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

Бондарь Ілля Вікторович

Метод цифрової стеганографії мультимедіа-об'єктів / Method of Digital Steganography for Multimedia Objects

спеціальність: 122 - Комп'ютерні науки
освітньо-професійна програма - Комп'ютерні науки

Кваліфікаційна робота

Виконав студент групи КНм-21
І.В. Бондарь
_____

Науковий керівник:
к.т.н., доцент І.В. Турченко
_____

Кваліфікаційну роботу
допущено до захисту:
«___» _____ 20___ р.
Завідувач кафедри
_____ М.П. Комар

ТЕРНОПІЛЬ - 2023

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

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ М.П. Комар
«____» _____20__р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Бондарь Ілля Вікторович
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи
Метод цифрової стеганографії мультимедіа-об'єктів / Method of Digital Steganography for Multimedia Objects
керівник роботи к.т.н., доцент І.В. Турченко
затверджені наказом по університету від 8 грудня 2022 року № 491.

2. Строк подання студентом закінченої кваліфікаційної роботи 1 грудня 2023 р.

3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.

4. Основні питання, які потрібно розробити:
- огляд предметної області;
- аналіз існуючих методів та алгоритмів цифрової стеганографії;
- дослідження основних вимог до стеганографічних систем;
- аналіз літературних джерел досліджуваної тематики;
- постановка задачі дослідження;
- розробка алгоритму вбудовування прихованих даних у медіафайли;
- дослідження можливостей застосування нейромереж;
- програмна реалізація стеганографічного методу;
- дослідження розробленого методу та порівняльний аналіз з відомими рішеннями.

5. Перелік графічного матеріалу у роботі:
- схема алгоритму модуля аудіостеганографії;
- схема алгоритму адаптації методу F5 для відео;
- схема алгоритму роботи програми без графічного інтерфейсу.

6. Консультанти розділів кваліфікаційної роботи

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

7. Дата видачі завдання 8 грудня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Теоретичні основи цифрової стеганографії та аналіз існуючих методів	12.2022 р. – 03.2023 р.	
2	Метод цифрової стеганографії для мультимедіа-об’єктів	03.2023 р. – 05.2023 р.	
3	Програмна реалізація та дослідження методу	05.2023 р. – 11.2023 р.	
4	Повне завершення та представлення кваліфікаційної роботи на кафедру	01.12.2023 р.	

Студент _____ І.В. Бондарь
підпис

Керівник роботи _____ к.т.н., доцент І.В. Турченко
підпис

РЕЗЮМЕ

Кваліфікаційна робота на тему «Метод цифрової стеганографії мультимедіа-об'єктів» на здобуття освітнього ступеня «Магістр» зі спеціальності 122 «Комп’ютерні науки» освітньої програми «Комп’ютерні науки» написана обсягом в 112 сторінок і містить 23 ілюстрації, 1 таблицю, 6 додатків та 44 використаних джерела.

Метою даної кваліфікаційної роботи є розроблення та практична реалізація методу цифрової стеганографії для приховування конфіденційних даних у мультимедіа файлах на основі штучних нейронних мереж.

Методи досліджень: системний аналіз, стегоаналіз, теорія штучних нейронних мереж.

Результати дослідження: запропоновано метод цифрової стеганографії для мультимедіа-об’єктів, який, на відміну від інших, дозволяє досягти адаптивного вбудовування прихованої інформації з урахуванням особливостей контейнера через застосування технологій штучних нейронних мереж, що дозволяє покращити результати та підвищити ефективність застосування стеганографії.

Результати роботи можуть бути корисні для покращення захисту конфіденційної інформації шляхом її приховування у мультимедіа файлах за допомогою цифрової стеганографії.

Ключові слова: СТЕГАНОГРАФІЯ, СТЕГОАНАЛІЗ, ШТУЧНИЙ ІНТЕЛЕКТ, НЕЙРОННА МЕРЕЖА, МУЛЬТИМЕДІА-ОБ’ЄКТИ.

ABSTRACT

Qualification work on the topic «Method of Digital Steganography for Multimedia Objects» for Master's degree on speciality 122 «Computer Science» educational and professional program «Computer Science» is written on 112 pages and it contains 23 figures, 10 formulas, 1 table, 6 annexes and 44 sources.

The purpose of this qualification work is to develop and implement a method of digital steganography method for hiding confidential data in multimedia files based on neural network models.

Research methods: system analysis, stegoanalysis, theory of artificial neural networks, comparative analysis, experimental method.

Research results: proposed a method of digital steganography for multimedia objects, which, unlike others, allows for adaptive embedding of hidden information, taking into account the container's characteristics through the application of artificial neural network technologies. This approach enhances results and improves the efficiency of steganography application.

The findings of this work can be valuable for enhancing the protection of confidential data by concealing it within multimedia files using digital steganography.

Keywords: STEGANOGRAPHY, STEGOANALYSIS, ARTIFICIAL INTELLIGENCE, NEURAL NETWORK, MULTIMEDIA OBJECTS.

ЗМІСТ

СПИСОК УМОВНИХ СКОРОЧЕНЬ

LSB – Least Significant Bit (найменш значущий біт)

DCT – Discrete Cosine Transform (дискретне косинусне перетворення)

FFT – Fast Fourier Transform (швидке перетворення Фур'є)

PSNR – Peak Signal-to-Noise Ratio (пікове співвідношення сигнал/шум)

MSE – Mean Squared Error (середньоквадратична помилка)

SSIM – Structural Similarity Index (індекс структурної подібності)

ШПФ – Швидке перетворення Фур’є

PCM – Pulse-Code Modulation (імпульсно-кодова модуляція)

ВСТУП

Актуальність теми. В умовах стрімкого розвитку інформаційних технологій та популяризацією використання мультимедіа контенту, гостро постає питання його захисту. Адже поширення цифрових даних в мережі Інтернет породжує загрози несанкціонованого доступу та модифікації важливої інформації. У зв'язку зі стрімким розвитком інформаційних технологій, особливо гостро постало питання захисту даних на тлі повномасштабного вторгнення Росії в Україну в 2022 році. Адже в умовах війни різко зростають кіберзагрози, пов’язані зі шпигунством, дезінформацією, зламами баз даних. Це актуалізує проблематику забезпечення конфіденційності та цілісності інформації, особливо в мережевих комунікаціях.

Одним з ефективних методів вирішення цієї проблеми є стеганографія – наука про приховування даних всередині цифрових носіїв. На відміну від широко відомої криптографії, стеганографія дозволяє не просто закодувати, а й приховати сам факт існування повідомлення. Це досягається шляхом вбудовування додаткової інформації у мультимедіа контейнер – зображення, аудіо чи відео.

За допомогою стеганографії інформація може бути прихована від небажаних або недозволених користувачів, що дозволяє передавати та зберігати дані без підозрілих поглядів. Основним принципом стеганографії є те, що приховані дані є непомітними або незрозумілими для незаконного спостерігача, який не знає, що дані приховані.

Стеганографія має велике значення в багатьох сферах, включаючи військову розвідку, криміналістику, корпоративну безпеку та особисту приватність. Вона може бути використана для передачі та зберігання секретної інформації, яка має важливе значення для державної безпеки, комерційної конфіденційності або особистої безпеки. Використання стеганографії також дозволяє уникнути детектування та блокування системами безпеки, оскільки приховані дані не виглядають підозрілою або зловмисною діяльністю. Це робить стеганографію

потужним інструментом для забезпечення конфіденційності та безпеки інформації. Тому тема кваліфікаційної роботи є актуальною.

Мета і завдання дослідження. Метою є розроблення та практична реалізація методу цифрової стеганографії для приховування конфіденційних даних у мультимедіа файлах на основі штучних нейронних мереж.

Зазначена мета передбачає вирішення таких завдань:

- огляд предметної області;
- аналіз літературних джерел досліджуваної тематики;
- аналіз існуючих методів та алгоритмів цифрової стеганографії;
- дослідження основних вимог до стеганографічних систем;
- постановка задачі дослідження;
- розробка алгоритму вбудовування прихованих даних у зображення/аудіо/відео;
- дослідження можливостей застосування нейромереж;
- програмна реалізація стеганографічного методу;
- дослідження розробленого методу та порівняльний аналіз з відомими рішеннями.

Об’єктом дослідження є процеси зберігання та передачі інформації в комп’ютерних системах.

Предметом дослідження є стеганографія мультимедіа-об’єктів в процесах зберігання та передачі інформації з метою її приховування.

Методи досліджень: системний аналіз, стегоаналіз, теорія штучних нейронних мереж.

Науково-теоретичною основою дослідження стали праці: Андерсона Р. Дж, Лям. Б.Ф., Барга А., Шенона К., Коха Т., Зао Цз., Мазурчука А., Сенка Д., Чхан Ч., Ші Е., Чан-Чу Ліна, Вен-Шаь Цая, Абдулаха Басахеля, Мумамеда Яміна, Адман Ахмеда абі Сена.

Наукова новизна одержаних результатів: запропоновано метод цифрової стеганографії для мультимедіа-об’єктів, який, на відміну від інших, дозволяє

досягти адаптивного вбудовування прихованої інформації з урахуванням особливостей контейнера через застосування технологій штучних нейронних мереж, що дозволить покращити результати та підвищити ефективність застосування стеганографії.

Практичне значення отриманих результатів: результати можуть бути використані як широким колом користувачів для приховування особистих даних з метою їх зберігання чи передачі, так і фахівцями у галузі захисту інформації чи розробниками захищених систем зв'язку. Отримані результати тестування системи за критеріями ємності, стійкості, непомітності можуть бути використані для подальшого вдосконалення стеганографічних методів.

Результати дослідження можуть бути корисні для покращення захисту конфіденційної інформації шляхом її приховування у мультимедіа файлах за допомогою цифрової стеганографії.

Публікації та апробація кваліфікаційної роботи. Результати дослідження опубліковано та апробовано в матеріалах міжнародної науково-практичної інтернет-конференції "Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення", 07-08 грудня 2023р., м. Тернопіль, та в збірнику матеріалів науково-практичного симпозіуму "Захист інформації", 30 листопада 2023 р., м. Тернопіль, (додаток Д).

.

1 ТЕОРЕТИЧНІ ОСНОВИ ЦИФРОВОЇ СТЕГАНОГРАФІЇ ТА АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ

1.1 Визначення та основні поняття цифрової стеганографії

Цифрова стеганографія є актуальним напрямом сучасних інформаційних технологій, що охоплює методи приховування даних з метою забезпечення конфіденційності їх передавання в кіберпросторі. В науковому середовищі склалося декілька підходів до тлумачення базових понять цієї галузі.

У фундаментальній монографії Вадима Гребеннікова "Стеганографія та цифровий захист інформації" [1] запропоновано визначати стеганографію як "науку про приховування самого факту існування повідомлення в цифрових даних".

Ця дефініція концентрує увагу на ключовій відмінності стеганографії від тісно пов’язаної з нею науки криптографії. Якщо криптографія опікується лише шифруванням змісту повідомлень, то стеганографія ставить за мету замаскувати сам факт наявності якоїсь прихованої інформації в контейнері-носії. Тобто, стеганографічні методи "ховають" дані таким чином, що сторонній спостерігач навіть не може ідентифікувати наявність якогось таємного повідомлення. Це досягається за рахунок його "розчинення" в мультимедійному контенті – зображенні, аудіо чи відео.

Іншу точку зору висвітлено в роботі Пузиренко О.Ю., Прогонов Д.О., Конахович Г.Ф. "Комп’ютерна стеганографічна обробка" [2]. На відміну від попереднього тлумачення, автори фокусуються саме на технічному боці питання – процесах вбудовування прихованої інформації в мультимедійні дані. Зокрема, вони трактують стеганографію як "сукупність методів цифрової обробки контейнерів з метою приховання даних для їх збереження чи передачі в неявному, замаскованому вигляді". Тобто акцент робиться на застосуванні технологій цифрової обробки сигналів і зображень (стиснення, фільтрації, перетворення тощо) задля

вбудовування інформації в мультимедійні файли таким чином, аби її присутність була непомітною для стороннього спостерігача.

Ще один погляд пропонує Rainer Böhme у своїй монографії "Advanced Statistical Steganalysis" [3]. Автор означує стеганографію як "мистецтво приховування самого факту передачі повідомлень". Таким чином, даний підхід концентрує увагу виключно на аспекті забезпечення анонімної комунікації між користувачами, щоб навіть сам факт обміну інформацією залишався непоміченим сторонніми особами.

Найбільш комплексним вважається визначення, запропоноване в роботі Пузиренко О.Ю., Прогонов Д.О., Конахович Г.Ф. [2]. Дане трактування стеганографії є широким і дозволяє опиратись на себе в ході теоретичного чи практичного дослідження теми, окреслює коло задач, які має вирішувати стеганографія як галузь науки.

В наукових колах дискусійним залишається питання щодо визначення базових структурних елементів та понять стеганографії. У праці В.О. Хорошко, Ю.Є. Яремчук, В.В. Карпінець "Комп’ютерна стеганографія" [4] стверджується, що "ключовими складниками стеганографічної системи є: приховане повідомлення, контейнер-носій та ключ (пароль)". Натомість, інші дослідники, зокрема Дельмоліно К. [5] та Кер А. [6], переконані, що "поряд із зазначеними атрибутами не менш важливе значення має стеганографічний канал передачі даних та його пропускна здатність".

Група науковців на чолі з Лі В., Чанг Ч. [7] пропонує також включити до переліку базових категорій стеганографії явища стеганоаналізу та робастності методів. Перше пов’язане з розробкою технологій виявлення прихованих повідомлень, а друге – зі стійкістю закладених даних до навмисних або випадкових спотворень контейнера, в який вони вбудовані. У [8] стверджується, що "не менш важливі поняття – це ємність контейнеру, а також непомітність чи детектабельність прихованих даних".

Тобто, основними аспектами залишаються приховане повідомлення, контейнер носій, та ключ:

- приховане повідомлення (embedded message) – це власне дані, які потрібно передати прихованим чином. Це може бути текст, зображення, аудіо, відео або будь-які інші дані, що містять конфіденційну інформацію. Приховане повідомлення вбудовується у контейнер за допомогою спеціальних алгоритмів так, щоб його присутність була неможливо ідентифікувати;

- контейнер (host signal) – це носій прихованого повідомлення. Ним найчастіше виступає якийсь мультимедійний файл: зображення, аудіо чи відео. Саме в такий контейнер за допомогою методів цифрової обробки вбудовують потрібні дані. Вимоги до контейнера – містити "шуми" чи надлишкову інформацію для ефективного приховування;

- ключ (stego key) – це пароль або криптографічний ключ, яким додатково шифрується приховане повідомлення перед його вбудовуванням в контейнер. Це посилює криптостійкість та захищеність всієї стегосистеми.

Також доцільно розглянути елементи, на які звернена увага у роботах [5-6], а саме стеганографічний канал передачі даних та його пропускна здатність:

- стеганографічний канал (subliminal channel) – це прихований канал передачі даних, який утворюється в результаті вбудовування одного повідомлення в інше за допомогою методів цифрової стеганографії. Фактично – це шлях руху прихованої інформації від відправника до одержувача. Такий канал існує паралельно із звичайним каналом зв'язку, який передає відкрите повідомлення (контейнер). Завдяки стеганографічному каналу досягається анонімний обмін закритими даними без привернення сторонньої уваги;

- пропускна здатність стегоканалу (capacity) – це максимальний обсяг прихованої інформації в бітах, яку можна вбудувати в контейнер за допомогою обраного стеганографічного методу без втрати якості носія та помітних ознак наявності прихованих даних. Чим більша пропускна здатність, тим ефективніша стегосистема.

З оглядом на роботу [7], варто оглянути суть стегоаналізу і робастності методів. Стеганоаналіз (steganalysis) являє собою сукупність методів та алгоритмів, спрямованих на виявлення наявності прихованих повідомлень у цифрових контейнерах: зображенні, аудіо чи відео. Тобто стеганоаналіз – це зворотній бік стеганографії, технології розкриття таємних даних, вбудованих у носії за допомогою стегосистем. Стеганоаналіз спрямований на компрометацію методів цифрового приховування даних та викриття секретних каналів передачі інформації. Тобто це сукупність інструментів та підходів нейтралізації стеганографічного захисту певної інформації.

Згідно з класифікацією доктора наук Кошкіної Наталії Василівни, основними методами стеганоаналізу є [9]:

- візуальний аналіз полягає у візуальній перевірці цифрових контейнерів, – графічних зображень, спектрограм звукових файлів тощо, – з метою ідентифікації ознак наявності прихованих повідомлень. Такий аналіз базується на тому, що вбудовування сторонньої інформації, особливо при недостатньому рівні якості стеганографічного методу, може призвести до появи візуально помітних спотворень носія. Наприклад, при цифровому маркуванні зображень можуть з’явитися розпливчатості, артефакти стиснення, дефекти кольору тощо. Аудіосигнал із вбудованим повідомленням може мати спотворення амплітуди на спектрограмах [9-10];

- статистичний аналіз полягає у використанні статистичних властивостей сигналів (звукових, відео тощо) та зображень з метою виявлення в цих контейнерах ознак наявності прихованих стеганографічних даних. Статистичний стеганоаналіз базується на оцінці й аналізі статистичних характеристик сигналів і зображень-контейнерів з використанням математичного апарату теорії ймовірностей, математичної статистики та обчислювального інтелекту. Сутність методу полягає в тому, що будь-яке вбудовування прихованих даних впливає на статистичні параметри носія – розподіл яскравості пікселів, гістограми кольорів, кореляційні

залежності тощо. Аналізуючи ці параметри, можна ідентифікувати "статистичні сліди" прихованої стеганографічної інформації [10];

- аналіз структури файлів-контейнерів полягає у ретельній перевірці їх внутрішньої побудови з метою виявлення ознак наявності прихованих сторонніх даних. Як зазначають фахівці Дженг-Шан Пян, Хунсан Чен Пюан "структурний аналіз контейнерів базується на глибинному вивченні технічних аспектів форматів даних – стиснення, кодування, моделі репрезентації інформації тощо" [11]. Сутність методу полягає в тому, що будь-яке вбудовування прихованого контенту потребує зміни внутрішньої структури носія, що може бути виявлено шляхом ретельної перевірки окремих байтів контейнеру. Зокрема, аналізуються заголовки, таблиці кольорів, методи стиснення файлів тощо.

