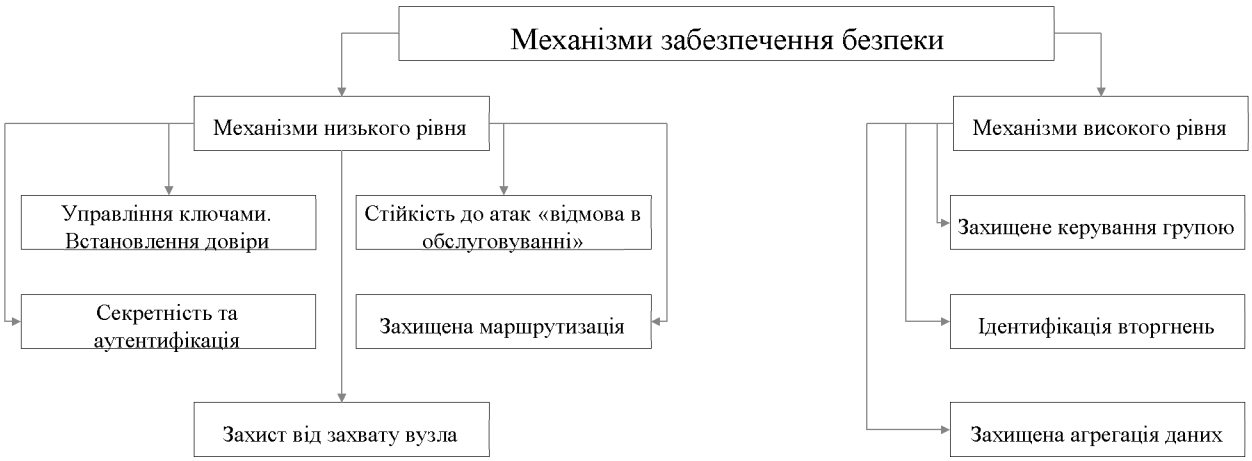

Рисунок 1.8 – Структура механізмів забезпечення безпеки мережі IoT

Ключовим компонентом є керування ключами та довіра. Через енергетичні обмеження найчастіше застосовують симетричні схеми шифрування, встановлюючи ключі між взаємодіючими вузлами, зокрема з головним вузлом кластера. Водночас симетричний підхід має суттєвий ризик: у разі компрометації значної кількості вузлів можливе відновлення пулу ключів і, як наслідок, розшифрування трафіку. Конфіденційність і автентифікація забезпечуються шифруванням та механізмами підтвердження справжності повідомлень; однак у сенсорних мережах застосування шифрування ускладнюється особливостями бездротового середовища та режимами роботи вузлів (зокрема при кластерній передачі даних). Додатково підвищують стійкість мережі протоколи безпечної маршрутизації та сучасні схеми автентифікації, що зменшує ризики підробки або витоку маршрутної інформації [14].

Окремий блок становить захист від DoS/DDoS, які можуть бути спричинені як зловмисними діями, так і збоями/обмеженнями ресурсів. Для протидії перевантаженню радіоканалу можуть застосовуватись методи розширення спектра, що підвищують завадостійкість шляхом використання ширшої смуги частот і ускладнюють навмисне придушення зв'язку. Важливими є також запобігання захопленню вузлів (захищені корпуси, шифрування/приховування даних, хешування, керування групами безпеки) та організація безпечного приєднання нових вузлів у динамічних кластерах: автентифікація при вході, контроль доступу до головного вузла та захищений внутрішньокластерний обмін. Для зниження ризиків підробки даних застосовують безпечну агрегацію: вузли агрегації (головні вузли кластерів) є привабливою ціллю, тому потребують посиленої автентифікації та захищених маршрутів доставки. Додатково використовуються механізми виявлення вторгнень, що забезпечують безперервний моніторинг, реєстрацію інцидентів і оперативне сповіщення користувачів про підозрілу активність. Доцільним є поєднання таких заходів із обмеженням швидкості запитів, пріоритизацією критичного трафіку та адаптивним керуванням потужністю/часом передачі. Регулярне оновлення прошивок, сегментація мережі та резервування ключових компонентів підвищують стійкість системи й скорочують час її відновлення.

Фундаментальним інструментом практичного виявлення аномального трафіку є система виявлення атак (ADS). Вона збирає дані про роботу мережі та аналізує їх для встановлення факту атаки: характеристики пакетів, навантаження вузлів при обробці, споживання пам'яті й обчислювальних ресурсів, швидкодію програм, статистику доступу тощо. Процес виявлення зазвичай поділяють на чотири фази (рис. 1.9), починаючи зі збору індикаторів стану вузлів і мережі. Для цього в систему керування IoT інтегрують програмні агенти SBA (Security Business Analytics): мережеві агенти збирають метрики трафіку контрольованого сегмента, а агенти вузлів – подієві метрики конкретних пристроїв (надіслані/отримані/оброблені пакети тощо). Зібрані метрики аналізуються інструментами SBA на відповідному рівні ієрархії мережі.

Рисунок 1.9 – Етапи виявлення атак у мережі Інтернету речей

Методи ADS зазвичай реалізуються як сигнатурні або поведінкові. Сигнатурний підхід використовує базу відомих шаблонів атак (контекстні ознаки, семантичні вирази або формалізовані моделі) і забезпечує низький рівень хибнопозитивних спрацювань, але погано виявляє нові атаки, відсутні в базі. Поведінкове виявлення ґрунтується на моделі «нормальної» роботи мережі та оцінює відхилення фактичної активності від еталонної; воно здатне виявляти невідомі атаки, проте може давати хибнопозитивні результати при нестандартній, але легітимній активності та потребує якісних моделей нормальної поведінки.

РОЗДІЛ 2
АЛГОРИТМИ ГЕНЕРАЦІЇ ДОМЕНІВ ТА МЕТОДИ ЇХ РОЗПІЗНАВАННЯ В IoT-МЕРЕЖАХ

Алгоритми генерації доменів (Domain Generation Algorithms, DGA) становлять одну з актуальних і складних загроз для сучасних IoT-систем, оскільки вони істотно ускладнюють виявлення та блокування шкідливої активності в мережах, де різноманітні пристрої працюють у напівавтономному режимі. Попри те, що IoT часто асоціюється з мікроконтролерами та обмеженими обчислювальними ресурсами, більшість практичних реалізацій IoT-пристроїв використовують повноцінні мережеві стеки, мають модулі TCP/IP і працюють на прошивках, побудованих поверх Linux, FreeRTOS чи інших операційних систем. Саме на цьому рівні і виникає вразливість: шкідливий код, інтегрований у firmware, отримує змогу генерувати динамічні доменні імена та встановлювати приховані канали зв'язку з командними серверами ботнету.

Механізм роботи DGA полягає у тому, що шкідлива програма автоматично створює сотні або тисячі доменів за певним алгоритмом, який може враховувати дату, приховані ключі, випадкові послідовності або криптографічні функції. Це дозволяє зловмисникам регулярно змінювати адреси командних серверів та уникати фільтрації. Для захисних систем стає надзвичайно складно блокувати всі потенційні домени, оскільки вони постійно змінюються, а заражені пристрої продовжують успішно встановлювати зв'язок з атакувальником. Через це IoT-пристрій, який заражений хоча б один раз, залишається під контролем навіть після блокування попередніх шкідливих доменів. Така поведінка підвищує стійкість ботнетів, зокрема тих, що орієнтовані на використання великої кількості простих і масових IoT-платформ, таких як камери відеоспостереження, домашні маршрутизатори, відеореєстратори, системи розумного дому, комунікаційні модулі в системах «розумного міста» та промислові шлюзи IIoT.

У цьому контексті доцільно узагальнити послідовність взаємодії зараженого IoT-пристрою з командною інфраструктурою ботнету. На рис. 2.1 показано типову схему де DGA-модуль генерує множину потенційних доменів і ініціює масові DNS-запити, після чого пристрій встановлює з'єднання з активним C2-доменом. Такий підхід суттєво знижує ефективність класичного блокування на основі чорних списків, оскільки адреси керування регулярно змінюються.

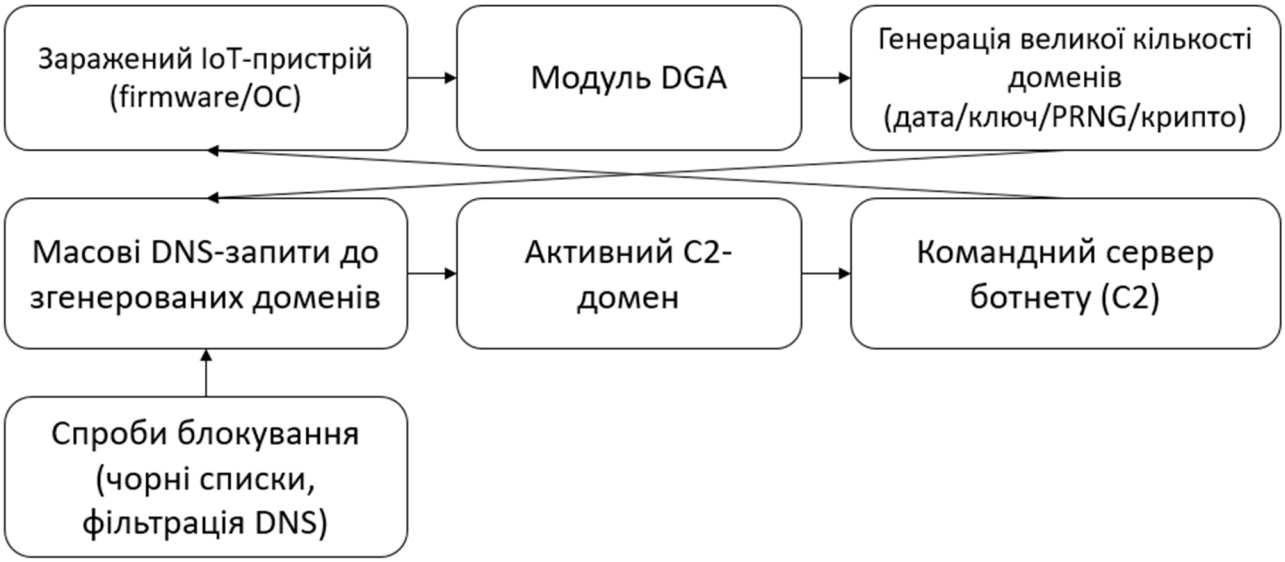

Рисунок 2.1 – Механізм взаємодії IoT-ботнету з командним сервером через DGA

Особливо небезпечним є той факт, що значна частина IoT-пристроїв працює без нагляду, рідко оновлюється та часто використовує стандартні логіни і паролі. Ці умови створюють ідеальне середовище для стійкого функціонування шкідливих DGA-модулів. Заражений IoT-пристрій може виконувати команди зловмисника, здійснювати DDoS-атаки, розповсюджувати шкідливе ПЗ або створювати приховані тунелі для подальшого проникнення в корпоративні мережі. Проблема ускладнюється тим, що IoT-пристрої використовуються не лише в побуті, а й у медичних, транспортних та промислових системах, де їх компрометація може призвести до реальних фізичних наслідків, а не лише до

витоку інформації. Наприклад, підроблені або спотворені дані сенсорів здатні впливати на параметри роботи обладнання, а неконтрольовані команди можуть вивести з ладу цілі системи автоматизації.

DGA є небезпечними для IoT також через те, що кількість згенерованих доменів настільки велика, що класичні методи чорних списків майже втрачають ефективність. Крім того, шкідливий код часто має мінімальний розмір та не потребує значних обчислювальних ресурсів, тому легко працює навіть на мікроконтролерах з обмеженою пам'яттю. Це означає, що хоча DGA не модифікує мікрокод процесора, він може ефективно працювати в прошивці або додатковому модульному ПЗ поверх самого пристрою. На практиці DGA-ботнети вже неодноразово використовувалися у великих кібератаках, зокрема із залученням тисяч камер і маршрутизаторів по всьому світу, що демонструє реальність загрози. Оскільки DGA-активність маскується під легітимні DNS-запити та постійно змінює доменні імена, її виявлення потребує не лише фільтрації доменів, а й поведінкового аналізу мережевого трафіку та кореляції подій на рівні всієї IoT-інфраструктури.

Таким чином, алгоритми генерації доменів становлять серйозну небезпеку для сучасних IoT-систем. Вони дозволяють зловмисникам створювати розподілені та надзвичайно стійкі мережі заражених пристроїв, які важко виявляти, майже неможливо повністю блокувати і які здатні виконувати складні атаки. З огляду на це, питання дослідження DGA, виявлення їхньої активності та розробка методів захисту стають важливими елементами проєктування безпечних IoT-рішень. Для розробників сучасних IoT-систем розуміння механізмів роботи DGA є необхідною умовою створення стійкої архітектури, що передбачає контроль DNS-трафіку, аналіз поведінки пристроїв та регулярне оновлення firmware. Впровадження комплексних заходів протидії DGA є ключовим кроком до того, аби уникнути перетворення IoT-інфраструктури на частину глобальних ботнетів або інструмент масштабних кібератак.

