Вінницький національний технічний університет
Факультет менеджменту та інформаційної безпеки
Кафедра менеджменту та безпеки інформаційних систем

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

Дослідження нейроквантронних систем перетворення для збільшення стеганоконтейнера під час передавання потокової конфіденційної інформації

Виконав: ст. 2-го курсу, групи 1КІТС - 22 спеціальності 125 – Кібербезпека
Освітня програма – Кібербезпека інформаційних технологій та систем
Кириченко Олександр Вікторович
(прізвище та ініціали)

Керівник: к.т.н., доц., доцент каф. МБІС
Сачанюк-Кавецька Н.В.,к.т.н., доцент
«04» грудня 2023 р.

Опонент: к.т.н., доц., доцент каф. ОТ
Шевченко І.Р.
(прізвище та ініціали)
«04» грудня 2023 р.

Допущено до захисту
Голова секції УБ кафедри МБІС
Юрій ЯРЕМЧУК
«04» грудня 2023 р.

Вінниця ВНТУ - 2023 рік

Вінницький національний технічний університет
Факультет менеджменту та інформаційної безпеки
Кафедра менеджменту та безпеки інформаційних систем

Рівень вищої освіти ІІ-й (магістерський) Галузь
знань 12 – Інформаційні технології
Спеціальність 125 – Кібербезпека
Освітньо-професійна програма – Кібербезпека інформаційних технологій та систем

ЗАТВЕРДЖУЮ
Голова секції УБ кафедри МБІС
_____Юрій ЯРЕМЧУК
«10» вересня 2023 р.

З А В Д А Н Н Я
на магістерську кваліфікаційну роботу студенту
Кириченко Олександр Вікторович
(прізвище, ім`я, по-батькові)

1. Тема роботи__Дослідження нейроквантронних систем перетворення для збільшення стеганоконтейнера під час передавання потокової конфіденційної інформації_____
_____
Керівник роботи_Сачанюк-Кавецька Н.В.,к.т.н., доцент_____
(прізвище, ім`я, по-батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від “18”вересня 2023року №247
2. Строк подання студентом роботи за тиждень до захисту_____
3. Вихідні дані до роботи: Підручники, електронні джерела, що стосуються теми магістерської кваліфікаційної роботи_____
4. Зміст текстової частини: Для досягнення мети даної роботи були поставлені наступні задачі: дослідити потреби людей у засобах захисту інформації для передавання; У першому розділі проаналізувати основні стеганографічні методи; У другому розділі розробити алгоритм внесення інформації в зображення для збільшення стеганоконтейнера на основі нейронної мережі; В третьому розділі реалізований алгоритм внесення інформації в зображення на основі нейронної мережі
_____

5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень)
В першому розділі наведено 6 рисунків, у другому розділі – 8 рисунків, у третьому розділі– 6 рисунків.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Основна частина	Сачанюк-Кавецька Н.В.,к.т.н., доцент		
I	Сачанюк-Кавецька Н.В.,к.т.н., доцент		
II	Сачанюк-Кавецька Н.В.,к.т.н., доцент		
III	Сачанюк-Кавецька Н.В.,к.т.н., доцент		
Економічна частина			
IV	Причепа І.В., к.е.н., доц. каф. ЕПВМ		

7. Дата видачі завдання 20 вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи		Примітка
1	Визначення напрямку дослідження в роботі	20.09.2023	31.09.2023	Виконано
2	Аналіз стеганографічних методів роботи	1.10.2023	15.10.2023	Виконано
3	Розробка роботи	16.10.2023	26.10.2023	Виконано
4	Написання магістерської роботи	27.10.2023	15.11.2023	Виконано
5	Передзахист магістерської роботи	16.11.2023	24.11.2023	Виконано
6	Виправлення та корегування магістерської роботи	27.11.2023	04.12.2023	Виконано
7	Захист МКР	11.12.2023	17.12.2023	

Студент _____ Кириченко О.
(підпис)

Керівник роботи _____ Сачанюк-Кавецька Н.В
(підпис)

АНОТАЦІЯ

УДК 004.056.55:004.032.26

Кириченко О.В.. Дослідження нейроквантронних систем перетворення для збільшення стеганоконтейнера під час передавання потокової конфіденційної інформації. Магістерська кваліфікаційна робота зі спеціальності 125 – «Кібербезпека», освітня програма «Кібербезпека інформаційних технологій та систем». Вінниця: ВНТУ, 2023.129с.

На укр.мові. Бібліогр.: 55 назв; рис.: 21; табл. 9.

Основними етапами виконання роботи є: аналітичний огляд існуючих стеганографічних методів, опис обраного алгоритму та методу, аналіз сучасних методів стеганоаналізу з використанням нейронних мереж та розробка програми, що виконує стеганоаналіз.

У першому та другому розділах магістерської роботи розглянуті стеганографічні методи, які використовують математичні перетворення, такі як дискретне косинусне перетворення (ДКП), вейвлет-перетворення, перетворення Уолша-Адамара та інші. Також розглянута їх реалізація на основі нейронних мереж .

В роботі пропонується рішення, яке спрямоване на збільшення обсягу стеганоконтейнера і, отже, підвищення швидкодії виконання перетворень, які є ключовою, але складною складовою в стеганографічних методах.

Результат виконання кваліфікаційної роботи – створення програмного продукту для вбудовування секретного повідомлення в зображення з використанням нейронної мережі.

У четвертому розділі роботи проведено аналіз економічної доцільності розробки, визначено високий комерційний потенціал та можливості подальшого впровадження розробленого інформаційного ресурсу.

Ключові слова: цифрова стеганографія, вбудовування додаткової інформації, статистичні особливості, дискретно- косинусне перетворення, jpeg, дкп коефіцієнти, методи ущільнення, нейронна мережа.

ABSTRACT

UDK 004.056.55:004.032.26

Kirichenko O.V.. Research of neuro-quantum transformation systems for increasing the stegano-container during the transmission of confidential information. Master's qualification work in speciality 125 - "Cybersecurity", educational programme "Cybersecurity of information technologies and systems". Vinnytsia: VNTU, 2023.

In Ukrainian. Bibliography: 55 titles; Figures: 22; Table 9.

The main stages of the work are: analytical review of existing steganographic methods, description of the chosen algorithm and method, analysis of modern methods of steganalysis using neural networks and development of a program that performs steganalysis.

The first and second chapters of the master's thesis describe steganographic methods that use mathematical transformations, such as the Discrete Cosine Transform (DCT), Wavelet Transform, Walsh-Adamar Transform, and others. Their implementation on the basis of neural networks is also considered.

The work proposes a solution aimed at increasing the size of the stegano container and, consequently, increasing the speed of performing transformations, which are a key but complex component in steganographic methods.

The result of the qualification work is the creation of a software product for embedding a secret message in an image using a neural network.

The fourth section of the work analyses the economic feasibility of the development, determines the high commercial potential and possibilities for further implementation of the developed information resource.

Keywords: digital steganography, embedding of additional information, statistical features, discrete cosine transform, jpeg, dcp coefficients, compression methods, neural network.

ВСТУП

Актуальність теми дослідження. Тема захисту інформації завжди була актуальною у різних сферах життя. Це завдання можна умовно розділити на дві основні області: стеганографію і кодування. Методи стеганографії спрямовані на приховування факту передачі або зберігання інформації, тоді як методи кодування та ущільнення зображень спрямовані на зменшення розміру файлу.

У цифровій стеганографії для захисту інформації, а точніше для таємного збереження самого факту захисту, використовуються контейнери – цифрові об'єкти, в які вбудовується інформація, при цьому викликаючи деяке спотворення.

Зазвичай в якості контейнерів використовуються мультимедіа-об'єкти, такі як зображення та відео. Спотворення, які вносяться до контейнеру для захисту, розташовані нижче порога чутливості людини, тому вони практично непомітні без використання спеціальних апаратних засобів і методів. У даній роботі в якості контейнерів використовуються формати зображень з втратою та без втрати якості.

Інформація, захищена таким способом, залишається безпечною лише до моменту її виявлення. В разі, коли інформація не захищена, тобто дані не зашифровані, стає досить легкою задачею стати власником цих даних. Однак це можливо лише у випадку, якщо зловмисник виявить наявність потенційно корисної інформації, яка прихована в контейнері. Цей процес виявляється досить неочевидним та вимагає значної зусилів.

Найчастіше методи стеганографії та методи ущільнення використовуються разом, утворюючи потужну систему захисту інформації. Ще одним важливим фактором у використанні перетворень зображень є розробка більш стійких до стегоаналізу алгоритмів для внесення інформації з метою приховування та використання різних форматів як контейнерів і стегоповідомлень. Це стосується не лише зображень і текстової інформації, але й файлів у будь-яких можливих форматах.

Мета та завдання дослідження. Виходячи із зазначеного вище,

поставлено завдання дослідження, а саме розробити програму, яка покращить швидкість виконання перетворень та приховання графічної інформації у зображенні для передачі потокової конфіденційної інформації. Для досягнення поставленої мети були визначені наступні завдання:

- Аналізувати методи внесення інформації в зображення різних форматів.
- Вдосконалити метод перетворення зображення для швидкої передачі потокової конфіденційної інформації.
- Покращити метод внесення інформації в зображення.
- Розробити конкурентоздатну програму на основі розробленого методу.
- Протестувати розроблений програмний додаток.

Об'єкт дослідження – система внесення інформації в зображення.

Предметом роботи є методи та засоби внесення інформації в зображення.

Методи дослідження. Для вирішення поставлених завдань використовувалися методи ущільнення, теорії автоматів, математичної статистики, теорії множин, нейронні мережі.

Наукова новизна роботи полягає в наступному:

1. Вперше удосконалити метод перетворення та приховування інформації в зображеннях для подальшої швидкої передачі по каналам зв'язку.

2. Запропонований метод перетворення зображення для подальшого приховування в ньому інформації.

3. Запропоновано алгоритм внесення інформації в зображення на основі нейронної мережі і який відрізняється підвищеною ефективністю швидкодії.

Практична значущість результатів. У ході дослідження були розглянуті засоби та методи внесення інформації в зображення для використання користувачами. Здобуті результати можуть служити основою для розробки програмного забезпечення в організаціях, спрямованого на внесення інформації в зображення. Робота складається зі вступу та 4 розділів.

1. ОГЛЯД МЕТОДІВ ПРИХОВУВАННЯ ІНФОРМАЦІЇ В СИСТЕМАХ

1.1 Використання стеганографічних методів в системах передачі потокової інформації

Завдання ефективного захисту інформації від несанкціонованого доступу є однією з найдавніших проблем, яка залишається актуальною до наших днів. Засоби та методи схування конфіденційних повідомлень відомі з давніх часів і визначаються терміном "стеганографія". Це слово походить від грецьких слів "steganos" (секрет, таємниця) і "graphy" (запис), і буквально означає "тайнопис", хоча методи стеганографії, ймовірно, виникли навіть до винайдення писемності, коли використовувалися умовні знаки і позначення.

Так, як і будь-які інструменти, стеганографічні методи потребують обережного та уважного використання, оскільки вони можуть служити як для захисту, так і для атак. Розвиток комп'ютерних технологій цього століття сприяв вдосконаленню та розширенню стеганографічних методів, що призвело до виникнення нового напрямку - комп'ютерної стеганографії.

У цьому новому напрямку інформація може бути прихована в будь-якому типі файлу, чи то текстовому, графічному, аудіо чи відео. Файл, в якому розташовані стеганоповідомлення, отримав назву "контейнер", а самі приховані повідомлення - "стеганоповідомлення". Канал передачі такої інформації отримав назву стеганографічного каналу або стегоканалу. У всій системі стеганографії може бути використано один або кілька ключів в залежності від рівнів захисту. Схоже на кодування, ключ може бути відкритим чи прихованим. Однак, на відміну від кодування, де зловмисник може точно визначити, що передане повідомлення є зашифрованим текстом, стеганографія дозволяє непомітно вбудовувати стеганоповідомлення в, на перший погляд, безпечні файли.

До стеганографії належать різноманітні методи, такі як відоме всім "невидиме чорнило" і менш відомі техніки, такі як умовне розташування знаків, мікрофотознімки, засоби зв'язку на плаваючих частотах і таємні канали. Розвиток комп'ютерних технологій відзначився новим імпульсом у

розвитку та вдосконаленні стеганографії, що призвело до виникнення нового напрямку в області захисту інформації - комп'ютерна стеганографія.

Сучасний прогрес в глобальних комп'ютерних мережах та засобах мультимедіа призвів до розробки нових методів, спрямованих на забезпечення безпеки передачі даних через телекомунікаційні канали та їх використання в неоголошених цілях. Зазначені методи, враховуючи природні неточності пристроїв оцифровки та особливості відео чи аудіо сигналу, дозволяють приховувати повідомлення в комп'ютерних файлах, відомих як "контейнери". Основними положеннями сучасної комп'ютерної стеганографії є наступні:

1. Методи приховування інформації повинні забезпечувати автентичність та цілісність файлів.

2. Передбачається, що противнику повністю відомі всі можливі стеганографічні методи.

3. Безпека методів базується на збереженні стеганографічного перетворенням основних властивостей відкрито переданого файлу чи файлів при внесенні до нього секретного повідомлення і деякої невідомої противнику інформації - ключа.

4. Якщо факт приховування повідомлення став відомий супротивнику через спільника, витяг секретного повідомлення являє складну обчислювальну задачу.

З урахуванням зростання ролі глобальних комп'ютерних мереж стає все важливішим використання стеганографії. Проведений аналіз джерел інформації в Інтернеті дозволяє зробити висновок, що стеганографічні системи активно використовуються для вирішення ряду основних завдань в сучасному світі:

1. Захист переданої інформації від несанкціонованого доступу.

2. Подолання систем моніторингу та управління мережевими ресурсами.

3. Камуфляж програмного забезпечення.

4. Захист авторських прав на певні види інтелектуальної власності..

В області захисту конфіденційної інформації від несанкціонованого доступу використання стеганографії є найефективнішим засобом розв'язання цієї проблеми. Наприклад, лише одна секунда оцифрованого звуку з частотою дискретизації 44100 Гц і рівнем 8 біт в стереорежимі може приховати приблизно 10 Кбайт інформації шляхом заміни найменш значущих молодших розрядів на приховане повідомлення. При цьому зміна значень відліків становить менше 1%. Такий зміщений звук практично не виявляється при прослуховуванні більшістю людей. [4].

Стеганографічні методи, які націлені на протидію системам моніторингу і управління ресурсами промислового шпигунства, дозволяють відперечити спробам контролю за інформаційним простором під час передачі даних через сервери управління локальних і глобальних обчислювальних мереж.

Ще однією сферою використання стеганографії є захист авторського права від піратства. На комп'ютерні графічні зображення може бути застосована спеціальна мітка, яка залишається невидимою для очей, але може бути виявлена за допомогою спеціалізованого програмного забезпечення. Такий підхід вже використовується в комп'ютерних версіях деяких видань. Цей напрямок стеганографії призначений не лише для обробки зображень, але також для аудіо- та відеофайлів і має на меті забезпечити захист інтелектуальної власності. Стеганографічна система — це є сукупність засобів і методів, які використовуються для формування прихованого каналу передачі інформації. При розробці необхідно враховувати такі аспекти:

1. Зловмисник повинен розуміти структуру та реалізацію стеганографічної системи повністю. Єдиним елементом, який залишається невідомим для потенційного противника, є ключ, який дозволяє тільки його власнику встановлювати факт присутності та зміст прихованого повідомлення.

2. У випадку, якщо зловмисник дізнається про існування прихованого повідомлення, це не повинно надавати йому можливості отримати доступ до відповідних повідомлень в інших даних, поки ключ залишається

конфіденційним.

3. Потенційному злоумиснику має бути ускладнено будь-яке визначення чи розкриття вмісту секретних повідомлень, позбавивши його будь-яких технічних або інших переваг у процесі розпізнавання.Узагальнена модель стеганосистеми зображена на рисунку. 1.1.

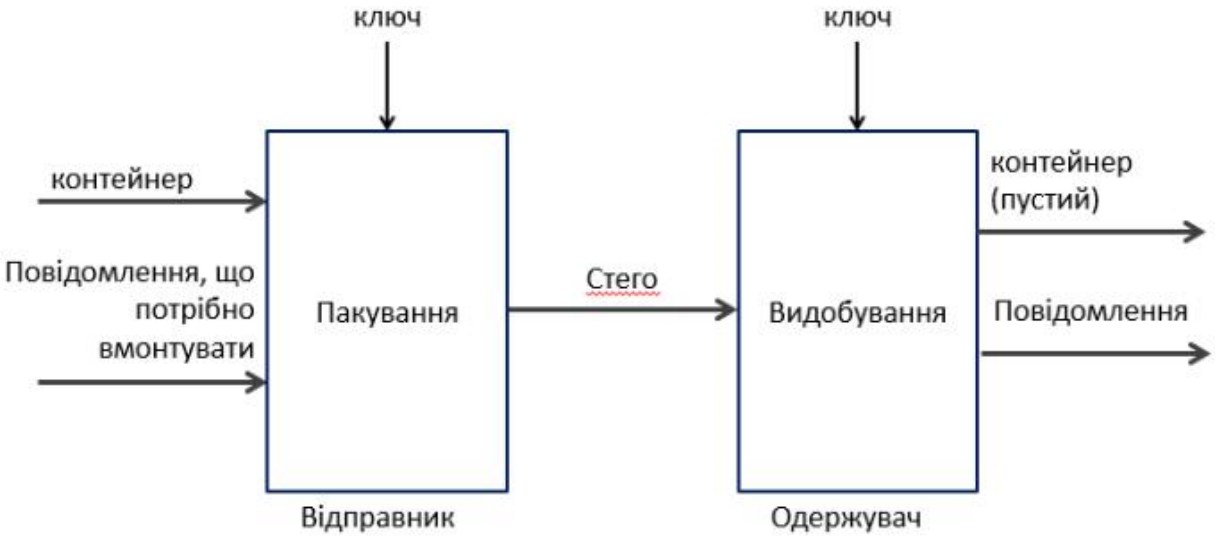


Рисунок 1.1 − Узагальнена модель стеганосистеми для передавання

В якості даних може використовуватися будь−яка інформація: текст, повідомлення, зображення і т. п. У загальному випадку доцільно використовувати слово "повідомлення", так як повідомленням може бути як текст або зображення, так і, наприклад, аудіо дані. Для позначення прихованої інформації, використовується саме термін повідомлення [6].

Аналіз тенденцій розвитку стеганографії сьогодні показує, що в найближчі роки інтерес до розвитку методів стеганографіх буде посилюватися все більше і більше. Передумови до цього вже сформувалися сьогодні. Зокрема актуальність проблеми інформаційної безпеки постійно зростає і стимулює пошук нових методів захисту інформації. З іншого боку, бурхливий розвиток інформаційних технологій забезпечує можливість реалізації цих нових методів. І звичайно сильним каталізатором цього процесу є лавиноподібний розвиток комп'ютерної мережі загального користування Internet, в тому числі такі невирішені суперечливі проблеми Internet, як захист авторського права, захист прав на особисту таємницю,

організація електронної торгівлі, протиправна діяльність хакерів, терористів і т.п .

Актуальною тенденцією в сучасності є впровадження методів ущільнення інформації. Однак цей напрямок супроводжується численними невирішеними проблемами, пов'язаними з руйнівною дією складових інформаційної зброї, таких як комп'ютерні віруси, логічні бомби, автономні реплікативні програми і інші. Поєднання методів комп'ютерної стеганографії та ущільнення може виявитися перспективним вихідним шляхом з цієї ситуації. У цьому випадку можна було б подолати слабкі сторони відомих методів захисту інформації і розробити більш ефективні та новаторські методи забезпечення інформаційної безпеки. [7].

Вибір контейнера суттєво впливає на надійність стеганосистеми та можливість виявлення прихованого повідомлення. Наприклад, здатне визначити зміну колірної гами при впровадженні повідомлення в репродукцію «12 Апостолів» Рафаеля або "Соняшники" Пабло Пікассо має значення досвідченого цензора з художньою освітою.

Контейнери можна розділити на два типи за їхньою протяжністю: потокові (безперервні) і обмеженої (фіксованої) довжини. У потокових контейнерах неможливо визначити точний початок або кінець [3]. Крім того, немає можливості передбачити, якими будуть наступні шумові біти, що вимагає включення приховуючих бітів в потік в реальному часі. Сами приховуючі біти вибираються за допомогою спеціального генератора, який задає відстань між послідовними бітами в потоці.

У неперервному потоці інформації основна складність для одержувача полягає в труднощах визначення моменту початку прихованого повідомлення. У випадку наявності у потоковому контейнері сигналів синхронізації чи меж пакета приховане повідомлення розпочинається безпосередньо після одного з цих елементів. З іншого боку, для відправника може виникнути проблема, якщо він не впевнений, що довжина потоку контейнера буде достатньою для розміщення усього таємного повідомлення.

При використанні контейнерів фіксованої довжини відправник заздалегідь визначає розмір файлу і може вибирати приховуючі біти у псевдовипадковій послідовності. Проте контейнери фіксованої довжини, як вже зазначалося вище,

обмежені обсягом, і іноді приховане повідомлення може не вміститися в файл-контейнері.

Додатковий недолік полягає в тому, що відстані між приховуючими бітами розподілені рівномірно від найкоротших до найдовших вказаних відстаней, тоді як справжній випадковий шум має експоненціальний розподіл довжини інтервалу. Хоча, звісно, можна генерувати псевдовипадкові експоненціально розподілені числа, цей підхід, як правило, дуже трудомісткий. Однак на практиці найчастіше використовують контейнери фіксованої довжини через їхню широку поширеність і доступність. Існують кілька можливих варіантів контейнерів [4]:

- Контейнер, що створюється самою стеганосистемою. Наприклад, програма MandelSteg використовує фрактали як контейнер для вбудовування повідомлення. Цей підхід можна охарактеризувати як конструюючу стеганографію.

