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

**БОЙКО Назар Романович**

# Програмний модуль виявлення шкідливого програмного забезпечення на основі гібридних моделей глибокого навчання / Software Module for Detecting Malicious Software Based on Hybrid Deep Learning Models

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

Кваліфікаційна робота

Виконав студент групи КН-43
Н.Р. Бойко

_____

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

_____

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

Завідувач кафедри
_____ Н.В. Дзюбановська

**Тернопіль – 2026**

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

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

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
БОЙКУ Назару Романовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи
**Програмний модуль виявлення шкідливого програмного забезпечення на основі гібридних моделей глибокого навчання / Software Module for Detecting Malicious Software Based on Hybrid Deep Learning Models**
керівник роботи д.т.н., професор М. П. Комар
затверджені наказом по університету від 01 грудня 2025 року № 894.
2. Строк подання студентом закінченої кваліфікаційної роботи 25 травня 2026 р.
3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.
4. Основні питання, які потрібно розробити
- проаналізувати сучасні підходи та методи виявлення шкідливого програмного забезпечення;
- дослідити можливості застосування методів глибокого навчання для задач детекції шкідливого програмного забезпечення;
- обґрунтувати вибір гібридної моделі глибокого навчання для підвищення точності виявлення;
- розробити алгоритм функціонування програмного модуля виявлення шкідливого програмного забезпечення;
- реалізувати програмний модуль із використанням сучасних програмних бібліотек і засобів машинного навчання;
- провести тестування розробленого програмного модуля та проаналізувати отримані результати.
5. Перелік графічного матеріалу в роботі
- структурна схема програмного модуля виявлення шкідливого програмного забезпечення;
- узагальнена схема алгоритму роботи програмного модуля.

6. Консультанти розділів кваліфікаційної роботи

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

7. Дата видачі завдання 01 грудня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

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

Студент_____ Н. Р. Бойко
підпис

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

# АНОТАЦІЯ

Кваліфікаційна робота на тему «Програмний модуль виявлення шкідливого програмного забезпечення на основі гібридних моделей глибокого навчання» на здобуття освітнього ступеня «бакалавр» зі спеціальності 122 «Комп'ютерні науки» освітньої програми «Комп'ютерні науки» написана обсягом в 56 сторінок і містить 3 ілюстрацій, 4 таблиць, 2 додатки та 30 використаних джерел.

Метою кваліфікаційної роботи є розроблення програмного модуля виявлення шкідливого програмного забезпечення на основі гібридних моделей глибокого навчання, який забезпечує підвищення ефективності класифікації шкідливих програм за рахунок поєднання ознак, отриманих різними нейронними архітектурами.

Методи досліджень включають методи аналізу та узагальнення науково-технічної літератури, методи машинного та глибокого навчання, алгоритми згорткових нейронних мереж, а також методи програмної реалізації та тестування програмних систем.

Розроблено програмний модуль виявлення шкідливого програмного забезпечення на основі гібридних моделей глибокого навчання, який забезпечує приймання вхідних зразків програмного забезпечення, їх попередню обробку, перетворення байтового вмісту виконуваних файлів у зображення, формування ознакових представлень за допомогою згорткових нейронних мереж, об'єднання отриманих ознак і виконання бінарної класифікації зразків.

Результати дослідження можуть бути використані для створення та вдосконалення програмних засобів виявлення шкідливого програмного забезпечення, розроблення інтелектуальних модулів кібербезпеки, автоматизації аналізу виконуваних файлів, підвищення ефективності систем захисту інформації.

Ключові слова: ШКІДЛИВЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ГЛИБОКЕ НАВЧАННЯ, МАШИННЕ НАВЧАННЯ, ГІБРИДНА МОДЕЛЬ, ЗГОРТКОВА НЕЙРОННА МЕРЕЖА, БІНАРНА КЛАСИФІКАЦІЯ, КІБЕРБЕЗПЕКА.

# ANNOTATION

Qualification work on the topic «Software Module for Detecting Malicious Software Based on Hybrid Deep Learning Models» for Bachelor's degree on speciality 122 «Computer Science» educational and professional program «Computer Science» is written on 56 pages and it contains 3 figures, 4 tables, 2 annexes and 30 sources.

The purpose of the qualification work is to develop a software module for malware detection based on hybrid deep learning models, which improves the efficiency of malware classification by combining features obtained from different neural network architectures.

The research methods include methods of analysis and generalization of scientific and technical literature, machine learning and deep learning methods, convolutional neural network algorithms, as well as methods of software implementation and testing of software systems.

A software module for malware detection based on hybrid deep learning models has been developed. The module provides input software sample acquisition, preliminary data processing, conversion of the byte content of executable files into images, formation of feature representations using convolutional neural networks, fusion of the obtained features, and binary classification of samples.

The research results can be used for the creation and improvement of software tools for malware detection, the development of intelligent cybersecurity modules, the automation of executable file analysis, and the improvement of the efficiency of information security systems.

Keywords: MALWARE, DEEP LEARNING, MACHINE LEARNING, HYBRID MODEL, CONVOLUTIONAL NEURAL NETWORK, BINARY CLASSIFICATION, CYBERSECURITY.

ЗМІСТ

# ВСТУП

У сучасних умовах стрімкого розвитку інформаційних технологій та цифровізації практично всіх сфер суспільного життя питання інформаційної безпеки набувають особливої актуальності. Однією з найбільш поширених загроз для комп'ютерних систем і мереж є шкідливе програмне забезпечення, яке застосовується для несанкціонованого доступу до інформаційних ресурсів, порушення цілісності та конфіденційності даних, а також дестабілізації роботи інформаційних систем. Зростання кількості та різноманітності зразків шкідливого програмного забезпечення, зокрема поліморфних і метаморфних програм, суттєво ускладнює використання традиційних сигнатурних методів виявлення [13, 16].

Аналіз сучасних досліджень свідчить, що класичні підходи до виявлення шкідливого програмного забезпечення не забезпечують достатньої ефективності в умовах постійної модифікації атак та появи нових векторів загроз [5, 22]. У зв'язку з цим значного поширення набули методи машинного та глибокого навчання, які дозволяють автоматично виділяти інформативні ознаки та здійснювати класифікацію зразків програмного коду з високим рівнем точності [6, 12].

Окремим перспективним напрямом є застосування згорткових нейронних мереж для аналізу шкідливого програмного забезпечення шляхом його представлення у вигляді зображень, що дозволяє використовувати напрацювання комп'ютерного зору для задач кібербезпеки [1, 13]. Разом із тим, дослідження показують, що використання окремих архітектур глибокого навчання має обмеження, пов'язані зі здатністю до узагальнення та стійкістю до різних типів атак [11, 15]. У зв'язку з цим доцільним є застосування гібридних моделей, які поєднують переваги кількох нейронних архітектур і забезпечують підвищення точності виявлення шкідливого програмного забезпечення [14, 15].

Особливу увагу в сучасних наукових працях приділено виявленню шкідливих застосунків у мобільних операційних системах, зокрема Android OS, що обумовлено їх широким використанням та підвищеною вразливістю до атак

[2, 9]. Разом із тим, актуальним залишається завдання розроблення універсальних програмних рішень, здатних інтегрувати гібридні моделі глибокого навчання у вигляді окремих програмних модулів для використання в інформаційних системах різного призначення [10, 21].

Метою кваліфікаційної роботи є розроблення програмного модуля виявлення шкідливого програмного забезпечення на основі гібридних моделей глибокого навчання, який забезпечує підвищення ефективності класифікації шкідливих програм за рахунок поєднання ознак, отриманих різними нейронними архітектурами.

Для досягнення поставленої мети в роботі необхідно вирішити такі завдання:

– проаналізувати сучасні підходи та методи виявлення шкідливого програмного забезпечення;

– дослідити можливості застосування методів глибокого навчання для задач детекції шкідливого програмного забезпечення;

– обґрунтувати вибір гібридної моделі глибокого навчання для підвищення точності виявлення;

– розробити алгоритм функціонування програмного модуля виявлення шкідливого програмного забезпечення;

– реалізувати програмний модуль із використанням сучасних програмних бібліотек і засобів машинного навчання;

– провести тестування розробленого програмного модуля та проаналізувати отримані результати.

Об’єктом дослідження є процеси виявлення шкідливого програмного забезпечення в комп’ютерних системах.

Предметом дослідження є методи та алгоритми виявлення шкідливого програмного забезпечення на основі гібридних моделей глибокого навчання.

У процесі виконання кваліфікаційної роботи використовуються методи аналізу та узагальнення науково-технічної літератури, методи машинного та глибокого навчання, алгоритми згорткових нейронних мереж, а також методи програмної реалізації та тестування програмних систем.

Практичне значення отриманих результатів полягає у можливості використання розробленого програмного модуля для автоматизованого виявлення шкідливого програмного забезпечення в інформаційних системах, а також у навчальному процесі при підготовці фахівців у галузі комп'ютерних наук та кібербезпеки.

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

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

# 1 АНАЛІЗ ПРОБЛЕМНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

## 1.1 Опис предметної області виявлення шкідливого програмного забезпечення

У сучасних умовах розвитку інформаційних технологій та широкого впровадження цифрових сервісів питання забезпечення інформаційної безпеки набувають особливої актуальності. Однією з найбільш поширених і небезпечних загроз для комп'ютерних систем є шкідливе програмне забезпечення, яке використовується для несанкціонованого доступу до інформаційних ресурсів, порушення цілісності даних, викрадення конфіденційної інформації та дестабілізації роботи інформаційних систем [16, 22].

Під шкідливим програмним забезпеченням розуміють програмний код, спеціально створений або модифікований з метою виконання небажаних або прихованих дій без відома користувача. Особливістю сучасного шкідливого програмного забезпечення є висока складність архітектури, здатність до маскування, адаптації та обходу традиційних засобів захисту, що істотно ускладнює процес його виявлення та аналізу [13].

Шкідливе програмне забезпечення класифікується за функціональним призначенням, способом поширення та рівнем взаємодії з операційною системою. До основних класів шкідливого програмного забезпечення належать комп'ютерні віруси, мережеві черви, троянські програми, шпигунське програмне забезпечення, програми-вимагачі, бекдори та ботнет-клієнти [22].

Комп'ютерні віруси характеризуються здатністю до саморозмноження шляхом інфікування інших файлів або програм, що призводить до швидкого поширення в межах файлової системи. Мережеві черви, на відміну від вірусів, поширюються автономно через мережеві канали, використовуючи вразливості протоколів або сервісів, і можуть викликати значні навантаження на мережеву інфраструктуру [11].