2.1 Обґрунтування вибору мови програмування та середовища розробки

Сьогодні не існує такого поняття, як «найкраща мова машинного навчання». Кожна мова має свої релевантні сценарії застосування. Водночас певні мови програмування справді краще підходять для розв'язання таких завдань, ніж інші. Багато інженерів з машинного навчання обирають мову, орієнтуючись на конкретні бізнес-завдання, з якими вони працюють. Наприклад, більшість фахівців із машинного навчання віддають перевагу Python для задач обробки природної мови (NLP), тоді як для аналізу настроїв (тональності) текстів часто використовують R. Для інших задач машинного навчання, зокрема у сферах безпеки та виявлення загроз часто обирають Java.

Python – найпопулярніша мова програмування високого рівня з динамічною семантикою. Вона проста у використанні та читанні, що зменшує витрати на розробку й обслуговування програм. Python часто вважають однією з найпростіших мов програмування, тому її так широко застосовують. По суті, машинне навчання – це підхід, який дає змогу програмам штучного інтелекту навчатися та автоматично формувати результати без безпосереднього втручання людини.

Завдання експертів з машинного навчання полягає у зборі, впорядкуванні та аналізі даних, а також у використанні цієї інформації для створення алгоритмів штучного інтелекту. Python добре підходить для таких завдань, оскільки є простішою для розуміння порівняно з багатьма іншими мовами, а також чудово справляється з обробкою даних. Однією з головних причин використання Python у машинному навчанні є велика кількість фреймворків і бібліотек, які спрощують процес програмування та скорочують час розробки. NumPy застосовується для наукових обчислень, SciPy – для розширених обчислювальних задач, а Scikit-learn – для аналізу даних і побудови моделей. Ці

бібліотеки можуть працювати в таких фреймворках, як TensorFlow, CNTK та Apache Spark.

Ще однією перевагою Python є її широка підтримка та якісна документація. Існує багато практичних ресурсів для вивчення та застосування Python, а програмісти можуть отримати допомогу й поради на будь-якому етапі розробки.

Для запуску проєктів із машинного та глибокого навчання важливо мати в розпорядженні відповідні бібліотеки, API та середовища розробки. Google активно займається дослідженнями в галузі машинного навчання й надає хмарне середовище розробки під назвою Colaboratory (також відоме як Google Colab), яке є відкритим для широкого загалу. Це браузерна хмарна платформа, що дає змогу створювати складні моделі на великих наборах даних за допомогою Jupyter Notebooks.

Програми машинного та глибокого навчання є обчислювально ресурсоємними й потребують потужних машин, таких як графічні процесори (GPU) або тензорні процесори (TPU), для стабільної роботи. Однак такі ресурси є дорогими. Google Colab надає безкоштовний доступ до GPU і TPU. Сеанс може працювати безперервно до 12 годин.

Платформа надає середовище виконання коду мовою Python. Google Colab попередньо налаштований з найпоширенішими бібліотеками, такими як NumPy, Pandas, Matplotlib та Scikit-learn. Він також підтримує TensorFlow, Keras, PyTorch та Caffe. Крім того, можна встановити будь-які інші бібліотеки Python у Google Colab за допомогою команди !pip install <назва_бібліотеки>. Можна виконувати серію термінальних команд безпосередньо з комірок Google Colab. Перед кожною командою має стояти знак оклику (!), наприклад !ls, !pwd тощо.

Очевидно, що програми машинного та глибокого навчання потребують великих наборів даних для навчання. Ці дані можуть бути як публічними, так і власними, спеціально підготовленими для аналізу та тренування моделей. Google Colab підтримує Google Диск, використовуючи ключ доступу. Набори даних також можна завантажувати з GitHub або Kaggle.

2.2 Моделі й методи формування ознак для розпізнавання DGA-трафіку

Навчальний набір даних може бути сформовано шляхом аналізу та компіляції загальнодоступних і/або подібних наборів даних для ідентифікації DGA-трафіку. Для цієї роботи потрібні реальні дані, що охоплюють різноманітний сучасний і традиційний мережевий трафік, сценарії вторгнень, а також детальну структуровану інформацію про мережеві з'єднання. Використані у роботі вибірки поділятимуться на дві категорії. Перша категорія – це легітимні домени (Legit), взяті зі списку одного мільйона реальних доменів, ранжованих Alexa. Друга категорія – це DGA-домени, що включають домени з колекції 360 Lab (зібрання доменів, згенерованих DGA, яке підтримується та щодня оновлюється китайським постачальником послуг безпеки 360) і списку вихідних доменів DGA від Bambenek. Список вихідних доменів DGA від Bambenek сформовано за допомогою існуючих алгоритмів зворотного інжинірингу для генерування шкідливих доменів, виділених з екземплярів шкідливого програмного забезпечення, що реально циркулює в Інтернеті.

Для підвищення відтворюваності дослідження важливо формалізувати процес підготовки вибірок і маркування доменів. На рис. 2.2 узагальнено пайплайн формування набору даних: збір легітимних доменів і DGA-доменів з визначених джерел, уніфікація та очищення записів, присвоєння міток класу (Legit/DGA) і сімейства (для DGA), а також розподіл даних на навчальну і тестову підвибірки у пропорції 75:25.

Для формування вибірок з необроблених даних їм будуть присвоюватись такі параметри як ім'я домену, тип домену (легітимний домен або DGA-домен) та тип сімейства DGA (для легітимних доменів не використовується). Заплановано проведення класифікації за схемою 75:25, тобто 75 % вихідних даних буде використано для навчання, а решта 25 % – для тестування алгоритму.

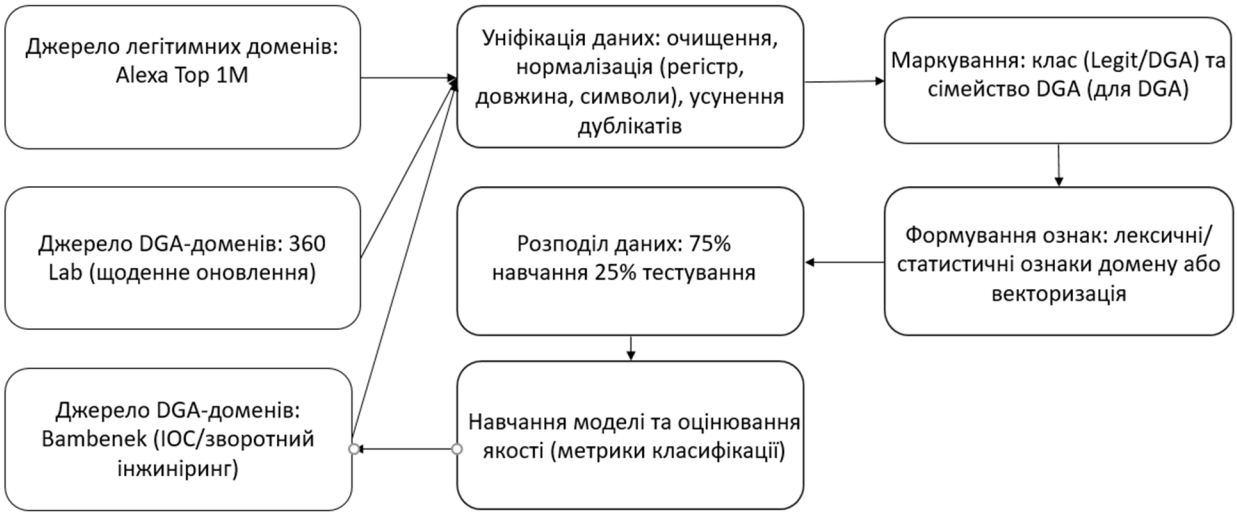

Рисунок 2.2 – Пайплайн формування набору даних для розпізнавання DGA-доменів і DGA-трафіку

2.3 Архітектура та алгоритм глибинного навчання для розпізнавання DGA-трафіку

Фактично, для створення узагальненого класифікатора, здатного розпізнавати трафік DGA в режимі реального часу, доцільно використовувати архітектуру нейронної мережі, яка може аналізувати інформацію, отриману на попередніх кроках, і враховувати її під час опрацювання даних на поточному кроці навчання. Рекурентні нейронні мережі (RNN) (рис. 2.3) мають саме такі властивості. Їхньою головною особливістю є наявність циклів, що дають змогу зберігати й повторно використовувати інформацію, отриману на попередніх кроках обчислень.

Кожен домен розглядається як послідовність символів з фіксованого словника і подається на вхід RNN. Навчання такої мережі здійснюється методом зворотного поширення помилки з метою максимізації ймовірності правильного віднесення домену до відповідної категорії. Водночас на практиці, якщо

«відстань» між минулою та поточною інформацією є надто великою, цей зв'язок втрачається, і мережа вже не може коректно врахувати попередній контекст.

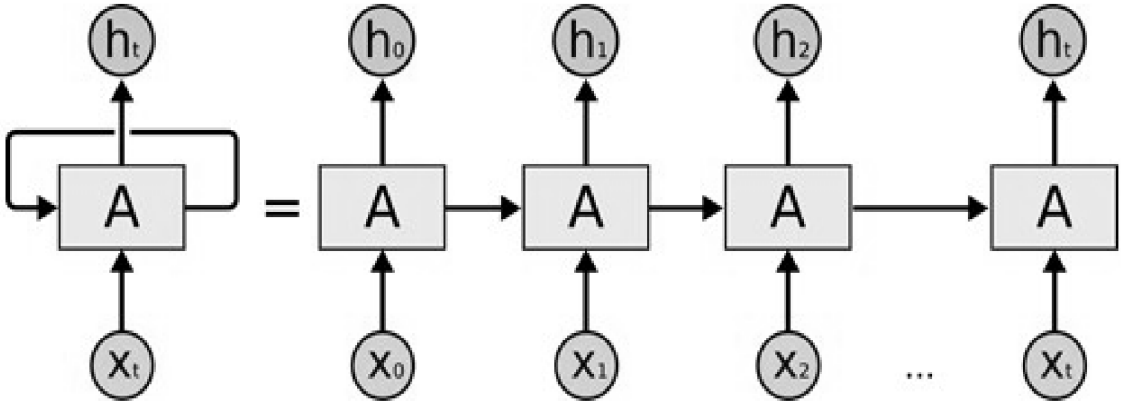

Рисунок 2.3 – Базова структура рекурентної нейронної мережі (RNN)

У 1997 році дослідники Хохрайтер і Шмідхубер запропонували спосіб розв'язання цієї проблеми. У роботі [9] вони представили нову модель – LSTM (рис. 2.4), яку багато років потому було використано, зокрема, й авторами [5]. Наразі існує значна кількість удосконалених варіантів архітектури LSTM, які широко застосовуються для розв'язання різних задач, зокрема розпізнавання мовлення, обробки природної мови тощо. LSTM складається з кількох тісно пов'язаних між собою підмереж, що називаються блоками пам'яті.

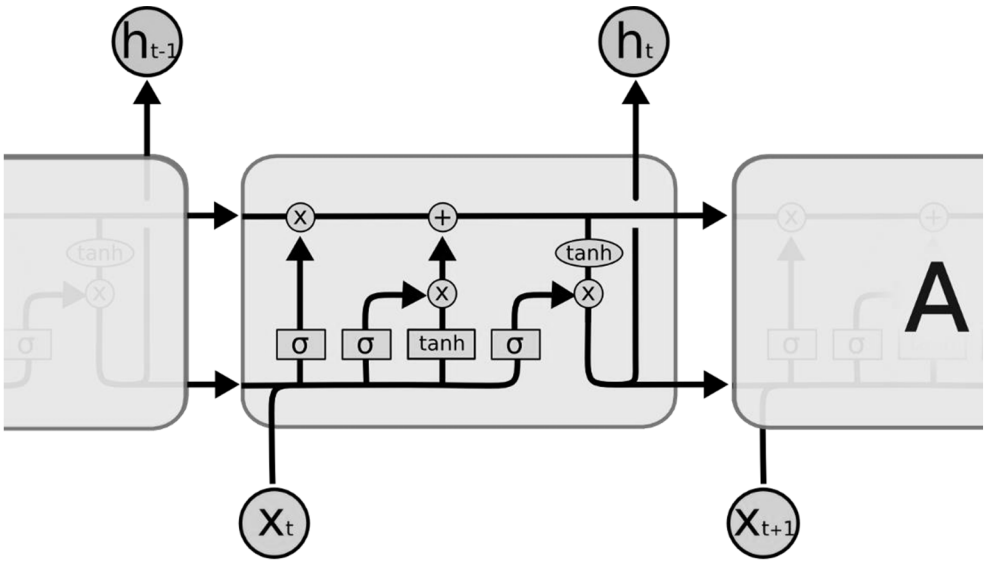

Рисунок 2.4 – Структура LSTM-блоку з чотирма взаємодіючими шарами

На відміну від простої одношарової структури нейронних мереж, LSTM використовує чотири шари, які взаємодіють між собою специфічним чином. Рекурентний блок (GRU) (рис. 2.5) є характерним прикладом LSTM-подібної архітектури, який був застосований у роботах [3] та [6]. GRU – це механізм затворів у рекурентних нейронних мережах, запропонований у 2014 році. Він подібний до LSTM із затвором забуття, але містить менше параметрів завдяки відсутності вихідного затвора. Дослідження показали, що GRU демонструє продуктивність, співставну з LSTM, під час моделювання багатоголосих музичних сигналів і мовлення.