Стосовно робастності методів, робастність (robustness) стеганографічних методів – це їх стійкість до навмисних чи випадкових перетворень або спотворень контейнера-носія, в який вбудована прихована інформація. Іншими словами, – це здатність забезпечувати цілісність та можливість подальшого вилучення секретних даних навіть за умов впливу на мультимедіа-файл різноманітних шумів, фільтрацій, стиснень тощо [12].

Згідно опису фахівців Нейла Джонсона і Зорана Дюріка. у фундаментальній праці "Цифрова стеганографія" [13], чим вищий ступінь робастності певного методу вбудовування даних в контейнери, – тим стійкішою є стегосистема в цілому. Адже прихована інформація залишається доступною навіть за агресивного впливу на носій. Для підвищення рівня робастності використовуються спеціальні перетворення та алгоритми посилення стійкості прихованих даних до можливих спотворень контейнерів [14-15]. Це критично важливо для надійності та живучості стегосистем при реальних умовах застосування.

Також існують різні класифікації видів стеганографії, зокрема за типом контейнера: текстова, зображень, аудіо чи відео.

Аудіо стеганографічні методи використовуються для приховування конфіденційної інформації в аудіофайлах. Це розділ стеганографії, який дозволяє

вбудовувати приховані повідомлення в звукові дані, такі як музика, мовлення, звукові ефекти тощо.

Одним з прикладів таких методів є Least Significant Bit (LSB) – як зазначають Бендер і Груль у своїй роботі [15], суть цього підходу полягає в тому, що "найменш значущі біти оцифрованого звукового сигналу послідовно змінюються на біти повідомлення, що приховується".

Згідно з аналізом фахівця Жуана Ж. [16], метод LSB відрізняється простотою та швидкістю реалізації. Адже потрібна лише пряма побітова заміна в молодших бітах аудіо. Водночас цей метод може бути вразливим до статистичних атак, описаних в роботі Ванг Г. Ж. [17], при яких шляхом порівняння "чистого" та модифікованого сигналів виявляють ознаки прихованої інформації.

LSB вважається простим методом, оскільки він вимагає лише заміни найменш значущих бітів аудіосигналу. Це робить його легким для реалізації та швидким для виконання. Також, оскільки LSB використовує найменш значущі біти аудіосигналу, зміни, внесені в аудіосигнал, зазвичай непомітні людського слуху. Це означає, що якість аудіосигналу зберігається, а приховане повідомлення важко виявити. Однак, LSB може бути вразливим до статистичного аналізу та атак на стеганографію. Зокрема, якщо атакуючий знає алгоритм LSB та має доступ до оригінального аудіосигналу, він може порівняти оригінал із стеганографічним аудіосигналом та виявити приховане повідомлення.

Поряд з описаним вище підходом LSB, іншою поширеною технологією цифрового приховування даних в аудіо є метод на основі дискретного косинусного перетворення (Discrete Cosine Transform, скорочено DCT). Як зазначають автори Прадхан Дж. і Наяк Дж. у своїй праці [18], його суть полягає в тому, що "аудіосигнал спочатку перетворюється в частотну область за допомогою DCT, після чого прихована інформація вбудовується в отримані коефіцієнти перетворення".

На думку експерта Сепаннена Т. [19], такий підхід дозволяє досягти більшої стійкості стеганографічної системи, завдяки розподілу прихованих даних по різних

частотних складових аудіосигналу. Крім того, вбудовування в коефіцієнти DCT практично не погіршує якість оригінального аудіо [20].

Ще одним поширеним підходом приховування інформації в медіа контейнерах є використання алгоритму швидкого перетворення Фур'є (Fast Fourier Transform, FFT). Відповідно до аналізу в роботі Дхея Мехти, подібно до розглянутого вище DCT, FFT дозволяє перевести аудіосигнал в частотну область, але з використанням комплексних коефіцієнтів перетворення. Це відкриває додаткові можливості щодо "розподілу та розсіювання цифрового водяного знаку між різними частотними компонентами" [21].

Експерт Гопалан К. стверджує, що FFT стеганографія відрізняється високим рівнем криптостійкості, адже пошук та вилучення даних у комплексному частотному спектрі є дуже складним завданням [22]. Водночас вбудовування інформації в коефіцієнти FFT практично не спотворює якість оригінального аудіо.

Відео стеганографічні в свою чергу методи використовуються для приховування конфіденційної інформації в цифрових відеозаписах. Це ще одна галузь стеганографії, яка дозволяє вбудовувати приховані повідомлення в кадри відео без помітних змін у візуальному вигляді. Відео стеганографія має широкий спектр застосувань, включаючи захист конфіденційної інформації, заходи проти піратства, цифрове підписування та авторські права, а також створення прихованих копій інформації для забезпечення її надійності та недоступності для сторонніх користувачів.

Зокрема, поширеним є метод заміни найменш значущих бітів (LSB) окремих кадрів відео, про який йшлося вище. Відповідно до огляду в роботі Ванга Р. З. та Ліна С. Ф. [23], суть полягає в тому, що "найменш інформативні біти кожного пікселя кадрів послідовно замінюються на біти прихованого контенту".

Іншим ефективним підходом є технологія вбудовування даних в коефіцієнти дискретного косинусного перетворення (DCT) кадрів відео. Цікавим підходом приховування даних у відеоконтенті є метод, що базується на використанні векторів руху між сусідніми кадрами. Згідно з аналізом у дослідженні, його суть

полягає в тому, що "прихована інформація вбудовується шляхом модифікації напрямку та/або величини векторів руху об’єктів між кадрами" [24].

Більш базовим є метод приховування у зображенні. Це галузь стеганографії, яка дозволяє вбудовувати приховані повідомлення в пікселі зображення без впливу на його зовнішній вигляд. Стеганографічні методи цифрових зображень використовуються для приховування в них конфіденційної інформації. Це галузь стеганографії, яка дозволяє вбудовувати приховані повідомлення в пікселі зображення без впливу на його зовнішній вигляд.

LSB в пікселях зображення – цей метод аналогічний LSB для аудіо та відео, але застосовується до пікселів зображення. Найменш значущі біти кожного пікселя замінюються на біти прихованого повідомлення. Його характеристика також описана вище. Дискретне косинусне перетворення (DCT) є широко використовуваним методом для обробки та стиснення зображень. Він перетворює зображення з просторової області в частотну область, що дозволяє аналізувати та обробляти зображення на різних частотах.

Проте одними з найцікавіших методів є методи на основі вейвлетів, що використовують вейвлет-перетворення для розбиття зображення на різні частотні компоненти. Вейвлет-перетворення (wavelet transform) є потужним математичним інструментом, який використовується для аналізу сигналів та обробки даних. Він дозволяє представити сигнал у вигляді суперпозиції вейвлет-функцій різної частоти та масштабу. Основна ідея вейвлет-перетворення полягає в тому, щоб аналізувати сигнал не лише на основі фіксованої частоти, як у традиційних методах, але й на основі масштабу. Вейвлет-функції, які використовуються в перетворенні, мають обмежену часову тривалість і плавно змінюються від низької до високої частоти. Це дозволяє виявити як малочастотні, так і високочастотні компоненти сигналу з високою точністю.

Однією з важливих переваг вейвлет-перетворення є його здатність до локалізації сигнальних змін в часі та частоті. Завдяки цьому, вейвлет-перетворення

знаходить широке застосування в областях, таких як сигнальний аналіз, обробка зображень, стиск даних, виявлення шуму та артефактів [25].

Останнім залишається текстова стеганографія – найпримітивніший та найстаріший метод. Текстова стеганографія полягає у приховуванні інформації в текстових повідомленнях: електронних документах, веб-сторінках, програмному коді тощо [26]. Це найдавніший метод цифрового приховування даних, що базується на вбудовуванні прихованих відомостей у текст-контейнер без порушення його читабельності. До основних технік текстової стеганографії відносяться: "використання синонімів, додавання невидимих символів, заміна розміру шрифту тощо" [27]. Незважаючи на простоту реалізації, текстова стеганографія широко застосовується на практиці завдяки легкості створення текстових контейнерів.

1.2 Методи виявлення стеганографії

Так само як важко приховати стеганографію, також і важко її виявити. Цей процес називають стегоаналізом і він є основним компонентом стеганографії. В цьому розділі детальніше розглянуто виявлення прихованих стеганографічних повідомлень, адже вони є важливим завданням із забезпечення інформаційної безпеки.

Одним з основних методів виявлення прихованих даних є статистичний стеганаліз. Він базується на аналізі статистичних характеристик контейнерів – зображень, аудіо чи відео [28]. Зокрема аналізуються гістограми, кореляції, ентропія та інші статистичні параметри файлів. Статистичний метод стегоаналізу призначений для використання разом із формальними методами приховування інформації, залишаючи певну позначку (підпис) під час процесу вбудовування, за допомогою якої можна виявити приховані вбудовування. Клас статистичних методів стегоаналізу представляє введення прихованої інформації в контейнер як порушення статистичного зв’язку вихідного контейнера та дає імовірнісне

представлення. Проводиться аналіз статистичних характеристик певної послідовності бітів, щоб визначити, чи корелює вона з характеристиками порожніх стеганографічних контейнерів того ж типу. Цей підхід використовує багато статистичних характеристик, таких як коефіцієнти кореляції, оцінки ентропії, ймовірності появи та залежності між елементами послідовності, дискримінаційні розподіли за допомогою критеріїв хі-квадрат тощо. До переваг цієї групи методів стегоаналізу можна віднести необмежену область застосування. Основним недоліком таких методів є те, що неявно передбачається існування деякого примітивного контейнера. Однак результати методів, заснованих на критерії хі-квадрат, сильно залежать від методу приховування даних. Цей метод дає хороші результати, коли молодші біти (LSB) елементів контейнера записуються послідовно, але не працює, коли молодші біти вибираються псевдовипадково, а повідомлення розподіляється по всій довжині контейнера. Крім того, окремі області зображення можна вибрати для подальшого аналізу. Цей метод з певною ймовірністю дозволяє виявити наявність інформації, прихованої псевдо випадковим чином.

Для перевірки можливості виявлення стеганографії у зображені даний метод опробовано та візуалізовано на файлах зображень зі стеганографією. На рисунку 1.1 зображено результат роботи алгоритму при заповненні контейнера на 9%. По осі X – відсоток перевірених коефіцієнтів, Y – ймовірність вбудовування. Суцільна лінія – звичайний Хі-квадрат, пунктир – блоковий Хі-квадрат. На рисунку 1.2 зображено результат роботи алгоритму при заповненні контейнера на 3%.

Більшість стеганографічних алгоритмів приховують дані в молодші біти коефіцієнтів, відмінних від 0 та 1. Як наслідок, частоти 0-х і 1-х коефіцієнтів не змінюються, у той час як всі інші частоти або зменшуються, або збільшуються, – залежно від алгоритму вбудовування. Проте окрім сигнатурного методу, існують ще й інші, такі як візуальний аналіз, що полягає в огляді зображень на предмет видимих ознак стеганографії, методи машинного навчання, частотного аналізу і

методи аналізу вразливостей.

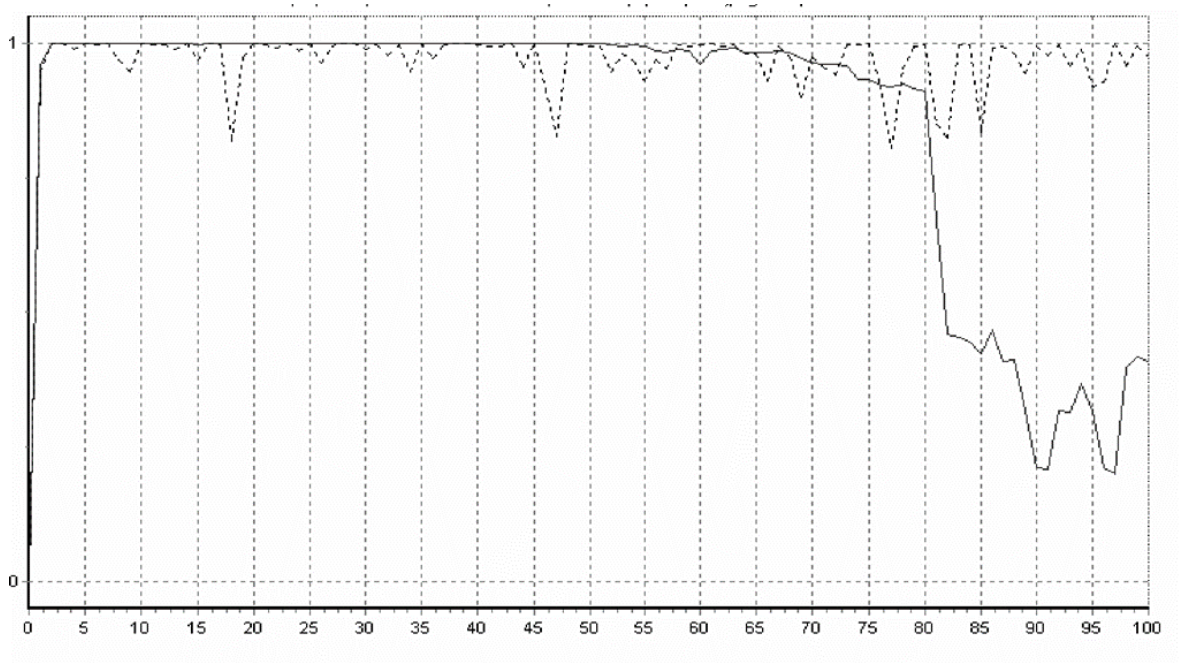


Рисунок 1.1 – Графік ймовірності наявності приховування інформації у зображенні при заповненні на 9%

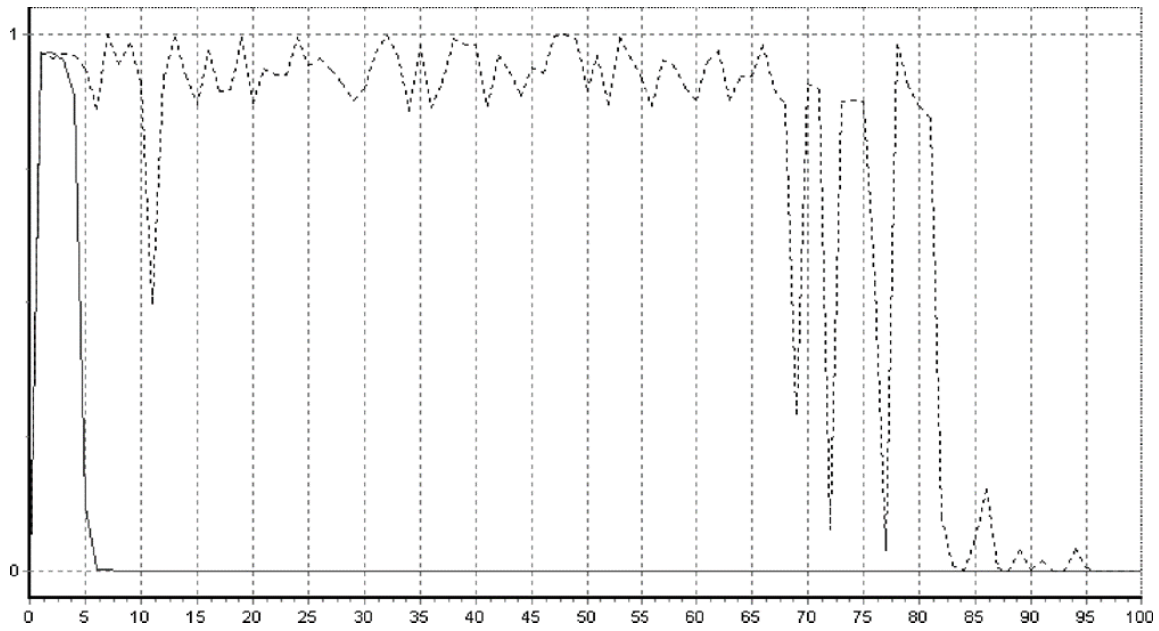
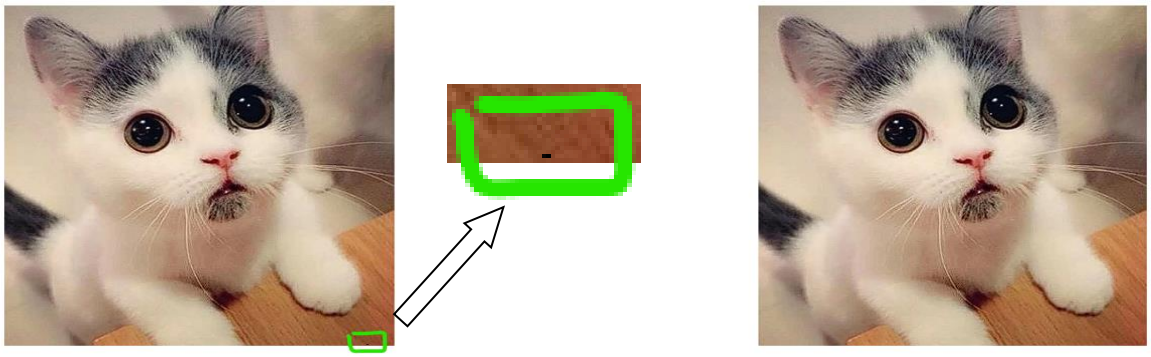


Рисунок 1.2 – Графік ймовірності наявності приховування інформації у зображенні при заповненні на 3%

Візуальний аналіз є одним з базових підходів виявлення прихованих даних, який базується на зоровому сприйнятті людини. Суть методу полягає в тому, що

експерт візуально перевіряє цифрові контейнери, – зображення, сонограми аудіо, відео тощо, – на предмет наявності підозрілих шумів, спотворень, дефектів, які можуть вказувати на стеганографічне маркування [29]. Наприклад, стиснення зображень з вбудованими даними може призводити до появи специфічних артефактів та втрати чіткості. Аудіосигнали з прихованою інформацією при перетворенні у сонограми також можуть мати характерні спотворення обвідної [30].

Для прикладу спеціально оброблено власні зображення та аудіо для показу візуального аналізу такої стеганографії. На рисунку 1.3 зображено візуальний аналіз зображення з вмістом стеганографії і без, на рисунку 1.4 зображено візуальний аналіз для аудіо. Приклад для відео підходить аналогічним чином до прикладу для зображень. Також на рисунку 1.3 показано і виділено зеленим прямокутником елемент, який підпадає під стеганографію, саме при приближені такого зображення можна помітити биті пікселі, які заміняються на чорні квадрати.