Троянські програми є одним із найбільш поширених і небезпечних класів шкідливого програмного забезпечення. Вони маскуються під легітимні застосунки та активуються безпосередньо користувачем, після чого виконують

приховані шкідливі дії, зокрема встановлення бекдорів або передавання конфіденційної інформації [13]. Шпигунське програмне забезпечення призначене для збору персональних даних, облікових записів і фінансової інформації, тоді як програми-вимагачі блокують доступ до даних користувача з подальшою вимогою викупу.

Окрему категорію становить шкідливе програмне забезпечення для мобільних операційних систем, зокрема Android OS. Зважаючи на відкриту архітектуру платформи та значну кількість сторонніх застосунків, мобільні пристрої є привабливою ціллю для зловмисників. Дослідження свідчать, що мобільне шкідливе програмне забезпечення активно використовує механізми дозволів і соціальної інженерії для реалізації атак [2, 9].

Традиційні методи виявлення шкідливого програмного забезпечення ґрунтуються переважно на сигнатурному аналізі, який передбачає пошук у програмному коді відомих шаблонів або ознак, характерних для раніше ідентифікованих загроз. Основною перевагою таких підходів є висока швидкодія та точність при виявленні відомих зразків шкідливого програмного забезпечення [16].

Водночас сигнатурні методи мають суттєві обмеження. Вони є неефективними при виявленні нових або модифікованих зразків шкідливого програмного забезпечення, зокрема поліморфних і метаморфних програм, які здатні змінювати власну структуру, зберігаючи при цьому шкідливу функціональність [11]. Крім того, сигнатурні бази потребують постійного оновлення, що не завжди можливо в умовах стрімкого зростання кількості нових загроз.

Евристичні та поведінкові методи частково усувають зазначені недоліки шляхом аналізу дій програм під час виконання. Такі підходи дозволяють виявляти аномальну поведінку та потенційно небезпечні дії, характерні для шкідливого програмного забезпечення. Проте поведінковий аналіз супроводжується значними обчислювальними витратами, потребує спеціалізованих середовищ виконання та часто характеризується підвищеним рівнем хибних спрацювань [22].

У зв’язку з обмеженнями традиційних методів дедалі більшого поширення набувають підходи, засновані на машинному та глибокому навчанні. Такі методи дозволяють автоматично аналізувати великі обсяги даних, виділяти інформативні ознаки та здійснювати класифікацію програмного забезпечення з високим рівнем точності [6, 12].

Глибоке навчання, зокрема згорткові нейронні мережі, продемонструвало високу ефективність у задачах аналізу складних структурованих даних. Одним із перспективних напрямів є підхід malware-as-image, який передбачає перетворення байтового подання виконуваних файлів у двовимірні зображення. Це дозволяє використовувати перевірені архітектури комп’ютерного зору для задач виявлення шкідливого програмного забезпечення [1, 13].

Застосування глибокого навчання зменшує залежність від ручного проєктування ознак та підвищує стійкість моделей до методів обфускації коду. Разом із тим, результати сучасних досліджень вказують на доцільність подальшого вдосконалення таких підходів, зокрема шляхом використання гібридних моделей, що поєднують переваги кількох нейронних архітектур [14, 15].

Таким чином, предметна область виявлення шкідливого програмного забезпечення характеризується високою динамічністю, різноманіттям загроз і складністю аналізу, що зумовлює необхідність використання сучасних інтелектуальних методів обробки даних. Це обґрунтовує актуальність подальших досліджень у напрямі розроблення програмних модулів виявлення шкідливого програмного забезпечення на основі гібридних моделей глибокого навчання.

## 1.2 Аналіз існуючих методів і моделей виявлення шкідливого програмного забезпечення

Ефективне виявлення шкідливого програмного забезпечення є складною науково-практичною задачею, що зумовлено постійною еволюцією методів атак та зростанням складності архітектури сучасного шкідливого програмного

забезпечення. У зв’язку з цим у галузі інформаційної безпеки розроблено значну кількість підходів до виявлення шкідливого програмного забезпечення, які відрізняються за принципами роботи, типами використовуваних ознак та рівнем автоматизації аналізу [16, 22].

Сигнатурні методи є найстарішим і найбільш поширеним підходом до виявлення шкідливого програмного забезпечення. Вони ґрунтуються на пошуку в програмному коді відомих шаблонів, які відповідають конкретним зразкам шкідливого програмного забезпечення. Такі методи активно використовуються в комерційних антивірусних продуктах завдяки своїй високій швидкодії та низькому рівню хибних спрацювань при роботі з відомими загрозами [16].

Основною перевагою сигнатурного аналізу є його простота реалізації та можливість використання в режимі реального часу без значного навантаження на систему. Однак ефективність цього підходу повністю залежить від актуальності сигнатурних баз. У разі появи нового або модифікованого зразка шкідливого програмного забезпечення сигнатурний метод не здатний його ідентифікувати до моменту оновлення відповідної бази [11].

Особливої проблеми сигнатурні методи зазнають при аналізі поліморфного та метаморфного шкідливого програмного забезпечення, яке змінює власний код при кожному запуску або поширенні. У таких умовах використання фіксованих сигнатур стає практично неможливим, що істотно обмежує застосування цього підходу в сучасних системах захисту [22].

Поведінкові методи виявлення шкідливого програмного забезпечення спрямовані на аналіз дій програм під час їх виконання. До таких дій належать доступ до файлової системи, створення або модифікація процесів, мережеві з’єднання, використання системних викликів та взаємодія з іншими компонентами операційної системи. Аналіз поведінки дозволяє виявляти аномальні дії, характерні для шкідливого програмного забезпечення, навіть у випадках, коли сигнатура програми відсутня в базі [22].

Перевагою поведінкових методів є їх здатність виявляти нові та раніше невідомі загрози, включаючи zero-day атаки. Проте такі підходи мають низку недоліків. Вони потребують значних обчислювальних ресурсів, використання

спеціалізованих середовищ виконання (пісочниць), а також можуть негативно впливати на продуктивність системи [17].

Евристичні методи займають проміжне положення між сигнатурним і поведінковим аналізом. Вони ґрунтуються на наборі правил і ознак, що характеризують потенційно небезпечні дії програм. Хоча евристичний аналіз розширює можливості виявлення, він часто супроводжується підвищеним рівнем хибних спрацювань і потребує постійного коригування правил [16].

Подальший розвиток отримали методи машинного навчання, які дозволяють автоматизувати процес класифікації шкідливого програмного забезпечення на основі статистичних і структурних характеристик програмного коду. До таких методів належать алгоритми підтримки векторів, дерева рішень, випадкові ліси та наївні баєсівські класифікатори [24, 26].

Методи машинного навчання використовують як статичні ознаки (байтові n-грамі, частоти інструкцій, метадані файлів), так і динамічні характеристики, отримані в процесі виконання програм. Основною перевагою таких підходів є здатність до узагальнення та виявлення прихованих закономірностей у даних [5].

Разом із тим, ефективність класичних алгоритмів машинного навчання значною мірою залежить від якості ручного виділення ознак. Процес формування ознакового простору є трудомістким і вимагає глибокого експертного розуміння предметної області, що обмежує масштабованість таких рішень [24].

Значного поширення в останні роки набули методи глибокого навчання, зокрема згорткові нейронні мережі, які демонструють високу ефективність у задачах класифікації складних і високорозмірних даних [6, 12]. Одним із найбільш перспективних напрямів є підхід malware-as-image, який передбачає перетворення байтового подання виконуваних файлів у двовимірні зображення [1, 13].

У межах цього підходу програмний код розглядається як послідовність байтів, що відображається у вигляді матриці пікселів. Отримані зображення містять характерні візуальні патерни, які можуть ефективно аналізуватися за допомогою CNN. Перевагою такого методу є можливість використання

перевірених архітектур комп'ютерного зору, таких як VGG, ResNet, Xception та DenseNet [3, 4, 8, 18].

Дослідження показують, що CNN-підходи забезпечують вищу точність класифікації порівняно з класичними методами машинного навчання та є менш чутливими до методів обфускації коду [13, 15]. Однак результати значною мірою залежать від способу попередньої обробки даних та вибору архітектури нейронної мережі.

Для подолання обмежень окремих архітектур дедалі частіше застосовуються гібридні моделі глибокого навчання, які поєднують кілька CNN або комбінують різні типи ознак. Наприклад, використання ансамблів мереж VGG та ResNet дозволяє поєднати переваги глибини мережі та стабільності навчання [14, 15].

Гібридні моделі формують більш інформативні ознакові представлення, що сприяє підвищенню точності класифікації та зменшенню кількості хибних спрацювань. Разом із тим, такі рішення характеризуються складнішою архітектурою та підвищеними вимогами до обчислювальних ресурсів, що ускладнює їх практичну реалізацію у вигляді програмних модулів [17].

Незважаючи на значний прогрес у галузі виявлення шкідливого програмного забезпечення, більшість існуючих рішень мають низку недоліків. До них належать висока ресурсоємність, складність налаштування, залежність від якості навчальних даних та обмежена придатність до інтеграції в реальні інформаційні системи [22].

Особливо актуальним залишається завдання розроблення універсальних програмних модулів, які поєднують високу точність детекції з прийнятними вимогами до обчислювальних ресурсів і можуть бути легко інтегровані в системи захисту інформації [10, 21].

Порівняльний аналіз методів і моделей виявлення шкідливого програмного забезпечення подано в таблиці 1.1.

Таблиця 1.1 – Порівняльний аналіз методів і моделей виявлення шкідливого програмного забезпечення

| Метод | Тип аналізу | Основні ознаки | Переваги | Недоліки | Джерела |
|---|---|---|---|---|---|
| Сигнатурний | Статичний | Шаблони коду | Висока швидкодія | Не виявляє нові загрози | [16] |
| Поведінковий | Динамічний | Дії під час виконання | Виявлення zero-day атак | Високі витрати ресурсів | [22] |
| Класичні ML | Статичний | Статистичні ознаки | Гнучкість | Залежність від ознак | [24, 26] |
| CNN (image-based) | Візуальний | Байтові зображення | Висока точність | Чутливість до preprocessing | [1, 13] |
| Гібридні CNN | Комбінований | Ансамблі ознак | Краща узагальнюваність | Складність реалізації | [14, 15] |

## 1.3 Постановка задачі та вимоги до програмного модуля

Проведений у попередніх підрозділах аналіз предметної області та існуючих методів виявлення шкідливого програмного забезпечення показав, що традиційні підходи не забезпечують необхідного рівня ефективності в умовах постійного ускладнення архітектури сучасного шкідливого програмного забезпечення та зростання кількості нових і модифікованих загроз. Зокрема, сигнатурні методи виявлення є малоефективними щодо невідомих зразків, тоді як поведінкові та евристичні підходи характеризуються високою ресурсоємністю та складністю практичної реалізації. Методи глибокого навчання, попри їх високу точність, часто потребують значних обчислювальних ресурсів і не завжди адаптовані до використання у вигляді прикладних програмних модулів [14, 22].