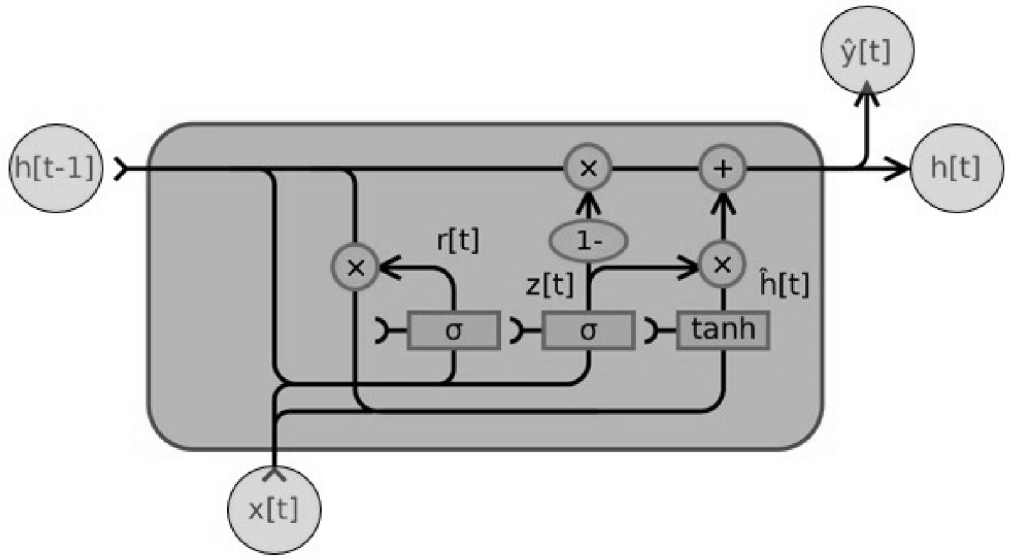

Рисунок 2.5 – Структура рекурентного блоку GRU

Ще один відносно новий підхід до подальшого підвищення якості розпізнавання DGA полягає у використанні часових згорткових мереж (Temporal Convolutional Network, TCN) (рис. 2.6). TCN використовує архітектуру, що застосовує каузальні згортки та дилатацію, завдяки чому вона добре адаптується до послідовних даних із часовими характеристиками та забезпечує великі рецептивні поля. Часовий ряд – це послідовність точок даних, упорядкованих (перелічених або відображених на графіку) у хронологічному порядку. Як правило, під часовим рядом розуміють сукупність значень, зібраних у рівновіддалені моменти часу.

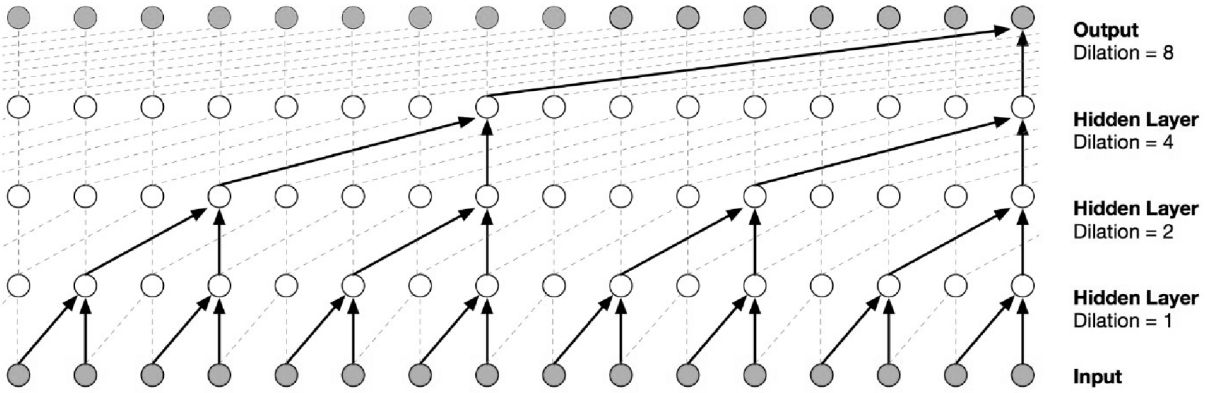

Рисунок 2.6 – Структура стека часової згорткової мережі (TCN)

Відомі приклади використання часових згорткових мереж (TCN) для задачі сегментації дій на основі відео. Загальний процес складається з двох етапів. Спочатку низькорівневі ознаки обчислюють за допомогою типової згорткової нейронної мережі (CNN) для кодування просторово-часової інформації. Потім ці низькорівневі ознаки подають на вхід класифікатору, який використовує типову рекурентну нейронну мережу (RNN) для захоплення високорівневої часової інформації. Основним недоліком такого підходу є необхідність побудови та узгодження роботи двох окремих моделей. Натомість TCN забезпечує уніфікований підхід, який ієрархічно захоплює як низькорівневу, так і високорівневу інформацію.

Донедавна моделювання послідовностей у глибокому навчанні здебільшого асоціювалося з архітектурами рекурентних нейронних мереж, такими як Long Short-Term Memory (LSTM) та Gated Recurrent Unit (GRU). Сьогодні більшість дослідників і практиків вважають, що такий підхід є застарілим, і CNN слід розглядати як одного з провідних кандидатів для моделювання послідовних даних. Згорткові нейронні мережі можуть перевершувати RNN у багатьох завданнях, усуваючи поширені недоліки рекурентних моделей, зокрема проблему вибуху/зникнення градієнтів та обмеження, пов’язані з об’ємом пам’яті. Крім того, використання CNN замість RNN підвищує продуктивність завдяки можливості паралельного обчислення.

У кваліфікаційній роботі буде використано TCN для виявлення потоків DGA. Для цього заплановано проведення навчання мережі TCN на згенерованих наборах даних із подальшим застосуванням для прогнозування тенденцій протягом кількох часових кроків.

2.4 Критерії та методи валідації моделей машинного навчання

Після навчання моделі необхідно оцінити точність класифікатора у прогнозуванні цільових результатів (нормальна чи аномальна мережева активність) на основі вибраних ознак. Для цього можна використовувати різні алгоритми машинного навчання, а також різні навчальні дані та набори гіперпараметрів. Критерії валідації дають змогу порівнювати між собою різні алгоритми навчання та моделі даних.

У кваліфікаційній роботі використано метод валідації на основі позитивної та негативної вибірки з наступними параметрами:

– позитивні приклади (P) – загальна кількість позитивних прикладів у вибірці;

– негативні приклади (N) – загальна кількість негативних прикладів у вибірці;

– істинно позитивні приклади (TP) – позитивні приклади, правильно визначені класифікатором;

– істинно негативні приклади (TN) – негативні приклади, правильно визначені класифікатором;

– хибнопозитивні приклади (FP) – негативні приклади, помилково віднесені класифікатором до позитивних;

– хибнонегативні приклади (FN) – позитивні приклади, помилково віднесені класифікатором до негативних.

Ці значення зведені в матрицю плутанини (таблиця 2.1), яка узагальнює здатність класифікатора коректно розрізняти категорії даних. Відповідно, помилки класифікації поділяються на два типи: хибнонегативні та хибнопозитивні.

Таблиця 2.1 – Матриця плутанини для двокласового класифікатора

	Прогнозований позитивний результат $\hat{y} = 1$	**Прогнозований негативний результат** $\hat{y} = 0$
Позитивний результат $y = 1$	TP	FN
Негативний результат $y = 0$	FP	TN

*$\hat{y}$ – мітка класу визначена за результатом роботи алгоритму розпізнавання, y – правдива мітка класу відповідного об'єкту.

А у табл. 2.2 наведено перелік допоміжних показників для оцінювання роботи класифікатора на етапі прогнозування.

Таблиця 2.2 – Основні метрики оцінювання якості класифікатора

Міра (коротка назва)	**Вираз**	**Тлумачення**
Accuracy (точність)	$\frac{TP + TN}{P + N}$	Частка всіх правильних класифікацій серед загальної кількості прикладів.
Error rate (частка помилок)	$\frac{FP + FN}{P + N}$	Частка всіх неправильних класифікацій серед загальної кількості прикладів.

Продовження Таблиці 2.2

Міра (коротка назва)	**Вираз**	**Тлумачення**
Recall (чутливість)	$\frac{TP}{P}$	Частка справжніх позитивів, які модель правильно виявила, відносно всіх позитивних прикладів.
Specificity (специфіч-ність)	$\frac{TN}{N}$	Частка справжніх негативів, які модель правильно розпізнала, відносно всіх негативних прикладів.
Precision (точність позитивних)	$\frac{TP}{TP + FP}$	Частка правильних позитивних передбачень серед усіх об'єктів, віднесених моделлю до позитивного класу.
Loss (MSE) (сер. квадр. втрата)	$MSE = \frac{1}{N} \sum_{(x,y)\in D} (y - prediction(x))^2$	Середній рівень помилки прогнозу моделі: показує, наскільки передбачення відхиляються від істинних значень; нуль означає ідеальний прогноз.
F1-score (F-міра)	$\frac{2 * Precision * Recall}{Precision + Recall}$	Узагальнена міра якості, що поєднує Precision і Recall через гармонійне середнє та балансує їх внесок.
Fβ-score (зважена F-міра)	$(1 + \beta^2) * \frac{2 * Precision * Recall}{(\beta^2 * Precision) + Recall}$	Узагальнена міра, що поєднує Precision і Recall із ваговим коефіцієнтом β, який задає пріоритет повноті або точності.

Вивчення параметрів функції прогнозування та її тестування на тих самих даних є методологічною помилкою. Модель, яка просто «запам'ятовує» мітки зразків, що були використані під час навчання, може демонструвати дуже високі результати на цих даних, але буде неспроможною коректно передбачати нові, раніше невідомі приклади. Це явище називається перенавчанням. Щоб його уникнути, під час експериментів з машинного навчання частину наявних даних зазвичай виділяють у тестовий набір.

Однак можлива ситуація, коли частина даних використовується лише як навчальний набір і ніколи не потрапляє до тестового. Для розв'язання цієї проблеми ще одну частину набору даних виділяють у так званий валідаційний набір. Навчання моделі відбувається на навчальному наборі, потім виконується оцінка на валідаційному наборі, а остаточне тестування проводять на тестовому наборі після того, як експеримент вважають успішним.

Перехресна перевірка – це метод оцінювання аналітичної моделі та її продуктивності на незалежних даних. Під час перехресної перевірки доступний масив даних поділяється на k частин. Далі модель навчається на k–1 частинах, а решта частини використовується для тестування. Цей процес повторюється k разів так, щоб кожна з k частин хоча б один раз виступала тестовою. У результаті отримуємо оцінку ефективності обраної моделі за умови максимально рівномірного використання всіх доступних даних.

Для коректної оцінки якості створеного класифікатора у кваліфікаційній роботі застосовано перехресну перевірку та обчислено основні метрики, що використовуються в задачах класифікації: точність, повноту та F1-міру.

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ ТЕХНОЛОГІЇ АНАЛІЗУ DGA-ТРАФІКУ У МЕРЕЖАХ IoT

3.1 Побудова навчальних і тестових вибірок та попередня обробка даних

Для навчання моделі використано набори даних, зібраних у файлах формату .csv. В кожному рядку такого файлу значення ознак розділені комами, а кожен рядок відповідає окремому запису даних. Для його перегляду і ручного редагування підходить звичайний текстовий редактор. Дані звичайних (non-DGA) доменів взято з Alexa Top Million. Дані DGA було зібрано з двох незалежних джерел: 360 Lab DGA та Bambenek DGA.

Через це дані Alexa Top Million подано у форматі CSV, тоді як дані DGA – у форматі звичайного тексту. Тому першою задачею була попередня обробка даних, щоб забезпечити однотипне форматування, а також виділити основні ознаки. Після обробки було сформовано три початкові файли даних: nonDGA_domains.csv, 360_dga_domains.csv та bambenek_dga_domains.csv. Статистичний опис вмісту кожного з них наведено на рис. 3.1–3.3 відповідно.

	Вид_DGA	Домен	Тип
count	1000000	1000000	1000000
unique	1	1000000	1
top	Відсутній	aknology.com	Легітимний
freq	1000000	1	1000000

Рисунок 3.1 – Статистичні характеристики набору легітимних доменів nonDGA_domains.csv

	Вид_DGA	Домен	Тип
count	1380024	1380024	1380024
unique	52	1380024	1
top	banjori	pbgmvinskycattederifg.com	DGA
freq	469990	1	1380024

Рисунок 3.2 – Статистичні характеристики набору DGA-доменів 360_dga_domains.csv

	Вид_DGA	Домен	Тип
count	830202	830202	830202
unique	35	828202	1
top	banjori	fycejfsotscqwcwgaibmdnpbkf.ru	DGA
freq	439223	2	830202

Рисунок 3.3 – Статистичні характеристики набору DGA-доменів bambenek_dga_domains.csv

Далі файли з доменами DGA було об’єднано у єдиний файл allDGA_domains.csv та видалено дублікати доменів (рис. 3.4).

	Вид_DGA	Домен	Тип
count	1689083	1689083	1689083
unique	69	1671886	1
top	banjori	ekwxmwiugkeq.biz	DGA
freq	470169	2	1689083

Рисунок 3.4 – Статистичний профіль об’єднаного набору DGA-доменів allDGA_domains.csv

Щоб отримати остаточний набір даних, файл nonDGA_domains.csv, який містить легітимні домени, було об'єднано з файлом, що містить усі DGA-домени. Підсумковий об'єднаний файл all_domains.csv містить 2 689 083 записи (рис. 3.5).