Із стеганографією Без стеганографії

Рисунок 1.3 – Візуальний аналіз на прикладі зображень

Візуальний стегоаналіз аудіо даних базується на аналізі графічного представлення аудіосигналу у вигляді сонограми чи спектрограми. Це дво- або тривимірне зображення, яке відображає розподіл частотних та амплітудних характеристик аудіо в часі [43].

Рисунок 1.4 – Візуальний аналіз стеганографії аудіо

При вбудовуванні прихованої інформації в аудіо за допомогою цифрової стеганографії можуть виникати специфічні шуми та спотворення на сонограмі – додаткові гармоніки, зміна обвідної сигналу, провали та викиди окремих частотних складових тощо [30]. Експерт візуально аналізує сонограму оригінального та модифікованого аудіо, порівнює їх та ідентифікує підозрілі відмінності, які можуть вказувати на наявність вбудованого стеганографічного контенту.

Аналіз вразливостей (vulnerability analysis) є одним з підходів виявлення прихованих даних, який базується на пошуку слабких місць та недоліків у реалізації стеганографічних методів [31]. Суть полягає у комплексному дослідженні алгоритмів цифрового маркування з метою ідентифікації можливостей для зламу системи приховування даних. Аналізуються математичні перетворення, криптографічні примітиви, особливості вбудовування інформації в контейнери

тощо [32].

Знайдені вразливості дають змогу розробити спеціальні методи для вилучення або декодування прихованих відомостей без знання стегоключа чи алгоритму.

Відносно новим методом аналізу та виявлення є метод на основі штучного інтелекту. Суть полягає у побудові та навчанні штучних нейронних мереж (ШНМ) для розпізнавання цифрових вбудовувань [33]. На етапі навчання ШНМ подаються приклади оригінальних та модифікованих стеганографією зображень, аудіо чи відео. Мережа аналізує навчальні дані та вибудовує внутрішню модель для подальшої класифікації файлів на позначені та непозначені. Наприклад, розглянемо існуючі моделі AlexNet, VGG-16 і ResNet50.

AlexNet – це одна з піонерських архітектур глибинних згорткових нейромереж, запропонована Алексом Кріжевським та співавторами у 2012 році [3]. Вона складається з 8 шарів: 5 згорткових та 3 повнозв'язаних. На вхід AlexNet подаються зображення розміром 227x227 пікселів. У статті Цянь Юе та співавторів досліджено використання цієї архітектури ШНМ для стегоаналізу. "AlexNet демонструє точність 83% та чутливість 79% в класифікації зображень на стеганографічно марковані та немодифіковані. За рахунок глибинних згорткових шарів мережа навчається виокремлювати дискримінативні ознаки, які вказують на наявність цифрового вбудовування даних" [34].

VGG-16 – це глибинна згорткова нейронна мережа, запропонована вченими з Visual Geometry Group (VGG) Оксфордського університету у 2014 році [35]. Вона містить 16 шарів з параметрами (звідси назва VGG-16): 13 згорткових та 3 повнозв'язаних. Як зазначають Жан Пейсон та Бенджамін Блейх [36], ця архітектура продемонструвала точність 91% та чутливість 87% у задачах класифікації зображень для стегоаналізу. Завдяки великій глибині та "полю зору" згорткових фільтрів, VGG-16 ефективно навчається розпізнавати локальні та глобальні ознаки наявності цифрових вбудовувань.

І останнім є ResNet50. ResNet-50 – це глибинна нейронна мережа на основі

залишкових з'єднань (ResNet – Residual Network), розроблена вченими з Microsoft Research Азія у 2015 році [37]. Вона складається з 50 шарів, включаючи згорткові, залишкові блоки та повнозв'язані шари наприкінці. Згідно з дослідженням Ліна Юе та співавторів [38], для задач стегоаналізу цифрових зображень мережа ResNet-50 продемонструвала найвищі результати серед розглянутих архітектур глибинного навчання. Зокрема, точність склала 94%, а чутливість – 90% на тестовому наборі даних BOSS.

Ефективність ResNet-50 пояснюється оптимізованим потоком градієнтів та посиленою здатністю до узагальнення ознак стеганографічного маркування зображень завдяки "залишковим" зв'язкам між шарами мережі.

Ці алгоритми використовують навчальні файли, які задаються уже з готовими рішеннями, для визначення стеганографії у різноманітних файлах, таких як аудіо, текст, відео чи зображення. Кожен алгоритм записує вище описані методи з розділу 1.1, запам’ятовує їх розв'язки і орієнтується на матеріали, які вже були оброблені людьми. Зокрема, для навчання моделей стегоаналізу на базі глибинних нейромереж в якості вхідних даних використовують аудіо, тексти, зображення та відео з відомим "маркуванням" – з заздалегідь вбудованими чи відсутніми прихованими даними. Мережі аналізують надані файли-приклади та формують внутрішні аналітичні моделі для подальшої класифікації нових даних на стеганографічно "чисті" та модифіковані.

1.3 Вимоги та критерії оцінки стеганографічних систем

Щоб уникнути розшифрування стеганографії методом перевірки алгоритму роботи самої програми, існують критерії та вимоги для створення таких стеганографічних системи. Серед основних критерій вимог включаються захищеність. Цей критерій вимірюється за допомогою метрик, таких як перевірка відповідності експериментального розподілу теоретичному, оцінка надлишковості зображення після стеганографування, тест Колмогорова-Смирнова та аналіз

спектральних аномалій.

Одним з поширених статистичних методів виявлення цифрових вбудовувань в дані є непараметричний критерій узгодженості Колмогорова-Смирнова (К-С) [39]. Він базується на аналізі відмінностей у розподілах ймовірностей між оригінальним та потенційно модифікованим контейнером. Сутність полягає в наступному:

- будується емпірична функція розподілу ймовірностей (ЕФР) оригінальних даних FO(x), вона зображена на рисунку 1.5;

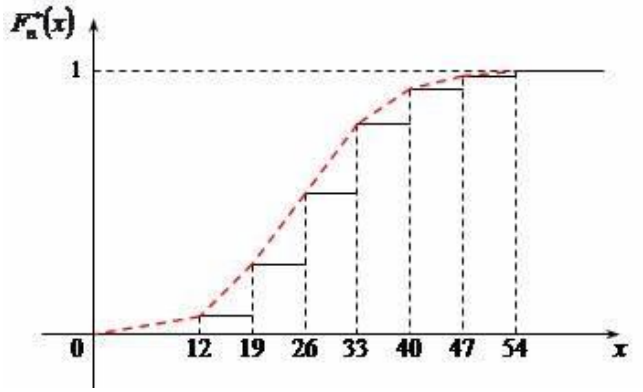


Рисунок 1.5 – Емпірична функція розподілу

- обчислюється ЕФР для перевіряємого контейнеру FE(x);

- знаходиться максимальна різниця між цими функціями за формулою:

$$D = \max |FO(x) - FE(x)| \qquad (1.1)$$

Чим більше значення D, тим ймовірніше, що дані було модифіковано шляхом стеганографічного приховування інформації. FO і FE – емпіричні функції для оригіналу і заміненого файлу.

Іншою ключовою характеристикою якості стегосистеми є її ємність – максимальний об'єм приховуваних даних. Для кількісної оцінки ємності при цифровому маркуванні зображень використовують такі показники:

- вбудовування в бітах на піксель (BPP):

$$BPP = S / N, \qquad (1.2)$$

де S – розмір прихованого повідомлення в бітах,

N – загальна кількість пікселів зображення

- відносна ємність (CR) у %:

$$CR = (S / N) / (D / 8) \cdot 100\%, \quad (1.3)$$

де D – глибина кольору зображення в бітах (напр., 24 біти).

Чим вищі BPP та CR, тим більший обсяг даних може бути вбудовано в контейнер без втрати якості. Оптимальні значення цих показників залежать від вибраного методу цифрового маркування та типу носія.

Ще одним з важливих критеріїв якості стегосистем є стійкість до різноманітних перетворень та атак, тобто здатність зберігати цілісність прихованих даних за несприятливих умов. Для кількісної оцінки стійкості використовують такі показники:

- бітова помилка (BER) – відношення кількості помилково декодованих бітів прихованого повідомлення Ne до його початкового об'єму N:

$$BER = Ne / N \quad (1.4)$$

- середньоквадратичне відхилення (MSE) між оригінальним повідомленням X та декодованим X:

$$MSE = (1/N)*\Sigma(Xi - Xi)2, \quad (1.5)$$

де Σ – сума за усіма бітами повідомлення

Чим менші значення BER та MSE, тим стійкішою є стегосистема до впливів, що вносять спотворення в приховані дані під час передачі.

Не менш важливою складовою оцінювання якості стегосистем є аналіз їх стійкості до криптоаналітичних атак (безпека), метою яких є несанкціоноване розкриття або модифікація прихованих даних [40]. Основними видами атак є:

- атака грубою силою (brute force) – перебір усіх можливих ключів шифрування;

- диференціальний криптоаналіз – аналіз статистичних залежностей між вхідними та вихідними даними шифру;

- лінійний криптоаналіз – побудова лінійних наближень до алгоритму шифрування.

Для оцінки стійкості аналізується оцінювана складність злому стегосистеми зазначеними методами в залежності від довжини ключа, режиму шифрування тощо [41].

Оцінка якості стегосистеми та внесених нею спотворень у контейнер здійснюється за допомогою об'єктивних метрик. Однією з найбільш поширених є пікове відношення сигнал/шум, відоме акронімом PSNR. Воно дозволяє кількісно оцінити рівень спотворень, внесених під час цифрового маркування, шляхом аналізу відхилень між оригінальним та стеганографічно модифікованим сигналом. Іншою ефективною метрикою є індекс структурної схожості SSIM. На відміну від PSNR, який оперує лише амплітудними характеристиками, SSIM додатково враховує структурні властивості сигналу, що робить його більш адекватним для оцінки спотворень, які сприймає людське око чи вухо. Обидві метрики широко застосовуються для контролю якості в процесі розробки нових цифрових стеганографічних методів.

1.4 Огляд та порівняльний аналіз існуючих стеганографічних методів для мультимедіа-об’єктів та постановка задачі дослідження

Існує багато програмних засобів для стеганографії, які дозволяють приховувати інформацію в різних типах медіафайлів.

OpenStego є вільним та відкритим програмним засобом стеганографії, який надає можливість приховувати дані в зображеннях [42]. Він створений з метою забезпечити простий та ефективний спосіб використання стеганографії для захисту конфіденційної інформації. Основною функцією OpenStego є використання методів стеганографії на основі LSB (Least Significant Bit). Цей метод передбачає

вбудовування прихованої інформації в найменш значущі біти зображення. Такий підхід дозволяє впроваджувати дані в зображення таким чином, що зовнішній спостерігач майже не може виявити наявність прихованої інформації. OpenStego підтримує різні формати зображень, включаючи BMP, PNG, JPEG та інші. Це дозволяє користувачам вибирати найбільш підходящий формат для їх потреб. Програмний засіб надає інтерфейс користувача, який дозволяє зручно вибирати зображення для вбудовування і витягування прихованої інформації. OpenStego також надає додаткові функції, такі як шифрування прихованих даних за допомогою паролю, що забезпечує додатковий рівень захисту. Крім того, він підтримує багатократне вбудовування, що дозволяє вбудовувати кілька шарів прихованої інформації в одне зображення.

Steghide є безкоштовним і відкритим програмним засобом стеганографії, який забезпечує можливість приховування інформації в аудіофайлах та зображеннях [27]. Це програмне забезпечення розроблено з метою надання простого та ефективного способу захисту конфіденційних даних. Steghide підтримує різноманітні формати файлів, включаючи JPEG, BMP, WAV та інші. Це дозволяє користувачам вибирати найбільш підходящий формат для їх потреб. Програма використовує методи стеганографії на основі LSB (Least Significant Bit), які полягають у вбудовуванні прихованої інформації в найменш значущі біти файлів. Однією з переваг Steghide є його можливість застосовувати парольну захист для прихованої інформації. Користувач може встановити пароль для шифрування прихованої інформації, що забезпечує додатковий рівень безпеки. Steghide надає командний рядок інтерфейсу, що дозволяє використовувати його у скриптах або в автоматизованих процесах. Це дає користувачеві більшу гнучкість у використанні програмного забезпечення з іншими інструментами та розширеннями.

SilentEye є кросплатформним графічним середовищем для стеганографії, яке дозволяє приховувати інформацію в зображеннях та аудіофайлах [42]. Цей програмний засіб розроблений з метою надання зручного та простого інтерфейсу для захисту конфіденційної інформації. SilentEye підтримує різні формати файлів

зображень, включаючи BMP, JPEG, PNG, GIF та інші. Він також підтримує аудіофайли у форматах WAV та FLAC. Це дає можливість користувачам вибирати найбільш підходящий формат для своїх потреб. Програмний засіб використовує різні методи стеганографії, зокрема LSB (Least Significant Bit) та DCT (Discrete Cosine Transform). LSB метод вбудовує приховану інформацію в найменш значущі біти зображень або аудіофайлів, тоді як DCT метод використовує перетворення дискретного косинусу для приховування даних у частотному діапазоні. Однією з переваг SilentEye є його простий та зрозумілий інтерфейс користувача. Він надає зручні інструменти для вбудовування та витягування прихованої інформації, а також налаштування параметрів стеганографії.

Outguess є вільним та відкритим програмним засобом стеганографії, який спеціалізується на приховуванні інформації в зображеннях [42]. Його основна мета – забезпечити наявність прихованої інформації і зберегти оригінальний вигляд зображення. Програмний засіб Outguess використовує статистичні методи для розміщення прихованої інформації в зображеннях. Він використовує аналіз характеристик зображення, таких як гістограми, розподіл пікселів та інші параметри, для внесення даних без видачі зайвої інформації. Це дозволяє прихованій інформації залишатись практично невидимою для зовнішнього спостереження. Outguess підтримує різні формати зображень, включаючи JPEG, BMP, PNG та інші. Він надає можливість вбудовувати текстові повідомлення, файлові архіви або будь-яку іншу форму даних у зображення. Вбудовані дані можуть бути витягнуті зображенням з використанням відповідного ключа або паролю. Outguess має простий та зрозумілий інтерфейс користувача, що робить його доступним навіть для непрофесійних користувачів. Він надає можливість контролювати рівень вбудовування та забезпечує можливість вказати пароль або ключ для доступу до прихованої інформації.

S-Tools є програмним засобом для операційної системи Windows, який дозволяє приховувати інформацію в зображеннях та аудіофайлах [27]. Його головна функція полягає у вбудовуванні даних в найменш значущі біти зображення

або аудіо файлу, використовуючи методи стеганографії на основі LSB (Least Significant Bit). LSB-стеганографія використовує найменш значущі біти (LSB) пікселів або аудіо семплів для збереження прихованої інформації. Це означає, що значущість даних не змінюється візуально або аудіально, що дозволяє зберігати конфіденційні дані без підозрілих ознак. S-Tools підтримує різні формати файлів, включаючи зображення у форматах BMP, GIF, JPEG та PNG, а також аудіофайли у форматах WAV і AU. Користувач може вибрати бажаний файл-носій і вбудувати в нього секретні дані з використанням пароля або ключа, якщо необхідно. Додатково S-Tools надає можливість виконувати розбір файлів для вилучення прихованої інформації зображень або аудіофайлів. Це дозволяє отримати доступ до прихованої інформації, якщо ви маєте відповідний пароль або ключ. S-Tools має простий та зрозумілий інтерфейс користувача, що робить його доступним для широкого кола користувачів. Він може бути використаний як для особистого використання, так і для деяких професійних сфер, де конфіденційність даних є важливою.

Отже, після проведеного аналізу в попередніх підрозділах визначено, що розробка методів цифрової стеганографії для мультимедійних даних з використанням технологій штучного інтелекту є актуальною задачею.

Метою кваліфікаційної роботи є розробка методу цифрової стеганографії для приховування даних у мультимедіа-об’єктах на основі нейронних мереж.

Завдання роботи включають:

- огляд предметної області;
- аналіз літературних джерел досліджуваної тематики;
- аналіз існуючих методів та алгоритмів цифрової стеганографії;
- дослідження основних вимог до стеганографічних систем;
- постановка задачі дослідження;
- розробка алгоритму вбудовування прихованих даних у зображення/аудіо/відео;
- дослідження можливостей застосування нейромереж;
- програмна реалізація стеганографічного методу;

- дослідження розробленого методу та порівняльний аналіз з відомими рішеннями.

Розробка методу цифрової стеганографії повинна забезпечувати приховування даних в широкому спектрі мультимедійних файлів: зображеннях, аудіо та відео різних форматів. Стеганографічна система має надавати можливість вибору оптимального стеганографічного алгоритму залежно від типу контейнера та вмісту приховуваного повідомлення.

Запропонований підхід поєднує традиційні методи цифрової стеганографії з можливостями глибинного навчання для створення високоефективної стеганографічної системи приховування даних у мультимедіа контенті.

Висновки до розділу 1

1. Розглянуто теоретичні основи цифрової стеганографії та проведено огляд існуючих методів приховування даних у мультимедіа файлах, аналіз виявлення стеганографії у файлах.

2. Проаналізовано базові поняття, вимоги до стегосистем, а також основні підходи до стеганографії зображень, аудіо та відео. Досліджено роботи інших експертів, проаналізовано їх погляд на суть стеганографії, основні її компоненти та огляд методів. Проаналізовано наявні методи стегоаналізу.

3. Обґрунтовано, що актуальною задачею є розробка методу цифрової стеганографії для мультимедіа даних на основі адаптивного підходу з урахуванням властивостей конкретного контейнера, що дозволить підвищити збалансованість ключових характеристик, – ємності, непомітності, стійкості, – і зроблено постановку задачі дослідження.