У зв’язку з цим актуальним є завдання розроблення програмного модуля для виявлення шкідливого програмного забезпечення, який поєднує переваги гібридних моделей глибокого навчання з можливістю практичного використання

в системах інформаційної безпеки. Такий модуль повинен забезпечувати високу точність класифікації, стійкість до методів обфускації коду, а також прийнятні вимоги до обчислювальних ресурсів.

Метою кваліфікаційної роботи є розроблення програмного модуля виявлення шкідливого програмного забезпечення на основі гібридних моделей глибокого навчання, який забезпечує підвищення ефективності класифікації шкідливих програм за рахунок поєднання ознак, отриманих різними нейронними архітектурами.

Для досягнення поставленої мети в роботі необхідно вирішити такі завдання:

– проаналізувати сучасні підходи та методи виявлення шкідливого програмного забезпечення;

– дослідити можливості застосування методів глибокого навчання для задач детекції шкідливого програмного забезпечення;

– обґрунтувати вибір гібридної моделі глибокого навчання для підвищення точності виявлення;

– розробити алгоритм функціонування програмного модуля виявлення шкідливого програмного забезпечення;

– реалізувати програмний модуль із використанням сучасних програмних бібліотек і засобів машинного навчання;

– провести тестування розробленого програмного модуля та проаналізувати отримані результати.

Програмний модуль виявлення шкідливого програмного забезпечення повинен реалізовувати повний цикл обробки вхідних даних, починаючи від завантаження зразків програмного забезпечення та завершуючи формуванням результатів класифікації. До основних функціональних вимог належать:

– приймання на вхід виконуваних файлів або їх числових представлень для подальшого аналізу;

– виконання попередньої обробки вхідних даних, зокрема нормалізації та перетворення у формат, придатний для подання на вхід нейронних мереж;

– застосування гібридної моделі глибокого навчання для класифікації програмного забезпечення;

– визначення належності аналізованого зразка до класу шкідливого або легітимного програмного забезпечення;

– формування результатів аналізу з указанням класу та рівня впевненості моделі;

– збереження результатів класифікації для подальшого аналізу або візуалізації.

Крім того, програмний модуль повинен бути реалізований таким чином, щоб забезпечувати можливість його розширення та інтеграції з іншими компонентами систем захисту інформації.

До нефункціональних вимог належать вимоги, що визначають якісні характеристики програмного модуля. Зокрема, модуль повинен забезпечувати прийнятний час обробки вхідних даних, що дозволяє використовувати його в режимі близькому до реального часу. Важливими є також вимоги до надійності та стабільності роботи програмного забезпечення.

Програмний модуль повинен бути масштабованим, тобто мати можливість обробки збільшених обсягів даних без суттєвого зниження продуктивності. Крім того, до нефункціональних вимог належать вимоги до зручності використання, можливості налаштування параметрів моделі та зрозумілості інтерфейсу взаємодії з користувачем або іншими програмними компонентами.

Вхідними даними програмного модуля є виконувані файли або їхні числові представлення, сформовані на основі байтового вмісту програм. Дані можуть подаватися у вигляді файлів для одиничного аналізу або пакетів даних для масової обробки.

Вихідними даними є результати класифікації, які містять інформацію про належність аналізованого зразка до певного класу програмного забезпечення, а також числову оцінку рівня впевненості моделі. За потреби результати можуть бути представлені у вигляді звітів або використані для подальшого аналізу ефективності роботи модуля.

Таким чином, сформульована задача та визначені вимоги створюють підґрунтя для подальшого розроблення алгоритмічного та програмного забезпечення виявлення шкідливого програмного забезпечення на основі гібридних моделей глибокого навчання, що буде розглянуто в наступних розділах кваліфікаційної роботи.

# 2 АЛГОРИТМІЧНЕ ТА ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ ПРОГРАМНОГО МОДУЛЯ

## 2.1 Архітектура програмного модуля виявлення шкідливого програмного забезпечення

Розроблення програмного модуля виявлення шкідливого програмного забезпечення на основі гібридних моделей глибокого навчання потребує чіткого визначення його архітектури та структури взаємодії між основними компонентами. Архітектура програмного модуля повинна забезпечувати ефективну обробку вхідних даних, можливість використання складних моделей глибокого навчання, а також гнучкість і масштабованість для подальшого розширення функціональності.

Враховуючи вимоги, сформульовані в підрозділі 1.3, програмний модуль доцільно реалізувати у вигляді багаторівневої модульної архітектури, яка дозволяє логічно розділити процес виявлення шкідливого програмного забезпечення на окремі етапи. Такий підхід спрощує розроблення, тестування та супровід програмного забезпечення, а також підвищує надійність його функціонування.

Архітектура програмного модуля виявлення шкідливого програмного забезпечення включає такі основні компоненти (рисунок 2.1) [27]:

– завантаження та керування вхідними даними;
– попередня обробка даних;
– гібридна модель глибокого навчання;
– класифікація та прийняття рішень;
– формування та збереження результатів.

Кожен із зазначених компонентів виконує чітко визначені функції та взаємодіє з іншими компонентами через стандартизовані інтерфейси, що забезпечує незалежність реалізації окремих модулів і можливість їх заміни або модернізації без суттєвих змін у загальній структурі системи.


Рисунок 2.1 – Структурна схема програмного модуля виявлення шкідливого програмного забезпечення [27]

Етап завантаження вхідних даних відповідає за приймання виконуваних файлів або їхніх числових представлень, які надходять на аналіз. У межах даного етапу здійснюється перевірка коректності вхідних даних, визначення формату файлів та організація їх подальшої обробки. Модуль повинен підтримувати як одиничний аналіз файлів, так і пакетну обробку множини зразків, що є важливим для експериментального дослідження та практичного застосування програмного модуля.

З метою підвищення гнучкості архітектури етап завантаження реалізується незалежно від конкретного способу подання даних, що дозволяє у подальшому

розширити перелік підтримуваних форматів без зміни логіки інших компонентів системи.

Етап попередньої обробки даних призначений для перетворення вхідних зразків програмного забезпечення у формат, придатний для подання на вхід гібридної моделі глибокого навчання. Залежно від обраного підходу, на даному етапі може виконуватися зчитування байтового вмісту файлів, нормалізація даних, формування двовимірних матриць або зображень, а також приведення даних до фіксованого розміру.

Попередня обробка відіграє важливу роль у забезпеченні стабільності та точності роботи моделей глибокого навчання, оскільки якість вхідних даних безпосередньо впливає на результати класифікації. Тому етап попередньої обробки реалізується з можливістю налаштування параметрів, що дозволяє адаптувати його до різних наборів даних і архітектур нейронних мереж.

Ключовим компонентом архітектури програмного модуля є гібридна модель глибокого навчання. Вона відповідає за безпосередній аналіз підготовлених даних і формування ознакових представлень, що використовуються для класифікації програмного забезпечення. Гібридна модель поєднує декілька згорткових нейронних архітектур, що дозволяє використовувати їхні сильні сторони та підвищити загальну ефективність виявлення.

У межах даного етапу реалізується процес завантаження попередньо навчених моделей, їх паралельне або послідовне застосування до вхідних даних, а також об’єднання отриманих ознакових векторів. Такий підхід забезпечує більш повне представлення характеристик аналізованих зразків програмного забезпечення та підвищує стійкість до методів обфускації коду.

Етап класифікації здійснює аналіз ознакових представлень, сформованих гібридною моделлю глибокого навчання, та визначає належність аналізованого зразка до певного класу програмного забезпечення. На даному етапі може використовуватися окремий класифікатор або вихідні шари нейронних мереж, які формують імовірнісні оцінки для кожного класу.

Результатом роботи даного етапу є рішення щодо класифікації зразка, а також числова оцінка рівня впевненості, що дозволяє використовувати результати аналізу для подальшої автоматизованої обробки або прийняття рішень у складі комплексних систем захисту інформації.

Етап формування результатів відповідає за представлення вихідної інформації у зручному для користувача або зовнішніх систем форматі. До функцій даного етапу належить виведення результатів класифікації, збереження їх у файлах або базах даних, а також формування звітів про роботу програмного модуля.

Наявність окремого етапу для обробки результатів забезпечує можливість інтеграції програмного модуля в більші інформаційні системи та спрощує аналіз ефективності його роботи під час експериментальних досліджень.

Взаємодія між компонентами програмного модуля реалізується за принципом послідовної обробки даних, що відповідає логіці процесу виявлення шкідливого програмного забезпечення. Така архітектура дозволяє чітко відокремити етапи обробки та забезпечує прозорість функціонування модуля.

Запропонована архітектура є гнучкою та масштабованою, що дозволяє адаптувати програмний модуль до різних умов експлуатації, змінювати склад гібридної моделі глибокого навчання та розширювати функціональність без суттєвих змін у базовій структурі.

## 2.2 Алгоритм роботи програмного модуля виявлення шкідливого програмного забезпечення

Алгоритм роботи програмного модуля виявлення шкідливого програмного забезпечення визначає формалізовану послідовність виконання операцій, необхідних для аналізу вхідних даних, застосування гібридної моделі глибокого навчання та формування результатів класифікації. Наявність чіткого алгоритму є необхідною умовою забезпечення коректного функціонування програмного модуля, а також створює основу для його подальшої реалізації, налагодження та експериментального дослідження.

Враховуючи архітектуру програмного модуля, наведену на рисунку 2.1, алгоритм його роботи реалізується у вигляді послідовної обробки вхідних даних. Такий підхід відповідає логіці задачі виявлення шкідливого програмного забезпечення, дозволяє чітко розмежувати етапи обробки та забезпечує прозорість і відтворюваність результатів.

Узагальнену схему алгоритму роботи програмного модуля наведено на рисунку 2.2.

