МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
КАФЕДРА КОМП'ЮТЕРИЗОВАНИХ СИСТЕМ ЗАХИСТУ ІНФОРМАЦІЇ

ДОПУСТИТИ ДО ЗАХИСТУ

Завідувач кафедри Комп'ютеризованих систем захисту інформації

_____ Михайло СТЕПАНОВ

«____»_____ 2023 р.

На правах рукопису
УДК 004.415.5:007.52

КВАЛІФІКАЦІЙНА РОБОТА

ЗДОБУВАЧА ВИЩОЇ ОСВІТИ ОСВІТНЬОГО СТУПЕНЯ «МАГІСТР»

Тема: Програмний засіб стегоаналізу повідомлень

Виконавець: Ігор БОРИСЕВИЧ

Науковий керівник: к.т.н., доцент Андрій ПЕТРЕНКО

Консультант розділу «Охорона навколишнього середовища»: к.т.н., доцент Тетяна ДМИТРУХА

Нормоконтролер: к.т.н., доцент Андрій ПЕТРЕНКО

Київ 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет: Кібербезпеки та програмної інженерії

Кафедра: Комп'ютеризованих систем захисту інформації

Освітній ступінь: «Магістр»

Спеціальність: 125 «Кібербезпека»

Освітньо-професійна програма: «Безпека інформаційно-комунікаційних систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри Комп'ютеризованих систем захисту інформації

_____ Михайло СТЕПАНОВ
«__» _____ 2023 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи
здобувача вищої освіти Борисевича Ігоря Олексійовича

1. Тема: *Програмний засіб стегоаналізу повідомлень*
затверджена наказом ректора від «15» вересня 2023 р. № 1814/ст.
2. Термін виконання: з 16.10.2023 р. по 31.12.2023 р.
3. Вихідні дані: Аналіз сучасних методів стегоаналізу повідомлень; на основі аналізу вивести кінцевий набір даних для навчання нейронної мережі; розробка моделі нейронної мережі з навчанням без вчителя.
4. Зміст пояснювальної: Огляд стеганографії та стегоаналізу, як надійний спосіб захисту повідомлень і аналіз найпоширеніших атак; проведення аналізу бібліотек Python для штучного інтелекту; створення програмного модулю для перевірки зображення на наявність прихованої інформації.

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної роботи

№ п/п	Етапи виконання дипломної роботи	Термін виконання етапів	Примітка
1.	Уточнення постановки завдання	16.10.2023	*Виконано*
2.	Аналіз літературних джерел	20.10.2023	*Виконано*
3.	Обґрунтування рішення	24.11.2023	*Виконано*
4.	Збір інформації	26.11.2023	*Виконано*
5.	Аналіз найпоширеніших атак, методики зловмисників	10.11.2023	*Виконано*
6.	Дослідження бібліотек Python для роботи зі штучним інтелектом, їх подальше намагання інтеграції в проект	16.11.2023	*Виконано*
7.	Розробка навчального датасету, навчання нейронної мережі, створення віконного додатку	22.11.2023	*Виконано*
8.	Тестування створеного ПЗ.	24.11.2023	*Виконано*
9.	Перевірка на антиплагіат	12.12.2023	*Виконано*
10.	Оформлення і друк пояснювальної записки	13.12.2023	*Виконано*
11.	Оформлення презентації	14.12.2023	*Виконано*
12.	Отримання рецензій від рецензента	20.12.2023	*Виконано*

Консультанти з окремих розділів

Розділ	Консультант (посада, П.І.Б.)	Дата, підпис	
		Завдання видав	Завдання прийняв
Охорона навколишнього середовища	Дмитруха Т.І.		

7. Дата видачі завдання: «16» жовтня 2023 р.

Здобувач вищої освіти Ігор Борисевич
(підпис, дата)

Керівник кваліфікаційної роботи Андрій Петренко
(підпис, дата)

РЕФЕРАТ

Кваліфікаційна робота складається зі вступу, чотирьох розділів, загальних висновків, списку використаних джерел, додатків і має 83 сторінки основного тексту, 24 рисунки, 4 таблиці, 12 сторінок додатків. Список використаних джерел містить 42 найменування і займає 4 сторінки. Загальний обсяг роботи 105 сторінок.

Метою роботи є підвищення рівня захищеності комп'ютерних систем за рахунок аналізу атак за допомогою прихованих повідомлень, навести перелік найвідоміших інцидентів, визначити всі необхідні аспекти стеганографії.

В роботі вирішено задачу розробки чітких методики та алгоритми для виявлення прихованих повідомлень в зображеннях, які можуть містити зловмисний код та інші шкідливі повідомлення.

В роботі розроблено програмне забезпечення для стегоаналізу зображень з метою виявлення прихованих повідомлень для безпеки користувача.

Розроблене програмне забезпечення відносяться до галузі інформаційної безпеки і може бути використані для захищеності та свідомості користувачів у корпоративній мережі.

Можливі напрямки розвитку цієї роботи пов'язані із розширенням моделі і алгоритму програмного забезпечення відповідно до вимог міжнародних стандартів, наприклад ISO 27001, для більш повного аналізу та оцінки ризиків.

Ключові слова: стеганографія, стегоаналіз, контейнер, найменший значущий біт, заміна палітри, машинне навчання.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

LSB – Least Signifacant Bit

ПЗ – програмне забезпечення

ПВЧ – псевдовипадкові числа

RGB – Red, Green, Blue

BMP – BitMap Picture

PNG – Portable Network Graphics

JPEG – Joint Photographic Experts Group

API – Application Programming Interface

GUI – Graphical User Interface

ВСТУП

Актуальність. У світі швидко розвиваючихся технологій та динамічних цифрових середовищ стеганографія, мистецтво приховування інформації в безпечних видимих носіях, стає не лише інтригуючим, але й потенційно небезпечним знаряддям для передачі конфіденційної інформації. У цьому контексті стегоаналіз, наука про виявлення та аналіз прихованих даних, відіграє ключову роль у забезпеченні цифрової безпеки та запобіганні невпізнаним атакам.

Стеганографія - це наука про те, як приховувати інформацію в інших даних так, щоб ця фактура була практично невидимою для сторонніх спостерігачів. У порівнянні з традиційним шифруванням, де зміст повідомлення зазвичай перетворюється в іншу форму або кодується за певним алгоритмом, стеганографія використовує хитрий метод вбудовування інформації в невинні дані, такі як текст, зображення або аудіофайли. Основна ідея полягає в тому, що ніхто, крім одержувача, не має навіть здогадуватися про те, що відбувається прихована комунікація.

Стеганографія існує вже багато століть і використовується в різних сферах, включаючи військовий зв'язок, розвідку, криміналістику, інформаційну безпеку та багато інших. За допомогою цієї технології можна передавати та зберігати секретну інформацію в найневинніших контекстах, що робить її особливо цінною в областях, де конфіденційність даних є критично важливою.

Заходження у цю область дозволить висвітлити не лише поточний стан, але й передбачити майбутні напрями розвитку стегоаналізу та засобів захисту від стеганографії в епоху постійної цифрової еволюції.

Метою роботи є розробка програмного засобу для стегоаналізу зображень з використанням методу машинного навчання.

Виходячи з мети **завданням** для даної дипломної роботи є:

1. Провести аналіз поняття стегоаналізу, його методів та ролі в кібербезпеці, розглянути існуючі інструменти для стегоаналізу.

2. Провести аналіз машинного навчання в аналізі зображень, його методів, бібліотек.

3. Розробка алгоритма та програмного засобу та його тестування для виявлення прихованного повідомлення в зображеннях методом машинного навчання.

Об’єктом дослідження є визначення зображення, яке містить приховану інформацію.

Предметом дослідження є методи та технічні засоби, які можна використовувати для автоматичного виявлення прихованих повідомлень в зображеннях.

Методами дослідження дипломної роботи є:

1. Проведення статистичного аналізу реальних випадків атак з використанням стеганографії.

2. Дослідження та аналіз історії атак на користувачів та підприємства.

3. Реалізація використання методів обробки зображень для аналізу ознак з метою виявлення прихованої інформації.

Наукова новизна. Розроблений алгоритм стегоаналізу зображень для виявлення прихованої інформації, що поєднує методи обробки зображень з машинним навчанням для точного розпізнавання підозрілих ознак.

Практична цінність полягає у розробці програмного засобу який здійснює стегоаналіз зображення з використанням технологій: Python, Machine Learning. Тестування розробленого коду програмного засобу показало стійкість розробленого методу до різних поставлених задач.

РОЗДІЛ 1. ВИЗНАЧЕННЯ СТЕГАНОГРАФІЇ ТА СТЕГОАНАЛІЗУ

1.1. Поняття стеганографії

У запропонованій роботі зображення використовується як стеганографічний носій, який можна класифікувати як з втратами та без втрат. Стиснення з втратами забезпечує значний рівень стиснення і, отже, економить більше місця. Однак це може призвести до загального коригування частин і вплинути на винахідливість зображення. Тоді як стиснення без втрат точно відтворює повідомлення. Ми реалізували наш алгоритм із форматами стиснення зображень без втрат, результати показали задовільний рівень надійності. У запропонованій нами схемі ми використали просторову область. У випадку просторового фокусування базується на значеннях пікселів зображення.

Основна мета стеганографії - приховати наявність вбудованої інформації в сигналі так, щоб це було непомітно для сторонніх аналізаторів чи спостерігачів.

Спотворення в стеганографії може виникати в результаті:

- Малопомітних змін в сигналі: Сучасні методи стеганографії, особливо ті, які використовують LSB (Least Significant Bit) в зображеннях чи аудіо, можуть вбудовувати інформацію шляхом невеликих змін в менш значущих бітах сигналу. Ці зміни повинні бути досить непомітними, щоб не спотворити візуальний або аудіо сигнал для спостерігача.
- Внесення артефактів: Деякі методи стеганографії можуть призводити до внесення артефактів чи шуму в сигнал. Важливо, щоб ці артефакти були мінімальними та не порушували сприйняття сигналу.
- Зміни статистичних властивостей: Додавання чи видалення інформації може змінювати статистичні властивості сигналу, такі як середнє значення, дисперсія чи інші характеристики. Деякі методи стеганографії стараються зберегти ці статистичні властивості.
- Вплив на якість сигналу: Вбудовання інформації може впливати на якість сигналу, особливо якщо додається багато бітів для великих обсягів

інформації. Важливо зберігати баланс між прихованою інформацією та збереженням якості сигналу.

Для успішної стеганографії важливо досягти такої міри спотворення, яка робить вбудовану інформацію максимально непомітною для сторонніх. Розробка методів стеганографії, які забезпечують ефективність та невидимість, є одним із завдань в цій галузі.

Рис.1.1. Вбудовування в 8-бітне зображення

Зловмисники можуть використовувати стеганографію з різних мотивів та для різних цілей.

Однією з основних цілей використання стеганографії є забезпечення конфіденційності комунікацій між злочинцями. Прихована передача інформації в інших формах дозволяє уникнути виявлення і перехоплення конфіденційних даних.

Зловмисники можуть використовувати стеганографію для уникнення виявлення прихованої інформації під час здійснення злочинних дій. Це може бути важливим в аспектах кіберзлочинності, де виявлення може призвести до розслідувань та відкриття злочинів.

Сховане вбудовування зловмисної інформації в широко використовувані файли (зображення, аудіо, відео) може слугувати як частина кібератак. Зловмисники можуть використовувати стеганографію для приховування шкідливого коду, вірусів чи інших видів зловмисного програмного забезпечення від антивірусних програм.

Вбудовання фальшивої або маніпульованої інформації в медіафайли дозволяє зловмисникам поширювати дезінформацію. Це може використовуватися для впливу на громадську думку, виборчий процес або інші соціально-політичні процеси.

Використання стеганографії може допомогти зловмисникам ухилитися від відповідальності, приховавши свої дії або плани. Це може бути важливим при плануванні злочинів чи терористичних актів.

Хоча стеганографія може мати інноваційне та корисне застосування, зловмисники також можуть використовувати ці техніки для здійснення негативних дій [1].

Стеганографія може бути використана для реалізації різних типів атак. Ось деякі загальні типи атак з використанням стеганографії:

Стеганографічні атаки *Табл. 1.1.*

Атака	Опис атаки
Приховання інформації	Контрабанда даних: Зловмисники можуть використовувати стеганографію для контрабанди конфіденційної інформації через контрольовані точки (наприклад, файли зображень або звукові файли).
	Приховування атак: Атакувачі можуть вбудовувати свої атаки в невинні файли, щоб уникнути виявлення та розслідування

Атака	Опис атаки
Розповсюдження шкідливих програм	Схована доставка шкідливого коду: Атакувачі можуть вбудовувати шкідливий код в безпечно виглядаючі файли (наприклад, зображення, відео, документи) для обману антивірусного програмного забезпечення та інших засобів виявлення загроз.
Дезінформація та обман	Розповсюдження фейків: Атакувачі можуть використовувати стеганографію для прихованого вбудовування фейкової або маніпульованої інформації в медіафайли для поширення дезінформації.
Атаки на керівництво та державу	Кібершпигунство: Спецслужби або кіберзлочинці можуть використовувати стеганографію для приховування та обміну конфіденційною інформацією у рамках кібершпигунства.
Обхід систем фільтрації та блокування	Обхід контролю вводу/виводу

Атака	Опис атаки
Витік конфіденційної інформації	Прихована передача конфіденційних даних: Зловмисники можуть використовувати стеганографію для таємної передачі конфіденційних даних, уникаючи виявлення і контролю.

1.2. Кібергігієна

Кібергігієна, або кібергігієнічні заходи, в контексті стеганографії включають в себе превентивні заходи для захисту від використання стеганографії в зловмисницьких цілях та максимізації безпеки інформації.

Ось деякі рекомендації з кібергігієни при роботі підприємства з потенційно стеганографічними даними:

1. Моніторинг та виявлення:

Застосування системи моніторингу трафіку та виявлення вторгнень для виявлення незвичайної або підозрілої активності в мережі.

2. Фільтрація та блокування файлів:

Використання антивірусне програмне забезпечення та системи фільтрації, які можуть розпізнавати та блокувати стеганографічні файли.

3. Освіта користувачів:

Проведення навчання користувачів стосовно можливих загроз стеганографії та навчайте їх розпізнавати підозрілі дії.

4. Політика використання обладнання:

Встановлення політики використання обладнання, що обмежують можливість використання зовнішніх пристроїв, які можуть бути використані для стеганографічного обміну даними.

5. Перевірка конфіденційності вмісту:

Застосування механізмів контролю конфіденційності для управління доступом до чутливих даних та обмеження можливостей стеганографії.

6. Вдосконалення методів стеганалізу:

Розвиток та вдосконалення методи стеганалізу для виявлення стеганографічних вкраплень та аналізу захованих даних.

7. Шифрування даних:

Використання шифрування для захисту важливої інформації, навіть якщо вона стає об'єктом стеганографії.

8. Аудит безпеки:

Проводення регулярний аудит безпеки для виявлення можливих слабків у системі та ефективного реагування на виявлені загрози.

9. Строгий контроль над обмінними пристроями:

Обмеження використання зовнішніх пристроїв із можливістю запису (флешки, зовнішні жорсткі диски) та контролюйте їхній доступ до комп'ютерних систем.

10. Системи детекції стеганографії:

Впровадження спеціалізованих систем детекції стеганографії для виявлення прихованої інформації [2].

1.3. Випадки кібератак з використанням стеганографії

Існують випадки, коли стеганографія використовувалася в різних сферах, включаючи кіберзлочинів, шпигунство та тероризм. Ось кілька прикладів відомих інцидентів:

1. Шпигунство через електронну пошту (APT29 - Cozy Bear):

В 2015 році група APT29 (відома також як Cozy Bear) використовувала техніки стеганографії для прихованого передавання команд іншому шпигунському програмному забезпеченню. Інформація була захована в зображеннях, щоб уникнути виявлення.

Cozy Bear використовує власні та високоспеціалізовані інструменти для проведення атак. Вони розробили унікальне ПЗ та застосовують розширені тактики для уникнення виявлення та ідентифікації.

Група активно веде фішингові кампанії, використовуючи електронну пошту для розсилання шкідливих вкладень та посилань. Це може включати в себе лукаві електронні листи, що маскуються під легітимні комунікації.

Cozy Bear використовує вразливості у програмному забезпеченні та операційних системах для впровадження своїх інструментів та отримання доступу до цільових систем.

Також група активно застосовує методи камуфляжу та залишення невиявленими, використовуючи захисні механізми для уникнення виявлення своїх атак.

Cozy Bear відома своєю довготривалою діяльністю та надалі залишається активною у кіберпросторі. Вони продовжують здійснювати атаки та шпигунську діяльність, адаптуючи свої методи відповідно до змін у кіберпросторі.

APT29 має глобальний охоплення та вплив, здійснюючи атаки на різні країни та сектори, включаючи політику, енергетику, високі технології, фінанси та інше.

2. Інциденти на основі фейкових зображень:

У багатьох випадках стеганографія використовується для створення та розповсюдження фейкових зображень, включаючи фотографії політиків або військових подій. Це може викликати дезінформацію та паніку.

3. Використання стеганографії в кіберзлочині (Carbanak):

Група кіберзлочинців, відома як Carbanak, використовувала стеганографію для приховування своїх зловмисних дій у великому обсязі нормального мережевого трафіку. Вони приховували конфігураційні дані та команди у звичайних зображеннях.

Carbanak може використовувати стеганографію для завантаження шкідливого коду на компрометовані системи. Код може бути вбудований у

нормально виглядаючі файли або трафік, що ускладнює виявлення та блокування.

Використання стеганографії також може допомагати групі Carbanak уникати виявлення та блокування від захисних заходів, які зазвичай виявляють стандартні методи атак.

4. Використання терористами:

Інциденти свідчать про те, що терористи можуть використовувати стеганографію для прихованої комунікації та обміну інформацією, оскільки це дозволяє їм уникати виявлення та перехоплення. Ось деякі загальні тенденції та приклади:

- Інтернет-рекрутинг та пропаганда:

Терористичні організації можуть використовувати стеганографію для приховування текстової або мультимедійної інформації у звичайних зображеннях або відео. Це може бути спробою обійти системи фільтрації та виявлення контенту.

- Комунікація між клітинами:

Злочинці та терористи можуть використовувати стеганографію для прихованого обміну повідомленнями та інструкціями між членами клітини. Вона дозволяє їм уникнути перехоплення та виявлення з боку правоохоронних органів.

- Прихована публікація інструкцій:

Замаскування інструкцій та кодів для виконання терористичних актів може здійснюватися через стеганографію. Це може включати в себе вбудування інформації в непомітні файли чи мультимедійний контент.

- Обхід систем моніторингу та перехоплення:

Використання стеганографії може допомагати уникнути виявлення комунікації та обміну інформацією під час пересилання через мережі та платформи, які мають системи моніторингу та фільтрації [3].

1.4. Сучасні методи захисту повідомлень

Можна виділити чотири напрями стеганографії: класична, цифрова, лінгвістична та квантова стеганографія. Класична (традиційна) стеганографія – метод приховування даних, що здійснюється за допомогою технічних засобів захисту інформації. Сучасна стеганографія (рис.1.2) включає матеріальні та інформаційні методи.

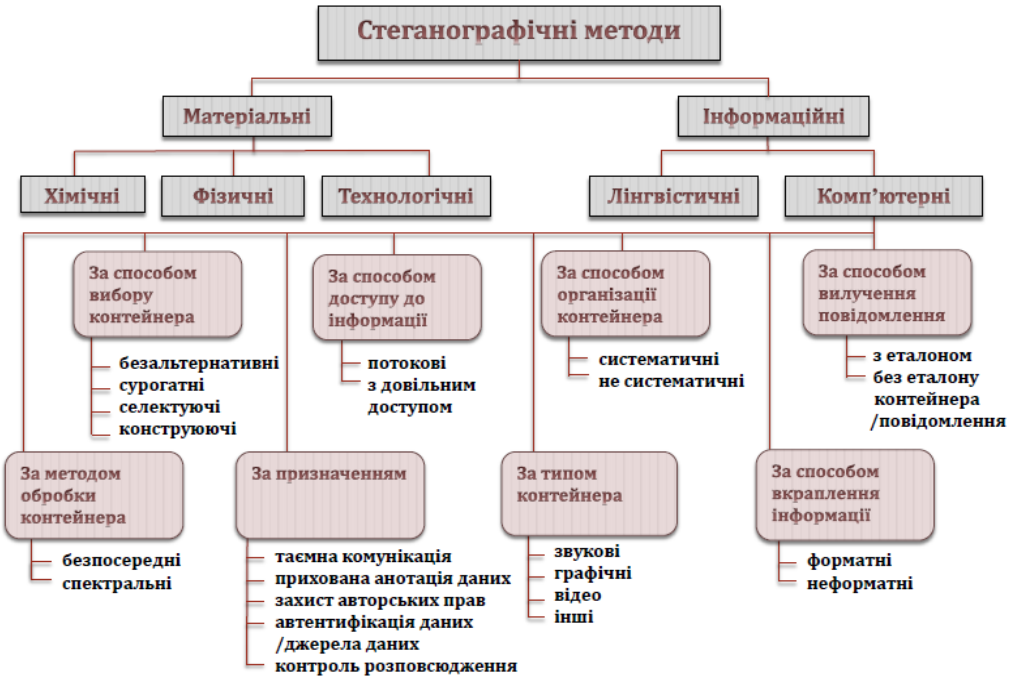


Рис. 1.2. Методи стеганографії

Цифрова стеганографія (рис.1.3) – заснована на приховуванні або вбудовуванні додаткової інформації в цифрові об'єкти, тим самим спотворюючи їх. Як правило, ці об'єкти є мультимедійними і внесення спотворень, що є нижче порога чутливості звичайної людини, не призводить до їх видимих змін [4].