2 МЕТОД ЦИФРОВОЇ СТЕГАНОГРАФІЇ ДЛЯ МУЛЬТИМЕДІА-ОБ’ЄКТІВ

2.1 Суть методу цифрової стеганографії для мультимедіа-об’єктів

Для реалізації програмного продукту необхідно розпочати з постановки ідеї програми. Суть роботи полягалає в тому, щоб розробити алгоритм програми, яка бере два файли – файл з інформацією для приховування та файл мультимедіа (напр., mp3 аудіофайл), в якому приховується інформація для зберігання чи надсилання іншому користувачу, в якого буде лише один вихідний файл, наприклад, – mp3 з прихованим всередині змістом. На рисунку 2.1 зображено схему алгоритму програми:

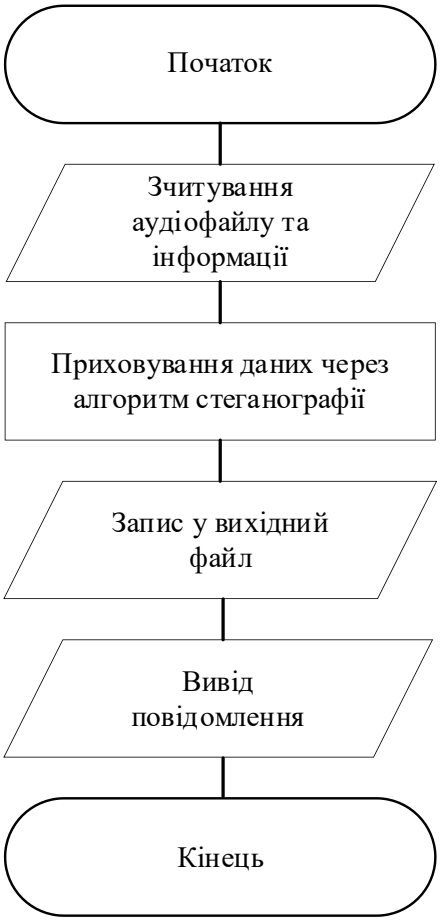


Рисунок 2.1 – Схема алгоритму програми

Описуючи більш детально, задум полягає в тому, що файли, які передаються, будуть стискатись, тобто це може бути будь-яка інформація: від

невеликої стрічки до повноцінного архіву. За основу взято LSB метод, що означає, тип стеганографії буде відноситись до процесу заміни молодшого значущого біта байтів файлу-контейнера на біти, які утворюють дані, що потрібно приховати. Першим викликом є те, що існують причини, через які неможливо безпосередньо маніпулювати окремими бітами окремих семплів, як у форматі WAV. Тобто, щоб використати метод, потрібно буде конвертувати любий вхідний файл MP3 у WAV. Таким чином можна модифікувати окремі біти семплів, що є надто малими, для надійного стиснення файлів. Таким чином, після конвертування при такій техніці стеганографії є можливість розбивати аудіосигнал на невеликі фрагменти, а потім обчислювати ШПФ.

Тобто, щоб повернути закодований аудіосигнал у «часову область», відбувається зворотне ШПФ кожного кодованого фрагменту в серії ШПФ. Кодування виконується чергуванням фрагментів, використовуючи немодифіковані фрагменти як контрольні точки. Лінійна інтерполяція між цими контрольними точками визначає поріг між одиницями та нулями, який нам потрібен для їх відновлення.

Важливо, щоб ряд FFT мав послідовні вибірки, які сильно корельовані. Ось чому якщо взяти, наприклад, білий шум як вхід, буде важче приховати секретне повідомлення. На рисунку 2.2 продемонстровано ідею таких перетворень:

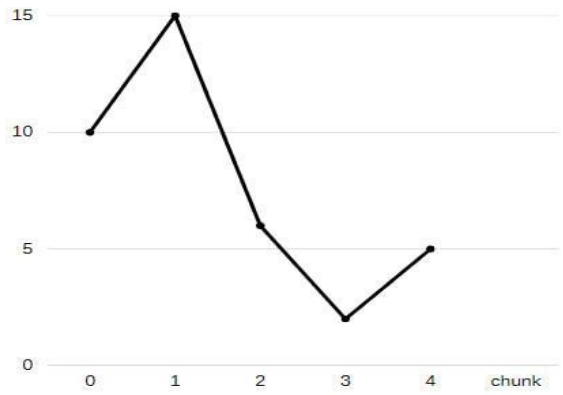


Рисунок 2.2 – Лінійна інтерполяція в рамках БПФ

Переходячи до логіки програми, найважливіше, що слід зауважити, це те, що, орієнтуючись на досить «високу ємність», зберігаючи спотворення на дуже низькому рівні (не чутному), не всі маніпуляції сигналом «витримають» стиснення аудіо. Тобто, після вбудовування повідомлення всередину носія, який обробляється в PCM-форматі, а потім перетворюється в стиснутий аудіофайл, результат не міститиме точно всіх внесених нами змін.

PCM – імпульсно-кодова модуляція, яка представляє аудіосигнал у не стисненому цифровому форматі. Стиснене аудіо не відтворює оригінальний сигнал PCM точно. Розширений опис PCM (Pulse-Code Modulation) – формату подання аудіо даних: PCM (імпульсно-кодова модуляція) – це метод представлення аналогових аудіосигналів у цифровому вигляді шляхом регулярного вимірювання амплітуди хвилі з фіксованою частотою та подання виміряних значень у бінарному коді.

Фактично PCM забезпечує не стиснене цифрове подання аудіосигналу у вигляді послідовності відліків його миттєвих значень, квантованих з певною точністю. Це дозволяє з високою якістю відтворювати оригінальний аналоговий запис в цифровому форматі. На відміну від стислих аудіоформатів, PCM дає точну бітову копію сигналу, але потребує значних об'ємів пам'яті. Саме тому дані у форматі PCM часто використовуються в задачах цифрової обробки та аналізу звуку.

Тобто, під час розробки циклу роботи програми введено крок зворотного читання, який перевіряє, які біти були успішно записані в стиснутий файл, а які ні. Цей крок повторюється, поки всі біти не будуть успішно закодовані. Під час кодування проблемні біти обробляються в кілька етапів із застосуванням наступних правил одне за одним у разі послідовних помилок:

- перекодувати: незначним чином підвищити рівень кодування, щоб збільшити ймовірність успіху в наступній ітерації;

- примусовий плюс: кодовий біт, щоб перевищити критерії прийняття та бути відкинутим (перевірка правдоподібності максимального значення буде

невдалою під час наступної ітерації, і біт буде позначено як пропущений).

Такий метод називається ітераційним підходом. Важливо під час обробки цих послідовностей відстежувати збіжність алгоритму, досягаючи бажаних результатів протягом найкоротшого часу. Отже, один із критеріїв завершення – це максимальна дозволена кількість ітерацій, після яких робота переривається та завершується з повідомленням про помилку.

Це може статися, наприклад, якщо спробувати вставити файл, який є занадто великим, щоб помістится в носій. Потрібно зауважити, що використовуються терміни «носій» і «контейнер», «файл» або «сигнал» як синоніми.

На рисунку 2.3 зображено схему алгоритму програми, враховуючи конвертацію файлів, перехоплення помилок та ідею самої роботи.

Оскільки алгоритм на основі аудіофайлу розглянуто першим, можна описати метод приховування аудіо. Метод LSB використовує надмірність звукових файлів. Як відомо, молодші біти цифрових відліків містять дуже мало корисної інформації, тому є можливість замістити їх своїм матеріалом. Їх заповнення своєю інформацією ніяк не вплине на якість сприйняття, що забезпечує чудову можливість приховування.

Негативні ж особливості – те, що зі зміною інформації спотворенню піддаються власне статистичні атрибути цифрових потоків. Зважаючи на це, для зменшення компрометуючих ознак необхідно застосувати їх корекцію. Переваги полягають у можливості прихованої передачі великих даних, а також можливості захисту.

Дані часто архівуються (щоб зменшити розмір) або шифруються (щоб забезпечити додатковий захист, якщо повідомлення потраплять до чужих рук), перш ніж їх приховати. Це робить біти даних дуже близькими до випадкових. Тут послідовна інсталяція такої інформації змінює MZR звукового файлу випадковими бітами. У цьому і полягає головна суть методу – фіксація різниці між розподілом первинних сортів у порожній і заповненій тарі.

Рисунок 2.3 – Схема алгоритму модуля аудіостеганографії

Щоб перевірити цей метод, був розроблений спеціальний метод, який включав використання інструменту статистичного тестування NIST,

розробленого в Сполучених Штатах.

Тест програмного забезпечення NIST був розроблений для статистичного тестування випадковості двійкових послідовностей.

Для відпрацювання методики була створена тестова база даних з 108 різнотипних аудіо файлів, припускаючи використання їх як контейнер. У базу було включено шуми (білий, чорний, коричневий, рожевий, зелений тощо), рівні синусоїди, записи мови, інструментів (гітара, саксофон, барабани тощо), і навіть повноцінні музичні композиції. База даних складається з файлів різних бітностей (8, 16, 24, 32 біти) та частоти дискретизації. Розмір файлів від 38КБ до 41 522КБ.

Щоб відстежувати поведінку випадковості молодших біт, необхідно впорядкувати файли за якоюсь ознакою. Як таку ознаку було обрано відносну кількість нульових байт у файлі. Відносна кількість нульових байт визначається як відношення нульових байт файлу до загальної кількості байт.

Щоб перевірити цю гіпотезу, частково заповнені файли були додані до вихідної бази даних (за винятком порожніх контейнерів), тобто всі файли в базі даних були позначені (за допомогою розробленого програмного забезпечення Hide_Wave_Stego) як 10%, 50% і 100% – такі файли вважаються максимально місткими для стеганографічного контейнера.

Як тест, використовувався частотний бітовий тест пакету NIST, який оцінює співвідношення між 0 і 1 у двійковій послідовності. Алгоритм для цього тесту такий: файл сприймається як послідовність бітів, де 1 дорівнює +1, а 0 дорівнює 0, і обчислюється сума цих послідовностей. Далі статистика обчислюється за формулою:

$$S_{obs} = \frac{|S|}{\sqrt{n}}, \tag{2.1}$$

де | S | – сума послідовності,

n – кількість елементів у послідовності.

Розраховується P значення за формулою через додаткову функцію

помилок, і якщо результат більший за 0,01, то послідовність визнається випадковою з рівнем довіри 99%.

Якщо розглядати алгоритм із відео, то це метод F5, що був розроблений Андреасом Вестфельдом для стеганографії в JPEG-зображеннях, але його концепції можуть бути адаптовані для відеофайлів. F5 відрізняється від інших методів стеганографії тим, що він протистоїть візуальним та статистичним атакам, забезпечуючи велику ємність для приховування комбінації.

Для адаптації методу F5 до відеофайлів, можна розглянути кроки, зображені на рисунку 2.4.

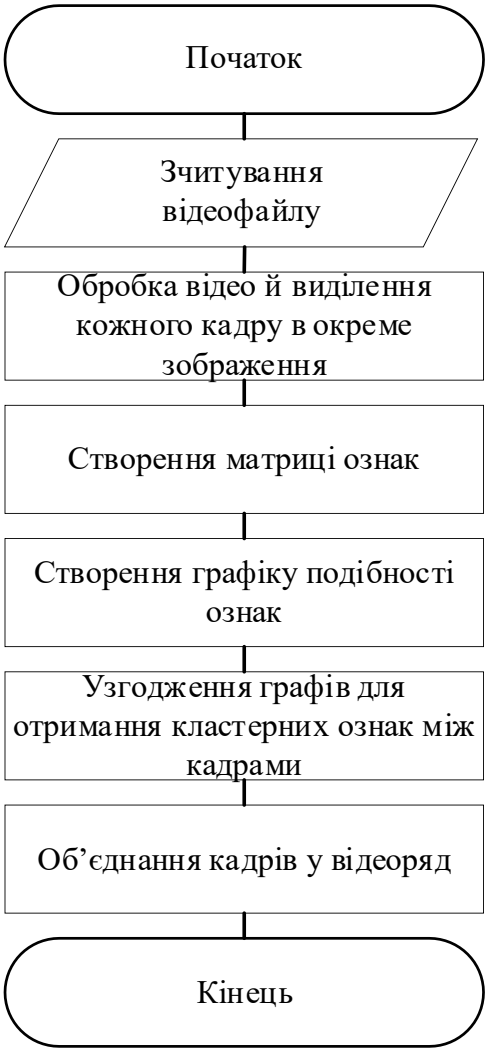


Рисунок 2.4 – Схема алгоритму адаптації методу F5 для відео

Основні ідеї методу F5, які можуть бути адаптовані для відео, включають:

- матричне кодування: F5 використовує матричне кодування для

підвищення ефективності вбудовування. Це зменшує кількість необхідних змін у відеофайлі, забезпечуючи менш помітне вбудовування;

- взаємозамінне розподілення: F5 використовує Взаємозамінне розподілення для рівномірного розподілу змін по всій стеганограмі (прихованому відео). Це допомагає зробити приховану інформацію менш помітною та важкою для виявлення.

Існує декілька способів представлення зображення у частотній області. Наприклад, з використанням дискретних косинусних перетворень (ДКП), швидкого перетворення Фур'є або вейвлет-перетворення. Ці перетворення можуть застосовуватись як до всього зображення, так і до певних його частин. Одним із сучасних, розповсюджених на сьогоднішній день методів, є метод стеганографічного захисту інформації F5. Замість вбудовування в найменший біт, операція вбудовування в F5 може лише зменшити абсолютне значення ДКП коефіцієнта на одиницю. Така операція не міняє форму ДКП гістограми, що після вбудовування виглядає як у випадку ущільнення оригінального зображення з меншим параметром якості. Метод F5 вбудовує біти повідомлення у псевдо-випадково обрані коефіцієнти, ігноруючи DC-коефіцієнт та коефіцієнти з нульовим значенням. Якщо значення коефіцієнта змінюється з 1 або -1 на 0 ("стягування"), то біт повідомлення, що в такому випадку завжди є нулем, заново вбудовується у наступний коефіцієнт. Це пояснюється врахуванням лише нульових коефіцієнтів при витяганні даних одержувачем. Однак, при повторному вбудовуванні нульового біта кількість нульових коефіцієнтів перевищує кількість одиничних, що може сприяти утворенню "східчастої" гістограми завдяки її монотонності на (-∞,0] та [0,∞). У F5 ця задача розв'язується шляхом повторного визначення найменшого біту для від'ємних значень коефіцієнтів:

$$\mathrm{LSB}(X)\ 1\ x \bmod 2 \qquad (2.2)$$

для $x < 0$. Максимальний обсяг даних, що можна вбудувати за допомогою F5, складає:

$$n = n\ Hist(0)\ (Hist(\ 1)\ Hist(1))\ /\ 2\ 64 \quad (2.3)$$

де n — загальна кількість ДКП-коефіцієнтів,

Hist — гістограма ДКП-коефіцієнтів.

Перші три складові n – Hist (0) – n/64 визначають кількість ненульових АС-коефіцієнтів, остання складова (Hist (-1) + Hist (1))/2 визначає втрати, обумовлені «стягуванням». Метод стеганографічного захисту інформації F5 є методом з відносно високою ємністю, що в середньому дозволяє вбудувати 0,75 біт на кожен ненульовий ДКП-коефіцієнт при якості JPEG-ущільнення 80%.

Незважаючи на зміну методом F5 гістограми ДКП-коефіцієнтів, деякі важливі характеристики гістограми зберігаються, як, наприклад, монотонність. Оскільки принцип F5 не ґрунтується на обміні значень коефіцієнтів в парах, він є стійким до атаки з використанням статистики (гістограм) першого порядку. Однак F5 може бути виявлено за допомогою стеганоаналітичної процедури калібрації. У якості модифікації F5 для вбудовування коротких повідомлень додатково застосовується шаблонний метод, що підвищує питомий показник змін шляхом мінімізації кількості змін, які вносяться методом F5.

Проте існує і інший варіант – методи шаблонного вбудовування даних на основі матричного представлення кодів Хеммінга.

Змінюючи різні параметри зображення, можна вбудувати таємне повідомлення декількома способами. Наприклад, при вбудовуванні найменшого біту змінюються відповідні наймолодші біти значень інтенсивності пікселів зображення. Для цього методу вбудовування ціною однієї зміни в середньому досягається вбудовування 2 бітів зображення. Однак зазначений показник змін можливо покращити за умови, якщо повідомлення, яке вбудовується, є набагато коротшим у порівнянні з максимальною довжиною. Надалі через β позначено функцію, що ставить у відповідність біти повідомлення певним параметрам зображення. Передбачається, що існують такі функції вбудовування та витягування, які вимагають щонайбільше R змін для будь-якого повідомлення m є M:

$$\text{Emb} : \{0,1\} \text{ M } \{0,1\} \text{ ; Ext} : \{0,1\} \text{ M} \quad (2.4)$$

Враховуючи, що обсяг бітів повідомлення складає log2|M| та за рівномірного розподілу повідомлення m в зображенні x, питомий показник змін для Ra≤R визначається як e=(log2|M)/Ra. Через e=(log2|M)/R позначено нижню межу питомого показника змін. Шаблонне вбудовування реалізується шляхом використання лінійного коду ζ, що описується параметрами [n, k], де n>k та позначається кодове слово і порція вбудовуваних даних, відповідно. Будь який лінійний [n, k] код ζ повністю описується його твірною матрицею G, що представляє собою регулярну бінарну матрицю з k рядками та n стовпчиками. Будь-яке кодове слово c є ζ можна отримати як лінійну комбінацію рядків G, де коефіцієнти описуються k бітами. Для матриці відповідності H розміром (n – k)×n та деякого кодового слова c' справедливо:

$$Hc' = 0, \text{якщо } c' \in \zeta$$
$$Hc' = 0, \text{якщо } c' \notin \zeta \quad (2.5)$$

Може існувати багато різних кодових слів c', що задають єдиний вектор s = Hc', і множину яких позначають ζ(s). Отже, використовуючи [n, k] код, можна вбудувати (n – k) бітів повідомлення. Надалі m вважається (n – k)-бітним повідомленням. Вбудовування вимагає зміни певних бітів у послідовності x, отриманої з оригінального зображення за допомогою β. Результуюча послідовність y, одержана з x, має задовольняти Hy= m. Якщо визначити y=x+e, де e – вектор змін, то вага за Хеммінгом для e рівна кількості внесених при вбудовуванні змін та визначається: H(x e) Hy m He m Hx . Якщо задовольняється будь-яким e з множини ζ(m – Hx). Отже, для лінійного [n, k] коду ζ з радіусом R необхідно обрати з найменшою вагою за Хеммінгом, що однозначно не перевищує R.

Останнім алгоритмом є методи роботи з зображеннями. Метод JSteg є одним з підходів до стеганографії, який дозволяє приховувати дані всередині JPEG-зображень. Цей метод базується на техніці заміни найменш значущих бітів

(LSB) коефіцієнтів дискретного косинусного перетворення (DCT) зображення.