- Контейнер обирається з числа наявних варіантів. У цьому випадку генерується значна кількість альтернативних контейнерів, після чого обирається найбільш підходящий для вбудовування повідомлення. Цей метод можна назвати селектуючою стеганографією. При виборі оптимального контейнера з великого розмаїття створених варіантів ключовою є вимога природності контейнера. Однак проблема полягає в тому, що навіть оптимально організований контейнер може заховати обмежену кількість даних при значному обсязі самого контейнера.

- Контейнер постачається ззовні. У цьому випадку відсутній вибір контейнера, і для приховування повідомлення використовується перший доступний контейнер, який не завжди є оптимальним для вбудовування повідомлення. Цей метод можна назвати безальтернативною стеганографією.

1.2 Аналіз методів приховування інформації в графічних зображеннях

Усі методи, спрямовані на конфіденційне приховування даних, можна класифікувати залежно від основних принципів на форматні та неформатні.

Форматні методи приховування, також відомі як форматні стеганографічні системи, ґрунтуються на особливостях формату зберігання графічних даних.

Розробка таких методів включає в себе аналіз формату з метою виявлення службових полів, зміна яких у конкретних умовах не вплине на візуальну репрезентацію графічного зображення. Наприклад, для приховування можуть використовуватися службові поля формату, які присутні у графічних файлах, але в даний момент не використовуються [7,8]. Проте всі форматні методи мають спільний недолік - можливість побудови повністю автоматизованого алгоритму для виявлення прихованих даних, що знижує їхню стійкість до атак пасивних супротивників.

Неформатні методи використовують безпосередньо самі дані, що представлені у конкретному форматі зображення.

Метод використання неправдивих таблиць квантування.

Метод представляє собою дальший розвиток попереднього. Його сутність полягає в створенні додаткових таблиць квантування, що дозволяє значно збільшити обсяг приховуваних даних порівняно з попереднім методом. У стандарті JPEG передбачена можливість використання декількох таблиць квантування, що не порушує внутрішню структуру формату. За винятком вищезазначених недоліків для даного методу, він частково стає форматним, оскільки використовує особливість формату, яка є допустимою, але не стандартною. Загалом на практиці використовують два варіанти використання неправдивих таблиць квантування [30]. У першому випадку таблиці додаються для підвищення ефективності стиснення та зменшення втрат під час стиснення, що передбачено в специфікації алгоритму JPEG. Однак для більшості зображень кількість додаткових таблиць є невеликою.

Другий варіант включає в себе введення помилкових таблиць квантування з певним (не завжди фіксованим) інтервалом, де зазвичай використовуються одні й ті ж таблиці, і відмінності між ними обмежені молодшими бітами, де приховано повідомлення [1]. Очевидно, що цей підхід є форматним і має низьку стійкість до пасивних атак, спрямованих на виявлення факту прихованого повідомлення.

Метод приховування в спектрі зображення після квантування.

Метод базується на використанні частот блоків зображення після їх квантування, але перед етапом кодування. При цьому використовуються стандартні

методи комп'ютерної стеганографії для приховування. Цей метод дозволяє вбудовувати значно більше бітів, ніж попередні, і не є форматним. Таким чином, його стійкість до пасивних атак може значно перевищувати рівень стійкості раніше згаданих методів, в залежності від конкретної реалізації. При використанні цього методу обсяг приховуваних даних пропорційний обсягу стисненого зображення. Однак збільшення обсягу вбудовуваної інформації може впливати на вихідне зображення та знижувати ефективність наступного етапу кодування. Необхідність варіювати якість стисненого зображення у широкому діапазоні ускладнює виявлення можливих змін у зображенні, що може бути наслідком приховування даних або використання великих коефіцієнтів квантування.

Суть даного методу полягає в наступних кроках. Нехай m – біти приховуваного повідомлення, $B_{i,\ j}$ - значення ненульових елементів блоків квантованного спектра немодифікованого зображення, впорядковані згідно з порядком їх кодування в алгоритмі JPEG, де i – номер біта елемента, j – номер елемента, $B_{i,j}'$ – відповідні блоки модифікованого зображення [8].

Методи приховування в графічних зображеннях з палітрою кольорів. Використання палітри, або відображення кольорів, в графічних форматах пов'язане з спробою зменшити обсяг зберіганої інформації. Зазвичай, палітра вперше з'явилася в графічних адаптерах як спрощення їхньої структури та забезпечення вищого роздільного розширення при меншому обсязі оперативної пам'яті графічного адаптера [9]. З цим пов'язане створення форматів для зберігання растрових графічних зображень, які базуються на використанні палітри, де деякі з них активно використовуються і до цього часу. Прикладом такого формату є GIF, який широко використовується в мережі Інтернет і став необхідною частиною дизайну сучасних веб-сторінок та Інтернет-реклами [34]. Кількість переданих файлів у форматі GIF по мережі перевищує кількість переданих сторінок і листів більш ніж вдвічі (на заміну цьому застарілому формату був розроблений формат PNG, який також підтримує використання палітри кольорів, хоча він ще не отримав такого широкого розповсюдження).

Кожна точка у звичайному реєстрованому графічному зображенні визначає інтенсивність кольорових компонентів у будь-якому фіксованому колірному

просторі (RGB, CMY, CMYK). У випадку, якщо формат для зберігання зображень використовує палітру, точки зображення можуть представляти лише один колір із наявних в палітрі [9]. Палітра кольорів - це набір елементів, кожен з яких визначає (аналогічно точці у звичайному зображенні) інтенсивність кольорових компонентів у конкретному колірному просторі (зазвичай RGB). При цьому кожна точка зображення містить лише номер кольору з палітри, а не інформацію про сам колір у колірному просторі.

Внизу наведений приклад 8-бітного RGB-зображення розміром 4x4 точки, в якому використовується палітра і містить точки зі змінюваними зеленим і синім кольорами, які чергуються в шахматному порядку. Зображення записано у вигляді матриці з (R, G, B) – елементами :

(0,255,0) (0,0,255) (0,255,0) (0,0,255)
(0,0,255) (0,255,0) (0,0,255) (0,255,0)
(0,255,0) (0,0,255) (0,255,0) (0,0,255)
(0,0,255) (0,255,0) (0,0,255) (0,255,0)

Для зберігання даної матриці необхідно 384 біт пам'яті .

Якщо використовувалось зображення з палітрою, то для даного зображення потрібна палітра, що складається з двох кольорів:

$$0 \rightarrow (0,255,0);\ 1\ \rightarrow (0,0,255)$$

Тоді в цій палітрі зображення набуде вигляду

0 1 0 1
1 0 1 0
0 1 0 1
1 0 1 0

Отже, для зберігання інформації про використану палітру потрібно 48 біт в пам'яті, а для зберігання самого зображення - 16 біт.

Метод приховування з використанням молодших біт даних зображення.

З наведеного вище прикладу видно, що використання методу приховування в молодших бітах для зображень з палітрою без додаткового оброблення не є доцільним, оскільки елементи палітри, що відрізняються лише молодшим бітом, можуть мати абсолютно різні кольори, що призводить до помітних змін у самому

зображенні.

Простий спосіб подолати ці труднощі полягає в тому, що перед приховуванням повідомлення в молодших бітах зображення проводиться аналіз палітри зображення. Серед усіх пар (2i, 2i + 1) елементів палітри визначаються ті, різниця між колірними інтенсивностями яких не перевищує визначеної порогової величини d. Приховування здійснюється в молодших бітах лише тих точок зображення, які посилаються на відібрані елементи палітри. Оскільки палітра залишається незмінною під час приховування, аналіз виконується аналогічно перед вилученням. Проте кількість пар елементів палітри, придатних для приховування за цим методом, зазвичай невелика.

Даний метод можна поліпшити, відсортувавши палітру зі збереженням оригінальних номерів елементів у порядку зростання їхньої ваги, наприклад, (65536R + 256G + B). Потім визначаються пари, придатні для приховування, шляхом аналізу відсортованої палітри. Проте приховування в цьому випадку проводиться трошки інакше. Тепер для приховування біта повідомлення потрібно змінити не лише молодший біт точки зображення, але й її значення взагалі на нове, яке отримується шляхом зміни молодшого біта номера відсортованої палітри в тому випадку, якщо він є придатним для приховування.. Інакше кажучи, з кожним елементом палітри тепер пов'язані два числа i, j_i , де i - номер i -го елементу палітри, отриманий в результаті її сортування. Приховування полягає ще в тому, що послідовно проглядаються крапки зображення, за значенням точки k визначається відповідний номер j_k. Якщо номер j_k придатний для приховування, то його молодший біт замінюється на черговий біт повідомлення. Потім з j_k визначається пов'язаний з ним вихідний номер k, який і присвоюється поточній точці [10].

Метод приховування з використанням молодших біт елементів палітри.

В усіх форматах, де використовується палітра кольорів, сама палітра повинна бути включена у файл зображення. З цієї причини можна використовувати метод приховування в менш значущих бітах елементів палітри, оскільки формат зберігання цих елементів аналогічний формату точок звичайного зображення без палітри. Але слід зауважити, що розмір палітри обмежений 256 елементами, і кожен з них може

містити не більше 3 біт. За таким підходом можна приховати повідомлення об'ємом не більше 768 біт, що значно менше обсягу самого зображення.

Додатково, при використанні цього методу в палітрі можуть з'явитися елементи, що кодують однакові кольори. Присутність таких "однакових" елементів в палітрі може служити критерієм для виявлення наявності прихованого повідомлення в менш значущих бітах палітри.

Метод приховування, заснований на наявності однакових елементів палітри.

Очевидно, що у випадку, коли палітра включає два чи більше ідентичних елементів, неважливо, якому з них призначено кольоровий ідентифікатор точки зображення, оскільки при перегляді вона буде виглядати однаково. Цю особливість можна використовувати для приховування інформації, не вносячи жодних змін у саме зображення. Слід відзначити, що з точки зору кодування графічних зображень використання ідентичних елементів у палітрі не тільки є непрактичним, але й небажаним, оскільки це може призвести до збільшення обсягу файлу зображення [11].

З точки зору стеганографії, метод, що базується на використанні однакових кольорів у палітрі, вважається форматним, оскільки він ґрунтується на використанні особливостей, що є спільними для всіх графічних форматів, що підтримують палітру кольорів. У загальному випадку цей метод полягає у виявленні кількох елементів палітри, які зустрічаються найчастіше в графічному зображенні. До палітри додають їх "двійники", і потім кожна точка зображення проглядається послідовно. Якщо точка посилається на елемент, який має "двійника", вона використовується для приховування наступного біта повідомлення (наприклад, якщо біт повідомлення дорівнює 1, то значення точки замінюється на "двійника").

Нижче розглянуто приклад використання даного методу. Нехай повідомлення m = 10010110, палітра складається з двох кольорів:

$$0 \rightarrow (0,255,0);\ 1 \rightarrow (0,0,255)$$

0 1 0 1

і зображення має вигляд

1 0 1 0
0 1 0 1
1 0 1 0

Після додавання в палітру елементу 2 → (0,255,0) в зображенні можна приховати повідомлення m (Виділено жирним шрифтом).Цей метод можна продовжити на випадок додавання ще декількох однакових кольорів, проте при цьому зменшиться його стійкість.

2 1 **0** 1
1 **0** 1 **2**
0 1 **2** 1
1 **2** 1 **0**

Метод приховування шляхом перестановки елементів палітри.

Суть цього методу полягає в використанні порядку елементів палітри зображення для приховування інформації. Припустимо, що палітра довільного фіксованого зображення містить n різних елементів, тобто серед них немає жодної пари однакових. За принципами комбінаторики відомо, що кількість перестановок n різних елементів дорівнює n1. Зрозуміло, що якщо використовувати ці перестановки для приховування повідомлення, то його максимальна довжина становитиме приблизно $\log_2(n1)$ біт. У загальному випадку метод приховування через перестановку елементів палітри полягає в утворенні відображення, яке при фіксованому ключі однозначно встановлює відповідність між будь-яким повідомленням допустимої довжини та конкретною перестановкою елементів палітри контейнера. Нижче представлено приклад найпростішого методу перестановки елементів палітри. Припустимо, що повідомлення m - це ціле число від 0 до n! - 1, де n - кількість різних елементів палітри. Усі елементи в палітрі впорядковані за зростанням ваги, що дорівнює (65536R + 256G + B). Місця в палітрі, яка отримана в результаті приховування, порожні та пронумеровані від 0 до n-1. Місце для першого елемента палітри визначається як залишок від ділення m на n

[38]. Місце для другого елементу одиничної палітри обчислюється шляхом ділення m на n без залишку і взяття залишку від ділення отриманого результату на n-1. Таким самим чином обчислюються позиції решти елементів палітри, і отримується нова палітра, яка відповідає вихідному повідомленню m. Після отримання нової палітри необхідно змінити відповідним чином значення всіх точок зображення. Таким чином, після приховування палітра набуває вигляду: cba.

Метод приховування з використанням таблиць квантування.

Цей метод є одним із найпоширеніших приховувань даних у файлах JPEG на сьогодні. Суть його полягає в тому, щоб використовувати для стеганографії молодші біти чисел, які представляють коефіцієнти квантування. Його перевагою є те, що він не порушує стандартну структуру потоку JPEG, а, отже, є абсолютно неформатним. Однак недоліками є обмежена місткість (так як зазвичай у файлах JPEG лише одна або дві таблиці квантування), що обмежує обсяг приховуваних даних (дозволяє приховати лише 8 байт, наприклад, у всіх молодших бітах однієї таблиці квантування). Також важливо відзначити, що зміна молодших бітів коефіцієнтів квантування може спричинити зміни в статистичних характеристиках стиснених блоків, що негативно впливає на ефективність подальшого кодування і призводить до збільшення розмірів файлу.

Застосування неформатних методів призводить до спотворень, внесених стеганографічною системою. Однак такі методи є більш стійкими до атак як пасивних, так і активних противників. У зв'язку з цим у роботі розглядаються методи з втратами, які краще відповідають вимогам передачі інформації. Після докладного аналізу алгоритму стиснення з втратами JPEG, його режимів роботи і проміжних етапів (таких як перетворення зображення в оптимальний колірний простір, субдискретизація, дискретне косинусне перетворення, перетворення Уолша-Адамара, квантування і кодування), давайте розглянемо ці методи більш докладно. Вони надають можливість виконувати неформатне приховування даних в файли, формати яких відповідають стандарту JPEG.

Стандарт JPEG дозволяє використовувати стиснення зображень без втрат (режим Lossless JPEG), що істотно відрізняється від режиму з втратами, заснованого

на квантуванні коефіцієнтів дискретного косинусного перетворення (ДКП). Lossless JPEG використовує кодування з прогнозуванням, застосовуючи схему двовимірної диференціальної імпульсно-кодової модуляції (ДІКМ). За цією схемою значення кожного пікселя об'єднуються зі значеннями його сусідів для формування прогнозуючого параметра, який потім віднімається від вихідного значення. Результати обробки таким чином усіх точок зображення піддаються стисненню за допомогою арифметичного кодування або кодування за методом Хаффмана.

Отже, у випадку використання Lossless JPEG можна зазначити можливість приховування інформації безпосередньо в самому зображенні. При цьому застосовується основний метод комп'ютерної стеганографії – метод приховування в молодших бітах та його модифікацій.

Однак використання Lossless JPEG відбувається рідко, і використання формату JPEG в режимі стиснення з втратами для приховування інформації цим методом є складним, хоча можливим. Це особливо важливо в сценаріях, де необхідно враховувати швидкість передачі потокової інформації.

З цієї причини далі ми розглянемо більш детально метод Уолша–Адамара, який визначається як найбільш ефективний з точки зору коефіцієнта стиснення зображень.[12]

1.3 Метод Уолша-Адамара для попереднього ущільнення зображення

Усі стеганографічні методи, які використовують математичні перетворення (наприклад, ДКП, вейвлет-перетворення, перетворення Уолша-Адамара і інші), виявляються обчислювально більш складними, ніж "прямі" стеганографічні методи. Ми пропонуємо рішення, яке спрямоване на збільшення розміру стеганоконтейнера, що в свою чергу підвищить швидкість виконання математичних перетворень – найбільш трудомісткої складової стеганографічних методів. Розглянемо дану методику.

Всі методи ущільнення інформації для передачі базуються на припущенні, що набір даних завжди містить надлишкові елементи. Ущільнення досягається шляхом пошуку та кодування цих надлишкових елементів [12].

Потік даних зображення містить значну кількість зайвої інформації, яку можна усунути практично без помітних для ока змін. При цьому можна виділити два типи надлишковості.

Статистична надмірність пов'язана з кореляцією та передбачуваністю даних. Це відмінності можна усунути, не втрачаючи інформації, і при цьому можна повністю відновити вихідні дані [13]. Найвідоміші методи ефективного кодування символів базуються на знанні частоти кожного символу в повідомленні. За допомогою цих частот будується таблиця кодів, яка має такі властивості:

- Різні коди можуть мати різну кількість біт.

- Коди для символів з більшою частотою зустрічей мають менше біт, ніж коди для символів з меншою частотою.

Хоча коди мають різну бітову довжину, вони можуть бути відновлені єдиним чином, тобто вони будуються як префіксні.Цими властивостями володіє відомий алгоритм Хаффмана [14].

Візуальна (суб'єктивна) надмірність може бути усунута шляхом часткової втрати даних, яка мінімально впливає на якість відтворених зображень. Це відноситься до інформації, яку можна видалити з зображення, при цьому не порушуючи візуально сприйману якість зображень. Усунення візуальної надмірності є ключовим фактором для скорочення переданої інформації [14]. З метою оптимізації процесу кодування з метою передачі мінімального обсягу інформації, важливо забезпечити два аспекти: з одного боку, уникнути передачі зайвої інформації, а з іншого боку - уникнути надмірної втрати якості зображення. Дотепер не існує простої і адекватної моделі візуального сприйняття зображень, придатної для оптимізації їхнього кодування [15].

Задача ущільнення зображення включає дві основні фази: кодування та декодування. Якщо відновлене зображення завжди точно відповідає вихідному кодованому зображенню, то такий алгоритм кодування-декодування вважається алгоритмом ущільнення без втрат. У випадку, коли декодоване зображення відрізняється від кодованого, ми використовуємо термін "алгоритм ущільнення із втратами". Загальна схема процесу ущільнення зображення представлена на рис. 1.2

[16]. Давайте розглянемо етапи цієї процедури узагальнено. Будь-який метод ущільнення включає три основні етапи (див. рисунок 1.2):

- Кодування або первинне ущільнення;
- Вторинне ущільнення;
- Декодування або відновлення зображення.

На першому етапі робиться перетворення вихідних даних з однієї форми подання в іншу. Так, при ущільненні зображень в залежності від виду алгоритму ущільнення може бути виконаний перехід від вихідного зображення до наступних видів подання.

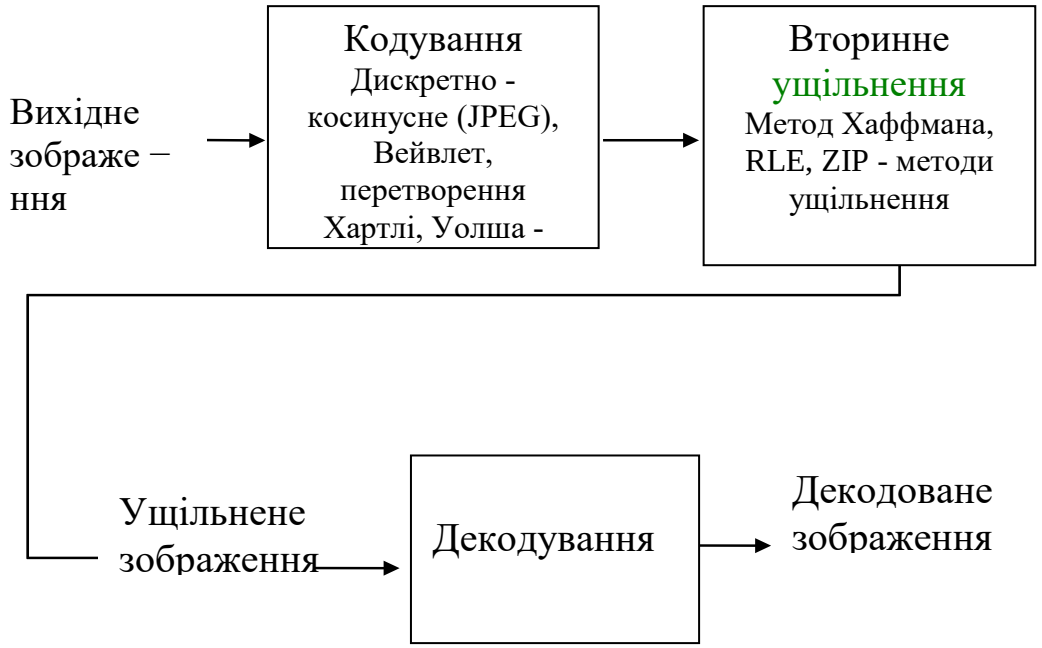


Рисунок - 1.2 Основні етапи ущільнення цифрових зображень

На другому етапі компоненти перетворення піддаються квантуванню і приводяться до форми, зручної для статистичного кодування, після чого їх кодують. Цей етап сприяє ущільненню інформаційного потоку. Для симетричних методів ущільнення процедури ущільнення та відновлення є однотипними, і час, необхідний для ущільнення та відновлення, порівнянний. Однак для несиметричних методів процедура ущільнення відрізняється від процедури відновлення і, зрозуміло, вимагає більше машинного часу.

Визначимо основні величини, що характеризують метод ущільнення.

1. Коефіцієнт ущільнення (K_y)

$$K = {}^{1}V_2$$

Цей параметр визначає в скільки разів файл, що зберігає ущільнене зображення, менший від файла, що зберігає вихідне зображення. Величини V_1 і V_2 виражаються в байтах. K_y − величина безрозмірна.

