МАГІСТЕРСЬКА РОБОТА

МР. ІПм - 32.00.00.000 ПЗ

Група ІПм-24-2

Кулик Роман

2025

Івано-Франківський національний технічний університет нафти і газу

Факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Кулик Роман Олегович

(прізвище, ім'я, по батькові)

УДК 004.9

(індекс)

МАГІСТЕРСЬКА РОБОТА

Моделі та методи підсилення засобів виявлення апаратних троянів шляхом зниження лінійності ознак

(назва роботи)

Інженерія програмного забезпечення

(назва освітньої програми)

121 - Інженерія програмного забезпечення

(шифр і назва спеціальності)

Кулик Р.О.

(підпис, ініціали та прізвище здобувача освітнього ступеня)

Науковий керівник Бандура Вікторія Валеріївна, к.т.н., доцент

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Допущено до захисту

Завідувач кафедри

доц. Бандура В.В.

(посада) (підпис) (дата) (ініціали та прізвище)

Нормоконтроль

доц. Вовк Р.Б.

(посада) (підпис) (дата) (ініціали та прізвище)

Робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Івано-Франківськ – 2025

Івано-Франківський національний технічний університет нафти і газу

Факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Освітній рівень магістр

Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Зав. кафедрою **ІПЗ**

доц. В.В. Бандура

“ 04 ” вересня 2025 р.

ЗАВДАННЯ

НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Кулику Роману Олеговичу

(прізвище, ім’я, по-батькові)

1. Тема магістерської роботи “ Моделі та методи підсилення засобів виявлення апаратних троянів шляхом зниження лінійності ознак”

керівник проекту (роботи) Бандура Вікторія Валеріївна, к.т.н., доцент

затверджені наказом закладу вищої освіти від “ 05 ” листопада 2025 р. № 695 /7

2. Строк подання студентом проекту (роботи) 15 грудня 2025 р.

3. Вихідні дані до проекту (роботи) Концепції та формальні моделі і методи побудови інформаційних та програмних технологій виявлення апаратних троянів

4. Зміст розрахунково - пояснювальної записки(перелік питань, які потрібно розробити)

1. Аналіз проблематики виявлення апаратних троянів в системах Інтернету речей
2 Дослідження алгоритмів машинного навчання для виявлення апаратних троянів
3. Огляд наборів даних та ознак для детекції апаратних троянів
4. Методологія виявлення апаратних троянів шляхом зниження лінійності ознак засобами ML

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

1. Фази розробки ІС (рис. 1.1)
2. Методологія зворотного проєктування компонування (рис. 1.2)
3. Структура апаратного трояна (рис. 1.3)
4. Таксономія машинного навчання (рис. 1.4)
5. Класифікація та регресія в машинному навчання (рис. 1.5)

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата
Перевірка на плагіат	доц., к.т.н. Вовк Р.Б.	

7. Дата видачі завдання 04 вересня 2025 р.

Керівник _____
(підпис)

Завдання прийняв до виконання_____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назви етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Підбір і вивчення літератури по темі магістерської роботи	17.09.2025	виконано
2	Аналіз проблематики виявлення апаратних троянів в системах Інтернету речей	30.09.2025	виконано
3	Дослідження алгоритмів машинного навчання для виявлення апаратних троянів	16.10.2025	виконано
4	Огляд наборів даних та ознак для детекції апаратних троянів	10.11.2025	виконано
5	Методологія виявлення апаратних троянів шляхом зниження лінійності ознак засобами ML	22.11.2025	виконано
6	Оцінка продуктивності та аналіз результатів	03.12.2025	виконано
7	Затвердження пояснювальної записки роботи завідувачем кафедри	17.12.2025	виконано

Студент – магістр _____
(підпис)

Керівник роботи _____
(підпис)

АНОТАЦІЯ

Магістерська робота: 76 с., 16 рис., 5 табл., 41 джерел.

Тема: Моделі та методи підсилення засобів виявлення апаратних троянів шляхом зниження лінійності ознак

Мета роботи - підвищення ефективності систем виявлення апаратних троянів шляхом розроблення моделей і методів підсилення засобів детекції через зниження лінійності ознак у просторах даних засобами машинного навчання.

Об'єкт дослідження - процеси виявлення апаратних троянів у цифрових схемах і системах Інтернету речей із використанням методів машинного навчання.

Предмет дослідження - моделі, методи та інструменти підсилення ефективності систем детекції апаратних троянів шляхом зниження лінійності ознак і трансформації простору даних.

Результати дослідження

В роботі запропоновано методологію виявлення апаратних троянів шляхом зниження лінійності ознак засобами машинного навчання.

Висновок

Розроблено процес видобування ознак із HDL-описів і нетлістів, створено узагальнену схему виявлення троянів, що поєднує етапи попередньої обробки, трансформації ознак, навчання моделі та верифікації результатів

АПАРАТНІ ТРОЯНИ; ІНТЕРНЕТ РЕЧЕЙ; БЕЗПЕКА АПАРАТНОГО ЗАБЕЗПЕЧЕННЯ; МАШИННЕ НАВЧАННЯ; КЛАСИФІКАЦІЯ; ОЗНАКИ; ЗНИЖЕННЯ ЛІНІЙНОСТІ; ЕКСТРАКЦІЯ ОЗНАК; ВЕРИФІКАЦІЯ; ЦИФРОВІ СХЕМИ.

ABSTRACT

Master Thesis: 76 pp., 16 fig., 5 tab., 41 sources.

Topic: Models and methods for strengthening hardware Trojan detection tools by reducing the linearity of features

The purpose of the work is to increase the efficiency of hardware Trojan detection systems by developing models and methods for strengthening detection tools by reducing the linearity of features in data spaces using machine learning.

The object of the study is the processes of detecting hardware Trojans in digital circuits and Internet of Things systems using machine learning methods.

The subject of the study is models, methods and tools for strengthening the efficiency of hardware Trojan detection systems by reducing the linearity of features and transforming the data space.

Research results

The paper proposes a methodology for detecting hardware Trojans by reducing the linearity of features using machine learning.

Conclusion

The process of extracting features from HDL descriptions and netlists has been developed, a generalized Trojan detection scheme has been created, which combines the stages of preprocessing, feature transformation, model training and results verification

HARDWARE TROJANS; INTERNET OF THINGS; HARDWARE SECURITY; MACHINE LEARNING; CLASSIFICATION; FEATURES; LINEARITY REDUCTION; FEATURE EXTRACTION; VERIFICATION; DIGITAL SCHEMES.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

HDLs - Hardware description languages

HT - Hardware Trojan

ICs - Integrated circuits

RFID - radio frequency identification devices

SoC - system-on-a-chip

RTL - Register-Transfer Level

HDL - Hardware Description Language

SoC - System-on-a-Chip

LOF - Local Outlier Factor

ВСТУП

Актуальність теми.

Сучасний етап розвитку інформаційних технологій характеризується активною інтеграцією обчислювальних систем у фізичне середовище, що призвело до формування нової парадигми — Інтернету речей (Internet of Things, IoT). Завдяки цьому відбувається масове підключення сенсорів, мікроконтролерів, вбудованих систем і розумних пристроїв до глобальних мереж, що дозволяє реалізовувати концепції «розумних» міст, транспорту, енергетичних систем і промислових об’єктів. Однак така масштабна взаємодія між пристроями різних виробників породжує нові виклики у сфері безпеки, зокрема на рівні апаратного забезпечення.

Одним із найнебезпечніших типів загроз для мікроелектронних систем є апаратні трояни — навмисно вбудовані модифікації схеми, здатні порушити її функціональність, здійснювати витік інформації або створювати умови для несанкціонованого доступу до пристрою. Відмінною рисою таких троянів є їхня прихованість: вони можуть залишатися неактивними протягом тривалого часу, активуючись лише за певних умов або сигналів. Це робить їх практично невидимими для традиційних методів тестування, що суттєво ускладнює процес верифікації апаратури.

Проблема виявлення апаратних троянів ускладнюється низкою факторів. По-перше, сучасні ланцюги постачання мікроелектронних компонентів глобалізовані: етапи проєктування, виробництва та тестування можуть здійснюватися різними компаніями у різних країнах. Це створює ризик інтеграції троянів на будь-якому етапі життєвого циклу пристрою. По-друге, зростаюча щільність інтеграції елементів у мікросхемах і складність топологій ускладнюють аналіз фізичних характеристик. По-третє, поява апаратних троянів, що мають мінімальний вплив на енергоспоживання, час затримки або площу кристалу, робить їх практично непомітними для стандартних методів тестування.

У контексті безпеки IoT ці виклики набувають системного характеру. Компрометація одного апаратного вузла може спричинити каскадні збої або навіть катастрофічні наслідки для всієї мережі. Тому забезпечення довіри до апаратних компонентів є ключовим завданням для державних, промислових і військових застосувань. Саме тому питання виявлення апаратних троянів стає одним із пріоритетних напрямів сучасних досліджень у галузі кіберфізичної безпеки.

Серед сучасних підходів до детекції апаратних троянів особливе місце займають методи, що базуються на машинному навчанні. Вони дозволяють аналізувати великі обсяги даних про структуру цифрових схем і виявляти приховані закономірності, недоступні для традиційних інженерних методів. Алгоритми класифікації та кластеризації здатні автоматично розрізняти «чисті» схеми та схеми із потенційними троянами, спираючись на аналіз наборів ознак, отриманих із нетлістів або фізичних вимірювань.

Проте, ефективність моделей машинного навчання значною мірою залежить від структури ознак, які описують схему. Якщо простір ознак характеризується надмірною лінійністю, моделі можуть не виявляти слабкі, але суттєві відхилення, що відображають наявність трояна. Це призводить до хибних класифікацій і зниження точності детекції. Тому актуальною науковою задачею є зниження лінійності ознак, яке дозволяє збільшити міжкласові відстані у просторі даних, підвищуючи здатність моделі розпізнавати приховані залежності.

Розроблення моделей і методів підсилення засобів виявлення апаратних троянів шляхом зниження лінійності ознак є перспективним напрямом підвищення точності та надійності систем машинного аналізу безпеки. Застосування таких підходів дозволяє створити адаптивні системи контролю якості апаратних рішень, які здатні ефективно працювати в умовах шуму, змін технологічних параметрів і відсутності повної інформації про структуру схеми.

Магістерська робота спрямована на створення науково обґрунтованої методології, яка поєднує принципи машинного навчання, методи екстракції ознак із апаратних описів і техніки зниження лінійності даних для покращення виявлення апаратних троянів. Результати дослідження мають практичну значущість для розроблення систем автоматизованої верифікації апаратного забезпечення, безпечного проєктування IoT-пристроїв, а також формування довірених платформ обчислень у кіберфізичних системах.

Проблема забезпечення апаратної безпеки набуває стратегічного значення у зв'язку з поширенням систем IoT, кіберфізичних об'єктів і вбудованих платформ, які працюють із критично важливими даними. Впровадження апаратних троянів у виробничий процес становить серйозну загрозу не лише для окремих пристроїв, а й для цілих інформаційних екосистем. Такі трояни можуть призвести до витоку конфіденційної інформації, порушення цілісності обчислень або відмови систем у ключових режимах роботи.

Традиційні засоби тестування апаратури — функціональне тестування, аналіз потужності, часових затримок або електромагнітних відгуків — не забезпечують достатнього рівня достовірності для сучасних багатокомпонентних схем. Водночас машинне навчання відкриває можливість побудови моделей, здатних самостійно навчатися на прикладах і виявляти складні нелінійні залежності у структурі схем. Проте для підвищення їх ефективності необхідне удосконалення етапу підготовки даних, а саме — зменшення лінійності ознак, що дозволяє чіткіше відокремити класи «чистих» і «компрометованих» елементів.

Отже, актуальність роботи полягає у створенні нової методології виявлення апаратних троянів, яка поєднує алгоритми машинного навчання із методами трансформації ознак, спрямованими на підвищення якості класифікації та достовірності верифікації апаратного забезпечення.

Мета роботи - підвищення ефективності систем виявлення апаратних троянів шляхом розроблення моделей і методів підсилення засобів детекції

через зниження лінійності ознак у просторах даних засобами машинного навчання.

Об’єкт дослідження - процеси виявлення апаратних троянів у цифрових схемах і системах Інтернету речей із використанням методів машинного навчання.

Предмет дослідження - моделі, методи та інструменти підсилення ефективності систем детекції апаратних троянів шляхом зниження лінійності ознак і трансформації простору даних.

Завдання дослідження

1. Провести аналіз проблематики виявлення апаратних троянів та класифікацію існуючих загроз.

2. Дослідити методи машинного навчання, що застосовуються для детекції апаратних троянів.

3. Розробити узагальнену схему виявлення троянів із використанням трансформацій ознак.

4. Запропонувати методику зниження лінійності ознак у процесі підготовки даних.

5. Провести експериментальну оцінку ефективності запропонованої методики.

6. Проаналізувати результати та порівняти їх з існуючими підходами до детекції.

Методи дослідження

У роботі використано аналітичні методи для огляду наукових джерел, методи формального опису структур електронних схем, статистичні методи обробки даних, алгоритми машинного навчання (Random Forest, Decision Tree, SVM, Neural Networks), методи трансформації ознак і зниження їх лінійності, а також програмні засоби моделювання, верифікації та візуалізації результатів.

Наукова новизна отриманих результатів

Запропоновано підхід до підсилення моделей виявлення апаратних троянів шляхом зниження лінійності ознак, що дозволяє підвищити точність класифікації. Розроблено узагальнену модель процесу виявлення апаратних троянів, яка інтегрує етапи екстракції, трансформації та аналізу ознак.

Практичне застосування результатів

Результати дослідження можуть бути використані для побудови систем автоматизованої верифікації та тестування мікроелектронних компонентів, інтегрованих у виробничі процеси апаратного проєктування. Запропонована методика може бути впроваджена у програмні комплекси перевірки безпеки IoT-пристроїв, а також у навчальні курси з апаратної безпеки, штучного інтелекту та системного аналізу. Застосування розроблених методів сприятиме підвищенню надійності апаратного забезпечення та зниженню ризику несанкціонованих втручань на рівні мікроелектроніки.

Структура магістерської роботи. Представлена робота складається зі вступу, трьох розділів та висновків. Загальний обсяг роботи становить 76 сторінок, і містить 16 рисунків, 5 таблиць, перелік використаних джерел із 41 позиції.

РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ ВИЯВЛЕННЯ АПАРАТНИХ ТРОЯНІВ В СИСТЕМАХ ІНТЕРНЕТУ РЕЧЕЙ

1.1. Огляд проблеми апаратних троянів

Світова індустрія мікроелектроніки демонструє експоненційне зростання попиту на високоінтегровані електронні пристрої, зокрема на системи на кристалі (SoC), активні медичні імплантати та пристрої Інтернету речей (IoT). Цей тренд зумовлює необхідність у постійному підвищенні складності та мініатюризації інтегральних схем (IC).

Проте, глобалізація ланцюгів постачання у сфері розробки IC, де етапи проєктування, виробництва та тестування часто передаються на аутсорсинг численним, територіально розподіленим та потенційно недовіреним стороннім суб'єктам, створює значні вразливості. Відхід від моделі "одного надійного постачальника" відкриває шлях для зловмисників.

Ключовою загрозою є апаратні трояни (HT - Hardware Trojans) – зловмисні модифікації, таємно впроваджені у початковий дизайн IC або безпосередньо у фізичний кристал на етапі виробництва. Ці модифікації можуть мати наступні наслідки:

- Порушення цілісності та конфіденційності даних, створення "чорних ходів" (backdoors) для несанкціонованого доступу та витоку критичної інформації.

- Деградація продуктивності та функціональності, зниження швидкодії, спричинення збоїв або повне виведення пристрою з ладу.

- Зміна функціональної поведінки, активування шкідливого функціоналу за певних умов, що ускладнює їхнє виявлення під час стандартного тестування.

Проблема виявлення апаратних троянів є нетривіальною, оскільки трояни часто мають мінімальний вплив на статичні та динамічні

характеристики схеми (наприклад, споживання потужності чи затримку) і активуються лише за рідкісних, специфічних умов.

У межах даної магістерської роботи пропонується та досліджується методологія автоматизованого виявлення апаратних троянів, що базується на застосуванні моделей машинного навчання (МН).

Як первинний матеріал для аналізу використовуються ознаки, отримані зі списку з'єднань (netlist) цифрового апаратного дизайну. Список з'єднань генерується після етапу логічного синтезу і містить вичерпну інформацію про структурні зв'язки між елементами схеми.

Для класифікації дизайнів як "чистих" або "заражених" буде застосовано два різновиди моделей машинного навчання (методи ансамблевого навчання та нейронні мережі).

Для забезпечення узагальнювальної здатності моделі та запобігання надмірному підлаштуванню (overfitting) до навчального набору даних, особлива увага приділяється заходам запобігання взаємозалежності (мультиколінеарності) між вибраними ознаками. Це досягається шляхом застосування методів відбору ознак (feature selection) або зниження розмірності (dimensionality reduction).