Алгоритм роботи такого продукту заключається в роботі зі зображенням, декодування ентропії, інверсивного ДКП і заміни LSB. Алгоритм роботи зображено на рисунку 2.5.

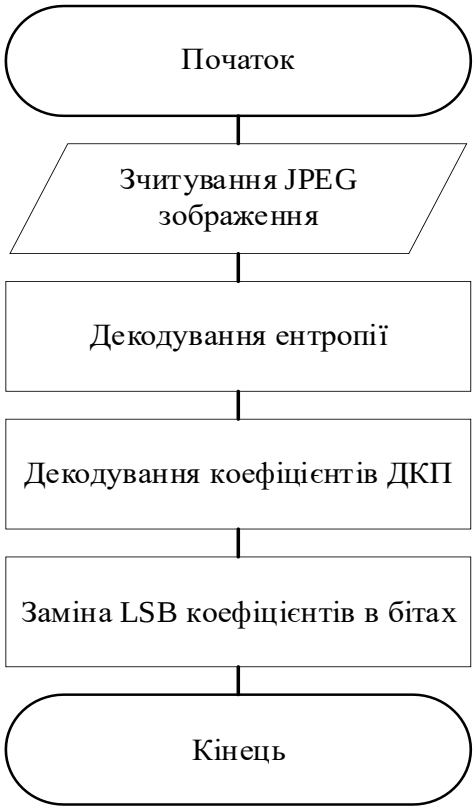


Рисунок 2.5 – Схема алгоритму методу JSteg

2.2 Алгоритми візуалізації програмного продукту

Візуальний інтерфейс користувача (UI) відіграє важливу роль в будь-якому програмному забезпеченні, оскільки саме він визначає зручність та простоту використання продукту. Вдало спроєктований UI дозволяє користувачам швидко розібратися в функціоналі, інтуїтивно знаходити потрібні опції, ефективно взаємодіяти з програмою. Натомість поганий UI може викликати незадоволення, стрес і в результаті відмову від використання продукту.

Є декілька варіантів розробки алгоритмів для візуалізації роботи та процесу в продукті. Першим варіантом є робочий і простіший варіант з консолі. Алгоритм

роботи зображений на рисунку 2.6.

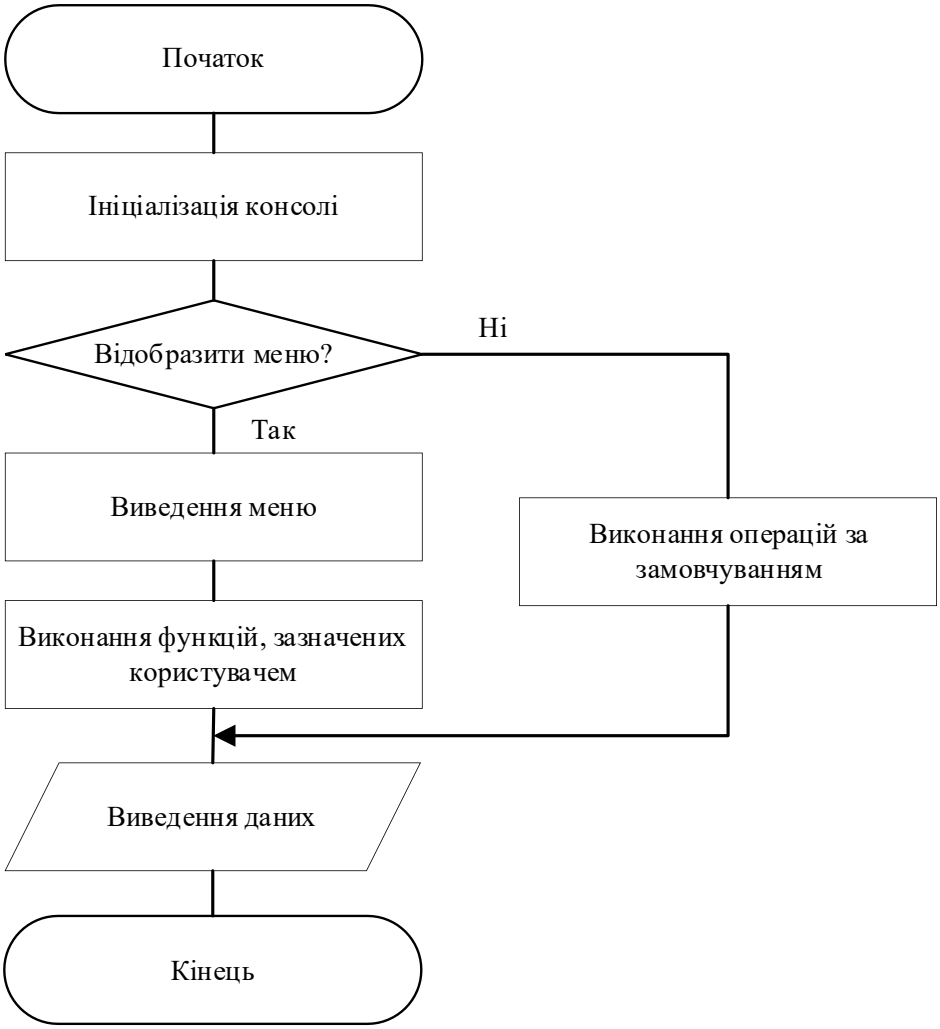


Рисунок 2.6 – Схема алгоритму роботи програми без графічного інтерфейсу

Варіант включає в себе відображення меню з різними опціями. Це можуть бути різноманітні елементи, вбудовані в програму, такі як аналіз чи метод стеганографії.

Ініціалізація кнопок відповідатиме за ввід даних з консолі, а саме прийняття рішення через умовний оператор "ЯКЩО", де відповідь може бути "ТАК" або "НІ". Таким чином, такий метод буде зручний для використання продукту в операційних системах, відмінних від MacOS або Windows. Адже для роботи з такими продуктами найчастіше використовують операційні системи, які не включають в себе користувацький інтерфейс.

Особливо зручним буде режим користування програмою за замовчуванням,

який буде запам’ятовувати останні вибрані параметри в програмі і використовуватиме всі необхідні змінні зразу, не підтверджуючи кожен раз набір змінних.

Введення даних включає в себе декілька елементів, починаючи з елементарного вибору опції, так і більш гнучких параметрів, таких як швидкість обробки або ж назва файлу.

Другим варіантом є створення повноцінного GUI для користувача, що є менш пріоритетним, проте заслуговує бути розглянутим у цьому розділі. Для реалізації GUI потрібно виділити менше елементів, аніж це відбуватиметься при розробці без нього. Наприклад, основними елементами є: відображення головного вікна (полотна програми), додавання кнопок, розташування елементів керування і підключення обробника подій. Оскільки важко виділити схему алгоритму для GUI, можна запропонувати схему розробки такого елемента.

Початок розробки, який перетече в ініціалізацію UI. Встановлення власних стилів, який також розподіляється на декілька етапів, адже якщо використання буде PyQt5, то цей процес буде реалізовано за допомогою CSS та HTML, а якщо через Tkinter, то через внутрішній код самого модуля.

Розглянемо детальніше алгоритм ініціалізації графічного інтерфейсу користувача (UI) для програми стеганографії на Python з використанням бібліотеки PyQt5.

Імпорт необхідних модулів PyQt5:

- QtCore, QtWidgets – базові модулі для створення UI;

- QtGui – робота з графікою, стилями;

- QtWebEngineWidgets – браузерні віджети.

Створення головного вікна додатку:

- встановлення розмірів, заголовку, значка.

Завантаження зовнішніх файлів стилів CSS:

- можливість гнучкої зміни оформлення за допомогою каскадних таблиць стилів.

Ініціалізація віджетів інтерфейсу:

- меню, панелі інструментів, кнопки, текстові поля;
- полотно для перегляду зображень;
- таблиці, список файлів;
- браузер для перегляду документації;
- розміщення віджетів з використанням менеджерів компонування;
- підключення функцій-обробників подій і сигналів.

Такий підхід дозволяє створити гнучкий та функціональний GUI з використанням переваг мультиплатформенної бібліотеки PyQt. Проте це лише частина етапу розробки, адже після створення стилів продовжується розташування основних кнопок, підключення їх логіки та подій, які вони викликатимуть.

Проте, крім класичних способі візуалізації ПЗ є також декілька інших підходів, які також дозволяють користувачу користуватись програмою. Одним з таких може бути веб-інтерфейс, який також включає в себе можливість виклику програми. Проте, сам продукт працюватиме в двох напрямках: виконання методів стеганографії через вбудовані запити, – розробка бекенду та фронтенду, – а також виконання запитів через субпроцеси.

Виконання через веб-запити:

- на фронтенді створюється веб-форма для завантаження файлів та вибору параметрів стеганографії;
- дані відправляються на бекенд у вигляді HTTP запиту;
- на бекенді приймається запит та файли;
- викликається функція з вибраним алгоритмом стеганографії;
- результат зберігається на сервері;
- фронтенд отримує результат для скачування.

Виконання через субпроцеси:

- створюється окремий процес для стеганографічних перетворень;
- файли та параметри передаються в цей процес;

- в субпроцесі відбувається необхідна обробка даних;
- результат повертається в головний процес;
- користувач отримує готові файли від головного потоку.

Таким чином, серед всіх пропонованих методів обрано алгоритм, описаний першим (реалізація через консоль), адже він здається найбільш адаптивним для користувачів, які використовують його на пристроях, де може не бути інтернету, де може не бути GUI, де може бути занадто слабкий апаратний елемент, щоб підгрузити візуал. Проте, задля надання можливості більш широкого застосування серед звичайних користувачів також додатково вирішено реалізувати графічний інтерфейс.

2.3 Використання технологій нейронних мереж

Одним з перспективних напрямків для підвищення ефективності стеганографічних методів є використання технологій нейронних мереж.

Нейронні мережі можуть застосовуватися в стеганографії для вирішення багатьох завдань. Наприклад, оптимізація існуючих алгоритмів шляхом підбору оптимальних параметрів для конкретних носіїв чи видів приховуваних даних може вирішуватись через ШІ. Нейромережа аналізує зображення-носії та визначає найкращі налаштування стеганографічного методу.

Оптимізація параметрів стеганографічних методів за допомогою штучного інтелекту – перспективний напрямок досліджень, що дозволяє підвищити ефективність приховування даних в цифрових контейнерах.

Традиційні стеганографічні системи мають ряд параметрів, які потребують налаштування під конкретний тип носія та вбудовуваних даних. Наприклад, деякі алгоритми працюють з окремими бітами зображення, інші – з коефіцієнтами дискретних перетворень. Від налаштувань залежить як ємність сховища, так і стійкість методу до виявлення та атак. Оптимальний вибір цих параметрів для конкретної стегосистеми є складним науковим завданням.

Тут на допомогу приходять методи штучного інтелекту, зокрема нейронні мережі. Спеціальна нейромережева модель може проаналізувати велику кількість зображень-контейнерів, а також статистичні характеристики приховуваних повідомлень (текст, зображення, аудіо). На основі глибинного навчання на масивах зразків мережа здатна виявляти складні залежності між властивостями носіїв, стегоданих та ефективністю алгоритмів вбудовування для цих типів даних. У результаті нейромережа пропонує оптимальні налаштування існуючих методів приховування інформації для конкретних вхідних файлів користувача.

Даний напрямок досліджень і розробок відкриває широкі можливості покращити якість стеганографічних програмних та апаратних комплексів. Адаптивний підбір параметрів під задані умови дозволить суттєво збільшити обсяги безпечного приховування важливих даних в мультимедійних файлах та потоках, таких як зображення, відео, звукозаписи. Застосування штучного інтелекту сприятиме подальшому прогресу комп'ютерних методів стеганографії як важливої складової інформаційної безпеки.

На рисунку 2.7 зображено алгоритм задання параметрів людиною. Проблема полягає в тому, що швидкість перебору параметрів ШІ значно більша і зачасту більш вдала, аніж людиною, оскільки ШІ навчений на параметрах має змогу підібрати найкращі комбінації елементів, які людина може перебирати для кожного окремого випадку сама. Більше того, часто буває таке, що параметр може бути не один, а декілька.

Людина-дослідник послідовно перевіряє різні значення параметрів, зокрема:

- частоту дискретизації – кількість відліків звукового сигналу за секунду (наприклад: 8000 Гц, 44100 Гц, 48000 Гц, 96000 Гц.);

- бітову глибину – розрядність представлення кожного відліку (наприклад: 16 біт (96 дБ), 24 біти (144 дБ));

- тривалість аудіо.

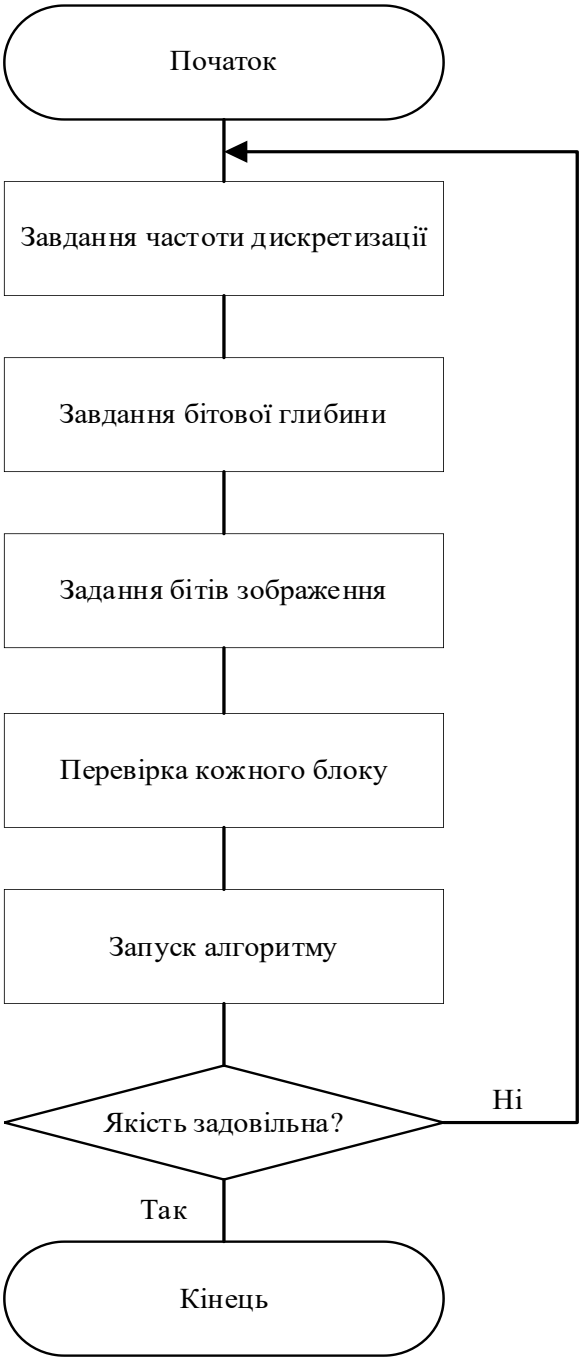


Рисунок 2.7 – Схема алгоритму задання параметрів

Для кожного набору запускається стегоалгоритм, аналізується якість аудіо та обсяг вбудованого зображення. Це потребує багато часу для обробки та прослуховування. Зрештою обираються оптимальні параметри.

Нейронна мережа аналогічно перебирає можливі поєднання вхідних даних та параметрів алгоритму. Але робить це набагато швидше за рахунок паралельних обчислень. Замість прослуховування здійснює цифровий аналіз отриманих

аудіоданих. Мережа здатна перевірити тисячі комбінацій за лічені секунди та запропонувати досліднику кращий результат. Такий підхід значно прискорює налаштування складних стегосистем.

Розробка принципово нових стеганографічних методів на основі технологій глибинного навчання є вкрай перспективним напрямком сучасних досліджень. На відміну від традиційного підходу, який передбачає реалізацію дослідниками певних алгоритмів приховування інформації, тут пропонується доручити розробку стеганографічних схем новітнім нейронним мережам.

Завдяки досягненням у галузі глибинного навчання з'явилася можливість конструювати складні архітектури, що складаються з багатьох шарів і здатні до самостійного набуття знань при аналізі великих масивів даних. Саме такі мережі пропонується застосовувати для розв'язання задач стеганографії. Зокрема, згорткові нейронні мережі добре підходять для обробки зображень, відео та аудіо – типових носіїв прихованої інформації. Їх можна навчити безпосередньо вбудовувати повідомлення в мультимедійні файли за допомогою алгоритмів, які мережа синтезує самостійно на основі аналізу прикладів.

Такий підхід відкриває принципово нові можливості для розвитку комп'ютерних методів стеганографії. Нейромережі здатні знаходити неочевидні для людей закономірності і конструювати надзвичайно складні алгоритми приховування, які складно розпізнати. Це виводить стеганографію на якісно новий рівень захисту інформації.

Використання нейромереж для стеганоаналізу – визначення наявності прихованих повідомлень в контейнерах та їх вилучення. Методи штучного інтелекту та нейронних мереж знаходять застосування не лише для вдосконалення стеганографії, але й для побудови ефективних засобів стеганоаналізу – виявлення прихованих повідомлень. Адже з розвитком технологій приховування даних виникає протилежне завдання – знайти сліди вбудованої інформації в контейнерах, таких як зображення, відео, аудіо.

Для стеганоаналізу також дедалі частіше використовують нейронні мережі

глибинного навчання. Наприклад, згорткові мережі здатні відшукувати мікроскопічні ознаки в цифровому контенті, що вказують на присутність прихованих даних. Мережі навчають на великих обсягах зразків з "чистими" та стегоконтейнерами певного типу. В результаті модель може з високою ймовірністю класифікувати файл як такий, що містить секретні дані.

Крім детектування стеганографії, досліджуються методи вилучення самих вбудованих повідомлень за допомогою нейромереж. Для цього потрібно навчити мережу реверсувати процес приховування даних в контейнері на основі її "підозрілих" ділянок чи артефактів. Хоча це складне завдання, певні успіхи вже досягнуті завдяки глибинному машинному навчанню.

Виходячи з роботи "Використання штучного інтелекту в стеганографії" [44], нейронні мережі дозволяють істотно автоматизувати процес вбудовування прихованих повідомлень в мультимедійні файли, такі як зображення, аудіо та відео.