Рис. 1.3. Методи цифрової стеганографії

За методом вибору контейнера можна виділити безальтернативні, вибіркові та конструктивні стеганографічні методи.

Безальтренативні методи включають вибір першого можливого контейнера для приховування повідомлення. Вибіркові методи припускають, що приховане повідомлення повинно відображати певні характеристики шуму контейнера. Конструктивні методи передбачають, що контейнер генерується самою стеганосистемою. [5]

Залежно від типу доступу до конфіденційних даних розрізняють потокові методи та фіксовані контейнери.

Потокові контейнери – це послідовності бітів, що постійно змінюється. Повідомлення вбудовується у нього в реальному режимі часу, тому заздалегідь невідомо, чи вистачить розміру контейнера для передачі всього повідомлення. Фіксовані контейнери мають фіксований розмір, тому можна вибрати оптимальний контейнер для передачі повідомлення [6].

За типом організації розрізняють методи систематичних і несистематичних контейнерів. У перших можна виділити біти шуму та сам контейнер. Для других інших така операція неможлива.

Відповідно за принципом приховування існують два основні класи: методи прямої заміни та спектральні методи. Перші використовують надлишковість контейнера і замінюють несуттєві частини контейнера бітами секретного повідомлення, а другі приховують дані, використовуючи спектральне

представлення елементів у середовищі, в яке вбудовані секретні дані (наприклад, коефіцієнти масивів перетворень Фур'є).

За призначенням у сфері захисту авторського права можна виділити методи прихованої передачі або зберігання даних і методи приховування даних у цифрових об’єктах.

Приховати дані в просторовій області можна за допомогою таких методів:

1) метод заміни найменш значущого біта (НЗБ), який полягає в заміні бітами приховуваного повідомлення останніх значущих бітів в контейнері;

2) метод псевдовипадкового інтервалу – полягає у довільному розподілі бітів прихованого повідомлення по контейнеру, що дає в результаті відстань між встановленими бітами, яка визначається псевдовипадковим чином;

3) метод псевдовипадкової перестановки заснований на тому, що генератор псевдовипадкових чисел (ПВЧ) генерує послідовність індексів $j_1, j_2, \dots, j_{IM}$ та зберігає k-й біт повідомлення в пікселі з індексом j_k. Отже, приховані біти будуть однаково розподілені по всьому бітовому просторі контейнера;

4) метод приховування блоку полягає в тому, що вихідне зображення розбивається на l_M непересічних блоків $\Delta_i (1 \le i \le l_m)$ будь-якої конфігурації, для кожного з якої біт парності $b(\Delta_i)$: $b(\Delta_i) = \sum_{J-\Delta_i}^{mod\ 2} LSB(C_j)$. Один біт відповідає одному блоку M_i . Якщо біт парності $b(\Delta_i \ne M_i)$ то один з блоків НЗБ буде інвертований Δ_i^d в результаті чого $b(\Delta_i = M_i)$;

5) метод заміни палітри: палітра з N-ї кількості кольорів визначається як список пар індексів (i, Δ_i), де і – індекс, а Δ_i - вектором кольорів. Кожному пікселю зображення у таблиці присвоюється певний індекс. Конфіденційну інформацію можна приховати шляхом перестановки кольорів у палітрі, бо порядок кольорів у палітрі не важливий для реконструкції загального зображення,;

6) метод квантування зображення відбувається таким чином, що інформація приховується шляхом регулювання різницевого сигналу Δ_i.

Стеганоключ представляє - таблиця, яка відповідає кожному можливому значенню Δ_i із певним бітом;

7) метод Куттера-Джордана-Боссена. Людське око менш за все відчуває різницю в синьому кольорі. Метод заснований на вставці секретного повідомлення в синій канал.

8) Метод Коха-Джао (відносний ДКП (дискретне косинусне перетворення)). Вихідне зображення розбивається на блоки 8x8 пікселів. В результаті використання ДКП до кожного блоку створюється таблиця ДКП коефіцієнтів. Для блоку, перетвореного ДКП, використовується три пари коефіцієнтів:

Якщо

$$B_i^Q(u_1, v_1) > B_i^Q(u_3, v_3) + D \text{ та } B_i^Q(u_2, v_2) > B_i^Q(u_3, v_3) + D, \quad (1.1)$$

B^i кодує 1

Якщо

$$B_i^Q(u_1, v_1) + D < B_i^Q(u_3, v_3) \text{та } B_i^Q(u_2, v_2) + D < B_i^Q(u_3, v_3), \quad (1.2)$$

B^i кодує 0,

де u_i, v_i – коефіцієнти квантування.

9) F5 – метод, який враховує недоліки LSB (низька стійкість до атак на основі статистичних досліджень контейнера). Подібний до методу Коха-Джао, але більше вдосконалений для підвищення стійкості до атак на основі статистичних вимірювань, в тому числі χ^2-атак. Спочатку графічний файл розбивається на блоки 8x8. Кожен блок піддається дискретному косинусному перетворенню (ДКП), після чого відбувається квантується за допомогою деякої таблиці квантування. Останній крок це стиснення без втрат (відомі методи – RunLengthEncoding, EntropyEncoding тощо).

10) Метод Бенгама-Мемона-Ео-Юнг (Benham-Memon-Yeo-Yeung). По-перше, використовуються тільки найбільш налаштовані блоки. По-друге, замість двох обрано три коефіцієнта ДКП, що дозволяє зменшити спотворення в контейнері.

11) Метод Хсу-Ву (Hsu-Wu) – алгоритм для вбудовування бінарних цифрових водяних знаків. Пікселі можуть мати лише значення "0" або "1", через що безпосереднє спостереження такого зображення відкидається, оскільки інтенсивності "0" і "1" відповідають чорному. Водяний знак може бути створений чорно-білим, а потім все поле можна розділити на 255, замінивши інтенсивність білих пікселів на «1» [7].

1.5. Порівняння різних методів та їхні переваги та недоліки

До способів приховання даних в тексті належать:

1) синтаксичний та семантичний методи. До синтаксичних методів належать методи зміни пунктуації та методи зміни будови і стилю тексту. Семантичні методи схожі на синтаксичні, вони визначають два синоніми які відповідають значенням приховуваних біт. Щоб використати семантичні методи необхідна таблиця синонімів;

2) методи довільних інтервалів базується на трьох методах (заміни пробілів між реченнями, заміни кількості інтервалів у кінці рядків, заміна кількості пробілів між словами в тексті з вирівнюванням по ширині). Вони використовують вільний простір у тексті, щоб приховати дані. У деяких джерелах описані вище методи відносяться до лінгвістичної стеганографії [8].

При аналізі методів цифрової стеганографії, можна виділити їхні переваги та недоліки. Переваги включають:

1) простота реалізації методів;

2) висока стійкість до атак;

3) візуальну узгодженність між відредагованим і вихідним повідомленнями;

4) наявність безкоштовного ПЗ для реалізації методів.

Недоліками цифрової стеганографії є:

1) висока чутливість до найменших викривлень контейнера;

2) ймовірність помилок виявлення;

3) складність розміщення інформації в контейнер (у разі великого обсягк секретної інформації) [9].

1.6. Обрані методи приховування інформації в зображеннях

1.6.1. Метод найменшого значущого біта (LSB).

Суть методу полягає в заміні молодшого біта для приховування інформації шляхом зміни останніх бітів кольору графічного об'єкту на біти прихованого повідомлення. Різниця між порожніми та заповненими контейнерами не повинна сприйматися людськими органами чуття.

Найчастіше зустрічається в електронній стеганографії. Він заснований на обмежених можливостях почуття, через що людині дуже важко розрізнити найменші зміни звуку чи кольору. Розглянемо цей спосіб на прикладі 24-розрядного растрового зображення RGB. Кожна точка закодована 3 байтами, кожен з яких визначає інтенсивність червоного, зеленого та синього кольорів. Набір інтенсивностей кольору в кожному з 3 каналів визначає відтінок пікселя. Змінюючи молодший біт, ми змінюємо значення байту на одиницю. Такі градації не тільки непомітні для людини, але й можуть взагалі не з'являтиметься при використанні пристроїв виведення низької якості. Приклад нижче показує, як повідомлення можна приховати в першом вісім байтів, що представляють три пікселі в 24-бітному зображенні.

```
Pixels: (00100111 11101001 11001000)
        (00100111 11001000 11101001)
        (11001000 00100111 11101001)
A: 01000001
Result: (00100110 11101001 11001000)
        (00100110 11001000 11101000)
        (11001000 00100111 11101001)
```

Рис.1.4. Приклад роботи методу LSB

Стеганаліз методом LSB. Порушення статистичних закономірностей природні контейнери є одним із найбільш перспективних підходів виявити наявність прихованого каналу передачі інформації це підхід, який вводить приховану інформацію у файл. Цей підхід аналізує статистичні характеристики досліджуваного послідовності та визначити, чи схожі вони за характеристиками природні контейнери (якщо так, то немає прихованої передачі інформації), або вони подібні до характеристик стего (якщо так, то існування прихований канал передачі інформації). Цей клас атак steg імовірнісні, тобто не дають певної відповіді, а формують оцінки типу «ця досліджувана послідовність має 90% ймовірність того, що містить приховане повідомлення». Метод використовує аналіз гістограми, отриманої за елементами зображення та оцінка розподілу пар значень цієї гістограми. Для файлів BMP пари значень утворюються значеннями пікселів зображення; для JPEG вони квантуються дискретними коефіцієнтами косинусного перетворення, які відрізняються найменшим значущим розрядом. Молодші біти зображень – не випадкові. Частоти двох сусідніх елементів контейнера повинні бути досить далеко від частотного значення середнього арифметичного цих елементів. В «порожньому» зображенні ситуація, коли частоти елементів зі значеннями 2N і 2N + 1 близьке за значенням, зустрічається досить рідко. При вбудовуванні інформації ці частоти наближаються або стають рівними.

Метод стегоаналізу атаки хі-квадрат.

Ідея атаки хі-квадрат полягає в тому, щоб знайти близькі значення та обчислити ймовірность вбудовування на основі того, наскільки близько розташовуються значення частот парних та непарних елементів обраного контейнера. Характерною особливістю алгоритму є послідовний аналіз всього графічного файлу і таким чином накопичення частот елементів.

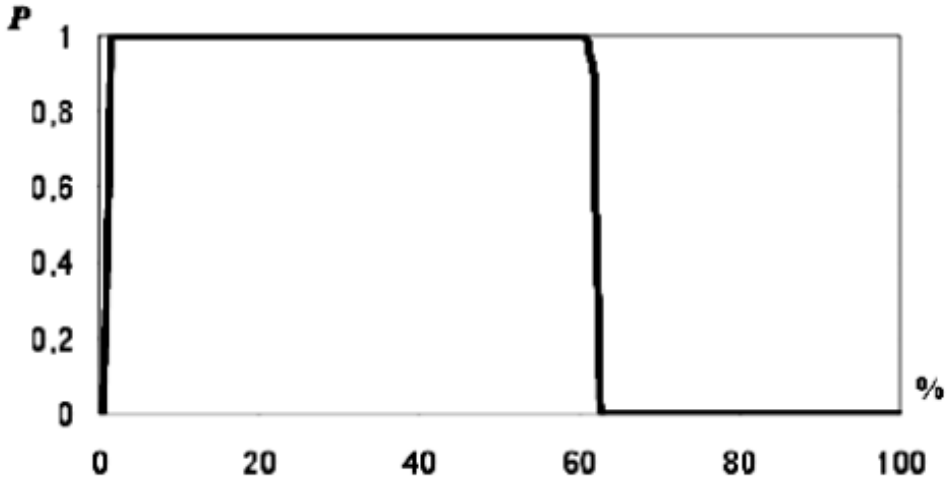


Рис.1.5. Імовірність вставки за критерієм Хі – квадрат при аналізі стегоконтейнера, отриманого методом поетапної заміни

Метод Хі-квадрат є універсальним, оскільки підходить для аналізу графічних файлів, створених різними програмами маскування. Але результати роботи методу Хі-квадрат значною мірою залежать від способу кодування даних. Метод дає хороші результати, коли елементи контейнера послідовно записуються в НЗБ (рис.1.5), але метод не працює, коли молодші біти вибираються псевдовипадково і повідомлення розповсюджується по всій довжині контейнера [10].

1.6.2. Метод заміни палітри.

Ви також можете використовувати заміну палітри кольорів у форматі зображення, щоб приховати дані. Палітра з N кольорів визначається як список пар індексів (i, Δi), що визначає відповідність між індексом i та його кольоровим вектором Δi. Кожен піксель графічного файлу відповідає певному індексу у таблиці. Оскільки порядок кольорів на панелі не важливий для реконструкції загального зображення, конфіденційну інформацію можна приховати, перефарбувавши панель.

Є N! різні способи перегрупування палітри кольорів N, чого достатньо, щоб приховати невелике повідомлення. Однак методи приховування, засновані

на порядку створення палітри, нестабільні: будь-яка атака із зміною палітри, руйнує приховане повідомлення.

Найчастіше сусідні кольори на палітрі не обов'язково схожі, тому деякі стеганометоди використовують палітру перед вставленням так, що сусідні кольори були подібними. Наприклад, значення кольору можна впорядкувати на відстані d в RGB-просторі, де

$$d = \sqrt{R^2 + G^2 + B^2} \text{ (1.3).}$$

Оскільки ЗСЛ більш чутливий до змін яскравості кольору, потрібно сортувати вміст палітри відповідно до значень яскравості сигналу. Після того, як панель вирівняна, ви можете змінювати значення НЗБ індексу кольору без надмірного спотворення зображення..

Деякі методи можуть зменшити загальну кількість значень кольорів (до N/2) шляхом "розмивання" зображення. При цьому елементи палітри кольорів дублюються таким чином, щоб значення кольорів для них трохи відрізнялося. В результаті кожне значення кольору розмитого зображення відповідає двом елементам палітри, вибраним відповідно до біту прихованого повідомлення [11].

1.7. Основні поняття та терміни у стегоаналізі

Стегоаналіз — частина стеганографії, наука про пошук факту передавання прихованої інформації в даному повідомленні. Стеганаліз — це мистецтво і наука визначення того, чи містить повідомлення приховану інформацію. Стеганаліз бере участь у визначенні відблисків або особливостей зображення для пошуку прихованої інформації, а також методів планування виділення або вилучення прихованої інформації. Техніка стегоаналізу буде ефективною, якщо вона може ідентифікувати та витягувати імплантовану приховану інформацію. В деяких випадках стегоаналіз також означає вилучення прихованої інформації з повідомлення, яке містить її, і (при необхідності) подальшу розшифровку [12].

Перші спроби аналізу та виявлення прихованої інформації супроводжували появу стеганографії. Деякі з найдавніших методів стеганографії

включали в себе вбудовання тексту в текст, зміни символів або використання непомітних символів для кодування інформації.

З виникненням нових технологій, таких як фотографія, аудіо та відео, стеганографія отримала нові можливості для приховання інформації. Спроби аналізу стали зорієнтовані на розпізнавання характерних змін у цифрових сигналах.

З поширенням цифрових технологій і використанням комп'ютерів для обробки та передачі даних стеганографія набула нових розмірів. Аналіз став складнішим, інноваційні методи вбудовування інформації та адаптація до різних типів медіа стали важливими аспектами стегоаналізу.

З поширенням Інтернету, соціальних мереж та великих обсягів цифрової інформації зросла і важкість завдань стегоаналізу. З'явилися нові методи та інструменти для аналізу великих обсягів даних з різних джерел [13].

Сучасні методи стеганографії використовують розширені алгоритми та техніки для вбудовування інформації. Стегоаналіз стає складнішим через використання адаптивних та інтелектуальних методів приховання. Дослідники в галузі стегоаналізу постійно розвивають нові методи для виявлення та аналізу прихованої інформації.

Стегоаналіз існує у двох видах – статичному та динамічному. Метою статичного стеганоаналізу є визначення існування/неіснування прихованого повідомлення та розпізнавання алгоритму вбудовування, динамічного – прийняття гіпотези по параметр(и) алгоритму вбдовування або секретного повідомлення (здогадка про довжину, позиції вбудованого повідомлення, ключ тощо) [14].

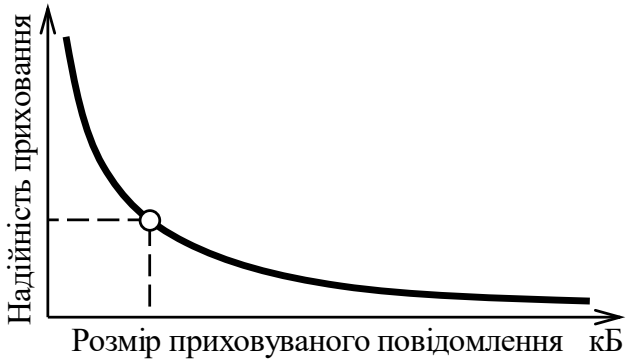


Рис.1.6. Взаємозв’язок між стійкістю стеганосистеми та об’ємом приховуваного повідомлення при незмінному розмірі файлу – контейнера

Щодо загальної стійкості стеганографічних систем, можна виділити такі типи:

- Теоретично стійка стеганографічна система приховує інформацію лише в тих частинах контейнера, значення елементів яких не перевищують рівень шумів чи похибок квантування, і при цьому теоретично доведено неможливість створити стеганоаналітичний метод пошуку прихованої інформації;
- Практично стійка стеганосистема передбачає таку модифікацію фрагментів контейнера, зміни яких можуть бути знайдені, але відомо, що зараз недостатньо стеганоаналітичних методів в порушника або їх поки що не існує в достатній кількості;
- Нестійка стеганосистема приховує інформацію таким чином, що стеганоаналітичні засоби, які вже існують, можуть її виявити.

Також, враховуючи можливі атаки (стискання із втратою даних, атака на знищення повідомлення, атака на основі відомого чи вибраного контейнера (оригіналу чи результату), на основі відомого вбудованого повідомлення тощо), можна виділити стеганографічні системи, стійкі до пасивних та активних атак [15]. Щодо пасивних атак, визначимо що таке абсолютно стійка стеганосистема:

Нехай P_C – ймовірність розподілу контейнерів, а P_S – ймовірність формування стеганограми E(c, m, k) на кількості S всіх можливих стеганограм,

одержаних за допомогою стеганосистеми. Зауважимо, що неавторизована особа не повинна мати доступ до набору контейнерів, що призначені для таємного зв'язку. Тоді відносна ентропія на множині Q може бути визначена наступним чином:

$$D(P_0||P_1) = \sum_{q\in Q} P_0(q) \log_2 \left(\frac{P_0(q)}{P_1(q)}\right), \quad (1.4)$$

де P_0 та P_1 – ймовірності приймання вірної та невірної гіпотези про розподіл відповідно. Відносна ентропія між двома розподіляннями завжди є невід'ємною і дорівнює 0 лише у випадку тотожності даних розподіляння.

Нехай Σ – стеганосистема; P_S – розподіл імовірностей передавання каналом зв'язку стеганограм; P_C – розподіл імовірностей передавання каналами зв'язку пустих контейнерів. Система Σ називається ρ-надіною до пасивних атак, якщо$\boldsymbol{D(P_c||P_s)} \leq \rho$, і є абсолютно надійною, якщо $\rho = 0$. Як було зазначено, співвідношення $\boldsymbol{D(P_c||P_s)}$ дорівнює 0 тільки тоді, коли обидва розподіли імовірностей дорівнюють один одному. Отже, стеганографічна система Σ є теоретично абсолютно надійною, якщо процес вбудовування таємного повідомлення в контейнер не змінює розподіл P_C.

Абсолютно надійна система може бути створена, наприклад, на основі одноразової гами.

Вище було визначено P_0 та P_1, які відповідають гіпотезам:

- H_0 – повідомлення не містить таємних даних;
- H_1 – у повідомленні приховано таємні дані.

З точки зору можливих дій, хибним позитивним є рішення заблокувати передавання звичайного повідомлення. Хибним негативним, відповідно, є рішення дозволити передачу таємного повідомлення.

Основним показником оцінки якості стеганографічної системи є ефективність вбудовування: чим вона вища, тим більше одиниць інформації можна вкласти в контейнер без суттєвих спотворень вихідної послідовності, що, у свою чергу, зменшує ймовірність знаходження каналу передачі атакуючою стороною.

Пропускна здатність стеганографічного каналу є ще одним важливим показником; розуміється відношення розміру контейнера до розміру повідомлення, для систем прихованого передавання даних повинна бути значно вище, ніж для інших типів систем [16].

Іншим важливим показником нещодавно стала стійкість (Robustness) – міра здатності, при використанні алгоритму, зберегти приховані дані навіть після того, як контейнер було стиснено та декомпресовано із втратами або іншими змінами, як-от перетворенню в аналоговий сигнал та назад у цифровий.

Найбільшим недоліком відомих показників спотворення як для графічних, так і для аудіо- та відео-контейнерів, є порушення реагування органами чуття людини. Розробка нових методів, а також встановлення співвідношення реальних показників із теоретичними, все ще є важливими напрямками розвитку стеганографії.

Ступінь сприйнятої людської якості працює з системою візуалізації людини, чутливою до контрасту та феномену маскування, і базується на багатоканальній моделі просторового бачення людини. Розрахунок цього показника відбувається при: крупнокроковій сегментації зображення; розкладання помилки кодування та первинного зображення на перцептивні (що стосуються сприйняття органами чуття) компоненти за допомогою гребінчастих фільтрів; обчислення порогу для кожного пікселя, використовуючи первинне зображення як маску; розподіл відфільтрованої помилки з використанням порогу прийняття рішення, суміщеного по всіх колірних каналах. Одиниця вимірювання показника збільшена як одиниця перевищення порогу, яке виконується нижче лише суттєвої (помітної) різниці (JustNoticeableDifference). Загальний показник, масковане максимальне співвідношення сигнал/шум (MaskedPeakSignaltoNoiseRatio):

$$\boldsymbol{MPSNR = 10 \log\left(\frac{255^2}{\varepsilon^2}\right)} \quad (1.5)$$

де ε – обчислене спотворення. Оскільки даний показник якості не відповідає суті, яку було закладено у поняття децибел, його називають візуальним або зоровим децибелом (ВдБ).