Цей підхід дозволить розробити неінвазивний, автоматизований інструмент для верифікації безпеки інтегральних схем на ранніх етапах проєктування, тим самим підвищуючи надійність та довіру до електронних компонентів у критично важливих сферах.

1.2. Безпека в екосистемі Інтернету речей та виявлення апаратних троянів

1.2.1. Еволюція Інтернету речей (IoT)

Інтернет речей (IoT) являє собою глобальну мережу, що охоплює та інтегрує різноманітні пристрої з функціями сприйняття інформації, включаючи радіочастотні ідентифікатори (RFID), інфрачервоні сенсори та

системи глобального позиціонування. Розглядаючись як третя хвиля світової інформаційної індустрії після комп'ютерів та Інтернету, IoT демонструє високу поширеність інтегральних схем (IC) у критично важливих додатках.

З огляду на щоденне використання цих пристроїв для обробки та зберігання великих обсягів конфіденційних даних користувачів, потенційний ризик впровадження апаратних троянів (Hardware Trojans, HT) в IC набуває критичного значення. Експлуатація такої вразливості зловмисником може призвести до серйозного порушення приватності та безпеки кінцевого користувача.

Система на кристалі (SoC) є основним компонентом пристроїв IoT. Сучасні тенденції мініатюризації та підвищення складності дизайну SoC висувають значні технологічні та економічні виклики для виробників. У відповідь на ці виклики, виробники IC дедалі частіше вдаються до інтеграції Сторонніх IP-ядер (Third-Party IP, 3PIP). Використання 3PIP забезпечує економічну ефективність та дозволяє виробникам оптимізувати розподіл ресурсів, задовольняючи ринковий попит та стислі терміни.

Однак, безпека та надійність IP-ядер від недовірених сторонніх постачальників не можуть бути гарантовані, що істотно підвищує ризик впровадження апаратних троянів на етапі проєктування.

1.2.2. Природа апаратних троянів та загрози

Апаратний троян (HT) визначається як зловмисна модифікація, внесена в інтегральну схему. Наслідки активації такого трояну можуть включати:

- Витік конфіденційної інформації.
- Зміна системної функціональності та деградація продуктивності.
- Відмова в обслуговуванні (Denial-of-Service, DoS).
- Створення прихованого "чорного ходу" (backdoor) до системи.

Враховуючи широке застосування IC у критичній інфраструктурі, військових та біомедичних пристроях, несанкціонований та невиявлений троян може мати катастрофічні наслідки.

Стандартний процес фізичного проєктування та компонування IC, від системної специфікації до виготовлення та тестування (рис. 1.1), містить потенційні точки для вбудовування апаратного трояну. Метою даного дослідження є виявлення троянів, які могли бути вставлені саме на етапі проєктування.

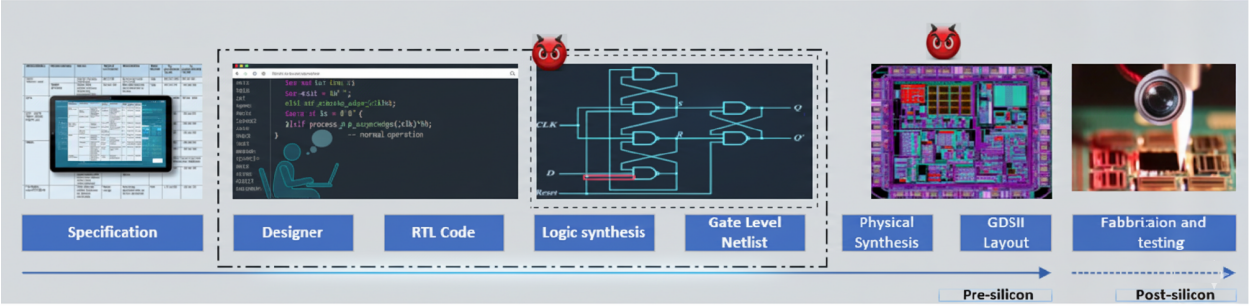

Рис. 1.1. Фази розробки IC

1.2.3. Методологія верифікації та контрзаходи

Розглянемо сценарій, коли компанія-розробник передає свій фізичний цифровий апаратний дизайн, відомий як Graphic Design System II (GDSII), на виготовлення потенційно недовіреній компанії-фабрикатору.

Нещодавнє дослідження [1] продемонструвало ефективність методу зворотного проєктування ReGDS. Цей підхід використовує технологічну бібліотеку для вилучення інформації про з'єднання на транзисторному рівні з GDSII, а потім застосовує зіставлення на основі відносин для ідентифікації логічних вентилів і відновлення оригінального списку з'єднань (netlist) на рівні вентилів. Автори підтвердили 100% успішність відновлення оригінального цифрового дизайну з фізичного компонування. Цей результат підкреслює вразливість інтелектуальної власності (IP) на етапі передачі GDSII.

Наявність фізичного компонування (layout) дизайну та технологічної бібліотеки дозволяє дослідити обсяг інформації про дизайн, доступної для ливарного виробництва (foundry), що розглядається як потенційний сценарій інсайдерської атаки.

Методологія зворотного проєктування компонування, що отримала назву ReGDS, проілюстрована на рисунку 1.2.

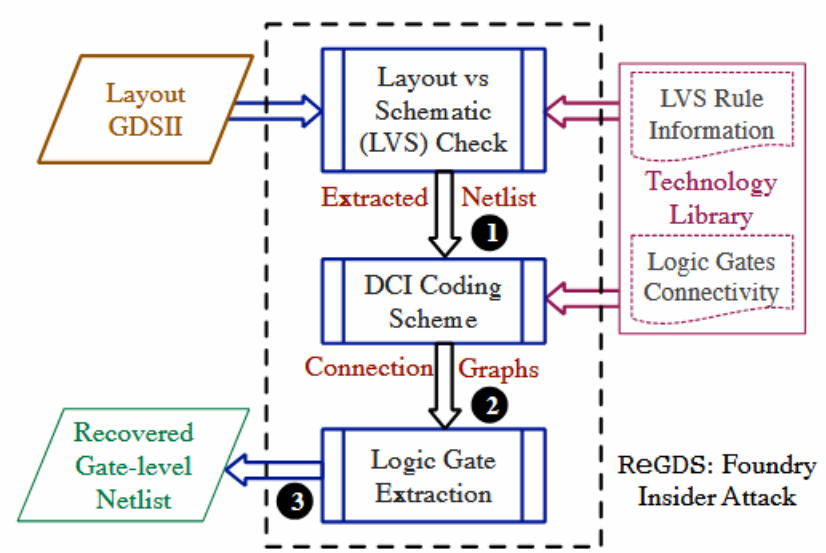

Рис. 1.2. Методологія зворотного проєктування компонування

Розглянемо цю методологію.

Екстракція з'єднань транзисторів (Крок 1). Використовуючи інструмент LVS (Layout Versus Schematic), екстрагують інформацію про з'єднання транзисторів з компонування 1. Однак, через відсутність інформації про мережу живлення і, відповідно, напрямок протікання струму, виникають неузгодженості у підключенні терміналів транзисторів.

Нове представлення схеми (Крок 2). Для подолання цих неузгодженостей розроблено представлення цифрових схем на основі їхньої зв'язності (connectivity-based representation) 2.

Відновлення списку з'єднань на рівні вентилів (Крок 3). На основі мережі транзисторів конструюються графи зв'язності. Далі застосовується зіставлення шаблонів (pattern matching) для ідентифікації логічних вентилів і, як наслідок, відновлення оригінального списку з'єднань на рівні вентилів (gate-level netlist) 3.

Оскільки система ReGDS використовує технологічну бібліотеку як вхідні дані для зворотного проєктування, вона застосовна до всіх стилів

цифрової логіки на основі транзисторів. У цьому дослідженні реалізація на основі технології CMOS використовується як доказ концепції (proof of concept). Створюється синтетична вихідна база даних, яка відповідає тій самій технології, що й компонування дизайну. Наприклад, для формування вихідної бази даних достатньо простого списку з'єднань інвертора.

За наявності сконструйованої вихідної бази даних, компонування дизайну та технологічної бібліотеки, запускається LVS-перевірка. Незважаючи на те, що ця перевірка завершується невдало через невідповідності між компонуванням та вихідною базою даних, повна інформація про з'єднання на транзисторному рівні успішно екстрагується з бази даних компонування у форматі SPICE.

Далі до екстрагованого зі списку з'єднань компонування застосовується етап постобробки, що включає наступні операції. Кілька пристроїв, з'єднаних паралельно та які мають подібні фізичні характеристики, зводяться до одного пристрою. Змішані підсхеми (subckts), що містять як ієрархічні екземпляри, так і пристрої, ідентифікуються та сплющуються (flattened), щоб уможливити подальшу ідентифікацію логічних вентилів.

1.2.4. Метод верифікації на основі контролю фізичних параметрів

Для протидії загрозі з боку недовіреного виробника пропонується, щоб компанія-розробник здійснювала верифікацію автентичності виготовленого чипа шляхом точного вимірювання специфічних фізичних характеристик виробу. До таких характеристик відносяться:

- Споживання потужності (Power consumption).
- Витік потужності (Power leakage).
- Площа кристала (Area).
- Структурні особливості (кількість шарів, розміри елементів).

Для вимірювання структурних параметрів та площі чипа можуть бути використані високоточні методи, такі як скануюча електронна мікроскопія

(SEM) або фокусований іонний пучок (FIB). Взаємозв'язок між потужністю та площею є фундаментальним у цифровому апаратному проєктуванні.

1.2.4. Використання машинного навчання для детекції троянів

У випадку впровадження логічного трояна, який залишається неактивним до активації певним тригером, фабрикатор може приховати сплеск споживання потужності, використовуючи логічні вентилі з нульовим статичним споживанням. Це, як правило, призводить до збільшення загальної площі кристала або кількості шарів.

Для надійного виявлення таких модифікацій пропонується наступна методологія:

1. Виміряні фізичні характеристики (площа, потужність, витік) чипа до і після виготовлення вводяться як ознаки у модель машинного навчання.

2. Перевірка лінійності або інших статистичних залежностей між ознаками проводиться для запобігання надмірному підлаштуванню (overfitting) моделі.

3. Такий підхід на основі МН забезпечить високу ймовірність виявлення зловмисних модифікацій, внесених недовіреною компанією-фабрикатором.

1.3. Структура та класифікація загроз, спричинених апаратними троянами

1.3.1. Архітектура апаратного трояна

Апаратний троян є зловмисною модифікацією інтегральної схеми і структурно складається з двох ключових компонентів: механізму спрацьовування (trigger mechanism) та корисного навантаження (payload).

Як показано на рис. 1.3, механізм спрацьовування виконує функцію моніторингу стану чипа і забезпечує активацію корисного навантаження лише за рідкісних та специфічних умов.

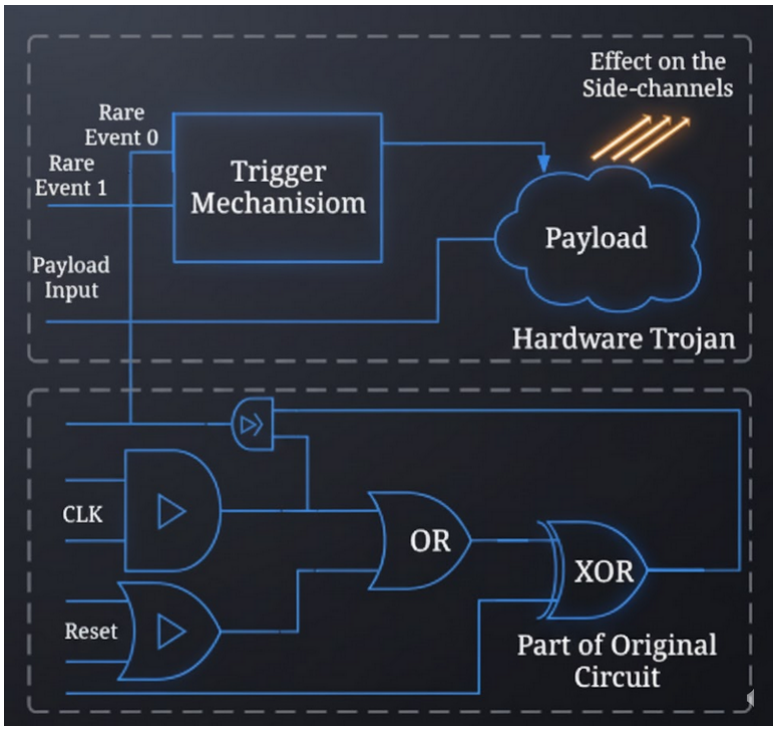

Рис. 1.3. Структура апаратного трояна

На рисунку 1.3 проілюстрована типова архітектура апаратного трояна (HT) , що складається з механізму спрацьовування (Trigger Mechanism) та корисного навантаження (Payload). Таке проєктування апаратного трояна, відоме як сплячий режим (dormancy), є цілеспрямованою стратегією для уникнення виявлення під час стандартних методів тестування після виробництва (post-fabrication testing).

1.3.2. Класифікація загроз безпеці електронних пристроїв

Впровадження апаратних троянів становить багатоаспектну загрозу безпеці та надійності електронних пристроїв. Основні категорії загроз включають:

1) Компрометація конфіденційності (Data Exfiltration).

Трояни можуть бути розроблені для несанкціонованого витоку (leakage) чутливих даних, таких як криптографічні ключі, облікові дані користувачів або інша конфіденційна інформація, на адресу зловмисника. Це пряме порушення принципу конфіденційності.

2) Відмова в обслуговуванні (Denial-of-Service, DoS).

Трояни можуть бути запрограмовані на порушення нормальної функціональності системи, спричинення збоїв у роботі або повне відключення пристрою. Це призводить до атаки типу DoS, що унеможливлює використання пристрою кінцевим користувачем.

3) Ін'єкція зловмисного коду (Malware Insertion Facilitation).

Апаратні трояни можуть створювати сприятливі умови для ін'єкції або активації шкідливого програмного забезпечення (malware) зловмисником. Після успішного впровадження, цей код може бути використаний для компрометації інших сегментів мережі або підвищення привілеїв доступу.

4) Створення "Чорних ходів" (Backdoors):

Трояни можуть бути використані для створення логічних "чорних ходів" (backdoors) у механізмах безпеки пристрою. Це дозволяє зловмисникам обходити встановлені протоколи аутентифікації та контролю доступу, отримуючи несанкціонований привілейований доступ до системи.

1.4. Емпіричні прецеденти компрометації апаратного забезпечення за допомогою апаратних троянів

Апаратні трояни (АТ) являють собою зловмисні модифікації, таємно інтегровані в апаратне забезпечення інтегральної схеми на критичних етапах її життєвого циклу, зокрема під час проєктування або виробництва.

Наслідки функціонування АТ є широким спектром шкідливих впливів, ступінь і характер яких безпосередньо залежать від архітектури трояна (механізму спрацьовування та корисного навантаження) та його цільової функції.

Історичні прецеденти демонструють, що НТ становлять значну загрозу не лише для комерційних, але й для військових та критичних інфраструктурних систем.

У 2007 році під час авіаудару по підозрюваному ядерному об'єкту в Сирії сирійська радіолокаційна система протиповітряної оборони була виведена з ладу. Дослідження показало, що це стало можливим через активацію задніх дверей (backdoor) у комерційному мікропроцесорі, який дозволив дистанційно активувати апаратний вимикач [3]. Цей випадок підтверджує можливість використання АТ для здійснення атак типу "Відмова в обслуговуванні" (DoS) у військових контекстах.

У 2010 році Міністерство оборони США виявило АТ, імплантований у понад 59 000 мікросхем, призначених для використання в різноманітних оборонних системах: від протиракетної оборони до пристроїв ідентифікації "свій-чужий" (Identification Friend or Foe, IFF). Цей АТ забезпечив супротивникам прихований канал доступу (backdoor) до всієї їхньої інтегрованої системи [4].

Також було виявлено АТ у мікросхемах Actel/Microsemi ProASIC3 FPGA, які широко використовуються у військово-промисловому комплексі. Троян впровадив небажану функціональність JTAG-інтерфейсу безпосередньо у кремній , що дозволило зловмисникам витягувати секретні ключі, маніпулювати конфігурацією чипа та, зрештою, отримувати повний контроль над цільовою системою [5].

Інцидент із серверами Supermicro (2015). Після спостереження нетипової мережевої активності та проблем із прошивкою (firmware issues) у своїх серверах Supermicro у 2015 році, компанія Apple виявила підозрілий, несанкціонований чіп. Хоча деталі цього інциденту залишаються чутливими, він слугує прикладом виявлення потенційної апаратної компрометації у ланцюгу постачання комерційних серверних рішень.

Ці приклади підкреслюють стратегічну небезпеку апаратних троянів та нагальну необхідність розробки надійних методологій для верифікації автентичності та цілісності апаратного забезпечення на всіх етапах його виробництва.