Вдало навчені нейронні мережі можуть ефективно аналізувати візуальний контент і визначати найкращі місця для вбудовування з мінімальним викривленням. Переваги використання CNN (convolutional neural network – згорткова нейронна мережа) для стеганографії в зображеннях:

- висока точність – мережі аналізують зміст зображень та вибирають найкращі місця для приховування даних;

- мінімальне візуальне спотворення – нейромережі враховують унікальні особливості кожного зображення: розподіл кольорів, текстур, об'єктів тощо. Це дозволяє мінімізувати візуальні спотворення та зробити стеганографію практично непомітною;

- адаптивність – за рахунок підлаштування архітектури та навчальних даних, одна й та ж мережа може ефективно працювати з різною графікою, що значно розширює сфери застосування стеганографії;

- масштабованість – здатність обробляти зображення будь-якого розміру.

Також ШІ може використовуватись у стеганографії аудіо. Нейронні мережі відкривають нові можливості для стеганографії в цифрових аудіофайлах. На

відміну від традиційних алгоритмів, які приховують дані шляхом заміни або модуляції окремих фрагментів звуку, нейромережі дозволяють комплексно аналізувати аудіо та адаптивно підбирати оптимальні місця для вбудовування повідомлень.

Зокрема, рекурентні LSTM (long short-term memory – довга короткочасна пам'ять) мережі здатні враховувати особливості звукових хвиль, їх частоти, гучності та подібності між фрагментами. На основі цього аналізу обираються найменш помітні для людського вуха ділянки, куди можна зашифрувати та вбудувати потрібну інформацію.

Мережа LSTM є штучною нейронною мережею, яка містить вузли LSTM замість, або на додачу, до інших вузлів мережі. Вузол LSTM – це вузол рекурентної нейронної мережі, який виділяється запам'ятовуванням значень для довгих, або коротких проміжків часу. Ключем до цієї здатності є те, що він не використовує функції активації в межах своїх рекурентних складових. Таким чином, значення, що зберігається, не розплющується ітеративно з плином часу, і член градієнту або вини (англ. blame) не має схильності розмиватися, коли для його тренування застосовується зворотне поширення в часі [44].

Мета – вибрати пари сусідніх стовпчиків з мінімальною різницею значень так, щоб зміна їх місцями вносила найменші візуальні спотворення зображення. Це пришвидшить роботу програми, а також зробить її більш стійкою.

Структура нейронної мережі включає декілька елементів:

- вхід: вектор значень гістограми каналу зображення;
- приховані шари: кілька шарів зворотного поширення для екстракції ознак;
- вихід: матриця пар стовпчиків і оцінка "придатності" кожної пари.

Висновки до розділу 2

1. Розглянуто основні аспекти проєктування алгоритмів для задач цифрової стеганографії.

2. В рамках загальної концепції запропоновано гібридний підхід, що поєднує традиційні методи з технологіями штучного інтелекту. Розроблено архітектуру системи, що включає модулі аналізу носіїв, формування стегоконтейнерів, оцінювання якості, оптимізації параметрів на основі нейромереж. З метою зручної взаємодії користувача розроблено алгоритми візуалізації у вигляді консольного та графічного інтерфейсів. Реалізовано відображення меню, графіків, панелей інструментів, логіки обробки подій та запитів.

3. Досліджено застосування нейронних мереж для оптимізації параметрів існуючих стегометодів, розробки принципово нових алгоритмів на основі глибинного навчання, а також для стегоаналізу та виявлення прихованої інформації. Запропоновані алгоритмічні рішення дозволять побудувати потужний програмний комплекс цифрової стеганографії з ефективним інтерфейсом та розвиненими можливостями на основі штучного інтелекту.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ДОСЛІДЖЕННЯ МЕТОДУ

3.1 Реалізація основних модулів програми

Алгоритм стиснення даних, відомий як кодування Хаффмана, є основою для стиснення файлів. Розглянемо правила префіксів побудови дерева Хаффмана, унікально декодовані коди та кодування як змінної, так і фіксованої довжини.

Кожен символ містить послідовність 0 і 1, яку зберігають вісім бітів. Оскільки кожен символ зберігає однакову кількість бітів, це називається «кодуванням фіксованої довжини».

Існує концепція використання «кодування змінної довжини». Скориставшись тим фактом, що деякі символи зустрічаються частіше за інші в тексті, можна розробити алгоритм, який може використовувати меншу кількість бітів, щоб представити той самий фрагмент тексту. У кодуванні змінної довжини присвоєно певну кількість бітів кожному символу відповідно до його частоти в тексті. Таким чином, деякі символи можуть мати лише 1 біт, інші – 2 біти, а деякі можуть бути закодовані з 3 бітами і так далі. Декодування змінної довжини є проблемою кодування.

Припустимо, існує рядок "aabacdab". Він має вісім символів і використовує 64 біти пам'яті за допомогою кодування фіксованої довжини. Частота символів a, b, c та d у рядку дорівнює 4, 2, 1, 1 відповідно. Спробуємо представити "aabacdab" з меншим числом бітів, беручи до уваги, що a зустрічається частіше, ніж b, а b зустрічається частіше, ніж c і d. Почнемо з того, щоб випадково присвоїти 1-бітовий код 0 символу a, 2-бітовий код 11 символу b і 3-бітові коди 100, 011 символам c і d відповідно.

Приклад рядка: a 0 b 11 c 100 d 011.

Якщо ці коди задовольняють правилу префікса, декодування буде однозначним (і навпаки).

Для реалізації потрібно створити декілька класів. Node – клас, який представляє вузол у дереві Хаффмана. У нього є атрибути, такі як символ (char),

частота зустрічання символу у тексті (freq), а також посилання на лівого та правого нащадків. PriorityQueue – клас, що реалізує пріоритетну чергу для вузлів дерева Хаффмана. Він використовується для побудови дерева на основі частоти символів. Build_huffman_tree(text) – функція, яка отримує текст і будує дерево Хаффмана. Вона обчислює частоту кожного символу в тексті, використовуючи пріоритетну чергу для створення дерева. Generate_huffman_codes(root) – функція, яка генерує коди Хаффмана для кожного символу на основі дерева Хаффмана. Huffman_encoding(text) – функція кодування тексту Хаффманом. Вона будує дерево Хаффмана, створює коди для кожного символу та кодує вихідний текст, записуючи закодований текст у бінарний файл 'encoded.bin'. Huffman_decoding(encoded_text, huffman_tree) – функція декодування закодованого тексту Хаффманом. Вона використовує побудоване дерево Хаффмана під час кодування для розкодування закодованого тексту та повертає початковий текст.

Цей алгоритм дозволяє закодувати вхідний текст за допомогою алгоритму Хаффмана, зберегти закодований текст у бінарний файл та, використовуючи дерево Хаффмана, розкодувати його назад до початкового вигляду. Спочатку в функції build_huffman_tree послідовно будується оптимальне дерево Хаффмана на основі частоти символів у вхідному тексті:

```
freq = {}
for char in text:
    freq[char] = freq.get(char, 0) + 1
```

У цьому прикладі спочатку створюється порожній словник freq для збереження частот символів. Потім за допомогою циклу for відбувається прохід по всіх символах вхідного тексту text і підраховується кількість повторень кожного унікального символу за допомогою функції get. Так, в freq зберігається інформація про частоту кожного окремого символу тексту.

```
pq = PriorityQueue()
for char, frequency in freq.items():
```

```
    pq.push(Node(char, frequency))
```

Далі створюється структура даних PriorityQueue, яка використовується як пріоритетна черга для побудови оптимального дерева Хаффмана. За допомогою циклу for відбувається прохід по всіх парах ключ-значення зі словника freq та додавання їх у чергу pq у вигляді об'єктів Node, що містять символ та його частоту.

```
while len(pq) > 1:
    left = pq.pop()
    right = pq.pop()
    total = left.freq + right.freq
    pq.push(Node(None, total, left, right))
return pq.pop()
```

Цикл while виконується поки в черзі pq більше 1 елементу. На кожній ітерації беруться 2 вузли з найменшими вагами за допомогою функції pq.pop() та об'єднуються в новий вузол так, щоб виконувались умови оптимальності кодування Хаффмана. Новий комбінований вузол додається назад в чергу pq. Після завершення циклу в черзі залишиться єдиний елемент – корінь оптимального дерева Хаффмана, який і повертається з функції. Повна версія коду подана в додатку А.

Далі в функції generate_huffman_codes здійснюється рекурсивний обхід побудованого оптимального дерева Хаффмана з метою генерації кодів Хаффмана для кожного окремого символу вхідного тексту:

```
def generate_code(node, code=""):
    if node:
        if node.char:
            huffman_codes[node.char] = code
        generate_code(node.left, code + "0")
        generate_code(node.right, code + "1")
```

Рекурсивна функція generate_code приймає поточний розглянутий вузол

дерева node та поточне значення коду символу code. За допомогою рекурсивних викликів для лівого і правого піддерев, де код доповнюється 0 або 1 відповідно, здійснюється обхід дерева Хаффмана. Коли розглядається вузол-листок, то поточне значення коду записується у словник huffman_codes за відповідним символом node.char.

Так формується словник кодів Хаффмана для символів вхідного тексту. Чим частіше зустрічається символ у тексті, тим коротшим буде його код Хаффмана.

Наостанок, у функції huffman_encoding ці коди Хаффмана використовуються для стиснення та запису вхідного тексту у бінарний файл:

```
encoded_text = ''.join(huffman_codes[char] for char in text)
with open('encoded.bin', 'wb') as f:
    #...
    f.write(bytes(bytes_list))
```

Спочатку текст замінюється на послідовність бітових кодів Хаффмана, потім ця бітова строка перетворюється у послідовність байтів з урахуванням вирівнювання до байтової межі, і в результаті записується у бінарний файл encoded.bin. Розширений код знаходиться у додатку А.

Ввівши текст "Це моя перша робота", отримуємо результат:

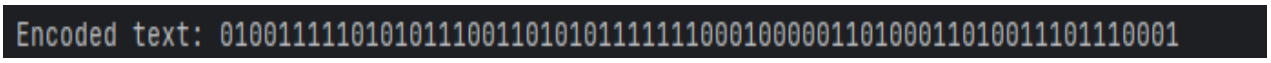


Рисунок 3.1 – Результат роботи програми кодування Хаффмана

Для реалізації методу декодування застосовується функція huffman_decoding, що приймає закодований бітовий рядок encoded_text та дерево Хаффмана huffman_tree, яке використовувалося при кодуванні, і на його основі виконує послідовне декодування:

```
current_node = huffman_tree
current_bit = 0
while current_bit < len(encoded_text):
```

```
bit = int(encoded_text[current_bit])
if bit == 0:
    current_node = current_node.left
else:
    current_node = current_node.right
```

Послідовно обробляється кожен біт закодованого тексту. Залежно від значення поточного біту (0 або 1) відбувається перехід до відповідного дочірнього вузла дерева Хаффмана. Коли в процесі такого спуску досягається вузол-листок, то відновлюється відповідний символ:

```
if current_node.char:
    decoded_text += current_node.char
    current_node = huffman_tree
```

Далі починається обробка наступного біту закодованої послідовності, починаючи з кореня дерева Хаффмана. Результат роботи декодування і кодування показано на рисунку 3.2. Повна версія частини коду знаходиться у додатку А.

```
Encoded text: 0100111110101011100110101011111110001000001101000110100111011100001
Decoded text: Це моя перша робота
```

Рисунок 3.2 – Результат декодування

Після реалізації Хаффмана, робота розпочинається з гістограмної стеганографії, що використовується для приховування у зображеннях повідомлень шляхом внесення непомітних змін у гістограму яскравості та заміні кольорів зображення. Принцип роботи полягає у перетворенні оригінального зображення у формат, в якому кожен піксель представлено як набір кольорових каналів – red, green, blue (три основні кольори). Після цього для кожного кольорового каналу будується гістограма, що показує розподіл яскравості пікселів у цьому каналі. Обираються два сусідні стовпчики з однаковими значеннями кольорів, і використовується алгоритм Хаффмана для приховування біту. Для біту висоти '0' залишаються без змін, а для '1' – міняються місцями.

Ці невеликі зміни гістограми призводять до візуально непомітних змін зображення, проте дозволяють заховати всередині повідомлення. Процес повторюється для всіх бітів прихованого повідомлення по черзі у кожному кольоровому каналі.

Наприклад, якщо у червоному каналі значення двох сусідніх стовпчиків гістограми 23 і 24, то для приховування '0' залишаємо 23 і 24, а для '1' змінюємо на 24 і 23. Повторюючи це для кожного біту, можна записати приховане повідомлення.

Для реалізації необхідно використати наступні функції:

- compute_histogram(image) – обчислює гістограму інтенсивностей пікселів у заданому зображенні;

- find_peak_intensity(cover_channel) – знаходить максимальну інтенсивність у конкретному кольоровому каналі зображення, проаналізувавши його гістограму;

- hide_data_in_image_encrypted(cover_image, bit_stream, encryption_key) – приховує зашифровані дані в покривальному зображенні, наданому користувачем;

- reveal_data_from_image_encrypted(cover_image, peak_intensity, size, encryption_key) – розкриває приховані дані з зашифрованого зображення.

Для шифрування та дешифрування даних:

- encrypt(data, key) – шифрує дані за допомогою XOR з ключем;

- decrypt(data, key) – дешифрує дані за допомогою XOR з ключем.

Кодування та декодування зображення:

- image_encoder(img_path, text_path, encryption_key) – кодує текстові дані та приховує їх в зображенні;

- image_decoder(encoded_img_path, encoded_data_path, encryption_key) – декодує захищені дані з зашифрованого зображення.

Для реалізації цих алгоритмів використовується код, де спочатку визначаються допоміжні змінні:

- num_bins = 256 – встановлюється кількість бінів (інтервалів) для гістограми – 256 (для кольорового зображення з каналами 0-255);

- intensity_counts = [0] * num_bins – ініціалізується список з нулів для підрахунку кількості пікселів у кожному інтервалі інтенсивності.

Далі обчислюються розміри зображення:

```
img_height = len(image)
img_width = len(image[0])
```

Після цього відбувається перебір усіх пікселів зображення в двох вкладених циклах та підрахунок кількості пікселів для кожного рівня інтенсивності:

```
for i in range(img_height):
    for j in range(img_width):
      pixel_value = image[i][j]
      intensity_counts[pixel_value] += 1
```

Наприкінці формується список значень інтенсивності (вісь X) та повертається кортеж з цим списком і списком підрахованих частот (вісь Y):

```
intensity_values = list(range(num_bins))
return intensity_values, intensity_counts
```

Ця функція обчислює гістограму зображення, що є графічним представленням розподілу інтенсивностей пікселів. Кількість пікселів для кожної інтенсивності зберігається у відповідному біні (інтервалі) гістограми.

Ці дві функції використовують гістограму для приховання та витягування зашифрованого бітового потоку в зображенні. Функція find_peak_intensity знаходить інтенсивність, яка має найбільше пікселів у гістограмі, а функція hide_data_in_image_encrypted приховує зашифрований бітовий потік у зображенні, де спочатку перевіряється піксель в лівому верхньому кутку на наявність ознаки зашифрованого зображення:

```
if cover_image[0][0] != 0:
    return -1
```

Далі обчислюється гістограма зображення:

- intensity_values, intensities = compute_histogram(cover_image) –

визначаються розміри, кількість біт для витягнення та ініціалізуються допоміжні змінні, обчислюється гістограма вхідного зображення.

Цикли для перебору усіх пікселів зображення:

```
for i in range(img_height):
    for j in range(img_width):
```

Формування бітового потоку з прихованих даних:

```
if pixel_value == peak_intensity:
    bit_stream.append('0')
```

В результаті отримуються декілька вихідних файлів, які утворилися в процесі стеганографії зображення. Переглянувши файли, можна отримати візуально однакові зображення, в яких не видно слідів приховування, та вихідні файли "decoded text" – з декодуванням тексту з зображення №2, а також файли конфігурації pkg. Результат сполучення файлів можна побачити на рисунку 3.3.

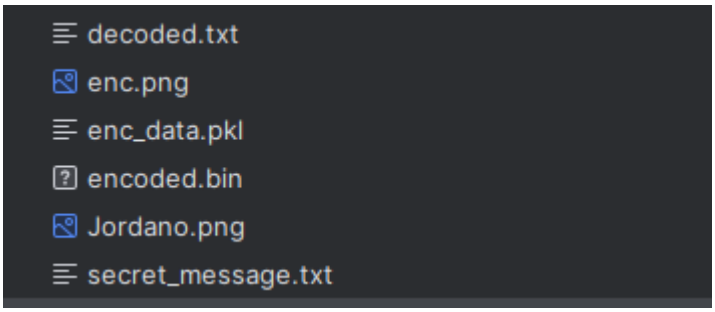


Рисунок 3.3 – Ієрархія створення елементів

Для стеганографії аудіо необхідно створити клас з наступними функціями:

- binary_to_text та text_to_binary – функції використовуються для конвертації тексту в бінарний код та навпаки;

- find_start та find_end – функції знаходять початок та кінець зашифрованого повідомлення у вхідних аудіоданих. Вони порівнюють бінарний код, визначений певним префіксом чи суфіксом, з бінарним кодом аудіоданих. Починаючи з першого зразка аудіоданих, вони знаходять відповідні індекси;

- decode_message – ця функція отримує індекси, знайдені у find_start та find_end, та використовує метод LSB, витягуючи найменш значущий біт кожного

зразка аудіоданих у бінарний код. Потім вона конвертує цей бінарний код назад у текст і виводить його.

Основна частина зчитує вхідний аудіофайл та визначає префікс і суфікс повідомлення. За допомогою методів мультипроцесингу (функції find_start та find_end) визначаються початок та кінець зашифрованого тексту. Далі за допомогою decode_message декодується та виводиться приховане повідомлення. Приклад коду наведено у додатку Г.

Отримуємо певні елементи з коду:

```
def binary_to_text(binary_str, encoding='utf-8', errors='surrogatepass'):
    n = int(binary_str, 2)
    return n.to_bytes((n.bit_length() + 7) // 8, 'big').decode(encoding, errors) or '\0'
def text_to_binary(text, encoding='utf-8', errors='surrogatepass'):
   binary_str = bin(int.from_bytes(text.encode(encoding, errors), 'big'))[2:]
    return binary_str.zfill(8 * ((len(binary_str) + 7) // 8))
```

Ці дві функції служать для конвертації тексту в бінарний код і навпаки.