У разі протидії активним атакам існують два підходи до створення надійних стеганосистем:

1) передбачаючи можливість атаки на стеганограми з боку порушників, стеганографічне перетворення негайно проектується таким чином, щоб бути стійким до руйнування прихованих даних певним класом модифікацій;

2) здійснюються перетворення, які мають властивість бути зворотними до можливих модифікацій із зазначенням відновлення початкового вигляду стеганограми. При цьому передбачається оцінка параметрів перетворення, вимірювання змін форми, розміру та напрямку деяких кодованих зображень [17].

Атака (коаліція) небезпечна і специфічна для цього типу стеганосистеми: кілька користувачів, кожен з яких отримав свій екземпляр контейнера з вбудованим у нього унікальним ідентифікаційним номером, стають порушниками і скоординованими діями намагаються побудувати найближчу до початкової оцінки порожнього контейнера, який зберігає його функціональність, але не містить ідентифікаційної інформації. На практиці це актуально, наприклад, для завдань із захисту авторських і майнових прав на CD/DVD диски з музикою чи фільмами тощо. На рис.1.7. наведено схему конфлікту між характеристиками стеганографічної системи.

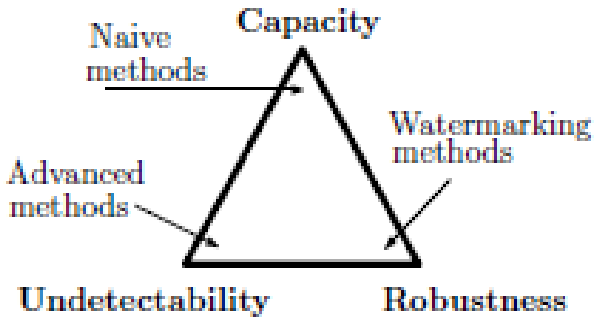


Рис.1.7. Конфлікт між значеннями характеристики стеганографічних систем

Безсумнівно, практичний інтерес становлять стеганографічні рішення. Їх можна використовувати як альтернативу криптографічним засобам приховування інформації, в деяких випадках більш ефективно. Крім того, вони служать потужною основою для створення цифрових і нецифрових водяних знаків, а також для контролю автентичності. Нарешті, хоча використання криптосистем законодавчо регулюється та дещо обмежено, жодна країна не має подібних обмежень щодо розробки та розповсюдження стеганосистем [18].

1.8. Методи стегоаналізу

На першому етапі аналітик зі стеганографії представляє досліджуване повідомлення у формі контейнера, який, як відомо, відповідає його методу стеганографії для цього типу повідомлення. Щоб визначити контейнер, потрібно розуміти спосіб введення отриманої інформації та знати місце в повідомленні, куди можна розмістити стего. Таким чином, на першому етапі аналітик:

• обирає метод стеганографії, за допомогою якого можна було б внести приховану інформацію в досліджуване повідомлення,

• структурує повідомлення у вигляді відповідного контейнера

• отримує виявлення про можливість додавання тегу до повідомлення вибраним способом.

Простір у контейнері (або його об’єм), куди можна ввести стего за допомогою цього методу стеганографії, як правило, використовує корисну ємність контейнера.

Другим кроком є можлива атака на досліджуване повідомлення — тобто модифікація контейнера (яким є це повідомлення в рамках обраного методу стеганографії) з метою стегоаналізу. Як правило, атаки здійснюються шляхом введення вільно отриманої інформації в контейнер за допомогою обраного для аналізу методу стеганографії.

Третій і останній крок - безпосередньо стегоаналіз: контейнер піддається атаці, і на основі вивчення отриманих «атакованих» повідомлень, а також

початкового повідомлення робиться висновок про наявність або відсутність стего в досліджуване повідомлення. Сукупність розроблених атак і методів дослідження отриманих повідомлень є методом стегоаналізу. Атака (атаки), за допомогою якої вдалося виявити наявність прихованої інформації, називається успішною атакою [19].

Завдання стегоаналізу – виявити факт передачі прихованої інформації в аналізованому повідомленні. У деяких випадках стегоаналіз також означає вилучення прихованої інформації з повідомлення, яке її містить, і (якщо можливо) подальше розшифрування вилученої інформації [20].

Існують такі методи стегоаналізу графічних файлів:

• методи, призначені для виявлення даних, прихованих за певним алгоритмом;

• методи «сліпого» розпізнавання;

• пасивні методи стегоаналізу, що визначають наявність/відсутність прихованих даних у стегоконтейнері, або методи, що визначають алгоритм, за яким відбулося вбудовування;

• методи активного стегоаналізу, які визначають довжину вбудованого документа, його розташування, деякі параметри алгоритму впровадження, а також видаляють приховану інформацію;

• методи підпису, засновані на пошуку в стеганограмах так званих «відбитків пальців» - фрагментів коду, який залишають стеганографічні програми після своєї роботи;

• імовірнісні методи, які базуються на аналізі ймовірнісних показників, характерних для стегоповідомлень;

• методи аналізу зображення безпосередньо, тобто в просторовій формі зображення;

• методи аналізу частотних форм представлення зображення, тобто після перетворення його в частотну форму за допомогою дискретного косинусного або вейвлет-перетворення;

• методи, які використовують статистичні критерії узгодження (наприклад, хі-квадрат);

• методи, які використовують міру подібності цифрової сукупності;

• метод визначення сумісності JPEG.

Існує багато методів стеганалізу, які відрізняються своїми характеристиками зображень та методів вбудовування, які вони використовують для протистояння. Залежно від використовуваних вихідних даних, методи стеганалізу є традиційними поділяються на сигнатурні, статистичні та евристичні.

Методи стеганалізу на основі підписів розроблені для роботи з методами приховування формату інформації, яка в процесі приховування залишає за собою специфічні маркери (сигнатури), за якими можна виявити приховане вкладення [21].

Методи статистичного стеганалізу базуються на аналізі статистичних характеристик досліджуваного зображення з метою визначення того, як вони співвідносяться з характеристиками порожніх контейнерів того ж типу. Найбільш відомими статистичними методами є стеганаліз RS і WS-стеганаліз, гістограма, SPAM (subtractive pixel adjacency matrix) стегоаналіз та інші підходи. Ці методи можуть показати дуже висока чутливість для виявлення наповненого контейнера для ковдри та навіть ідентифікації кількість прихованої в ньому інформації, але їх точність багато в чому залежить від алгоритму вбудовування.

Евристичні методи стеганалізу представляють великий інтерес для дослідників, тому що вони більш універсальні, оскільки не прив'язаний ні до чого алгоритм введення прихованої інформації, хоча в цілому дещо менш точний. В основному ці методи базуються на вирішенні задачі бінарної класифікації за допомогою методів машинного навчання. Розглянемо деякі з них детальніше, наприклад, у надає метод стеганалізу на основі аналізу гістограм, побудованих на основі кодової таблиці Хаффман використовував для кодування значень коди дискретного косинусного перетворення (DCT). змінної довжини. Для аналізу гістограм використовується машинне навчання за допомогою штучної нейронної

мережі. За висновками авторів, цей метод дозволяє виявляти наповнену стьобану тару, отриману за двома алгоритмами: Steghide і OutGuess з точністю від 95,4% до 98,8%. Метод дозволяє досягти більшої точності зображень великий розмір (4200 × 2358 пікселів) [22].

У роботі запропоновано алгоритм стеганалізу, також на основі сегментації зображення, але утворені фрагменти формуються відповідно зі складністю фактури. В якості вектора характеристик зображення використовується набір PEV-274, запропонований в і в даний час поширений у системах стеганалізу. Завдання класифікація вирішується шляхом застосування опорних векторних машин. Розрахункова точність методу авторів, коливається від 85 до 97% для алгоритму JPHide, від 67 до 77% - для алгоритму F5 і лише 57-62% для Алгоритм PQ.

Окремий, досить цікавий і перспективний напрям у розвитку евристичних методів стегоаналізу можна використовувати для виділення штучних імунних систем (ШІС), біологічний прототип яких це імунна система живих організмів. Основною функцією імунної системи є виявлення та знешкодження чужорідних об’єктів (антигенів), які включають, наприклад, бактерії та віруси. Антигени провокують імунну відповідь організму який починає виробляти захисні клітини імунної системи - антитіла різного типу, до є антитіло, яке специфічно зв’язується з антигеном і нейтралізує його, забезпечуючи тим самим найприродніший захист організму. Тотальність антитіла, що утворюються в процесі життя імунітет організму. У свою чергу ШІС є якийсь функціональний аналог імунної системи, здатний навчатися та бути децентралізованою розподіленою системою обробки та аналізу інформації. Застосування інформаційних інформаційних систем для вирішення завдань, Однак стегананаліз відносно новий за останні кілька років уже було опубліковано ряд робіт в цій області [23].

1.9. Роль стегоаналізу у кібербезпеці

В контексті кібербезпеки стегоаналіз відіграє важливу роль у виявленні та запобіганні захопленню та незаконному використанню прихованої інформації, що може приховуватися в різноманітних медіафайлах, текстових повідомленнях та інших формах цифрової комунікації. Основні аспекти ролі стегоаналізу в кібербезпеці включають наступне:

1. Виявлення стеганографічних загроз: Зловмисники можуть використовувати стеганографію для приховування вірусів, шкідливих програм, або навіть конфіденційних даних в невинних файлових форматах. Стеґаналіз допомагає ідентифікувати такі загрози і забезпечує можливість їхнього виявлення та нейтралізації.

2. Виявлення витоку інформації: У багатьох сферах, таких як корпоративна безпека та розвідка, важливо виявляти можливі витоки конфіденційної інформації. Стеґаналіз допомагає ідентифікувати випадки, коли конфіденційна інформація приховано відсилається або зберігається в незаконному порядку.

3. Захист від атак на канали комунікації: Зловмисники можуть використовувати стеганографію для передачі інструкцій або команд шкідливим програмам через звичайні канали комунікації. Стеґаналіз допомагає виявити та перешкодити таким атакам, попереджуючи можливі наслідки для систем безпеки.

4. Пошук прихованих повідомлень: В дослідженні кримінальних справ та справ з порушеннями кібербезпеки, стегоаналіз може бути корисним для виявлення прихованих повідомлень, які можуть мати важливе значення для розслідування.

5. Захист від соціального інжинірингу: Зловмисники можуть використовувати стеганографію для приховування фішингових атак та спроб обману користувачів. Стеґаналіз допомагає виявити такі спроби та попередити користувачів від відкриття шкідливих повідомлень.

6. Вдосконалення методів захисту: Вивчення стеганографії і стеґоаналізу допомагає розробникам та інженерам удосконалювати методи захисту і розвивати нові алгоритми для виявлення інформації, прихованої в різних типах медіафайлів та повідомлень [24].

Військові та урядові органи активно використовують стегоаналіз для виявлення прихованих загроз та забезпечення національної безпеки.

Важливо відзначити, що стегоаналіз і стеганографія розвиваються разом з іншими аспектами кібербезпеки, і обидві галузі вимагають постійного вдосконалення і досліджень, оскільки загрози в цій області постійно змінюються. Розробка нових методів стегоаналізу та розширення засобів захисту стають ключовими завданнями для забезпечення цифрової безпеки в сьогоднішньому світі [25].

1.10. Огляд інструментів та програм для стегоаналізу

Існує ряд потужних інструментів для стегоаналізу, призначених для виявлення та аналізу стеганографічних артефактів. Наприклад, "Stegdetect" (рис.1.8) визначає використання відомих стеганографічних методів, допомагаючи ідентифікувати приховані дані у зображеннях та звуку.

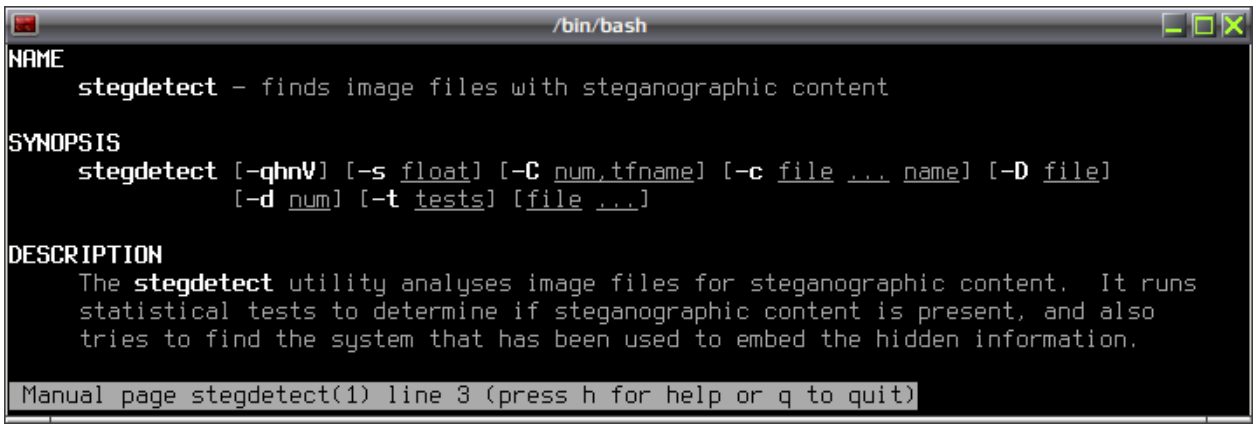


Рис.1.8. Використання Stegdetect за допомогою консолі

Steghide (рис. 1.9) - це програмний інструмент для стеганографії, який використовується для приховування конфіденційної інформації у файлі зображення чи аудіо. Основна ідея стеганографії полягає в тому, щоб приховати існуючу інформацію так, щоб це було важко помітити для стороннього спостерігача.

Програмний засіб використовується для вбудовування конфіденційної інформації в зображення чи аудіофайли та вилучення її з них. Це може бути використано для безпечної передачі та обміну інформацією, яку бажають залишити конфіденційною.

Steghide підтримує різні формати файлів, включаючи зображення у форматах JPEG, BMP, аудіо у форматах WAV та інші. Програма використовує різні методи стеганографії для вбудовування інформації, зазвичай зберігаючи стійкість до деяких форм аналізу [26].

Steghide використовується через командний рядок. Користувач може використовувати різні команди та ключі для вбудовування та вилучення інформації. Програма підтримує можливість встановлення пароля для захисту прихованого файлу, забезпечуючи додатковий рівень безпеки.

```
root@kali:~# steghide --help
steghide version 0.5.1

the first argument must be one of the following:
 embed, --embed          embed data
 extract, --extract      extract data
 info, --info            display information about a cover- or stego-file
   info <filename>       display information about <filename>
 encinfo, --encinfo      display a list of supported encryption algorithms
 version, --version      display version information
 license, --license      display steghide's license
 help, --help            display this usage information

embedding options:
 -ef, --embedfile        select file to be embedded
   -ef <filename>        embed the file <filename>
 -cf, --coverfile        select cover-file
   -cf <filename>        embed into the file <filename>
 -p, --passphrase        specify passphrase
   -p <passphrase>       use <passphrase> to embed data
 -sf, --stegofile        select stego file
   -sf <filename>        write result to <filename> instead of cover-file
 -e, --encryption        select encryption parameters
   -e <a>[<m>]|<m>[<a>]  specify an encryption algorithm and/or mode
   -e none               do not encrypt data before embedding
 -z, --compress          compress data before embedding (default)
   -z <l>                 using level <l> (1 best speed...9 best compression)
 -Z, --dontcompress      do not compress data before embedding
 -K, --nochecksum        do not embed crc32 checksum of embedded data
 -N, --dontembedname     do not embed the name of the original file
 -f, --force             overwrite existing files
 -q, --quiet             suppress information messages
 -v, --verbose           display detailed information
```

Рис.1.9. Використання Steghide

За допомогою інструмента OpenStego (рис.1.10) ви можете або приховати дані (файл) всередині зображення, або витягнути дані із зображення. Також ви можливо робити водяні знаки / перевіряти зображення своїм підписом. Спочатку потрібно створити файл підпису, а потім його можна буде використовувати для зображення водяних знаків або перевірити те саме пізніше.

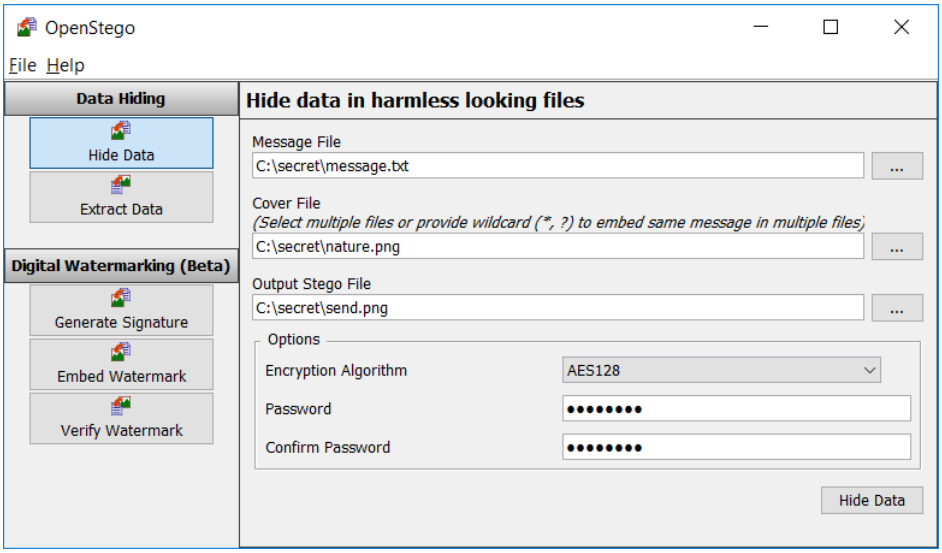


Рис.1.10. Використання OpenStego

Програма "OutGuess" (рис.1.11) є інструментом для стеганографії, але також використовується для виявлення захованих повідомлень. За допомогою аналізу статистики та властивостей контейнера, вона намагається визначити наявність стеганографічних змін. OutGuess підтримує різні формати зображень, включаючи JPEG, BMP, PNG та інші. Програма використовує різні методи для вбудовування прихованої інформації в пікселі зображення, при цьому вона старається зберегти оригінальний вигляд зображення. За вірних умов OutGuess може забезпечити ефективне приховування інформації, одночасно зберігаючи якість оригінального зображення. OutGuess є програмним засобом з відкритим кодом, що дозволяє користувачам переглядати та адаптувати його за потреби [27].

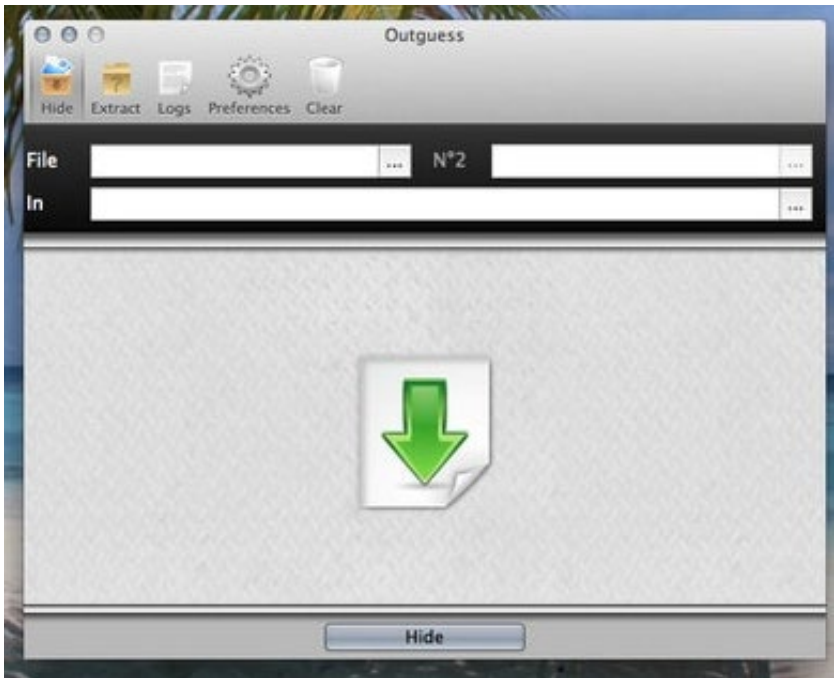


Рис.1.11. Інтерфейс OutGuess

"StegoTool" (рис. 1.12) є іншим інструментом, який використовується для виявлення стеганографії у різних медіафайлах. Він включає в себе різні методи, такі як аналіз частот та статистики, що дозволяє покращити виявлення прихованих повідомлень.

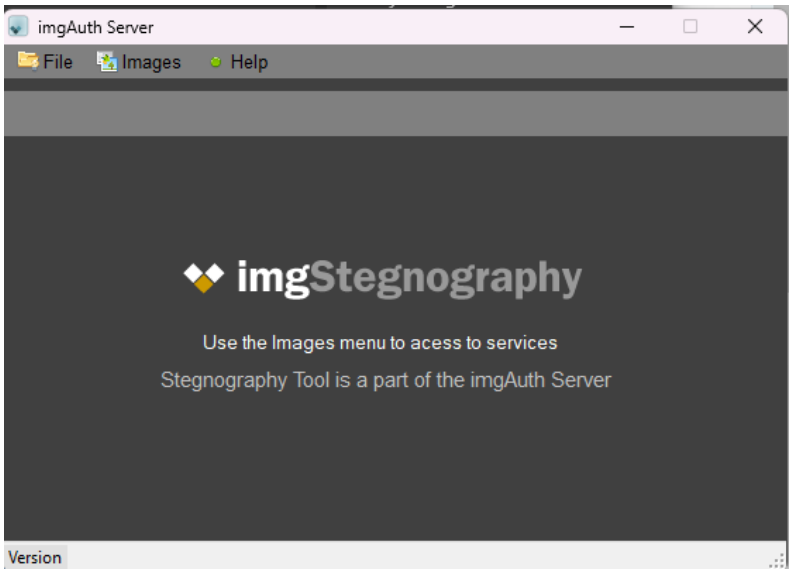


Рис.1.12 Інтерфейс StegoTool

Ці інструменти сприяють розвитку стегоаналізу, забезпечуючи засоби для виявлення та аналізу стеганографії в різноманітних цифрових носіях інформації [28].