	Вид_DGA	Домен	Тип
count	2689083	2689083	2689083
unique	70	2671886	2
top	Відсутній	wglqssuiwcymao.biz	DGA
freq	1000000	2	1689083

Рисунок 3.5 – Статистичні показники інтегрованого набору доменів all_domains.csv

Інформація про кожен запис у об'єднаному файлі розміщена у трьох стовпцях: DGA_Type, Domain, Type. Приклад перших десяти рядків файлу all_domains.csv наведено на рисунку 3.6.

	Вид_DGA	Домен	Тип
1	Відсутній	google.com	Легітимний
2	Відсутній	youtube.com	Легітимний
3	Відсутній	facebook.com	Легітимний
4	Відсутній	baidu.com	Легітимний
5	Відсутній	wikipedia.org	Легітимний
6	Відсутній	amazon.com	Легітимний
7	Відсутній	qq.com	Легітимний
8	Відсутній	yahoo.com	Легітимний
9	Відсутній	taobao.com	Легітимний
10	Відсутній	tmall.com	Легітимний

Рисунок 3.6 – Фрагмент інтегрованого набору доменів all_domains.csv

Після створення набору даних була проведена їх попередня обробка, щоб зробити дані придатними для навчання нейронної мережі. Із великого списку доменів, що містить загалом 1 мільйон доменних імен, що не належать до DGA, та 2 мільйони DGA-доменних імен з усіма доступними даними, формується словник допустимих символів. Допустимий символ – це унікальний символ, наявний у вихідних даних. Маючи набір допустимих символів, можна визначити максимальну кількість ознак у даних. Це важливо для подання тексту доменного імені як послідовності числових значень. Кількість ознак дорівнює кількості допустимих символів у словнику плюс один спеціальний нульовий символ, який використовується для заповнення доменних імен, коротших за максимальну довжину.

На основі словника кожен символ можна перетворити на елемент послідовності. Нейронні мережі не можуть безпосередньо обробляти текстові символи – алгоритми градієнтного спуску та зворотного поширення працюють із числовими значеннями ознак. Тому для перетворення доменного імені на числове подання кожному символу в доменному імені зіставляється відповідне значення зі словника (рисунок 3.7).

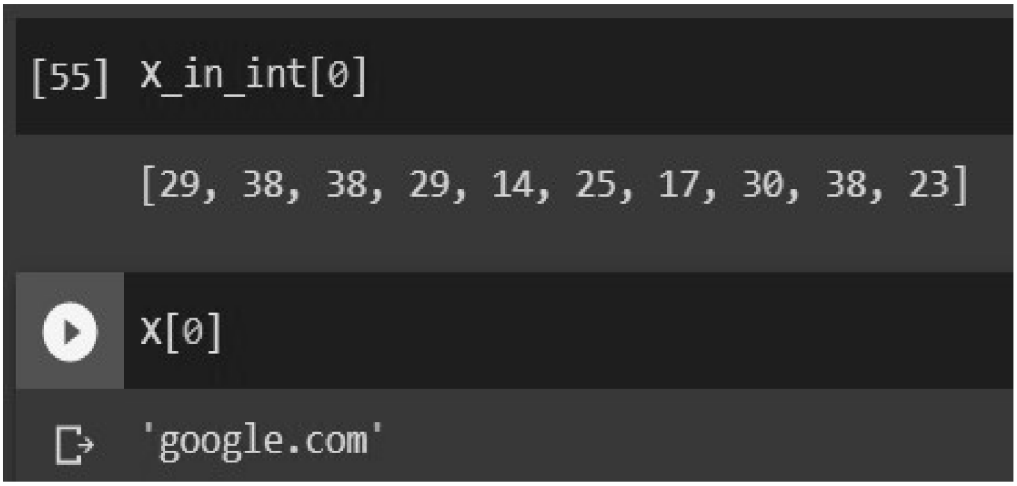

Рисунок 3.7 – Приклад числового кодування символів доменного імені

Оскільки вхідні дані нейронної мережі мають мати однакову форму, потрібно було забезпечити, щоб усі вибірки (домени) були однакової довжини.

Для цього можна обрати довжину, яка охоплює більшість вибірок, або довжину, яка охоплює всі вибірки.

У першому випадку, якщо довжини даних є незбалансованими (тобто більшість доменів мають помірну довжину, а лише невелика частина – дуже довгі), тоді їхній розподіл нагадує логнормальний (рис. 3.8).

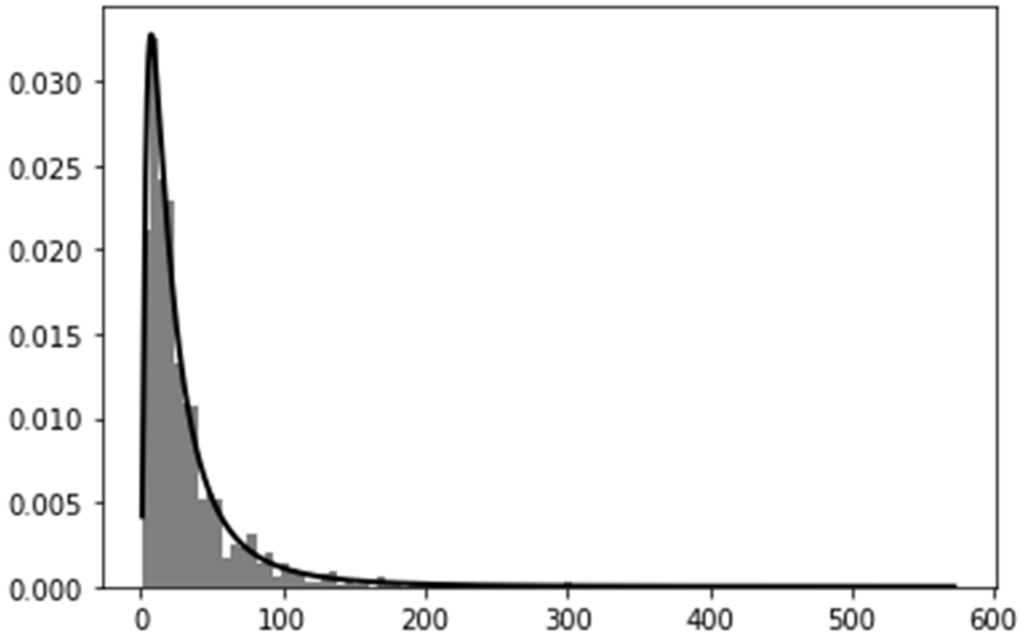

Рисунок 3.8 – Розподіл довжин доменних імен, наближений до логнормального

У цьому випадку, коли вибрана довжина не охоплює всі доменні імена, довжина доповнення може бути значно меншою, ніж у разі вибору максимальної довжини, що покриває всі наявні дані. Оскільки така довжина доповнення охоплює більшу частину даних, можна й надалі коректно їх представляти, не втрачаючи значної кількості інформації (усікатимуться лише надто довгі доменні імена). Це означає, що, обравши таку довжину, ми можемо уникнути зміщення навчальної моделі, яке в іншому разі є неминучим. Водночас менша довжина дозволяє суттєво зменшити обчислювальні витрати.

Максимальна довжина доменного імені в нашій моделі становить 73 символи, що є прийнятним значенням, тому ми вирішили встановити довжину доповнення рівною 73. У результаті отримуємо впорядковану комбінацію

дійсних доменних імен і DGA-доменів, поданих у числовому вигляді. Останнім кроком є створення маски належності до DGA, яка зіставляє кожен елемент цієї впорядкованої послідовності з відповідним бінарним значенням (0 або 1), де 0 позначає неналежність до DGA, а 1 – належність до DGA.

Таким чином сформовано два набори даних (масиви): перший містить числові подання кожного доменного імені, а другий – бінарні значення, що відповідають ознаці належності цих доменів до DGA, тобто елементам першого набору.

3.2 Програмне забезпечення й бібліотеки для реалізації моделі штучного інтелекту

Основною мовою програмування для всіх програм у цьому дипломному проєкті є Python. Python – це високорівнева мова програмування загального призначення, орієнтована на підвищення ефективності розробників і читабельності коду. Вибір Python як основної мови був невипадковим: вона має невисокий поріг входу, стабільно працює на більшості операційних систем і, що особливо важливо, чимало бібліотек, які використовуються для побудови різних моделей машинного навчання, реалізовано саме цією мовою.

Pandas – це відомий пакет для обробки структурованих даних з відкритим вихідним кодом, ліцензований за ліцензією BSD. Ця бібліотека є високопродуктивним і простим у використанні інструментом для роботи зі структурами даних та їх аналізу.

NumPy – один із найпопулярніших пакетів Python. Це ефективний і зручний інструмент для виконання векторних та матричних операцій. Завдяки великій кількості вбудованих функцій він дає змогу уникати надмірного

ускладнення коду. Багато інших бібліотек можуть безпосередньо взаємодіяти з NumPy, що робить його основою для числових обчислень у Python.

Pyplot, Matplotlib та Seaborn є потужними інструментами для візуалізації даних. Методи цих бібліотек дають змогу гнучко налаштовувати різні типи діаграм, гістограм та інших графічних представлень даних, що є одним із найкращих способів наочного подання результатів. Seaborn є розширенням базової бібліотеки Matplotlib і надає зручніші засоби для створення статистичних графіків. Модуль Pyplot – це набір функцій командного стилю, які дозволяють використовувати Matplotlib подібно до середовища MATLAB. Кожна функція Pyplot змінює об'єкт Figure та його області побудови графіків: створює об'єкти Figure, задає області побудови, відображає лінії, додає підписи тощо. Pyplot відстежує поточний стан об'єктів Figure та їхніх областей побудови, а всі операції виконуються над поточним об'єктом.

TensorFlow – це комплексна платформа машинного навчання з відкритим вихідним кодом, розроблена компанією Google. Вона надає гнучку екосистему інструментів, бібліотек і ресурсів, яка дає змогу дослідникам розвивати найсучасніші підходи в машинному навчанні, а розробникам – легко створювати та розгортати прикладні системи на основі цих моделей. Спочатку TensorFlow був розроблений командою Google Brain і використовується Google у таких сервісах, як розпізнавання мовлення, розпізнавання облич на фотографіях, аналіз схожості зображень, фільтрація спаму в Gmail, формування стрічки новин Google News та семантичний переклад. Завдяки вбудованій підтримці розподілених обчислень на кількох процесорах або графічних процесорах системи машинного навчання на основі TensorFlow можуть працювати на відносно звичайному апаратному забезпеченні. TensorFlow добре підходить, зокрема, для автоматичного анотування зображень у системах на кшталт DeepDream. З 26 жовтня 2015 року Google використовує систему RankBrain для підвищення релевантності результатів пошуку, і вона також базується на TensorFlow. TensorFlow підтримує навчання генеративно-змагальних мереж.

Дистрибутив Anaconda забезпечує зручну інтеграцію TensorFlow з Python. Інший приклад застосування – програми на кшталт FakeApp, які дають змогу поєднувати фотографії та відеозображення для створення реалістичних підроблених відео (Deepfakes).

TensorFlow надає готову бібліотеку алгоритмів числових обчислень, реалізованих у вигляді графів потоку даних. Вузли такого графа відповідають математичним операціям або точкам введення/виведення, тоді як ребра представляють багатовимірні масиви даних (тензори), що передаються між вузлами. Вузли можуть бути прив'язані до різних обчислювальних пристроїв і виконуватися асинхронно, обробляючи відповідні тензори паралельно. Така організація обчислень дає змогу моделювати узгоджену роботу вузлів нейронної мережі, подібно до синхронізації активації нейронів у мозку. Окрім цього, TensorFlow забезпечує ефективне навчання та інференс моделей на CPU/GPU/TPU і підтримує автоматичне диференціювання для обчислення градієнтів.

Scikit-learn – це високорівнева бібліотека для машинного навчання. Вона надає широкий набір засобів як для навчання з учителем, так і без учителя. Попри значні обсяг і функціональні можливості, бібліотека має зручний, логічно організований і добре задокументований інтерфейс, що спрощує використання її компонентів. Додатковою перевагою є велика кількість навчальних матеріалів, довідників і прикладів реалізації класичних алгоритмів машинного навчання. Саме завдяки якісній документації та численним прикладам Scikit-learn широко застосовується як досвідченими фахівцями, так і початківцями в галузі машинного навчання. Крім алгоритмів, бібліотека містить інструменти для попередньої обробки даних, відбору ознак та оцінювання моделей, що робить її зручною для побудови повного конвеєра експериментів.

3.3 Результати навчання та оцінювання моделей розпізнавання DGA-трафіку

Для порівняння було використано дві моделі TCN. Перша – регресійна модель, друга – модель класифікації на основі рейтингів. Обидві моделі натреновано з використанням платформи TensorFlow: першу модель навчали протягом двох епох, а другу – протягом десяти епох. Для навчання моделей було використано функцію train_test_split із бібліотеки Scikit-learn, щоб розділити повний набір даних у співвідношенні 75 % : 25 % на навчальну та тестову вибірки. Таке співвідношення є цілком доцільним для великих наборів даних, що містять приблизно 3 мільйони зразків. У процесі навчання моделей і подальшого тестування отримано значення відгуку моделі (рис. 3.9–3.10). А на рис. 3.11–3.12 показано залежність точності моделей від епох навчання.