Більшість методів ущільнення із втратою базуються на кодуванні не самих даних, а на деяких лінійних перетвореннях від них, наприклад, апроксимації і визначенні коефіцієнтів дискретного перетворення Фур'є (ДПФ), коефіцієнтів перетворення Хартлі, перетворень Хаара, Уолша тощо [20]. .

Розглянемо декілька методів ущільнення зображень в базисах ортогональних функцій.

Особливості перетворення Уолша-Адамара. Одним із найбільш важливих та відомих несинусоїдальних ортогональних перетворень є перетворення Уолша-Адамара. Його відмінною особливістю є використання операцій додавання у процесі обчислень. Під час перетворення використовується послідовно упорядкована квадратна матриця Адамара H. Елементи цієї матриці можуть набувати лише значень +1 або -1, а відповідні рядки та стовпці взаємно ортогональні.Матриця перетворення H для двох випадкових величин матиме вигляд [27]:

$$\mathrm{H} = \begin{vmatrix} 1 & 1 \\ 1 & -1 \end{vmatrix}$$

Якщо з елементів матриці H скласти базисні вектори

$$\mathrm{e}_1 = \left[\mathrm{h}_{11}, \mathrm{h}_{12}\right] = \left[1,1\right] \mathrm{e}_2 = \left[\mathrm{h}_{21}, \mathrm{h}_{22}\right] = \left[1,-1\right],$$

то вони характеризуватимуть поворот ортогональної системи координат на 45° щодо одиничного базису (рис. 1.3). Якщо випадкові величини x_1 і x_2 мають кореляційну залежність, то проекція вектора X $[x_1, x_2]^T$ на базисний вектор e_1 буде, в середньому, більша, ніж проекція цього ж вектора на базисний вектор e_2. Це дозволяє сконцентрувати інформацію в першому коефіцієнті перетворення, тоді як другий коефіцієнт використовується для поліпшення представлення вектора X в новому базисі.

При збільшенні розмірності простору значна частина інформації буде

сфокусована в невеликій кількості коефіцієнтів, які необхідно зберегти з мінімальними втратами. Інші коефіцієнти можна відкинути або піддати квантуванню з більшим кроком.

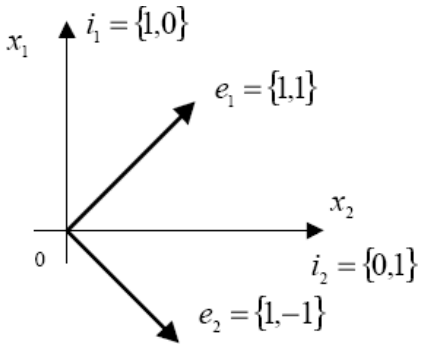


Рисунок. 1.3. - Розташування базисних векторів перетворення Адамара щодо одиничного базису

Матрицю Адамара розмірності 4x4 елементи легко побудувати з матриці Адамара H розмірністю 2x2 елементи:

$$H = \begin{vmatrix} H & H \\ H & -H \end{vmatrix} = \begin{vmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{vmatrix}$$

Користуючись співвідношенням (1.2), можна побудувати матрицю Адамара будь-якої розмірності $N = 2^n$, де n - будь-яке ціле позитивне число.

Перетворення Уолша-Адамара початкового фрагменту, яке містить N елементів зображення, здійснюється в матричному вигляді у відповідності з формулою:

$$[F(u,v)] = [H(u,v)][f(x,y)][H(u,v)]^T, \qquad (1.1)$$

де [f(x,y)] - відліки початкового сигналу f(x,y), [H(u,v)] - матриця Адамара порядку N, [F(u,v)] - трансформанта перетворення Уолша-Адамара сигналу f(x,y), (x,y) - координати, що визначають розташування відліку на площині початкового фрагменту, (u,v) - узагальнені частоти в області

трансформанти.

$$f(x, y) = \frac{1}{N * N}[H(u, v)]^T [F(u, v)][H(u, v)]. \qquad (1.2)$$

Формула (1.3) представляє собою розкладення початкового сигналу f(x,y) і визначає амплітуди множин прямокутних функцій Уолша.

Перевагою цього перетворення є його простота в реалізації та низька обчислювальна складність. Воно ефективно застосовується для кусково-постійних функцій, оскільки базові функції перетворення Адамара визначають постійні компоненти сигналу. У випадку кусково-постійних сигналів багато коефіцієнтів розкладання у перетворенні Адамара буде невеликим, що призводить до зменшення ентропії оброблених даних та підвищення коефіцієнта ущільнення. Однак на практиці такі сигнали є рідкісними, тому потрібно знайти перетворення, яке надає оптимальний або майже оптимальний розклад сигналів, більш точно характеризуючи реальні зображення. Для теоретичного аналізу функціонування оптоелектронних вузлів і пристроїв некогерентних процесорів можна використовувати елементарні квантрони, які виступають у ролі автоматів.

1.3 Квантрони, як функціонально повні операційні автомати із внутрішньою пам'яттю

Такі автомати складаються із сукупності елементарних КВ − автоматів, які пов'язані між собою певними функціональними зв'язками.

Модель структури елементарного КВ − автомата представимо у вигляді граф − схеми (рис 2.1), де прийняті такі літерні позначення: $G_0(t)$,$G_1(t)$ − внутрішні стани КВ − автоматів.

Квантрон перебуває в стані спокою, коли $G_0(t) = 1$ й $G_1(t) = 0$, і перебуває в збудженому стані, коли $G_0(t) = 0$ і $G_1(t) = 1$, $X_1(t)$ − вхідні функції КВ − автоматів,

$Y_{01}(t)$і $Y_{10}(t)$ перехідні функції в момент фіксації й видачі інформаційних сигналів.

Як видно із цієї граф − схеми, перехідні функції в момент часу t + 1 можуть бути сигнал, але не видає його на виході. Якщо КВ − автомат перебуває в одиничному станій на його вхід надходить вхідний сигналX1 (t) цей автомат (маркерний режим роботи) зі стану G1 (t) переходить у стан G0 (t) . У цей час перехідна функція Y01 (t) =0, а вихідна функція Y10 (t)= 1 , тобто КВ −автомат, що перебуває в стані фіксації, збуджується вхідним сигналом і на виході видає інформаційний сигнал.X1 (t) =1 . У момент надходження вхідних інформаційних сигналів перехідні функції перетворюються у вихідні функції. При дотриманні певних умов цей автомат або приймає, або фіксує, але на виході не видає сигналів.

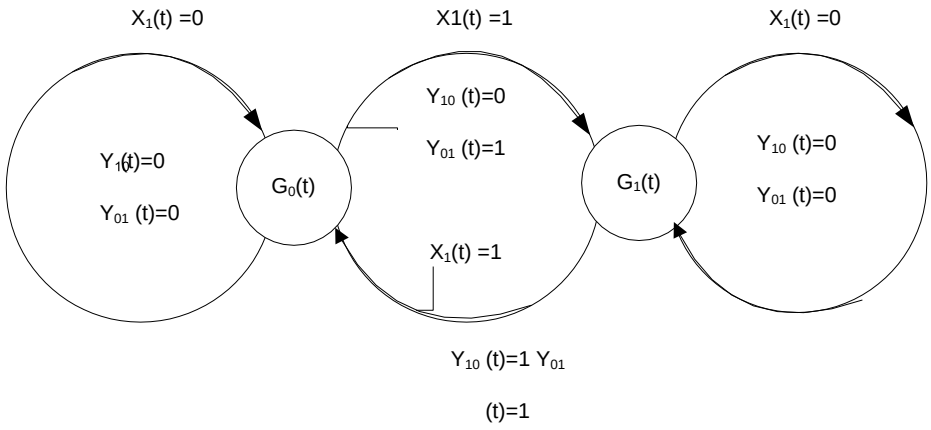


Рисунок. 1.4 - Модель структури елементарного КВ − автомата

На рисунку 1.4 граф − схема показує характер функціонування КВ − автомата в автоматичному маркерному режимі, тобто при впливі вхідних збурювань, збережений у якийсь момент часу *t+1* інформаційний сигнал видається, а сам автомат вертається (автоматично) у вихідний стан (такий режим роботи називається маркерним режимом роботи КВ - автомата. Якщо ж в момент видачі сигналів автомат залишається в поточному стані, то такий режим роботи прийнятий називати шторочним [48].

У маркерному режимі роботи КВ - автомата перехідні (вихідні) функції задовольняють такі умови:

$$Y_{01}(t)=\begin{cases} 0, \text{коли } X_1(t)=1, \text{ при } G_0(t)=1, G_1(t)=0, \\ 1, \text{коли } X_1(t)=1, \text{ при } G_0(t)=0, G_1(t)=1; \end{cases}$$

$$Y_{10}(t)=\begin{cases} 0, \text{коли } X_1(t)=1, \text{ при } G_0(t)=0, G_1(t)=1, \\ 1, \text{коли } X_1(t)=1, \text{ при } G_0(t)=1, G_1(t)=0. \end{cases}$$

Квантрон - це автомат, графічна схема якого показана на рис. 1.4, який не повністю відображає характер зміни перехідних функцій оптоелектронних схем. Особливо тоді, коли існують складні взаємозв'язки між функціональними елементами, що створюють складні інформаційні оптико-електричні сигнали.

Зрозуміло, що для повного функціонального опису перехідних функцій складних оптоелектронних схем також доцільно розглядати складні математичні моделі КВ-автоматів, що є моделями функціональних операційних схем оптоелектронних процесорів [49]. При русі інформаційних сигналів, складний оптоелектронний вузол, що реалізує ряд додаткових функцій (частково або повністю відсутній в електронних вузлах) можна найбільш повно моделювати тільки за допомогою двох, трьох, чотирьох і т. д. елементарних КВ − автоматів. Такі складні КВ − автомати, як показано в роботі [49], іменуються: диквантрон − $DA^k(t)$, триквантрон − $TA^k(t)$, тетраквантрон − $T1A^k(t)$ і т. д. квантрон −автоматами.

(тобто вхідні сигнали, коли КВ − автомат перебуває в 0 - му і 1 - му станах.). А за допомогою спеціальних операторів d і D досить зручно описати перехідні процеси в складних КВ - автоматах [84].

Як конкретний приклад практичної реалізації КВ-автомата можна вказати синтезовану площинну матрицю, створену за допомогою квантрон-автоматів і використовуючи результати теорії ущільнення даних.

Цей підхід може бути використаний для оптимізації технічних засобів індикації в графічному вигляді. (рис. 1.6) [48].

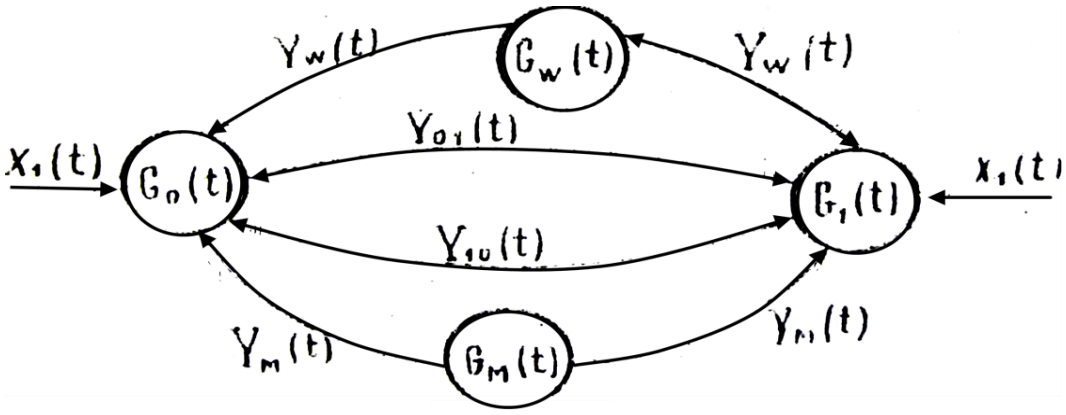


Рисунок. 1.5 - Граф - схема елементарного КВ – автомата

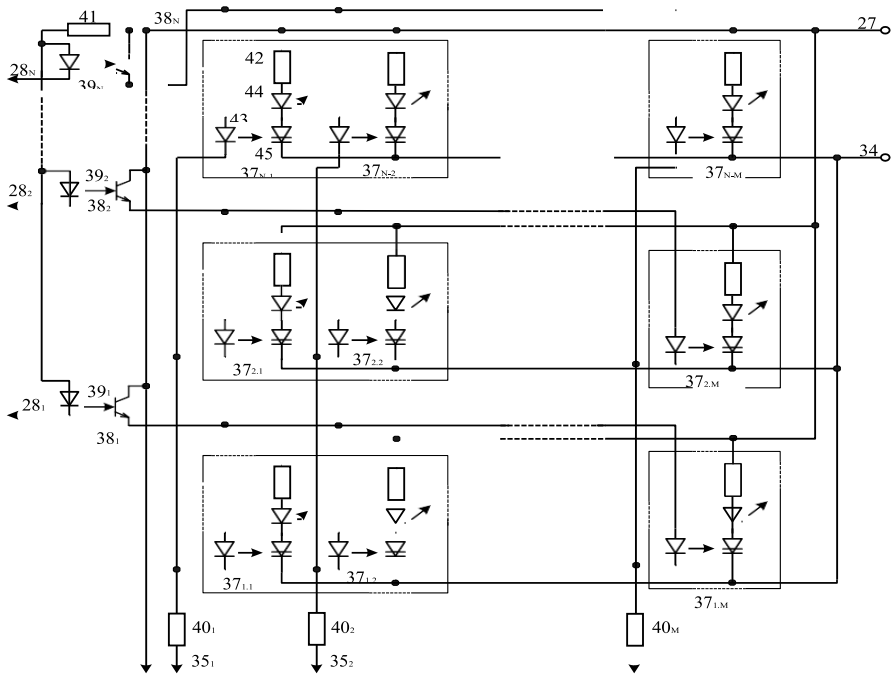


Рисунок 1.6 - Схема асинтезованої площинної матриці

Ця розробка на КВ - автоматах може використовуватись для зберігання рухомих зображень в цифрових носіях пам’яті з використанням стандарту JPEG та MPEG [26 – 49].

Висновки до розділу

У першому розділі вивчалися основні концепції та терміни в сфері стеганографії, а також проведено аналіз існуючих методів вбудовування інформації в зображення. Зазначений аналіз свідчить, що наявні методи відзначаються високою стійкістю, але виявлено суттєві обмеження, особливо у відношенні обсягу інформації, який можна вбудувати.

Також розглянуто ортогональне перетворення Уолша-Адамара з метою ущільнення та зміни зображень для їхньої передачі. У наступному розділі ми докладніше розглянемо методи та засоби реалізації даного підходу.

Постановка задачі

У зв'язку з тим, що виникла потреба в підвищенні швидкості виконання перетворень, які у стеганографічних методах становлять найбільш складний компонент, виникає необхідність розгляду можливостей збільшення ефективності методу. Першочергове вдосконалення полягатиме в стисненні зображення, а після цього виконуватиметься приховування інформації в цьому зменшеному зображенні.

РОЗДІЛ 2. РОЗРОБКА МЕТОДУ ПЕРЕТВОРЕННЯ ЗОБРАЖЕНЬ НА ОСНОВІ НЕЙРОННОЇ МЕРЕЖІ

Нейронна мережа представляє собою послідовність алгоритмів, які намагаються визначити основні взаємозв'язки в наборі даних, імітуючи спосіб функціонування людського мозку. Ці мережі можуть адаптуватися до змін у вхідних даних, що дозволяє їм генерувати оптимальний результат без необхідності повторного проектування вихідних критеріїв.

Штучні нейронні мережі базуються на особливостях, які сприяють їх успішному вирішенню нерегулярних завдань [22, 23]:

- простий обчислювальний елемент - нейрон;
- велика кількість нейронів, які беруть участь в обробці інформації;
- кожен нейрон пов'язаний з великою кількістю інших нейронів (глобальні зв'язки);
- змінювані ваги зв'язків між нейронами;
- паралельна обробка інформації великою кількістю нейронів.

Поведінка штучної нейронної мережі залежить від значень параметрів та функції активації нейронів. Три основних види функцій активації включають граничні, лінійні та сигмоїдальні. Граничні елементи встановлюють вихід на одному з двох рівнів в залежності від того, чи більше чи менше сумарний сигнал на вході нейрона певного граничного значення. Лінійні елементи виражають вихідну активність пропорційно сумарному зваженому входу нейрона. Для сигмоїдальних елементів характерно те, що вихід безперервно змінюється відносно вхідного сигналу, проте не лінійно при зміні входу. Сигмоїдальні елементи мають більше спільностей з реальними нейронами, ніж лінійні або граничні, проте кожен з цих типів можна розглядати лише як наближення. Нейронна мережа представляє собою систему великої кількості відносно простих елементів - нейронів, топологія з'єднань яких залежить від типу мережі. Для розв'язання конкретного завдання потрібно вибрати, як нейрони будуть з'єднані між собою і відповідним чином налаштувати вагові параметри цих зв'язків. Вплив одного елементу

на інший може бути залежним від встановлених з'єднань. Вага з'єднання визначає силу впливу. Нейронні мережі дозволяють вирішувати складні задачі за час виконання ланцюжків електронних і/або оптичних елементів. Рішення мало залежить від несправності окремого нейрона, що робить їх ефективними для створення моделей штучного інтелекту.

Існують два способи реалізації нейронних мереж:

перший - програмна модель нейронних мереж;

другий - апаратна реалізація.

Дослідження в галузі нейронних мереж для ущільнення зображень прогресують у різних напрямках, зокрема:

- Розробка нейроалгоритмів;

- Створення спеціалізованого програмного забезпечення для моделювання нейронних мереж;

- Розробка спеціалізованих процесорних плат для імітації нейромереж;

- Електронні реалізації нейронних мереж;

- Оптоелектронні реалізації нейронних мереж [31, 32].

В галузі наукових досліджень, спрямованих на створення обчислювальних систем шостого покоління, що відомі як нейрокомп'ютери, активно розглядається концепція нейрокомп'ютингу. З переходом до оптичних носіїв інформації виникає можливість реалізації око-процесорних образних нейрокомп'ютерів на квантронах (квантових процесорах) [35, 36].

Нейрокомп'ютери представляють собою системи з великою кількістю простих обчислювальних елементів, які виконують роботу паралельно. Ці елементи, що взаємодіють між собою, формують нейронну мережу, яка спроектована для виконання обчислювальних завдань без зовнішнього керування, забезпечуючи високу швидкодію.

Перші спроби створення паралельних оптико-електронних квантронних комп'ютерів нейроподібного типу на операційних екранах в Європі були проведені в СКТБ «Квантрон» на початку 80-х років і завершилися створенням макетного пристрою в 1985 році (проект "Перетворювач") [47]. На сьогодні розробка нейрокомп'ютерів

активно ведеться в більшості розвинених країн світу і підтримується численними міжнародними та національними програмами. У різних сферах вже успішно експлуатується понад 50 нейросистем - від фінансових прогнозів до експертизи [23].

Нейрокомп'ютери ефективно вирішують різноманітні інтелектуальні завдання, такі як розпізнавання образів, ущільнення зображень, адаптивне керування, прогнозування, діагностика і т.д. Особливо вразливими в цьому відношенні виявилися паралельно-ієрархічні перетворення, запропоновані професорами В. П. Кожем’яком і Л. І. Тимченком [49].

Нейрокомп'ютери відрізняються від попередніх поколінь ЕОМ не лише збільшеними можливостями, але і принципово новим підходом до використання машини. Замість традиційного програмування, важіль керування переходить до процесу навчання, де нейрокомп'ютер самостійно набуває навичок у вирішенні завдань.

Навчання полягає у постійному внесенні коректив ваг зв'язків, що призводить до формування відповідного вихідного сигналу для кожного вхідного впливу [48]. Для теоретичної аналізу роботи складних оптоелектронних вузлів можна використовувати моделі більш складних КВ-автоматів [49].

2.1 Модель абстрактного КВ - автомата в логіко − часовому середовищі

Відомі методи синтезу цифрових автоматів базуються на табличному описі їхньої роботи і не є оптимальними для автоматів, де всі сигнали визначаються логіко-часовими функціями. Скінчений автомат може бути описаний як пристрій з вхідним та вихідним каналами, що функціонують в кожний дискретний момент часу (тактовий момент) у певному стані. Кожен такт вхідні сигнали подаються на вхідний канал, визначається закон переходу між станами від поточного моменту до наступного залежно від вхідних сигналів і стану пристрою у попередньому моменті, а також значення вихідного сигналу в поточний тактовий момент, що є функцією як стану, так і вхідного сигналу. Розрізняють різні підходи до визначення скінченного

автомата, і ми цікавимося переважно макропідходом, який досліджує зовнішню поведінку пристрою, процес обробки вхідної інформації у вихідну та послідовність його станів. При такому підході виникає поняття абстрактного скінченного КВ-автомата [49].

Найбільш перспективною є реалізація моделі абстрактного автомата у логіко-часовому базисі, використовуючи матричу форму представлення ЛЧФ [49], яка ґрунтується на ідеї Δ-розбиття.

Характеристики квантрона

Оптоелектронний квантрон представляє собою функціонально-повний елемент в цифрових пристроях квазіімпульсно-потенційного типу [32, 43]. Ми розглядаємо характеристики оптронного квантрона, який зображений на схемі на рис. 3.2. У випадку відсутності світлового потоку, величина "темнового" струму фотодіода (ФД) іф не досягає достатнього рівня для активації транзистора, тому струм через світлодіод (СІД) становить Іс, що дорівнює Іо, де Іо - граничне значення. Світлодіод не випромінює, що відповідає нульовому значенню Ввих. Якщо вхідний світловий потік впливає на ФД, то фотострум $i_ф$ підсилюючись транзистором, досягає значенні I_c I_o . СІД збуджується й $B_{вих}$ відповідає одиничному значенню.