1.11. Висновки до першого розділу.

В ході виконання першого розділу, присвяченого аналізу стеганографії та стегоаналізу, розглядаємо сучасні аспекти важливих галузей кібербезпеки. Стеганографія, як техніка приховування інформації в беззвучний спосіб, виявляється ключовою в контексті кібербезпеки. Розглядаємо різноманітні аспекти кібергігієни та методи захисту повідомлень від стеганографічних атак.

Висвітлення випадків кібератак з використанням стеганографії свідчить про актуальність проблеми та необхідність вдосконалення засобів виявлення та захисту. Порівняння різних методів стеганографії та їхні переваги та недоліки надає обґрунтовану основу для вибору ефективних засобів захисту.

Особливу увагу слід звернути на обрані методи приховування інформації в зображення, де розглядаються метод найменшого значущого біта та метод заміни палітри. Це дозволяє визначити конкретні техніки, які можуть бути використані як для захисту, так і для виявлення прихованої інформації.

Розділ також ставить акцент на стегоаналізі, визначаючи його роль у забезпеченні кібербезпеки та зазначаючи основні методи й інструменти стегоаналізу. Зазначена інформація може слугувати підставою для розробки ефективних заходів з протидії стеганографічним загрозам та збереження цілісності та конфіденційності інформації в кіберпросторі.

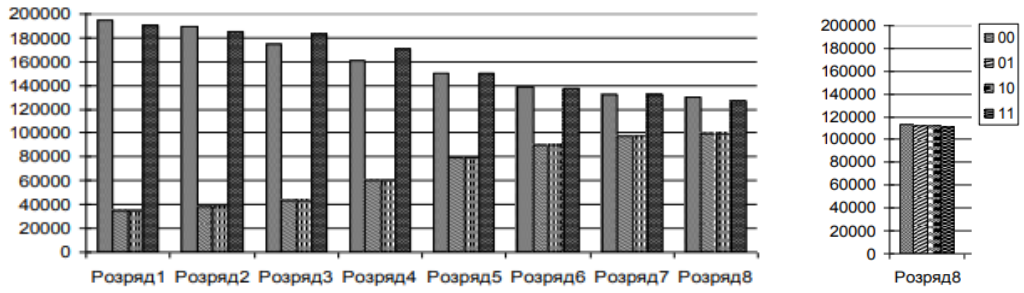
РОЗДІЛ 2. МАШИННЕ НАВЧАННЯ У СТЕГОАНАЛІЗІ.

2.1. Методи виявлення стеганографічних атак на зображення

Методи виявлення стеганографічних атак на зображення можна класифікувати на декілька категорій відповідно до їхніх основних принципів та застосування. Ось деякі типові методи:

- Статистичні методи:

Аналіз гістограм: Стеганографічні методи можуть впливати на статистичні властивості зображення. Аналіз гістограми може виявити аномалії у розподілі пікселів, які можуть свідчити про наявність стеганографічної інформації. Для пустого контейнера і контейнера, що містить вкраплену інформацію, число переходів у потоці НЗБ буде різним. Розподіл НЗБ стеганоконтейнера має, як правило, випадковий характер. Відповідно число переходів у потоці НЗБ для всіх станів буде приблизно однаковим, що не властиво пустому контейнеру (рис. 2.1, а, б).



а) б)

Рис 2.1(а,б) Гістограма частот переходів бітових значень: а – пустого контейнера, б – стеганоконтейнера (восьмий розряд контейнера, в який були внесені зміни)

- Кореляційний аналіз:

Дослідження кореляцій між пікселями чи блоками пікселів може допомогти виявити вбудовані зміни, які можуть бути притаманні стеганографічним алгоритмам.

- Фрактальний аналіз:

Використання фрактальних характеристик для виявлення аномалій у структурі зображення, що можуть бути викликані стеганографічним вбудовуванням.

- Аналіз часових та просторових властивостей:

Аналіз маркерів часу: Виявлення аномальної поведінки у часових даних, які можуть бути притаманні стеганографічним методам.

Аналіз текстурних властивостей: Виявлення відмінностей у текстурних особливостях, оскільки стеганографічні зміни можуть впливати на характеристики текстури зображення.

- Машинне навчання:

Класифікація за допомогою нейронних мереж: Використання навчених моделей для визначення, чи є зображення підозрілим на наявність стеганографічної інформації.

Використання ансамблевих методів: Об'єднання декількох алгоритмів для покращення точності виявлення.

- Фізичні методи:

Аналіз цифрових слідів: Виявлення слідів, залишених стеганографічними алгоритмами, які можуть включати в себе помітні зміни у певних областях зображення.

- Використання технологій блокчейн:

Застосування технологій блокчейн для підтвердження цілісності: Створення хешів та їхнє зберігання в розподілених реєстрах для виявлення будь-яких змін у зображенні [29].

Ці методи можуть використовуватися як окремо, так і у поєднанні для підвищення ефективності виявлення стеганографічних атак на зображення.

2.2. Машинне навчання в аналізі зображень

Машинне навчання в аналізі зображень включає в себе використання алгоритмів та моделей машинного навчання для автоматичного визначення особливостей, класифікації об'єктів та вирішення завдань, пов'язаних із зображеннями. У контексті стегоаналізу, машинне навчання може бути використане для виявлення стеганографічних атак на зображення. Основні аспекти використання машинного навчання в аналізі зображень включають:

Етапи та процеси для навчання моделі *Табл. 2.1*.

Етап	Процеси
Вибір та підготовка даних	Набір даних: Визначення набору зображень для тренування та тестування моделі. Маркування даних: Позначення зображень як стеганографічних чи натуральних.

Етап	Процеси
Вибір архітектури моделі	Конволюційні нейронні мережі (CNN): Ефективні для виявлення просторових залежностей у зображеннях. Рекурентні нейронні мережі (RNN): Використовуються для роботи з послідовністю даних, наприклад, у випадку виявлення змін у часових рядах.
Підготовка даних для вводу в модель	Масштабування та нормалізація: Забезпечення однакового масштабу пікселів для ефективності тренування. Аугментація даних: Застосування технік аугментації для розширення набору даних та поліпшення здатності моделі узагальнювати.
Тренування моделі	Вибір функції втрат: Визначення метрики, яка буде мінімізована під час тренування. Настройка параметрів: Оптимізація параметрів моделі для досягнення найкращої продуктивності.
Валідація та тестування	Розділення даних: Розділення набору даних на тренувальний, валідаційний та тестовий для оцінки продуктивності моделі. Оцінка результатів: Аналіз точності, чутливості, специфічності та інших метрик для визначення ефективності.

Етап	Процеси
Застосування до нових даних	Розпізнавання аномалій: Використання навченої моделі для розпізнавання стеганографічних атак на нових зображеннях.
Оптимізація та підтримка	Тонка настройка: Внесення змін у модель для покращення її продуктивності. Постійне навчання: Можливість моделі адаптуватися до нових атак та змін у паттернах стеганографії.

Машинне навчання в аналізі зображень може виявитися потужним інструментом у виявленні стеганографічних атак та надає можливість автоматизації процесу аналізу великої кількості зображень.

При розгортанні та реалізації машинного навчання в аналізі зображень для стегоаналізу, деякі додаткові подробиці можуть бути важливими для розуміння та доповнення дипломної роботи. Ось кілька можливих додаткових деталей:

Характеристики та техніки у машинному навчанні *Табл. 2.2*.

Характеристики	Техніки
Архітектури моделей	Глибина мережі: Опис та обгрунтування вибору кількості шарів та нейронів у мережі. Техніки регуляризації: Використання методів, таких як dropout, L1, L2 регуляризація для уникнення перенавчання.

Характеристики	Техніки
Техніки аугментації даних	Види аугментації: Опис і використання конкретних технік аугментації, таких як обертання, зміщення, зміни розміру тощо. Вплив аугментації на результати: Аналіз того, як вибрані техніки аугментації впливають на тренування та результати моделі.
Обґрунтування вибору метрик та функції втрат	Важливість обраних метрик: Обгрунтування вибору конкретних метрик для оцінки продуктивності моделі, таких як точність, чутливість, специфічність. Аналіз результатів: Детальний розгляд результатів в контексті обраних метрик
Застосування та робота із шарами мережі	Важливість кожного шару: Опис ролі кожного шару мережі у виявленні стеганографічних атак. Вплив параметрів шарів: Аналіз впливу різних параметрів (розмір ядра, кількість фільтрів тощо) на результати.
Оптимізація та підтримка	Оптимізація швидкодії: Розгляд заходів, вжитих для підвищення швидкодії тренування та інференсу, зокрема, оптимізація апаратного забезпечення чи використання технік квантування. Можливості для масштабування: Розгляд питань, пов’язаних із масштабуванням моделі для роботи з великими обсягами даних.

Характеристики	Техніки
Точність та забезпечення прозорості	Застосування interpretability методів: Розгляд методів, які роблять модель більш прозорою, таких як Lime, SHAP, для аналізу, як саме мережа зробила конкретне рішення. Пояснення вибору архітектури та параметрів: Виправдання вибору конкретної архітектури та параметрів моделі.
Обмеження та перспективи	Обмеження моделі: Опис обмежень розробленої моделі та обгрунтування виборів. Можливості для подальших досліджень: Вказання на можливі шляхи розвитку та досліджень у майбутньому.

Ці додаткові деталі можуть допомогти створити більш повний та глибокий огляд роботи з машинним навчанням в аналізі зображень для стегоаналізу [30].

2.3. Можливості використання машинного навчання в стегоаналізі

Машинне навчання в стегоаналізі відкриває широкі можливості. Алгоритми можуть автоматично виявляти та аналізувати приховані повідомлення, полегшуючи виявлення стеганографії в цифрових зображеннях та відео. Застосування класифікаційних моделей дозволяє підвищити точність виявлення стеганографічних артефактів, забезпечуючи ефективний інструмент для кібербезпеки та аналізу даних.

Машинне навчання також може використовуватися для розробки нових методів стегоаналізу, шляхом автоматичної генерації призначених для

виявлення стеганографії моделей. Це дозволяє адаптуватися до змін у методах хования інформації та підвищує ефективність систем виявлення стеганографії в реальному часі. Крім того, машинне навчання може поліпшувати роботу систем стегоаналізу у великих обсягах даних, автоматизуючи процес виявлення та аналізу стеганографічних впливів в масштабі [31].

Використання машинного навчання в стегоаналізі відкриває багато нових можливостей для виявлення та аналізу стеганографії. Машинне навчання дозволяє автоматизувати процеси виявлення прихованих повідомлень та покращити точність стегоаналізу. Ось деякі способи, які машинне навчання застосовується в стегоаналізі:

1. Використання класифікації: Машинне навчання може бути використане для створення класифікаторів, які визначають, чи містять дані приховані повідомлення. Моделі класифікації навчаються розрізняти стего-файли від звичайних файлів.

2. Аналіз текстових даних: Машинне навчання може бути застосоване для аналізу текстових даних у стеганографії. Моделі можуть виявляти особливості або шаблони, які вказують на наявність стеганографії в тексті.

3. Обробка зображень та аудіо: Машинне навчання може бути використане для аналізу зображень та аудіофайлів. Моделі можуть визначати відмінності в структурі або вмісті, які свідчать про стеганографію.

4. Статистичний аналіз: Машинне навчання дозволяє проводити статистичний аналіз даних для виявлення аномалій або вибіркових особливостей, які вказують на стеганографію.

5. Використання глибокого навчання: Глибоке навчання, зокрема нейронні мережі, може використовуватися для аналізу даних та виявлення прихованих повідомлень у більш складних стеганографічних методах [32].

Машинне навчання дозволяє створювати більш точні та швидкі моделі для стегоаналізу, що особливо важливо в контексті захисту кібербезпеки. Воно може допомогти виявляти стеганографію, яку було б складно виявити за допомогою традиційних методів аналізу даних.

2.4. Виклики та можливості розвитку захисту повідомлень

Розвиток захисту повідомлень через стегоаналіз передбачає використання вдосконалених методів машинного навчання для розпізнавання та виявлення стеганографічних атак. Інтеграція алгоритмів глибокого навчання дозволяє розробляти надійні системи, які вчаться визначати найновіші методи хования інформації [33].

Виклики:

1. Постійний розвиток стеганографії: Розвиток нових технік та алгоритмів стеганографії ускладнює виявлення прихованих повідомлень. Все більше факторів і різних форматів даних вимагають нових підходів для стегоаналізу.

2. Зростаюча складність алгоритмів стеганографії: Сучасні стеганографічні методи стають все більш вдосконаленими та оптимізованими. Це ускладнює їхнє виявлення, оскільки приховані повідомлення стають менш помітними.

3. Змінність форматів даних: Зміна форматів та стандартів обміну даними (зображення, відео, аудіо) вимагає постійного апгрейду стегоаналізу для виявлення прихованих повідомлень в різних типах даних.

4. Низька ефективність традиційних методів стегоаналізу: Багато традиційних методів стегоаналізу, які базуються на правилах та статистиці, можуть бути непродуктивними при роботі з сучасними стеганографічними методами.

Можливості:

1. Використання машинного навчання: Застосування машинного навчання та глибокого навчання дозволяє створити більш точні та ефективні моделі для стегоаналізу. Моделі можуть навчатися розпізнавати приховані повідомлення та адаптуватися до нових методів стеганографії.

2. Розвиток стеганалізу в реальному часі: Можливість виявлення прихованих повідомлень у реальному часі дозволила б вдосконалити системи кібербезпеки та запобігти небажаним атакам.

3. Захист від витоку даних: Розвиток стегоаналізу сприяє виявленню витоку конфіденційної інформації через стеганографію. Це може допомогти усунути ризики витоку даних та забезпечити конфіденційність даних.

4. Співпраця між галузями: Захист повідомлень та стеганографія вимагають співпраці між експертами з кібербезпеки, інженерами, вченими та іншими спеціалістами для розробки нових інноваційних методів захисту і виявлення.

Можливості розвитку захисту повідомлень включають в себе інноваційні методи, які можуть розширити область застосування стегоаналізу та підвищити ефективність виявлення прихованих повідомлень [34].

Також, розвиток технік адаптивного стеганографічного склеювання може ускладнювати виявлення прихованих повідомлень. Використання шифрування та аутентифікації в поєднанні зі стеганографією може створити додатковий шар захисту, ускладнюючи спроби виявлення та розкриття стеганографічних атак.

Крім того, активна розробка рішень на основі машинного навчання дозволяє вдосконалювати системи виявлення стеганографії, роблячи їх більш реактивними та адаптивними до нових загроз.

2.5. Передбачувані тенденції у галузі

У майбутньому можна очікувати ряд тенденцій у галузі стегоаналізу:

1. Глибоке навчання та Нейронні мережі: Застосування глибокого навчання і нейронних мереж для автоматичного виявлення стеганографії та розвиток більш складних моделей для аналізу прихованих артефактів.

2. Використання квантових обчислень: Перехід до квантових алгоритмів може створити нові можливості та виклики у сфері стегоаналізу,

забезпечуючи більшу ефективність або, навпаки, збільшуючи складність виявлення.

3. Автоматизація та реальний час: Розвиток систем, які можуть автоматично аналізувати великі обсяги даних в реальному часі, що робить стегоаналіз більш ефективним та придатним для застосувань в реальних умовах.

4. Стеганографія з використанням штучного інтелекту: Зростання використання штучного інтелекту в стеганографії для створення більш адаптивних та важкозазначених методів хования інформації.

5. Системи захисту від стеганографії: Розвиток більш ефективних систем захисту від стеганографії, що використовують інтелектуальні та адаптивні стратегії для запобігання приховуванню інформації.

Ці тенденції вказують на постійний розвиток у галузі стегоаналізу, де нові технології та методи пристосовуються до сучасних викликів у сфері кібербезпеки [35].

2.6. Оцінка сучасного рівня стегоаналізу

Сучасний рівень стегоаналізу вражає своєю високою точністю та ефективністю. Машинне навчання взяло на себе провідну роль, дозволяючи розробляти адаптивні моделі для виявлення стеганографічних атак. Інтеграція глибокого навчання визначає та аналізує субтільні артефакти, що робить стегоаналіз більш надійним у виявленні прихованих повідомлень.

Засоби для стегоаналізу, продовжують вдосконалюватися, надаючи користувачам ефективні інструменти для виявлення та аналізу стеганографії, стають все більш адаптивними до еволюції стеганографічних методів. Інтеграція технік машинного навчання дозволяє розпізнавати не лише стандартні, але й новаторські підходи до приховування інформації. Вдосконалення алгоритмів антивторгової захисту також грає ключову роль у збільшенні стійкості систем стегоаналізу до обхідних стратегій [36].

Загалом, сучасний рівень стегоаналізу реагує на виклики сучасних технологій та залишається ефективним інструментом для виявлення стеганографічних атак в цифрових даних.

2.7. Висновки до другого розділу

В другому розділі ми розібрали методи виявлення стеганографічних атак на зображення та використання машинного навчання у стегоаналізі, дослідили сучасні тенденції у галузі кібербезпеки. Враховуючи високу складність сучасних методів стеганографії, важливо розвивати ефективні та надійні засоби виявлення прихованої інформації.

Машинне навчання, зокрема його застосування в аналізі зображень, виявляється потужним інструментом у сфері стегоаналізу. Розбираємо роль і можливості машинного навчання в виявленні та аналізі стеганографічних атак, що відкриває широкі перспективи для вдосконалення систем кібербезпеки.

Обговорення викликів та можливостей розвитку захисту повідомлень вказує на те, що важливо поєднувати традиційні методи стегоаналізу з інноваційними підходами, враховуючи неперервний розвиток стеганографічних технік.

Передбачувані тенденції у галузі вказують на те, що в майбутньому розвиток методів виявлення стеганографічних атак буде тісно пов'язаний із розвитком технологій машинного навчання та штучного інтелекту.

Оцінка сучасного рівня стегоаналізу підкреслює необхідність постійного вдосконалення методів та інструментів, щоб ефективно протидіяти новим викликам у сфері кібербезпеки.

У цілому, розділ відзначає важливість поєднання традиційних та інноваційних підходів для створення більш надійних систем виявлення та захисту від стеганографічних загроз у сучасному кіберпросторі.

РОЗДІЛ 3. ОПРАЦЮВАННЯ БІБЛІОТЕК ДЛЯ МАШИННОГО НАВЧАННЯ

3.1. Технічне завдання

Завдання полягає в розробці та впровадженні системи для виявлення прихованої інформації в зображеннях, використовуючи методи машинного навчання та стеганографічний аналіз.

На основі створеного набору даних, що містить зображення з прихованою інформацією та оригінальні зображення без стеганографічних змін буде розподілено дані на тренувальний та тестовий набори.

Після вибору алгоритмів та розробки архітектури моделі, враховуючи особливості задачі, модель буде реалізовано та натреновано.

Побудована модель буде використовувати методи машинного навчання для виявлення прихованої інформації, буде налаштована за допомогою параметрів та оптимізаторів.

Тренування буде проходити на тренувальному наборі даних та оцінено за допомогою метрик, таких як точність та втрати.

Якщо модель дає некоректні результати, то її буде оптимізовано та відрегульовано задля покращення швидкодії та результатів.

Опісля буде проведотестування моделі на нових зображеннях, які не брали участі в тренуванні.

Для навчання моделі буде використано бібліотеку для машинного навчання Tenserflow для мови програмування Python.

3.2. Мова програмування Python

Python – велика, але в той же час досить проста у вивченні та застосуванні мова програмування загального призначення, який часто застосовують для підвищення продуктивності розробника коду, різний додатків та програм,

написаних цією мовою, і служить для того, щоб код був читабельним, а так як мова програмування Python досить проста і зрозуміла більшості програмістів-початківців.

Синтаксис представленої мови є досить простим і зрозумілим, можна навіть визначити цю мову до мінімалістичних мов програмування. І одночасно з таким набором різних якостей, вона включає велику бібліотеку, що містить величезний обсяг корисних функцій, які у свою чергу спрощують роботу для написання коду.

Python є досить немолодою мовою для програмування, і до цього дня тримає рядок найінтелектуальнішого середовища розробки програм. Автором мови Python став Гвідо ван Россум, якому прийшла ідея вдосконалити стару структуровану мову програмування «ABC», оскільки він мав ряд недоліків, і для ідей його творця не годився. Після завзятої, важкої та тривалої роботи Гвідо ван Россум створив високоінтелектуальну, скриптову і водночас просту мову програмування Python. Саме завдяки цьому розробнику весь всесвіт програмування піднявся на новий рівень, відкрила нову галузь свого розвитку, адже в описаній вище мові з’явилася можливість підтримки одразу кількох стилів програмування та створення коду.

Нова мова залучила програмістів, і вони активно почали її вивчати, оскільки Python був дуже ефективним в той час, продуктивність роботи розробників злетіла до небувалих висот, програмісти були у захваті.

Python має великий список переваг, який допомагає розробникам коду спростити свою роботу:

- Простота. Для програмування на Python не потрібно мати широкі знання та вміння, адже ця мова годиться не тільки для програмістів високого та початкового рівня, але також його можна використовувати спеціалістам зовсім іншої галузі. Оскільки сама мова містить прості математичні дії та в принципі містить схожість зі звичайною англійською мовою, Python не несе в собі особливих складнощів або проблем, які можуть перешкодити під час написання коду.