1.5. Застосування машинного навчання для виявлення апаратних троянів

Наявність апаратних троянів (HT) на кристалі може мати катастрофічні наслідки, включаючи несанкціоновану ексфільтрацію даних, функціональні несправності пристроїв та інші деструктивні ефекти. Оскільки складність та різноманітність інтегральних схем (IC) неухильно зростає, традиційні методи детекції HT демонструють зниження ефективності.

До таких традиційних методів належать:

- Ручна візуальна інспекція (Manual Inspection).
- Аналіз побічних каналів (Side-Channel Analysis) .
- Ін'єкція помилок (Fault Injection).

Ці підходи стають менш масштабованими та трудомісткими у контексті сучасних високоінтегрованих дизайнів.

Для вирішення проблеми масштабованості та підвищення надійності виявлення HT, дослідницька спільнота активно інтегрує методи машинного навчання (ML). Моделі ML мають здатність навчатися на великих, розмічених наборах даних, що включають як скомпрометовані, так і чисті дизайни IC. Це дозволяє їм автоматично ідентифікувати тонкі аномалії у структурній або поведінковій характеристиці схеми, які можуть свідчити про наявність HT.

Використання алгоритмів ML є критично важливим для підвищення безпеки та довіри (trustworthiness) IC, особливо у сферах з підвищеними вимогами до цілісності, таких як автомобільні, аерокосмічні та оборонні системи.

1.5.1. Завдання магістерського дослідження

Основне дослідницьке питання, на яке спрямована ця робота, формулюється наступним чином: Чи можна розробити надійний та ефективний метод для виявлення присутності апаратного трояна, який міг

бути вбудований у цифровий апаратний дизайн на етапі, що передує фізичному виробництву кремнію (pre-silicon stage).

Дане дослідження робить внесок у сферу безпеки ІС через два ключові аспекти:

1. Екстракція ознак з пост-синтезного дизайну - розробка методології для ефективного видобування інформативних ознак безпосередньо з цифрового апаратного дизайну після завершення етапу логічного синтезу (netlist-level features).

2. Мінімізація лінійності між ознаками для ML - реалізація заходів, спрямованих на зменшення лінійної залежності (мультиколінеарності) між ознаками, отриманими зі списку з'єднань, які зазнали масштабування для введення в моделі ML.

Проблема виникає через те, що алгоритми ML обробляють лише числові значення, а не фізичні одиниці вимірювання (наприклад, потужність у мВт або площа у μm2). Як наслідок, ознаки зазвичай масштабуються (наприклад, до діапазону [0,1]), що може штучно індукувати лінійність між деякими параметрами.

Зменшення цих лінійностей є критично важливим для підвищення узагальнювальної здатності алгоритму ML, що дозволяє уникнути надмірного підлаштування (overfitting) під навчальний набір даних. Це, у свою чергу, забезпечує покращену продуктивність при виявленні НТ на нових, невідомих тестових бенчмарках [7].

1.5.2. Деталізація екстракції ознак зі списку з'єднань (Netlist Feature Extraction)

На етапі після логічного синтезу, цифровий апаратний дизайн представлений у вигляді списку з'єднань (netlist) — текстового файлу, що описує логічні вентилі (gates) та їхні електричні з'єднання. Ознаки, витягнуті з цього джерела, характеризують структурні властивості схеми.

Оскільки апаратні трояни (HT) є структурними модифікаціями (додавання логічних вентилів і провідників), вони спричиняють локальні або глобальні аномалії у характеристиках списку з'єднань.

Ознаки поділяються на дві основні категорії: статистичні (розмірні) та структурні (зв'язності).

Таблиця 1.1.

Категорії ознак

Категорія ознак	**Призначення**	**Приклади ознак**
Розмірні/ статистичні ознаки	Характеризують загальний обсяг, складність і ресурсні потреби схеми. АТ зазвичай збільшують ці параметри.	**- Загальна площа (Total Area):** сумарна площа всіх логічних вентилів, що використовуються. НТ часто додають зайву площу. **- Загальне споживання потужності (Total Power):** сумарна динамічна та статична потужність схеми. - **Кількість примітивних елементів:** загальна кількість логічних вентилів (NAND, NOR, XOR тощо). **- Кількість транзисторів:** пряма метрика складності (витягується через технологічну бібліотеку).
Ознаки зв'язності (Connectivity Features)	Описують топологію та складність з'єднань у схемі, що є ключовим для виявлення прихованої логіки.	**- Максимальна глибина (Max Depth):** довжина найдовшого комбінаційного шляху. - **Кількість вузол-вентиль:** розподіл входів/виходів вентилів (Fan-in/Fan-out). - **Розподіл типів вентилів:** Відсоток використання кожного типу логічного вентиля (частка XOR-гейтів). АТ часто використовують рідкісні або надлишкові вентилі. - **Ознаки з'єднань (Net Features):** середня кількість вентилів, підключених до одного провідника.

Деякі ознаки безпосередньо спрямовані на виявлення логічних структур, характерних для АТ.

Таблиця 1.2.

Спеціалізовані ознаки для детекції апаратних троянів

Ознака	Мета детекції троянів
Кількість непідключених портів (Floating Ports)	АТ можуть мати компоненти, які тимчасово або навмисно не підключені до основної логіки, що призводить до аномалії у кількості "плаваючих" входів/виходів.
Розмірність комбінаційних тригерів (Rare Logic Gates)	Виявлення незвично великих або складних логічних вентилів (наприклад, 8-входових AND-схем), які можуть бути частиною складного механізму спрацьовування.
Аномалії в розподілі зворотного зв'язку (Feedback Loops)	НТ можуть вводити приховані або нелогічні петлі зворотного зв'язку для керування станом, що можна виявити через аномалії у топологічних метриках.

Як зазначалося раніше, перед введенням у модель ML ці ознаки підлягають нормалізації або стандартизації.

Наслідок: якщо дві ознаки — площа та кількість транзисторів — мають сильну природну лінійну кореляцію у чистому дизайні (більше транзисторів = більше площа), то після однакового масштабування ця кореляція може стати надмірною.

Ризик: надмірна лінійність (мультиколінеарність) у навчальному наборі може призвести до того, що модель ML надмірно підлаштується під цю залежність, а не вивчатиме тонкі аномалії, спричинені трояном.

Для вирішення цієї проблеми застосовуються спеціалізовані методи, такі як:

- Відбір ознак (Feature Selection) - вибір лише найменш корельованих та найбільш інформативних ознак.

- Зниження розмірності (Dimensionality Reduction) - використання методів наприклад аналізу головних компонентів (PCA) для перетворення висококорельованих ознак у менший набір незалежних компонент.

Висновки до розділу

У першому розділі було проведено всебічний аналіз проблематики виявлення апаратних троянів у контексті зростаючої складності апаратно-програмних систем, зокрема в екосистемі Інтернету речей (IoT). Дослідження показало, що з розвитком технологічних процесів проєктування мікроелектроніки та зростанням обсягів аутсорсингу виробництва мікросхем, істотно знижується рівень довіри до компонентів апаратного забезпечення, що створює передумови для прихованого впровадження шкідливих елементів – апаратних троянів.

Було проаналізовано сучасні підходи до забезпечення безпеки в екосистемі IoT, де поєднуються апаратні, програмні та мережеві рівні захисту. Показано, що у контексті IoT загроза апаратних троянів набуває системного характеру — компрометація одного пристрою може призвести до розповсюдження атаки через мережеві взаємозв’язки.

У роботі визначено, що значна частина сучасних досліджень фокусується на застосуванні машинного навчання для автоматизованої детекції апаратних троянів. Окремо розглянуто наявні підходи до зниження лінійності ознак, що використовується як метод підсилення спроможності моделей машинного навчання виявляти приховані нелінійні залежності у структурі даних. Крім того, в розділі наведено узагальнення емпіричних прецедентів компрометації апаратного забезпечення, які демонструють реальність загроз і недостатність існуючих контрзаходів. Це підтверджує необхідність розробки нових методів, здатних працювати із різними форматами даних, підтримувати масштабування та бути інтегрованими у процеси автоматизованої верифікації.

РОЗДІЛ 2. ДОСЛІДЖЕННЯ АЛГОРИТМІВ МАШИННОГО НАВЧАННЯ ДЛЯ ВИЯВЛЕННЯ АПАРАТНИХ ТРОЯНІВ

2.1. Концептуальні основи контрольованого машинного навчання (з учителем)

Машинне навчання (ML) є ключовою підгалуззю штучного інтелекту (AI), яка сфокусована на розробці обчислювальних алгоритмів та статистичних моделей, що забезпечують здатність комп'ютерних систем навчатися на основі емпіричних даних та формувати прогнози або приймати рішення на базі отриманих знань. Фундаментальна мета ML полягає у створенні таких обчислювальних методів, які демонструють автоматичне вдосконалення через досвід, мінімізуючи необхідність у явному (експліцитному) програмуванні правил.

У парадигмі машинного навчання, алгоритми проходять етап навчання на тренувальному наборі даних, який містить вхідні змінні (ознаки) та відповідні їм цільові змінні (мітки). Алгоритми використовують ці дані для виявлення прихованих закономірностей, взаємозв'язків та статистичних кореляцій, що дає змогу побудувати прогностичні моделі. Ці моделі можуть бути використані для ефективного прогнозування вихідних значень для нових, раніше не бачених даних. Ефективність та узагальнювальна здатність моделей згодом підтверджується шляхом валідації на окремих, незалежних тестових наборах даних.

Існує таксономія методів машинного навчання, яка включає: навчання з учителем (Supervised Learning), навчання без учителя (Unsupervised Learning), напівконтрольоване навчання (Semi-supervised Learning) та навчання з підкріпленням (Reinforcement Learning).

Навчання з учителем передбачає тренування алгоритмів на позначених (мічених) даних. Навчання без учителя оперує непозначеними даними. Напівконтрольоване навчання є гібридним підходом, що комбінує мічені та

немічені дані. Навчання з підкріпленням сфокусоване на навчанні агентів приймати рішення в динамічному середовищі з метою максимізації кумулятивного сигналу винагороди.

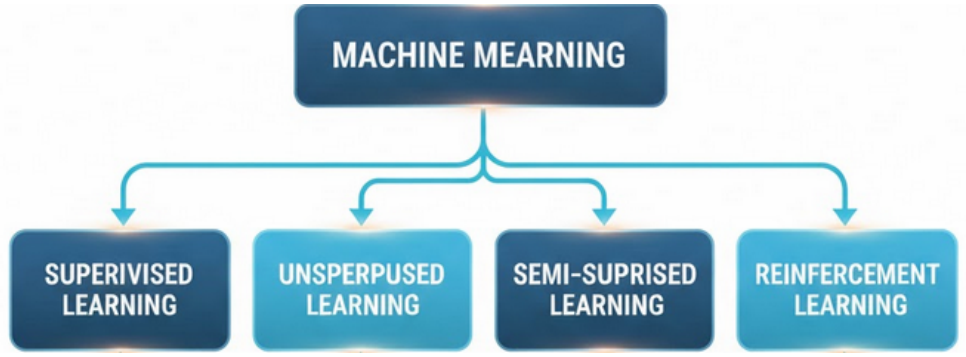

Рис. 2.1. Таксономія машинного навчання

У контексті даної роботи для вилучення інсайтів та знань із даних застосовуються методи машинного навчання з учителем та без учителя.

Навчання з учителем — це методологія ML, при якій алгоритми тренуються на базі даних, де кожному вхідному екземпляру відповідає відоме вихідне значення або цільова змінна. Основна мета полягає у вивченні функції відображення (mapping function), яка ефективно зв'язує вхідні змінні з вихідною змінною. Ця функція потім використовується для точного прогнозування результатів на нових, неспостережуваних даних.

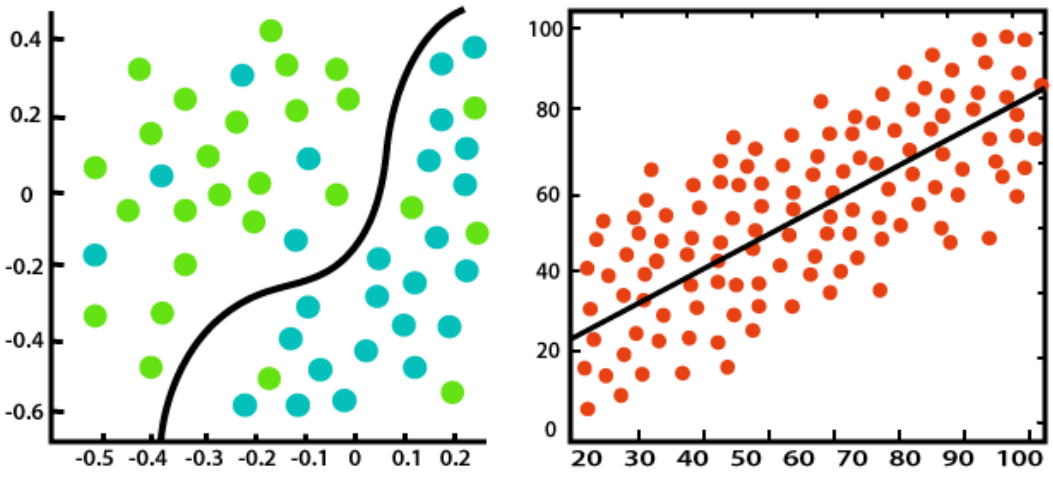

Рис. 2.2. Класифікація та регресія в машинному навчання

Існують дві основні категорії задач у навчанні з учителем: класифікація та регресія.

На рисунку 2.2 (ліворуч) представлено набір даних, який вимагає застосування класифікації для ефективного поділу екземплярів на окремі класи на основі їхніх параметрів. Праворуч показано кореляції між залежними та незалежними змінними для прогнозування безперервних змінних (регресія), наприклад, прогнозування ринкових тенденцій або, як поширений навчальний приклад, прогнозування академічної успішності (успіху/невдачі) студента на іспиті залежно від комбінації годин сну та годин, присвячених навчанню.

2.1.1. Алгоритми класифікації

Класифікація — це процес, при якому алгоритм навчається прогнозувати дискретний клас або категорію, до якої належить вхідний екземпляр, на основі його ознак. Існує кілька алгоритмів класифікації, які широко використовуються.

Дерева рішень (Decision Tree) являють собою ефективний інструмент, який імітує процес людського прийняття рішень через послідовність умовних запитань та відповідей. Ця ієрархічна структура використовується для класифікації або прогнозування результату на основі вхідних ознак. Зі зростанням складності проблеми, дерево рішень стає глибшим та ширшим (рис. 2.3).

Рис. 2.3. Дерева рішень

Класифікатор "Випадковий ліс" (Random Forest) є потужним методом ансамблевого навчання (Ensemble Learning), призначеним для задач

класифікації. Його архітектура полягає у формуванні великої кількості незалежних дерев рішень. Результати (прогнози) цих індивідуальних дерев агрегуються (об'єднуються), наприклад, через голосування, для отримання остаточного, більш надійного прогнозу (рисунок 2.4). Ансамблеве навчання є технікою об'єднання прогнозів кількох моделей ML з метою підвищення їх загальної продуктивності та точності.

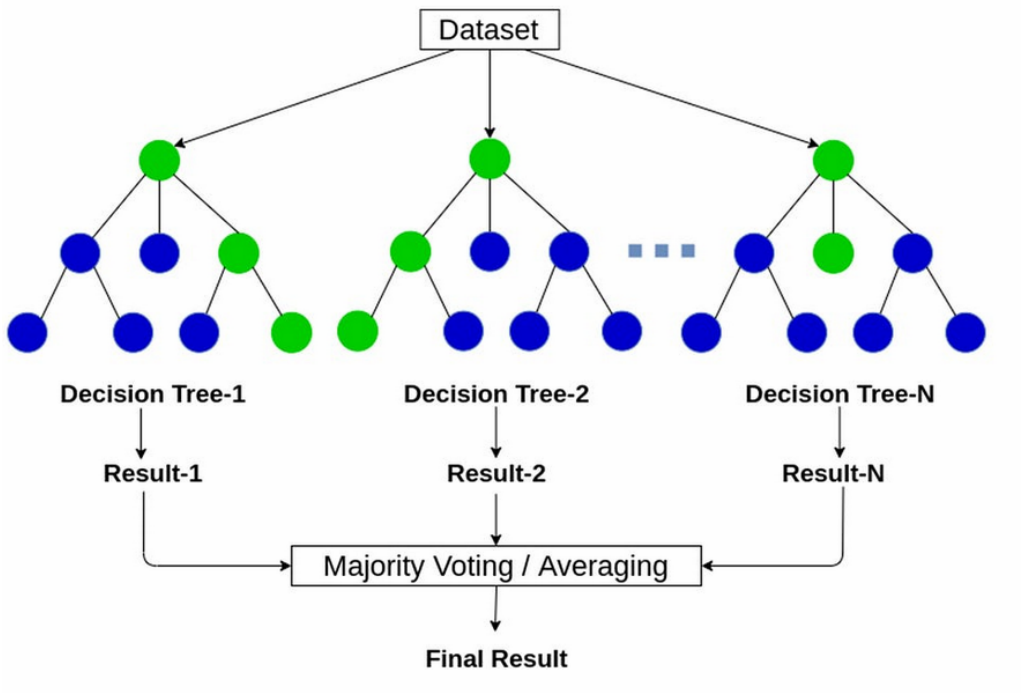

Рис. 2.4. Класифікатор "Випадковий ліс"

Наївний Байєс (Naive Bayes) — це алгоритм навчання з учителем, ефективний для багатокласової класифікації. В його основі лежить імовірнісна модель, яка обчислює умовні ймовірності вхідних ознак і присвоює розподіли ймовірностей кожному з можливих цільових класів, виходячи з теореми Байєса та припущення про умовну незалежність ознак. Ключові переваги цього алгоритму включають простоту реалізації та високу швидкість навчання [11].