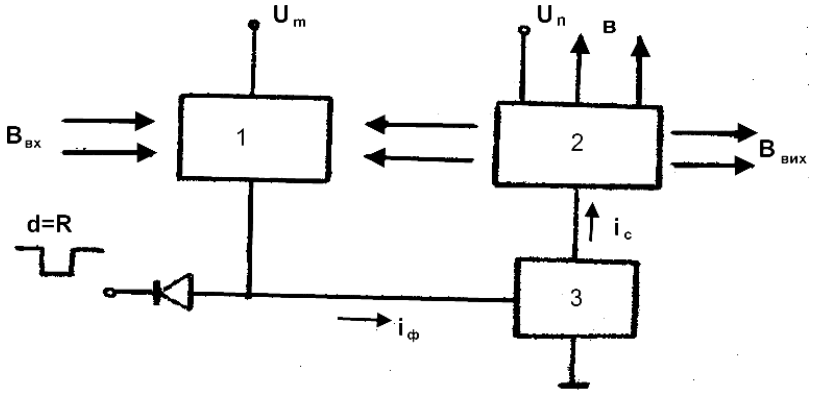


Рисунок. 2.1.- Приклад оптоелектронного квантрона

На рис. 2.1 прийняті позначення: 1 − фотоелемент, 2 − світловипромінювач, 3 − комутатор, U_m – напруга модуляції, U_n – напруга живлення, $B_{вих}$ – вихід променистого потоку індукції, *в* − вхід імпульсу скидання, $B_{вх}$ − вхід порушення.

Давайте розглянемо вплив конструкторсько-технологічних факторів на технічні

характеристики компонентів оптронного квантрона.

Механізм інжекційної люмінесценції в світлодіоді (СІД) включає три процеси: випромінювальну рекомбінацію, інжекцію надлишкових неосновних носіїв у базу світлодіода та вивід випромінювання з області генерації. Випромінювальну рекомбінацію характеризує внутрішній квантовий вихід, що визначається відношенням імовірності випромінювальної рекомбінації до повної рекомбінації. Внутрішній квантовий вихід є ключовою характеристикою матеріалу, використовуваного у світлодіоді. У випадку високочистих матеріалів для світлодіода.

Визначимо квантрон як елемент індикації інформаційно-обчислювальної техніки, що виконує додатково функції генерації імпульсів та запам'ятовування інформації, включаючи оптичні інформаційні та електричні керуючі входи. Принцип сполучення функцій пам'яті й генерації в одному елементі індикації здійснений за допомогою бістабильної схеми [46] , що містить діодний оптрон: світловипромінювальний діод (СІД), диференціальний фотодіод (ФД), підсилювач на транзисторі (Т), опір зсуву (R). Входи $U_{упр}$ і «запуск» служать для визначення режиму роботи схеми.

Аналіз функціональних властивостей квантрона

Основні принципи роботи квантрона можна сформулювати наступним чином:

1. Частотний вихід:

- Інформація про рівні збудження кодується за допомогою серії нервових імпульсів, які мають фіксовану частоту.

2. Амплітуда та тривалість імпульсів:

- Амплітуда та тривалість окремих нервових імпульсів, які розповсюджуються по одному і тому ж волокну, залишаються постійними.

- Частота та кількість нервових імпульсів в послідовності залежать від інтенсивності збудження.

Якщо використовувати КВ − автомат (квантрон) у ролі нейрона, то основні параметри нейрона будуть:

1. Реобаза:

- Найменша сила постійного електричного струму, необхідна для збудження квантрона при достатньому часі впливу, в результаті чого генерується імпульс.

- Характеризує збудження квантрона.

2. Хронаксія:

- Найменший час, протягом якого постійний електричний струм, величина якого в дві реобази, діючи на квантрон, викликає вихідний імпульс.

- Характеризує швидкодію квантрона з малими збудженнями.

3. Лабільність:

- Максимально можлива кількість імпульсів, які може відтворити квантрон без трансформації ритму в одиницю часу, або максимальна, найбільш стійка частота імпульсів (поточна лабільність).

- Характеризує швидкодію квантрона під час великих збуджень. Кожен квантрон генерує імпульс лише з одним знаком. Зазвичай, дослідники спрощують модель квантрона, призводячи її до бінарної форми, і не обирають більшість з цих властивостей. Хоча визначення найбільш важливої характеристики квантрона є завданням з певною неоднозначністю, наявність частотного багатозначного виходу нейрона визначається як визначальний фактор при розробці моделей.

2.2 Способи реалізації нейронних мереж для перетворення зображень

Зазвичай нейрон представляється як елемент з наборами збуджуючих і гальмівних входів, де кожен вхід характеризується вагою (ω_i) та має один вихід. Вхідні значення зазвичай нормалізуються в межах від 0 до 1. Для багатофункціональних нейронних мереж потрібно значна кількість нейронних елементів (НЕ), і їх фізична реалізація впливає на складність, вартість та ефективність мережі. Незважаючи на це, вимоги до фізичної реалізації нейрона включають моделювання найбільш важливих функцій біологічного нейрона.

Класифікація відомих моделей нейронних елементів є об'єктом розгляду в літературі відомих учених. За рівнем складності НЕ можна поділити на прості і складні

моделі. Складні моделі спрямовані на реалізацію більшості процесів біологічного нейрона, але їх створення для великих масивів є важливим завданням. Інший підхід передбачає створення простих апаратних реалізацій НЕ, які зберігають основні функції, такі як суматор або пороговий пристрій і генератор імпульсних послідовностей. Більшість відомих апаратно простих реалізацій моделей нейронних елементів суттєво є бінарними пороговими пристроями. Однак таке спрощення може призвести до збільшення розмірів нейронної мережі, орієнтованої на вирішення конкретного функціонального перетворення, у порівнянні з мережею на основі повнофункціональних НЕ.

З цієї точки зору важливими є нейрони з частотним виходом, де частотно-динамічна модель нейронного елемента визначається динамікою частоти вихідних імпульсів в залежності від рівня та тривалості вхідних сигналів. Ця характеристика має особливе значення серед інших властивостей біологічного нейрона. Частотно – динамічні нейронні елементи мають набагато більше функціональних можливостей в порівнянні з бінарними, що дозволяє будувати на їх основі нейронні мережі з меншою кількістю НЕ.

Для створення нейронних мереж на основі частотно-динамічних нейронів, останні повинні бути максимально простими та легкоінтегрованими. В цьому контексті особливо важливі властивості біспін-пристроїв. Біспін-пристрій є новим напівпровідниковим триполюсним елементом, який може виконувати роль інтегратора, порогового елемента та генератора імпульсів одночасно. В ряді досліджень [43, 49, 11] висунуто оптоелектронну модель нейрона на базі біспін-пристрою, яка має частотний вихід і відрізняється простотою реалізації та широким спектром функціональних можливостей.

Отже, при апаратній реалізації нейронних мереж для ущільнення зображень використання оптоелектронної елементної бази вважається перспективним напрямком.

2.3 Спосіб реалізації методу кодування і передавання інформації із захистом

Спочатку розглянемо блок-схему кодування – декодування зображень. Блок-схема системи кодування за допомогою перетворення для передачі зображень представлена на рисунку. 2.5. Перетворення здійснюється над усією кількістю елементів зображення або при кодуванні блоками повторюється окремо для кожного блоку [49].

Схема на рисунку 2.5 є фактично узагальнена схема, яка передбачає розробку способу кодування і передавання інформації із захистом та пристрій для його здійснення (патент Кулік).

Пропонується використання оптоелектронного десяткового матричного індикатора сигналу для реалізації нейронних мереж. Цей індикатор дозволяє представляти сигнал у двовимірному форматі. У такому індикаторі логіко-числові коди взаємодіють, першопочатково генеруючись на оптоелектронному лічильному тригері. Лічильний тригер виступає як основний елемент оптоелектронного лічильника імпульсів та цифрового вимірювача одиничних часових інтервалів. Утворена ієрархічна система створюється за допомогою паралельного ущільнювача часоімпульсних сигналів.

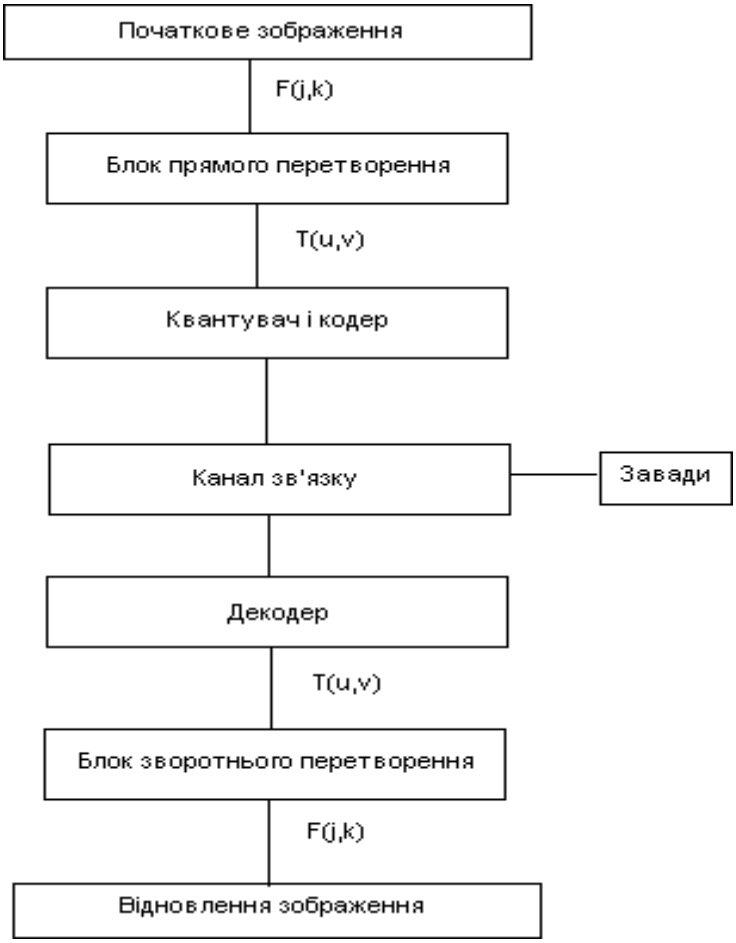


Рисунок. 2.5 - Система кодування на базі перетворення для передачі зображень.

Зазвичай процес перетворення зображень включає розподіл зображення на часткові підобласті або фрагменти для їхньої окремої обробки. З метою спрощення процесу обробки та полегшення апаратної реалізації використовують двовимірні або одновимірні фрагменти. У випадку двовимірного фрагмента загальна кількість елементів становить M, розташованих на N стрічках одного поля зображення. Одновимірний фрагмент, натомість, охоплює обробку конкретної кількості M елементів зображення однієї телевізійної стрічки.

Теоретично формат оброблюваних фрагментів визначається діапазоном виявлення існуючих статистичних залежностей між елементами зображення, а практично - ступенем складності апаратних засобів. В граничному випадку, використовуючи кодування з перетворенням, можна обробити весь обсяг або декілька обсягів. Результатом перетворення зображень є перехід відеоінформації з часової області в спектральну. Отримані результати представлені спектральними

коефіцієнтами перетворення, які називають трансформантами і визначаються амплітудою та узагальненою просторовою частотою у спектральній області.

Для основи перетворення зображень можуть застосовуватися різні принципи. Найчастіше використовуються методи лінійних ортогональних перетворень, і в даному випадку використовується перетворення Уолша-Адамара.Лінійність ортогонального перетворення означає, що операції додавання, віднімання і множення на скаляр дійсні і після перетворення, а ортогональність - що перетворений фрагмент представляється обмеженим набором ортогональних функцій. Унітарність ортогонального зображення означає схожість математичних апаратів при прямому і зворотньому перетвореннях. Лінійне перетворення можливо здійснити в загальному випадку, з безперервним або дискретним сигналом, процесу перетворення при цьому буде відповідати або інтегральна, або матрична форма запису.

1. Лінійні ортогональні перетворення характеризуються тим, що між елементами зображень ліквідуються статистичні залежності і розподілення енергії в перетвореному спектральному фрагменті є нерівномірним. Ці особливості використовуються в процесах кодування.

Далі для векторного квантування ми використовуєм нейронні мережі на основі карт Кохонена.

Розробка двовимірних карт Кохонена покомпонентним кодуванням на основі нейронної мережі. Схема ущільнення зображень з використанням карти Кохонена представлена на рис. 2.6. Після векторизації, коли блоки зображення перетворюються у вектори, застосовується векторне квантування з використанням карти Кохонена. Вихідні дані від векторного квантувача передаються на арифметичний кодер, який забезпечує кодування зображення без втрат. Процес декодування виконується у зворотньому порядку [48, 49].

У вихідний файл додаються квантовані значення вхідних векторів і кодова книга. Однак розмір кодової книги невеликий порівняно з вхідним зображенням, і це не впливає на коефіцієнт ущільнення. Розмір векторного квантувача - це карта

Кохонена розміром 16x16 або більше. Вибір розміру карти Кохонена пояснюється тим, що зображення представлено з точністю 8 біт на елемент зображення для напівтонових зображень або 24 біти для кольорових.

Використання арифметичного кодера на етапі ущільнення без втрат забезпечує найбільший коефіцієнт ущільнення порівняно з іншими методами кодування без втрат [38].

Існують різні підходи до використання нейронних мереж для ущільнення зображень [100]. Проте, особливою увагою вирізняються підходи, які ґрунтуються на принципах векторного квантування зображень, оскільки це забезпечує високу швидкодію ущільнення при збереженні високої якості відновленого зображення. Ідея векторного квантування є досить простою. Зображення поділяється на квадратні блоки, наприклад, 2x2, 4x4 або 8x8. Кожен блок розглядається як вектор у 4-вимірному, 16-вимірному або 64-вимірному просторі. З цього простору обирається обмежена кількість векторів, які становлять кодову книгу, з метою найточнішого апроксимування векторів, які вилучаються із вхідного зображення. У каналі зв'язку або файлі зберігаються номери векторів з кодової книги, які мають найменшу відстань від векторів, вилучених із вихідного зображення, та сама кодова книга. Оскільки кількість векторів у кодовій книзі значно менша за загальну кількість векторів у вихідному зображенні, то для подання номера вектора витрачається менше бітів, ніж для початкового вектора, що забезпечує ущільнення**.** Ідеальними для вирішення цих завдань є нейронні мережі, що самоорганізуються, запропоновані фінським ученим Т. Кохоненом, а саме самоорганізуюча мережа у вигляді двовимірної карти Кохонена.

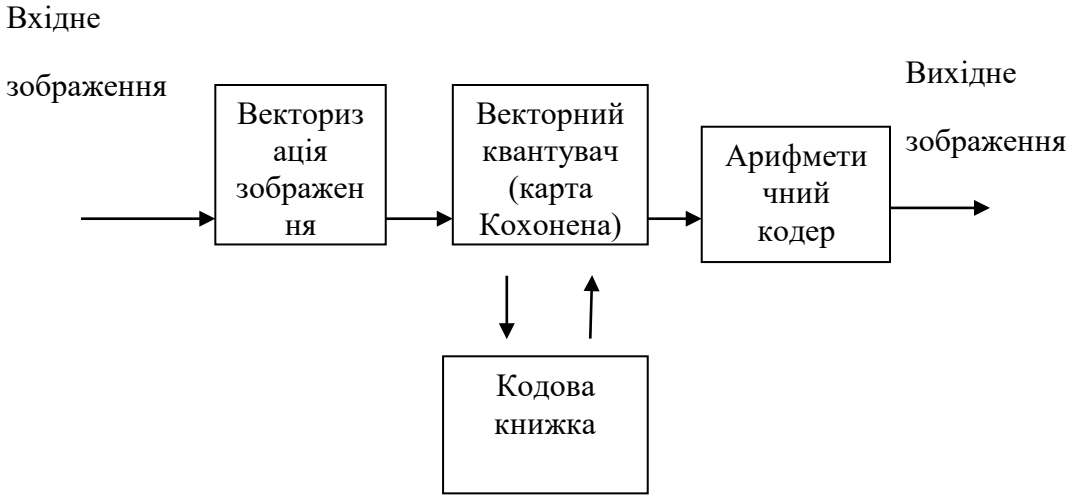


Рисунок. 2.6. - Загальна схема кодування

Карта Кохонена використовується у методах векторного квантування при ущільненні зображень, і вона володіє двома ключовими властивостями. По-перше, вона подібна до інших методів векторного квантування, що застосовуються при ущільненні зображень із втратами. По-друге, близькі кластери вхідних векторів корелюють із близько розташованими нейронами, що підвищує ефективність ущільнення без втрат на наступному етапі кодування [49, 35].

Самоорганізуюча карта Кохонена (SOFM) складається з набору вхідних елементів, кількість яких відповідає розмірності вхідних векторів, і набору вихідних елементів, які виступають у ролі прототипів. Базова архітектура мережі SOFM показана на рис. 2..

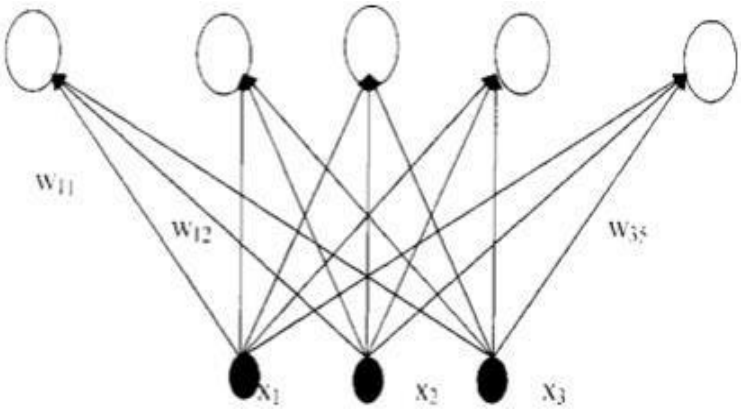


Рисунок. 2.7. Базова архітектура мережі SOFM

Вихідні елементи називаються кластерними елементами. Кластерні елементи або

кодові слова розміщуються у вигляді одно − або двовимірного масиву. Звичайна кількість кластерних елементів значно менша в порівнянні з кількістю навчальних зразків, оскільки метою є одержання спрощеної характеристики вхідних даних.

Це і дає можливість використання SOFM, як векторного квантувача. Після навчання дана мережа може апроксимувати вектори вхідного простору найкращим способом. Кожний нейрон представляється ваговими коефіцієнтами w_{ij} .Алгоритм навчання мережі такий:

1. Ініціалізувати вагові коефіцієнти випадковими значеннями.
2. Для кожного кластерного елемента обчислити відстань від навчального вектора:

$$d_j = \sum_i (w_{ij} - x_i)^2 \qquad (2.1)$$

3. Знайти кластерний елемент j для якого d_j, мінімальне.
4. Для кластерних елементів з кола заданого радіуса із центром в j елементі обновити вагові коефіцієнти відповідно до формули:

$$w_{ij}(n+1) = w_{ij}(n) + ή(n)[x_i - w_{ij}(n)], \qquad (2.2)$$

де ή- норма навчання, x_i - координата навчального вектора.
5. Обновити норму навчання (і радіус при необхідності) і повторити пункти 1 -5 для наступного навчального вектора.

Норма навчання із часом змінюється. Вона може, наприклад, міть значення 0,9, а потім змінюватися лінійно до деякого фіксованого значення, наприклад 0,01, після чого залишатися незмінної. Радіус також спочатку вибирається досить більшим, щоб обновлялися всі елементи. Із часом радіус зменшується й наприкінці повинен обновлятися тільки сам елемент-переможець.

2.4 Розробка алгоритму ущільнення зображення за допомогою векторного квантування на основі нейронної мережі SOFM

Для розробки алгоритму необхідно вирішити кілька питань:

1. Який розмір карти Кохонена вибрати?

2. Яка розмірність вхідного вектора?

3. Яка розрядність вагових коефіцієнтів карти Кохонена?

Розрядність вагових коефіцієнтів вибирається з урахуванням розрядності вхідних даних. Оскільки зображення звичайно представляються у форматі 8x8x8, тобто тридцятилітні кольорів RGB представляються одним байтом, те й розрядність вагових коефіцієнтів вибираємо рівної 8-і биткам.

Розмірність вхідних векторів вибирається з урахуванням кореляції вхідних даних. Відомо, що найбільшими кореляційними зв'язками характеризуються сусідні відліки зображення, які можуть мати близькі значення [43,44]. Тому зображення розбивається на квадрати, що примикають, розміром 2x2, кожний з яких розглядається як вектор в 4-мірному просторі.

Розмір карти Кохонена визначається мінімальною кількістю розрядів, що представляє елемент зображення й забезпечує достатня якість зображення. З наукової літератури відомо, що менше 2 біт на елемент зображення при будь-яких перетвореннях досягти проблематично при заданих високих вимогах до якості зображення [45]. До того ж збільшення розміру карти приводить до значної втрати швидкодії навчання мережі. Швидкість навчання прямо пропорційна квадрату розміру сторони карти , тобто:

$$Тн = k\ N^2,$$

де N - розмір сторони карти, k - коефіцієнт пропорційності, залежить від конкретної реалізації.

З врахуванням вище сказаного максимальний розмір карти Кохонена, що задовольняє цим суперечливим вимогам, вибираємо 16x16. Оскільки розмір вхідного вектора дорівнює чотирьом, то карта при векторному

квантуванні забезпечує таку кількість біт на елемент зображення:

$$M = \log_2 N^2/n = 8/4 = 2 \text{ біти/ел.}$$

що прийнятно, як з погляду швидкодії, так і забезпечення високої якості відновленого зображення.