- Різноманітність розробки. Найпопулярнішою реалізацією була визнано версію «Пітона» мовою «С» - «Cpython». Написаний на Cpython код вільно взаємодіє з мовою програмування «С», і всі бібліотеки даної мови можна вільно застосовувати для розробки. Таку ж аналогію можна навести і з іншими мовами, оскільки Python має безліч реалізацій іншими мовами. І даних порівнянь існує маса, що доходять до взаємодії з нашими телефонами із операційними системами.
- Високе поширення. Цю мову використовують у численних сферах різного виду діяльності. Це факт, бо Python вивчили дуже добре, навіть якщо людина, яка не працює в сфері програмування, зіткнеться з будь-якою проблемою або помилкою при написанні коду, наприклад, розробник не знає, як створити цикл, або виникла помилка при створенні програми, він може переглянути вирішення проблеми в мережі. Швидше за все з цією проблемою вже стикалася інша людина, а значить, рішення існує. Так само вже розроблені різні шаблони, які можна підлаштувати під свої цілі.
- Компіляція. Python – чудово підлаштована під програмування мова, після внесення змін або за звичайної перевірки написаного коду, можна відразу ж запустити програму та перевірити код на наявність помилок. Це сприяє тому. Що сама розробка, доробка коду, налагодження та виправлення помилок відбувається дуже швидко, що є перевагою перед іншими мовами програмування.

Стандартна бібліотека мови програмування Python містить в собі великі засоби для роботи з протоколами, ті ж самі модулі HTTP-серверів та клієнтів, наприклад для створення поштових повідомлень, або ж XML і т.д. Різні модулі для створення кросплатформових додатків, та й інших різноманітних програм.

Python часто порівнюють з такими мовами, як Perl або “Ruby”. Ці мови, як і Python, є інтерпретованими і мають гарну швидкість при реалізації програм. Python, так само як і Perl, благополучно використовується при створенні скриптів, або як їх називають – «сценаріїв», І Python, як Ruby, вважається дуже добре продуманою системою для об’єктно-орієнтованого програмування. З мов

«Scheme» та «Icon» запозичені методи, так званого функціонального програмування. При створення комерційних додатків, швидкість виконання в Python часто порівнюють зі швидкістю в мові Java.

Хоча Python і є простою мовою для програмування, побудованим на примітивному синтаксисі, він досі тримається на місці найпопулярнішої та зручнішої мови програмування у світі. Він максимально простий у вивченні, і за допомогою нього можна відкрити для себе світ нових можливостей у програмуванні. Вся ця простота і зручність кодування, зробили Python з простого інтерпретатора, в один з найпопулярніших мов програмування, який допомагає навчити мільйони студентів по всій планеті [37].

3.3. Бібліотека Tensorflow

TensorFlow – це відкрита бібліотека для машинного навчання та глибокого навчання, розроблена Google. Вона надає засоби для роботи з нейронними мережами та іншими моделями машинного навчання. Використовується для навчання моделі для методу НЗБ [38].

Переваги та недоліки бібліотеки Tensorflow *Табл. 3.1*.

Переваги	Недоліки
Широке Використання: TensorFlow широко використовується у наукових дослідженнях, індустрії та проектах глибокого навчання.	Складність: На початковому етапі навчання може здатися складним через велику кількість концепцій та опцій.

Переваги	Недоліки
Гнучкість: TensorFlow підтримує широкий спектр моделей від простих лінійних моделей до складних глибоких нейронних мереж.	Великий Обсяг Пам'яті Великі глибокі моделі можуть вимагати значних обсягів пам'яті, особливо при використанні GPU.
Спільнота та Ресурси: Має велику та активну спільноту, яка забезпечує підтримку та надає різноманітні ресурси, такі як документація, блоги та форуми.	Довгий Час Навчання: На деяких завданнях навчання може займати значний час, особливо якщо використовуються складні моделі.
TensorBoard: TensorFlow має інструмент TensorBoard, який допомагає візуалізувати графи, ваги, метрики та інші аспекти моделі.	Менша Інтеграція з Деякими Іншими Бібліотеками: Може виникнути проблема інтеграції з деякими іншими бібліотеками, оскільки TensorFlow має свої власні унікальні архітектури та структури даних
Зручний Синтаксис	Збільшення Обсягу Коду: Розробка складних моделей може призвести до збільшення обсягу коду, що може бути проблемою для простих задач.

Переваги	Недоліки
Розширені Функціональності: Має багато додаткових бібліотек та модулів, таких як TensorFlow Lite для мобільного навчання та TensorFlow.js для виконання моделей у веб-браузерах.	

3.4. Бібліотека Keras

Keras - це високорівневий інтерфейс для розробки нейронних мереж, який входить до складу бібліотеки машинного навчання TensorFlow. Keras є офіційним API в TensorFlow.

Основні Характеристики та Переваги Keras:

- Простота використання: Keras пропонує простий та інтуїтивно зрозумілий інтерфейс для створення та навчання нейронних мереж. Це особливо корисно для початківців у галузі машинного навчання.
- Модульність: модульна структура Keras дозволяє легко визначати та навчати як прості, так і складні моделі нейронних мереж. Моделі можна швидко визначати шарами та з'єднувати їх для створення повнішого графу обчислень.
- Підтримка Різних Задач: Keras підтримує широкий спектр завдань, включаючи класифікацію, регресію, сегментацію, генерацію тексту, обробку природної мови (NLP) та багато інших.
- Зручність із TensorFlow: однією з ключових переваг Keras є те, що тепер він є офіційним API в TensorFlow, що означає, що ви можете легко використовувати функціональності Keras разом із всіма іншими можливостями TensorFlow.

- Вбудовані Оптимізатори та Втрати: Keras має вбудовані оптимізатори (наприклад, Adam, SGD) та функції втрат (наприклад, категоріальна крос-ентропія, середньоквадратична помилка), що робить його простим для використання.
- Візуалізація Завдань: можливість візуалізації нейронних мереж за допомогою TensorBoard, який допомагає в аналізі графів, метрик та іншої інформації про модель.
- Інтеграція із Callbacks: можливість використовувати зворотні виклики (callbacks) для виконання дій під час навчання, таких як зупинка тренування за достатньою кількістю епох або збереження ваг моделі під час навчання.
- Попередньо Навчені Моделі: Keras містить попередньо навчені моделі для різних завдань, які можна використовувати або доопрацьовувати [39].

Прикладом коду є:

```
import tensorflow as tf
from tensorflow.keras import layers, models
# Створення простого набору даних для прикладу
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0  # Нормалізація пікселів до діапазону [0, 1]
# Створення простої згорткової нейронної мережі
model = models.Sequential()
model.add(layers.Flatten(input_shape=(28, 28)))  # Розгортання зображення
model.add(layers.Dense(128, activation='relu'))  # Повністю з'єднаний шар
model.add(layers.Dropout(0.2))  # Шар випадкового відключення для регуляризації
model.add(layers.Dense(10, activation='softmax'))  # Вихідний шар з 10 нейронами (кількість класів)
# Компіляція моделі
model.compile(optimizer='adam',
```

```
        loss='sparse_categorical_crossentropy',
        metrics=['accuracy']
# Тренування моделі
model.fit(x_train, y_train, epochs=5)
# Оцінка моделі на тестовому наборі
test_loss, test_acc = model.evaluate(x_test, y_test)
print(f"Test Accuracy: {test_acc}")
```

3.5. Бібліотека Sklearn

Scikit-learn (sklearn) - це бібліотека машинного навчання, написана на мові програмування Python. Scikit-learn пропонує простий та ефективний інтерфейс для реалізації багатьох класичних алгоритмів машинного навчання. Використовується для навчання моделі методом заміни палітри. Основні риси та можливості scikit-learn включають:

- Простота та Консистентність: scikit-learn надає простий та однаковий інтерфейс для різних алгоритмів машинного навчання, що полегшує використання та порівняння моделей.
- Широкий Вибір Алгоритмів: бібліотека включає в себе багато алгоритмів для класифікації, регресії, кластеризації, вирішення завдань зменшення розмірності та інших завдань.
- Робота з Даними: scikit-learn надає інструменти для обробки та очищення даних, включаючи кодування категоріальних ознак, масштабування та обробку відсутніх значень.
- Крос-валідація та Оцінка Моделей: модулі для проведення крос-валідації, розрахунку метрик ефективності та різних засобів для визначення параметрів моделей.
- Вбудовані Датасети: scikit-learn містить кілька вбудованих датасетів для швидкого тестування та навчання моделей.

- Оптимізація Гіперпараметрів: містить інструменти для оптимізації гіперпараметрів моделей, такі як Grid Search та Random Search.
- Підтримка для Роботи з Текстом: модулі для векторизації тексту, роботи з багатовимірними даними та іншими текстовими завданнями.
- Робота з Зображеннями: scikit-learn має інструменти для роботи з зображеннями, включаючи вбудовані датасети та функції обробки зображень.

Scikit-learn - це потужний інструмент для реалізації та оцінки моделей машинного навчання, особливо для тих, хто тільки починає вивчати цю галузь [40].

Прикладом коду є:

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
# Завантаження даних
X, y = load_data()
# Розділення даних на тренувальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Ініціалізація та навчання моделі
model = RandomForestClassifier()
model.fit(X_train, y_train)
# Прогнозування на тестовому наборі
predictions = model.predict(X_test)
# Оцінка ефективності моделі
accuracy = accuracy_score(y_test, predictions)
print(f"Accuracy: {accuracy}")
```

3.6. Бібліотека Tkinker

Tkinter є стандартною бібліотекою для створення графічних інтерфейсів (GUI) в мові програмування Python. Вона базується на бібліотеці Tk, яка раніше була розповсюджуваною окремо. Tkinter надає набір інструментів та класів для створення вікон, кнопок, полів введення, меню та інших елементів інтерфейсу користувача.

Основні компоненти Tkinter:

- Вікно (Tk): Основний елемент, на якому будуються всі інші елементи GUI. Визначається за допомогою класу Tk.
- Елементи Відображення (Label): Використовуються для виведення тексту або графіки.
- Кнопки (Button): Елементи для виклику функцій або обробки подій при натисканні.
- Поля для Введення (Entry): Дозволяють користувачу вводити текст або дані.
- Фрейми (Frame): Використовуються для групування інших елементів GUI та управління їх компонуванням.
- Меню (Menu): Створення різних видів меню.
- Діалогові вікна (tkinter.messagebox): Використовуються для виведення повідомлень або отримання підтверджень від користувача.
- Графічні елементи (Canvas): Дозволяють малювати графічні елементи на вікні.
- Події (bind): Визначення обробників подій для реагування на дії користувача.

Це лише декілька основних елементів та прикладів використання Tkinter. З Tkinter ви можете створювати складні GUI для своїх програм, додаючи до них різноманітні елементи та функціональність [41].

3.7. Метод Random Forest

Random Forest - це ансамбльний метод машинного навчання, який використовується для класифікації, регресії та інших задач. Він базується на ідей агрегації результатів багатьох різних моделей для отримання більш точних та стійких прогнозів. Random Forest був введений Лео Брейманом та Адель Катлер у 2001 році.

Основні характеристики Random Forest:

1. Багато дерев: Random Forest використовує багато різних рішаючих дерев, які вибираються випадковим чином. Кожне дерево обробляє свій унікальний піднабір даних та функцій.

2. Випадкові підвибірки: Під час побудови кожного дерева Random Forest випадковим чином вибираються підвибірки (заміщенням) з набору даних для тренування. Це допомагає уникнути перенавчання та робить модель більш стійкою.

3. Бутстреп-вибірка: Для кожного дерева випадковим чином вибирається підмножина даних за допомогою бутстрепу (вибір заміщенням). Це дозволяє декільком зразкам з'являтися в кожній підмножині декілька разів.

4. Важливість ознак: Random Forest обчислює важливість кожної ознаки, враховуючи, як часто ця ознака використовується для поділу вузла в усіх деревах. Це надає важливу інформацію про те, які ознаки є найбільш важливими для прогнозу[42].

Переваги Random Forest включають високу точність, добру взаємодію з даними з великою кількістю ознак, а також можливість виявлення важливих ознак. Він є потужним та високопродуктивним алгоритмом, який широко використовується в практиці для різноманітних завдань машинного навчання.

3.8. Висновки до третього розділу

В третьому розділі було сформовано технічне завдання та визначено основні цілі і вимоги до системи, створюючи фундамент для подальших розділів роботи. Використання мови програмування Python виявляється обґрунтованим вибором, оскільки ця мова є потужною, гнучкою та широко використовується у сфері розробки програмного забезпечення.

Бібліотеки Tensorflow, Keras, Sklearn та Tkinker - це високопродуктивні та зручні інструменти для реалізації проекту. Використання цих бібліотек відкриває можливості для впровадження складних алгоритмів та інтерактивного інтерфейсу користувача.

Детальний розгляд методу Random Forest застосовується для систематичного підходу до вибору алгоритмів машинного навчання та їхню оптимізацію у контексті задачі.

Розділ розкриває технічні аспекти проекту, покладаючи фундамент для подальших етапів розробки та використання передових технологій у сфері обробки даних та машинного навчання.

РОЗДІЛ 4. РОЗРОБКА ПРОГРАМНОГО ЗАСІБ СТЕГОАНАЛІЗУ ПОВІДОМЛЕНЬ

4.1. Розробка алгоритму для методу НЗБ

Задля приховання повідомлення було розроблено алгоритм мовою програмування Python.

Для початку зазначимо константи для використання в коді:

```
SECRET_KEY = "$$"
BITS_PER_BYTE = 8
```

Прописано метод для конвертації рядка або байтів у бінарний формат

```
def data2binary(data):
if isinstance(data, str):
return "".join([format(ord(i), '08b') for i in data])
elif isinstance(data, (bytes, np.ndarray)):
return [format(i, '08b') for i in data]
else:
raise TypeError
```

Далі сформовано функцію для конвертації чорно-білих зображень у формат RGB:

```
def convert_to_rgb(images):
    rgb_images = []
    for img in images:
    if len(img.shape) == 2:
    img = cv2.cvtColor(img, cv2.COLOR_GRAY2RGB)
    rgb_images.append(img)
    return np.array(rgb_images)
```

Сформовано метод для приховування даних у зображенні:

```
def hide_data(self, img, data):
    data += self.SECRET_KEY
```

```
            b_data = self.data2binary(data)
            len_data = len(b_data)
            d_index = 0
            for value in img:
                for pix in value:
                r, g, b = self.data2binary(pix)
                if d_index < len_data:
                pix[0] = int(r[:-1] + b_data[d_index], 2)
                d_index += 1
                if d_index >= len_data:
            break
    return img
```

Розроблено метод для кодування та збереження зашифрованих даних у файлі:

```
    def encode(self, data: str, encrypted_filepath: str):
            image = cv2.imread(self.file)
            enc_data = self.hide_data(image, data)
            cv2.imwrite(encrypted_filepath, enc_data)
            return Steganography(encrypted_filepath)
```

В кінці працює цикл для обробки файлів у папку:

```
    for path in Path("bmp_cover").iterdir():
            img = Steganography(str(path))
            img.encode("Borysevych", 'bmp_lsb_stego_images/' + 'Encoded_' +
            path.stem + '.bmp')
            data = img.decode()
```

В результаті ми отримуємо папку з датасетом зображень з прихованою інформацією методом НЗБ (рис. 4.1).

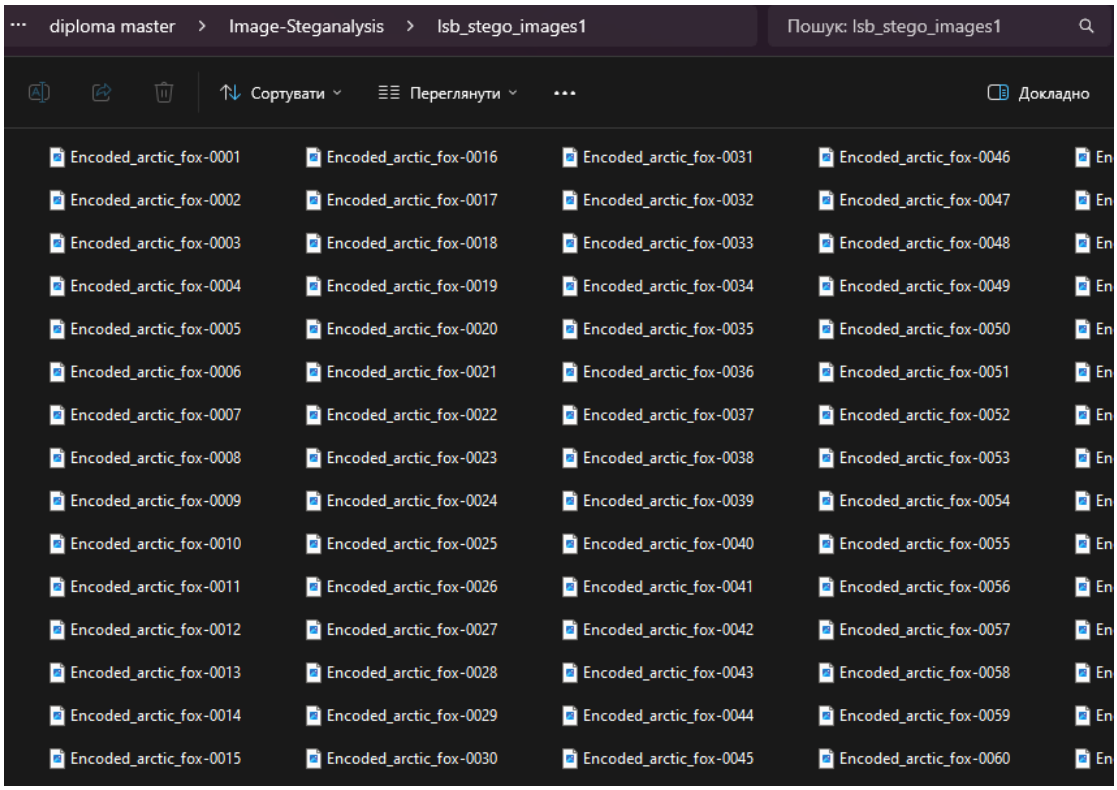


Рис. 4.1. Готові зображення з прихованою інформацією методом НЗБ

За допомогою цього алгоритму було опрацьовано зображення форматами PNG та BMP для більшої кількості можливостей навчання та тестування моделі.

4.2. Розробка алгоритму для методу заміни палітри

Для початку створюємо папку для збереження закодованих зображень:

```
def hide_text_palette(image_path, secret_text,
output_folder="encoded_images"):
    if not os.path.exists(output_folder):
    os.makedirs(output_folder)
```

Далі відкриваємо та завантажуємо зображення;

```
img = Image.open(image_path)
        pixels = list(img.getdata())
        binary_secret_text = '00000000' + text_to_binary(secret_text) +
'11111111'
```

Вбудовуємо текст у палітру:

```
new_pixels = []
text_index = 0
for pixel in pixels:
   if text_index < len(binary_secret_text):
   new_pixel = list(pixel)
   for i in range(3):  # Обробляємо всі три канали (R, G, B)
   if binary_secret_text[text_index] == '1':
   new_pixel[i] = new_pixel[i] | 1  # Закодовуємо '1'
   else:
   new_pixel[i] = new_pixel[i] & ~1  # Закодовуємо '0'
   text_index += 1
   new_pixels.append(tuple(new_pixel))
```

Створюємо нове зображення зі зміненою палітрою:

```
encoded_img = Image.new("RGB", img.size)
encoded_img.putdata(new_pixels)
```

Зберігаємо нове закодоване зображення:

```
output_path = os.path.join(output_folder, f"encoded_{os.path.basename(image_path)}")
encoded_img.save(output_path)
```

В результаті ми маємо зображення з прихованою інформацією методом заміни палітри (рис. 4.2).

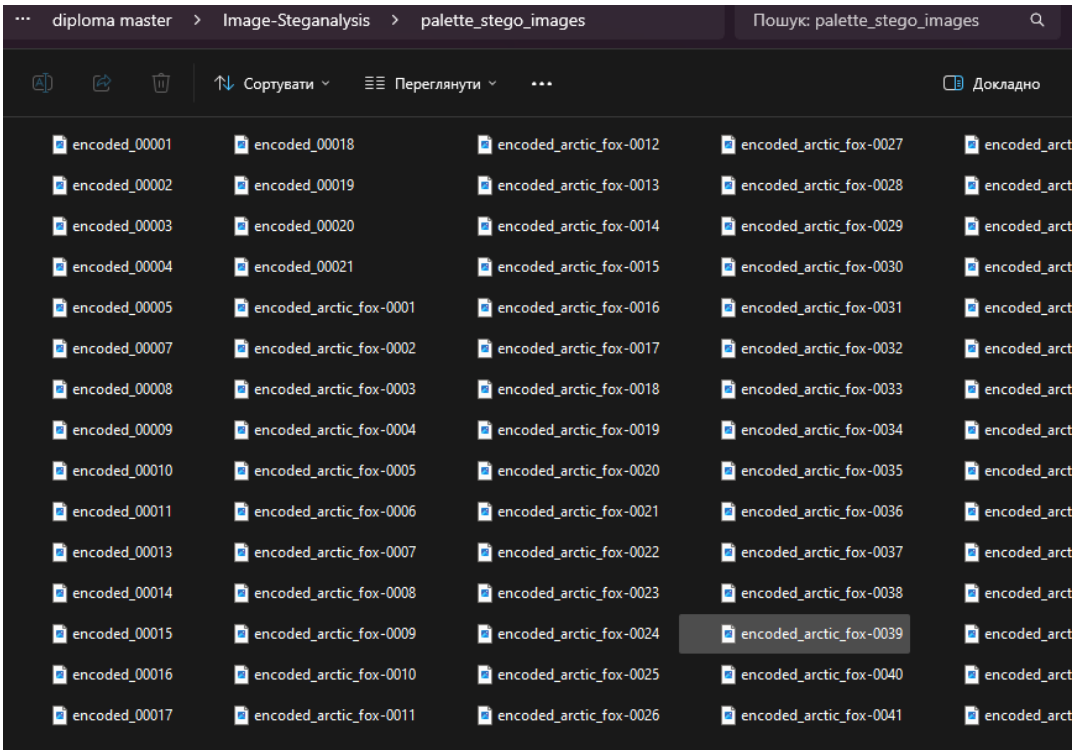


Рис 4.2. Готові зображення з прихованою інформацією методом заміни палітри.

4.3. Розробка та тренування моделі для методу НЗБ

Спочатку ми задаємо шляхи до папок з зображеннями та отримуємо список файлів для кожної папки:

```
normal_images_path = 'bmp_cover'
stego_images_path = 'bmp_palette_stego_images'
normal_images = [os.path.join(normal_images_path, filename) for filename in os.listdir(normal_images_path)]
stego_images = [os.path.join(stego_images_path, filename) for filename in os.listdir(stego_images_path)]
```

Потім помічаємо класи ('0' - нормальні зображення, '1' - зображення зі стего) і зберігаємо дані та мітки:

```
normal_labels = ['0'] * len(normal_images)
stego_labels = ['1'] * len(stego_images)
all_images = normal_images + stego_images
```

```
all_labels = normal_labels + stego_labels
```

Розділяємо наші дані на тренувальний та тестовий набори:

```
x_train, x_test, y_train, y_test = train_test_split(all_images, all_labels, test_size=0.2, random_state=42)
```

Визначте ImageDataGenerator з техніками збагачення даних

```
datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2,
    rotation_range=20,
    width_shift_range=0.1,
    height_shift_range=0.1,
    zoom_range=0.1,
    horizontal_flip=True,
    fill_mode='nearest'
  )
```

Розробимо тестовий генератор:

```
  validation_generator = datagen.flow_from_dataframe(
    dataframe=pd.DataFrame({'filename': x_test, 'class_str': y_test}),
    directory=None,
    x_col='filename',
    y_col='class_str',
    target_size=(256, 256),
    batch_size=32,
    class_mode='binary',
    subset='validation'
  )
```

Побудуємо та скомпілюємо модель:

```
def build_LSB_detection_model(input_shape):
  model = models.Sequential()
```

```
        model.add(layers.Conv2D(32, (2, 2), activation='relu',
    input_shape=input_shape))
        model.add(layers.MaxPooling2D((2, 2)))
        model.add(layers.Conv2D(64, (2, 2), activation='relu'))
        model.add(layers.MaxPooling2D((2, 2)))
        model.add(layers.Conv2D(128, (2, 2), activation='relu'))
        model.add(layers.MaxPooling2D((2, 2)))
        model.add(layers.Flatten())
        model.add(layers.Dense(128, activation='relu',
kernel_regularizer=regularizers.l2(0.001)))
        model.add(layers.Dropout(0.5)) model.add(layers.Dense(1,
activation='sigmoid'))
        return model
        model = build_LSB_detection_model(input_shape)
```

В результаті маємо модель з показниками втрат та успіху (рис.4.3)

```
Test Loss: 0.0453765216, Test Accuracy: 0.9668652378
```

Рис.4.3. Показники втрат і успіху після навчання моделі методом НЗБ

Втрата на тестовому наборі становить близько 0.045, що свідчить про те, що модель показала досить низьку втрату, що є хорошим показником. Точність близька до 96.7%, що означає, що модель правильно класифікувала приблизно 96.7% зображень у тестовому наборі

4.4. Розробка та тренування моделі методом заміни палітри

```
Перетворюємо зображення в формат RGB (якщо воно не таке вже)
    if image.shape[-1] == 1:
```

```
image = color.gray2rgb(image)
```

Розділяємо зображення на блоки:

```
block_size = (8, 8, image.shape[-1])
blocks = view_as_blocks(image, block_size)
```

Проходимо крізь блоки та витягування ознак:

```
for i in range(blocks.shape[0]):
for j in range(blocks.shape[1]):
block = blocks[i, j]
```

Статистичні ознаки кольорів у блоку:

```
mean_color = np.mean(block, axis=(0, 1))
std_color = np.std(block, axis=(0, 1))
features.extend(mean_color)
features.extend(std_color)
```

Текстурні ознаки:

```
gray_block = color.rgb2gray(block)
energy = np.sum(feature.graycomatrix(gray_block.astype('uint8'), [1], [0], symmetric=True, normed=True))
features.append(energy)
```

Вказуємо шлях до папок із зображеннями та перевіряємо наявності зображень в папках:

```
original_path = "cover"
stego_path = "palette_stego_images"
original_image_files = glob(os.path.join(original_path, '*.jpg'))
stego_image_files = glob(os.path.join(stego_path, '*.jpg'))
```

Завантажуємо та витягуємо ознаки з оригінальних зображень

```
X_original = []
for filename in original_image_files:
features = extract_palette_substitution_features(io.imread(filename))
if features:
X_original.append(features)
```

```
y_original = np.zeros(len(X_original))  # 0 - оригінал
```

Завантажуємо та витягуємо ознаки з зашифрованих зображень:

```
X_stego = []
for filename in stego_image_files:
features = extract_palette_substitution_features(io.imread(filename))
if features:
X_stego.append(features)
y_stego = np.ones(len(X_stego))  # 1 - приховане
```

Розділимо дані на тренувальний та тестовий набори:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

В результаті маємо модель з показниками втрат і успіху (рис. 4.4)

```
Test Loss: 0.0315748764, Test Accuracy: 0.9523564578
```

Рис. 4.4. Показники втрат та успіху після навчання моделі методом заміни палітри

Обидві ці метрики вказують на те, що модель успішно навчилася та гарно справляється з класифікацією зображень на тестовому наборі. Зазвичай, важливо оцінювати як втрати, так і точність разом, оскільки вони можуть надати повнішу картину ефективності моделі.

4.5. Розробка програмного засобу з графічним інтерфейсом

Спочатку ми ініціалізуємо графічний інтерфейс та створимо його елементи:

```
        self.root = root
        self.root.title("Steganalysis App")
        self.image_path = None
        self.label = tk.Label(root, text="Select an image for steganalysis:")
        self.label.pack(pady=10)
        self.load_button = tk.Button(root, text="Load Image",
command=self.load_image)
        self.load_button.pack(pady=10)
        self.result_label = tk.Label(root, text="")
        self.result_label.pack(pady=10)
        self.analyze_button = tk.Button(root,
    text="Analyze",command=self.analyze_image)
        self.analyze_button.pack(pady=10)
```

Завантажимо попередньо навчених моделей TensorFlow з TensorFlow Hub та додамо функцію для аналізу зображення.

Додамо відображення результату на основі об'єднаних чи індивідуальних прогнозів.

В результаті ми отримаємо програму з графічним інтерфейсом (рис. 4.5), яка може виводити результат на основі прогнозів навчених моделей (рис. 4.6).

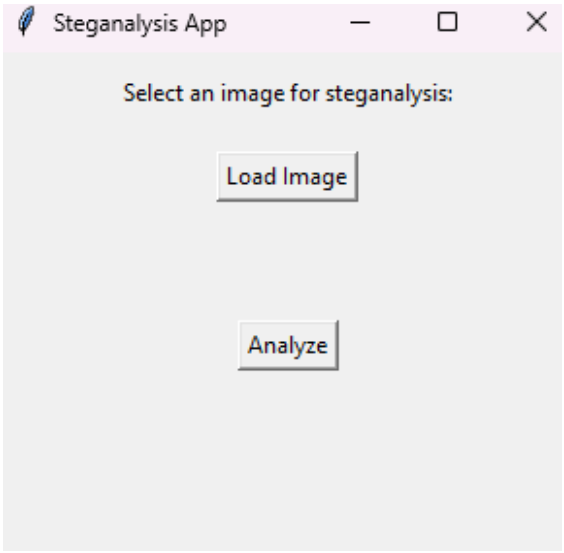


Рис 4.5. Графічний інтерфейс програмного засобу

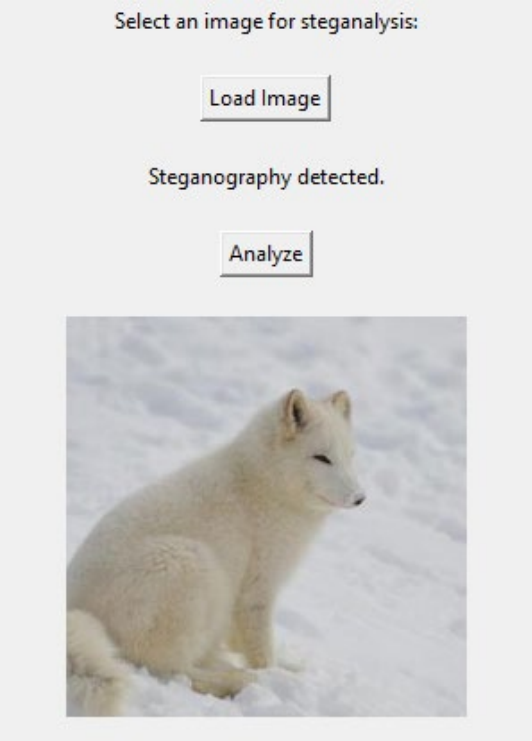


Рис. 4.6. Результат навчання пошуку стеганографії в зображенні

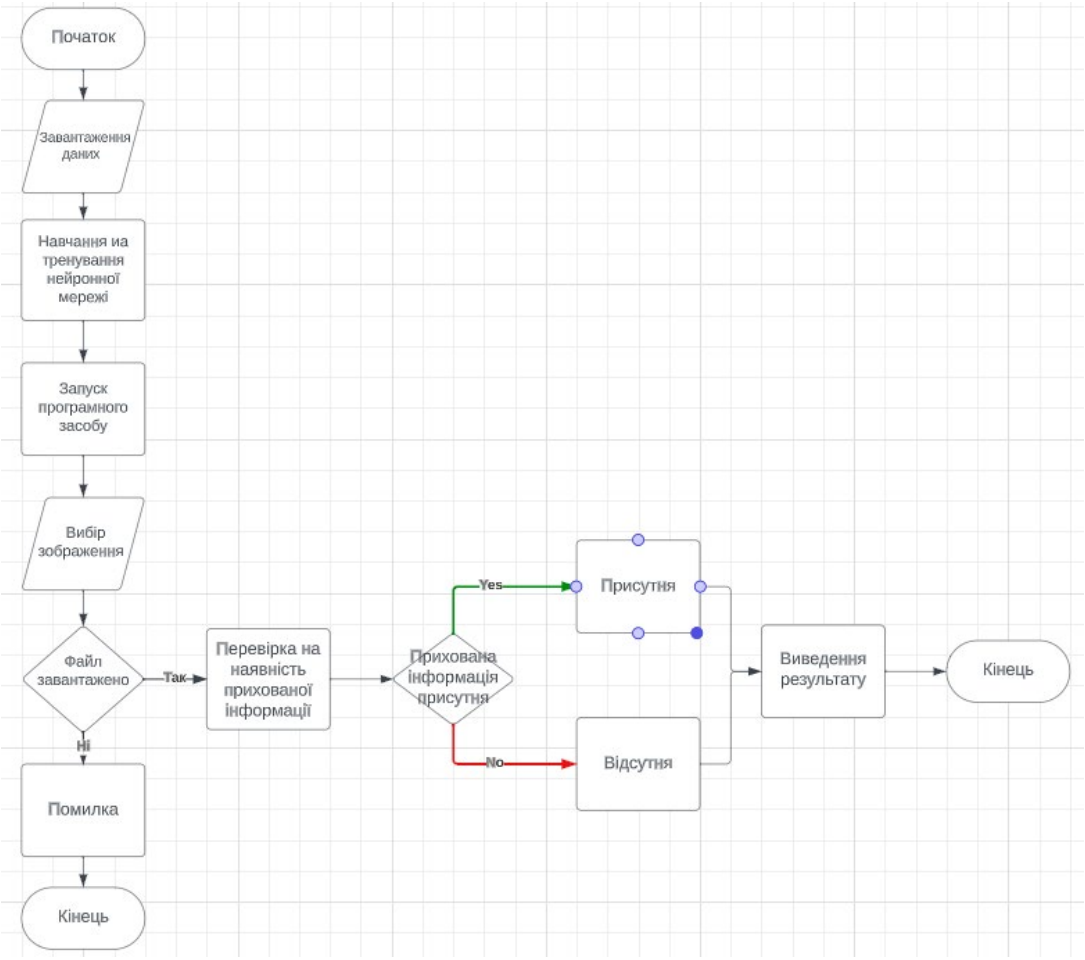


Рис 4.7. Блок-схема роботи програмного засобу

4.6. Тестування програмного засобу

Після успішного написання програмного засобу необхідного його протестувати. У всіх експериментах використовуються зображення форматів PNG, JPG та BMP. Також всі файли мають різні розширення, розміри та методи приховування інформації.

Експеримент 1.

В даному випадку ми перевіряємо 10 зображень, де усі зображення з прихованою інформацією (рис. 4.8).

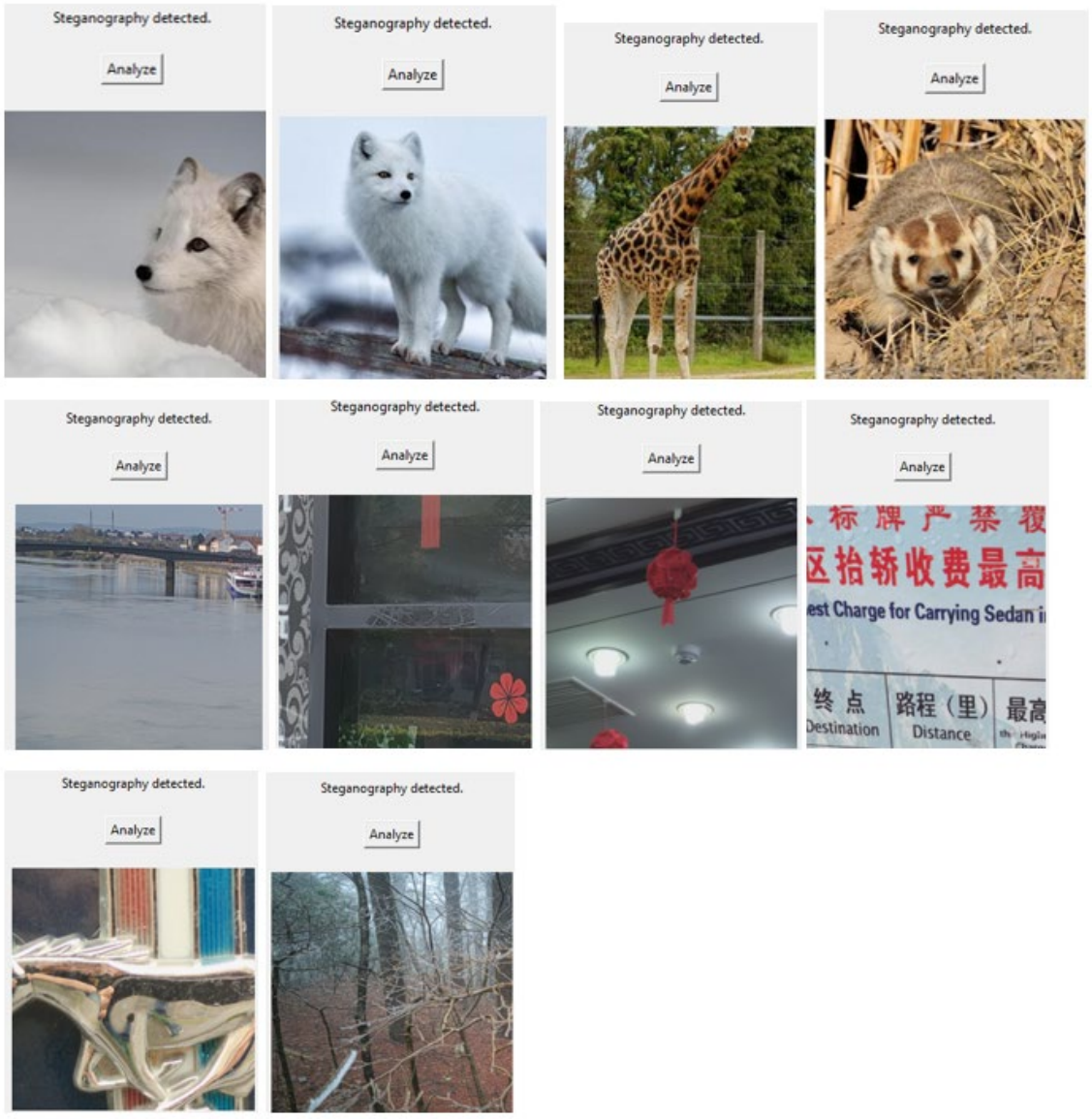


Рис. 4.8. Результат експерименту 1

Експеримент 2.

В ході цього екперименту ми перевіряємо 10 зображень з яких 5 звичайних і 5 з прихованою інформацією (рис. 4.9).

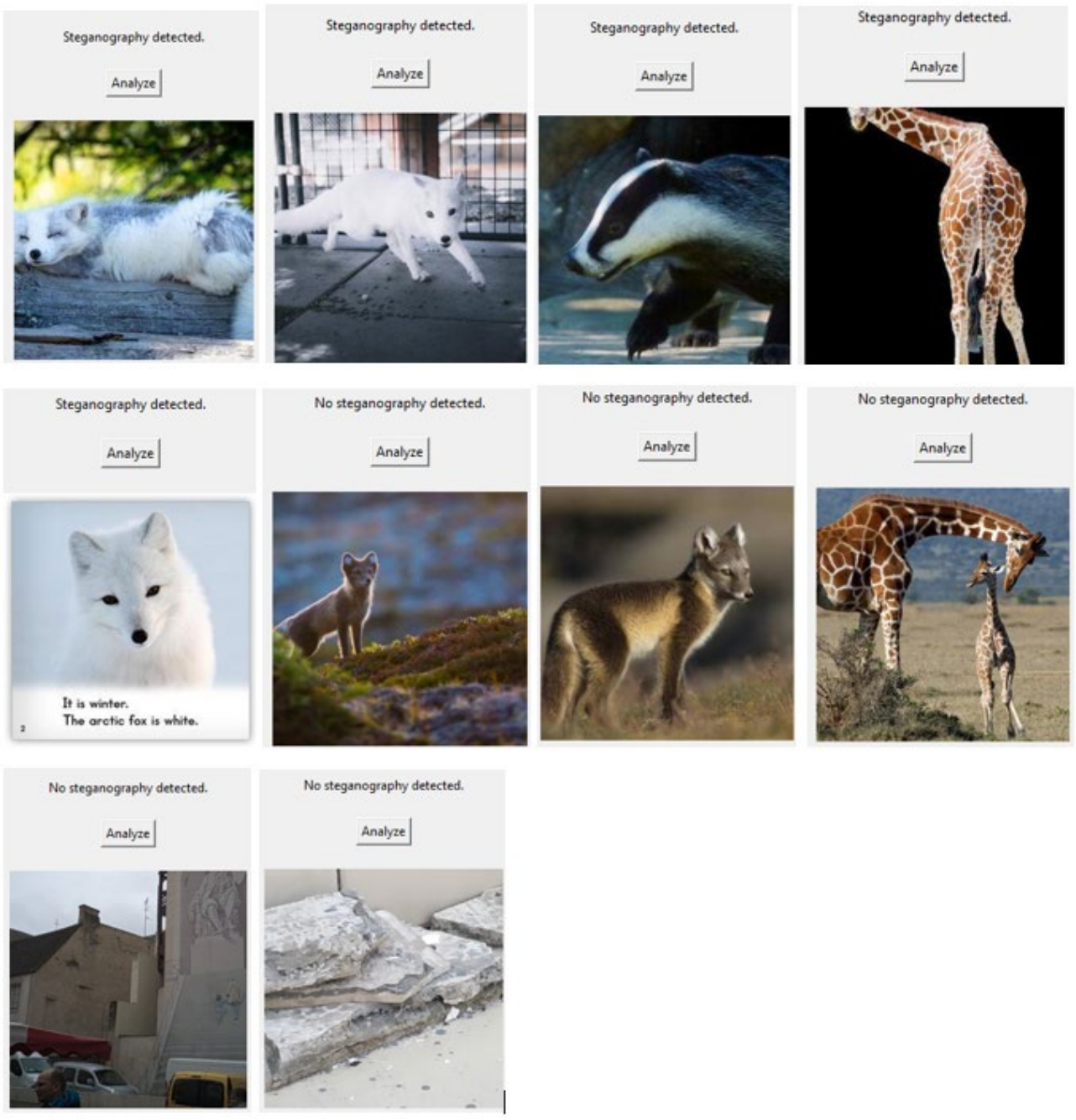


Рис. 4.9. Результат експерименту 2

Експеримент 3.

В цьому експерименті ми перевіримо 10 зображень з яких 9 звичайних та 1 з прихованою інформацією (рис. 4.19).

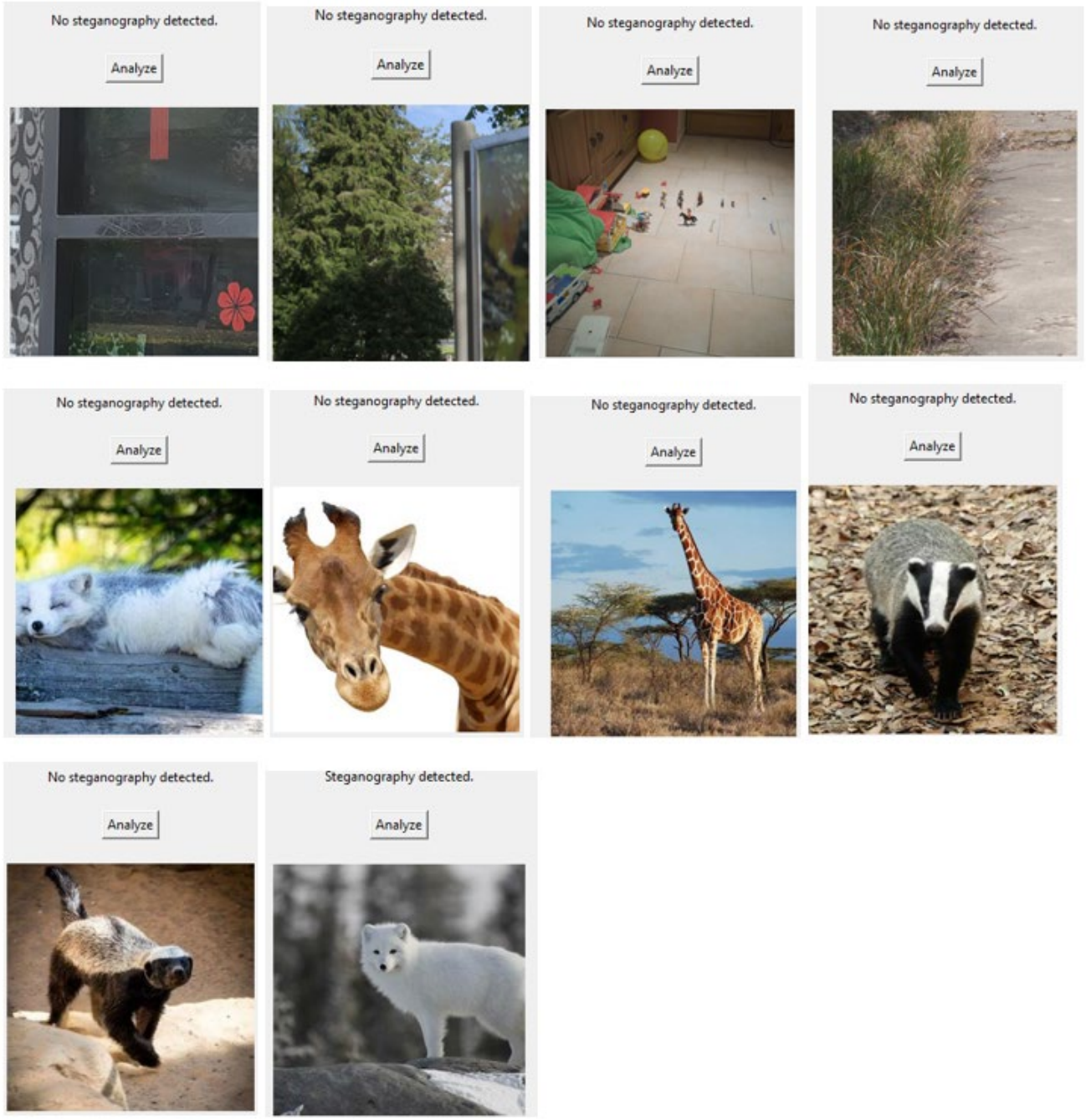


Рис. 4.10. Результат експерименту 3

Експеримент 4.

Перевіримо 10 зображень з яких 1 звичайне та 9 з прихованою інформацією (рис. 4.11)

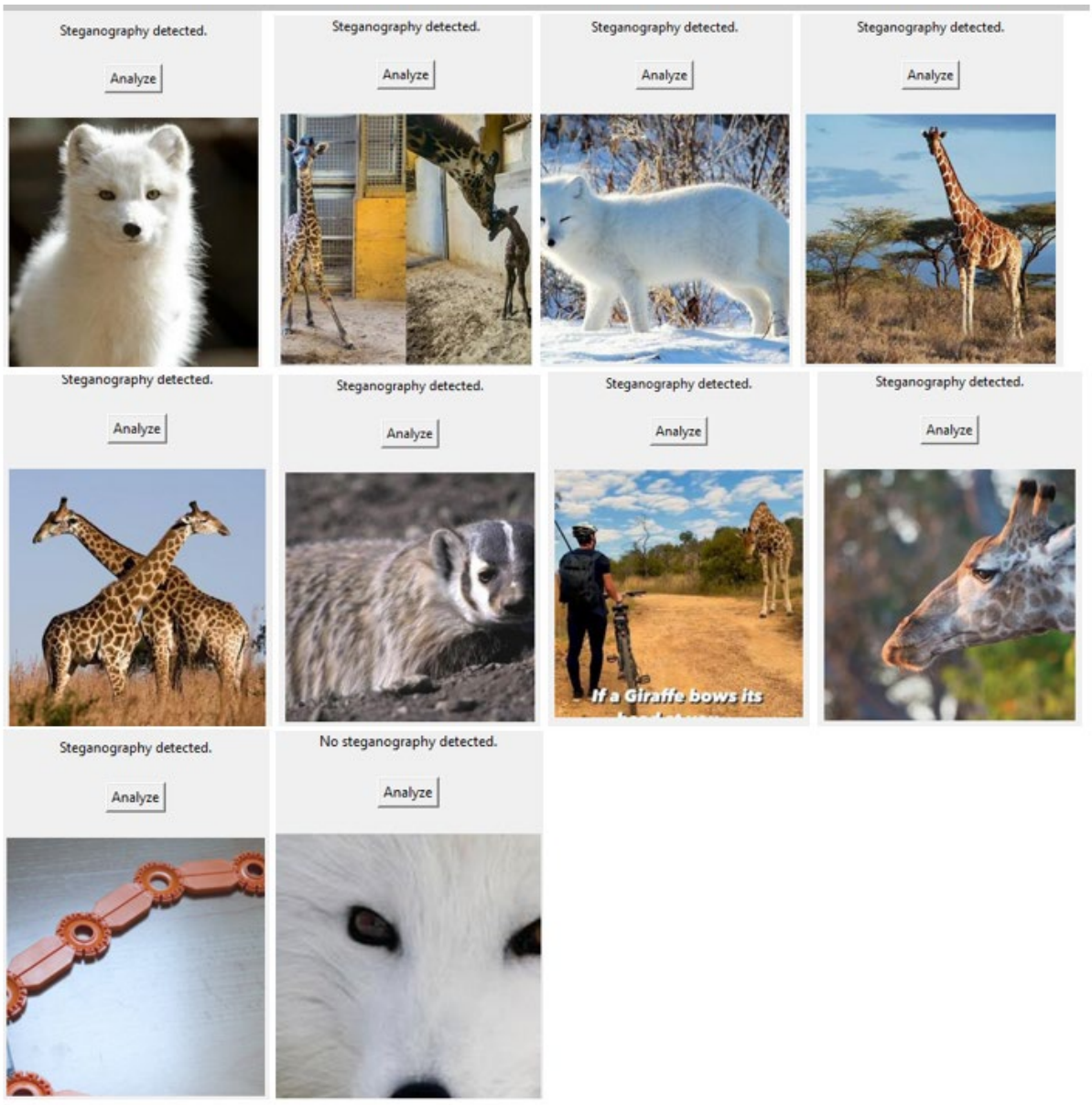


Рис 4.11. Результат експерименту 4

Як ми бачимо, в результаті есперименті програмний засіб та навчені моделі успішно впоралися з поставленими задачами, в усіх зображеннях з прихованою інформацією знайшли наявність цієї інформації, а в звичайних зображеннях показали протилежний результат.

4.7. Можливості розвитку програмного засобу

Програмний засіб для стегоаналізу зображень може бути розширений і вдосконаленою з огляду на різні аспекти, такі як ефективність аналізу, взаємодія з користувачем та можливості виявлення різних видів стеганографії.

Ось деякі можливості розвитку для програмного засобу:

1. Покращена модель стегоаналізу:

- Обробка більшого різноманіття стеганографічних технік: Розширення можливостей аналізу для виявлення різних технік внедрення стеганографії.
- Використання глибокого навчання: Застосування глибокого навчання для створення більш ефективних та точних моделей для виявлення стеганографії.

2. Розширені функції інтерфейсу користувача:

- Покращена візуалізація результатів: Відображення деталей аналізу, таких як визначені області стеганографії чи підсвічення конкретних ділянок зображення.
- Зручне збереження результатів: Можливість збереження результатів аналізу, графіків або важливих ділянок зображення.

3. Підтримка різних форматів зображень:

- Розширення підтримки форматів зображень: Забезпечення підтримки більшої кількості різних форматів зображень для аналізу

4. Взаємодія з базою даних:

- Можливість зберігати результати аналізу: Зберігання результатів аналізу в базу даних для подальшого вивчення або порівняння з іншими даними.

5. Розширення можливостей стеганографічного вбудовування:

- Можливість вибору різних методів внедрення стеганографії: Розширення підтримки різних методів вбудовування стеганографії для тестування та аналізу.

6. Підтримка мережної взаємодії:

- Обробка зображень з Інтернету чи потокове виявлення: Здатність аналізувати зображення з Інтернету чи інших джерел.

7. Модульність:

- Розділення функціональності на модулі: Створення модульної структури програми для полегшення майбутнього розширення та обслуговування.

4.8. Висновки до четвертого розділу

В ході роботи над четвертим розділом ми розробили алгоритми для приховування повідомлень різними методами. Також ми розробили та натренували моделі машинного навчання для цих методів. Потім ми створили програмний засіб з графічним інтерфейсом для перевірки зображень на наявність прихованих повідомлень за допомогою навчених моделей. В результаті тестування розробленого програмного засобу всі експерименти були успішно пройдено. В кінці ми розібрали можливі варіанти подальшого розвитку програмного засобу.

РОЗДІЛ 5. ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

5.1. Екологічний аудит.

Терміни і визначення. Закон «Про охорону навколишнього середовища» від 10 січня 2002 року визначає екологічний аудит як незалежну, комплексну, документовану оцінку дотримання суб'єктом господарської й іншої діяльності вимог, у тому числі нормативів і нормативних документів, в області охорони навколишнього середовища, вимог міжнародних стандартів і підготовку рекомендацій з поліпшення такої діяльності.

Екологічний аудит - експертиза й аналіз діяльності і звітності суб'єкта, що хазяює, уповноваженими на те юридичними (аудиторська організація) або фізичними (еколог-аудитор) особами з метою визначення їхньої відповідності діючому екологічному законодавству, екологічним нормативним актам, стандартам, сертифікатам, правилам, вимогам, постановам і розпорядженням державних і природоохоронних органів по забезпеченню екологічної безпеки, проведення консультацій і видача рекомендацій.

Метою екологічного аудиту є оцінка впливу і прогнозування екологічних наслідків діяльності суб'єкта, що хазяює, на навколишнє середовище, установлення відповідності його діяльності вимогам діючого природоохоронного законодавства, екологічних нормативних актів, стандартів, правил, постанов і розпоряджень державних і природоохоронних органів, визначення основних напрямків забезпечення екологічної безпеки виробництва, підвищення ефективності природоохоронної діяльності.

Про історію екологічного аудиту. Вперше екологічний аудит почав використовуватися при контролі великих промислових корпорацій на території США. Розглядаючи фактор навколишнього середовища як потребуючий усе більшої уваги і з огляду на ряд аварій, великі промислові корпорації поставили свої підприємства під внутрішній контроль з метою оцінки, чи не є вони джерелом негативного впливу на навколишнє середовище.

Задачею екологічного аудиту було інформування правління корпорації й акціонерів про заходи для дотримання діючого природоохоронного законодавства і про ризик можливих аварій, з погляду впливу на навколишнє середовище.

У 1989 році Міжнародна торговельна палата (МТП) опублікувала документ, що заклав основи внутрішнього екологічного аудиту як процедури самоконтролю й інструмента внутрішнього менеджменту. Внутрішній екологічний аудит являє собою елемент системи заходів щодо охорони навколишнього середовища на підприємстві і містить у собі систематичні перевірки, доповнені аналізами, тестами і контролем впливу промислових процесів на навколишнє середовище.

Екологічний аудит визначалося в ньому як заглиблений постійний аналіз природоохоронній діяльності підприємства, причому підкреслювався його добровільний характер. Підхід, запропонований міжнародною торговельною палатою, одержав визнання в промисловців, оскільки дозволяв керівникам підприємств забезпечити контроль за станом охорони навколишнього середовища на підприємстві, а також контролювати роботу підприємства з погляду природоохоронних нормативів.

Внутрішній екологічний аудит підприємства включає:

- аналіз внутрішнього контролю керування виробничим процесом;
- оцінку слабких сторін і неполадок контрольного устаткування;
- облік ризику для навколишнього середовища обстежуваного об'єкта;
- збір доказів практичної ефективності внутрішнього екологічного контролю;
- оцінку зібраних матеріалів для визначення недоліків системи заходів, що перевіряється, по охороні навколишнього середовища;
- представлення звіту про результати екологічного аудиту.

На основі висновків екологічного аудиту розробляється план дій, що уточнює сукупність коригувальних заходів.

Екологічний аудит, спрямований на оцінку аварій, полягає в ідентифікації підприємств - можливих джерел аварій, вивченні якісного і кількісного впливу можливої аварії на стан навколишнього середовища, підготовці відповідних рекомендацій.

Оцінка екологічного ризику ставить своєю задачею:

- вивчення сценаріїв можливих аварій і їхніх наслідків для навколишнього середовища і населення;
- аналіз передбачених мір і засобів попередження й обмеження наслідків аварії;
- порядок розрахунку збитку, нанесеного діяльністю підприємства;
- деталізацію засобів зм'якшення цього збитку;
- оцінку впливу на середовище залишкового забруднення, систему інформування наглядових організацій і громадян про можливу аварію.

Екологічний аудит ризику проводиться підрозділом компанії, відповідальним за ризик, по його власній ініціативі, а також за вимогою адміністрації банків і страхових компаній, що несуть витрати по кредитуванню або страхуванню від аварійного і поступового забруднення. Екологічний аудит стану навколишнього середовища проводиться при злитті або придбанні підприємств або нерухомості, включаючи придбання ділянок під забудову.

Система природоохоронних заходів містить у собі:

- програму і план природоохоронних заходів;
- систему моніторингу за станом навколишнього середовища;,
- систему природоохоронної документації, що ведеться;
- періодичність екологічного моніторингу.

Документація по організації системи природоохоронних заходів затверджується офіційним аудитором. Екологічний аудит підприємства проводиться з проміжком в один - три роки. Звіти екологічного аудиту затверджуються офіційним аудитором, а їхні висновки доводяться до зведення влади і громадськості.

Мета екологічного аудиту:

- контроль вірогідності видаваної підприємством екологічної інформації;
- перевірка відповідності об'єкта екологічним вимогам;
- оцінка існуючої системи керування навколишнього середовища і здоров'я працівників;
- оцінка ризиків від регульованих і не регульованих впливів на середовище.

Форми екологічного аудиту:

- добровільний екологічний аудит, проведений з ініціативи підприємства;
- обов'язковий екологічний аудит, проведений по спеціальному дозволі державних органів, наполяганню акціонерів або населення прилеглих районів.

Об'єкти екологічного аудиту:

- сировина;
- продукти харчування, харчоблоки;
- технологічні процеси;
- продукція;
- викиди в атмосферу;
- стічні води;
- відходи;
- засобу індивідуального і колективного захисту;
- техніка безпеки;
- положення про політика Компанії в області охорони праці і навколишнього середовища;
- екологічний паспорт підприємства й ін.

Зміст екологічного аудиту. Види і параметри аудиту.

Екологічний аудит повинний бути орієнтований на внутрішні індивідуальні потреби підприємства відповідно до його політики і встановлених цілями. Крім

того, важливо чітко ідентифікувати мети і задачі діяльності підприємства, перш ніж визначити, який тип екологічного аудиту йому необхідний.

При придбанні земельної нерухомості може виникнути необхідність у проведенні спеціальних досліджень з метою виявлення користувача останніх 50 років, якщо виникають підозри на забруднення навколишнього середовища.

При визначенні параметрів екологічного аудиту варто брати до уваги:

- ступінь детальності аналізу діяльності підприємства, що використовується в аудиті, тобто торкається окремих сторін або всієї діяльності компанії;
- вироблення концепції і мір, спрямованих на досягнення узгодження з нормативами і лімітами, установленими природоохоронними органами;
- географічне положення об'єкта (число офісів і місце розташування заводів);
- тимчасова структура (частота проведення аудитів);
- предмет екологічного аудиту (повітря, вода, землі, енергоспоживання, відходи).

Процедура екологічного аудиту повинна забезпечувати можливість оцінки відповідності об'єкта, що перевіряється, установленим, для нього критеріям екологічного аудиту - процедура проведення екологічного аудиту простій і доступної в керуванні і виконанні.

Перед проведенням аудиту необхідно:

- ідентифікувати процеси, використовувані в комерційній діяльності або виробничому процесі об'єкта аудитування;
- чітко установити цільову спрямованість політики і планів підприємства;
- визначити компетенцію аудиту, включаючи його структуру і масштаб;
- розробити процедури, що встановлюють порядок виконання аудиту.

Основними етапами процедури екологічного аудиту є:

- перевірка первинної документації, журналів реєстрації й інших матеріалів, що реєструють показники природоохоронної діяльності;

- збір інформації в рамках проведення аудиту, включаючи співбесіду з персоналом;
- візуальне обстеження об'єкта, перевірка стану й експлуатації технічних засобів;
- інструментальний аналіз параметрів навколишнього середовища і факторів негативного впливу;
- вироблення рекомендацій з удосконалювання природоохоронної діяльності і раціональному використанню природних ресурсів.

При проведенні аудиту підприємство одержує ряд переваг:

- визначення можливих шляхів досягнення екологізації діяльності підприємства і виявлення причин, що перешкоджають досягненню цієї мети;
- зниження імовірності піддатися ризикові судового позову і виплати великої компенсації за заподіяний збиток, а також екологічних платежів і штрафів;
- посилення екологізаціі діяльності і політики підприємства, екологічної свідомості і підвищення екологічної відповідальності персоналу підприємства;
- визначення ступеня відповідності діяльності підприємства екологічним нормативам якості навколишнього середовища, встановленим органами федеральної державної влади і суб'єктами федерації;
- розробка заходів для поліпшення екологічної діяльності підприємства й одночасно по зниженню вартості виробленої продукції;
- розробка інформаційної бази негайного реагування на випадок виникнення небезпеки;
- поліпшення методів керування при рішенні екологічних проблем;
- економічне стимулювання проектів, що забезпечують зниження негативного впливу на навколишнє природне середовище;
- поліпшення взаємин зі структурами влади і громадськістю;

- оцінки-ризику і збитку, зв'язаного з забрудненням навколишнього середовища в результаті виробничої або іншої діяльності підприємства;
- відповідність стандарту, що встановлює рівень екологічного ризику.

5.2. Висновки до п'ятого розділу

Екологічний аудит, визначений як систематичне оцінювання впливу діяльності підприємства чи організації на довкілля, є важливим інструментом для досягнення сталого розвитку та забезпечення екологічної стійкості. У результаті проведення екологічного аудиту отримується комплексна інформація про екологічну ефективність та вплив діяльності на природне середовище.

Проведення екологічного аудиту дозволяє ідентифікувати потенційні ризики та негативні наслідки для довкілля, а також розробляти стратегії для їх управління та зменшення. Застосування принципів зеленого бізнесу та впровадження екологічних інновацій можливе завдяки ретельному аналізу та вдосконаленню виробничих процесів на основі здобутків екологічної науки та технологій.

Важливим аспектом екологічного аудиту є стимулювання підприємств та організацій до впровадження екологічно відповідальних практик, що сприяє підвищенню їх конкурентоспроможності та позитивному впливу на здоров'я людей та довкілля загалом.

Необхідно враховувати, що екологічний аудит – це постійний процес, і вдосконалення екологічних стандартів є ключовим елементом для забезпечення ефективного управління природними ресурсами. Відповідальний та осмислений підхід до вирішення екологічних питань є запорукою сталого розвитку, який сприяє гармонійному співіснуванню людства та природи.

ВИСНОВОК

Темою дипломної роботи було «Програмний засіб стегоаналізу». Головною метою є впровадження сучасних методів захисту інформації, які базуються на основі комп’ютерного стеганоаналізу, програмування та машинного навчання. Проведено аналіз існуючих алгоритмів та методів стегоаналізу, розробка та дослідження програмного коду та засобів машинного навчання.

Робота складається з чотирьох розділів. В роботі було проаналізовано та розроблено: поняття стеганографії та стегоаналізу, методи та алгоритми машинного навчання. В межах підрозділів про стеганографію та стеганоаналіз були окреслені основні задачі та методики алгоритмів.

Розроблено програмний модуль для перевірки зображень на наявність прихованого змісту.

Практична цінність полягає в створенні модулю для розробленого алгоритму, перевірки зображень на предмет наявності прихованого змісту для захисту користувачів та організацій від зловмисного програмного коду та для протистояння прихованому листування методом стеганографії.

Даний метод і варіант модулю може використовуватись в організаціях, а також охоронних державних та міжнародних установах. Безпосередньо доцільно використовувати даний метод і в соціальних мережах.

Завдяки проведеним дослідженням та тестуванням можна зробити висновок про те, що розроблений код системи дозволяє в комплексному рішенні забезпечити максимально можливу захищеність комп'ютерних систем та їх користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Конахович Г. Ф., Прогонов Д. О., Пузиренко О. Ю. Комп'ютерна стеганографічна обробка й аналіз мультимедійних даних [підручник]. — К. : «Центр навчальної літератури», 2018. — 558 с.

2. Конахович Г. Ф., Пузиренко О. Ю. Комп'ютерна стеганографія. Теорія і практика. — К.: «МК-Пресс», 2006. — 288 с., іл.

3. Грибунин В. Г., Оков И.Н., Туринцев И.В. "Цифровая стеганография" - М.: Солон-Пресс, 2017. — 262 с.

4. Кошкіна Н.В. Стегоаналіз цифрових зображень із застосуванням контрольного вкраплення [Текст] / Кошкіна Н.В. // Матеріали 3 Міжнародної науково-технічної конференції «Захист інформації і безпека інформаційних систем», Львів. – 2014 р. – 98-100 с.

5. Cox I.J., Miller M., Bloom J., Fridrich J., Kalker T. Digital watermarking and steganography. [Текст] / Morgan Kaufmann, 2007 p. – 593 c

6. Zadiraka V. Spectral methods of computer steganography problem decision / V. Zadiraka, N. Koshkina // Methods of effective protection of information flows /ed. By V. Zadiraka, Y. Nykolaichuk. – Ternopil: Terno-graf, 2014. – P. 96–120.

7. Поліновський В.В. Інформаційна технологія для досліджень методів стеганографії і стеганоаналізу / В.В. Поліновський, В.Ю. Корольов, В.А. Герасименко, М.Л. Горинштейн // Комп’ютерно-інтегровані технології: освіта, наука, виробництво. – 2011. – №5. – С. 236–242.

8. I.Borysevych, A.Petrenko. Message stegoanalysis software // Information Technology & Implementation : матеріали конференції, 20-21 листопада 2023 р. – Київ : 2023. – С. 76-77

9. Westfeld A. Attacks on Steganographic Systems. Breaking the Steganographic Utilities EzStego, Jsteg, Steganos, and S-Tools – and Some Leassons Learned / A. Westfeld, A. Pfitzmann // Proceeding of the Workshop on Information Hiding. – 1999.

10. Husrev T. Sencar. Data Hiding Fundamentals And Applications / Husrev T. Sencar, Mahalingam Ramkumar, Ali N. Akansu // Digital Multimedia. ELSEVIER science and technology books. – 2004. – 364 p

11. Стаття «Захист інформації в компютерних системах», URL: http://zahyst-informatsiyi-v-kompyuternyh-systemah/

12. Стаття «Приховування даних в нерухомих зображеннях»,URL: https://revolution.allbest.ru/programming/00553679_0.html

13. Gary C. Kessler , Null Ciphers. An Overview of Steganography for the Computer Forensics Examiner, Forenic Science Communications, vol. 4, No5, 2004, p.27

14. Crandall R. Some Notes on Steganography. 1999. URL: https://www.semanticscholar.org/paper/Some-Notes-onSteganographyCrandall/9d84704f34e594fef44edd2515203a1db44f0af3

15. G. M. and D. R. Fridrich J, “Detecting LSB steganography in color, and gray-scale images.,” IEEE Multimed., pp. 8(4): 22–28

16. Бабич І.В. Огляд стеганографічних методів перетворення інформації в зображеннях / І.В. Бабич, С.А. Паламарчук, Н.А. Паламарчук, В.В. Овсянніков // Захист інформації. – 2012. - №1. – С. 1-7.

17. Корольов В. Ю. Дослідження стійкості НЗБ-стеганографії до RS-аналізу / В. Ю. Корольов, В. В. Поліновський, В. А. Герасименко: матеріали IV Міжнар. конф [“Сучасні проблеми радіоелектроніки, телекомунікацій та приладобудування ”]. Ч. 1. – Вінниця: ВНТУ Мін. освіти і науки України, 2009. – С. 53.

18. Весельська О. В., Зюбіна Р. В., Фролов О. В. Систематизація та класифікація наявних стеганографічних методів приховування інформації / Весельська О. В., Зюбіна Р. В., Фролов О. В. // Наукоємні технології № 2 (30), 2016. – С.188-194.

19. Кузнецов О. О. Стеганографія : навчальний посібник / О. О. Кузнецов, С. П. Євсеєв, О. Г. Король. – Х. : Вид. ХНЕУ, 2011. – 232 с.

20. Навроцький Д.О. Методи комп'ютерної стеганографії / Навроцький Д.О. // Вісник Національного технічного університету України "КПІ" Серія – Радіотехніка. Радіоапаратобудування.-2007.-№35. – С.105-108.

21. Юдін О.К. Аналіз стеганографічних методів приховування інформаційних потоків у контейнери різних форматів / О. К. Юдін, Р. В. Зюбіна, О. В. Фролов // Радиоэлектроника и информатика. - 2015. - № 3. - С. 13-21.

22. Andreas Westfeld and Andreas Pfitzmann, ”Attacks on Steganographic Systems Breaking the Steganographic utilities EzStego, Jsteg,Steganos, and S-Tools—and Some Lessons Learned

23. Westfeld A., Pfitzmann A. Attacks on steganographic systems. URL: https://link.springer.com/chapter/10.1007/10719724_5

24. Kamaldeep Joshi, Swati Gill, Rajkumar Yadav, "A New Method of Image Steganography Using 7th Bit of a Pixel as Indicator by Introducing the Successive Temporary Pixel in the GrayScale Image", Journal of Computer Networks and Communications, vol. 2018, Article ID 9475142, 10 pages, 2018

25. Быков С. Ф., Мотуз О. В. Основы стегоанализа // Защита информации. Конфидент. — СПб.: 2000. — № 3. — С. 38-41

26. Weng, S., Chen, M., Yu, L. & Sun, S. Lightweight and effective deep image steganalysis network. IEEE Signal Process. Lett.https://doi.org/10.1109/LSP.2022.3201727 (2022)

27. Geetha S., Silva S.Sivatha Sindhu, Kamaraj N. Close color pair signature ensemble adaptive threshold based steganalysis for LSB embedding in digital images. Transactions on Data Privacy, 1(2009) 140-161.

28. Johnson Neil F., Jajodia Sushil. Steganography: Seeing the Unseen. IEEE Computer, February 1998, pp.26-34.

29. Tang W., Li B., Tan S., Barni M., Huang J. CNN Based Adversarial Embedding with Minimum Alteration for Image Steganography. 2018. URL: https://arxiv.org/abs/1803.09043

30. Schaathun, . "Steganography and Steganalysis", Machine Learning in Image Steganalysis, Schaathun/Machine Learning in Image Steganalysis, 2012

31. Fabian Pedregosa, Ga¨el Varoquaux, Alexandre Gramfort , Vincent Miche, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, David Cournapeau,” Scikitlearn: Machine Learning in Python”,

Journal of Machine Learning Research 12 (2011) 2825-2830 Submitted 3/11; Revised 8/11; Published 10/11

32. Y. Qian, J. Dong, W. Wang, and T. Tan, “Deep learning for steganalysis via convolutional neural networks,” Media Watermarking, Secur. Forensics 2015, vol. 9409, p. 94090J, 2015

33. L. Pibre, J. Pasquet, D. Ienco, and M. Chaumont, “Deep learning is a good steganalysis tool when embedding key is reused for different images, even if there is a cover sourcemismatch,” IS T Int. Symp. Electron. Imaging Sci. Technol., pp. 14–18, 2016

34. Y. Qian, J. Dong, W. Wang, and T. Tan, “Learning and transferring representations for image steganalysis using convolutional neural network” Department of Automation , University of Science and Technology of China Center for Research on Intelligent Perception and Computing , Institute of Automat,” 2016

35. G. Xu, H. Z. Wu, and Y. Q. Shi, “Structural design of convolutional neural networks for steganalysis,” IEEE Signal Process. Lett., vol. 23, no. 5, pp. 708–712, 2016

36. J. Zeng, S. Tan, B. Li, and J. Huang, “Large-Scale JPEG Image Steganalysis Using Hybrid Deep-Learning Framework,” IEEE Trans. Inf. Forensics Secur., vol. 13, no. 5, pp. 1200–1214, 2018

37. 10 интересных фактов про Python. IT новости. URL: https://itproger.com/news/10- interesnih-faktov-pro-python

38. https://www.tensorflow.org/learn

39. https://keras.io/guides/

40. https://scikit-learn.org/stable/tutorial/basic/tutorial.html

41. https://docs.python.org/uk/3/library/tkinter.html

42. https://www.ibm.com/topics/random-forest

Додаток А

Вихідний код програмного комплексу

Алгоритм приховання інформації за допомогою НЗБ