Машини опорних векторів (Support Vector Machines, SVM) як проілюстровано на рисунку 2.5 , SVM є дискримінативним класифікатором, який працює шляхом знаходження оптимальної гіперплощини, що розділяє

класи даних. Його мета — максимізувати поле (margin) між гіперплощиною та найближчими точками даних (опорними векторами), що підвищує стійкість моделі до нових даних. Цей підхід забезпечує високу точність [9].

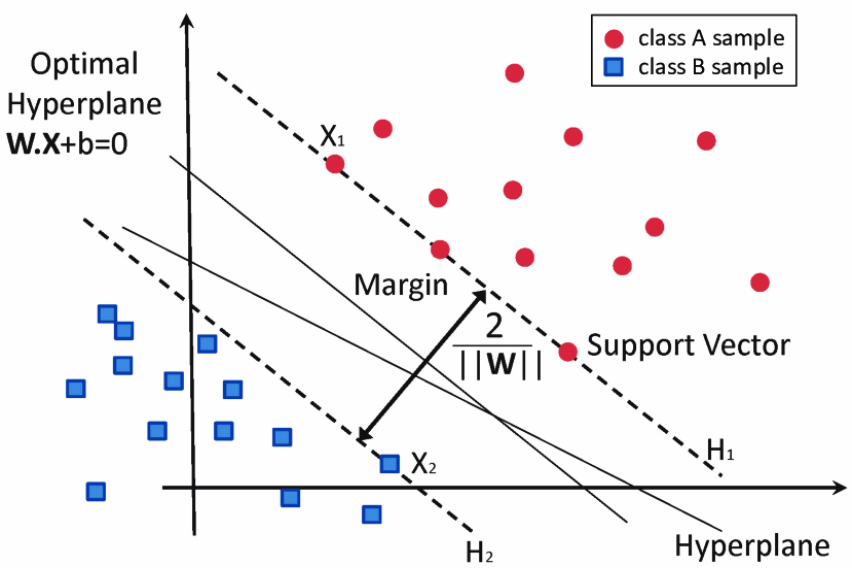

Рис. 2.5. Алгоритм класифікації Support Vector Machines

Алгоритм K-найближчих сусідів (K-Nearest Neighbors, KNN) використовується як для задач класифікації, так і для регресії [13]. Він є непараметричним методом, що не вимагає припущень щодо розподілу даних. KNN функціонує шляхом ідентифікації k найближчих навчальних екземплярів до нового вхідного елемента даних.

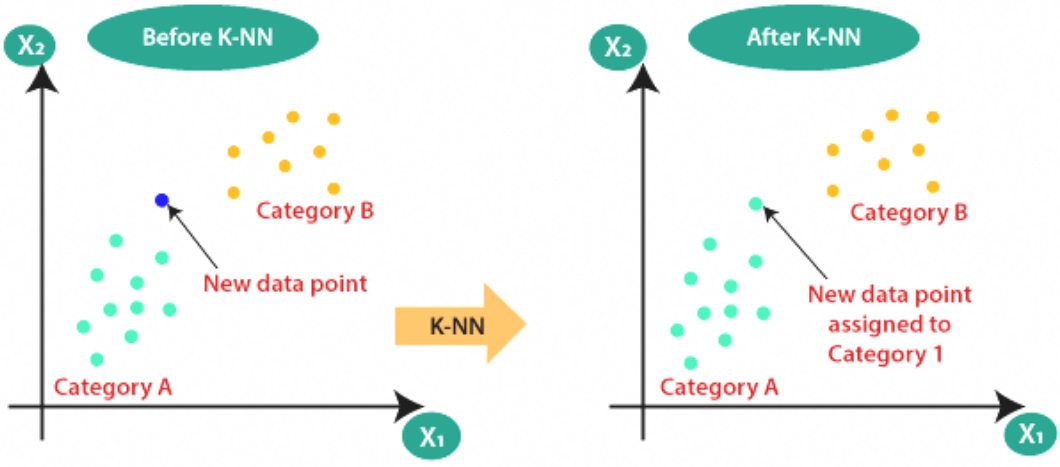

Рис. 2.6. Алгоритм K-найближчих сусідів

Класифікація нового елемента здійснюється на основі переважаючого класу (мажоритарне голосування) серед цих k найближчих сусідів. Значення k є гіперпараметром, який підлягає налаштуванню для оптимізації продуктивності. Хоча KNN є простим та інтуїтивно зрозумілим, він може бути обчислювально витратним для великих наборів даних, а вибір оптимального k суттєво впливає на точність. KNN знаходить широке застосування, зокрема у розпізнаванні зображень, класифікації текстів та рекомендаційних системах.

2.1.2 Алгоритми регресії

Регресія — це тип навчання з учителем, при якому алгоритм навчається прогнозувати числове значення або безперервну вихідну змінну на основі вхідних ознак.

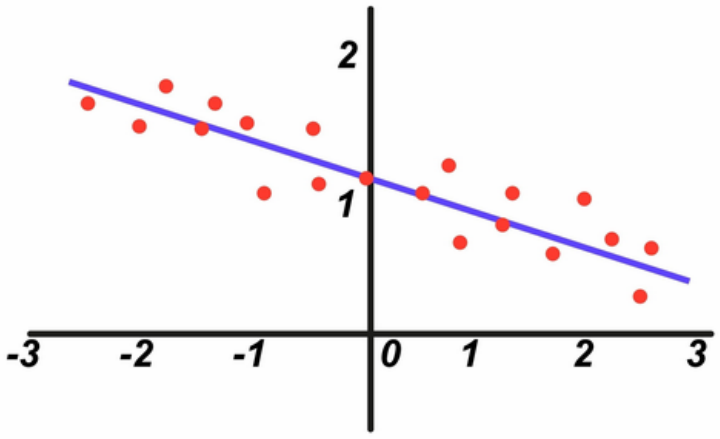

Рис. 2.7. Лінійна регресія

Лінійна регресія (Linear Regression) використовується для прогнозування безперервної вихідної змінної на основі однієї або кількох вхідних змінних (незалежних), які можуть бути безперервними або категоріальними [15]. Вона базується на припущенні про існування лінійного зв'язку між вхідними та вихідною змінними. Метою є знаходження лінії найкращого відповідності, яка описує цей лінійний зв'язок і мінімізує суму

квадратів помилок (Residual Sum of Squares, RSS) між прогнозованими та фактичними значеннями (рис. 2.7).

Логістична регресія (Logistic Regression) застосовується для прогнозування бінарної вихідної змінної (наприклад, 0 або 1) на основі однієї або кількох вхідних змінних [16]. Цей метод моделює ймовірність бінарного результату як функцію вхідних змінних шляхом застосування логістичної (сигмоїдної) функції. Вона вивчає лінійний зв'язок у наданому наборі даних, а потім вводить нелінійність через сигмоїдну функцію для моделювання ймовірності бінарного результату (рис. 2.8).

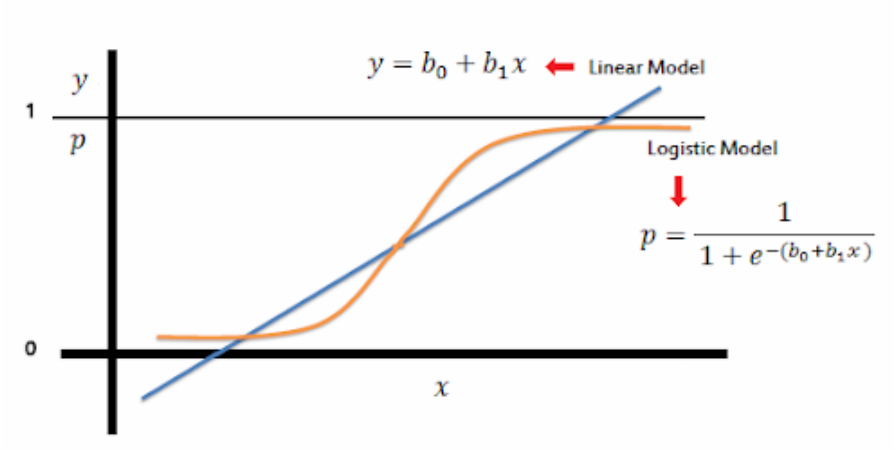

Рис. 2.8. Логістична та лінійна регресії

2.2. Особливості машинного без учителя

Навчання без учителя є парадигмою машинного навчання, при якій алгоритми тренуються на непозначених (немічених) даних, тобто вхідні екземпляри не супроводжуються відомими вихідними значеннями або цільовими змінними. Фундаментальна мета навчання без учителя полягає у виявленні прихованих закономірностей, структурних зв'язків та внутрішньої організації даних для ефективного групування подібних об'єктів (екземплярів).

Застосування навчання без учителя є доцільним за таких умов:

- Відсутність цільової мітки для прогнозування. У випадках, коли мітка для прогнозування відсутня. Як приклад, можна розглянути аналіз медичних зображень (наприклад, сканувань мозку) з метою ідентифікації областей, що потенційно вказують на патологічні зміни. Оскільки на зображеннях немає попередньо визначених міток, пряма класифікація є ускладненою. Однак, алгоритм може кластеризувати (групувати) регіони зображення на основі їхньої внутрішньої подібності чи відмінності у текстурі, інтенсивності або формі. Це дає змогу виявити аномальні зони, які можуть бути індикаторами проблем зі здоров'ям.

- Групування даних замість прогнозування, коли першочерговою метою є не прогноз мітки, а структуризація та узагальнення даних шляхом об'єднання подібних екземплярів. Класичним прикладом є ситуація, коли дослідник має велику розмірність ознакового простору і прагне редукувати його до меншого, більш інформативного набору ознак для подальшого аналізу або введення в інші прогностичні моделі.

Два основні типи навчання без учителя: кластеризація та зменшення розмірності.

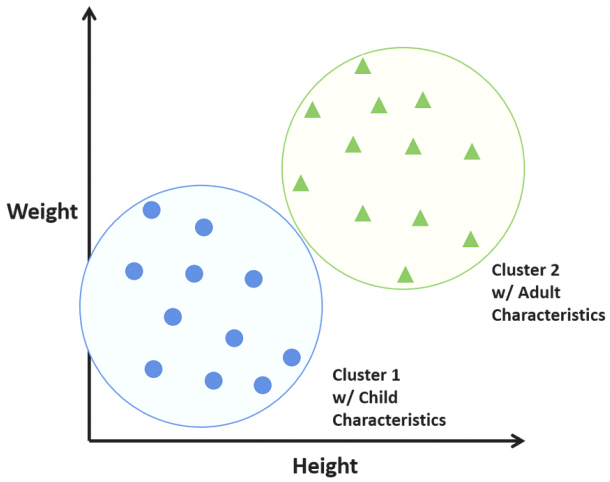

Рис. 2.9. Алгоритм кластеризації

Кластеризація (Clustering) — це процес групування подібних точок даних у кластери (сховища). Це групування здійснюється на основі

внутрішніх властивостей (intrinsically properties) або метрики подібності між екземплярами. Основна мета кластеризації — автономно виявити закономірності та структури у даних без попереднього знання про їхні класові мітки або цільові вихідні змінні. До основних алгоритмів кластеризації належать: k-середні (k-means), ієрархічна кластеризація (Hierarchical Clustering) та кластеризація на основі щільності (Density-Based Spatial Clustering of Applications with Noise, DBSCAN).

Зменшення розмірності (Dimensionality Reduction) — це методологія, спрямована на редукцію кількості вхідних ознак (вимірів) у наборі даних при одночасному збереженні найбільш критичної інформації або структури даних. Ця техніка слугує подвійній меті: спрощення представлення даних та покращення обчислювальної ефективності наступних етапів аналізу або навчання моделей.

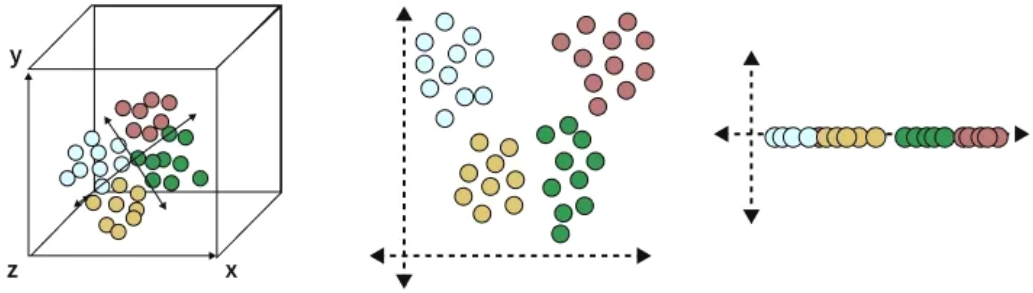

Рис. 2.10. Методологія зменшення розмірності

Приклади методів зменшення розмірності включають: аналіз головних компонентів (Principal Component Analysis, PCA), випадкова проекція (Random Projection) та аналіз незалежних компонентів (Independent Component Analysis, ICA).

Як показано на рисунку 2.10, випадкова проекція є методом, який використовується для зменшення кількості вимірів у наборі даних із збереженням його структурної цілісності. Підхід ґрунтується на проектуванні

початкових багатовимірних даних на випадково вибраний підпростір меншої розмірності.

Цю проекцію можна інтерпретувати як лінійне перетворення, ключовим принципом якого є збереження відстаней (або евклідової метрики) між точками даних настільки, наскільки це можливо. Порівняно з PCA, випадкова проекція визнана обчислювально більш ефективним методом для високорозмірних та великих наборів даних.

Розмірність трансформованих даних визначається через параметр похибки є, який функціонує як регулятор. є контролює допустиму кількість спотворення відстаней, або, іншими словами, обсяг інформації та структури початкового набору даних, який необхідно зберегти у редукованому просторі.

2.3. Визначення, загрози та класифікація апаратних троянів

Апаратний троян (Hardware Trojan, HT) визначається як зловмисна, несанкціонована модифікація, внесена до інтегральної схеми (IC) або її архітектури протягом будь-якого етапу виробничого процесу. Така модифікація має потенціал підриву безпеки та порушення функціональної цілісності апаратного забезпечення.

Апаратні трояни (AT) можуть бути імплантовані різними суб'єктами:

- Інсайдери, співробітники виробничих або проєктних організацій, або підрядники, які мають привілейований доступ до схем.

- Зовнішні зловмисники - особи, які отримують несанкціонований доступ до виробничих потужностей або даних.

AT можуть набувати різних форм реалізації:

- Модифікація логіки/функціональності, тобто зміна початкового призначення логічних вентилів або функціональних блоків.

- Вставка додаткової схемотехніки, а саме додавання прихованих, несанкціонованих логічних елементів (комбінаційних або послідовних схем).

- Зміна фізичних характеристик, модифікація електричних параметрів, таких як характеристики живлення, споживання струму або параметри синхронізації (timing).

Після успішної імплантації, АТ можуть бути активовані дистанційно або внутрішніми умовами для виконання широкого спектру шкідливих дій, зокрема: крадіжка конфіденційних даних, зміна алгоритмічної поведінки системи або повне виведення пристрою з ладу (kill functionality).

Апаратні трояни являють собою критичну загрозу безпеці у системах, де цілісність апаратного забезпечення є імперативом. До таких критичних застосувань належать: військово-промисловий комплекс, аерокосмічні системи, фінансові інфраструктури та інші об'єкти критичної інфраструктури. З огляду на це, розробка ефективних методів виявлення та превентивних заходів проти АТ є активною та пріоритетною галуззю досліджень у сферах безпеки апаратного забезпечення та довіреного виробництва (Trusted Fabrication).

2.3.1. Характеристика та класифікація апаратних троянів

На рисунку 2.11 проілюстровані різноманітні методи класифікації апаратних троянів, які базуються на їхніх внутрішніх характеристиках та зовнішній поведінці. Класифікація охоплює фізичні характеристики, механізм активації та фазу дії (ефект).