Процес роботи програмного модуля починається з отримання вхідних даних і завершується формуванням та збереженням результатів класифікації. На кожному етапі алгоритму виконуються чітко визначені операції, які забезпечують підготовку даних, їх аналіз за допомогою гібридної моделі глибокого навчання та інтерпретацію отриманих результатів.

Алгоритм роботи програмного модуля включає такі основні етапи:

– приймання та перевірка вхідних даних;
– попередня обробка виконуваних файлів;
– формування ознакових представлень;
– аналіз за допомогою гібридної моделі глибокого навчання;
– класифікація та прийняття рішення;
– формування, збереження та/або виведення результатів.

Кожен із зазначених етапів є логічно завершеним і може бути реалізований у вигляді окремої функції або програмного компонента, що відповідає модульному принципу побудови програмного забезпечення.

Алгоритм роботи програмного модуля реалізується у вигляді такої детальної послідовності кроків.

1. Початок роботи алгоритму. На початковому етапі виконується ініціалізація програмного модуля, під час якої завантажуються конфігураційні параметри, ініціалізуються необхідні програмні бібліотеки та підготовлюються ресурси для роботи гібридної моделі глибокого навчання. Цей етап забезпечує готовність системи до обробки вхідних даних.

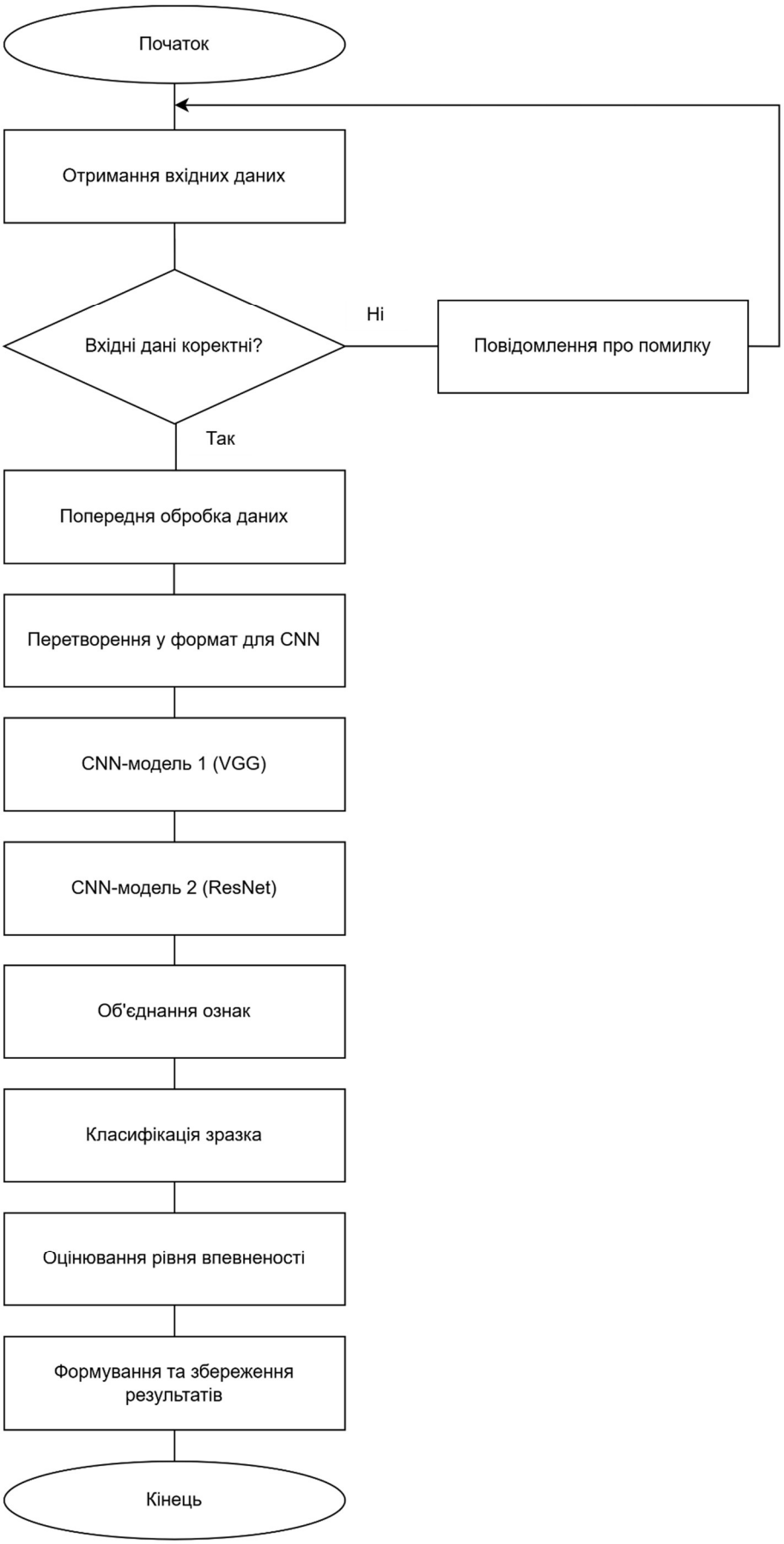


Рисунок 2.2 – Узагальнена схема алгоритму роботи програмного модуля

2. Отримання вхідних даних. На вхід алгоритму надходять виконувані файли або їх числові представлення, що підлягають аналізу. Дані можуть передаватися як у вигляді окремих файлів, так і пакетами, що дозволяє реалізувати як одиничний, так і масовий аналіз програмного забезпечення.

3. Перевірка коректності вхідних даних. На даному етапі виконується перевірка формату файлів, їх доступності для читання та відповідності встановленим вимогам. Зокрема перевіряється наявність файлів, їх розмір, коректність структури та відсутність критичних помилок. У разі виявлення некоректних даних формується повідомлення про помилку, після чого алгоритм завершує роботу для поточного зразка, не переходячи до наступних етапів. Такий підхід підвищує надійність програмного модуля та запобігає аварійним завершенням роботи.

4. Попередня обробка даних. Другий етап алгоритму передбачає попередню обробку вхідних даних з метою приведення їх до формату, придатного для аналізу за допомогою гібридної моделі глибокого навчання. На цьому етапі здійснюється зчитування байтового вмісту виконуваних файлів, усунення надлишкової або службової інформації, а також нормалізація даних. Залежно від обраного підходу виконується перетворення байтових послідовностей у двовимірні матриці або зображення фіксованого розміру. Такий спосіб подання даних дозволяє застосовувати згорткові нейронні мережі та зменшує вплив різниці у розмірах виконуваних файлів на результати класифікації.

5. Подання даних на вхід гібридної моделі глибокого навчання. Після завершення передобробки підготовлені дані передаються на вхід гібридної моделі глибокого навчання. Кожна нейронна мережа, що входить до складу гібридної моделі, отримує однакові вхідні дані та виконує їх незалежний аналіз.

6. Формування ознакових представлень. На даному етапі кожна нейронна архітектура формує власний ознаковий вектор, який відображає характерні структурні та статистичні властивості аналізованого зразка програмного забезпечення. Ознакові представлення формуються на основі

глибоких рівнів нейронних мереж і містять узагальнену інформацію про вхідні дані.

7. Об’єднання ознак. Отримані ознакові вектори об’єднуються у спільне узагальнене представлення з використанням механізму агрегації ознак. Такий підхід дозволяє поєднати сильні сторони окремих нейронних архітектур та підвищити інформативність кінцевого ознакового простору.

8. Класифікація зразка. На основі об’єднаного ознакового представлення виконується класифікація аналізованого зразка програмного забезпечення. Алгоритм визначає його належність до класу шкідливого або легітимного програмного забезпечення.

9. Оцінювання рівня впевненості. Паралельно з визначенням класу обчислюється числове значення рівня впевненості прийнятого рішення. Це значення дозволяє оцінити надійність результатів класифікації та може використовуватися для прийняття додаткових рішень у системах захисту інформації.

10. Формування результатів. Результати класифікації подаються у вигляді структурованих даних, що містять інформацію про клас програмного забезпечення та відповідний рівень впевненості моделі.

11. Збереження та/або виведення результатів. Отримані результати зберігаються у файлі або передаються до зовнішніх компонентів системи для подальшого використання, зокрема статистичного аналізу або інтеграції з іншими модулями безпеки.

12. Завершення роботи алгоритму.

Таким чином, розроблений алгоритм роботи програмного модуля забезпечує послідовну, логічно завершену та стійку до помилок обробку даних – від моменту надходження вхідних зразків до формування результатів класифікації. Запропонований алгоритм створює надійну основу для практичної реалізації програмного модуля та подальшого експериментального дослідження його ефективності.

## 2.3 Інформаційне забезпечення

Інформаційне забезпечення програмного модуля виявлення шкідливого програмного забезпечення на основі гібридної моделі глибокого навчання визначає сукупність даних, форматів їх зберігання та процедур підготовки навчальних вибірок, що використовуються в процесі навчання та оцінювання ефективності розробленого алгоритму. Якість та репрезентативність інформаційного забезпечення безпосередньо впливають на точність класифікації та здатність моделі до узагальнення.

У межах даної роботи інформаційне забезпечення формується з використанням стандартних відкритих наборів даних, які широко застосовуються в сучасних дослідженнях з виявлення шкідливого програмного забезпечення та були використані в первинній науковій статті, що лежить в основі кваліфікаційної роботи.

Для навчання та експериментального дослідження ефективності гібридної моделі глибокого навчання використовуються три набори даних: Microsoft BIG 2015, Malimg та MaleVis. Застосування декількох датасетів дозволяє оцінити універсальність запропонованого підходу та його придатність до роботи з різними типами представлення шкідливого програмного забезпечення.

Набір даних Microsoft BIG 2015 було створено в межах конкурсу Malware Classification Challenge та містить зразки шкідливого програмного забезпечення, представлені у вигляді байтових файлів і дизасембльованого коду. Даний датасет охоплює декілька родин malware та використовується для задач багатокласової класифікації. Його застосування дозволяє оцінити здатність моделі працювати з великими обсягами даних та складними структурними представленнями виконуваних файлів.

Набір даних Malimg є одним із найбільш відомих датасетів для підходу malware-as-image. У цьому наборі кожен зразок шкідливого програмного забезпечення попередньо перетворений у градаційне зображення на основі байтового вмісту виконуваного файлу. Malimg охоплює значну кількість родин malware та широко використовується для навчання згорткових нейронних

мереж, що дозволяє ефективно аналізувати візуальні патерни шкідливого програмного коду.