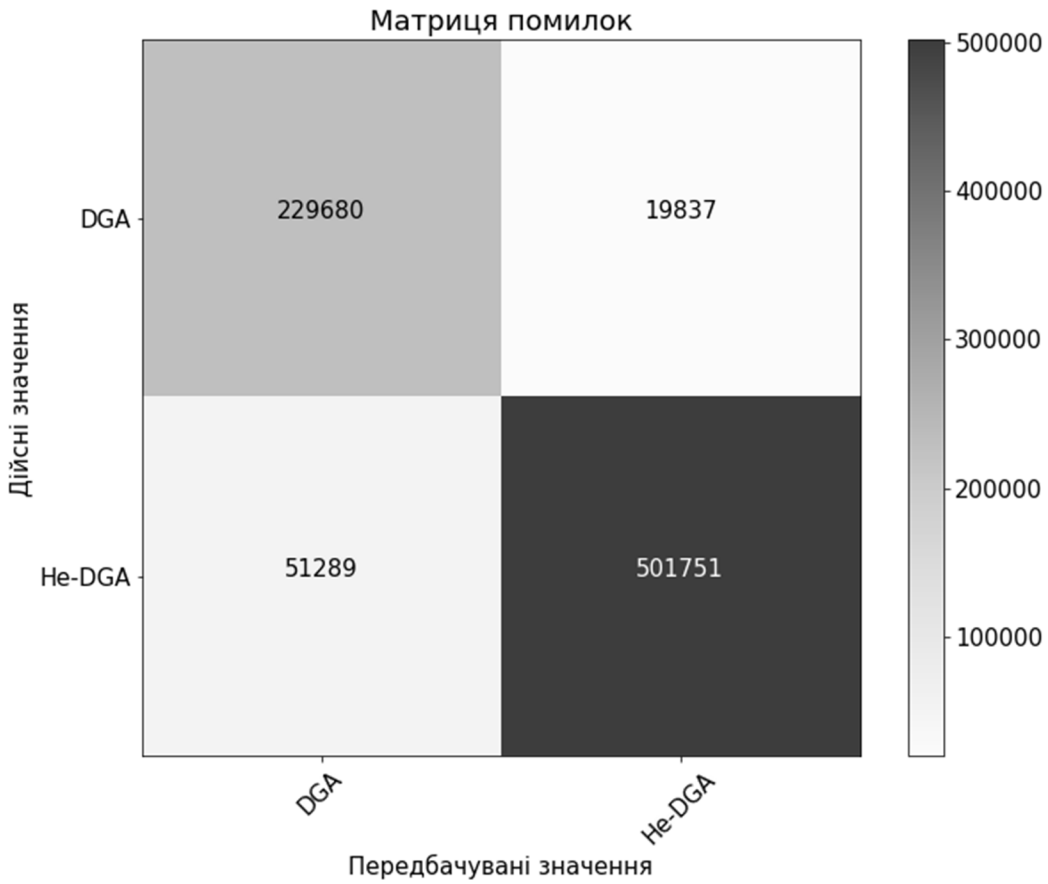

Рисунок 3.9 – Матриця помилок для першої моделі класифікації

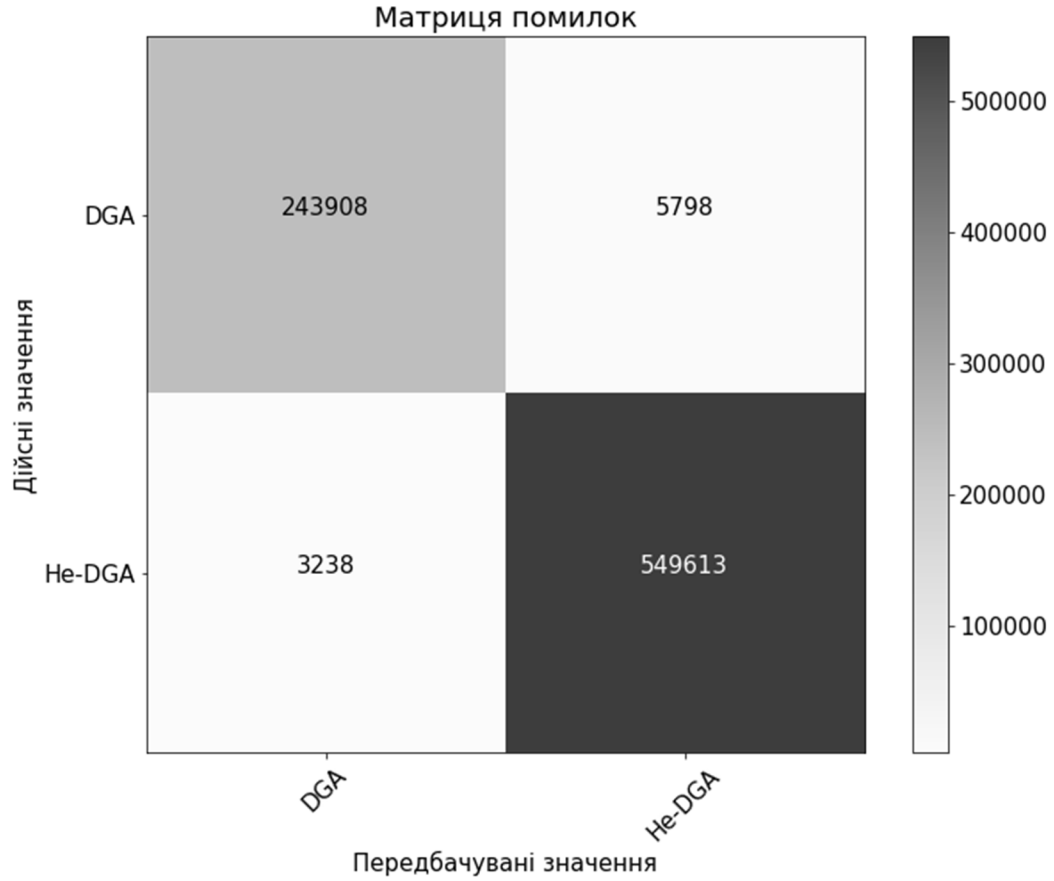

Рисунок 3.10 – Матриця помилок для другої моделі класифікації

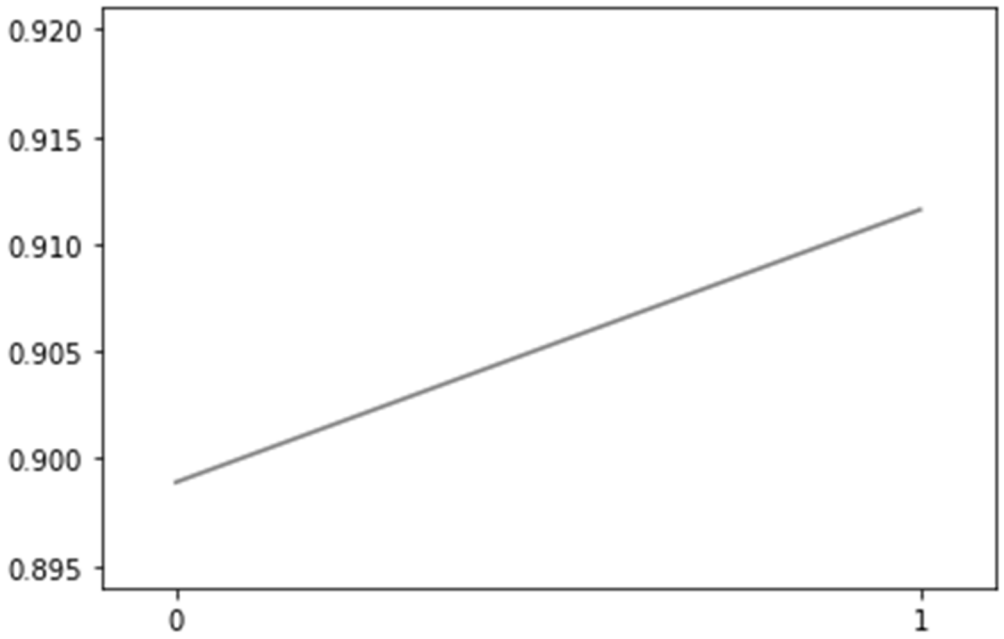

Рисунок 3.11 – Динаміка точності першої моделі залежно від кількості епох навчання

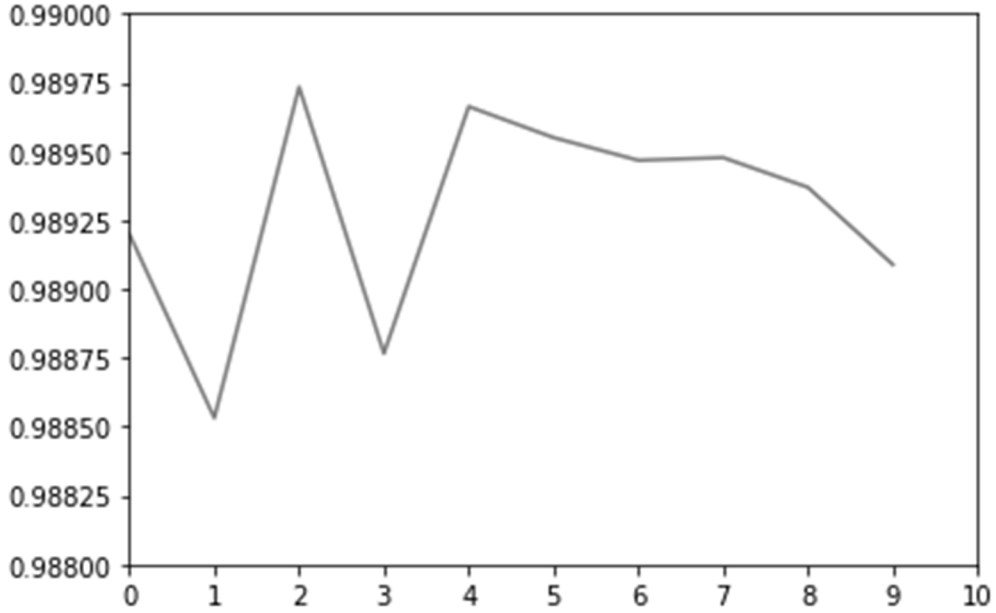

Рисунок 3.12 – Залежність точності другої моделі від кількості епох навчання

На основі цих результатів було розраховано основні метрики для оцінювання роботи класифікатора під час прогнозування, що наведено в таблицях 3.1–3.2.

Таблиця 3.1– Метрики якості для першої моделі класифікації DGA-трафіку

Міра	**Отримане значення**
Точність	0.9114
Частка помилок	0.0886
Чутливість	0.8174
Специфічність	0.9619
Точність	0.9204
Середньоквадратична втрата	0.0657
F-міра	0.86592295
Зважена F-міра	0.93381338

Таблиця 3.2 – Метрики якості для другої моделі класифікації DGA-трафіку

Міра	**Отримане значення**
Точність	0.9887
Частка помилок	0.0113
Чутливість	0.9869
Специфічність	0.9941
Точність	0.9768
Середньоквадратична втрата	0.03616
F-міра	0.9818135
Зважена F-міра	0.99184669

Отримані результати свідчать, що обидві моделі демонструють надзвичайно високу точність на тестовому наборі. Значення метрик істотно перевищують показники відомих моделей машинного навчання, зокрема й тих, що навчені з використанням методів поверхневого та глибокого навчання. Друга модель, призначена для класифікації рецензій на фільми на основі рейтингів IMDb, дещо перевершує модель, що використовується для задачі регресії: вона досягає точності, прецизійності, повноти та F1-міри на рівні 98,87 %, 98,69 %, 98,69 % та 99,18 % відповідно. Дещо нижчі значення метрик для першої моделі можуть бути зумовлені тим, що її було навчено лише протягом двох ітерацій, тоді як другу модель навчали протягом десяти ітерацій. Збільшення кількості ітерацій, імовірно, дасть змогу додатково покращити результати розпізнавання.

РОЗДІЛ 4
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Аналіз умов праці під час розробки систем машинного навчання

Розробка систем машинного навчання передбачає інтенсивну інтелектуальну діяльність фахівців у сфері інформаційних технологій, які працюють переважно за персональними комп'ютерами у приміщеннях офісного типу або лабораторіях. Умови праці в такому середовищі мають свої особливості, що пов'язані з впливом факторів фізичного, психофізіологічного та санітарно-гігієнічного характеру. Основними виробничими факторами є тривале перебування в сидячому положенні, статичне навантаження на хребет і зоровий аналізатор, недостатній рівень природного освітлення, а також вплив електромагнітного випромінювання від моніторів та іншого комп'ютерного обладнання.

Розробники систем машинного навчання часто працюють із великими обсягами даних і складними алгоритмами, що потребує високої концентрації уваги та тривалого перебування перед екраном. Це може призводити до втоми очей, головного болю, підвищеного нервового напруження і зниження продуктивності. Тому важливим аспектом організації робочого процесу є забезпечення оптимального режиму праці та відпочинку. Рекомендується робити короткі перерви через кожні 1–2 години роботи, під час яких працівники можуть виконати вправи для зняття м'язового напруження та відновлення зору.

Суттєве значення має також мікроклімат робочого приміщення. Температура повітря повинна підтримуватись у межах 20–24 °C, відносна вологість – не нижче 40%, а швидкість руху повітря – не більше 0,1 м/с. Важливим є регулярне провітрювання приміщення або використання системи

вентиляції, особливо в умовах тривалого перебування кількох людей у закритому просторі з комп'ютерною технікою.