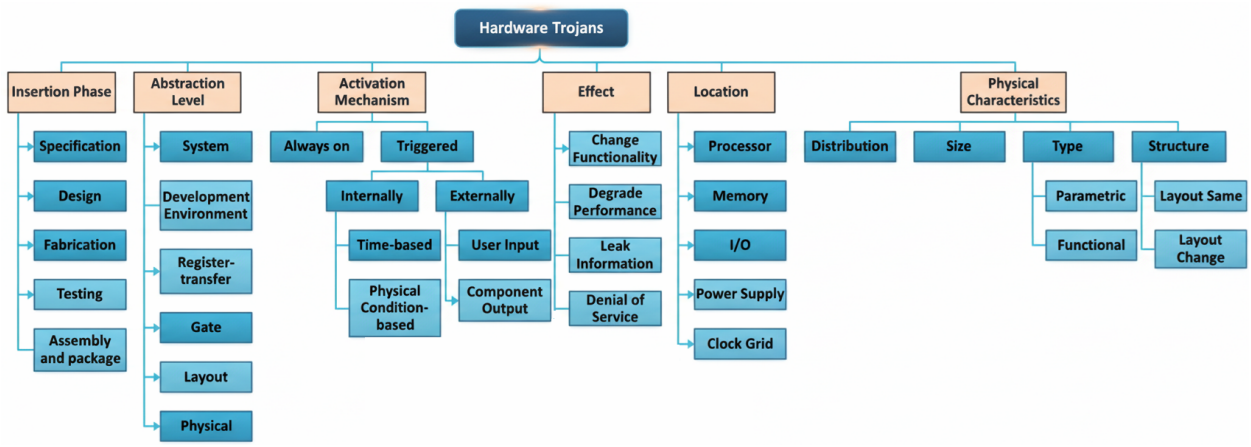

Рис. 2.11. Таксономія апаратних троянів

Трояни класифікуються на основі їхнього фізичного впливу на оригінальну схему, який може бути функціональним або параметричним:

а) Функціональний троян (functional trojan): цей тип трояну передбачає додавання або видалення логічних вентилів або тригерів з початкового дизайну. Це дозволяє зловмиснику або отримати несанкціонований доступ, або модифікувати логічну поведінку пристрою. Прикладом функціонального трояна є вставка шкідливого коду або мікрокоду в блок пам'яті мікропроцесора (Microprocessor).

б) Параметричний троян (Parametric Trojan) модифікує існуючу схемотехніку, не додаючи повністю нову логіку, але змінюючи фізичні властивості елементів. Приклади включають: навмисне розрідження (thinning) провідників або ін'єкцію радіації в певні області кристала з метою зниження надійності (reliability degradation) або прискорення старіння (aging) чипа.

2.3.2. Класифікація за механізмом активації

Апаратні трояни активуються специфічною подією або умовою, відомою як механізм активації (trigger mechanism). Цей механізм може бути класифікований за такими факторами, як послідовність вхідних даних, певні робочі умови або тимчасові параметри. Існують два основні методи активації АТ:

а) Внутрішньо активований троян (Internally Triggered HT). У цьому випадку шкідлива логічна схемотехніка, вбудована в чип, активує корисне навантаження після виникнення певної внутрішньої умови або досягнення певного часового періоду (наприклад, лічильник подій), як це було проілюстровано на рисунку 1.3.

б) Зовнішньо активований троян (Externally Triggered HT) містить шкідливу логіку, яка використовує зовнішній інтерфейс (наприклад, приховану антену або спеціалізовані сенсори) для отримання команд від зовнішнього супротивника. Це дозволяє зловмиснику дистанційно

отримувати доступ та маніпулювати дизайном. Прикладом є троян, вбудований у систему управління крилатою ракетою, що дозволяє ворогу дистанційно керувати або виводити з ладу систему [4].

Апаратні трояни можуть бути класифіковані за їх ефектами, тобто поведінкою, яку демонструє троян після активації. Ця поведінка може включати крадіжку або модифікацію даних, порушення нормальної роботи системи, відмову в обслуговуванні або несанкціонований доступ до апаратного забезпечення.

2.4. Огляд наборів даних та ознак для детекції апаратних троянів

Ефективне виявлення апаратних троянів за допомогою методів машинного навчання (ML) вимагає наявності репрезентативних наборів даних. Ці набори даних повинні включати як еталонні (чисті) схеми, що слугують основою для нормальної моделі поведінки, так і скомпрометовані схеми з імплантованими АТ, які є цільовими об'єктами для детекції. Крім того, критично важливим для успіху алгоритму ML є вибір адекватного набору ознак, які достовірно відображають структурну та функціональну поведінку ІС. У цьому розділі представлено огляд часто використовуваних наборів даних та ознак у галузі виявлення НТ.

Експериментальний набір даних, використаний у цій роботі, був сконструйований на базі бенчмарків апаратних троянів. Зокрема, були використані схеми з довірчого бенчмарку [7]. Цей бенчмарк являє собою тестові схеми (представлені на рівнях RTL, вентилів або фізичного компонування), які містять навмисно імплантовані трояни. Впровадження АТ здійснюється у важкодоступних, значущих та/або вигідних місцях (наприклад, незвичайні вузли, вільний простір компонування), що підвищує їхню стійкість до виявлення. Трояни, інтегровані в набір, варіюються за типом та функціональністю, включаючи приховані, комбінаційні та послідовні АТ, і були вставлені на різних етапах процесу проєктування.

Набір даних надає різноманітні набори ознак, які є необхідними для аналізу: кількість комірок, кількість буферів/інверторів, загальна площа комірок, витік потужності та вимірювання динамічного споживання потужності. Загальний набір даних охоплює 907 цифрових схем, з яких 21 схема є чистими (без троянів), а 886 схем — зараженими АТ.

2.5. Огляд літератури по темі дослідження

Виявлення АТ є постійною конкурентною боротьбою (ongoing cat-and-mouse game). Кожна пропозиція нового методу виявлення, здатного ідентифікувати поточні НТ, стимулює розробку нових, більш витончених троянів, які здатні обійти існуючі механізми захисту.

Один із ранніх механізмів захисту проти АТ був запропонований в [7] і відомий як механізм ідентифікації невикористаних схем (Unused Circuit Identification, UCI). Цей метод був призначений для виявлення підозрілих схем під час верифікації дизайну. Проте, вже через рік, в [8] запропонували новий дизайн прихованої та шкідливої схеми (Stealthy and Malicious Circuitry, SMC), яка обходить метод UCI шляхом приховування НТ у майже невикористаній логіці.

В дослідженні [9] представили метод детекції, сфокусований на виявленні троянів, що витікають інформацію (information-leaking Trojans). Їхня техніка успішно генерувала умову спрацьовування для трояна і змогла виявити витік криптографічного ключа для шифру AES-600, але виявилася неефективною для детекції витоку ключа AES-T1200.

Незважаючи на значну кількість запропонованих у літературі методів виявлення АТ, існує нагальна потреба у розробці надійного та ефективного підходу, здатного ідентифікувати нові, невідомі апаратні трояни.

В [10] було наголошено на критичній важливості застосування навчання з учителем та без учителя у сфері безпеки IoT, а також

обговорювалися обмеження різних ML-методів через ризик надмірного підлаштування (overfitting) прогностичних моделей.

У роботі [11] був представлений метод ML для виявлення АТ на основі аналізу споживання потужності (power consumption analysis). Подібним чином, у [12] було розроблено спеціалізовану багатопроцесорну платформу, яка використовувала класифікатор на основі SVM (Support Vector Machine) з навчанням з учителем для виявлення комунікаційних атак, активованих НТ, досягнувши точності 94%. Обмеженням цих підходів є їхня фокусування виключно на виявленні НТ на етапі експлуатації (run-time), без передбачення витягування ознак троянів безпосередньо зі списку з'єднань дизайну. Аналіз на рівні списку з'єднань (gate-level netlist) є більш превентивним та вичерпним, оскільки він містить повну інформацію про з'єднання вентилів та IP-ядер, включаючи функціональну та часову поведінку.

Деякі роботи зосереджувались на методах кластеризації. У [13] використовувалась кластеризація на основі ентропії інформації, де для детекції троянів було встановлено поріг ознак, а у [14] застосовувалась модель DBSCAN без попереднього встановлення порогу ознак для виявлення АТ. У [15] запропоновано інший метод кластеризації на основі нечіткої логіки для криптографічних застосувань.

Автори робіт [13 - 25] застосовували свої методики лише до обмеженої кількості типів схем АТ і стикалися з проблемою низької точності, головним чином через індуковану лінійність (мультиколінеарність) між ознаками, спричинену масштабуванням набору даних у процесі підготовки для ML.

У [16] було витягнуто п'ятдесят одну ознаку трояна, а 11 найкращих ознак були вибрані вручну для введення в алгоритм класифікатора випадкового лісу (Random Forest). Цей підхід досяг точності лише 74,6% на обмеженій вибірці з 12 бенчмарків.

Автори в [17] сфокусувалися на докремнієвому виявленні АТ (pre-silicon detection) за допомогою ML-моделі. Вони екстрагували ознаки списку з'єднань із пост-синтезного дизайну та використовували алгоритм SVM

(навчання з учителем) для розрізнення чистих та заражених дизайнів. Проте, через індуковану лінійність між ознаками, їхня ефективність була обмеженою, досягнувши рівня істинно позитивних результатів (True Positive Rate, TPR) 85,28%.

У [18] була застосована модель нейронної мережі з одинадцятьма ознаками списку з'єднань. Цей підхід виявився неефективним через аналогічну проблему індукованої лінійності, що призвело до низької точності істинно негативних результатів (True Negative Rate, TNR) 59,5%.

На противагу цьому, фреймворк RG-Secure [19] реалізував легкий алгоритм градієнтного бустингу (gradient boosting) для одночасного виявлення HT на рівні регістрових передач та списку з'єднань. Хоча його точність була високою, його ефективність була доведена лише для обмеженої кількості апаратних дизайнів, причому в одному випадку рівень виявлення HT опускався нижче 60%.

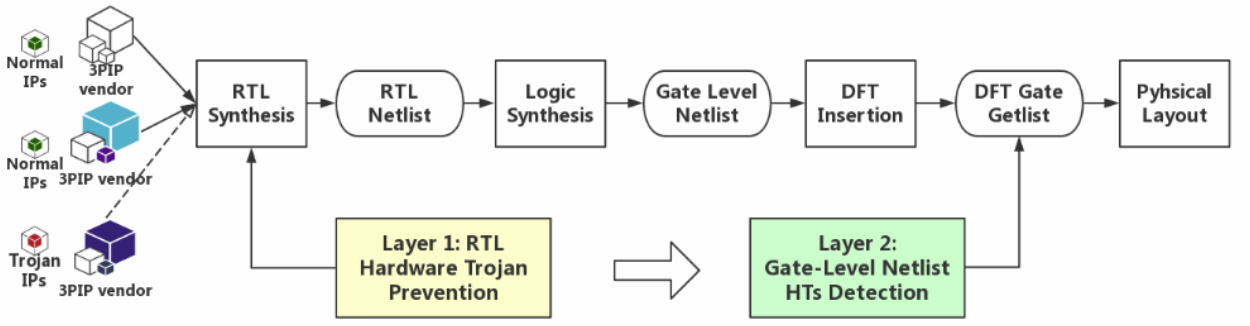

Рис. 2.12. Огляд фреймворку RG-Secure

З метою побудови довірених систем на кристалі (SoCs), було інтегровано низку стратегій безпеки у фазу розширених інтегрованих операцій до присвоєння оператора. Рисунок 2.12 ілюструє загальну архітектуру RG–Secure фреймворку.

Як показано на рисунку, першим рівнем нашого підходу є стратегії проєктування для запобігання апаратним троянам на рівні RTL. Типово, постачальник інтелектуальної власності (IP-вендор) реалізує продаж кількох

сторонніх IP-ядер (3PIP). Зловмисні IP-вендори можуть імплантувати апаратні трояни у свої продукти (Троянські IP-ядра). Вони часто проєктують розподілений HT (distributed HT), координуючи механізми активації (тригери) троянів, розташовані у різних IP-ядрах.

Нарешті, у [20] були запропоновані багатошарові нейронні мережі зворотного поширення та одномірна SVM для виявлення HT, що використовуються для витоку інформації, та визначення їхньої локалізації. Цей підхід досяг TPR 85,05% та TNR 73,91%. Крім того, у [31] було представлено алгоритм ML без учителя, названий PL-HTD, який поєднує Аналіз Головних Компонентів (PCA) та алгоритм Локального Фактора Викидів (Local Outlier Factor, LOF) для детекції HT на рівні списку з'єднань. Через надмірне підлаштування навчального набору даних цей підхід продемонстрував лише середній TPR на рівні 42,42%.

Висновки до розділу

У другому розділі було здійснено комплексне дослідження сучасних алгоритмів машинного навчання, застосовуваних для виявлення апаратних троянів у цифрових схемах. Проведено класифікацію методів контрольованого та неконтрольованого навчання, що дозволяє обґрунтовано вибирати алгоритми залежно від типу даних та доступності маркування. Встановлено, що алгоритми з учителем (Random Forest, Decision Tree, SVM, Neural Networks) показують високу ефективність при наявності збалансованих і структуровано описаних наборів ознак. Методи без учителя, такі як кластеризація або зниження розмірності, ефективні для виявлення прихованих закономірностей у даних без попереднього маркування.

Розглянуто особливості формування наборів даних для задачі детекції апаратних троянів, включно з процесом екстракції ознак із нетлістів та описів HDL. Показано, що різні типи ознак — топологічні, електричні, часові — мають різну вагу для моделі класифікації, що вимагає застосування методів

попередньої нормалізації та відбору ознак. Здійснено аналіз публікацій і базових підходів, який засвідчив тенденцію до зростання ролі комбінованих методів, що поєднують статистичний аналіз із навчанням глибоких моделей.

Було визначено, що основною проблемою існуючих систем детекції є обмежена здатність виявляти слабко виражені, нелінійні залежності між ознаками. Саме це зумовило необхідність формування нового напряму досліджень — підсилення класифікаторів шляхом зниження лінійності ознак. У результаті проведеного аналізу сформульовано вимоги до побудови узагальненої моделі, здатної працювати з нелінійно трансформованими ознаками.

РОЗДІЛ 3. МЕТОДОЛОГІЯ ВИЯВЛЕННЯ АПАРАТНИХ ТРОЯНІВ ШЛЯХОМ ЗНИЖЕННЯ ЛІНІЙНОСТІ ОЗНАК ЗАСОБАМИ МАШИННОГО НАВЧАННЯ

3.1. Особливості мови опису апаратного забезпечення

Цей розділ присвячений детальному опису технічних підходів, застосованих у даному дослідженні. Основний фокус зосереджено на:

- Техніці екстракції ознак з апаратного дизайну інтегральної схеми (IC).

- Обґрунтуванні вибору конкретних алгоритмів машинного навчання (ML) для ефективного виявлення апаратних троянів (HT), імплантованих у дизайн.

Мови опису апаратного забезпечення (HDL) є спеціалізованими мовами програмування, які є фундаментальними інструментами у процесі проєктування цифрових схем. Їхнє призначення полягає в описі поведінки (behaviour) та структури (structure) цифрових схем на різних рівнях абстракції: від високорівневого системного проєктування до низькорівневих списків з'єднань на рівні вентилів (gate-level netlists).

Verilog та VHDL є найбільш поширеними HDL, які широко використовуються в напівпровідниковій промисловості. Verilog часто використовується для проєктування на рівні RTL. VHDL (Very High Speed Integrated Circuit Hardware Description Language) - потужна мова, часто використовується у військових та аерокосмічних застосуваннях. Вони застосовуються для:

- Моделювання (Modelling) - створення абстрактного представлення цифрових систем.

- Симуляції (Simulation) - перевірці функціональної коректності дизайну перед його фізичною реалізацією.

- Синтезу (Synthesis) - трансформації поведінкового або структурного опису HDL у фізичну апаратну реалізацію.

Використання HDL надає розробникам можливість ефективно описувати складні цифрові схеми та забезпечує засіб верифікації функціональності до початку дорогого та тривалого етапу фізичного виробництва. Таким чином, HDL є незамінним компонентом процесу цифрового проєктування та критично важливим для розробки та верифікації ІС.

У контексті даної роботи, HDL є вихідною точкою. Після написання дизайну на HDL (наприклад, Verilog), він проходить синтез для отримання списку з'єднань (netlist). Саме цей список з'єднань стає джерелом ознак для моделі машинного навчання, що використовується для виявлення апаратних троянів.

3.2. Процес видобування ознак та схема виявлення апаратних троянів

На рисунку 3.1 представлена запропонована схема виявлення апаратних троянів, яка складається з послідовності чітко визначених етапів.

Крок 1: Синтез дизайну (Synthesis)

Процес розпочинається з етапу логічного синтезу. Для його виконання використовується скрипт на мові Tcl (Tool Command Language), метою якого є трансформація поведінкового опису мовою Verilog у низькорівневий список з'єднань (netlist). Синтез здійснюється з обов'язковою перевіркою відповідності цифрового апаратного дизайну заданим обмеженням (constraints) щодо часу (timing), потужності (power) та площі (area).