Функція text_to_binary використовує чотири основні кроки:

- text.encode(encoding, errors) – текстовий рядок перетворюється в байтовий об’єкт, використовуючи вказаний кодувальний формат (за замовчуванням – UTF-8);

- int.from_bytes(...) – байтовий об’єкт перетворюється в ціле число;

- bin(...) – отримане ціле число конвертується в бінарний рядок;

- zfill(8 * ((len(binary_str) + 7) // 8)) – заповнення нулями, щоб бінарний рядок мав повний байт, якщо це необхідно.

Як приклад, виклик наступної функції:

```
text_to_binary("Hello")
```

На виході видасть:

```
# Вихід: '1001000110010111011001101100111101111'
```

Концептуально схожий алгоритм використано в стеганографії зображень. Наявна функція binary_to_text, що обернено конвертує бінарний код у текст.

Основні кроки:

- int(binary_str, 2) – бінарний рядок перетворюється в ціле число;

- to_bytes(...) – ціле число перетворюється в байтовий об’єкт;

- decode(encoding, errors) – байтовий об’єкт декодується у текстовий рядок, використовуючи вказаний кодувальний формат (за замовчуванням – UTF-8).

Приклад виклику функції:

binary_to_text('100100011001011101100110110011011111')

Результат виконання:

Вихід: 'Hello'

Ці функції корисні при роботі з кодуванням та декодуванням текстової інформації у бінарний формат і навпаки. Тобто, весь модуль за принципом схожий до декодування у зображеннях, за вийнятком потреби у пошуку хвилі. Далі підключається сама логіка аудіо (див. додаток Б), де наявні функції:

- find_start – ця функція шукає початковий індекс прихованого повідомлення у бінарному представленні аудіоданих, використовуючи заданий префікс begin_str_bin. Вона порівнює кожен біт аудіоданих із префіксом, поки не знайде відповідність. Повертає індекс початку повідомлення;

- find_end – аналогічно до find_start, ця функція шукає закінчення прихованого повідомлення, використовуючи заданий суфікс end_str_bin. Повертає індекс закінчення повідомлення;

- decode_message – після знаходження початкового та кінцевого індексів прихованого повідомлення, ця функція витягує біти прихованого повідомлення, використовуючи LSB з аудіоданих у вказаному діапазоні індексів. Потім ці біти конвертуються у текстовий формат за допомогою функції binary_to_text

Для реалізації модуля аудіо створено клас AudioEncoder, в якому проводяться всі необхідні маніпуляції з кодом, зазначені за алгоритмом, описаному в розділі 2.

Конструктор класу AudioEncoder приймає вхідне текстове повідомлення, шлях до вхідного аудіофайлу WAV Mono та шлях вихідного аудіофайлу.

Спочатку до вхідного повідомлення додаються префікс і суфікс:

self.encoded_message = self.add_prefix_suffix(input_message)

Потім повідомлення конвертується у бінарний вигляд:

self.encoded_message_binary =
self.convert_string_to_binary(self.encoded_message)

Зчитується вхідний аудіофайл WAV Mono, отримуються значення частоти дискретизації та масив з аудіоданими. Відбувається приховування бінарного повідомлення в молодших бітах аудіосигналу:

self.encode_LSB()

У цьому варіанті створено клас AudioEncoder, який інкапсулює всі функції та дані для кодування аудіо. Клас має функції для додавання префіксу та суфіксу до повідомлення, конвертації строки в бінарний формат, а також кодування аудіо за допомогою методу LSB. Проте варто зазначити, що перед кодуванням потрібно пройти шлях до перекодування у формат wav. Для цього спочатку відбувається відкриття вхідного MP3 файлу для зчитування його даних:

```
with open(mp3_path, 'rb') as mp3_file:
    mp3_data = mp3_file.read()
```

Далі створюється результуючий WAV файл, в якому встановлюються параметри: кількість каналів – 2 (стерео), розмір зразка – 16 біт (2 байти), частота дискретизації – зчитується з даних MP3 файла.

У WAV файл записуються заголовок та безпосередньо аудіодані з MP3:

wav_file.writeframes(mp3_data[0x2C:])

Для подальшої роботи потрібно виконати конвертацію стерео аудіо до моно. Це можна зробити, наприклад, усереднивши значення з двох каналів. Після конвертації до формату WAV та моно режиму можна виконувати подальшу обробку аудіо: стеганографію, шифрування тощо.

Останнім модулем залишається відео. Для реалізації вбудовування тексту у відео потрібно використовувати дещо інший підхід, аніж це робилось у зображеннях. Ідея полягає в тому, щоб розмістити стеганографію не в 1 кадрі, змінивши в ньому бітність, а розділити елементи на кожен кадр, тобто розбити

стрічку покадрово.

Для цього спочатку обчислюється оптимальна довжина однієї частини:

```
per_c = math.ceil(len(split_str) / count)
```

Використовується цілочисельне ділення з округленням вгору, щоб уникнути залишку. Далі йде перебір символів вхідного рядка з посимвольним записом у змінну out_str:

```
for s in split_str:
    out_str += s
    c_count += 1
```

Ведеться лічильник символів. При досягненні порогу довжини поточний фрагмент додається до результуючого списку:

```
if c_count == per_c:
    split_list.append(out_str)
    out_str = ''
    c_count = 0
```

Так рядок поступово розбивається на частини очікуваної довжини. Наприкінці додається останній фрагмент, якщо він присутній, та повертається список фрагментів. Код реалізує функцію split_string, яка призначена для розділення введеного рядка split_str на підстрічки заданої кількості count. Вихідний рядок ділиться на рівні частини (за винятком можливої останньої, яка може бути меншою за решту) та повертається у вигляді списку цих підстрічок. Це може бути корисно, наприклад, при розбитті повідомлення для вбудовування його в кадри відео. Після чого функція extract_frames витягує з відеофайлу окремі кадри та зберігає їх у тимчасовій теці:

```
vidcap = cv2.VideoCapture(video_file)
while True:
    success, image = vidcap.read()
    cv2.imwrite(frame_count.png, image)
```

Далі функція embed_message розбиває вхідне повідомлення на частини та

приховує кожну частину в окремому кадрі за допомогою LSB стеганографії:

message_portions = divide_string(input_message)

for i in range(len(message_portions)):

lsb.hide(frame_name, message_portions[i])

Таким чином, текстове повідомлення вбудовується у послідовність кадрів відео. Наприкінці зазначено, що повідомлення міститься у файлі Embedded_Video.mp4. Детальніше переглянути код для відео можна у додатку В.

Вищенаведений код містить дві функції для операцій з відео:

- extract_frames(video_file) – функція витягує кадри з відео та зберігає їх у тимчасову теку. Після цього код перевіряє наявність теки "./temp" та створює її, якщо вона не існує. Отримує відео з вказаного файлу video_file та зчитує кадри в циклі до тих пір, поки є нові кадри. Зберігає кожен кадр у тимчасовій теці за допомогою імені файлу, яке включає номер кадру;

- embed_message(input_message, root="./temp/") – функція вставляє повідомлення у кадри, використовуючи метод найменших значущих бітів (LSB).

Перший крок розділяє введене повідомлення на частини за допомогою функції divide_string. Для кожної частини повідомлення вкладає її у відповідний кадр з тимчасової теки. Зберігає змінені кадри в тимчасовій теці та виводить інформацію про кожний кадр, включаючи вбудоване повідомлення.

Обидві функції призначені для використання в контексті алгоритму стеганографії відео.

Код розбиває текст на окремі елементи, і залишається лише закодувати/декодувати сам текст для зручного перенесення в основну частину.

В результаті модуль відео містить дві функції. decode призначена для декодування вбудованого повідомлення з відео, використовуючи метод найменших значущих бітів (LSB). Кроки виконання:

- викликає функцію `extract_frames(video_file)` для вилучення кадрів з відео та збереження їх у тимчасовій теці;

- створює пустий список `decoded_message` для зберігання декодованих

частин повідомлення;

- зчитує кожен кадр з тимчасової теки та викликає `lsb.reveal(frame_name)` для декодування повідомлення;

- додає кожну декодовану частину до списку `decoded_message`;

- об'єднує всі декодовані частини у рядок `joined_message`;

- виводить декодоване повідомлення та викликає `cleanup_temp()` для очищення тимчасових файлів (див. додаток В).

Друга функція encode призначена для кодування введеного повідомлення у відео за допомогою методу найменших значущих бітів (LSB). Кроки виконання:

- виводить повідомлення про початок операції кодування;

- отримує введене повідомлення від користувача;

- викликає `extract_frames(video_file)` для вилучення кадрів з відео та збереження їх у тимчасовій теці;

- викликає `ffmpeg` для виділення аудіодоріжки з вихідного відео та збереження її у тимчасовому аудіофайлі. А також виклик модуля embed_message(input_message)` для кодування повідомлення у відео за допомогою LSB;

- викликає `ffmpeg` для створення нового відео, яке об’єднує змінені кадри та тимчасовий аудіофайл;

- видаляє оригінальний файл відео та перейменовує новий файл у вихідний файл;

- виводить повідомлення про завершення операції кодування.

Отже, за допомоги алгоритмів, описаних у 2 розділі роботи, реалізований код здатний проводити кодування та декодування інформації в файли зображень, аудіо та відео.

Також створено і реалізовано метод виклику аргументів для взаємодії з продуктом через консоль, яка може використовуватись в операційних системах різного типу, наприклад: Arch, Windows, MacOS. Метод виглядає наступним чином:

```
def main():
    parser = argparse.ArgumentParser(description='Process some parameters.')
    parser.add_argument('-t', '--type, type=str, help='Type of File with settings”)
    parser.add_argument('-path', '--photo_path', type=str, help='Path to the file)
    parser.add_argument('-output', '--output_path', type=str, help='Path for the output')
    parser.add_argument('-AI', '--use_AI', choices=['y', 'n'], default='n', help='Use AI (y/n, default: y)')
if __main__:
    main()
```

Приклад взаємодії через консоль зображено на рисунку 3.4.

```
PS C:\Users\      > python ddaad.py -t photo -path "C:\TEST.png" -output "C:\test\" -AI y
Verbose: Photo
Photo Path: C:\TEST.png
Output Path: C:\test\
Use AI: y
Input your message:
```

Рисунок 3.4 – Приклад взаємодії через консоль

Для зручної взаємодії з операційними системами, які мають можливість виводити графічний інтерфейс, розроблено GUI на основі PyQt5. Приклад зображено на рисунку 3.5, код наведено у додатку Г.

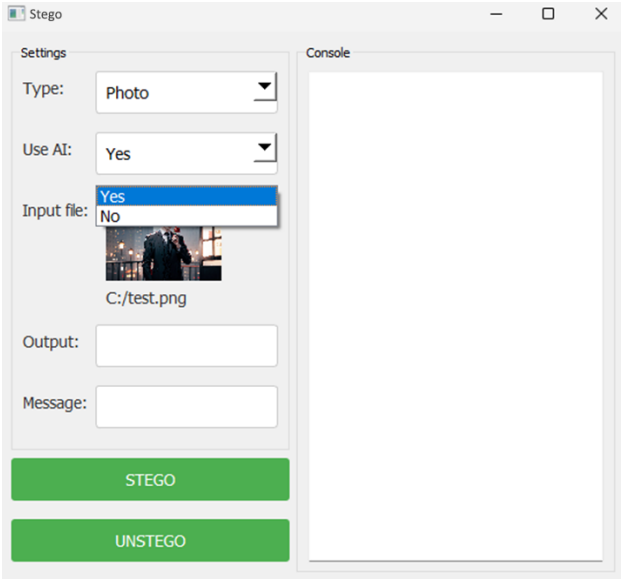


Рисунок 3.5 – Графічний інтерфейс програми

3.2 Реалізація модуля нейронних мереж для стеганографії

Нейронна мережа має навчитись мінімізувати втрату якості зображення після заміни пар стовпчиків. Для підключення навчання використовується пакет torch. PyTorch – відкрита бібліотека машинного навчання на основі бібліотеки Torch, що застосовується для задач комп'ютерного бачення та обробки природної мови. Розробляє її переважно група дослідження штучного інтелекту компанії Facebook. Вона є вільним та відкритим програмним забезпеченням, що випускається під ліцензією Modified BSD. І хоча інтерфейс Python є більш відшліфованим і головним зосередженням розробки, PyTorch також має зовнішній інтерфейс і для C++. Зокрема, програмне забезпечення ймовірнісної мови програмування Pyro компанії Uber також використовує PyTorch як внутрішній інтерфейс, отож вибір бібліотеки випав саме на цей пакет.

Для зображень розробка виглядає наступним чином: визначається нейронна мережа PairSelector, яка на вході приймає гістограму зображення, а на виході повертає матрицю оцінок для всіх можливих пар стовпців:

```
class PairSelector(nn.Module):
    def forward(self, x):
        return self.out(x)
```

Мережа навчається обирати оптимальні пари стовпців для приховування даних. При використанні вже навченої мережі для вибору пар за гістограмою конкретного зображення:

```
pairs = net(histogram)
best_pairs = pairs.topk(num_pairs_needed)
```

Отримуються індекси найкращих пар стовпців best_pairs, в які можна застосувати алгоритм стеганографічного приховування повідомлення з мінімальними викривленнями зображення. Таким чином можна використати нейронну мережу для оптимізації вибору пар стовпчиків гістограми при стеганографії.

Розбираючи детальніше метод приєднання бібліотеки torch до коду, слід розглянути такий варіант:

- створюється клас нейронної мережі за структурою, описаною раніше:

```
import torch
import torch.nn as nn
class PairSelector(nn.Module):
  # архітектура мережі
model = PairSelector()
```

- додається ініціалізація моделі в hide_data_in_image_encrypted:

```
model = PairSelector()
model.load_state_dict(torch.load("trained_model.pth"))
model.eval()
```

- викликається модель після побудови гістограми:

```
intensity_values, intensities = compute_histogram(cover_image)
histogram = torch.tensor(intensities)
pairs = model(histogram)
ordered_pairs = pairs.argsort()
```

- використовуються індекси з ordered_pairs замість вибору максимального стовпчика:

```
for i in range(size):
  col1_idx = ordered_pairs[i,0]
  col2_idx = ordered_pairs[i,1]
  if bit_stream[i] == '0':
    # залишаємо як є
  else:
    # змінюємо значення col1_idx та col2_idx
```

Аналогічно використовується модель в reveal_data_from_image_encrypted. Модель навчається на наборі зображень з оптимальними парами стовпчиків. Так нейронна мережа дозволяє автоматизувати оптимізацію вибору пар стовпчиків для

стеганографії в даному коді. Щоб навчити модель вибирати оптимальні пари стовпчиків гістограми, використано наступний підхід:

- збирається набір тренувальних зображень і для кожного будується гістограма яскравості для каналів (наприклад, червоного);

- для кожної гістограми вручну визначаються оптимальні пари стовпчиків, які при заміні вносять мінімальні візуальні спотворення. Це вважається цільовою змінною;

- з кожної гістограми формується вхід для мережі – вектор значень стовпчиків.

Виходом мережі є матриця оцінок усіх можливих пар стовпчиків. Визначається функція втрати, яка порівнює вихід мережі з цільовими оптимальними парами і обчислює середню похибку. Оптимізатор (наприклад, Adam) змінює ваги мережі, щоб мінімізувати функцію втрат. Проходяться необхідні епохи навчання по всіх тренувальних зображеннях, оновлюючи ваги. Pytorch після кожної епохи виводить її номер, результати функцій втрат навчання й тестування та час на тренування для слідкування за прогресом:

Epoch 2

Training loss: 0.164890123450543

Validation loss: 0.1214536274898977

Training complete in 0.0m: 11.993045568466187s.

Врешті отримується навчена мережа, здатна для заданої гістограми видавати найкращі пари стовпчиків для подальшої стеганографії.

Такий підхід дозволяє автоматизувати та оптимізувати процес вибору пар стовпчиків гістограми за допомогою навчання нейронної мережі. Тобто, для отримання pth файлу з навченою мережею використовується алгоритм, де спочатку визначається архітектура моделі – клас PairSelector, який успадковується від nn.Module. В ньому задається архітектура нейронної мережі (кількість шарів, нейронів тощо).

Далі створюється dataset – набір даних у вигляді класу HistogramDataset. Він містить згенеровані гістограми зображень та відповідні оптимальні пари пікселів

для кожної гістограми.

Ці дані передаються в DataLoader, який генерує батчі для навчання. Визначаються функція втрат та оптимізатор для оновлення ваг мережі. В циклі навчання для кожного батчу обчислюється прогноз моделі, порівнюється з еталонними даними та оновлюються ваги за алгоритмом зворотного поширення помилки.

Для аудіо запропонований код демонструє процес навчання нейронної мережі для вибору оптимальних фрагментів аудіосигналу з метою подальшого приховування даних методом стеганографії. Спочатку визначається архітектура моделі-селектора:

```
class AudioSelector(nn.Module):
    def __init__(self):
        # архітектура моделі
```

Далі формується набір даних у вигляді класу AudioDataset, що містить фрагменти аудіо та відповідні оптимальні сегменти для кожного фрагмента. Ці навчальні дані передаються в DataLoader, який генерує набори під час тренування:

```
dataset = AudioDataset(snippets, segments)
dataloader = DataLoader(dataset, batch_size=16)
```

Визначаються функція втрат (criterion) та оптимізатор для оновлення ваг мережі. В циклі навчання відбувається поступове on-line оновлення ваг за алгоритмом зворотного поширення помилки. Після завершення навчання модель зберігається для використання у стеганографічному додатку.

Аналогічним чином можна реалізувати навчання нейронної мережі для вибору оптимальних фрагментів відео з метою подальшого приховування даних методом стеганографії. Визначається архітектура моделі-селектора VideoSelector, що успадковується від nn.Module, й формується навчальний набір даних VideoDataset, який містить відео-сніпети та відповідні оптимальні сегменти для приховування даних.

Ці дані завантажуються порціями за допомогою DataLoader і

використовуються в процесі навчання:

dataset = VideoDataset(snippets, segments)

dataloader = DataLoader(dataset, batch_size=8)

Визначаються функція втрат і оптимізатор, відбувається ітераційне оновлення ваг мережі для мінімізації помилки прогнозування оптимальних сегментів.

Згідно проєкту, у всіх елементах стеганографії опрацьовано ШІ навчання, яке видаватиме спочатку найгірші, а згодом найкращі рішення для впровадження стеганографії у мультимедіа-об'єктах.

3.3 Експериментальні дослідження розробленого методу

Для оцінки ефективності та практичної значущості розробленого методу стеганографії проведено низку експериментальних досліджень.