Організація робочого місця розробника також впливає на безпеку та комфорт праці. Робочий стіл і крісло мають бути ергономічними, з можливістю регулювання висоти та нахилу, щоб забезпечити правильне положення тіла. Монітор слід розташовувати на відстані 50–70 см від очей і на рівні або трохи нижче лінії зору, що дозволяє уникнути перевтоми зору та шийних м'язів.

У процесі розробки систем машинного навчання можуть застосовуватися потужні обчислювальні сервери або кластери, які створюють додаткове теплове навантаження та шум. У таких випадках доцільно розміщувати серверне обладнання в окремих приміщеннях із системою охолодження та контролем доступу. Працівники, які обслуговують це обладнання, повинні бути забезпечені засобами індивідуального захисту у разі можливого контакту з електричними мережами.

Таким чином, умови праці під час розробки систем машинного навчання потребують комплексного підходу до забезпечення безпеки, комфорту та ефективності роботи персоналу. Дотримання санітарно-гігієнічних норм, правильна організація робочого місця, раціональний режим праці та відпочинку є важливими чинниками підтримання високої працездатності й запобігання професійним захворюванням.

4.2 Ідентифікація небезпечних і шкідливих виробничих факторів при роботі з IoT-інфраструктурою

Робота з інфраструктурою Інтернету речей (IoT) охоплює експлуатацію та обслуговування великої кількості електронних пристроїв, сенсорів, контролерів, мережевого обладнання та серверів. У процесі такої діяльності фахівці з інформаційних технологій можуть піддаватися впливу різних небезпечних і

шкідливих виробничих факторів, які мають як фізичний, так і психофізіологічний характер. Одним із основних ризиків є можливість ураження електричним струмом унаслідок пошкодження ізоляції проводів, неправильного підключення або недотримання правил техніки безпеки під час монтажу і налаштування обладнання. Підвищена напруга в електричних колах або коротке замикання можуть стати причиною не лише травмування працівника, але й пожежі або виходу з ладу дорогої техніки.

Іншим суттєвим фактором є перегрів обладнання, що особливо актуально для серверів, маршрутизаторів та інших пристроїв, які працюють безперервно. Недостатнє охолодження або несправність системи вентиляції може призвести до підвищення температури компонентів, що створює небезпеку займання або виходу з ладу критичних вузлів IoT-мережі. У місцях розташування серверів чи комунікаційного обладнання часто спостерігається також підвищений рівень шуму, який негативно впливає на нервову систему працівників, викликає втому та зниження концентрації уваги.

При роботі з бездротовими засобами передачі даних існує вплив електромагнітного випромінювання. Хоча рівень цього випромінювання зазвичай не перевищує допустимих норм, тривале перебування поруч із великою кількістю активних пристроїв може спричиняти негативний вплив на самопочуття працівників. Також можливе накопичення статичної електрики, що при відсутності заземлення створює ризик пошкодження електронних компонентів або короткочасного удару струмом.

Слід зазначити, що під час монтажу сенсорів або контролерів на висоті, наприклад, у промислових приміщеннях чи на відкритих об’єктах, існує небезпека падіння з висоти або травмування внаслідок використання драбин і підйомних механізмів. У зовнішніх умовах до небезпечних факторів додаються погодні умови – сильний вітер, дощ, обледеніння, які можуть ускладнювати виконання робіт та підвищувати ймовірність нещасних випадків.

Крім фізичних ризиків, важливими є й психофізіологічні фактори. Фахівці, які працюють із великими масивами даних, системами моніторингу та аналітики,

часто стикаються з високими розумовими навантаженнями, необхідністю приймати швидкі рішення та контролювати роботу численних пристроїв у реальному часі. Така діяльність може викликати нервове напруження, втому, а при порушенні режиму праці – емоційне виснаження.

Отже, робота з IoT-інфраструктурою супроводжується впливом комплексу небезпечних і шкідливих виробничих факторів, пов'язаних як із фізичними умовами середовища, так і з психологічним навантаженням на персонал. Ідентифікація цих факторів є важливою умовою для розроблення ефективних заходів із забезпечення безпеки праці, підтримання стабільного функціонування системи та запобігання аварійним ситуаціям.

4.3 Організаційно-технічні заходи з охорони праці при дослідженні та впровадженні системи

Організаційно-технічні заходи з охорони праці під час дослідження та впровадження системи машинного навчання для аналізу трафіку в мережах IoT спрямовані на створення безпечних умов роботи для працівників, запобігання виникненню нещасних випадків та зниження впливу шкідливих факторів. Основним принципом є комплексний підхід, який охоплює як технічні рішення, так і організаційні методи управління безпекою. У процесі проведення досліджень, налаштування обладнання та випробувань працівники повинні бути ознайомлені з правилами техніки безпеки, інструкціями з експлуатації апаратури та стандартами охорони праці. Перед початком роботи кожен співробітник має пройти первинний інструктаж на робочому місці, під час якого роз'яснюються потенційні ризики, порядок дій у разі аварійної ситуації та способи надання першої допомоги.

Важливу роль відіграє правильна організація робочих місць, які мають відповідати ергономічним вимогам. Робоче місце фахівця з обробки даних

повинно бути обладнане зручним кріслом із регулюванням висоти, столом відповідної висоти та достатньою кількістю освітлення. Освітлення має бути комбінованим, із використанням природного і штучного світла, причому джерела освітлення не повинні створювати відблиски на екрані монітора. Усі електроприлади, комп'ютери, маршрутизатори та інше обладнання повинні мати справне заземлення, а їх підключення здійснюється через сертифіковані мережеві фільтри або стабілізатори напруги. Це дозволяє зменшити ризик ураження електричним струмом і запобігти пошкодженню техніки під час перепадів напруги.

Під час роботи з серверним і мережевим обладнанням необхідно контролювати параметри мікроклімату, оскільки перевищення допустимої температури може призвести до перегріву пристроїв і створити пожежонебезпечну ситуацію. У приміщеннях, де розташоване технічне обладнання, повинна функціонувати система вентиляції або кондиціонування, що підтримує температуру в межах 20–24 °C та забезпечує достатній рівень вологості. Також важливо передбачити наявність засобів пожежогасіння – вуглекислотних або порошкових вогнегасників, які можна безпечно застосовувати в електронних приміщеннях.

Організаційні заходи також передбачають встановлення режиму праці та відпочинку, який дає змогу запобігти перевтомі працівників. Тривалість безперервної роботи за комп'ютером не повинна перевищувати двох годин, після чого рекомендовано робити короткі перерви для розминки, зняття м'язового напруження та відновлення зору. Працівники, які беруть участь у встановленні IoT-пристроїв або налаштуванні сенсорних вузлів на відкритих об'єктах, повинні бути забезпечені спецодягом, діелектричними рукавичками, касками та іншими засобами індивідуального захисту відповідно до характеру виконуваних робіт.

Додатковим елементом безпечної організації праці є контроль за технічним станом обладнання та проведення профілактичного обслуговування. Регулярні перевірки цілісності кабельних з'єднань, коректності роботи електронних модулів, чистоти повітряних фільтрів і стану мережевих пристроїв дають змогу

запобігти аваріям і знизити ризики пошкодження техніки або травмування персоналу. Важливо також, щоб працівники мали доступ до актуальної документації, технічних паспортів і планів евакуації, а також знали місцезнаходження аптечок і пожежного інвентарю.

Таким чином, організаційно-технічні заходи з охорони праці при дослідженні та впровадженні системи забезпечують не лише фізичну безпеку працівників, але й сприяють підвищенню ефективності роботи. Дотримання цих вимог створює безпечне виробниче середовище, знижує ймовірність виникнення надзвичайних ситуацій та гарантує надійне функціонування IoT-системи.

4.4 Дії персоналу в разі виникнення надзвичайних ситуацій у лабораторії або серверному приміщенні

У процесі розробки, тестування та експлуатації систем машинного навчання й IoT-інфраструктури можуть виникати надзвичайні ситуації техногенного або природного характеру. До найпоширеніших належать пожежі, короткі замикання, перегрів серверного обладнання, витоки диму, витік води внаслідок пошкодження систем кондиціонування, а також аварійні відключення електроенергії. Для запобігання людським жертвам, матеріальним збиткам і збереження цілісності інформаційної системи персонал повинен чітко знати порядок дій у разі настання таких подій.

Якщо виявлено ознаки загоряння, підвищення температури або дим, працівник зобов'язаний негайно припинити роботу, відключити електроживлення шляхом вимкнення головного рубильника чи автоматичного вимикача та повідомити відповідального за пожежну безпеку або безпосереднього керівника. У випадку поширення вогню потрібно негайно викликати службу порятунку за номером 101 і вжити заходів для евакуації персоналу згідно з планом евакуації, який має бути розміщений у видимому місці

в кожному приміщенні. Гасіння вогню дозволяється проводити лише за допомогою вуглекислотних або порошкових вогнегасників, придатних для електроустановок, що перебувають під напругою. Використання води в таких умовах суворо заборонене, оскільки це може призвести до ураження електричним струмом.

У разі короткого замикання або появи запаху гару слід оперативно знеструмити обладнання та від'єднати мережеві пристрої від електроживлення. Після усунення загрози необхідно перевірити стан проводки, блоків живлення та вентиляційної системи. Якщо ситуація пов'язана з витоком води чи охолоджувальної рідини, персонал повинен швидко вимкнути електроживлення, покинути небезпечну зону та повідомити технічну службу або адміністратора приміщення.

У випадку задимлення або витоку газу потрібно негайно відкрити вікна, забезпечити провітрювання приміщення та залишити його до прибуття аварійної служби. Не допускається використання відкритого полум'я, електроприладів або створення іскор, оскільки це може спровокувати вибух чи займання. Якщо в лабораторії або серверному приміщенні сталося аварійне вимкнення електроенергії, персонал повинен зберігати спокій, припинити роботу з обладнанням, дочекатися стабілізації напруги й лише після дозволу відповідального фахівця відновлювати функціонування системи.

Особливу увагу слід приділяти збереженню даних у разі аварійної ситуації. Працівники, які відповідають за IT-інфраструктуру, повинні забезпечити регулярне резервне копіювання інформації, щоб уникнути її втрати під час аварійного вимкнення живлення або пошкодження серверів. Після усунення наслідків надзвичайної ситуації необхідно провести технічне обстеження усіх пристроїв, перевірити стан електричних з'єднань, систем охолодження, вентиляції та протипожежного обладнання. Таким чином, дії персоналу в разі виникнення надзвичайних ситуацій у лабораторії або серверному приміщенні повинні бути чіткими, послідовними і спрямованими передусім на збереження життя та здоров'я людей.

РОЗДІЛ 5
ЕКОНОМІЧНА ЕФЕКТИВНІСТЬ

5.1. Характеристика об’єкта впровадження

Об’єктом упровадження є інтелектуальна система аналізу мережевого трафіку в Інтернеті речей, побудована з використанням методів машинного навчання. Система орієнтована на виявлення аномальної активності, зокрема трафіку, пов’язаного зі шкідливими доменами та алгоритмами генерації доменів (DGA), що застосовуються зловмисниками для організації ботнетів, розповсюдження шкідливого програмного забезпечення та несанкціонованого керування IoT-пристроями. Таким чином, об’єктом є не лише програмний модуль, а й сукупність технічних та організаційних засобів захисту мережевої інфраструктури підприємства.

Передбачається, що система впроваджується в локальній мережі середнього за розміром підприємства, де функціонує до декількох сотень IoT-пристроїв різного призначення: сенсорів моніторингу параметрів середовища, виконавчих контролерів, модулів збору та передавання даних. Типова конфігурація включає близько 200 підключених пристроїв із середньою швидкістю передавання даних на рівні 10 Мбіт/с та обсягом до 100000 подій (пакетів або з’єднань), що аналізуються системою протягом доби. При цьому система повинна забезпечувати оброблення трафіку в квазіреальному часі з тривалістю циклу аналізу одного пакета не більше 0,5 с, що дає змогу своєчасно блокувати підозрілі з’єднання.

З економічної точки зору важливими характеристиками об’єкта впровадження є очікуване зниження кількості успішних кібератак та пов’язаних із ними фінансових втрат, зменшення простоїв обладнання й трудомісткості ручного аналізу журналів подій. Попередні оцінки показують, що застосування

системи здатне зменшити кількість несанкціонованих з'єднань до 80–90 %, а витрати на усунення наслідків інцидентів – орієнтовно на 20–30 %. Окрім того, частина рутинної роботи фахівців IT-відділу автоматизується, що опосередковано підвищує продуктивність праці та дає змогу зосередитися на стратегічних завданнях захисту.

З технічної точки зору система реалізується як програмно-аналітичний модуль, розгорнутий на виділеному сервері або у хмарній інфраструктурі. Мережевий трафік від IoT-пристроїв надходить через шлюзи й маршрутизатори до вузла моніторингу, де здійснюється попередня фільтрація, нормалізація та формування ознак для подальшого аналізу. На цьому етапі також виконується часово-віконна сегментація трафіку та синхронізація подій, що забезпечує коректне порівняння потоків і стабільність сформованих ознак. Алгоритм машинного навчання (нейронна мережа, побудована за архітектурою TCN чи подібною до неї) класифікує потоки даних на «нормальні» та «аномальні», формує попередження для адміністратора та, за потреби, ініціює автоматичне блокування підозрілих сесій. Для зменшення кількості хибних спрацювань передбачено порогову політику реагування та механізм підтвердження інцидентів на основі кореляції з журнальними даними (IDS/Firewall) і контекстом мережевого сегмента.