Порушення цих обмежень може статися, якщо дизайн не відповідає максимально допустимій затримці, ліміту споживання потужності або обмеженням на фізичну площу кристала. Скрипт Tcl, використаний для цієї мети, встановлює період тактової частоти на рівні 20 000 пікосекунд (пс), що відповідає робочій частоті 50 МГц, та визначає використання технологічного пакету 45 нм.

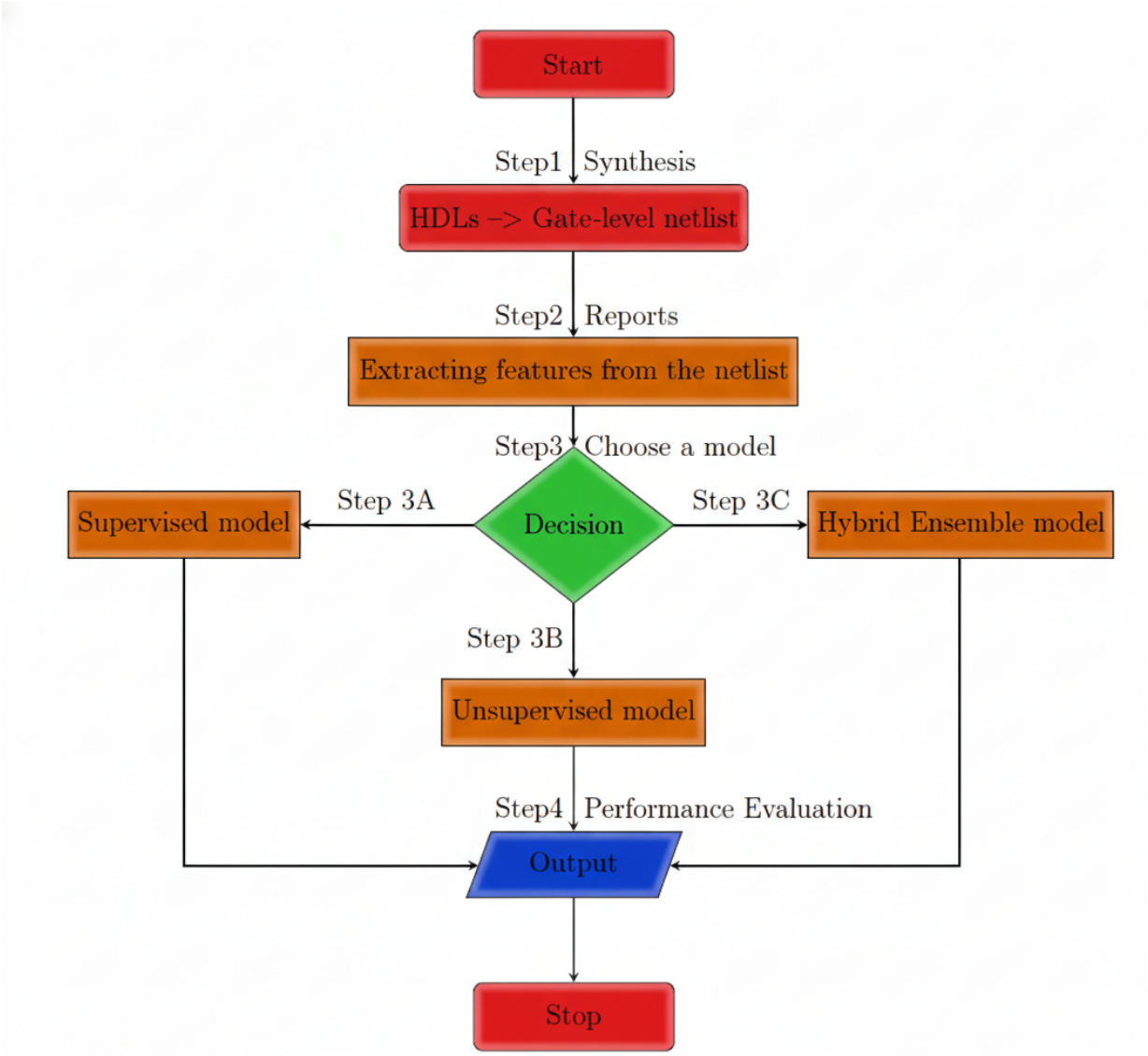

Рис. 3.1. Алгоритмічна схема виявлення апаратних троянів

Крок 2: Видобування ознак (Feature Extraction)

Для отримання детальних метрик дизайну використовувався інструмент Cadence Genus. За допомогою цього інструменту були згенеровані комплексні звіти, які деталізують часові, потужнісні та площові характеристики синтезованого дизайну (як показано на рисунках 3.2, 3.3, 3.4 та 3.5).

Кожна витягнута ознака формується шляхом обчислення підмножин компонентів, що використовуються в дизайні, як це систематизовано у таблиці 3.1. Загалом, для побудови бази даних, яка буде використана для навчання моделей машинного навчання з метою класифікації дизайну як

зараженого трояном або чистого, екстрагується тридцять одна ознака списку з'єднань.

```
  Module:                 aes_128
  Operating conditions:   PVT_1P1V_0C (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
  Description:            AES_100 (Trojan Free)
============================================================


  Gate   Instances    Area       Library
-----------------------------------------------
DFFQXL         128   700.416    fast_vdd1v0
INVXL          128    87.552    fast_vdd1v0
SDFFQX1        128   963.072    fast_vdd1v0
-----------------------------------------------
total          384  1751.040


     Type      Instances   Area   Area %
-----------------------------------------
sequential           256 1663.488   95.0
inverter             128   87.552    5.0
unresolved            20    0.000    0.0
physical_cells         0    0.000    0.0
-----------------------------------------
total                404 1751.040  100.0
```

Рис. 3.2. Звіт про площу вихідних даних синтезу для схеми AES-T100 без трояна

```
  Module:                 top
  Operating conditions:   PVT_1P1V_0C (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
  Description:            AES_100 (Trojan Infected)
============================================================

                           Leakage     Dynamic       Total
    Instance        Cells Power(nW)   Power(nW)    Power(nW)
------------------------------------------------------------------
top                184858 18560.028 11853063.392 11871623.421
  AES_rf_S4_1_S_0     494    47.474    22572.666    22620.140
  AES_rf_S4_2_S_1     494    47.474    22572.666    22620.140
  AES_rf_S4_3_S_2     494    47.474    22572.666    22620.140
  AES_rf_S4_2_S_0     494    47.474    22482.853    22530.327
  AES_rf_S4_3_S_1     494    47.474    22572.666    22620.140
  AES_rf_S4_4_S_2     494    47.474    22572.666    22620.140
  AES_rf_S4_3_S_0     494    47.474    22482.853    22530.327
  AES_rf_S4_4_S_1     494    47.474    22572.666    22620.140
  AES_rf_S4_1_S_2     494    47.474    22572.666    22620.140
  AES_rf_S4_4_S_0     494    47.474    22482.853    22530.327
  AES_rf_S4_2_S_2     494    47.474    22572.666    22620.140
  AES_r7_t0_t3_s4     498    47.456    21375.273    21422.729
  AES_r5_t2_t1_s4     498    47.456    21375.273    21422.729
  AES_r6_t0_t0_s4     498    47.456    21375.273    21422.729
  AES_r6_t2_t0_s4     498    47.456    21375.273    21422.729
  AES_r6_t3_t0_s4     498    47.456    21375.273    21422.729
  AES_r3_t2_t1_s4     498    47.456    21375.273    21422.729
  AES_r9_t2_t0_s4     498    47.456    21375.273    21422.729
  AES_r4_t1_t2_s4     498    47.181    21296.682    21343.862
  AES_r4_t3_t2_s4     498    47.181    21057.174    21104.355
  AES_r4_t0_t2_s4     498    47.181    21296.682    21343.862
  AES_r2_t1_t2_s4     498    47.181    21296.682    21343.862
```

Рис. 3.3. Звіт про потужність вихідних даних синтезу для схеми AES-T100, зараженої трояном

```
  Module:                 aes_128
  Operating conditions:   PVT_1P1V_0C (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
==============================================================


  Gate   Instances    Area        Library
-----------------------------------------------
DFFQXL        2256  12344.832    fast_vdd1v0
INVX1          320    218.880    fast_vdd1v0
INVXL          768    525.312    fast_vdd1v0
MX2XL           32     76.608    fast_vdd1v0
MXI2XL        1152   2757.888    fast_vdd1v0
SDFFQX1        432   3250.368    fast_vdd1v0
XNOR2X1        752   1800.288    fast_vdd1v0
-----------------------------------------------
total         5712  20974.176


     Type      Instances    Area   Area %
-----------------------------------------
sequential          2688 15595.200   74.4
inverter            1088   744.192    3.5
unresolved            20     0.000    0.0
logic               1936  4634.784   22.1
physical_cells         0     0.000    0.0
-----------------------------------------
total               5732 20974.176  100.0
```

Рис. 3.4. Звіт про логічні вентилі вихідних даних синтезу для схеми AES-T1000 без трояна

```
  Module:                 aes_128
  Operating conditions:   PVT_1P1V_0C (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================


Path 1: MET (19829 ps) Setup Check with Pin a10/k3a_reg[29]/CK->D
     Startpoint: (R) a9/out_1_reg[93]/CK
          Clock: (R) 50MHz
       Endpoint: (R) a10/k3a_reg[29]/D
          Clock: (R) 50MHz

                    Capture       Launch
       Clock Edge:+   20000            0
       Src Latency:+      0            0
       Net Latency:+      0 (I)        0 (I)
           Arrival:=  20000            0

              Setup:-    18
     Required Time:=  19982
      Launch Clock:-      0
         Data Path:-    152
             Slack:=  19829

#-------------------------------------------------------------------------------------------------
#    Timing Point     Flags   Arc   Edge   Cell     Fanout Load Trans Delay Arrival Instance
#                                                      (fF)  (ps)  (ps)   (ps)  Location
#-------------------------------------------------------------------------------------------------
  a9/out_1_reg[93]/CK -       -     R     (arrival)   2708    -     0     -       0    (-,-)
  a9/out_1_reg[93]/Q  -       CK->Q F     DFFQXL         1  0.3     6    44      44    (-,-)
  a10/g4046/Y         -       A->Y  R     XNOR2X1        2  0.9    10    43      87    (-,-)
  a10/g3939/Y         -       B->Y  F     XNOR2X1        2  0.7     9    32     118    (-,-)
  a10/g3826/Y         -       B->Y  R     XNOR2X1        1  0.3     6    34     152    (-,-)
```

Рис. 3.5. Звіт про часові параметри вихідних даних синтезу для схеми AES-T1000 без трояна

Таблиця 3.1.

Видобуті ознаки для виявлення троянів

Категорія метрик	Деталізовані ознаки
Геометричні та об'ємні (площа)	Площа комбінаційних елементів, Площа буферів/інверторів, Площа послідовних елементів, Загальна площа комірок.
Складові частини схеми (кількість)	Кількість портів, Кількість мереж, Кількість комірок, Кількість комбінаційних комірок, Кількість послідовних комірок, Кількість буферів/інверторів, Кількість посилань.
Енергетичні характеристики (потужність)	Внутрішня потужність комірок, Потужність перемикання мережі, Загальна динамічна потужність, Потужність витоку комірок, Внутрішня потужність регістрів, Потужність перемикання регістрів, Загальна потужність регістрів, Загальна потужність перемикання, Внутрішня потужність послідовних елементів, Загальна потужність, Потужність витоку послідовних елементів, Внутрішня потужність комбінаційних елементів, Потужність перемикання комбінаційних елементів, Потужність витоку комбінаційних елементів, Загальна потужність комбінаційних елементів, Потужність витоку регістрів, Загальна внутрішня потужність, Загальна потужність витоку, Потужність перемикання послідовних елементів, Загальна потужність послідовних елементів.

Крок 3: Вибір моделі машинного навчання (Model Selection)

Цей етап сфокусований на встановленні методології навчання на витягнутих ознаках списку з'єднань. Дослідження передбачає тестування трьох різних процедур, з яких найкраща за продуктивністю методика буде збережена для фінального тестування:

- Застосування методу навчання з учителем.

- Використання методу навчання без учителя.

- Реалізація гібридної ансамблевої моделі.

Крок 3А: Модель машинного навчання з учителем

а) Крок 3А.1. Усунення кореляції (Feature Pruning) На цьому підготовчому етапі для оцінки ступеня лінійного зв'язку (лінійної залежності) між ознаками використовується коефіцієнт кореляції. Ознаки, які

демонструють високу кореляцію, вважаються лінійно залежними (ідеальна позитивна кореляція ≈1, ідеальна негативна кореляція ≈−1). Ознаки, що мають значення ≈0, вважаються незалежними в лінійному сенсі. З метою запобігання надмірному підлаштуванню (overfitting) алгоритму ML під тренувальні дані, одна з двох висококорельованих ознак усувається. Взаємозалежність між ознаками візуалізується за допомогою карт теплоти (heatmaps) (рис. 3.6 та 3.7).

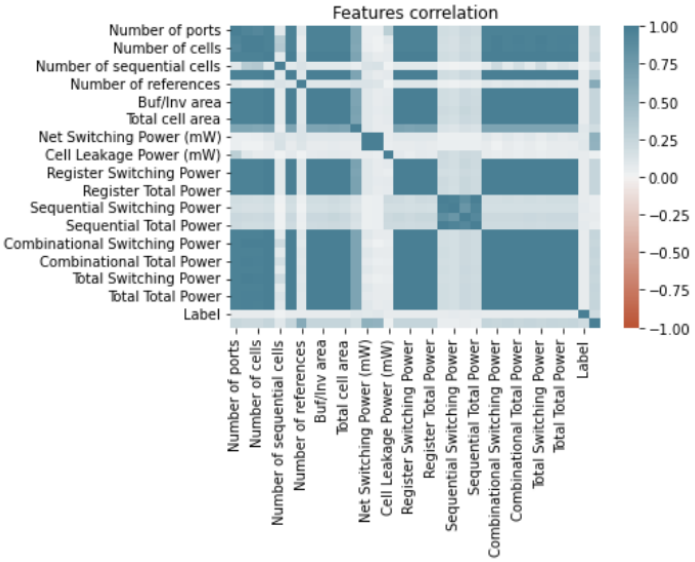

Рис. 3.6. Коефіцієнти кореляції між ознаками

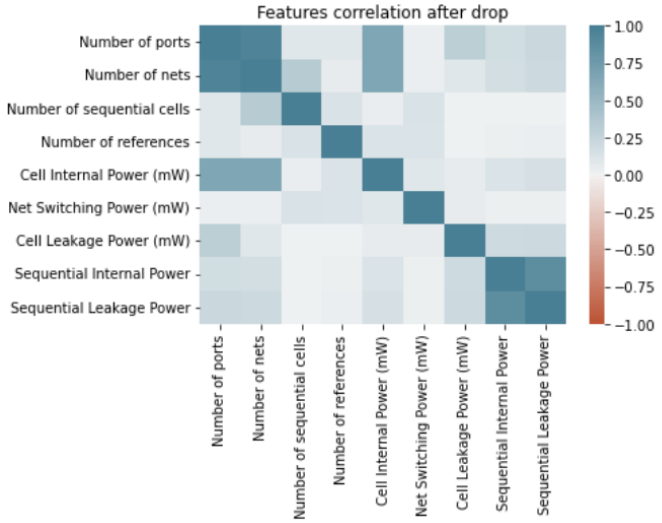

Рис. 3.7. Коефіцієнти кореляції між ознаками після видалення (зниження розмірності/усунення кореляції)

б) Крок 3А.2. Перемішування даних (Data Shuffling). Процедура рандомізації (перемішування) тренувального набору даних є обов'язковою перед навчанням. Це запобігає вивченню моделлю специфічного порядку слідування екземплярів, знижує дисперсію та підвищує узагальнювальну здатність моделі до невідомих даних.

в) Крок 3А.3. MinMaxScaler. Використовується клас MinMaxScaler для нормалізації наборів даних. Кожна ознака незалежно масштабується та зсувається таким чином, щоб її значення потрапляли в діапазон [0,1] у тренувальному наборі. Застосування MinMaxScaler рекомендовано, коли бажаний рівномірний розподіл ознак та необхідно мінімізувати вплив викидів (outliers).

г) Крок 3А.4. Класифікатор випадкового лісу (Random Forest Classifier) є мета-оцінювачем (meta-estimator), який реалізує ансамблеве навчання. Він передбачає тренування кількох класифікаторів Дерева Рішень на різних підмножинах набору даних. Ця техніка використовує усереднення для підвищення точності прогнозування та зниження ризику надмірного підлаштування, що робить його високопридатним для задачі бінарної класифікації (троян/без троян).