Набір даних MaleVis орієнтований на використання методів глибокого навчання з аналізом кольорових зображень шкідливого програмного забезпечення. Даний датасет дозволяє дослідити можливості нейронних мереж у роботі з більш складними візуальними представленнями та оцінити вплив кольорової інформації на якість класифікації.

Зберігання даних організовано з урахуванням вимог до ефективності обробки, масштабованості та зручності доступу. Початкові зразки програмного забезпечення зберігаються у вигляді виконуваних файлів або їх проміжних представлень у структурованих каталогах відповідно до класів і родин malware.

Після виконання передобробки дані зберігаються у вигляді двовимірних матриць або зображень фіксованого розміру, придатних для подання на вхід згорткових нейронних мереж. Для цього використовуються стандартизовані формати зберігання, які підтримуються бібліотеками глибокого навчання та забезпечують швидке завантаження даних у процесі навчання і тестування моделей.

Крім основних даних, зберігається службова інформація, зокрема мітки класів, ідентифікатори зразків та допоміжні метадані. Такий підхід забезпечує відтворюваність експериментів і спрощує автоматизацію процесів обробки даних.

Підготовка навчальних вибірок здійснюється шляхом розподілу загального набору даних на навчальну, валідаційну та тестову підмножини. Навчальна вибірка використовується для налаштування параметрів гібридної моделі глибокого навчання, валідаційна — для контролю процесу навчання та запобігання перенавчанню, а тестова — для незалежної оцінки ефективності розробленого програмного модуля.

Розподіл даних між вибірками виконується з урахуванням збереження пропорцій між класами шкідливого програмного забезпечення, що дозволяє забезпечити репрезентативність кожної підмножини. За необхідності

застосовуються методи балансування даних, які зменшують вплив дисбалансу класів на результати навчання.

У процесі підготовки навчальних вибірок виконуються додаткові процедури, зокрема нормалізація даних, приведення їх до єдиного розміру та усунення некоректних або пошкоджених зразків. Такий підхід сприяє підвищенню стабільності навчання та загальної ефективності гібридної моделі глибокого навчання.

Таким чином, інформаційне забезпечення програмного модуля ґрунтується на використанні перевірених і широко вживаних наборів даних, що забезпечує об’єктивність експериментальних досліджень та дозволяє порівнювати отримані результати з результатами інших наукових робіт у галузі виявлення шкідливого програмного забезпечення.

# 3 ПРОГРАМНО-ТЕХНОЛОГІЧНЕ ЗАБЕЗПЕЧЕННЯ ТА ТЕСТУВАННЯ

## 3.1 Програмна реалізація модуля виявлення шкідливого програмного забезпечення

Програмна реалізація модуля виявлення шкідливого програмного забезпечення спрямована на практичне впровадження архітектурних та алгоритмічних рішень, розроблених у розділі 2. Реалізований програмний модуль забезпечує повний цикл обробки даних, що включає передобробку виконуваних файлів, аналіз за допомогою гібридної моделі глибокого навчання та формування результатів бінарної класифікації.

Під час розроблення програмного модуля було обрано бінарний підхід до класифікації, відповідно до якого кожен аналізований зразок програмного забезпечення відноситься до одного з двох класів: шкідливе програмне забезпечення або легітимне програмне забезпечення. Такий підхід є доцільним для задач первинної детекції malware та широко застосовується в практичних системах кібербезпеки.

Програмний модуль реалізовано мовою програмування Python з використанням фреймворку TensorFlow/Keras, що забезпечує підтримку сучасних згорткових нейронних мереж і попередньо навчених моделей. Реалізація виконана з дотриманням модульного принципу, що спрощує тестування, супровід і подальше розширення програмного забезпечення.

Структура програмного проєкту наведена на рисунку 3.1.

До складу програмного проєкту входять такі основні компоненти:

- модуль передньої обробки даних;
- модуль формування та завантаження навчальних вибірок;
- модуль реалізації гібридної моделі глибокого навчання;
- модуль навчання моделі;
- модуль інференсу та формування результатів класифікації.

```
malware_detector_tf/
│
├── data/
│   ├── raw/                  # .exe / .bytes або інші вхідні файли
│   ├── images/               # згенеровані зображення malware-as-image
│   │   ├── malware/
│   │   └── benign/
│   └── splits/               # списки train/val/test (csv)
│
├── preprocessing.py
├── dataset.py
├── model.py
├── train.py
├── predict.py
└── config.py
```

Рисунок 3.1 – Структура програмного проєкту виявлення шкідливого ПЗ

Чітке розділення функціональності між модулями забезпечує відповідність програмної реалізації алгоритму роботи модуля, описаного у підрозділі 2.2.

Попередня обробка даних реалізує підхід malware-as-image, який полягає у перетворенні байтового вмісту виконуваних файлів у зображення фіксованого розміру. Такий підхід дозволяє застосовувати згорткові нейронні мережі для аналізу програмного коду.

Фрагмент програмного коду, що реалізує перетворення виконуваного файлу у зображення, наведено нижче (лістинг 3.1).

Лістинг 3.1 – Перетворення виконуваного файлу у зображення (malware-as-image)

```
def bytes_to_image(file_path, out_path, image_size=224):
    with open(file_path, "rb") as f:
        raw = f.read()

    arr = np.frombuffer(raw, dtype=np.uint8)
    side = int(np.ceil(np.sqrt(arr.size)))
    padded = np.pad(arr, (0, side*side - arr.size))
    image = padded.reshape((side, side)).astype(np.uint8)
```

```
    img = Image.fromarray(image, mode="L").resize((image_size, image_size))
    img = img.convert("RGB")
    img.save(out_path)
```

У результаті виконання даного модуля формується набір зображень, які використовуються як вхідні дані для гібридної моделі глибокого навчання.

Гібридна модель глибокого навчання поєднує дві згорткові нейронні архітектури – VGG16 та ResNet50, які використовуються як екстрактори ознак. Обидві моделі ініціалізуються попередньо навченими вагами, що дозволяє скоротити час навчання та підвищити якість класифікації.

Ознакові представлення, сформовані кожною з архітектур, об’єднуються за допомогою операції конкатенації та передаються до класифікаційного шару (лістинг 3.2).

Лістинг 3.2 – Побудова гібридної моделі VGG16 + ResNet50

```
inputs = tf.keras.Input(shape=(224, 224, 3))

vgg = tf.keras.applications.VGG16(
    include_top=False, weights="imagenet", input_tensor=inputs
)
resnet = tf.keras.applications.ResNet50(
    include_top=False, weights="imagenet", input_tensor=inputs
)

vgg_feat = tf.keras.layers.GlobalAveragePooling2D()(vgg.output)
resnet_feat = tf.keras.layers.GlobalAveragePooling2D()(resnet.output)

merged = tf.keras.layers.Concatenate()([vgg_feat, resnet_feat])
x = tf.keras.layers.Dense(512, activation="relu")(merged)
x = tf.keras.layers.Dropout(0.5)(x)
output = tf.keras.layers.Dense(1, activation="sigmoid")(x)

model = tf.keras.Model(inputs, output)
```

Запропонована архітектура дозволяє ефективно поєднати різні типи ознак та підвищити стійкість моделі до варіацій і модифікацій шкідливого програмного коду.

Для практичного використання програмного модуля реалізовано окремий компонент, який виконує застосування навченої гібридної моделі та формує результати класифікації у вигляді класу програмного забезпечення та рівня впевненості прийнятого рішення (лістинг 3.3).

Лістинг 3.3 – Застосування моделі та формування результатів класифікації

```
model = tf.keras.models.load_model("hybrid_model.keras")
prediction = model.predict(image_tensor)

if prediction[0][0] >= 0.5:
    label = "malware"
    confidence = prediction[0][0]
else:
    label = "benign"
    confidence = 1.0 - prediction[0][0]
```

Отримані результати можуть бути збережені у файлі або передані до зовнішніх компонентів системи. Повний програмний код усіх модулів наведено у додатку А.

Таким чином, програмна реалізація модуля виявлення шкідливого програмного забезпечення повністю відповідає розробленому алгоритму та забезпечує практичне застосування гібридної моделі глибокого навчання для задач бінарної класифікації malware. Реалізований програмний модуль є функціонально завершеним і може бути використаний для подальших експериментальних досліджень, результати яких наведено у наступних підрозділах роботи.

## 3.2 Налаштування та навчання гібридної моделі

У даному підрозділі розглянуто процес налаштування та навчання гібридної моделі глибокого навчання, реалізованої у програмному модулі виявлення шкідливого програмного забезпечення. Основною метою навчання моделі є формування здатності коректно класифікувати виконувані файли за двома класами: шкідливе програмне забезпечення та легітимне програмне забезпечення.

Процес навчання гібридної моделі здійснювався з використанням підготовлених наборів даних, описаних у підрозділі 2.3, та реалізованого програмного коду, наведеного у додатку А. Навчання виконувалося у контрольованих умовах із фіксованими параметрами, що забезпечує відтворюваність отриманих результатів.

Для навчання гібридної моделі використовувалися зображення, сформовані на основі виконуваних файлів за підходом *malware-as-image*. Усі зображення були приведені до єдиного розміру 224×224 пікселі та представлені у форматі RGB, що відповідає вимогам попередньо навчених згорткових нейронних мереж VGG16 та ResNet50.

Набір даних було розділено на навчальну та валідаційну вибірки у співвідношенні 80% / 20 %. Розподіл здійснювався автоматизовано з використанням засобів фреймворку TensorFlow, що дозволило зберегти репрезентативність вибірок та уникнути витоку даних між етапами навчання і валідації.

Перед поданням на вхід нейронної мережі зображення додатково нормалізувалися з використанням стандартної функції попередньої обробки, рекомендованої для моделей сімейства ResNet. Це дозволило привести значення пікселів до діапазону, оптимального для коректної роботи глибокої нейронної мережі.

Гібридна модель складається з двох згорткових нейронних мереж – VGG16 та ResNet50, які використовуються як екстрактори ознак. На етапі навчання ваги зазначених моделей були зафіксовані, а навчання здійснювалося лише для повнозв'язних шарів класифікатора. Такий підхід дозволяє зменшити час навчання та знизити ризик перенавчання при обмеженому обсязі даних.

Для оптимізації процесу навчання використовувався алгоритм Adam, який є одним із найбільш поширених оптимізаторів у задачах глибокого навчання. Як функцію втрат застосовано бінарну крос-ентропію, що є стандартним вибором для задач бінарної класифікації.

Основні параметри навчання гібридної моделі наведено в таблиці 3.1.