Векторне квантування з використанням карти Кохонена виконується за два проходи вихідного зображення:

- перший прохід - навчання мережі;
- другий прохід - векторне квантування.

Причому, навчальними векторами є всі фрагменти зображення з розмірами 2x2. Таким чином, алгоритм кодування буде таким:

Етап навчання

1.Ініціалізувати вагові коефіцієнти нейронів випадковими значеннями.

2. Вибрати із зображення перший фрагмент 2x2.

3. Представити його у вигляді навчального вектора.

4.Для кожного кластерного елемента карти обчислити відстань від навчального вектора:

$$d_j = \sum_{i=0}^{3} (w_{ij-x_i})^2$$

Знайти кластерний елемент j для якого d_j мінімальне.

5. Для даного кластерного елемента обновити вагові коефіцієнти відповідно до формули:

$$W_{ij}(n+1)=w_{ij}(n)+ή(n)[x_j-w_{ij}(n)]$$

де ή- норма навчання, x_i - координата навчального вектора.

6. Обновити норму навчання й вибрати із зображення наступний

фрагмент 2x2 і повторювати пункти 3-6 для наступних навчальних векторів доти, поки не будуть обрані всі фрагменти.

Векторне квантування

7. Вибрати із зображення перший фрагмент 2x2
8. Представити його у вигляді навчального вектора.
9. Для кожного кластерного елемента карти обчислити відстань до навчального вектора:

$$d_j = \sum_{i=0}^{3} (w_{ij} - x_i)^2$$

10. Номер кластерного елемента з мінімальним d_j, записати у вихідний файл.
11. Вибрати із зображення наступний фрагмент 2x2 і повторювати пункти 8-11 доти, поки не будуть обрані всі фрагменти.
12. Записати у вихідний файл значення коефіцієнтів W_{ij} - усього 1024 байти.
13. Записати у вихідний файл розмір початкового файлу.
14. Стиснути отриманий файл методом арифметичного кодування.

Декодування виконується значно швидше й включає такі етапи:

1. Декодувати стислий файл арифметичним методом.
2. Прочитати із вхідного файлу й записати у відповідні масиви значення номерів кластерних елементів, коефіцієнтів W_{ij} і розмір початкового зображення.
3. Вибрати з масиву номер кластерного елемента для першого фрагмента 2x2.
4. Коефіцієнти W_{ij} даного кластерного елемента (4 коефіцієнти) записати у вихідне зображення як значення елементів відповідному фрагменту 2x2.
5. Вибрати наступний номер кластерного елемента й повторювати пункти 4-5 доти, доки не будуть відновлені всі фрагменти

зображення.

Схема алгоритму показана на рисунку 2.8.

Рисунок 2.8. – Алгоритм ущільнення зображення на основі нейронної мережі типу карти Кохонена

Висновки до розділу

1. Проведений аналіз основних підходів до апаратної реалізації ущільнення зображень на основі нейронних мереж і оцінки їх складності.

2. Запропоновано оптоелектронні реалізації елементів і вузлів системи ущільнення зображень на основі карти Кохонена, які характеризуються простотою технічної реалізації й високим ступенем паралелізму.

3. Представлений алгоритм кодування зображень з використанням перетворення Уолша-Адамара на основі нейронної мережі Кохонена. Векторний квантувач на основі нейронних мереж типу карти Кохонена — це алгоритм стиснення зображень, який використовується для зменшення розміру зображення, зберігаючи при цьому його ключові характеристики. Основним компонентом такого алгоритму є карта Кохонена (SOM) або самоорганізуюча карта, яка навчається адаптуватися до структури вхідного зображення.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ДОДАТКУ ДЛЯ ПРИХОВУВАННЯ ІНФОРМАЦІЇ В ЗОБРАЖЕННЯХ

3.1 Реалізація алгоритму і програми кодування зображень з використанням перетворення Уолша-Адамара

Для написання програми кодування зображень в середовищі операційної системи Windows 10, можна використовувати такому найбільш поширені мови: C++, Object Pascal, та Java.

Програмне середовище Delphi має зручний інтерфейс користувача і широкий вибір візуальних компонент, вона пристосовано для написання програм з дружнім інтерфейсом і складними математичними обчисленнями.

Мова програмування Delphi візуальна, тобто при розробці програмного забезпечення не потрібно малювати кнопки, панелі і інші об'єкти, звичайні для нашого ока. І додатково набір стандартних компонент містить широкий спектр компонент по роботі із зображеннями, що дуже важливе при рішенні даної задачі. Ключовою особливістю Delphi є можливість створення нових компонент. Така можливість дозволяє не переходити в інше середовище розробки, а навпаки, вбудовувати нові інструменти в існуюче середовище. Крім того, можна поліпшити або цілком замінити компоненти Delphi, що існують за умовчанням. Крім того, перевагою Delphi є те, що кінцева програма вимагає лише одного exe-файлу. Тобто, для вирішення наукових завдань поза сумнівом переваги має середовище програмування Delphi.

Для доказу переваг, запропонованих підходів виконуватимуться дослідження в таких двох напрямах:

- дослідження ущільнення зображень на основі перетворення Уолша-Адамара з квантуванням компонент нерівномірними шкалами;
- дослідження на основі перетворення Уолша-Адамара з векторним квантуванням компонент зображення з використанням карти Кохонена.

При проведенні досліджень в трьох напрямах використовувалися однакові зображення, методика і устаткування.

Дослідження будемо виконувати в двох напрямках:

- з рівномірним квантуванням коефіцієнтів перетворення Уолша- Адамара;
- з векторним квантуванням коефіцієнтів перетворення Уолша- Адамара.

Розробка модуля перетворення Уолша-Адамара. При кодуванні зображень за допомогою перетворення Уолша-Адамара вихідне зображення розкладається на примикаючі один до одного фрагменти розміром 8*8 елементів в межах яких виконується двомірне перетворення Уолша-Адамара.

Швидке обчислення перетворення Уолша-Адамара в двомірному випадку зручно виконувати в два етапи. Спочатку виконується одномірне перетворення кожної стрічки (стовпця) фрагменту 8*8 елементів. Отримані коефіцієнти запам'ятовуються. Після обробки всіх стрічок фрагменту знову виконується одномірне перетворення кожного стовпця(стрічки) масиву коефцієнтів одержаних в результаті виконання першого етапу.

Кількість операцій при перетворенні масиву N*N при цьому складає

$$m_0 = 2N^2 \log_2 N$$

Без застосування швидкого обчислення перетворення Уолша-Адамара

$$m = N^2 (N^2-1).$$

При N=8, $m/m_0 = 10,5$

Тобто за рахунок швидкого перетворення кількість арифметичних операцій зменшена на порядок.

Час обміну з дисковими пристроями може бути зменшений в декілька раз зчитуванням оброблюваних файлів в масиви з використанням поцедури blockread, а після обробки записом масивів на диск з використанням процедури blockwrite, в яких значне збільшення швидкодії досягається за рахунок зменшення часу підводу

головок дискових пристроїв.

Одномірне швидке перетворення здійснюється в підпрограмі Ex_var.

З вихідного файлу DATA1.DAT зчитується фрагмент зображення рівний 40 стрічкам і записується в масив H1. Після чого управління передається процедурі прямого ПУА First_pr40_320. В ній реалізовані слідуючі дії.На масив H1 накладається маска розміром 8*8, пострічно зчитується по 8 елементів і застосовується до них швидкий алгоритм перетворення - процедуру Ex_var. Проміжні результати зберігаються в динамічних змінних X1^[40,320]. Для отримання коефіцієнтів прямого ПУА необхідно знов поетапно накладати маску розміром 8*8, але вже на масив X1, зчитуючи по 8 елементів не по стрічках, як у попередньому випадку, а по 8 елементів стовбців і обробляти за допомогою процедури Ex_var, враховуючи перегрупування, при цьому кожний елемент домножуєм на константу нормування 1/8 згідно формули. Кожний коефіцієнт квантується на відповідне число рівнів за рахунок ділення на матрицю квантування Matr8_8[8,8]. Значення трансформант зберігаються в проміжній динамічній змінній KK1^[40*320], за допомогою якої формується вихідний файл даних

DAT2.DAT. Для обробки всього зображення описані вище дії повторюються п'ять раз.

Перетворення Уолша-Адамара є унітарним ортогональним перет- воренням, а це в свою чергу означає подібність математичних апаратів при прямому і зворотньому перетвореннях. Зворотнє ПУА реалізоване в процедурі Second40_320 аналогічно як і при прямому з єдиною різницею, яка полягає у тому, що при зчитуванні дані з файлу DAT2.DAT домножуються на матрицю квантування Matr8_8[8,8]. Результат зворотнього перетворення записується у файл DATA3.DAT.

Програма складається з таких програмних блоків:

- блоки, в яких реалізується перетворення Уолша-Адамара на базі застосування швидкого алгоритму Кулі-Т'юкі. Це такі процедури як

Ex_var, First_pr40_320, Second_pr40_320, Uolsh_Adamar;

- Ex_var – процедура, яка організована на базі алгоритму швидкого перетворення Кулі-Т'юкі для матриці розміром 1*8;
- First_pr40_320 – процедура, яка реалізує пряме перетворення Уолша-Адамара для фрагменту зображення розміром 40*320;
- Second_pr40_320 – процедура, яка реалізує зворотнє перетворення Уолша-Адамара для фрагменту зображення розміром 40*320;
- Uolsh_Adamar – процедура, яка реалізує пряме і зворотнє пе- ретворення Уолша-Адамара для фрагменту зображення розміром 200*320.

Текст програми наведено в додатку А.

Розробка модуля векторного квантування . Дослідження стиснення зображень з використанням векторного квантування коефіцієнтів перетворення Уолша-Адамара виконувалися як для схеми приведеною на рис. 3.5.

Блок ПУА виконує перетворення Уолша-Адамара для примикаючих один до одного фрагментів зображення розміром 8x8.

Блок рівномірного квантування виконує цілочисленне ділення кожного коефіцієнта перетворення Уолша-Адамара на свій „коефіцієнт квантування".

Блок групування коефіцієнтів виконує відбір коефіцієнтів однакової частоти у відповідності з такими виразами:

$$tmpj := (j \bmod 8) * 40 + j \operatorname{div} 8;$$

$$tmpi := (i \bmod 8) * 25 + i \operatorname{div} 8,$$

де i, j – поточні координати в площині зображення, $tmpj$, $tmpi$ - нові координати. Таке групування покращує роботу векторного квантувача на етапі навчання (див. главу x).

Вихідні дані векторного квантувача поступають на арифметичний кодер, який виконує кодування зображення без втрат. Декодування виконується в зворотному порядку.

При декодуванні після векторного деквантування коефіцієнтів перетворення Уолша-Адамара вони знов переміщуються на свої позиції в площині зображення, що необхідно для виконання процедури зворотного перетворення Уолша-Адамара. Їх координати обчислюються так:

```
tmpi := (i mod 25) * 8 + i div 25;
tmpj:= (j mod 40) * 8 + j div 40.
```

Векторне квантування з використанням карти Кохонена виконується за три етапи:

- навчання мережі;
- векторне квантування.

Причому учбовими векторами є всі фрагменти зображення згрупованих коефіцієнтів Уолша-Адамара з розмірами 2x2.

Розроблений алгоритм реалізують такі процедури:

1. Процес векторного квантування з використанням мережі SOFM реалізує процедура veckv. Для цього використовується функція KV4L, яка залежно від значення змінної ln виконує як корекцію вагових коефіцієнтів при навчанні так і простій пошук найближчого кластерного елементу при квантуванні.
2. Деквантовання виконує процедура deckv, яка реалізує відповідні пункти алгоритму.
3. Процедура TForm1.N1Click завантажує вхідне зображення у форматі *.BMP у компоненти Image, а процедура TForm1.N4Click виконує запис стислого файлу на диск. Причому при завантаженні вхідного зображення читається і вміст компонент Edit, який задає точність квантування. Оператор може змінювати точність квантування ввівши будь-яке число із заданого діапазону в компонент Edit перед завантаженням зображення.

4. Процедура TForm1.Open1Click призначена для завантаження стислого файлу. У даній процедурі реалізовано ряд функцій необхідних для декодування стислого файлу.
5. Процедура TForm1.OpenBMPIm21Click виконує завантаження файлів зображень у форматах, які підтримуються Delphi в компонент Image2.
6. Процедури TForm1.SaveBMPIm11Click, TForm1.SaveBMPIm21Click дозволяє зберегти зображення Image1 і Image2 у форматі *.BMP на диску.

Окрім перерахованих процедур програма реалізує ряд допоміжних функцій, необхідних при дослідженні зображення:

Процедура TForm1.CKO1Click обчислює оцінку середньо- квадратичного відхилення відновленого після кодування і початкового зображення.

Процедура TForm1.Save1Click формує різницеве зображення між початковим і вхідним зображенням і записує його у файл у форматі *.BMP.

Текст програми приведене в додатку А.

Керівництво оператора.

Файловий склад програми :

1. uavk.exe – головний виконавчий файл програми.

2. dmc.exe – модуль арифметичного кодування.

3. girl256.bmp, lena.bmp – тестові зображення.

Файли формату *.vec є закодованими і стислими з використанням запропонованого методу зображеннями. При стисненні без втрат використовувався алгоритм арифметичного кодування, який забезпечує якнайкращі результати при стисненні закодованих даних [109].

Для роботи програми необхідно комп'ютері на який повинна бути встановлена операційна система Windows 10.

Вхідними зображеннями є файли у форматі *.BMP з такими додатковими вимогами:

1. Розміри вхідного зображення - до 640x400 пікселів.

2. Роздільна здатність вхідного зображення - до 24bit/ел..

Для виконання програми потрібно виконати такі дії:

1. Створити в кореневому каталозі на диску d:\ директорію vect.
2. Скопіювати файли uavk.exe, dmc.exe, girl256.bmp і lena.bmp з дискети в цю директорію
3. Запустити на виконання файл uavc.exe.
4. Задати точність квантування, ввівши розмір карти Кохонена у компонент правий компонент Edit або або задавши будь-які значення коефіцієнтів матриці квантування.
5. За допомогою вкладиша меню *File* - *Відкрити* - відкрити графічний файл, який потрібно стиснути (рис. 3.1).

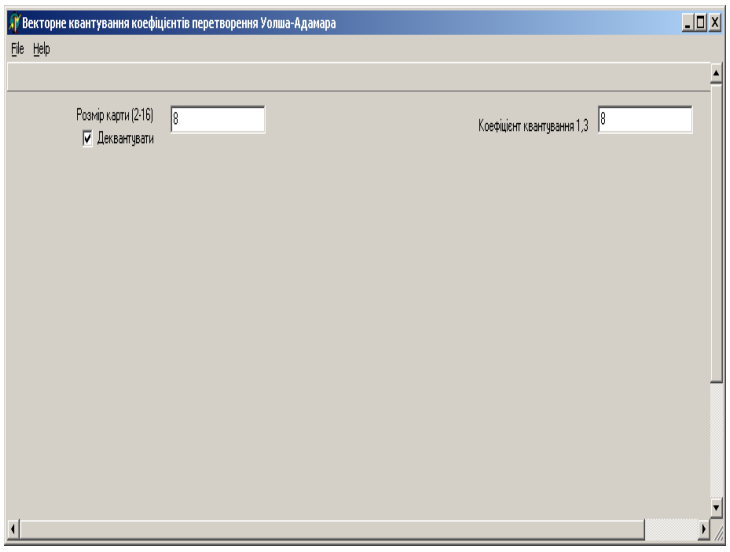


Рисунок. 3.1 - Головне вікно програми

5. За допомогою вкладиша меню *File* – *КомпресіяУАП* – кодувати зображення на основі перетворення Уолша-Адамара.
6. Використовуючи вкладиш меню *File* - *ДекомпресіяУАП* – декодувати зображення.
7. Вкладиш меню *File* - *КомпресіяУАПVK* призначений для кодування зображення на основі перетворення Уолша-Адамара з векторним квантуванням коефіцієнтів.
8. Вкладиш меню *File* - *ДекомпресіяУАПVDK* призначений для для декодування

зображення на основі перетворення Уолша- Адамара з векторним квантуванням коефіцієнтів.

9. Вкладиші меню *File – ОткрытьBMP_Im2, СохранитьBMP_Im1, СохранитьBMP_Im2* призначені для завантаження і запису зображень у форматі *.BMP у або з компонент Image1 і Image2 відповідно.

10. Вкладиш меню *File - СКВ* призначений для обчислення середньоквадратичного відхилення початкового зображення від відновленого після кодування.
11. Вкладиш меню *File -выход* – завершення роботи додатку (або значок “х” в правому верхньому кутку).
12. Для отримання інформації про розробника і програму необхідно вибрати пункт меню *Help*.

Вікно програми з завантаженим зображенням наведено на рис. 3.2.

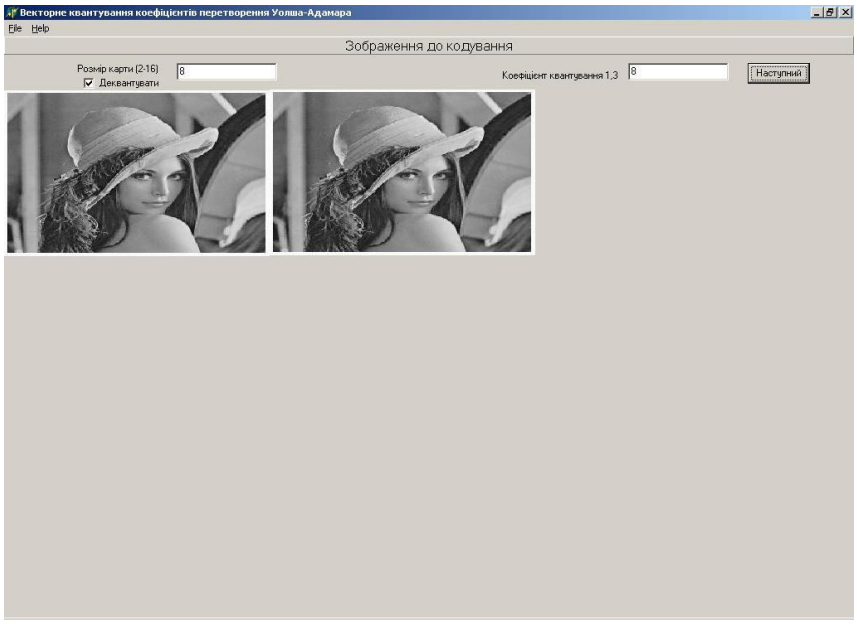


Рисунок. 3.2.- Вікно із завантаженим зображенням

Результати дослідження перетворення Уолша-Адамара з векторним квантуванням коефіцієнтів. При проведенні експериментальних досліджень використовувався комп'ютер з процесором Intel i5. Вхідне зображення представлене у форматі *.BMP. Якість відновленого зображення оцінювалася методом експертних

оцінок відповідно до 654 рекомендації МККР.

Виконувалися порівняльні дослідження залежності коефіцієнта ущільнення і якості відновленого зображення від точності квантування коефіцієнтів перетворення Уолша-Адамара векторним методом, а саме:

4.1.1.1 залежність середньоквадратичного відхилення і візуальної якості від об'єму кодової книги;

4.1.1.2 залежність векторного квантування від точності квантування коефіцієнтів перетворення Уолша-Адамара до векторного квантування.

Векторному квантуванню підлягають лише коефіцієнти з номерами i>1, j>1, оскільки коефіцієнти з меншими номерами вимагають високої точності. Згруповані коефіцієнти перетворення Уолша-Адамара після ділення на матрицю M[8x8] наведені на рис. 3.3.

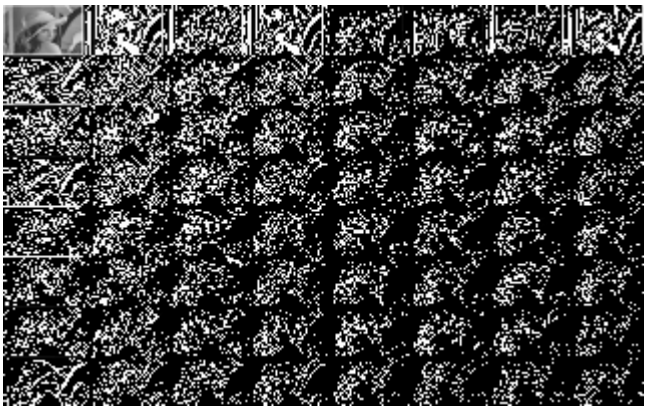


Рисунок. 3.3 - Згруповані коефіцієнти Уолша-Адамара

Дослідимо для цих коефіцієнтів залежність якості і коефіцієнта ущільнення відновленого зображення від розміру карти Кохонена (розміру кодової книги). Результати роботи алгоритму для файлу LENA.BMP при 4- вимірних вхідних векторах і розмірах карти від 2x2 до 16x16 представлені в табл.3.1, а також на рис. 3.4-3.5. На рис. 3.4 наведені квантовані векторним кантувачем коефіцієнти Уолша-Адамара, а на рис. 3.5 початкове і відновлене зображення.