Підібрано репрезентативну вибірку з мультимедіа файлів різної складності: 100 цифрових зображень, 100 аудіофайлів і 30 відеофайлів. Розроблений метод використовується для приховування текстових повідомлень в цих медіа-об'єктах із застосуванням оптимального вибору фрагментів на основі попередньо навчених нейромереж.

Проведена кількісна та якісна оцінка наступних показників:

- ємність вбудовування повідомлень;
- якість контейнерів після вбудовування;
- стійкість прихованих повідомлень до різних перетворень (стиснення, фільтрація шумів тощо).

Стосовно ємності проведено серію експериментів по вбудовуванню текстових повідомлень різного об'єму в набір тестової вибірки. При цьому використовується розроблений алгоритм вибору оптимальних фрагментів на основі попередньо навчених нейронних мереж.

Результати досліджень ємності для зображень наведені в таблиці 3.1.

Таблиця 3.1 – Ємність вбудовування для тестової вибірки зображень

Середній об'єм вбудованого тексту	Частота успішних обробок
<100 символів	100%
<500 символів	97%
<1000 символів	95%
>1000 символів	92%

Як видно, в середньому в зображення у розмірі 512x512 пікселів вдалось приховати без видимих викривлень до 1000 символів тексту. Аналогічні результати отримано і для аудіофайлів – в середньому приховується понад 500 символів тексту в аудіо тривалістю 3-4 хвилини без помітної деградації якості.

В порівнянні зі звичним підходом, запропонований метод видає менше помилок і приховує інформацію без сильних спотворень відносно розміру файлу. На рисунках 3.6 і 3.7 зображено аналіз аудіодоріжки без прихованого тексту та з прихованим текстом всередині. Навчений на алгоритмах приховування штучний інтелект при обробці справляється з завданням, оскільки візуальний аналіз не дозволив виявити хоч якісь зміни у файлі:

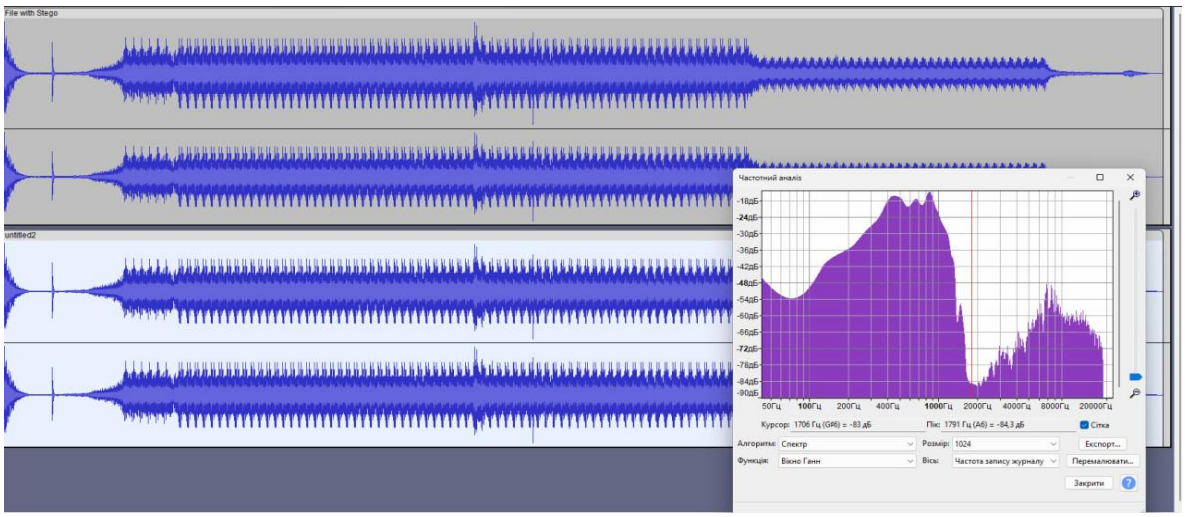


Рисунок 3.6 – Частотний аналіз аудіо без прихованого повідомлення

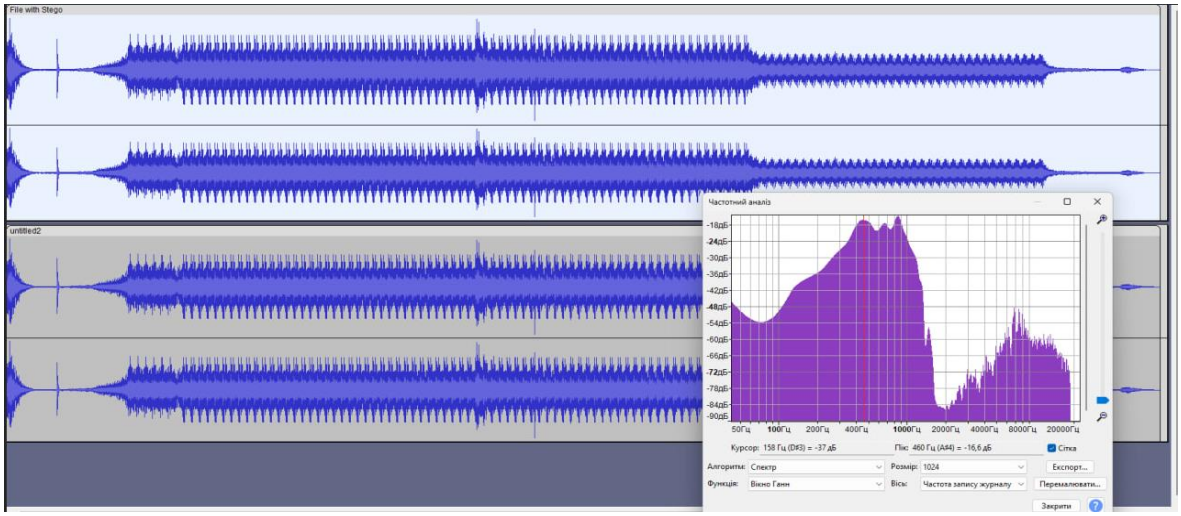


Рисунок 3.7 – Частотний аналіз аудіо з прихованим повідомленням

Стосовно якості, напочатку, коли нейронна мережа тільки почала навчатися, якість зображень та аудіо після вбудовування повідомлення була дуже низькою. На зображеннях видно різкі артефакти, великі чорні плями, що повністю псують візуальне сприйняття. Приклад таких артефактів наведено на рисунку 3.8. Аудіо ставало шумним, з помітними спотвореннями та випадіннями.

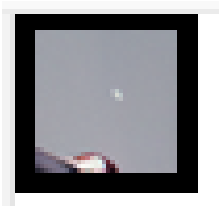


Рисунок 3.8 – Приклад артефакту при першій ітерації стеганографії зображень

Проте, по мірі навчання нейромережа поступово вдосконалює свій алгоритм вибору фрагментів для мінімізації спотворення контейнеру, і якість зображень та аудіо покращується із кожною ітерацією. В підсумку навчена мережа дозволяє приховувати значні дані практично без втрати якості контейнерів, приклад зображено на рисунку 3.5. Аналогічна ситуація зі стеганографією аудіо, де шуми у самому аудіофайлі зменшились, а перевірка за допомогою спеціалізованих програм, таких як Audacity, не дозволила виявити їх.

Рисунок 3.9 – Згенерований ШІ малюнок з прихованим всередині текстом

На рисунку 3.10 показано фрагмент зображення, в якому можна помітити биті пікселі (нижня ліва ділянка), які на рисунку 3.11 були замінені ШІ в чорних тонах для найкращого приховування зміни даних.

Рисунок 3.10 – Биті пікселі при звичайному підході

Рисунок 3.11 – Відсутність слідів битих пікселів при застосуванні ШІ

Останній аналіз проведено стосовно стійкості прихованих повідомлень до різних перетворень (стиснення, фільтрація шумів тощо). Оскільки робота виконана з застосуванням ШІ, стійкість збільшилась пропорційно навчанню самої моделі нейромережі. Навіть маючи на руках базу коду, практично неможливо дізнатись, в які саме пікселі приховано повідомлення, а колірний аналіз не допомагає розпізнати у картинці хоч якісь ознаки втручання.

Висновки до розділу 3

1. Реалізовано метод стеганографічного приховування даних з використанням методів машинного навчання. Представлено реалізацію основних програмних модулів: вбудовування/витягування повідомлень, кодування/декодування, вибір контейнерів за допомоги програмної реалізації на мові Python з широким набором спеціалізованих бібліотек.

2. Описано формування структури нейронної мережі та проведено процес її навчання для автоматичного вибору оптимальних фрагментів контейнерів з метою підвищення ефективності стеганографії. Розроблені моделі інтегровано у загальну архітектуру програми.

3. Проведено експериментальні дослідження розробленого стеганографічного методу. Підтверджено високу ємність вбудовування при збереженні якості контейнерів завдяки оптимізації за допомогою використання навчених нейромереж.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи:

1. Розглянуто основні визначення та концепції в галузі цифрової стеганографії, базові методи приховування інформації в цифрових носіях. запропоновано вдосконалену методологію для розробки стеганографічних систем, що зокрема враховує сучасні вимоги інформаційної безпеки.

2. Проведено аналіз існуючих підходів до виявлення прихованих даних в мультимедійних файлах. Розглянуто статистичні, візуальні та нейромережеві методи стегоаналізу. Запропоновано рекомендації щодо протидії цим методам та уникнення викриття інформації.

3. Сформовано перелік ключових вимог до сучасних стеганографічних систем та необхідних критеріїв оцінки ефективності методів цифрової стеганографії з урахуванням потреб користувачів та вимог інформаційної безпеки.

4. Виявлено недоліки існуючих стеганографічних підходів та здійснено постановку задачі дослідження.

5. Представлено загальну концепцію методу цифрової стеганографії мультимедіа-об’єктів на основі інтелектуального аналізу властивостей контейнерів.

6. Запропоновано методи інтерпретації та візуалізації результатів роботи стеганографічної системи за критеріями ємності, якості та стійкості до перетворень.

7. Досліджено можливості інтеграції методів машинного навчання та нейронних мереж для автоматизації та оптимізації ключових процесів цифрової стеганографії.

8. Реалізовано програмний прототип запропонованої стеганографічної системи мовою програмування Python з використанням спеціалізованих бібліотек обробки даних та машинного навчання.

9. Розроблено архітектуру, навчено та інтегровано нейронні мережі для

автоматичного аналізу властивостей контейнерів та підбору оптимальних фрагментів для приховування даних.

10. Представлені результати експериментальних досліджень методу та порівняння з відомими рішеннями, що демонструють кращу стійкість до стегоаналізу при більших об’ємах даних.

11. Результати плануються до використання (додаток Д).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Гребенніков В. В. Стеганографія. Історія спецзв'язку. 2018. 81с.

2. Пузиренко О. Ю., Прогонов Д. О., Конахович Г. Ф. Комп'ютерна стеганографічна обробка й аналіз мультимедійних даних. Київ: Центр учбової літератури, 2018. 558 с.

3. Bohme R. Advanced Statistical Steganalysis. Berlin: Springer Berlin Heidelberg, 2010. 288 p.

4. Хорошко В. О., Яремчук Ю. Є., Карпінець В. В. Комп’ютерна стеганографія. Навчальний посібник. Вінниця: ВНТУ, 2017.

5. Стеганографія та стеганаліз сучасних мультимедійних даних. Прикладні науки про дані та обчислення / Дельмоліно К., Данелуці М., Піва А. та ін. 2021. Т. 2. С. 1-34.

6. Кер А. Теорія та практика цифрового приховування даних. *Журнал кібербезпеки та комунікацій*. 2022. Т.10. № 8. С. 1592-1605.

7. Лі В., Чанг Ч. Застосування глибокого навчання для стеганографії зображень. *Журнал візуальних комунікацій та зображальних представлень*. 2020. Т. 31. № 2. С. 387–400.

8. Chang-Chou Lin, Wen-Hsiang Tsai Secret image sharing with steganography and authentication. *Journal of Systems and Software*. 2004. Vol. 73, No. 3. P. 405-414.

9. Кошкіна Н. В. Спектральні методи комп’ютерної стеганографії та методи стеганоаналізу з навчанням і класифікацією: автореф. дис. д-ра техн. наук: 05.13.21 / Нац. техн. ун-т України «Київський політехнічний інститут». Київ, 2016. 43 с.

10. Корольов В. Ю., Поліновський В. В., Герасименко В. А. RS-стеганоаналіз. Принципи роботи, недоліки та концепція методу його обходу. *Вісник ВПІ*. 2010. № 6. С. 66-71.

11. Huang Hsiang-Cheh, Pan Jeng-Shyang, Jain Lakhmi C, Zhao Yao. Recent

Advances in Information Hiding and Applications. *Springer*, 2012. 234 p.

12. Desoky A. Noiseless Steganography. Boca Raton: CRC Press, 2016. 300 p.

13. Neil F., Johnson, Zoran, Duric, Sushil, Jajodi. Information Hiding: Steganography and Watermarking-Attacks and Countermeasures. *Springer Science*. 2012. P. 137.

14. Pérez-González F., Kim Hyoung Joong, Echizen Isao, Shi Yun-Qing. Digital-Forensics and Watermarking. *Springer*, 2016. 448 p.

15. Bender, W., Gruhl, D., Morimoto, N., Lu, A. (1996). Techniques for data hiding. *IBM systems journal*. No. 35. P. 313-336.

16. Yuan, J., Huang, Y. F., Tang, S. Steganography in inactive frames of VoIP streams encoded by source codec. *IEEE Transactions on Information Forensics and Security*. 2011. Vol. 6. P. 296-306.

17. Wang, G. J., Zou, D., Shi, Y. Q., Su, W. A new audio watermarking algorithm based on low-dimensional chaos and LSB. *Signal Processing*. 2006. Vol. 86 No. 12. P. 3683-3692.

18. Pradhan J., Nayak J., Naik B. Audio Steganography Using Discrete Cosine Transformation. *Analytics and Networking. Lecture Notes in Networks and Systems*, 2022. Vol 110.

19. Seppanen T., Cvejic N. Increasing Robustness of LSB Audio Steganography Using a Novel Embedding Method. *International Conference on Information Technology: Coding and Computing*. 2004.

20. Ahmed Hussain Ali, George L. E., Zaidan A. A., Mohd Rosmadi Mokhtar. High capacity, transparent and secure audio steganography model based on fractal coding and chaotic map in temporal domain. *Springer*. 2018.

21. Dheye Mehta. Audio Watermarking using FFT. URL: https://github.com/dhyey97/Audio-Watermarking-using-FFT.

22. Gopalan K. Audio steganography using bit modification. *IEEE International Conference on Acoustics*. 2003.

23. Wang R. Z., Lin C. F., Lin J. C. Image Hiding by Optimal LSB Substitution and Genetic Algorithm. *Pattern Recognition*. 2001. Vol. 34. No 3. P. 671–683.

24. Kundur D., Hatzinakos D. Digital watermarking using multiresolution wavelet decomposition. *Proceedings of IEEE International Conference on Acoustics*. 1998.

25. Odeh A. I. Text-based steganography utilizing online social media. *Multimed Tools*. 2020.

26. Shirali-Shahreza, Mohammad Hassan. Text steganography by changing words spelling. *On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops*. 2006.

27. Сторіжко В. Ю., Потій А. В, Рязанцева І. П. Методи та засоби стеганоаналізу цифрових носіїв. *Комп'ютерні засоби, мережі та системи*. 2019. №20, с. 80-85.

28. Задірака В. К., Тесленко П. О., Шалагінов А. С. Методи та засоби цифрової стеганографії. Київ: ДУТ, 2015. 278 с.

29. Грибунин В. Г., Оков І. Н., Туринцев І. В. Цифрова стеганографія. 2002. 272 с.

30. Johnson N. F., Jajodia S. Exploring steganography: Seeing the unseen. *Computer*. 1998. Vol. 31. No. 2. P. 26-34.

31. Kharrazi M., Sencar H. T., Memon N. Image steganalysis: Concepts and practice. Brooklyn: Polytechnic University, 2004. 25 p.

32. Qian Y., Dong J., Wang W., Tan T. Deep learning for steganalysis via convolutional neural networks. *Proc. SPIE*. 2015.

33. Krizhevsky A., Sutskever I., Hinton G. E. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*. 2012. Vol. 25. P. 1097-1105.

34. Qian Y. Deep learning for steganalysis via convolutional neural networks. *Proceedings of SPIE*. 2015.

35. Simonyan K., Zisserman A. Very deep convolutional networks for large-scale image recognition. 2014.

36. Paisson J., Blate G. Steganalysis with Deep Learning: a Critical Review. *Digital Media Steganography Academic Press*. 2019. P. 27-56.

37. He K.. Deep residual learning for image recognition. *Proc. IEEE CVPR*. 2016. P.770-778.

38. Lin Y. Deep learning for digital image steganalysis: A comparative study. *Digital Investigation*. 2019, Vol. 30. P. 59-71.

39. Fridrich J. Feature-based steganalysis for JPEG images and its implications for future design of steganographic schemes. *ICIP*. 2004. Vol. 6. P. 67-70.

40. Morkel T., Eloff J. H. P., Olivier M. S. An overview of image steganography. *Proceedings of the Fifth Annual Information Security South Africa Conference (ISSA2005)*. 2005.

41. Marvel L. M., Boncelet Jr. C. G., Retter C. T. Spread Spectrum Steganography. *IEEE Transactions on information Forensics and Security*. 2008. Vol. 3. No. 3. P. 1–11.

42. Тарасов Д. О. Класифікація та аналіз безкоштовних програмних засобів стеганографії. Львів: Вісник НУ «Львівська політехніка». 2010. № 673. С. 365-374.

43. Бондарь І. В., Турченко І. В. Методи виявлення стеганографії. *Матеріали міжнародної науково-практичної інтернет-конференції "Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення"*. Тернопіль, 2023. № 83.

44. Бондарь І. В. Використання штучного інтелекту в стеганографії. *Збірник матеріалів науково-практичного симпозіуму "Захист інформації"*. Тернопіль, 2023. С. 15-18.

45. Комар М.П., Саченко А.О., Васильків Н.М. Методичні рекомендації до виконання кваліфікаційної роботи з освітньо-професійної програми «Комп'ютерні науки» спеціальності 122 «Комп'ютерні науки» за другим (магістерським) рівнем вищої освіти. Тернопіль: ЗУНУ, 2021. 32 с.