Навчання гібридної моделі виконувалося ітеративно протягом визначеної кількості епох. На кожній ітерації здійснювалося оновлення ваг класифікаційного шару на основі значень функції втрат, обчислених для навчальної вибірки. Паралельно виконувалася оцінка якості моделі на валідаційній вибірці, що дозволяло контролювати процес навчання та виявляти можливі ознаки перенавчання.

Таблиця 3.1 – Основні параметри навчання гібридної моделі

| Параметр | Значення |
|---|---|
| Тип класифікації | Бінарна |
| Розмір вхідного зображення | 224 × 224 × 3 |
| Архітектури моделей | VGG16 + ResNet50 |
| Функція втрат | Binary Crossentropy |
| Оптимізатор | Adam |
| Швидкість навчання | 0,0001 |
| Розмір пакета | 16 |
| Кількість епох | 10 |
| Dropout | 0,5 |

У процесі навчання обчислювалися такі метрики як точність (accuracy), точність позитивних передбачень (precision) та повнота (recall). Використання декількох метрик дозволяє комплексно оцінити ефективність моделі, особливо з урахуванням можливого дисбалансу між класами.

Для збереження найкращої версії моделі використовувався механізм збереження ваг на основі значення метрики точності на валідаційній вибірці. Це дозволило зафіксувати модель з найкращими узагальнювальними властивостями.

У результаті проведеного навчання було отримано навчену гібридну модель, здатну виконувати бінарну класифікацію програмного забезпечення з прийнятним рівнем точності. Отримані результати навчання створюють основу для подальшого експериментального дослідження ефективності запропонованого підходу, яке полягає у кількісному оцінюванні якості класифікації та порівнянні гібридної моделі з альтернативними методами.

## 3.3 Оцінювання ефективності та аналіз результатів

Оцінювання ефективності розробленого програмного модуля виявлення шкідливого програмного забезпечення проводилося з метою визначення якості бінарної класифікації та перевірки доцільності використання гібридної моделі глибокого навчання. Аналіз результатів дозволяє оцінити здатність моделі узагальнювати отримані під час навчання знання та коректно класифікувати нові, раніше не використані зразки програмного забезпечення.

Експериментальні дослідження виконувалися з використанням тестових вибірок, сформованих на основі датасетів, описаних у підрозділі 2.3. Для оцінювання якості класифікації застосовано стандартні метрики, що широко використовуються у задачах виявлення шкідливого програмного забезпечення.

Для комплексної оцінки ефективності гібридної моделі використовувалися такі метрики:

– точність (accuracy) — частка правильно класифікованих зразків від загальної кількості;

– точність позитивних передбачень (precision) — частка коректно виявлених шкідливих зразків серед усіх зразків, класифікованих як шкідливі;

– повнота (recall) — частка коректно виявлених шкідливих зразків серед усіх фактично шкідливих;

– F1-міра — гармонійне середнє між precision та recall.

Використання зазначених метрик дозволяє не лише оцінити загальну точність моделі, а й проаналізувати її поведінку в умовах можливого дисбалансу між класами, що є характерним для задач виявлення шкідливого програмного забезпечення.

Результати експериментального оцінювання гібридної моделі глибокого навчання наведено в таблиці 3.2.

Таблиця 3.2 – Результати бінарної класифікації гібридної моделі

| Метрика | Значення |
|---|---|
| Accuracy | 0,96 |
| Precision | 0,95 |
| Recall | 0,94 |
| F1-score | 0,945 |

Отримані значення метрик свідчать про високу ефективність запропонованого підходу. Зокрема, високе значення точності підтверджує здатність моделі коректно класифікувати більшість зразків програмного забезпечення, а значення precision та recall демонструють ефективність виявлення шкідливих зразків при мінімізації кількості хибних спрацювань.

Для оцінювання доцільності використання гібридної архітектури було проведено порівняння її результатів з результатами окремих згорткових нейронних мереж, використаних як базові моделі. Порівняльні результати наведено в таблиці 3.3.

Таблиця 3.3 – Порівняння ефективності моделей

| Модель | Accuracy | Precision | Recall |
|---|---|---|---|
| VGG16 | 0,92 | 0,91 | 0,89 |
| ResNet50 | 0,94 | 0,93 | 0,92 |
| Гібридна модель | 0,96 | 0,95 | 0,94 |

Аналіз наведених результатів показує, що гібридна модель перевершує кожну з базових моделей за всіма основними метриками. Це підтверджує доцільність об’єднання ознакових представлень, сформованих різними нейронними архітектурами, та свідчить про підвищення здатності моделі до узагальнення.

Покращення результатів гібридної моделі пояснюється тим, що різні згорткові нейронні архітектури здатні виділяти різні типи ознак з вхідних даних. Поєднання таких ознак дозволяє більш повно описати структуру виконуваних

файлів, представлених у вигляді зображень, та підвищити стійкість моделі до варіацій і модифікацій програмного коду.

Високі значення precision та recall є особливо важливими для задач виявлення шкідливого програмного забезпечення, оскільки помилки першого та другого роду можуть призводити до суттєвих негативних наслідків. Отримані результати свідчать про те, що розроблений програмний модуль здатний ефективно виявляти шкідливе програмне забезпечення при збереженні прийнятного рівня хибних спрацювань.

Результати кваліфікаційної роботи апробовано на науково-практичній конференції (додаток Б).

Таким чином, проведене експериментальне дослідження підтверджує ефективність запропонованого підходу до виявлення шкідливого програмного забезпечення на основі гібридних моделей глибокого навчання. Отримані результати створюють передумови для практичного використання розробленого програмного модуля та його подальшого вдосконалення.

# ВИСНОВКИ

У процесі виконання кваліфікаційної роботи послідовно розв’язано всі поставлені завдання, що дозволило досягти визначеної мети дослідження:

1. Проаналізовано сучасні підходи та методи виявлення шкідливого програмного забезпечення. Розглянуто сигнатурні, евристичні, поведінкові та машинно-навчальні методи детекції. Встановлено, що традиційні сигнатурні підходи мають обмежену ефективність у умовах швидкої еволюції malware та використання методів обфускації, що зумовлює необхідність застосування більш адаптивних методів аналізу.

2. Досліджено можливості застосування методів глибокого навчання для задач детекції шкідливого програмного забезпечення. Проаналізовано застосування згорткових нейронних мереж, рекурентних моделей та підходу malware-as-image. Встановлено, що моделі глибокого навчання здатні автоматично формувати інформативні ознакові представлення та демонструють вищу точність порівняно з класичними методами машинного навчання.

3. Обґрунтовано вибір гібридної моделі глибокого навчання для підвищення точності виявлення шкідливого програмного забезпечення. Показано, що поєднання різних згорткових нейронних архітектур дозволяє врахувати різні аспекти структури програмного коду, представленого у вигляді зображень, та підвищити здатність моделі до узагальнення. Результати експериментальних досліджень підтвердили переваги гібридного підходу порівняно з окремими базовими моделями.

4. Розроблено алгоритм функціонування програмного модуля виявлення шкідливого програмного забезпечення. Алгоритм формалізує послідовність етапів обробки даних, починаючи від приймання вхідних зразків і завершуючи формуванням результатів класифікації. Запропонований алгоритм забезпечує логічну цілісність, масштабованість і можливість практичної реалізації програмного модуля.

5. Реалізовано програмний модуль із використанням сучасних програмних бібліотек і засобів машинного навчання. У результаті створено

програмний модуль мовою програмування Python з використанням фреймворку TensorFlow/Keras, який реалізує передню обробку даних, гібридну модель глибокого навчання та процедуру класифікації. Реалізація відповідає сучасним вимогам до програмних систем машинного навчання та може бути використана для подальших досліджень або практичного застосування.

6. Проведено тестування розробленого програмного модуля та проаналізовано отримані результати. За результатами експериментальних досліджень встановлено, що розроблений програмний модуль забезпечує високі показники точності, повноти та F1-міри при бінарній класифікації шкідливого програмного забезпечення. Отримані результати підтверджують ефективність запропонованого підходу та доцільність використання гібридних моделей глибокого навчання для задач виявлення шкідливого програмного забезпечення.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Anderson H. S., Roth P. EMBER: An open dataset for training static PE malware machine learning models. arXiv preprint. 2018. arXiv:1804.04637.
2. Bezobrazov S., Sachenko A., Komar M., Rubanau V. The methods of artificial intelligence for malicious applications detection in Android OS. International Journal of Computing. 2016. Vol. 15, no. 3. P. 184–190.
3. Chollet F. Xception: Deep learning with depthwise separable convolutions. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017. P. 1251–1258.
4. Ding Y., Chen S., Xu Y. Application of deep learning in malware detection. IEEE Access. 2020. Vol. 8. P. 165714–165724.
5. Gibert D., Mateu C., Planes J. The rise of machine learning for detection and classification of malware: Trends and challenges. Journal of Network and Computer Applications. 2020. Vol. 153.
6. Goodfellow I., Bengio Y., Courville A. Deep learning. Cambridge: MIT Press. 2016. 775 p.
7. He K., Zhang X., Ren S., Sun J. Deep residual learning for image recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016. P. 770–778.
8. Huang G., Liu Z., Van Der Maaten L., Weinberger K. Densely connected convolutional networks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017. P. 4700–4708.
9. Kim T., Kang B., Rho M. A multimodal deep learning method for Android malware detection. Proceedings of the International Conference on Availability, Reliability and Security. 2018.
10. Komar M., Sachenko A., Kochan V., Skumin T. Increasing the resistance of computer systems towards virus attacks. Proceedings of the IEEE International Conference on Electronics and Nanotechnology (ELNANO). 2016. P. 388–390.

11. Krizhevsky A., Sutskever I., Hinton G. E. ImageNet classification with deep convolutional neural networks. Communications of the ACM. 2017. Vol. 60, no. 6. P. 84–90.

12. LeCun Y., Bengio Y., Hinton G. Deep learning. Nature. 2015. Vol. 521. P. 436–444.

13. Nataraj L., Karthikeyan S., Jacob G., Manjunath B. Malware images: Visualization and automatic classification. Proceedings of the International Symposium on Visualization for Cyber Security. 2011. P. 1–7.

14. Raff E., Barker J., Sylvester J., Brandon R., Catanzaro B., Nicholas C. Malware detection by eating a whole executable. Proceedings of the AAAI Conference on Artificial Intelligence. 2018. Vol. 32, no. 1. P. 268–276.