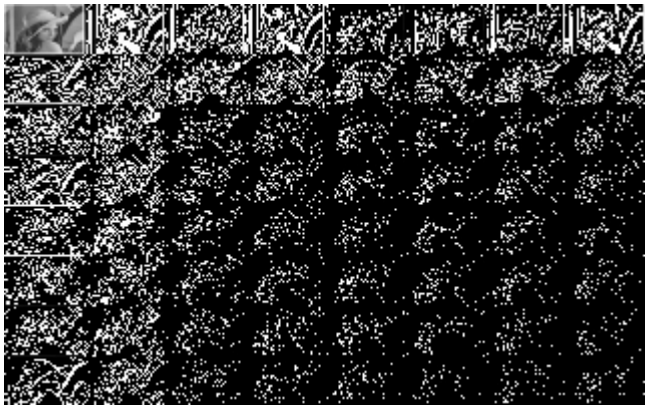


Рисунок. 3.4 - Коефіцієнти Уолша-Адамара після векторного квантування при розмірах карти Кохонена 8x8

Рисунок. 3.5 - Початкове і відновлене зображення при розмірах карти Кохонена 8x8, СКВ=5,74, розмір ущільненого зображення – 16296 байт

Таблиця 3.1 - Результати досліджень векторного квантування коефіцієнтів Уолша-Адамара при початковій матриці квантування М

Розмір початкового зображення, байт	Розмір зображення після ущільнення, байт	Середньо-квадратичне відхилення	Візуальна оцінка якості	Розмір карти Кохонена

64 000	21710	2,10	Відмінна	-
64 000	18493	4,54	Відмінна	16x16
64 000	16296	5,74	Відмінна	8x8
64 000	14486	7,17	Добра	4x4
64 000	13664	8,64	Добра	2x2
64 000	13106	5	відмінна	Jpeg

Результати свідчать, що застосування векторного квантування коефіцієнтів Уолша-Адамара в середньому на 25 % збільшує коефіцієнт ущільнення у порівнянні з рівномірним квантуванням без зниження візуальної якості зображення. Для збереження високої якості відновленого зображення достатньо матриці квантування з розмірами 8x8.

Результати досліджень векторного квантування при розмірах карти Кохонена 8x8 та різних матрицях рівномірного квантування наведені в табл.3.2

Таблиця 3.2 - Результати досліджень векторного квантування коефіцієнтів Уолша-Адамара при розмірах карти Кохонена 8x8

Розмір початкового зображення, байт	Розмір зображення після ущільнення , байт	Середньо-квадратичне відхилення	Візуальна оцінка якості	Матриця квантування
64 000	16296	5,75	Відмінна	M
64 000	15099	6,30	Відмінна	M1
64 000	14940	6,50	Відмінна	M2
64 000	15034	6,57	Відмінна	M3
64 000	13979	6,66	Добра	M4
64 000	14044	6,8	Добра	M5

Зменшення розміру карти Кохонена до 8x8, дозволить підвищити швидкість навчання мережі у декілька разів.

Цілий ряд експериментів з різними типами зображень показали, що коефіцієнти стиснення можуть знаходитися у межах 3-15. Для деяких зображень коефіцієнт стиснення перевершує стандарт JPEG при тій же якості зображення.

Приклади зображень наведені в додатку Б.

3.2 Розробка та дослідження алгоритму і програми для приховування інформації в зображення на основі нейронної мережі

Після ущільнення зображення виконаєм розробку програми для приховування інформації в зображення.

Розглянемо модель нейронної мережі, що виконує приховування текстового повідомлення у графічні файли. Модель реалізована з використанням потужного фреймворку PyTorch та платформи CUDA та зображена на рисунку 3.6. [50-55]

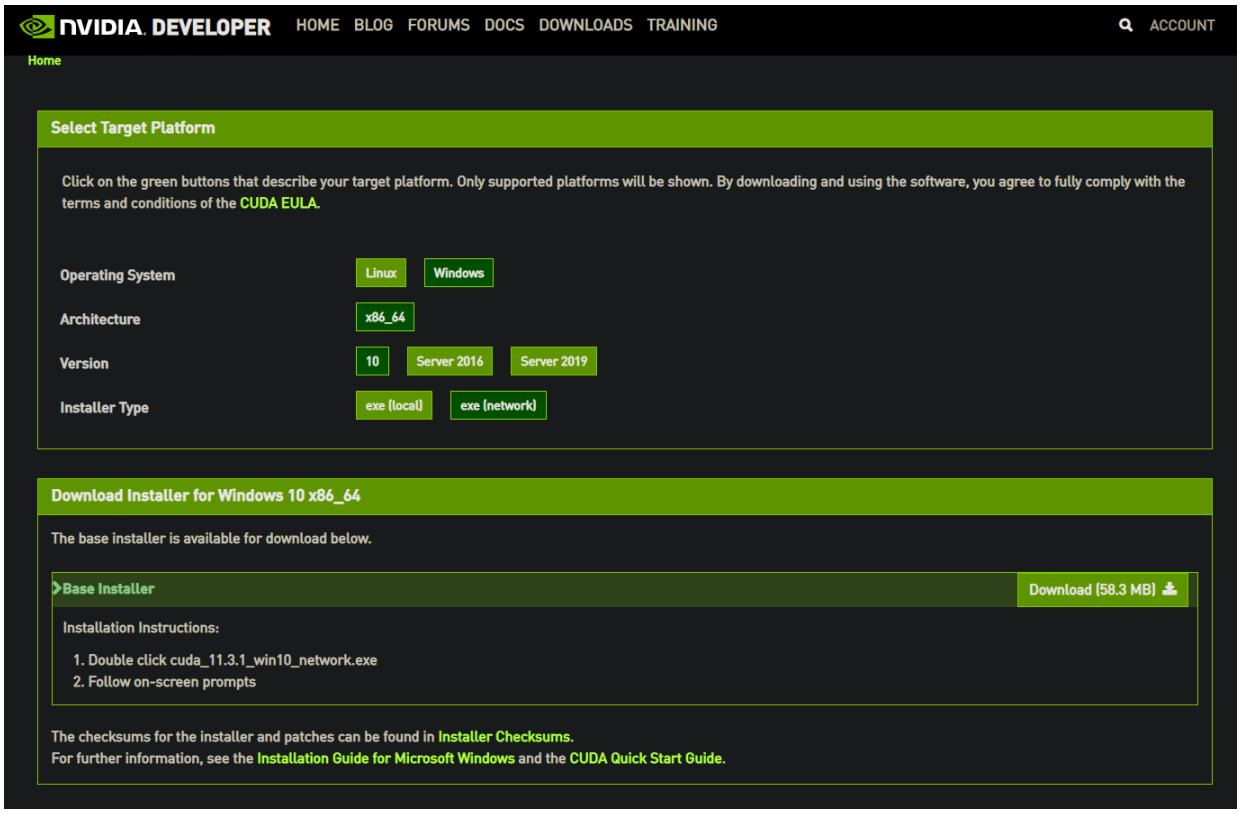


Рисунок 3.6 – Вигляд діалогового вікна платформи CUDA.

Для роботи ми використали Python 3.5. Ось розбитий на пункти основний алгоритм для внесення інформації в зображення на основі PyTorch:

1. Імпорт бібліотек:

```
import torch
import torch.nn as nn
```

```
import torch.optim as optim
```

2. Задання моделі для внесення інформації в зображення:

```
class SteganographyModel(nn.Module):
    def __init__(self):
        super(SteganographyModel, self).__init__()
```

Модель повинна мати шари для обробки зображення та приховування інформації

```
    def forward(self, image, secret_data):
        # Логіка для приховування інформації в зображенні
        # Повертаємо модифіковане зображення
        return modified_image
```

3. Задання даних, які ми хочемо приховати і завантаження зображення для обробки:

```
secret_data = torch.tensor([0.1, 0.2, 0.3])
image = torch.tensor(...)  # Завантажте ваше зображення тут
```

4. Ініціалізація моделі та вибір пристрою (CUDA):

```
model = SteganographyModel()
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
model.to(device)
image = image.to(device)
secret_data = secret_data.to(device)
```

5. Задання функції втрат та оптимізатора:

```
criterion = nn.MSELoss()
optimizer = optim.SGD(model.parameters(), lr=0.01)
```

6. Навчання моделі для збільшення внесеної інформації в зображенні:

```
for epoch in range(num_epochs):
    optimizer.zero_grad()
    output_image = model(image, secret_data)
    loss = criterion(output_image, image)
```

```
        loss.backward()
        optimizer.step()
    7. Збереження модифікованого зображення
    result_image = output_image.cpu().detach().numpy()
    # Зберігайте result_image та використовуйте його за потреби
```

Розглянемо детальніше роботу моделі нейронної мережі, що виконує приховування текстового повідомлення у графічні файли з використанням потужного фреймворку PyTorch.

У командному рядку виконуєм наступну команду:

py setup.py install

Після цього здійсниться встановлення необхідних компонентів для python. Формування датасету:

Для навчання мережі необхідний датасет, що буде складатися із зображень які використовуються в якості контейнера. Для роботи пропонується 2 датасети: div2k:

https://data.vision.ee.ethz.ch/cvl/DIV2K/DIV2K_valid_HR.zip

https://data.vision.ee.ethz.ch/cvl/DIV2K/DIV2K_train_HR.zip

mscoco:

https://images.cocodataset.org/zips/test2017.zip

https://images.cocodataset.org/zips/train2017.zip

Для навчання оригінальної мережі використано перший датасет та 1000 зображень із другого датасету. Датасети складаються із зображень стиснених до візуальної оцінки Добре. Для зменшення навантаження на наші обчислювальні засоби ми можемо здійснити конвертацію обраних датасетів із зменшенням розширення зображень, що також дозволить прискорити навчання мережі. Ви також можете створити свій датасет, або знайти готові у відкритому доступі. [51-52]

Зчитування датасету відбувається за допомогою класів DataLoader та ImageFolder, що успадковані від відповідних класів бібліотеки PyTorch:

```
class ImageFolder(torchvision.datasets.ImageFolder):
    def __init__(self, path, transform, limit=np.inf):
```

```
        super().__init__(path, transform=transform)
        self.limit = limit
    def __len__(self):
        length = super().__len__()
        return min(length, self.limit)

class DataLoader(torch.utils.data.DataLoader):
    def __init__(self, path, transform=None, limit=np.inf, shuffle=True,
                 num_workers=8, batch_size=4, *args, **kwargs):
        if transform is None:
            transform = DEFAULT_TRANSFORM
        super().__init__(
            ImageFolder(path, transform, limit),
            batch_size=batch_size,
            shuffle=shuffle,
            num_workers=num_workers,
            *args,
            **kwargs
        )
```

Клас ImageFolder здійснює зчитування та трансформацію зображень із директорії. Трансформації кожного зображення здійснюється за правилом:

```
DEFAULT_TRANSFORM = transforms.Compose([
    transforms.RandomHorizontalFlip(),
    transforms.RandomCrop(360, pad_if_needed=True),
    transforms.ToTensor(),
    transforms.Normalize(_DEFAULT_MU, _DEFAULT_SIGMA),
])
```

Зображення перевертається випадковим чином, обрізається випадковий квадрат 360x360, який зберігається у вигляді тензора та нормалізується (всі значення приводяться до проміжу від 0 до 1). Надалі зчитані зображення зберігаються у вигляді об’єкта DataLoader, для якого вказано 8 воркерів, що надалі працюватимуть із зображеннями, це здійснено для пришвидшення навчання та обробки даних.

Структура мережі.

Для моделі побудовано три структури мережі: BasicEncoder (рисунок 3.7), ResidualEncoder (рисунок 3.8) та DenseEncoder (рисунок 3.9). Далі робота моделі відрізняється відповідно до трьох структур:

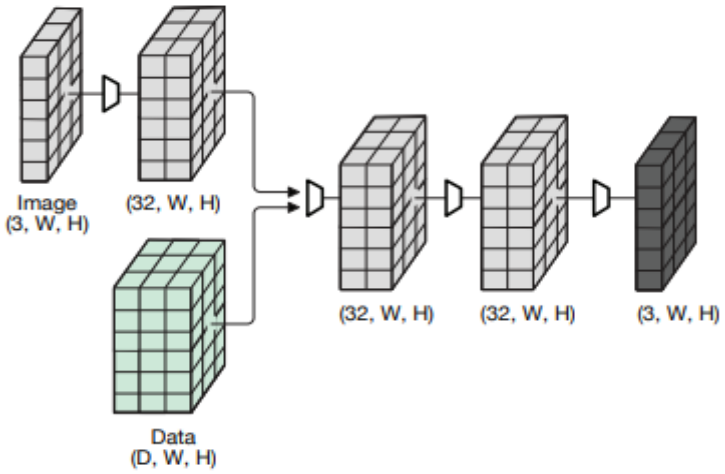


Рисунок 3.7 – Структура BasicEncoder

Клас реалізації BasicEncoder успадковано від модуля nn бібліотеки pytorch знаходиться в Додатку Б

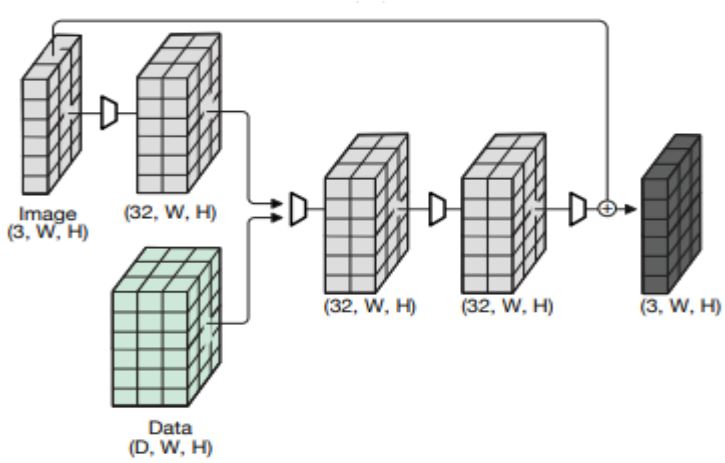


Рисунок 3.8 –Структура ResidualEncoder.

Клас реалізації ResidualEncoder успадковано від BasicEncoder та відрізняється від базового лише додаванням оригінального зображення в кінці шляхом встановлення параметра add_image = True

```
class ResidualEncoder(BasicEncoder):
    """
    The ResidualEncoder module takes an cover image and a data tensor and combines
    them into a steganographic image.
    Input: (N, 3, H, W), (N, D, H, W)
    Output: (N, 3, H, W)
    """
    add_image = True
```

```
def _build_models(self):
    self.features = nn.Sequential(
        self._conv2d(3, self.hidden_size),
        nn.LeakyReLU(inplace=True),
        nn.BatchNorm2d(self.hidden_size),
    )
    self.layers = nn.Sequential(
        self._conv2d(self.hidden_size + self.data_depth, self.hidden_size),
        nn.LeakyReLU(inplace=True),
        nn.BatchNorm2d(self.hidden_size),
        self._conv2d(self.hidden_size, self.hidden_size),
        nn.LeakyReLU(inplace=True),
        nn.BatchNorm2d(self.hidden_size),
        self._conv2d(self.hidden_size, 3),
    )
    return self.features, self.layers
```

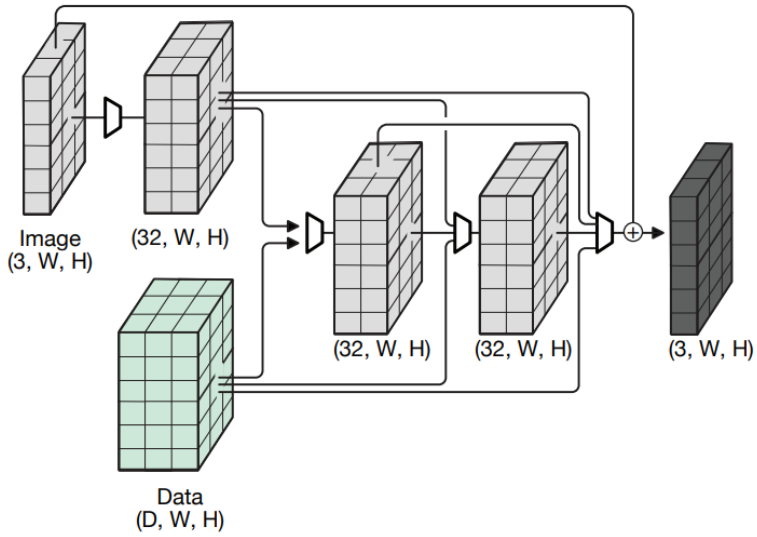


Рисунок 3.9 - Структура DenseEncoder.

Клас реалізації DenseEncoder від BasicEncoder та відрізняється від базового наявністю залежностей наступного стану не тільки від поточного, а й від попереднього стану.

```
Class DenseEncoder(BasicEncoder):
    """
    The DenseEncoder module takes an cover image and a data tensor and combines
    them into a steganographic image.
    Input: (N, 3, H, W), (N, D, H, W)
    Output: (N, 3, H, W)
    """
    add_image = True
    def _build_models(self):
        self.conv1 = nn.Sequential(
            self._conv2d(3, self.hidden_size),
```

```
        nn.LeakyReLU(inplace=True),
        nn.BatchNorm2d(self.hidden_size),
    )
    self.conv2 = nn.Sequential(
        self._conv2d(self.hidden_size + self.data_depth, self.hidden_size),
        nn.LeakyReLU(inplace=True),
        nn.BatchNorm2d(self.hidden_size),
    )
    self.conv3 = nn.Sequential(
        self._conv2d(self.hidden_size * 2 + self.data_depth, self.hidden_size),
        nn.LeakyReLU(inplace=True),
        nn.BatchNorm2d(self.hidden_size),
    )
    self.conv4 = nn.Sequential(
        self._conv2d(self.hidden_size * 3 + self.data_depth, 3)
    )
    return self.conv1, self.conv2, self.conv3, self.conv4
```

Алгоритм Encoder:

Спочатку відбувається згортка зображення для блок для отримання тензора *a* виду (32, W, H). Після цього відбувається конкатенація повідомлення M разом з тензором *a* з подальшою обробкою за допомогою блоку згортки у тензор *b* (D, W, H). D=1 (кількість бітів корисного навантаження для кожного пікселя).

У BasicEncoder відбувається подвійна згортка тензора *b* після чого результат останньої згортки і буде зображенням із вбудованою інформацією S.

У ResidualEncoder також відбувається подвійна згортка тензора *b* після чого результат додається із оригінальним контейнером-зображенням C після чого результат і буде зображенням із вбудованою інформацією S. Додавання оригінального зображення робиться для покращення якості вихідного зображення із прихованою інформацією.

У DenseEncoder використано згорткову нейронну модель DenseNet (Densely Connected Convolutional Network), особливістю якої є використання додаткових зв’язків між згортковими блоками. Такий підхід дозволяє отримати кращого результату з малими датасетами.

У DenseEncoder здійснюється конкатенація *a b* та M, після чого відбувається перша згортка результатом якої є новий тензор *c*. Далі відбувається те ж саме, але конкатенація вже йде з 4 тензерами: *a b c* M В результаті отримуємо тензор *d* , який далі

додається до початкового зображення C. В результаті отримаємо вихідне зображення із вбудованою інформацією S.

Окрім вбудовування, мереж також необхідно навчити робити зворотне перетворення, для чого використано структуру Decoder[53-56]. У Decoder також використано згорткову нейронну модель DenseNet, для досягнення більшої точності декодування прихованої інформації (рисунок 3.10).

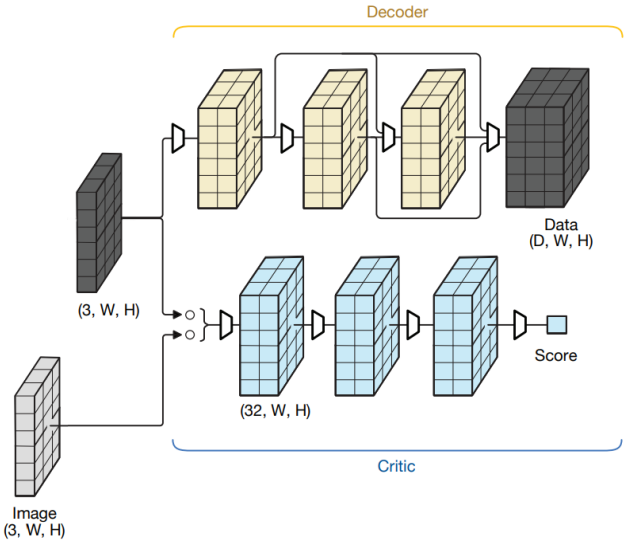


Рисунок 3.10 - Структура Decoder:

Декодер представлено у вигляді класів BasicDecoder та похідного від нього DenseDecoder. Використання BasicDecoder не завжди дозволяє отримати повідомлення, що було приховано після декодування, тому й здійснено його модифікацію з використанням DenseNet. (Додаток Б)

Алгоритм Decoder:

- Отримуємо тензор *a* шляхом згортки S
- Отримуємо тензор *b* шляхом згортки *a*
- Отримуємо тензор *c* шляхом згортки попередньо зконкатенованих *a b*
- Отримуємо тензор із декодованими даними D(S) шляхом шляхом згортки попередньо зконкатенованих *a b c*

Результуючий тензор D(S) міститиме бітовий вектор із копій прихованої інформації, який необхідно в подальшомо розпарсити та знайти приховане повідомлення.