Умови функціонування об'єкта впровадження характеризуються високою інтенсивністю мережевих обмінів, необхідністю безперервної роботи IoT-пристроїв, обмеженими людськими ресурсами IT-підтримки та зростаючими вимогами до інформаційної безпеки. Важливим є також те, що система інтегрується в уже існуючу мережеву інфраструктуру без суттєвої її перебудови, використовує стандартні протоколи передавання даних і може масштабуватися зі зростанням кількості підключених IoT-пристроїв. Масштабування досягається шляхом горизонтального розподілу обчислень та централізованого керування політиками, що дає змогу зберігати задані показники затримки й пропускної здатності. Це забезпечує прийнятний рівень інвестиційних витрат і створює передумови для досягнення відчутного економічного ефекту.

5.2. Розрахунок витрат на розробку та впровадження проєкту

Витрати на розробку включають оплату праці розробників, придбання необхідного обладнання, витрати на програмне забезпечення, електроенергію та інші супутні ресурси. Умовно приймемо, що розробка проєкту виконувалася командою з двох фахівців: розробника машинного навчання та інженера з кібербезпеки.

Таблиця 5.1 – Розрахунок витрат на розробку проєкту

Стаття витрат	**Одиниця виміру**	**Кількість**	**Вартість, грн**	**Сума, грн**
Заробітна плата розробника ML (20 днів × 8 год × 250 грн/год)	люд.-год	160	250	40000
Заробітна плата інженера з безпеки (20 днів × 8 год × 220 грн/год)	люд.-год	160	220	35200
Витрати на електроенергію (0,3 кВт × 160 год × 7 грн/кВт·год)	кВт·год	48	7	336
Придбання серверного обладнання (1 сервер, 16 ГБ RAM, 512 ГБ SSD)	шт.	1	25 000	25000
Ліцензії/хмарні сервіси (GitHub Copilot, API, GPU)	міс.	3	2 000	6000
Інші витрати (матеріали, амортизація, непередбачені витрати)	–	–	–	4000
Разом:				**110536**

Таким чином загальна вартість розробки системи становить приблизно 110500 грн.

Упровадження системи у виробниче середовище передбачає також одноразові витрати на налаштування мережевих компонентів, інсталяцію програмного забезпечення, тестування та навчання персоналу. Орієнтовно ці витрати становлять 15000 грн. Отже, загальні одноразові витрати на створення та впровадження проєкту становлять 125500 грн.

5.3. Розрахунок експлуатаційних витрат

До експлуатаційних витрат належать витрати на технічне обслуговування, оновлення програмного забезпечення, електроенергію для роботи серверів та заробітну плату персоналу, який супроводжує систему. Для підприємства, що має IoT-інфраструктуру з сотнею сенсорів, середньорічні витрати до впровадження системи становлять:

- усунення наслідків мережевих атак (80000 грн);
- час простою обладнання (40000 грн);
- профілактичні заходи без автоматизованого аналізу (30000 грн).

Тож до впровадження запропонованої системи витрати становлять 150000 грн/рік. Після впровадження системи машинного аналізу трафіку вищезазначена структура зазнає істотних змін. Очікуються наступні корекції:

- витрати на усунення атак знижуються на 70 % (до 24000 грн/рік);
- простої зменшуються на 50 % (до 20000 грн/рік);
- витрати на профілактику частково компенсуються автоматизацією (до 15000 грн/рік).

Водночас додаються нові витрати на технічну підтримку (10000 грн/рік) та електроенергію для серверів (5000 грн/рік).

Тож після впровадження запропонованої системи витрати становитимуть 74000 грн/рік. Отже, щорічне зниження експлуатаційних витрат становить 76000 грн, або приблизно 50 % порівняно з початковим рівнем.

5.4. Економічний ефект від впровадження проєкту

Економічний ефект визначається як різниця між скороченням експлуатаційних витрат та витратами на розробку і впровадження.

Річна економія:

$$E_{річн} = 150000 - 74000 = 76000 \text{ грн.}$$

Витрати на впровадження:

$$B = 125500 \text{ грн.}$$

Термін окупності визначається за формулою:

$$T_{ок} = \frac{B}{E_{річн}} = \frac{125500}{76000} \approx 1{,}65 \text{ року.} \quad (5.1)$$

Отже, проєкт повністю окупається приблизно за 1,7 року, що є економічно доцільним навіть для невеликого підприємства. Після цього система приносить чисту економію, зменшує ризики кібератак і забезпечує підвищення надійності та безперервності роботи IoT-мережі.

Крім фінансової вигоди, впровадження має також енергетичний ефект, адже скорочення кількості збоїв та атак зменшує непродуктивне споживання електроенергії на відновлення роботи пристроїв, що дає орієнтовну економію до 500 кВт·год/рік.

Впровадження системи аналізу трафіку з використанням методів машинного навчання у мережах IoT є економічно доцільним. Розробка окупається менш ніж за два роки, забезпечує скорочення експлуатаційних витрат на 50 %, підвищує продуктивність праці персоналу ІТ-відділу та зменшує ризики фінансових втрат через кіберінциденти. Окрім економічного ефекту, система має позитивний енергетичний вплив і підвищує технологічну безпеку підприємства.

ВИСНОВКИ

У кваліфікаційній роботі розв'язано актуальне прикладне завдання підвищення кібербезпеки мереж Інтернету речей шляхом застосування методів машинного (зокрема глибинного) навчання для аналізу мережевого трафіку та виявлення DGA-активності в DNS-трафіку. Мету роботи досягнуто, а поставлені завдання виконано в повному обсязі.

Проаналізовано сучасні підходи до побудови IoT-інфраструктур, особливості їх багаторівневої архітектури та специфіку загроз, притаманних гетерогенним середовищам з ресурсно-обмеженими вузлами, що обґрунтувало доцільність застосування інтелектуального моніторингу трафіку як інструмента раннього виявлення шкідливої активності.

Систематизовано відомості про алгоритми генерації доменів (DGA) та визначено, що аналіз доменних імен і пов'язаного DNS-трафіку є перспективним напрямом детекції ботнет-активності та інших кіберінцидентів у IoT-середовищі. Визначено вимоги до технології розпізнавання DGA-трафіку та критерії оцінювання якості класифікації, що забезпечило коректне порівняння моделей і відтворюваність експериментів.

Обґрунтовано підхід до організації даних і виконано підготовку вибірок для навчання та тестування, зокрема застосовано розділення даних у пропорції 75% для навчання та 25% для тестування, що дозволяє оцінити узагальнювальну здатність моделей на відкладеній вибірці.

Розроблено та описано пайплайн попереднього оброблення даних і формування наборів для задачі класифікації DGA-доменів, що створює основу для подальшого масштабування рішення та розширення експериментальної бази. Опрацьовано та застосовано нейромережеві підходи, придатні для аналізу послідовнісних структур, зокрема обґрунтовано використання рекурентних

нейронних мереж для виявлення закономірностей у доменних іменах, згенерованих DGA.

Проведено навчання та порівняння двох моделей класифікації, виконано оцінювання їх якості на тестовій вибірці за комплексом метрик (точність, F-міра, чутливість, специфічність).

Встановлено, що перша модель забезпечує точність 0,9114 при F-мірі 0,8659, чутливості 0,8174 і специфічності 0,9619, що підтверджує загальну працездатність підходу, однак вказує на потребу в підвищенні повноти виявлення. Друга модель демонструє суттєво кращі результати: точність 0,9887, F-міру 0,9818, чутливість 0,9869 і специфічність 0,9941, що характеризує високий рівень розпізнавання DGA-активності за мінімальної кількості хибних спрацювань. Що дає змогу зробити висновок про ефективність запропонованого підходу для детекції DGA-трафіку та можливість його використання як складової систем моніторингу кібербезпеки IoT-мереж.

БІБЛІОГРАФІЧНИЙ СПИСОК

1. Бондаренко О. М., Кравчук І. С. Методи машинного навчання в задачах аналізу мережевого трафіку. Наукові праці Одеської національної академії зв'язку. 2021. № 2. С. 45–53.

2. Гнатюк С. О., Іванченко Д. В. Використання алгоритмів класифікації для виявлення аномалій у трафіку IoT. Захист інформації. 2020. Т. 22, № 3. С. 112–121.

3. Зубенко П. П., Литвин А. В. Аналіз мережевого трафіку Інтернету речей із застосуванням глибокого навчання. Сучасні інформаційні системи. 2022. Т. 6, № 1. С. 35–42.

4. Коваль Р. В., Марченко Т. П. Машинне навчання у виявленні кіберзагроз в IoT-системах. Кібербезпека: освіта, наука, техніка. 2021. № 4. С. 58–67.

5. Лисенко О. А. Використання нейронних мереж для аналізу трафіку в інтернеті речей. Вісник Національного університету «Львівська політехніка». 2020. № 11. С. 73–81.

6. Мельник В. І., Ткаченко Ю. С. Методи обробки великих даних у системах моніторингу трафіку IoT. Проблеми інформатизації та управління. 2019. Т. 5, № 2. С. 89–97.

7. Chen Z., He J., Zhang Y. Machine learning-based traffic classification for Internet of Things networks. IEEE Internet of Things Journal. 2021. Vol. 8, No. 7. P. 5648–5659.

8. Doshi R., Apthorpe N., Feamster N. Machine learning DDoS detection for consumer Internet of Things devices. IEEE Security and Privacy Workshops. 2018. P. 29–35.

9. Meidan Y., Bohadana M., Shabtai A., et al. Detection of unauthorized IoT devices using machine learning techniques. Computer Communications. 2019. Vol. 144. P. 84–96.

10. Мокін В.Б., Дратований М.В. Наука про дані: машинне навчання та інтелектуальний аналіз даних [Електронний навчальний посібник]. – Вінниця: ВНТУ, 2024. – 263 с.

11. Болюбаш Н.М. Інтелектуальний аналіз даних: навч. посіб. – Миколаїв: Вид-во ЧНУ ім. Петра Могили, 2023. – 320 с.

12. Гороховатський В.О., Творошенко І.С. Методи інтелектуального аналізу та оброблення даних: навч. посіб. – Харків: ХНУРЕ, 2021. – 92 с.

13. Марченко О.П., Коваленко О.І., Знайдюк В.І. Аналіз методів виявлення аномального трафіку в мережах IoT // Системи управління, навігації та зв'язку. – 2024. – № 1(75). – С. 133–136.

14. Ветлицька Л.С., Треньова Г.С. Виявлення атак у мережах Інтернету речей методами машинного навчання // Сучасний захист інформації. – 2024. – № 1(57). – С. 39–49.

15. Гальчинський А.В. Виявлення вторгнень в мережу Інтернету речей шляхом аналізу наявності аномалій в мережевому потоці трафіку AAA протоколів // Інформатика та математичні методи в моделюванні. – 2022. – Т. 12, № 1–2. – С. 29–39.

16. Гіззатуллін Д.Д. Метод виявлення вторгнень в мережу Інтернету речей за допомогою нейромережі: дипломна робота бакалавра: спец. 172 «Телекомунікації та радіотехніка». – Київ: КПІ ім. Ігоря Сікорського, 2021. – 64 с.

17. Дяченко Д., Кайда В., Левченко А., Міхаль О. Методи функціонування пристроїв IoT з використанням машинного навчання // Системи управління, навігації та зв'язку. – 2024. – Т. 2, № 76. – С. 78–81.