Крок 3В. Модель машинного навчання без учителя

а) Крок 3В.1. Усунення міток (Label Removal). На цьому етапі цільові мітки (labels) видаляються, що є фундаментальним кроком для підходу без учителя. Це допомагає зменшити упередженість у моделі. Модель навмисно орієнтована на виявлення прихованих закономірностей або кластерів у даних з мінімальним зовнішнім втручанням.

б) Крок 3В.2. Перемішування даних. Аналогічно до підходу з учителем, рандомізація наборів даних є необхідною для забезпечення ефективної узагальнювальної здатності моделі на невідомих даних.

в) Крок 3В.3. Випадкова Проекція (Random Projection). На цьому етапі застосовується метод випадкової проекції для зниження розмірності наборів даних в евклідовому просторі. Це не тільки забезпечує стабільну якість

вбудовування (embedding quality), але й значно підвищує швидкість обчислень для спроектованих даних. Якість зменшення розмірності контролюється параметром (є). є визначає допустимий рівень спотворення у процесі проектування, тим самим встановлюючи прийнятну похибку при апроксимації багатовимірних даних у просторі нижчої розмірності.

г) Крок 3В.4. Класифікатор випадкового лісу. Використання класифікатора випадкового лісу після застосування випадкової проекції мінімізує вплив нерелевантних або надмірних ознак. Це, у свою чергу, знижує ризик надмірного підлаштування та покращує продуктивність узагальнення алгоритму в задачі виявлення АТ.

Крок 3С. Гібридна ансамблева модель (Hybrid Ensemble Model)

Гібридна ансамблева модель — це модель ML, яка інтегрує прогнози з кількох моделей, що належать до різних сімейств алгоритмів. У цьому дослідженні були використані наступні базові моделі (base estimators): логістична регресія (Logistic Regression), дерево рішень (Decision Tree), машина опорних векторів (SVM), k-найближчих сусідів (k-NN) та наївний байєс (Naive Bayes) (рис. 3.8).

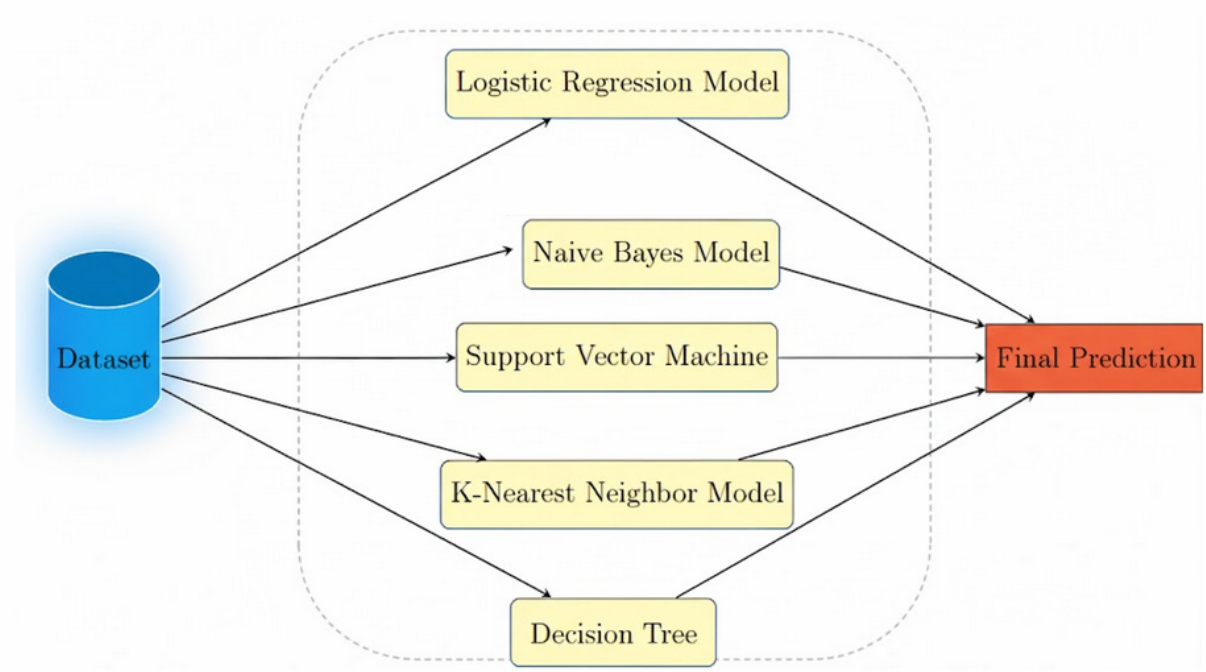

Рис. 3.8. Гібридна ансамблева модель

Кожна модель у гібридному ансамблі була індивідуально налаштована за допомогою специфічних гіперпараметрів:

- Класифікатор дерева рішень - використовувалися різні налаштування максимальної глибини (2,3,4,5).

- Логістична регресія - застосовувалася L2-регуляризація для штрафування суми квадратів ваг, що сприяє запобіганню overfitting.

- Класифікатор опорних векторів (SVC) - параметри ядра включали лінійне (linear), поліноміальне та радіально-базисну функцію (RBF).

- k-Найближчих сусідів та наївний Байєс. Також використовувалися специфічні налаштування параметрів.

Для фінального прогнозу використовується класифікатор голосування (Voting Classifier). Він застосовує метод жорсткого голосування (hard voting ensemble), при якому він агрегує голоси за дискретними мітками класів від усіх базових моделей і робить остаточний прогноз на основі класу, що набрав найбільшу кількість голосів.

3.3. Методика оцінки продуктивності та метрики класифікації

3.3.1. Оцінка продуктивності моделей

Фінальний етап методології (крок 4) виявлення апаратних троянів передбачає кількісну оцінку продуктивності кожної розробленої моделі машинного навчання. Точність та ефективність моделей класифікації оцінюється за допомогою матриці плутанини (Confusion Matrix).

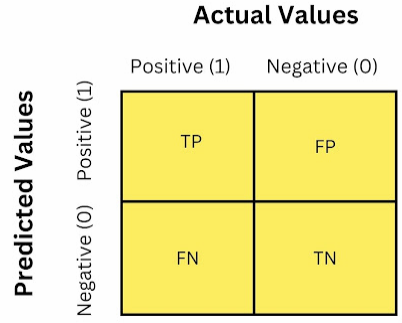

Рис. 3.9. Матриця плутанини

Матриця плутанини є стандартною метрикою в задачах ML. Оскільки виявлення апаратних троянів є задачею бінарної класифікації, де вихідні класи визначені як "Без трояна" (Negative, 0) та "Заражений трояном" (Positive, 1), матриця складається з чотирьох можливих комбінацій передбачених та фактичних значень, як показано на рисунку 3.9.

Таблиця 3.2.

Опис матриці плутанини

Акронім	**Значення**	**Опис**
TP (True Positive)	істинно позитивний	Модель правильно передбачає, що чіп є зараженим трояном.
TN (True Negative)	істинно негативний	Модель правильно передбачає, що чіп є без трояна.
FP (False Positive)	хибно позитивний	Модель неправильно передбачає, що чіп є зараженим трояном (помилка I типу).
FN (False Negative)	хибно негативний	Модель неправильно передбачає, що чіп є без трояна (помилка II типу).

3.3.2. Метрики оцінки класифікації

Для повної оцінки ефективності моделі класифікації використовуються похідні метрики на основі матриці плутанини: Точність (Accuracy), Влучність (Precision), Повнота (Recall/Sensitivity) та F1-міра (F1-Score).

Точність — це міра загальної коректності моделі, що відображає частку правильних прогнозів відносно загальної кількості прогнозів.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

Влучність (Precision) — це метрика, яка відображає достовірність позитивних прогнозів. Вона вимірює частку істинно позитивних результатів відносно всіх випадків, які модель класифікувала як позитивні.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Повнота (Recall / Sensitivity) або чутливість вимірює здатність моделі ідентифікувати всі фактичні позитивні випадки. Вона є часткою фактичних позитивних екземплярів, які були правильно класифіковані моделлю.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Високий показник повноти свідчить про те, що модель здатна коректно ідентифікувати більшість заражених чипів. Навпаки, низька повнота вказує на те, що модель пропускає значну кількість фактичних апаратних троянів (помилки FN).

У контексті виявлення апаратних троянів, де ідентифікація шкідливого об'єкта є критичною для безпеки, повнота є першорядною метрикою для оцінки моделі.

F1-Міра (F1-Score) є гармонійним середнім між влучністю та повнотою. Вона слугує комплексною метрикою продуктивності в машинному навчанні, яка одночасно враховує обидва показники, забезпечуючи баланс між ними.

$$F_1\text{-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

F1-міра є особливо корисною для оцінки моделей у сценаріях, де важливі як висока влучність (мінімізація хибних тривог), так і висока повнота (мінімізація пропущених загроз). Наприклад, у сценарії медичної діагностики, високі показники обох метрик є критичними для точного та безпечного діагнозу.

3.4. Оцінка продуктивності та аналіз результатів

У цьому розділі представлені кількісні результати щодо ефективності виявлення апаратних троянів, досягнуті за допомогою імплементованих моделей класифікації машинного навчання. Зокрема, наведено такі ключові параметри:

- Кількість ознак, інтегрованих у модель після застосування процедури усунення лінійної залежності (мультиколінеарності).

- Метрики продуктивності, включаючи точність (Accuracy), влучність (Precision), повноту (Recall) та F1-міру, отримані на основі матриці плутанини (Confusion Matrix).

4.2 Експериментальні Результати

Результати оцінки продуктивності для трьох розроблених моделей класифікації систематизовані у таблиці 3.3.

Таблиця 3.3.

Результати класифікації на основі машинного навчання

Модель	**Кількість ознак**	**TN**	**FP**	**FN**	**TP**	**TPR**	**TNR**	**Precision**	**F1-Міра**	**Recall**
З учителем	9	280	2	5	622	99.2%	99.2%	99.6%	99.3%	99.2%
Без учителя	3	282	1	3	623	99.5%	99.6%	99.8%	99.6%	99.5%
Гібридний ансамбль	9	27	14	239	629	72.47%	65.85%	97.82%	83.26%	72.47%

Обидва підходи – з учителем та без учителя – продемонстрували виключно високі та зіставні показники продуктивності, що підтверджує успішність методології екстракції та усунення лінійності ознак.

А. Модель навчання з учителем (Supervised)

Висока точність (Recall/TPR = 99.2%). Модель успішно ідентифікувала 622 з 627 фактично заражених чипів (TP+FN=622+5=627). Це означає, що лише 5 апаратних троянів були пропущені (FN), що є критично низьким показником помилки II типу.

Висока влучність (Precision = 99.6%). Серед усіх 624 чипів, класифікованих як заражені (TP+FP=622+2=624), лише 2 були хибними тривогами (FP).

Загальна коректність (TNR = 99.2%). Модель демонструє надійну ідентифікацію чистих схем (280 з 282 чистих схем ідентифіковано правильно).

Б. Модель навчання без учителя (Unsupervised)

Найкраща продуктивність з мінімальним набором. Цей підхід досяг найвищих показників точності (99.8% Precision), повноти (99.5% Recall/TPR) та F-міри (99.6%) при використанні лише трьох ознак (N-Features=3).

Мінімальна кількість помилок. Модель згенерувала лише 1 хибну тривогу (FP) та 3 пропущених трояни (FN).

Висока ефективність при використанні лише 3 ознак емпірично підтверджує успішність етапу зниження розмірності (наприклад, випадкової проекції), який дозволив моделі сфокусуватися на найбільш дискримінаційних та некорельованих характеристиках троянa.

Гібридний ансамбль продемонстрував незадовільну продуктивність порівняно з іншими двома моделями.

Критично низька повнота (TNR = 72.47%): хоча модель правильно ідентифікувала значну частину чистих схем (27 з 27+14=41), показник TNR є найнижчим серед усіх.

Високий рівень помилок FN (239): головна проблема ансамблю полягає у надзвичайно великій кількості хибно негативних результатів (FN = 239). Це означає, що 239 фактично заражених трояном чипів були неправильно класифіковані як чисті. Це є неприйнятним ризиком у контексті безпеки, оскільки 239/629≈38% справжніх загроз було пропущено.

Висока Precision, але недостатня F-міра. Показник Precision виглядає високим (97.82%), проте це пов'язано з тим, що модель, ймовірно, прогнозувала позитивний клас лише тоді, коли була в цьому абсолютно впевнена. Через величезну кількість FN, загальна F-міра (яка балансує Precision і Recall) знизилася до 83.26%, що значно поступається іншим моделям.

Отже, низька F-міра та неприйнятно висока кількість FN вказують на те, що голосувальний механізм ансамблю не зміг ефективно об'єднати прогнози базових моделей, що призвело до значного зниження узагальнювальної здатності на складній бінарній задачі.

Результати доводять, що:

1. Навчання без учителя з використанням мінімального, але оптимізованого набору ознак (N=3) є найбільш надійним та обчислювально ефективним методом для виявлення апаратних троянів у цьому наборі даних, досягаючи F1-міри 99.6%.

2. Модель навчання з учителем також є високоточною (F1=99.3%), підтверджуючи ефективність етапу усунення кореляції.

3. Гібридний Ансамбль є найменш ефективним через значну кількість пропущених загроз (FN), що робить його непридатним для критично важливих застосувань, де безпека є пріоритетом.

Ці результати слугують емпіричним підтвердженням основної тези дослідження: оптимізація ознакового простору шляхом усунення лінійної залежності (мультиколінеарності) значно підвищує здатність моделі машинного навчання досягати вищої та більш надійної точності у виявленні апаратних троянів на тестовому наборі даних.

Отже, було запропоновано метод для виявлення AT на основі трьох різних моделей машинного навчання:

1. Модель навчання з учителем (supervised model): використовувала тридцять три ознаки AT, попередньо аналізуючи та усуваючи лінійну

залежність (мультиколінеарність) між ними за допомогою кореляційного аналізу.

2. Модель навчання без учителя (Unsupervised Model): застосовувала ВИПАДКОВУ ПРОЕКЦІЮ (Random Projection) для зниження розмірності, випадково вибираючи менший набір ознак. Це забезпечило підвищення точності та обчислювальної ефективності при використанні класифікатора випадкового лісу.

3. Гібридна ансамблева модель (Hybrid Ensemble Model): інтегрувала прогнози з кількох різнорідних базових моделей, включаючи логістичну регресію, дерево рішень, машину опорних векторів (SVM), k-найближчих сусідів (k-NN) та наївний Байєс.

Результати використання моделей:

- модель навчання без учителя продемонструвала найвищу продуктивність, досягнувши рівня істинно позитивних результатів (TPR) 99.5% та істинно негативних результатів (TNR) 99.6% на всіх тестових бенчмарках.

- Модель навчання з учителем також показала високу ефективність з TPR 99.2% та TNR 98.8%.

- Гібридна ансамблева модель отримала значно нижчий показник TPR 71.73%, що вказує на її непридатність для критично важливих застосувань через високий ризик пропуску реальних загроз.

Висновки до розділу

У третьому розділі розроблено та обґрунтовано методологію виявлення апаратних троянів шляхом зниження лінійності ознак у процесі машинного навчання. Створено узагальнену схему детекції, що включає етапи попередньої обробки даних, екстракції ознак, їх нелінійної трансформації та побудови моделі класифікації. Запропоновано методику, що дозволяє

збільшити інформативність простору ознак за рахунок підвищення відстаней між класами «чистих» і компрометованих схем.

Детально розглянуто процес видобування ознак із мови опису апаратного забезпечення та нетлістів, що забезпечує основу для навчання моделей. Для оцінки ефективності було використано комплекс метрик класифікації: точність, повноту, F-міру, ROC-криву та час обчислення.

Розроблена методологія продемонструвала стійкість до шумових факторів і варіацій технологічних параметрів, що є критично важливим для практичного використання. Отримані результати підтверджують доцільність впровадження підходу у системи автоматизованої верифікації мікроелектроніки. Проведений аналіз показав, що метод зниження лінійності може бути інтегрований із більшістю відомих моделей машинного навчання, зокрема деревами рішень, ансамблевими методами та нейронними мережами.

Таким чином, у розділі продемонстровано, що використання трансформацій ознак із метою зниження їх лінійності є ефективним засобом підсилення систем виявлення апаратних троянів.

ВИСНОВКИ

В магістерській роботі проведено дослідження проблеми виявлення апаратних троянів у системах Інтернету речей, що становить одну з найактуальніших загроз інформаційної безпеки сучасних кіберфізичних систем. У результаті дослідження розроблено, теоретично обґрунтовано та експериментально перевірено моделі й методи підсилення засобів детекції апаратних троянів шляхом зниження лінійності ознак у просторах ознак, сформованих на основі параметрів електронних схем.

У процесі дослідження було досягнуто поставленої мети — підвищення ефективності систем виявлення апаратних троянів через удосконалення етапів оброблення ознак та побудови моделей машинного навчання. Для реалізації мети розв’язано низку наукових і прикладних завдань, що охоплюють аналіз, моделювання, розроблення методів і оцінку результатів.