Для отримання більш реальних зображень використовується додаткова мережа критиків. Мережа критиків складається з трьох згорткових блоків, за якими йде згортковий шар з одним виходом. Клас релізації BasicCritic використовує функцію mean для отримання середнього значення тензора і формування оцінки у вигляді єдиного значення від 0 до 1.

```
class BasicCritic(nn.Module):
    """
    The BasicCritic module takes an image and predicts whether it is a cover
    image or a steganographic image (N, 1).
    Input: (N, 3, H, W)
    Output: (N, 1)
    """
    def _conv2d(self, in_channels, out_channels):
        return nn.Conv2d(
            in_channels=in_channels,
            out_channels=out_channels,
            kernel_size=3
        )
    def _build_models(self):
        return nn.Sequential(
            self._conv2d(3, self.hidden_size),
            nn.LeakyReLU(inplace=True),
            nn.BatchNorm2d(self.hidden_size),
            self._conv2d(self.hidden_size, self.hidden_size),
            nn.LeakyReLU(inplace=True),
            nn.BatchNorm2d(self.hidden_size),
            self._conv2d(self.hidden_size, self.hidden_size),
            nn.LeakyReLU(inplace=True),
            nn.BatchNorm2d(self.hidden_size),
            self._conv2d(self.hidden_size, 1)
        )
    def __init__(self, hidden_size):
        super().__init__()
        self.version = '1'
        self.hidden_size = hidden_size
        self._models = self._build_models()
    def upgrade_legacy(self):
        """Transform legacy pretrained models to make them usable with new code versions."""
        # Transform to version 1
        if not hasattr(self, 'version'):
            self._models = self.layers
            self.version = '1'
    def forward(self, x):
```

```
        x = self._models(x)
        x = torch.mean(x.view(x.size(0), -1), dim=1)
        return x
```

Мережу представлено у вигляді модуля, що реалізовано класом Stego:

```
class Stego(object):
    def __init__(self, data_depth, encoder, decoder, critic,
                 cuda=False, verbose=False, log_dir=None, **kwargs):

        self.verbose = verbose
        self.data_depth = data_depth
        kwargs['data_depth'] = data_depth
        self.encoder = self._get_instance(encoder, kwargs)
        self.decoder = self._get_instance(decoder, kwargs)
        self.critic = self._get_instance(critic, kwargs)
        self.set_device(cuda)
        self.critic_optimizer = None
        self.decoder_optimizer = None
        self.fit_metrics = None
        self.history = list()
        self.log_dir = log_dir
        if log_dir:
            os.makedirs(self.log_dir, exist_ok=True)
            self.samples_path = os.path.join(self.log_dir, 'samples')
            os.makedirs(self.samples_path, exist_ok=True)
```

У класі Stego містяться відповідні об’єкти encoder, decoder та critic, що використовуються для навчання або виконання призначених їм дій приховування та відтворення інформації. Окрім того у класі міститься набір метрик, за якими здійснюється оцінка втрат мережі.

Основні методи класу Stego:

set_device – дозволяє обрати GPU (CUDA) або CPU для роботи із мережею, за замовченням обирається CUDA, у разі відсутності CUDA обирається CPU.

```
    def set_device(self, cuda=True):
        """Sets the torch device depending on whether cuda is avaiable or not."""
        if cuda and torch.cuda.is_available():
            self.cuda = True
            self.device = torch.device('cuda')
        else:
            self.cuda = False
            self.device = torch.device('cpu')
        if self.verbose:
            if not cuda:
                print('Using CPU device')
            elif not self.cuda:
                print('CUDA is not available. Defaulting to CPU device')
            else:
                print('Using CUDA device')
```

```
        self.encoder.to(self.device)
        self.decoder.to(self.device)
        self.critic.to(self.device)
```

save load – функції для збереження та завантаження моделі у файл.

```
    def save(self, path):
        torch.save(self, path)
    @classmethod
    def load(cls, architecture=None, path=None, cuda=True, verbose=False):
        if architecture and not path:
            model_name = '{}.steg'.format(architecture)
            pretrained_path = os.path.join(os.path.dirname(__file__), 'pretrained')
            path = os.path.join(pretrained_path, model_name)
        elif (architecture is None and path is None) or (architecture and path):
            raise ValueError(
                'Please provide either an architecture or a path to pretrained model.')
        stego = torch.load(path, map_location='cpu')
        stego.verbose = verbose
        stego.encoder.upgrade_legacy()
        stego.decoder.upgrade_legacy()
        stego.critic.upgrade_legacy()
        stego.set_device(cuda)
        return stego
```

encode та decode – функції для роботи із навченою мережею, попередньо завантаженою мережею. У функції encode відбувається зчитування зображення у масив, з якого формується і нормалізується тензор, після чого за допомогою функції _make_payload формується корисне навантаження (тензор із повідомлення, що необхідно приховати), яке вбудовується до завантаженого зображення завдяки навченій мережі Encoder. Результат формує вихідне зображення, що в подальшому зберігається за вказаним шляхом.

```
    def encode(self, cover, output, text):
        """Encode an image.
        Args:
            cover (str): Path to the image to be used as cover.
            output (str): Path where the generated image will be saved.
            text (str): Message to hide inside the image.
        """
        cover = imread(cover, pilmode='RGB') / 127.5 - 1.0
        cover = torch.FloatTensor(cover).permute(2, 1, 0).unsqueeze(0)
        cover_size = cover.size()
        # _, _, height, width = cover.size()
        payload = self._make_payload(cover_size[3], cover_size[2], self.data_depth, text)
        cover = cover.to(self.device)
        payload = payload.to(self.device)
        generated = self.encoder(cover, payload)[0].clamp(-1.0, 1.0)
        generated = (generated.permute(2, 1, 0).detach().cpu().numpy() + 1.0) * 127.5
        imwrite(output, generated.astype('uint8'))
```

```
        if self.verbose:
            print('Encoding completed.')
```

_make_payload – функція формується корисне навантаження (тензор із повідомлення, що необхідно приховати), яке вбудовується до завантаженого зображення завдяки навченій мережі Encoder. У функції спочатку повідомлення представляється у вигляді бітового вектору, до якого додається 4 нульових байти . Далі цей масив копіюється та формує більший масив розмірності зображення, у яке в подальшому буде вбудовуватися. Отримане навантаження представляється у вигляді тензора та повертається функцією.

```
    def _make_payload(self, width, height, depth, text):
        """
        This takes a piece of text and encodes it into a bit vector. It then
        fills a matrix of size (width, height) with copies of the bit vector.
        """
        message = text_to_bits(text) + [0] * 32
        payload = message
        while len(payload) < width * height * depth:
            payload += message
        payload = payload[:width * height * depth]
        return torch.FloatTensor(payload).view(1, depth, height, width)
```

У функції decode відбувається зчитування зображення у масив, з якого формується і нормалізується тензор, після чого за допомогою навченої мережі Decoder отримуємо «зображення», що є прихованим корисним навантаженням. Отримане корисне навантаження конвертується у масив байт, що розділяється на кандидатів (фрагменти масиву, розділені нульовими байтами), які є копіями прихованого повідомлення, однак зі спотвореннями. Рахується кількість кожного кандидата на приховане повідомлення. Кількість повторень оригінального повідомлення без спотворень буде найбільшою. Обирається кандидат, що зустрічається найчастіше. Це і є приховане повідомлення, що повертається функцією decode.

```
    def decode(self, image):
        if not os.path.exists(image):
            raise ValueError('Unable to read %s.' % image)
        image = imread(image, pilmode='RGB') / 255.0
        image = torch.FloatTensor(image).permute(2, 1, 0).unsqueeze(0)
        image = image.to(self.device)
        image = self.decoder(image).view(-1) > 0
        candidates = Counter()
        bits = image.data.cpu().numpy().tolist()
        for candidate in bits_to_bytearray(bits).split(b'\x00\x00\x00\x00'):
            candidate = bytearray_to_text(bytearray(candidate))
            if candidate:
```

```
                candidates[candidate] += 1
        if len(candidates) == 0:
            raise ValueError('Failed to find message.')
        candidate, count = candidates.most_common(1)[0]
        return candidate
```

fit – функція призначена для навчання мережі. На вхід подаються два об’єкта DataLoader, перший із вибіркою для тренування, другий із вибіркою для перевірки (валідації), а також кількість епох навчання мережі. Мережа навчає спочатку мережу критиків за допомогою функції _fit_critic , після чого здійснюється навчання основної мережі Encoder та Decoder, а також подальша перевірка мережі функцією _validate, після чого формується короткий звіт із розрахунком середніх метрик та збереженням проміжних зображень, що сформувала мережа.

```
    def fit(self, train, validate, epochs=5):
        """Train a new model with the given ImageLoader class."""
        if self.critic_optimizer is None:
            self.critic_optimizer, self.decoder_optimizer = self._get_optimizers()
            self.epochs = 0
        if self.log_dir:
            sample_cover = next(iter(validate))[0]
        #Start training
        total = self.epochs + epochs
        for epoch in range(1, epochs + 1):
            self.epochs += 1
            metrics = {field: list() for field in METRIC_FIELDS}
            if self.verbose:
                print('Epoch {}/{}'.format(self.epochs, total))
            self._fit_critic(train, metrics)
            self._fit_coders(train, metrics)
            self._validate(validate, metrics)
            self.fit_metrics = {k: sum(v) / len(v) for k, v in metrics.items()}
            self.fit_metrics['epoch'] = epoch
            if self.log_dir:
                self.history.append(self.fit_metrics)
                metrics_path = os.path.join(self.log_dir, 'metrics.log')
                with open(metrics_path, 'w') as metrics_file:
                    json.dump(self.history, metrics_file, indent=4)
                save_name = '{}.bpp-{:03f}.p'.format(
                    self.epochs, self.fit_metrics['val.bpp'])
                self.save(os.path.join(self.log_dir, save_name))
                self._generate_samples(self.samples_path, sample_cover, epoch)
            if self.cuda:
                torch.cuda.empty_cache()
            gc.collect()
```

_fit_critic – функція призначена для навчання мережі критиків. На вхід подається об’єкт DataLoader із вибіркою train для тренування, та масив метрик. Для кожного зображення

із train виконується вбудовування за допомогою Encoder випадкового корисного навантаження, після чого отримується значення від критика для оригінального зображення та зображення із вбудованим повідомленням. Знаходиться їх різниця яка і є втратами для критика, для цих втрат здійснюється зворотнє розповсюдження та градієнтний спуск.

```
def _critic(self, image):
    """Evaluate the image using the critic"""
    return torch.mean(self.critic(image))

def _fit_critic(self, train, metrics):
    """Critic process"""
    for cover, _ in tqdm(train, disable=not self.verbose):
        gc.collect()
        cover = cover.to(self.device)
        payload = self._random_data(cover)
        generated = self.encoder(cover, payload)
        cover_score = self._critic(cover)
        generated_score = self._critic(generated)
        self.critic_optimizer.zero_grad()
        (cover_score - generated_score).backward(retain_graph=False)
        self.critic_optimizer.step()
        for p in self.critic.parameters():
            p.data.clamp_(-0.1, 0.1)
        metrics['train.cover_score'].append(cover_score.item())
        metrics['train.generated_score'].append(generated_score.item())
```

_fit_coders– функція призначена для навчання мереж Encoder Decoder. На вхід подається об’єкт DataLoader із вибіркою train для тренування, та масив метрик. Для кожного зображення із train виконується вбудовування за допомогою Encoder випадкового корисного навантаження payload, після чого отриманий від Encoder тензор generated подається на Decoder та формується вихідний тензор decoded. Для generated критик дає свою оцінку, після чого знаходяться загальні втратияк сума метрик:

1) Точність декодування використовуючи перехресну ентропію
2) Подібність між стеганографічним зображенням та вхідним зображенням з використанням середньоквадратичної помилки
3) Значення, яке отримано з мережі критиків

Здійснюється зворотне розповсюдження та градієнтний спуск для мережі Encoder Decoder. Проміжні метрики зберігаються.

```
def _fit_coders(self, train, metrics):
    """Fit the encoder and the decoder on the train images."""
```

```
        for cover, _ in tqdm(train, disable=not self.verbose):
            gc.collect()
            cover = cover.to(self.device)
            generated, payload, decoded = self._encode_decode(cover)
            encoder_mse, decoder_loss, decoder_acc = self._coding_scores(
                cover, generated, payload, decoded)
            generated_score = self._critic(generated)
            self.decoder_optimizer.zero_grad()
            (100.0 * encoder_mse + decoder_loss + generated_score).backward()
            self.decoder_optimizer.step()
            metrics['train.encoder_mse'].append(encoder_mse.item())
            metrics['train.decoder_loss'].append(decoder_loss.item())
            metrics['train.decoder_acc'].append(decoder_acc.item())
    def _encode_decode(self, cover, quantize=False):
        """Encode random data and then decode it.
        Args:cover (image): Image to use as cover.
             quantize (bool): whether to quantize the generated image or not.
        Returns:
            generated (image): Image generated with the encoded message.
            payload (bytes): Random data that has been encoded in the image.
            decoded (bytes): Data decoded from the generated image.
        """
        payload = self._random_data(cover)
        generated = self.encoder(cover, payload)
        if quantize:
            generated = (255.0 * (generated + 1.0) / 2.0).long()
            generated = 2.0 * generated.float() / 255.0 - 1.0
        decoded = self.decoder(generated)
        return generated, payload, decoded
    def _coding_scores(self, cover, generated, payload, decoded):
        encoder_mse = mse_loss(generated, cover)
        decoder_loss = binary_cross_entropy_with_logits(decoded, payload)
        decoder_acc = (decoded >= 0.0).eq(payload >= 0.5).sum().float() / payload.numel()
        return encoder_mse, decoder_loss, decoder_acc
```

_validate – функція призначена для перевірки мережі на даних із validate. На вхід подається об’єкт DataLoader із вибіркою validate для перевірки, та масив метрик. Для кожного зображення із validate виконується алгоритм функції **_fit_coders** без зворотнього розповсюдження та градієнтного спуску, та з додатковим збереженням метрик перевірки.

```
def _validate(self, validate, metrics):
        """Validation process"""
        for cover, _ in tqdm(validate, disable=not self.verbose):
            gc.collect()
            cover = cover.to(self.device)
            generated, payload, decoded = self._encode_decode(cover, quantize=True)
            encoder_mse, decoder_loss, decoder_acc = self._coding_scores(
                cover, generated, payload, decoded)
            generated_score = self._critic(generated)
```

```
            cover_score = self._critic(cover)

            metrics['val.encoder_mse'].append(encoder_mse.item())
            metrics['val.decoder_loss'].append(decoder_loss.item())
            metrics['val.decoder_acc'].append(decoder_acc.item())
            metrics['val.cover_score'].append(cover_score.item())
            metrics['val.generated_score'].append(generated_score.item())
            metrics['val.ssim'].append(ssim(cover, generated).item())
            metrics['val.psnr'].append(10 * torch.log10(4 / encoder_mse).item())
            metrics['val.bpp'].append(self.data_depth * (2 * decoder_acc.item() - 1))
```

Приклад навчання мережі:

```
from modules.models import Stego
from modules.critics import BasicCritic
from modules.decoders import DenseDecoder
from modules.encoders import BasicEncoder, DenseEncoder, ResidualEncoder
from modules.loader import DataLoader
def main():
    torch.manual_seed(42)
    timestamp = str(int(time()))
    parser = argparse.ArgumentParser()
    parser.add_argument('--epochs', default=255, type=int)
    parser.add_argument('--encoder', default="dense", type=str)
    parser.add_argument('--data_depth', default=1, type=int)
    parser.add_argument('--hidden_size', default=32, type=int)
    parser.add_argument('--dataset', default="div2k", type=str)
    parser.add_argument('--output', default=False, type=str)
    args = parser.parse_args()
    train = DataLoader(os.path.join("data", args.dataset, "train"), shuffle=True)
    validation = DataLoader(os.path.join("data", args.dataset, "val"), shuffle=False)
    encoder = {
        "basic": BasicEncoder,
        "residual": ResidualEncoder,
        "dense": DenseEncoder,
    }[args.encoder]
    stego = Stego(
        data_depth=args.data_depth,
        encoder=encoder,
        decoder=DenseDecoder,
        critic=BasicCritic,
        hidden_size=args.hidden_size,
        cuda=True,
        verbose=True,
        log_dir=os.path.join('models', timestamp)
    )
    with open(os.path.join("models", timestamp, "config.json"), "wt") as fout:
        fout.write(json.dumps(args.__dict__, indent=2, default=lambda o: str(o)))
    stego.fit(train, validation, epochs=args.epochs)
    stego.save(os.path.join("models", timestamp, "weights.steg"))
    if args.output:
        stego.save(args.output)
```

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

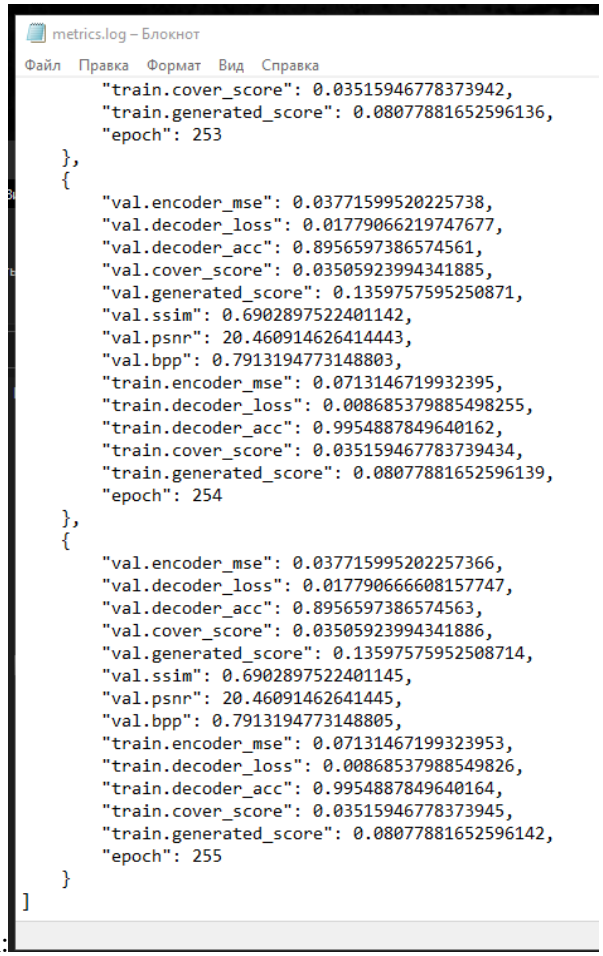
Виконання навчання:

Метрики першої епохи:

metrics.log – Блокнот

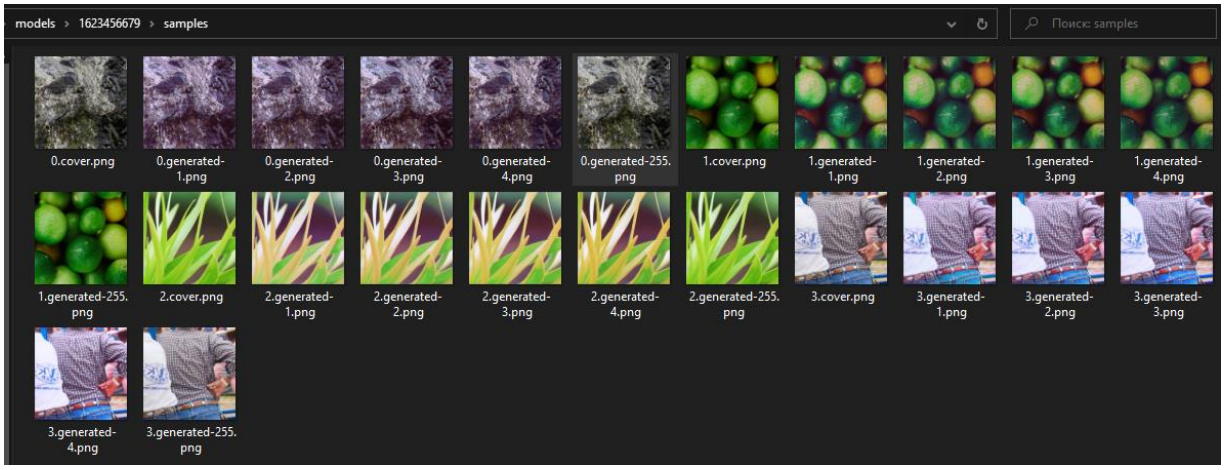
Файл Правка Формат Вид Справка


```
[
    {
        "val.encoder_mse": 0.06974720895290375,
        "val.decoder_loss": 0.591840615272522,
        "val.decoder_acc": 0.6983833312988281,
        "val.cover_score": 0.034787751734256744,
        "val.generated_score": 0.03652960941195488,
        "val.ssim": 0.4259791660308838,
        "val.psnr": 18.691817045211792,
        "val.bpp": 0.39676666259765625,
        "train.encoder_mse": 0.10645478256046773,
        "train.decoder_loss": 0.6427008631825447,
        "train.decoder_acc": 0.647671316564083,
        "train.cover_score": 0.03476091532036662,
        "train.generated_score": 0.03740753879770636,
        "epoch": 1
    }
]
```



Метрики після 255 епох:

Проміжні зображення:



Приклад використання мережі:

```
import argparse
import warnings
from torch.serialization import SourceChangeWarning
from modules.models import Stego
warnings.filterwarnings('ignore', category=SourceChangeWarning)
def _get_stego(args):
    stego_kwargs = {
        'cuda': not args.cpu,
        'verbose': args.verbose
```

```
            }
            if args.path:
                stego_kwargs['path'] = args.path
            else:
                stego_kwargs['architecture'] = args.architecture
            return Stego.load(**stego_kwargs)
        def _encode(args):
            stego = _get_stego(args)
            stego.encode(args.cover, args.output, args.message)
        def _decode(args):
            try:
                stego = _get_stego(args)
                message = stego.decode(args.image)
                if args.verbose:
                    print('Message successfully decoded:')
                print(message)
            except Exception as e:
                print('ERROR: {}'.format(e))

        def _get_parser():
            parent = argparse.ArgumentParser(add_help=False)
            parent.add_argument('-v', '--verbose', action='store_true', help='Be verbose')
            group = parent.add_mutually_exclusive_group()
            group.add_argument('-a', '--architecture', default='dense',
                               choices={'basic', 'dense', 'residual'},
                               help='Model architecture. Use the same one for both encoding and decoding')
            group.add_argument('-p', '--path', help='Load a pretrained model from a given path.')
            parent.add_argument('--cpu', action='store_true',
                                help='Force CPU usage even if CUDA is available')
            parser = argparse.ArgumentParser(description='Command Line Interface')
            subparsers = parser.add_subparsers(title='action', help='Action to perform')
            parser.set_defaults(action=None)

            encode = subparsers.add_parser('encode', parents=[parent],help='Hide a message into a steganographic
image')
            encode.set_defaults(action=_encode)
            encode.add_argument('-o', '--output', default='output.png', help='Path and name to save the output
image')
            encode.add_argument('cover', help='Path to the image to use as cover')
            encode.add_argument('message', help='Message to encode')

            decode = subparsers.add_parser('decode', parents=[parent],help='Read a message from a steganographic
image')
            decode.set_defaults(action=_decode)
            decode.add_argument('image', help='Path to the image with the hidden message')
            return parser
        def main():
            parser = _get_parser()
            args = parser.parse_args()
            if not args.action:
                parser.print_help()
                parser.exit()
```

```
        args.action(args)
    main()
```

```
C:\Users\Vitalii\Desktop\PNGStego>py main.py encode -p weights.steg input.png "password 1234567890" -o test.png

C:\Users\Vitalii\Desktop\PNGStego>py main.py decode -p weights.steg test.png
password 1234567890
```

Отримане секретне повідомлення відповідає вбудованому.