```
import cv2
import numpy as np
from pathlib import Path



class Steganography:
    SECRET_KEY = "$$"
    BITS_PER_BYTE = 8

    def __init__(self, file: str):
        self.file = file

    @staticmethod
    def data2binary(data):
        if isinstance(data, str):
            return "".join([format(ord(i), '08b') for i in data])
        elif isinstance(data, (bytes, np.ndarray)):
            return [format(i, '08b') for i in data]
        else:
            raise TypeError

    def convert_to_rgb(images):
        rgb_images = []
        for img in images:
            # Якщо img - чорно-біле зображення
            if len(img.shape) == 2:
                img = cv2.cvtColor(img, cv2.COLOR_GRAY2RGB)
            rgb_images.append(img)
        return np.array(rgb_images)

    def hide_data(self, img, data):
        data += self.SECRET_KEY
        b_data = self.data2binary(data)
        len_data = len(b_data)
        d_index = 0

        for value in img:
            for pix in value:
                r, g, b = self.data2binary(pix)
                if d_index < len_data:
                    pix[0] = int(r[:-1] + b_data[d_index], 2)
                    d_index += 1
                if d_index < len_data:
                    pix[1] = int(g[:-1] + b_data[d_index], 2)
                    d_index += 1
```

Продовження Додатку А

```
if d_index < len_data:


 pix[2] = int(b[:-1] + b_data[d_index], 2)


d_index += 1


if d_index >= len_data:
                    break
        return img

    def encode(self, data: str, encrypted_filepath: str):
        if not data:
            raise ValueError("Empty data")
        if not encrypted_filepath:
            raise ValueError("Empty path")

        image = cv2.imread(self.file)
        enc_data = self.hide_data(image, data)
        cv2.imwrite(encrypted_filepath, enc_data)
        return Steganography(encrypted_filepath)


    def find_data(self, img):
        bin_data = ""
        for value in img:
            for pix in value:
                r, g, b = self.data2binary(pix)
                bin_data += r[-1] + g[-1] + b[-1]

        all_bytes = [bin_data[i: i + self.BITS_PER_BYTE] for i in range(0,
len(bin_data), self.BITS_PER_BYTE)]

        readable_data = ""
        for i in all_bytes:
            readable_data += chr(int(i, 2))
            if readable_data[-len(self.SECRET_KEY):] == self.SECRET_KEY:
                break
        return readable_data[:-len(self.SECRET_KEY)]

    def decode(self):
        image = cv2.imread(self.file)
        return self.find_data(image)

for path in Path("bmp_cover").iterdir():
    img = Steganography(str(path))
    img.encode("Borysevych", 'bmp_lsb_stego_images/' + 'Encoded_' + path.stem +
'.bmp')
    data = img.decode()
```

Продовження Додатку А

Алгоритм приховання інформації методом заміни палітри

```
from PIL import Image
import os

def text_to_binary(text):
    binary_message = ''.join(format(ord(char), '08b') for char in text)
    return binary_message

def hide_text_palette(image_path, secret_text, output_folder="encoded_images"):
    # Створюємо папку для збереження закодованих зображень
    if not os.path.exists(output_folder):
        os.makedirs(output_folder)

    # Відкриваємо та завантажуємо зображення
    img = Image.open(image_path)
    pixels = list(img.getdata())

    binary_secret_text = '00000000' + text_to_binary(secret_text) + '11111111'

    # Перевірка, чи текст можна вбудувати в зображення
    if len(binary_secret_text) > len(pixels) * 3:
        raise ValueError("Text is too long to be encoded in the image")

    # Вбудовуємо текст у палітру
    new_pixels = []
    text_index = 0
    for pixel in pixels:
        if text_index < len(binary_secret_text):
            new_pixel = list(pixel)

            for i in range(3):  # Обробляємо всі три канали (R, G, B)
                if binary_secret_text[text_index] == '1':
                    new_pixel[i] = new_pixel[i] | 1  # Закодовуємо '1'
                else:
                    new_pixel[i] = new_pixel[i] & ~1  # Закодовуємо '0'

                text_index += 1

            new_pixels.append(tuple(new_pixel))
        else:
            new_pixels.append(pixel)

    # Створюємо нове зображення зі зміненою палітрою
    encoded_img = Image.new("RGB", img.size)
    encoded_img.putdata(new_pixels)

    # Зберігаємо нове закодоване зображення
    output_path = os.path.join(output_folder,
f"encoded_{os.path.basename(image_path)}")
```

Продовження Додатку А

```
    encoded_img.save(output_path)



print(f"Text encoded successfully for {os.path.basename(image_path)}")



def process_images_in_folder(folder_path, secret_text,
output_folder="encoded_images"):
    # Створюємо папку для збереження закодованих зображень
    if not os.path.exists(output_folder):
        os.makedirs(output_folder)

    # Проходимо всі файли у папці
    for filename in os.listdir(folder_path):
        if filename.endswith((".png", ".jpg", ".jpeg", ".bmp")):
            image_path = os.path.join(folder_path, filename)
            hide_text_palette(image_path, secret_text, output_folder)

# Загальна папка із зображеннями
images_folder = "bmp_cover"

# Текст для шифрування
secret_text = "Borysevych"

# Папка для закодованих зображень
output_folder = "bmp_palette_stego_images"

# Викликаємо функцію для обробки всіх зображень у папці
process_images_in_folder(images_folder, secret_text, output_folder)
```

Продовження Додатку А

Код навчання та тренування моделі методом НЗБ

```
import numpy as np
import tensorflow as tf
from tensorflow.keras import layers, models, losses, optimizers
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.model_selection import train_test_split
from tensorflow.keras.callbacks import LearningRateScheduler, EarlyStopping
from tensorflow.keras import regularizers

import os
import pandas as pd



normal_images_path = 'bmp_cover'
stego_images_path = 'bmp_palette_stego_images'

normal_images = [os.path.join(normal_images_path, filename) for filename in
os.listdir(normal_images_path)]
stego_images = [os.path.join(stego_images_path, filename) for filename in
os.listdir(stego_images_path)]



normal_labels = ['0'] * len(normal_images)
stego_labels = ['1'] * len(stego_images)

all_images = normal_images + stego_images
all_labels = normal_labels + stego_labels

x_train, x_test, y_train, y_test = train_test_split(all_images, all_labels,
test_size=0.2, random_state=42)


datagen = ImageDataGenerator(rescale=1. / 255, validation_split=0.2)

datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2,
    rotation_range=20,
    width_shift_range=0.1,
    height_shift_range=0.1,
```

Продовження Додатку А

```
 zoom_range=0.1,


 horizontal_flip=True,
    fill_mode='nearest'
)

train_generator = datagen.flow_from_dataframe(
    dataframe=pd.DataFrame({'filename': x_train, 'class_str': y_train}),
    directory=None,
    x_col='filename',
    y_col='class_str',
    target_size=(256, 256),
    batch_size=32,
    class_mode='binary',
    subset='training'
)

validation_generator = datagen.flow_from_dataframe(
    dataframe=pd.DataFrame({'filename': x_test, 'class_str': y_test}),
    directory=None,
    x_col='filename',
    y_col='class_str',
    target_size=(256, 256),
    batch_size=32,
    class_mode='binary',
    subset='validation'
)

validation_generator = datagen.flow_from_dataframe(
    dataframe=pd.DataFrame({'filename': x_test, 'class_str': y_test}),
    directory=None,
    x_col='filename',
    y_col='class_str',
    target_size=(256, 256),
    batch_size=32,
    class_mode='binary',
    subset='validation'
)



def build_LSB_detection_model(input_shape):
    model = models.Sequential()
    model.add(layers.Conv2D(32, (2, 2), activation='relu',
input_shape=input_shape))
```

Продовження Додатку А

```
model.add(layers.MaxPooling2D((2, 2)))


model.add(layers.Conv2D(64, (2, 2), activation='relu'))
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.Conv2D(128, (2, 2), activation='relu'))
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.Flatten())
    model.add(layers.Dense(128, activation='relu',
kernel_regularizer=regularizers.l2(0.001)))
    model.add(layers.Dropout(0.5))  # Доданий Dropout шар для регуляризації
    model.add(layers.Dense(1, activation='sigmoid'))  # Вихідний шар

    model.compile(optimizer=optimizers.Adam(learning_rate=0.0001),
loss=losses.BinaryCrossentropy(),
                  metrics=['accuracy'])

    return model


input_shape = (256, 256, 3)

model = build_LSB_detection_model(input_shape)

epochs = 5

def schedule(epoch, lr):
    if epoch < 3:
        return 0.0001
    else:
        return lr * tf.math.exp(-0.1)

lr_scheduler = LearningRateScheduler(schedule)

early_stopping = EarlyStopping(monitor='val_loss', patience=3,
restore_best_weights=True)

model.compile(optimizer=optimizers.Adam(),
              loss=losses.BinaryCrossentropy(),
              metrics=['accuracy'])

model.fit(train_generator, epochs=epochs, validation_data=validation_generator,


callbacks=[lr_scheduler, early_stopping])

model.save('lsb_model1.keras')

loss, accuracy = model.evaluate(validation_generator)
print(f"Test Loss: {loss}, Test Accuracy: {accuracy}")
```

Продовження Додатку А

Код навчання та тренування моделі методом заміни палітри

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from skimage import color, io, feature
from skimage.util import view_as_blocks
from skimage.measure import shannon_entropy
import os
from glob import glob

def extract_palette_substitution_features(image):
        if image.shape[-1] == 1:
        image = color.gray2rgb(image)

    block_size = (8, 8, image.shape[-1])
    blocks = view_as_blocks(image, block_size)

    features = []

    for i in range(blocks.shape[0]):
        for j in range(blocks.shape[1]):
            block = blocks[i, j]

            if block.shape[0] >= 2 and block.shape[1] >= 2:
                mean_color = np.mean(block, axis=(0, 1))
                std_color = np.std(block, axis=(0, 1))
                features.extend(mean_color)
                features.extend(std_color)

                gray_block = color.rgb2gray(block)
                energy = np.sum(feature.graycomatrix(gray_block.astype('uint8'),
[1], [0], symmetric=True, normed=True))
                features.append(energy)

                entropy = shannon_entropy(gray_block.flatten())
                features.append(entropy)

    return features
```

Продовження Додатку А

```
original_path = "cover"
stego_path = "palette_stego_images"
original_image_files = glob(os.path.join(original_path, '*.jpg'))
stego_image_files = glob(os.path.join(stego_path, '*.jpg'))

if not original_image_files:
    print("Помилка: Немає оригінальних зображень у папці 'cover'")
if not stego_image_files:
    print("Помилка: Немає зашифрованих зображень у папці
'palette_stego_images'")


X_original = []
for filename in original_image_files:
    features = extract_palette_substitution_features(io.imread(filename))
    if features:
        X_original.append(features)
y_original = np.zeros(len(X_original))


X_stego = []
for filename in stego_image_files:
    features = extract_palette_substitution_features(io.imread(filename))
    if features:
        X_stego.append(features)
y_stego = np.ones(len(X_stego))

X_original = np.array(X_original)
X_stego = np.array(X_stego)

X = np.vstack((X_original, X_stego))
y = np.hstack((y_original, y_stego))

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)


accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy}')
```

Продовження Додатку А

Код основної програми з графічним інтерфейсом

```
import tkinter as tk
from tkinter import filedialog
from PIL import Image, ImageTk
import numpy as np
import tensorflow as tf
import tensorflow_hub as hub
from sklearn.externals import joblib

scikit-learn

class SteganalysisApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Steganalysis App")

        self.image_path = None

        self.label = tk.Label(root, text="Select an image for steganalysis:")
        self.label.pack(pady=10)

        self.load_button = tk.Button(root, text="Load Image",
command=self.load_image)
        self.load_button.pack(pady=10)

        self.result_label = tk.Label(root, text="")
        self.result_label.pack(pady=10)

        self.analyze_button = tk.Button(root, text="Analyze",
command=self.analyze_image)
        self.analyze_button.pack(pady=10)

        # Load the pre-trained TensorFlow model from TensorFlow Hub
        self.tf_model = self.load_tf_model("lsb_model1.keras")

        # Load the scikit-learn model
        self.scikit_model = self.load_scikit_model("scikit_model.joblib")

    def load_image(self):
        self.image_path = filedialog.askopenfilename()
        if self.image_path:
            image = Image.open(self.image_path)
            image = image.resize((224, 224))

            photo = ImageTk.PhotoImage(image)

            if hasattr(self, 'image_label'):
                self.image_label.configure(image=photo)
                self.image_label.image = photo
            else:
                self.image_label = tk.Label(root, image=photo)
                self.image_label.image = photo
                self.image_label.pack(pady=10)
```

```
    def analyze_image(self):
        if not self.image_path:
            self.result_label.config(text="Please load an image first.")
            return


                        prediction_tf =
self.tf_model.predict(self.load_and_preprocess_image(self.image_path))
            prediction_scikit =
self.scikit_model.predict(self.load_and_preprocess_image(self.image_path))


            if any(prediction_tf[0] > 0.5) or any(prediction_scikit[0] > 0.5):
self.result_label.config(text="Steganography detected.")
            else:
                self.result_label.config(text="No steganography detected.")

    def load_tf_model(self, model_path):
        model = tf.keras.models.load_model(model_path)
        return model

    def load_scikit_model(self, model_path):
        model = joblib.load(model_path)
        return model

    def load_and_preprocess_image(self, path):
        img = Image.open(path)
        img = img.resize((256, 256))
        img = np.array(img) / 255.0
        img = img.reshape((1, 256, 256, 3))
        return img

if __name__ == "__main__":
    root = tk.Tk()
    app = SteganalysisApp(root)
    root.mainloop()
```