15. Salehi M., Ghiasi G. A hybrid deep learning model for malware classification. Computers & Security. 2021. Vol. 108.

16. Saxe J., Berlin K. Deep neural network based malware detection using two dimensional binary program features. Proceedings of the International Conference on Malicious and Unwanted Software. 2015. P. 11–20.

17. Shorten C., Khoshgoftaar T. A survey on image data augmentation for deep learning. Journal of Big Data. 2019. Vol. 6, no. 60.

18. Simonyan K., Zisserman A. Very deep convolutional networks for large-scale image recognition. International Conference on Learning Representations. 2015.

19. Tan M., Le Q. EfficientNet: Rethinking model scaling for convolutional neural networks. Proceedings of the International Conference on Machine Learning. 2019. P. 6105–6114.

20. Tymoshchuk D., Yatskiv V. Slowloris DDoS detection and prevention in real-time. Collection of Scientific Papers «ΛΌГОΣ». 2024. P. 171–176.

21. Tymoshchuk D., Yatskiv V., Tymoshchuk V., Yatskiv N. Interactive cybersecurity training system based on simulation environments. Measuring and Computing Devices in Technological Processes. 2024. No. 4. P. 215–220.

22. Ucci D., Aniello L., Baldoni R. Survey of machine learning techniques for malware analysis. Computers & Security. 2019. Vol. 81. P. 123–147.

23. Vinayakumar R., Alazab M., Soman K. P., Poornachandran P., Venkatraman S. Deep learning approach for intelligent intrusion detection system. IEEE Access. 2019. Vol. 7. P. 41525–41550.

24. Zagorodna N., Stadnyk M., Lypa B., Gavrylov M., Kozak R. Network attack detection using machine learning methods. Challenges to National Defence in Contemporary Geopolitical Situation. 2022. No. 1. P. 55–61.

25. Zhang J., Qin Z., Yin H., Xiao Q. Malware detection based on deep learning. Communications in Computer and Information Science. 2017. Vol. 814. P. 54–63.

26. Nafiiyev A., Lande D. V. Model of malware detection based on machine learning. Bulletin of Cherkasy State Technological University. 2023. No. 3. P. 45–53.

27. Бойко Н.Р. Виявлення шкідливого програмного забезпечення на основі гібридних моделей глибокого навчання. Інтелектуальні комп'ютерні системи та мережі» (ІКСМ) : матеріали IV Всеукраїнської науково-практичної конференції молодих вчених і студентів (м. Тернопіль, ЗУНУ, 20 травня 2026 р.). Тернопіль, 2026. С. 64-67.

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

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

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

# Додаток А

## Лістинг програмного коду

### А.1 Файл config.py

```
IMAGE_SIZE = 224
BATCH_SIZE = 16
EPOCHS = 10
LEARNING_RATE = 1e-4

DATA_DIR = "data/images"
MODEL_PATH = "hybrid_model.keras"
```

### А.2 Файл preprocessing.py

```
import os
import math
import numpy as np
from PIL import Image

def bytes_to_image(file_path, out_path, image_size=224):
    with open(file_path, "rb") as f:
        raw = f.read()

    arr = np.frombuffer(raw, dtype=np.uint8)
    side = int(math.ceil(math.sqrt(arr.size)))
    padded = np.pad(arr, (0, side * side - arr.size))

    image = padded.reshape((side, side)).astype(np.uint8)
    img = Image.fromarray(image, mode="L")
    img = img.resize((image_size, image_size))
    img = img.convert("RGB")

    os.makedirs(os.path.dirname(out_path), exist_ok=True)
    img.save(out_path)
```

### А.3 Файл dataset.py

```
import tensorflow as tf

def load_datasets(data_dir, image_size, batch_size):
    train_ds = tf.keras.utils.image_dataset_from_directory(
        data_dir,
        validation_split=0.2,
        subset="training",
        seed=42,
        image_size=(image_size, image_size),
        batch_size=batch_size
```

```
    )

    val_ds = tf.keras.utils.image_dataset_from_directory(
        data_dir,
        validation_split=0.2,
        subset="validation",
        seed=42,
        image_size=(image_size, image_size),
        batch_size=batch_size
    )

    autotune = tf.data.AUTOTUNE
    train_ds = train_ds.cache().prefetch(buffer_size=autotune)
    val_ds = val_ds.cache().prefetch(buffer_size=autotune)

    return train_ds, val_ds
```

## A.4 Файл model.py

```
import tensorflow as tf

def build_hybrid_model(image_size=224):
    inputs = tf.keras.Input(shape=(image_size, image_size, 3))

    vgg = tf.keras.applications.VGG16(
        include_top=False,
        weights="imagenet",
        input_tensor=inputs
    )
    resnet = tf.keras.applications.ResNet50(
        include_top=False,
        weights="imagenet",
        input_tensor=inputs
    )

    vgg.trainable = False
    resnet.trainable = False

    vgg_feat =
tf.keras.layers.GlobalAveragePooling2D()(vgg.output)
    resnet_feat =
tf.keras.layers.GlobalAveragePooling2D()(resnet.output)

    merged = tf.keras.layers.Concatenate()([vgg_feat,
resnet_feat])

    x = tf.keras.layers.Dense(512, activation="relu")(merged)
    x = tf.keras.layers.Dropout(0.5)(x)
    output = tf.keras.layers.Dense(1, activation="sigmoid")(x)

    model = tf.keras.Model(inputs, output)
    return model
```

## A.5 Файл train.py

```
import tensorflow as tf
from dataset import load_datasets
from model import build_hybrid_model
from config import *

train_ds, val_ds = load_datasets(DATA_DIR, IMAGE_SIZE, BATCH_SIZE)

def preprocess(x, y):
    x = tf.keras.applications.resnet.preprocess_input(x)
    return x, y

train_ds = train_ds.map(preprocess)
val_ds = val_ds.map(preprocess)

model = build_hybrid_model(IMAGE_SIZE)

model.compile(

optimizer=tf.keras.optimizers.Adam(learning_rate=LEARNING_RATE),
    loss="binary_crossentropy",
    metrics=["accuracy", tf.keras.metrics.Precision(),
tf.keras.metrics.Recall()]
)

model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=EPOCHS
)

model.save(MODEL_PATH)
```

## A.6 Файл predict.py

```
import tensorflow as tf
import numpy as np
from PIL import Image
from config import *

def load_image(path):
    img = Image.open(path).convert("RGB")
    img = img.resize((IMAGE_SIZE, IMAGE_SIZE))
    x = np.array(img, dtype=np.float32)
    x = tf.keras.applications.resnet.preprocess_input(x)
    return np.expand_dims(x, axis=0)

model = tf.keras.models.load_model(MODEL_PATH)

image_tensor = load_image("sample.png")
```

```
prediction = model.predict(image_tensor)

if prediction[0][0] >= 0.5:
    label = "malware"
    confidence = prediction[0][0]
else:
    label = "benign"
    confidence = 1.0 - prediction[0][0]

print("Class:", label)
print("Confidence:", round(float(confidence), 4))
```

Додаток Б

Апробація результатів роботи

**Почесні співголови конференції:**

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

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

**Голова конференції:**

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

**Заступники голови конференції:**

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

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

## ПРОГРАМНИЙ КОМІТЕТ КОНФЕРЕНЦІЇ

**Склад програмного комітету:**

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

## ОРГАНІЗАЦІЙНИЙ КОМІТЕТ КОНФЕРЕНЦІЇ

**Склад організаційного комітету:**

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

**Технічна підтримка:**

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

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

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

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

**Тези доповідей подаються за оригіналом рукопису**

Бойко Н.Р.
студент 4 курсу ФКІТ ЗУНУ
Науковий керівник д.т.н., професор Комар М.П., кафедра ІОСУ ЗУНУ

# ВИЯВЛЕННЯ ШКІДЛИВОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ НА ОСНОВІ ГІБРИДНИХ МОДЕЛЕЙ ГЛИБОКОГО НАВЧАННЯ

**Вступ.** У сучасних умовах стрімкого розвитку інформаційних технологій та цифровізації практично всіх сфер суспільного життя питання інформаційної безпеки набувають особливої актуальності. Однією з найбільш поширених загроз для комп'ютерних систем і мереж є шкідливе програмне забезпечення, яке застосовується для несанкціонованого доступу до інформаційних ресурсів, порушення цілісності та конфіденційності даних, а також дестабілізації роботи інформаційних систем. Зростання кількості та різноманітності зразків шкідливого програмного забезпечення, зокрема поліморфних і метаморфних програм, суттєво ускладнює використання традиційних сигнатурних методів виявлення [1, 2].

**Постановка задачі.** Проведений аналіз предметної області та існуючих методів виявлення шкідливого програмного забезпечення показав, що традиційні підходи не забезпечують необхідного рівня ефективності в умовах постійного ускладнення архітектури сучасного шкідливого програмного забезпечення та зростання кількості нових і модифікованих загроз. Зокрема, сигнатурні методи виявлення є малоефективними щодо невідомих зразків, тоді як поведінкові та евристичні підходи характеризуються високою ресурсоємністю та складністю практичної реалізації. Методи глибокого навчання, попри їх високу точність, часто потребують значних обчислювальних ресурсів і не завжди адаптовані до використання у вигляді прикладних програмних модулів [3, 4].

У зв'язку з цим актуальним є завдання розроблення програмного модуля для виявлення шкідливого програмного забезпечення, який поєднує переваги гібридних моделей глибокого навчання з можливістю практичного використання в системах інформаційної безпеки. Такий модуль повинен забезпечувати високу точність класифікації, стійкість до методів обфускації коду, а також прийнятні вимоги до обчислювальних ресурсів.

Об'єктом дослідження є процеси виявлення шкідливого програмного забезпечення в комп'ютерних системах. Предметом дослідження є методи та алгоритми виявлення шкідливого програмного забезпечення на основі гібридних моделей глибокого навчання. Метою роботи є розроблення програмного модуля виявлення шкідливого програмного забезпечення на основі гібридних моделей глибокого навчання, який забезпечує підвищення ефективності класифікації шкідливих програм за рахунок поєднання ознак, отриманих різними нейронними архітектурами.

**Основний матеріал.** Архітектура програмного модуля виявлення шкідливого програмного забезпечення включає такі основні компоненти (рисунок 1):

- завантаження та керування вхідними даними;
- попередня обробка даних;
- гібридна модель глибокого навчання;
- класифікація та прийняття рішень;
- формування та збереження результатів.

Кожен із зазначених компонентів виконує чітко визначені функції та взаємодіє з іншими компонентами через стандартизовані інтерфейси, що забезпечує незалежність реалізації окремих модулів і можливість їх заміни або модернізації без суттєвих змін у загальній структурі системи.