ДОДАТКИ

ДОДАТОК А
ПРОГРАМНА РЕАЛІЗАЦІЯ ПОПЕРЕДНЬОЇ ОБРОБКИ ДОМЕНІВ І ФОРМУВАННЯ НАВЧАЛЬНИХ МАСИВІВ

```
# appendix_a_prepare_dataset.py
# Підготовка датасету DGA/Legit доменів: уніфікація форматів, мітки, кодування, доповнення до 73.

from __future__ import annotations
import re
import os
import json
from dataclasses import dataclass
from typing import Dict, List, Tuple

import numpy as np
import pandas as pd

MAX_LEN = 73  # фіксована довжина (за текстом роботи)
SEED = 42

@dataclass
class Paths:
    alexa_csv: str              # наприклад: "AlexaTop1M.csv" або "alexa.csv"
    dga_360_txt: str            # "360_dga_domains.txt"
    dga_bambenek_txt: str       # "bambenek_dga_domains.txt"
    out_dir: str                # директорія для артефактів

DOMAIN_RE = re.compile(r"^[a-z0-9][a-z0-9\-\.]{0,252}[a-z0-9]$")  # спрощена валідація

def normalize_domain(s: str) -> str | None:
    """Нормалізація доменного імені: нижній регістр, базове очищення."""
    if not isinstance(s, str):
        return None
    s = s.strip().lower()

    # Забираємо схеми та шлях, якщо раптом трапилось у сирих даних
    s = re.sub(r"^https?://", "", s)
    s = s.split("/")[0].strip(".")

    if len(s) == 0 or len(s) > 253:
        return None
    if not DOMAIN_RE.match(s):
        return None
    return s
```

```
def load_alexa_domains(path: str) -> pd.Series:
    """
    Alexa Top Million часто має формат: rank,domain.
    Тут робимо максимально гнучке читання.
    """
    df = pd.read_csv(path, header=None)
    # Якщо 2 колонки: [rank, domain]
    if df.shape[1] >= 2:
        domains = df.iloc[:, 1].astype(str)
    else:
        domains = df.iloc[:, 0].astype(str)
    return domains

def load_txt_domains(path: str) -> pd.Series:
    """Завантаження доменів із текстового файлу (по одному в рядку)."""
    with open(path, "r", encoding="utf-8", errors="ignore") as f:
        lines = [line.strip() for line in f if line.strip()]
    return pd.Series(lines, dtype="string")

def build_vocab(domains: pd.Series) -> Dict[str, int]:
    """
    Символьний словник. 0 — PAD/UNK.
    Типові символи доменів: a-z, 0-9, '-', '.'.
    """
    base_chars = list("abcdefghijklmnopqrstuvwxyz0123456789-.")
    vocab = {ch: i + 1 for i, ch in enumerate(base_chars)}  # 0 лишаємо під PAD/UNK
    return vocab

def encode_domains(domains: pd.Series, vocab: Dict[str, int], max_len: int = MAX_LEN) -
> np.ndarray:
    """
    Перетворення доменів у числові послідовності та доповнення до max_len.
    """
    x = np.zeros((len(domains), max_len), dtype=np.int32)

    for i, d in enumerate(domains):
        # Ключовий момент: символи → індекси словника
        seq = [vocab.get(ch, 0) for ch in d[:max_len]]
        x[i, :len(seq)] = np.array(seq, dtype=np.int32)

    return x

def main(p: Paths) -> None:
    os.makedirs(p.out_dir, exist_ok=True)

    # 1) Завантаження сирих даних
    alexa_raw = load_alexa_domains(p.alexa_csv)
    dga_360_raw = load_txt_domains(p.dga_360_txt)
    dga_bambenek_raw = load_txt_domains(p.dga_bambenek_txt)

    # 2) Нормалізація
    alexa = alexa_raw.map(normalize_domain).dropna().drop_duplicates()
```

```
    dga_360= dga_360_raw.map(normalize_domain).dropna().drop_duplicates()
    dga_bambenek= dga_bambenek_raw.map(normalize_domain).dropna().drop_duplicates()

    # 3) Формування проміжних CSV (як у роботі)
    non_dga_csv = os.path.join(p.out_dir, "nonDGA_domains.csv")
    dga_360_csv = os.path.join(p.out_dir, "360_dga_domains.csv")
    dga_bambenek_csv = os.path.join(p.out_dir, "bambenek_dga_domains.csv")

    pd.DataFrame({"domain": alexa}).to_csv(non_dga_csv, index=False)
    pd.DataFrame({"domain": dga_360}).to_csv(dga_360_csv, index=False)
    pd.DataFrame({"domain": dga_bambenek}).to_csv(dga_bambenek_csv, index=False)

    # 4) Об'єднання та мітки класу: 0 — Legit, 1 — DGA
    legit_df = pd.DataFrame({"domain": alexa, "label": 0})
    dga_df = pd.DataFrame({"domain": pd.concat([dga_360, dga_bambenek],
ignore_index=True), "label": 1})

    full = pd.concat([legit_df, dga_df],
ignore_index=True).drop_duplicates(subset=["domain"])
    full = full.sample(frac=1.0, random_state=SEED).reset_index(drop=True) #
перемішування

    # 5) Кодування у числа + padding до 73
    vocab = build_vocab(full["domain"])
    x = encode_domains(full["domain"], vocab, max_len=MAX_LEN)
    y = full["label"].to_numpy(dtype=np.int32)

    # 6) Збереження артефактів
    np.save(os.path.join(p.out_dir, "X_domains.npy"), x)
    np.save(os.path.join(p.out_dir, "y_labels.npy"), y)
    with open(os.path.join(p.out_dir, "vocab.json"), "w", encoding="utf-8") as f:
        json.dump(vocab, f, ensure_ascii=False, indent=2)

    full.to_csv(os.path.join(p.out_dir, "dataset_full.csv"), index=False)

    print("Done.")
    print(f"Saved: {p.out_dir}")
    print(f"X shape: {x.shape}, y shape: {y.shape}, vocab size: {len(vocab)}")

if __name__ == "__main__":
    # Приклад локального запуску (шляхи адаптуйте під свою структуру)
    paths = Paths(
        alexa_csv="AlexaTop1M.csv",
        dga_360_txt="360_dga_domains.txt",
        dga_bambenek_txt="bambenek_dga_domains.txt",
        out_dir="artifacts"
    )
    main(paths)
```

ДОДАТОК Б
КОД НАВЧАННЯ МОДЕЛЕЙ TCN ДЛЯ РОЗПІЗНАВАННЯ DGA-ДОМЕНІВ

```
# appendix_b_train_tcn.py
# Навчання двох моделей TCN (регресія та класифікація) для розпізнавання DGA-доменів.

from __future__ import annotations
import json
import numpy as np

from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers

SEED = 42
MAX_LEN = 73

def set_seed(seed: int = SEED) -> None:
    tf.keras.utils.set_random_seed(seed)
    np.random.seed(seed)

class TCNResidualBlock(layers.Layer):
    """TCN-блок із каузальною згорткою та дилатацією (ключова ідея TCN)."""

    def __init__(self, filters: int, kernel_size: int, dilation_rate: int, dropout: float = 0.1):
        super().__init__()
        self.conv1 = layers.Conv1D(
            filters=filters,
            kernel_size=kernel_size,
            padding="causal",            # каузальність
            dilation_rate=dilation_rate,   # дилатація
            activation="relu"
        )
        self.dropout = layers.Dropout(dropout)
        self.conv2 = layers.Conv1D(
            filters=filters,
            kernel_size=kernel_size,
            padding="causal",
            dilation_rate=dilation_rate,
            activation="relu"
        )
        self.downsample = None
```

```
    def build(self, input_shape):
        if input_shape[-1] != self.conv1.filters:
            self.downsample = layers.Conv1D(self.conv1.filters, 1, padding="same")
        super().build(input_shape)

    def call(self, x, training=False):
        y = self.conv1(x)
        y = self.dropout(y, training=training)
        y = self.conv2(y)

        shortcut = x if self.downsample is None else self.downsample(x)
        return layers.add([shortcut, y])

def build_tcn_backbone(vocab_size: int, embed_dim: int = 32) -> keras.Model:
    """Спільний бекбон для обох моделей."""
    inputs = keras.Input(shape=(MAX_LEN,), dtype=tf.int32)
    x = layers.Embedding(input_dim=vocab_size + 1, output_dim=embed_dim,
mask_zero=False)(inputs)

    # Стек TCN-блоків із наростаючою дилатацією
    for d in [1, 2, 4, 8, 16]:
        x = TCNResidualBlock(filters=64, kernel_size=3, dilation_rate=d, dropout=0.15)(x)

    x = layers.GlobalAveragePooling1D()(x)
    x = layers.Dense(64, activation="relu")(x)
    x = layers.Dropout(0.2)(x)
    return keras.Model(inputs, x, name="tcn_backbone")

def build_regression_model(vocab_size: int) -> keras.Model:
    backbone = build_tcn_backbone(vocab_size)
    outputs = layers.Dense(1, activation="linear", name="score")(backbone.output)
    model = keras.Model(backbone.input, outputs, name="tcn_regression")
    model.compile(
        optimizer=keras.optimizers.Adam(1e-3),
        loss="mse",
        metrics=[keras.metrics.MeanAbsoluteError(name="mae")]
    )
    return model

def build_classification_model(vocab_size: int) -> keras.Model:
    backbone = build_tcn_backbone(vocab_size)
    outputs = layers.Dense(1, activation="sigmoid", name="p_dga")(backbone.output)
    model = keras.Model(backbone.input, outputs, name="tcn_classification")
    model.compile(
        optimizer=keras.optimizers.Adam(1e-3),
        loss="binary_crossentropy",
        metrics=[keras.metrics.BinaryAccuracy(name="acc"),
keras.metrics.Precision(name="precision"), keras.metrics.Recall(name="recall")]
    )
    return model

def main():
```

```
    set_seed(SEED)

    # Артефакти з Додатку А
    X = np.load("artifacts/X_domains.npy")
    y = np.load("artifacts/y_labels.npy")
    with open("artifacts/vocab.json", "r", encoding="utf-8") as f:
        vocab = json.load(f)

    # Ключовий момент: розбиття 75:25 (як у роботі)
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.25, random_state=SEED, stratify=y
    )

    vocab_size = len(vocab)

    # 1) Регресійна модель (таргет як float 0.0/1.0)
    reg = build_regression_model(vocab_size)
    reg.summary()
    reg.fit(
        X_train, y_train.astype(np.float32),
        validation_data=(X_test, y_test.astype(np.float32)),
        epochs=2,               # як у роботі для першої моделі
        batch_size=4096,
        verbose=2
    )
    reg.save("artifacts/model_tcn_regression.keras")

    # 2) Класифікаційна модель
    clf = build_classification_model(vocab_size)
    clf.summary()
    clf.fit(
        X_train, y_train.astype(np.float32),
        validation_data=(X_test, y_test.astype(np.float32)),
        epochs=10,              # як у роботі для другої моделі
        batch_size=4096,
        verbose=2
    )
    clf.save("artifacts/model_tcn_classification.keras")

    # Швидка оцінка (класифікація)
    p = clf.predict(X_test, batch_size=8192).reshape(-1)
    y_pred = (p >= 0.5).astype(np.int32)

    cm = confusion_matrix(y_test, y_pred)
    print("Confusion matrix:\n", cm)
    print(classification_report(y_test, y_pred, digits=4))

if __name__ == "__main__":
    main()
```

ДОДАТОК В
КОД ОЦІНЮВАННЯ ЯКОСТІ МОДЕЛЕЙ ТА ВІЗУАЛІЗАЦІЇ РЕЗУЛЬТАТІВ

```
# appendix_c_evaluate_and_plots.py
# Розрахунок метрик якості та візуалізація матриці помилок.

from __future__ import annotations
import numpy as np

from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, precision_score, recall_score, f1_score, accuracy_score, mean_squared_error

import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf

SEED = 42

def sensitivity_recall(y_true, y_pred) -> float:
    return recall_score(y_true, y_pred, pos_label=1)

def specificity(y_true, y_pred) -> float:
    # специфічність = TN / (TN + FP)
    cm = confusion_matrix(y_true, y_pred)
    tn, fp = cm[0, 0], cm[0, 1]
    return float(tn / (tn + fp + 1e-12))

def main():
    X = np.load("artifacts/X_domains.npy")
    y = np.load("artifacts/y_labels.npy")

    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.25, random_state=SEED, stratify=y
    )

    clf = tf.keras.models.load_model("artifacts/model_tcn_classification.keras")

    p = clf.predict(X_test, batch_size=8192).reshape(-1)
    y_pred = (p >= 0.5).astype(np.int32)

    # Ключові метрики (як у таблицях/описі)
    acc = accuracy_score(y_test, y_pred)
    prec = precision_score(y_test, y_pred, zero_division=0)
    rec = recall_score(y_test, y_pred, zero_division=0)
    f1 = f1_score(y_test, y_pred, zero_division=0)
```

```
    mse = mean_squared_error(y_test.astype(np.float32), p.astype(np.float32))

    sens = sensitivity_recall(y_test, y_pred)
    spec = specificity(y_test, y_pred)

    print("Accuracy:", acc)
    print("Precision:", prec)
    print("Recall (Sensitivity):", sens)
    print("Specificity:", spec)
    print("F1:", f1)
    print("MSE:", mse)

    # Матриця помилок
    cm = confusion_matrix(y_test, y_pred)
    plt.figure()
    sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
    plt.xlabel("Predicted")
    plt.ylabel("True")
    plt.title("Confusion Matrix (DGA classification)")
    plt.tight_layout()
    plt.savefig("artifacts/confusion_matrix.png", dpi=200)
    plt.show()

if __name__ == "__main__":
    main()
```

ДОДАТОК Г
ПРИКЛАД ВИКОРИСТАННЯ НАТРЕНОВАНОЇ МОДЕЛІ ДЛЯ АНАЛІЗУ ДОМЕНІВ

```
# appendix_d_inference.py
# Інференс: завантаження моделі та оцінка ймовірності DGA для заданих доменів.

from __future__ import annotations
import json
import numpy as np
import tensorflow as tf

MAX_LEN = 73

def normalize_domain(s: str) -> str:
    return s.strip().lower().replace("https://", "").replace("http://", "").split("/")[0].strip(".")

def encode(domains: list[str], vocab: dict[str, int]) -> np.ndarray:
    x = np.zeros((len(domains), MAX_LEN), dtype=np.int32)
    for i, d in enumerate(domains):
        d = normalize_domain(d)
        seq = [vocab.get(ch, 0) for ch in d[:MAX_LEN]]
        x[i, :len(seq)] = np.array(seq, dtype=np.int32)
    return x

def main():
    with open("artifacts/vocab.json", "r", encoding="utf-8") as f:
        vocab = json.load(f)

    model = tf.keras.models.load_model("artifacts/model_tcn_classification.keras")

    test_domains = [
        "google.com",
        "microsoft.com",
        "xj3kq9a-zz1p0.com",      # умовний приклад "схожого на DGA"
        "ab12cd34ef56.org"
    ]

    X = encode(test_domains, vocab)
    p = model.predict(X).reshape(-1)

    for d, prob in zip(test_domains, p):
        label = "DGA" if prob >= 0.5 else "Legit"
        print(f"{d:25s}  p(DGA)={prob:.4f}  -> {label}")

if __name__ == "__main__":
    main()
```