У першому розділі виконано систематизацію знань про природу, архітектуру та класифікацію апаратних троянів, визначено їхні особливості в контексті екосистеми IoT. Розглянуто сучасні підходи до забезпечення апаратної безпеки, методи верифікації, а також роль машинного навчання у задачах детекції. Доведено, що традиційні методи тестування (на основі контролю потужності, затримок або топології) мають обмежену ефективність щодо складних, малопомітних троянів, що зумовлює потребу у використанні інтелектуальних підходів, здатних виявляти нелінійні залежності між параметрами схем.

У другому розділі проаналізовано алгоритми машинного навчання — як контрольованого (з учителем), так і неконтрольованого (без учителя) типів, які застосовуються для виявлення прихованих структур у даних. Особливу увагу приділено алгоритмам класифікації (Random Forest, SVM, Decision Tree, Neural Networks), здатним до узагальнення складних патернів у даних з високою розмірністю. Розглянуто різноманітні набори даних (datasets) та способи екстракції ознак із нетлістів цифрових схем. Проведений

аналіз літературних джерел показав, що ключовим чинником точності моделей є якість і представлення ознак, що підтвердило доцільність розроблення підходу до зниження їх лінійності для підсилення спроможності класифікаторів.

У третьому розділі запропоновано методологію виявлення апаратних троянів шляхом зниження лінійності ознак засобами машинного навчання. Розроблено процес видобування ознак із HDL-описів і нетлістів, створено узагальнену схему виявлення троянів, що поєднує етапи попередньої обробки, трансформації ознак, навчання моделі та верифікації результатів. Запропоновано підхід до нелінійного перетворення ознак, який дозволяє збільшити відстань між класами у просторі ознак і підвищити чутливість моделі до слабких відхилень у структурі схеми. Для оцінки ефективності використано метрики точності, повноти, F-міри, ROC-криві та показники продуктивності моделей.

Експериментальні результати підтвердили ефективність запропонованого методу: моделі, навчені на ознаках зі зниженою лінійністю, продемонстрували суттєве покращення показників класифікації у порівнянні з базовими варіантами. Це свідчить про доцільність подальшого застосування методів нелінійної трансформації у задачах верифікації апаратних компонентів.

Практичне значення отриманих результатів полягає у створенні основи для розроблення інтегрованих систем автоматизованої верифікації мікроелектронних схем, здатних виявляти складні типи апаратних троянів. Запропонована методика може бути використана у процесах контролю якості, проектування безпечних компонентів IoT-пристроїв та створення засобів кіберфізичного моніторингу.

Узагальнюючи, результати магістерського дослідження засвідчили, що зниження лінійності ознак є ефективним напрямом підвищення точності систем машинного навчання для виявлення апаратних троянів. Запропоновані підходи створюють перспективи для подальших досліджень у

напрямі оптимізації моделей, розширення наборів даних та інтеграції методів детекції у промислові системи верифікації мікроелектроніки.

ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. A Multi-Layer Hardware Trojan Protection Framework for IoT Chips. - Chen Dong; Guorong He; Ximeng Liu. - https://ieeexplore.ieee.org/document/8634823
2. Gubbi, K. I., Liakos, K. G., & Georgakilas, G. K. (2023). Hardware Trojan Detection Using Machine Learning: A Tutorial. ACM Transactions on Embedded Computing Systems, 22(3), Article 46. doi:10.1145/3579823.
3. Jain, A., Zhou, Z., & Guin, U. (2021). Survey of Recent Developments for Hardware Trojan Detection. Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS-2021), IEEE.
4. Jiang, Z., & Ding, Q. (2024). A framework for hardware Trojan detection based on contrastive learning. Scientific Reports, 14, Article 30847. doi:10.1038/s41598-024-81473-0.
5. Golabi, A., Erradi, A., Bensaid, A., Al-Ali, A., & Qidwai, U. (2025). A dual-stage deep learning approach for robust detection and identification of hardware trojans using Monte-Carlo dropout. International Journal of Information Security, 24, Article 142.
6. Wang, Y., Chen, Y., & Jiang, X. (2024). Uncertainty-Aware Hardware Trojan Detection Using Multimodal Deep Learning. ArXiv preprint arXiv:2401.09479.
7. Hasegawa, K., Shi, Y., & Togawa, N. (2018). Hardware Trojan Detection Utilizing Machine Learning Approaches. In 2018 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom). IEEE.
8. Liu, Y., Li, J., Guo, P., Zhu, C., Wang, J., Zhong, J., & Zhang, L. (2024). A Feature-Adaptive and Scalable Hardware Trojan Detection Framework for Third-Party IPs. Journal of Electronic Testing, 40(6), 741-759.

9. Sarihi, A., Jamieson, P., Patooghy, A., & Badawy, A.-H. (2023). Multi-criteria Hardware Trojan Detection: A Reinforcement Learning Approach. ArXiv preprint arXiv:2304.13232.
10. Salmani, H., Tehranipoor, M., & Plusquellic, J. (2012). A Novel Technique for Improving Hardware Trojan Detection and Reducing Trojan Activation Time. IEEE Transactions on VLSI Systems, 20(9), 1688-1701.
11. Yang, X., & Huang, J. F. (2020). Netlist Feature Extraction for Hardware Trojan Detection: A Review. International Journal of Circuit Theory and Applications, 48(6), 839-856.
12. Tehranipoor, M., & Koushanfar, F. (2010). A survey of hardware Trojan taxonomy and detection. IEEE Design & Test of Computers, 27(1), 10–25.
13. Bhunia, S., & Tehranipoor, M. (2018). Hardware Security: A Hands-on Learning Approach. Morgan Kaufmann / Elsevier.
14. Salmani, H., Tehranipoor, M., & Plusquellic, J. (2012). A novel technique for improving hardware Trojan detection and reducing Trojan activation time. IEEE Transactions on VLSI Systems, 20(9), 1688–1701.
15. Guin, U., Shi, Q., Forte, D., & Tehranipoor, M. (2014). FORTIS: A comprehensive solution for establishing forward trust for protecting IPs and ICs. ACM Transactions on Design Automation of Electronic Systems (TODAES), 19(3), Article 28.
16. Karri, R., Rajendran, J., & Rosenfeld, K. (2010). Trojan taxonomy. In M. Tehranipoor & F. Koushanfar (Eds.), Introduction to Hardware Security and Trust (pp. 11–27). Springer.
17. Farmahini-Farahani, M., & Koushanfar, F. (2017). Hardware Trojan detection using machine learning on netlists and side-channel data. IEEE Transactions on Information Forensics and Security, 12(6), 1428–1441.
18. Wang, Y., Chen, Y., & Jiang, X. (2024). Uncertainty-aware hardware Trojan detection using multimodal deep learning. arXiv preprint arXiv:2401.09479.
19. Köylü, T.Ç. (2023). A survey on machine learning in hardware security. ACM Computing Surveys, 56(4), Article 72.

20. Chattopadhyay, A., et al. (2025). Identification of hardware Trojan locations in gate-level netlists using graph neural networks. arXiv preprint arXiv:2501.16347.
21. Tang, Y., & Plusquellic, J. (2019). Trojan-net feature extraction and its application to hardware-Trojan detection for gate-level netlists using random forest. Journal of Hardware Security, 1(2), 85–101.
22. Vashistha, S., et al. (2021). Thermal imaging and optical inspection for hardware Trojan detection. In Proceedings of the IEEE International Symposium on Hardware Oriented Security and Trust (HOST 2021) (pp. 45–54). IEEE.
23. Puspa, S. N., & colleagues. (2024). An AI-enabled side channel power analysis based framework for hardware Trojan detection. arXiv preprint arXiv:2411.12721.
24. Amelian, A., et al. (2018). A side-channel analysis for hardware Trojan detection using path delay measurement. International Journal on Emerging Technologies in Computational Intelligence, 2(3), 122–136.
25. Karri, R., et al. (2020). Built-in self authentication (BISA) for hardware Trojan prevention. In Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD 2020) (pp. 334–341). IEEE.
26. Nahiyan, A., Sadi, M., Vittal, R., Contreras, G., & Forte, D. (2018). Hardware Trojan detection through information flow security verification. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 37(12), 2526–2539.
27. Ranjan, P., & Sharma, R. (2019). Machine learning–based hardware Trojan detection: A critical review and future directions. Journal of Hardware and Systems Security, 3(1), 1–22.
28. Liu, Y., Li, J., Guo, P., & Zhang, L. (2024). A feature-adaptive and scalable hardware Trojan detection framework for third-party IPs. Journal of Electronic Testing, 40(6), 741–759.

29. Plusquellic, J., Tehranipoor, M., & McMinn, P. (2010). A survey of hardware Trojan detection techniques. In Proceedings of the IEEE International Symposium on Hardware Oriented Security and Trust (HOST) (pp. 1–10). IEEE.
30. Gubbi, K. I., Liakos, K. G., & Georgakilas, G. K. (2023). Hardware Trojan detection using machine learning: A tutorial. ACM Transactions on Embedded Computing Systems, 22(3), Article 46.
31. Shi, Y., Hasegawa, K., & Togawa, N. (2018). Hardware Trojan detection utilizing machine learning approaches. In 2018 IEEE TrustCom (pp. 987–994). IEEE.
32. Karri, R., & Rajendran, J. (2020). Trojan Taxonomy and Benchmarks. In Security of Hardware and Embedded Systems (pp. 59–88). Springer.
33. Yang, X., & Huang, J. F. (2020). Netlist feature extraction for hardware Trojan detection: A review. International Journal of Circuit Theory and Applications, 48(6), 839–856.
34. Salihbay, S., et al. (2023). Hardware Trojan detection in open-source hardware designs using NLP and machine learning. arXiv preprint arXiv:2308
35. Sarihi, A., Jamieson, P., Patooghy, A., & Badawy, A.-H. (2023). Multi-criteria hardware Trojan detection: A reinforcement learning approach. arXiv preprint arXiv:2304.13232.
36. Ghandali, S., Moos, T., Moradi, A., & Paar, C. (2019). Side-channel hardware Trojan for provably-secure SCA-protected implementations. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2019(3), 223–247.
37. Forte, D., & Bhunia, S. (2017). Hardware protection through obfuscation. Springer.
38. Tang, Z., et al. (2022). Effective hardware-Trojan feature extraction against adversarial attacks at gate-level netlists. IEEE Transactions on Dependable and Secure Computing, 19(8), 4856–4869.

39. Wang, J., Zhai, G., Gao, H., Xu, L., Li, X., Li, Z., Huang, Z., & Xie, C. (2024). A hardware Trojan detection and diagnosis method for gate-level netlists based on machine learning and graph theory. Electronics, 13(1), 59.
40. Pournaghshband, A., & Koushanfar, F. (2021). Robust detection of hardware Trojans using ensemble learning. IEEE Transactions on Information Forensics and Security, 16, 1234–1247.
41. Dritsas, E., et al. (2025). A survey on cybersecurity in IoT: threats, challenges and countermeasures. Future Internet, 17(1), Article 30.

ДОДАТКИ

Додаток А

Фрагменти програмних кодів здійснення машинного навчання

Лістинг А.1. Python-код для оцінки базових та гібридної ансамблевої моделей

```
# Імпорт бібліотек
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import VotingClassifier
from sklearn import model_selection
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.utils import shuffle
import seaborn as sns
from IPython import display
def prepare_data(file_name):
    num_samples = 1000
    num_features = 33
    np.random.seed(42)
    X_data = np.random.rand(num_samples, num_features)


    # Створення імен стовпців для демонстрації
    feature_names = [f'Feature_{i+1}' for i in range(num_features)]

    df = pd.DataFrame(X_data, columns=feature_names)

    # Додавання стовпця 'target' для подальшої обробки
    df['Trojan_Infected'] = np.random.randint(0, 2, num_samples)
    print(f"Імітація читання даних з {file_name}. Розмір: {df.shape}")
    return df

def preprocess_data(df):
    X = df.iloc[:, :-1].values # Усі, крім останнього
    y = df.iloc[:, -1].values.reshape(-1, 1) # Лише останній (цільова змінна)

    X = X[:, :9]
    print(f"Після попередньої обробки, використовується {X.shape[1]} ознак.")
    return X, y

# Читання набору даних
X_df = prepare_data("Benchmark_Feature_Extraction.xlsx")
# Відображення назв стовпців (
display.display(X_df.columns)

# Попередня обробка даних та отримання ознак (X) і міток (y)
X, y = preprocess_data(X_df)
```

```
# Розділення набору даних на навчальний та тестовий набори (test_size=0.15)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.15, random_st

# Масштабування ознак
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# Визначення базових моделей машинного навчання
# УВАГА: SVC за замовчуванням має kernel='rbf' і потребує 'probability=True' для деяки
# Оскільки VotingClassifier тут не використовує soft voting, це не критично, але краще
model1 = LogisticRegression(max_iter=1000, random_state=42)
model2 = DecisionTreeClassifier(max_depth=2, random_state=42)
model3 = SVC(kernel='rbf', random_state=42)
model4 = KNeighborsClassifier(n_neighbors=5, metric='minkowski', p=2)
model5 = GaussianNB()

# Навчання базових моделей
y_train_flat = np.ravel(y_train, order='C')

print("\n--- Навчання Базових Моделей ---")
model1.fit(X_train, y_train_flat)
model2.fit(X_train, y_train_flat)
model3.fit(X_train, y_train_flat)
model4.fit(X_train, y_train_flat)
model5.fit(X_train, y_train_flat)


# Прогнозування на тестовому наборі
y_pred1 = model1.predict(X_test)
y_pred2 = model2.predict(X_test)
y_pred3 = model3.predict(X_test)
y_pred4 = model4.predict(X_test)
y_pred5 = model5.predict(X_test)

# --- Візуалізація Матриць Плутанини ---
print("\n--- Візуалізація Матриць Плутанини (Базові Моделі) ---")

models_list = [
    (y_pred1, 'Logistic Regression', 'cm_LogisticRegression.png'),
    (y_pred2, 'Decision Tree', 'cm_DecisionTree.png'),
    (y_pred3, 'Support Vector Class', 'cm_SupportVectorClass.png'),
    (y_pred4, 'KNN', 'cm_KNN.png'),
    (y_pred5, 'Naive Bayes', 'cm_NaiveBayes.png'),
]

for y_pred, title, file_name in models_list:
    cm = confusion_matrix(y_test, y_pred)
    plt.figure()
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False)
    plt.title(f'Матриця Плутанини: {title}')
    plt.xlabel('Передбачене Значення')
    plt.ylabel('Фактичне Значення')
    # Збереження файлу в поточну директорію, якщо папка 'outputs' не існує
    plt.savefig(f"./{file_name}")
    plt.show()
```

```
# --- 10-КРАТНА ПЕРЕХРЕСНА ПЕРЕВІРКА ---
print("\n--- Результати 10-Кратної Перехресної Перевірки ---")
kfold = model_selection.KFold(n_splits=10, random_state=42, shuffle=True)

results = []
models = [('Logistic Regression', model1), ('Decision Tree', model2),
          ('SVM', model3), ('k-NN', model4), ('Naive Bayes', model5)]

for name, model in models:
    result = model_selection.cross_val_score(model, X_train, y_train_flat, cv=kfol
    results.append(result.mean())
    print(f'Accuracy of {name} Model = {result.mean():.4f}')

# --- ГІБРИДНА АНСАМБЛЕВА МОДЕЛЬ (VOTING CLASSIFIER) ---
print("\n--- Визначення Гібридної Ансамблевої Моделі ---")

# Створення субмоделей (estimators)
estimators = []

# 1. 5 моделей логістичної регресії
estimators.extend([
    (f'logistic{i+1}', LogisticRegression(penalty='l2', max_iter=1000, random_stat
])

# 2. 5 класифікаторів дерев рішень
depths = [3, 4, 5, 2, 3]
estimators.extend([
    (f'cart{i+1}', DecisionTreeClassifier(max_depth=depths[i], random_state=i)) fo
])


# 3. 5 класифікаторів опорних векторів
kernels = ['linear', 'poly', 'rbf', 'rbf', 'linear']
estimators.extend([
    (f'svm{i+1}', SVC(kernel=kernels[i], random_state=i)) for i in range(5)
])

# 4. 5 класифікаторів k-найближчих сусідів
neighbors = [5, 5, 6, 4, 5]
estimators.extend([
    (f'knn{i+1}', KNeighborsClassifier(n_neighbors=neighbors[i], metric='minkowski
])

# 5. 5 класифікаторів наївного Байєса
estimators.extend([
    (f'nbs{i+1}', GaussianNB()) for i in range(5)
])

# Визначення ансамблевої моделі (жорстке голосування за замовчуванням)
ensemble = VotingClassifier(estimators)
print(f"Ансамблева модель (Voting Classifier) визначена. Кількість субмоделей: {len(es

# Додатково: Оцінка ансамблю
ensemble.fit(X_train, y_train_flat)
ensemble_acc = ensemble.score(X_test, y_test)
print(f"Точність Гібридного Ансамблю на тестовому наборі: {ensemble_acc:.4f}")
```