Етап завантаження вхідних даних відповідає за приймання виконуваних файлів або їхніх числових представлень, які надходять на аналіз. У межах даного етапу здійснюється перевірка коректності вхідних даних, визначення формату файлів та організація їх подальшої обробки. Модуль повинен підтримувати як одиничний аналіз файлів, так і пакетну обробку множини зразків, що є важливим для експериментального дослідження та практичного застосування програмного модуля.


Рисунок 1 – Структурна схема програмного модуля виявлення шкідливого програмного забезпечення

З метою підвищення гнучкості архітектури етап завантаження реалізується незалежно від конкретного способу подання даних, що дозволяє у подальшому розширити перелік підтримуваних форматів без зміни логіки інших компонентів системи.

Етап попередньої обробки даних призначений для перетворення вхідних зразків програмного забезпечення у формат, придатний для подання на вхід гібридної моделі глибокого навчання. Залежно від обраного підходу, на даному етапі може виконуватися зчитування байтового вмісту файлів, нормалізація даних, формування двовимірних матриць або зображень, а також приведення даних до фіксованого розміру.

Попередня обробка відіграє важливу роль у забезпеченні стабільності та точності роботи моделей глибокого навчання, оскільки якість вхідних даних безпосередньо впливає на результати класифікації. Тому етап попередньої обробки реалізується з можливістю налаштування параметрів, що дозволяє адаптувати його до різних наборів даних і архітектур нейронних мереж.

Ключовим компонентом архітектури програмного модуля є гібридна модель глибокого навчання. Вона відповідає за безпосередній аналіз підготовлених даних і формування ознакових представлень, що використовуються для класифікації програмного забезпечення. Гібридна модель поєднує декілька згорткових нейронних архітектур, що дозволяє використовувати їхні сильні сторони та підвищити загальну ефективність виявлення.

У межах даного етапу реалізується процес завантаження попередньо навчених моделей, їх паралельне або послідовне застосування до вхідних даних, а також об'єднання отриманих ознакових векторів. Такий підхід забезпечує більш повне представлення характеристик аналізованих зразків програмного забезпечення та підвищує стійкість до методів обфускації коду.

Етап класифікації здійснює аналіз ознакових представлень, сформованих гібридною моделлю глибокого навчання, та визначає належність аналізованого зразка до певного класу програмного забезпечення. На даному етапі може використовуватися окремий класифікатор або вихідні шари нейронних мереж, які формують імовірнісні оцінки для кожного класу.

Результатом роботи даного етапу є рішення щодо класифікації зразка, а також числова оцінка рівня впевненості, що дозволяє використовувати результати аналізу для подальшої автоматизованої обробки або прийняття рішень у складі комплексних систем захисту інформації.

Етап формування результатів відповідає за представлення вихідної інформації у зручному для користувача або зовнішніх систем форматі. До функцій даного етапу належить виведення результатів класифікації, збереження їх у файлах або базах даних, а також формування звітів про роботу програмного модуля.

Наявність окремого етапу для обробки результатів забезпечує можливість інтеграції програмного модуля в більші інформаційні системи та спрощує аналіз ефективності його роботи під час експериментальних досліджень.

Взаємодія між компонентами програмного модуля реалізується за принципом послідовної обробки даних, що відповідає логіці процесу виявлення шкідливого програмного забезпечення. Така архітектура дозволяє чітко відокремити етапи обробки та забезпечує прозорість

функціонування модуля.

Запропонована архітектура є гнучкою та масштабованою, що дозволяє адаптувати програмний модуль до різних умов експлуатації, змінювати склад гібридної моделі глибокого навчання та розширювати функціональність без суттєвих змін у базовій структурі.

Враховуючи архітектуру програмного модуля, наведену на рисунку 1, алгоритм його роботи реалізується у вигляді послідовної обробки вхідних даних. Такий підхід відповідає логіці задачі виявлення шкідливого програмного забезпечення, дозволяє чітко розмежувати етапи обробки та забезпечує прозорість і відтворюваність результатів.

Процес роботи програмного модуля починається з отримання вхідних даних і завершується формуванням та збереженням результатів класифікації. На кожному етапі алгоритму виконуються чітко визначені операції, які забезпечують підготовку даних, їх аналіз за допомогою гібридної моделі глибокого навчання та інтерпретацію отриманих результатів.

Алгоритм роботи програмного модуля включає такі основні етапи:

- приймання та перевірка вхідних даних;
- попередня обробка виконуваних файлів;
- формування ознакових представлень;
- аналіз за допомогою гібридної моделі глибокого навчання;
- класифікація та прийняття рішення;
- формування, збереження та/або виведення результатів.

Кожен із зазначених етапів є логічно завершеним і може бути реалізований у вигляді окремої функції або програмного компонента, що відповідає модульному принципу побудови програмного забезпечення.

Алгоритм роботи програмного модуля реалізується у вигляді такої детальної послідовності кроків.

1. Початок роботи алгоритму. На початковому етапі виконується ініціалізація програмного модуля, під час якої завантажуються конфігураційні параметри, ініціалізуються необхідні програмні бібліотеки та підготовлюються ресурси для роботи гібридної моделі глибокого навчання. Цей етап забезпечує готовність системи до обробки вхідних даних.
2. Отримання вхідних даних. На вхід алгоритму надходять виконувані файли або їх числові представлення, що підлягають аналізу. Дані можуть передаватися як у вигляді окремих файлів, так і пакетами, що дозволяє реалізувати як одиничний, так і масовий аналіз програмного забезпечення.
3. Перевірка коректності вхідних даних. На даному етапі виконується перевірка формату файлів, їх доступності для читання та відповідності встановленим вимогам. Зокрема перевіряється наявність файлів, їх розмір, коректність структури та відсутність критичних помилок. У разі виявлення некоректних даних формується повідомлення про помилку, після чого алгоритм завершує роботу для поточного зразка, не переходячи до наступних етапів. Такий підхід підвищує надійність програмного модуля та запобігає аварійним завершенням роботи.
4. Попередня обробка даних. Другий етап алгоритму передбачає попередню обробку вхідних даних з метою приведення їх до формату, придатного для аналізу за допомогою гібридної моделі глибокого навчання. На цьому етапі здійснюється зчитування байтового вмісту виконуваних файлів, усунення надлишкової або службової інформації, а також нормалізація даних. Залежно від обраного підходу виконується перетворення байтових послідовностей у двовимірні матриці або зображення фіксованого розміру. Такий спосіб подання даних дозволяє застосовувати згорткові нейронні мережі та зменшує вплив різниці у розмірах виконуваних файлів на результати класифікації.
5. Подання даних на вхід гібридної моделі глибокого навчання. Після завершення передобробки підготовлені дані передаються на вхід гібридної моделі глибокого навчання. Кожна нейронна мережа, що входить до складу гібридної моделі, отримує однакові вхідні дані та виконує їх незалежний аналіз.
6. Формування ознакових представлень. На даному етапі кожна нейронна архітектура формує власний ознаковий вектор, який відображає характерні структурні та статистичні

властивості аналізованого зразка програмного забезпечення. Ознакові представлення формуються на основі глибоких рівнів нейронних мереж і містять узагальнену інформацію про вхідні дані.

7. Об'єднання ознак. Отримані ознакові вектори об'єднуються у спільне узагальнене представлення з використанням механізму агрегації ознак. Такий підхід дозволяє поєднати сильні сторони окремих нейронних архітектур та підвищити інформативність кінцевого ознакового простору.

8. Класифікація зразка. На основі об'єднаного ознакового представлення виконується класифікація аналізованого зразка програмного забезпечення. Алгоритм визначає його належність до класу шкідливого або легітимного програмного забезпечення.

9. Оцінювання рівня впевненості. Паралельно з визначенням класу обчислюється числове значення рівня впевненості прийнятого рішення. Це значення дозволяє оцінити надійність результатів класифікації та може використовуватися для прийняття додаткових рішень у системах захисту інформації.

10. Формування результатів. Результати класифікації подаються у вигляді структурованих даних, що містять інформацію про клас програмного забезпечення та відповідний рівень впевненості моделі.

11. Збереження та/або виведення результатів. Отримані результати зберігаються у файлі або передаються до зовнішніх компонентів системи для подальшого використання, зокрема статистичного аналізу або інтеграції з іншими модулями безпеки.

12. Завершення роботи алгоритму.

Таким чином, розроблений алгоритм роботи програмного модуля забезпечує послідовну, логічно завершену та стійку до помилок обробку даних – від моменту надходження вхідних зразків до формування результатів класифікації. Запропонований алгоритм створює надійну основу для практичної реалізації програмного модуля та подальшого експериментального дослідження його ефективності.

**Висновки.** У результаті проведеного дослідження розроблено архітектуру програмного модуля виявлення шкідливого програмного забезпечення, яка включає етапи завантаження даних, попередньої обробки, аналізу за допомогою гібридної моделі глибокого навчання, класифікації та формування результатів. Визначено функціональне призначення кожного компонента та забезпечено їх узгоджену взаємодію в межах єдиної системи .

Сформовано алгоритм роботи програмного модуля у вигляді послідовної обробки даних, що охоплює всі етапи – від приймання вхідних зразків до отримання результатів класифікації. Запропонований алгоритм забезпечує логічну цілісність, прозорість функціонування та можливість практичної реалізації програмного модуля.

Встановлено, що використання гібридної моделі глибокого навчання дозволяє ефективно поєднувати ознакові представлення різних нейронних архітектур, що підвищує точність виявлення шкідливого програмного забезпечення та стійкість до модифікацій програмного коду.

**Список літератури**

1. Nataraj L., Karthikeyan S., Jacob G., Manjunath B. Malware images: Visualization and automatic classification. Proceedings of the International Symposium on Visualization for Cyber Security. 2011. P. 1–7.

2. Saxe J., Berlin K. Deep neural network based malware detection using two dimensional binary program features. Proceedings of the International Conference on Malicious and Unwanted Software. 2015. P. 11–20.

3. Raff E., Barker J., Sylvester J., Brandon R., Catanzaro B., Nicholas C. Malware detection by eating a whole executable. Proceedings of the AAAI Conference on Artificial Intelligence. 2018. Vol. 32, no. 1. P. 268–276.

4. Ucci D., Aniello L., Baldoni R. Survey of machine learning techniques for malware analysis. Computers & Security. 2019. Vol. 81. P. 123–147.