Порівняння зображень:

Оригінал	Із текстовим повідомленням
	
Розміри файлів	
3760кб	4094кб

Використовуючи засоби стеганоаналізу (наприклад StegExpose) було здійснено дослідження зображень із вбудованою інформацією, що було отримано у попередньому завданні для кожної із мереж. Отримані результати представте у вигляді скріні.

Виконано навчання власної нейронної мережі з структурою, що показала найкращі результати. Для навчання використаний сформований у 2 завданні dataset. Навчання можете здійснити використавши функцію із прикладу навчання мережі, однак, оскільки навчання власної мережі займає багато часу.

Використовуючи засоби стеганоаналізу (наприклад StegExpose) було здійснено дослідження зображень із вбудованою інформацією. Отримані результати представлені у вигляді скрінів.

Висновки по розділу

1. Розроблено програмне забезпечення для кодування зображень на основі запропонованих алгоритмів, яке характеризується зручним інтерфейсом користувача і забезпечує достатні швидкісні характеристики.
2. Виконані експериментальні дослідження показали, що застосування двовимірних карт Кохонена для векторного квантування коефіцієнтів перетворення Уолша-Адамара дозволяє підвищити коефіцієнт ущільнення порівняно з методами, які не використовують нейронні мережі. Коефіцієнт ущільнення може знаходитися у межах 3-15.
3. У роботі представлено модель нейронної мережі, що виконує приховування текстового повідомлення у графічні файли. Модель реалізована з використанням потужного фреймворку PyTorch та платформи CUDA. Для навчання оригінальної мережі використано перший датасет та 2000 зображень із другого датасету. Датасети складаються із зображень середньої-високої якості. Тому для зменшення навантаження на наші обчислювальні засоби ми здійснили конвертацію обраних датасетів із зменшенням об’єму зображень, що також дозволить прискорити навчання мережі.

4 ЕКОНОМІЧНА ЧАСТИНА

Для оцінки економічної ефективності створення програми з приховування інформації в зображеннях слід виконати наступні кроки:

1. Оцінка комерційного потенціалу розробки (або проведення технологічного аудиту розробки).

2. Прогнозування витрат на проведення наукових досліджень та впровадження їх результатів.

3. Прогнозування комерційної вигоди від впровадження результатів розробки.

4. Розрахунок ефективності вкладених інвестицій та визначення періоду їх повернення.

4.1 Оцінка комерційного потенціалу розробки

Метою технологічного аудиту є оцінка комерційного потенціалу програми для приховування інформації. Для цього було залучено трьох незалежних експертів. Оцінювання комерційного потенціалу здійснюється за 12-ю критеріями, які представлені в таблиці 4.1.

Таблиця 4.1. – Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Критерій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					

2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних ви-трат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає

Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років

12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництв та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання комерційного потенціалу розробки було зведено у таблицю 4.1.

Таблиця 4.2 –розробки представлено результати оцінювання трьома незалежними експертами – викладачами кафедри МБІС

Критерії	Прізвище, ініціали, посада експерта		
	Експерт 1	Експерт 2	Експерт 3
	Бали, виставлені експертами:		
1	3	3	2
2	1	2	2
3	3	3	2
4	2	3	2
5	3	3	3
6	2	1	1
7	3	3	3
8	4	3	4
9	3	3	2
10	4	3	4
11	4	4	4
12	4	4	4
Сума балів	36	35	33
Середньоарифметична сума балів $\overline{СБ}$	СБ = 34,5		

У межах даної роботи такими експертами є викладачі кафедри МБІС:

Карпінець В. В. (к.т.н., доцент каф. МБІС ВНТУ)- Експерт 1;

Яремчук Ю. Є. (д.т.н., проф. МБІС ВНТУ)- Експерт 2;

Грицак А. В. (доц., викл. каф. МБІС ВНТУ)- Експерт 3;

Відповідно до отриманих результатів, можна зробити висновок, що рівень комерційного потенціалу розробки програми стеганографічного приховування інформації в зображеннях – вище середнього.

При аналізі комерційного потенціалу розробки рекомендується визначити загальну якість продукту. Рівень якості представляє собою кількісну характеристику придатності конкретного продукту для задоволення певного попиту, яку можна порівняти з відповідними базовими показниками при фіксованих умовах використання. Основні параметри інноваційного рішення, згідно якими буде проводитись оцінка рівня якості розробки, абсолютне значення параметру в балах та коефіцієнти вагомості кожного з параметрів наведені в таблиці 4.3.

Таблиця 4.3 – Основні параметри інноваційного рішення

Параметри	Абсолютне значення параметру			Коефіцієнт вагомості параметра
	Краще 8-10	Середнє 4-7	Гірше 0-3	
Зручність інтерфейсу програми (бали)	8			0,2
Ефективність (бали)	8			0,2
Функціонал (бали)		6		0,2
Кросплатформеність (бали)			4	0,1
Надійність (бали)	8			0,2

Виходячи з цього можна визначити абсолютний рівень якості інноваційного рішення за формулою 4.1:

$$К_{я.а.} = \sum_{i=1}^{n} P_{Нi} \cdot a_i, \quad (4.1)$$

де $P_{Нi}$ – числове значення і-го параметру інноваційного рішення;

n – кількість параметрів інноваційного рішення, що прийняті до оцінки;

a_i – коефіцієнт вагомості відповідного параметра.

З формули 4.1 маємо:

$К_{ЯА}=8\cdot0,2+9\cdot0,25+7\cdot0,2+3\cdot0,1+8\cdot0,25=7,55$

Окрім цього необхідно визначити відносний рівень якості. Для його аналізу потрібно порівняти продукт-розробку з певним існуючим конкурентом. Якість інноваційного товару буде порівняно з GifShuffle (таблиця 4.4).

Таблиця 4.4 – Основні параметри інноваційного рішення та програми GifShuffle.

Параметри	Абсолютне значення параметру		Відносний показник якості	Коефіцієнт вагомості параметра
	базовий	інновація		
Зручність інтерфейсу програми (бали)	6	8	1,33	0,2
Ефективність (бали)	8	8	1,1	0,2
Функціонал (бали)	5	7	1,3	0,2
Кросплатформеність (бали)	5	3	0,6	0,1
Надійність (бали)	8	8	1	0,2

Відносні (одиничні) показники якості будь-якого параметра q_i визначаються за формулами:

– збільшення параметра веде до покращення якості виробу:

$$q_i = \frac{P_{Нi}}{P_{Бi}}, \quad (4.2)$$

– збільшення параметра веде до погіршення якості виробу:

$$q_i = \frac{P_{Бi}}{P_{Нi}}, \quad (4.3)$$

$P_{НІ}$, $P_{БІ}$ - числові значення і-го параметра відповідно нового і базового виробів.

Відносний рівень якості інноваційного рішення визначаємо за формулою:

$$К_{я.в.} = \sum_{i=1}^{n} q_i \cdot a_i, \qquad (4.4)$$

$К_{Я.В.}$=1,33·0,2+1,1·0,25+1,4·0,2+0,6·0,1+1·0,25=1,13>1.

Значення відносного показника якості інноваційного продукту свідчить про те, що інноваційний продукт буде кращим за базовий.

4.2 Прогнозування витрат на виконання наукової роботи

Прогнозування витрат на виконання науково-дослідної чи дослідно-конструкторської роботи складається з таких етапів: розрахунок витрат, які безпосередньо стосуються виконавців даного розділу (частини) роботи, розрахунок загальних витрат на виконання всієї роботи та прогнозування загальних витрат на виконання і впровадження результатів дослідної роботи .

Розрахунок витрат, які безпосередньо стосуються виконавців даного розділу (частини) роботи здійснюється за такими статтями та формулами:

Основна заробітна плата кожного із розробників (дослідників) $З_о$, якщо вони працюють в наукових установах бюджетної сфери:

$$З_о = \frac{М}{Т_р} \times t, \text{грн} \qquad (4.5)$$

де М – місячний посадовий оклад конкретного розробника, грн; ТР– число робочих днів в місяці; приблизно ТР (21…23) дні;

t – число робочих днів роботи розробника (дослідника).

$$З_о = \frac{20000}{22} \cdot 44 = 40000 \text{ грн.}$$

Над розробкою програми приховування інформації в зображеннях працює одна людина, отже, загальні витрати на основну заробітну плату будуть складати 40000 грн.

Таблиця 4.5 – Витрати по заробітній платі

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату
Керівник проекту	25000	1136,36	47	53409
Розробник	20 000	909	44	40000
Всього				93409

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт розраховується за формулою:

$$З_р = \sum_{i=1}^{n} C_i \times t_i \quad (4.6)$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – норма часу (трудомісткість) на виконання конкретної операції, годин;

Погодинну тарифну ставку робітника відповідного розряду Сі можна визначити за формулою:

$$C_i = \frac{M_M \times K_i \times K_c}{T_p \times t_{зм}} \quad (4.7)$$

де $М_м$ – мінімальна місячна оплата праці (з 1.09.2022 р. становить 6700 грн.);

$К_і$ – тарифний коефіцієнт робітника даного розряду;

K_c – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об'єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати

$Т_р$ – число робочих днів у місяці ($Т_р$ = 21…23 дні); $Т_{зм}$ – тривалість зміни ($Т_{зм}$ = 8 год).

Отже, погодинна тарифна ставка робітника становить:

$$С_і = \frac{6700 \cdot 1{,}1 \cdot 1{,}65}{22 \cdot 7} = 78{,}96 \text{ грн/год.}$$

$$З_{р1} = 63{,}34 \times 5 = 380{,}02 \text{ грн.}$$

Таблиця 4.6 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Установка електронно-обчислювального обладнання	5	2	1,1	78,96	394,8
Підготовка робочого місця дослідника	3	2	1,1	78,96	236,88
Інсталяція програмного забезпечення	3	5	1,7	122	366
Тестування системи	2	2	1,1	78,96	157,92
				Всього	1 155,6

$$З_р = 1\ 155{,}6 \text{ грн.}$$

Додаткова заробітна плата $З_д$ працівників розраховується як 12% від основної заробітної плати:

$$З_д = (З_о + З_р) \times \frac{Н_д}{100} \quad (4.8)$$

$H_д$ – норма нарахування додаткової заробітної плати. Приймаємо 12%.

$З_о$ – основна заробітна плата розробника;

$З_р$ – витрати на основну заробітну плату робітників;

$$З_д = (93409 + 1\ 155{,}6) \times \frac{12}{100} = 11347{,}6 \text{ (грн.)}$$

Нарахування на заробітну плату $H_{зп}$ дослідників та робітників розраховується як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_д + З_р) \times \frac{\beta}{100} \quad (4.9)$$

$З_о$ – основна заробітна плата розробника;

$З_д$ – додаткова заробітна плата розробника;

$З_р$ – витрати на основну заробітну плату робітників

β – ставка єдиного внеску на загальнообов'язкове державне соціальне страхування – 22%

$$з_н = (93409 + 11347{,}6 + 1\ 155{,}6\ ) \times 0{,}22 = 23\ 300\ (грн.)$$

Розрахунок витрат на комплектуючі.

Витрати на комплектуючі ($К_в$), які використовують при проведенні НДР, відсутні.

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування, яке необхідне для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування та встановлення. В дослідній роботі витрати на спец устаткування відсутні.

Стаття витрат «Програмне забезпечення» належать витрати на придбання необхідного програмного забезпечення та витрати на їх проектування, формування та встановлення.

Балансову вартість програмного забезпечення розраховуємо за формулою:

$$В_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \times С_{прг.i} \times К_i,$$

(4.10)

$Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$С_{прг.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

$К_i$ – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, ($К_i$ = *1,10…1,12);*

k – кількість найменувань програмних засобів.

$$В_{прг1} = 11\ 600 \times 2 \times 1{,}1 = 6\ 646{,}2\ грн.$$

Таблиця 4.7 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
ОС Windows 11	2	11 600	25 520

Прикладний пакет Microsoft Office 2019	2	5500	12 100
		Всього	37 620

$$В_{прг} = 37\ 620 \text{ грн.}$$

Розрахунок амортизації для обладнання, комп'ютерів та приміщень, що використовувались на цьому етапі роботи, проводиться за наступною формулою:

$$А = \frac{Ц \times Т}{12 \times Т_в} \quad (4.11)$$

Ц – загальна балансова вартість обладнання, приміщення тощо, грн.;

Т – фактична тривалість використання, міс;

$Т_о$ – термін використання обладнання, приміщення тощо, роки.

Розробка програмного забезпечення ведеться приблизно 2 місяці.

Для офісного приміщення $А = \frac{70\ 000 \times 2}{12 \times 20} = 583{,}3$ грн.;

Для ноутбука $А = \frac{29\ 000 \times 2}{12 \times 4} = 1\ 208{,}3$ грн.;

Розрахунки зведено до таблиці 4.8:

Таблиця 4.8 – Амортизаційні відрахування

Найменування	Балансова вартість (грн.)	Термін використання (років)	Фактична тривалість використання, (міс.)	Величина амортизаційних відрахувань, (грн.)
Офісне приміщення	70 000	20	2	583,3
Ноутбук HP ENVY 15	25 000	4	2	1 041,6
Ноутбук HP Pavilion 15-eg	29 000	4	2	1208,2
ОС Windows 11	11 600	2	2	966,7
Прикладний пакет Microsoft Office 2019	5500	2	2	458,3
			Всього	4 258

$$А = 4\ 258 \text{ грн.}$$

Витрати на матеріали, що були використані під час виконання даного етапу роботи, розраховуються за формулою:

$$М = \sum_{1}^{п} Н_i \times Ц_i \times К_i - \sum_{j=1}^{n} В_j \times Ц_{в\,j}\ (грн.)$$

(4.12)

$Н_j$ – норма витрат матеріалу j-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу j-го найменування, грн/кг;

$К_j$ – коефіцієнт транспортних витрат, (Кj = 1,1 … 1,15);

$В_j$ – маса відходів j-го найменування, кг;

$Ц_{вj}$ – вартість відходів j-го найменування, грн/кг.

Таблиця 4.9 – Витрати на матеріали

Найменування матеріалів	Ціна за од, грн	Норма витрат, од	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Комп'ютерна мишка	1000 грн.	2	0	0	2200 грн.
Килимок для миші	250 грн.	2	0	0	550 грн.
Папір	240 грн.	1	0	0	264 грн
Офісний набір	150 грн.	2	0	0	300 грн
				Всього	3314 грн.

Витрати на силову електроенергію $В_е$ розраховуються за формулою:

$$В_е = \sum_{i=1}^{п} \frac{W_{yt} \times t_i \times Ц_в \times К_{впі}}{\eta_i}\ (грн.)$$

(4.13)

$Ц_в$ – вартість 1 кВт – год. (на сьогодні для підприємців вартість 7,5 грн./кВт-год.);

W_{yt} – установлена потужність обладнання;

t_i – фактична кількість годин роботи обладнання;

$К_{впі}$ – коефіцієнт використання потужності, $К_{впі} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

Таблиця 4.10 – Витрати на електроенергію

Найменування	Встановлена	Тривалість роботи,	Сума, грн

обладнання	потужність, кВт	год	
Ноутбук ENVY 15	0,6 кВт	378 год.	1665,8 грн.
Ноутбук HP Pavilion 15-eg	0,6 кВт	399 год.	1758,4 грн.
Робоче місце розробника	0.12 кВт	399 год.	351,5 грн.
		Всього	3775,7 грн.

$$B_е = 3775{,}7 \text{ грн.}$$

Інші витрати $I_в$ охоплюють:

- витрати на управління організацією;
- оплату службових відряджень;
- витрати на утримання, ремонт та експлуатацію, основних засобів;
- витрати на опалення, водопостачання, охорону праці тощо.

Інші витрати $B_{ін}$ можна прийняти як 100% від суми основної заробітної плати розробника:

$$I_н = 93409 \times 1 = 93409 \text{ (грн.)}$$

До статті «Накладні витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{нзв} = \left(З_о + З_р\right) \times \frac{Н_{нзв}}{100\%}, \tag{4.14}$$

$Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 100%.

$$B_{нзв} = (93409 + 1\,155{,}6) \times \frac{100}{100} = 94564{,}6 \text{грн}$$

$$В = 93409 + 1\ 155{,}6 + 111347{,}6 + 23300 + 37\ 620 + 4258 + 3\ 314 + 4\ 258 + 93409 + 94564{,}6 = \ 466635{,}8\ (грн.)$$

Проведемо прогнозування загальних витрат ЗВ на виконання та впровадження виконаної наукової роботи. Прогнозування здійснюється за формулою:

$$ЗВ = \frac{В_{заг}}{\beta}, грн. \quad (4.15)$$

β – коефіцієнт, який характеризує етап виконання даної роботи.

Так, якщо розробка знаходиться:

- на стадії науково-дослідних робіт, то $\beta \approx 0{,}1$;
- на стадії технічного проектування, то $\beta \approx 0{,}2$;
- на стадії розробки конструкторської документації, то $\beta \approx 0{,}3$;
- на стадії розробки технології, то $\beta \approx 0{,}4$;
- на стадії розробки дослідного зразка, то $\beta \approx 0{,}5$;
- на стадії розробки промислового зразка, то $\beta \approx 0{,}7$;
- на стадії впровадження, то $\beta \approx 0{,}9$.

$В_{заг}$ – загальна вартість всієї наукової роботи.

$$В = 466635{,}8\ (грн.)$$

$$ЗВ = \frac{466635{,}8}{0.9} = 518484{,}2\ (грн.)$$

Отже, прогноз загальних витрат ЗВ на виконання та впровадження результатів виконаної роботи складає $518484{,}2$ (грн.)

4.3 Прогнозування комерційних ефектів від реалізації результатів розробки

В даному розділі наведено прогнозовану кількісну оцінку очікуваних вигід та можливого прибутку від впровадження результатів наукових досліджень у майбутньому. У сучасних умовах ринкової конкуренції ключовим показником позитивного впливу, що виникає від впровадження розробок, є збільшення чистого прибутку. Оцінка зростання чистого прибутку враховує теперішню вартість грошей.

Підвищення чистого прибутку, що випливає з впровадження розробки,

призведе до збільшення надходження додаткових коштів для підприємства, сприяючи поліпшенню фінансових показників його діяльності. Оцінюється, що тривалість реалізації цієї наукової роботи та впровадження її результатів становить приблизно 7 місяців. Позитивні результати очікуються протягом першого місяця після впровадження.

Цей період впровадження не лише дозволить отримати перші позитивні вигоди, але також створить умови для стійкого зростання прибутковості підприємства у подальшому. Таким чином, реалізація проекту буде відзначатися швидким ефектом і позитивним внеском у фінансовий успіх підприємства.

Проведемо детальне прогнозування позитивних результатів та їх кількісне оцінювання по роках. Обчислення збільшення чистого прибутку підприємства (ΔП) для кожного року, протягом якого очікуються позитивні результати від впровадження розробки, розраховуються за визначеною формулою:

$$\Delta П_i = \sum_1^n (\pm \Delta Ц_о \times N + Ц_о \times \Delta N)_i \times \lambda \times \rho \times (1 - \frac{\vartheta}{100}), \qquad (4.16)$$

±ΔЦо – зміна вартості програмного продукту від впровадження результатів науково-технічної розробки, приймемо 2000,00 грн;

N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки, приймемо 150 користувачів;

$Ц_б$ – вартість програмного продукту у році до впровадження результатів розробки, приймемо 150 000,00 грн;

ΔN – збільшення кількості споживачів продукту, у періоди часу, що аналізуються, від покращення його певних характеристик;

- протягом першого року – збільшення на 35 одиниць;
- протягом другого року – додаткове зростання на 50 одиниць;

- протягом третього року – додаткове зростання на 75 одиниць.

λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2023 році ставка податку на додану вартість складає 20%, а коефіцієнт λ=0,8333;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту. Приймемо ρ=30%;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2023 році ϑ=18%;

Збільшення чистого прибутку ΔП1 протягом першого року складе:

$$\Delta П_1 = (2000 \times 150 + 152000 \times 35) \times 0{,}83 \times 0{,}3 \times \left(1 - \frac{0{,}18}{100}\right) = 1396861 \text{ (грн.)}$$

Обчислимо збільшення чистого прибутку ΔП2 протягом другого року:

$$\Delta П_2 = \left(2000 \times 150 + 152000 \times (35 + 50)\right) \times 0{,}83 \times 0{,}3 \times \left(1 - \frac{0{,}18}{100}\right) = 3285855 \text{ (грн.)}$$

Збільшення чистого прибутку ΔП3 протягом третього року становитиме:

$$\Delta П_3 = \left(2000 \times 150 + 152000 \times (35 + 50 + 75)\right) \times 0{,}83 \times 0{,}3 \times \left(1 - \frac{0{,}18}{100}\right) = 6119345 \text{ (грн.)}$$

Отже, відповідно до обчислень, комерційна вигода від впровадження розробки, як і передбачалося, буде суттєвою і виявиться в зростанні чистого прибутку підприємства.

4.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності

Головними показниками, які визначають прийнятність фінансування конкретного інвестора для наукової розробки, є абсолютна та відносна ефективність вкладених коштів, а також час, необхідний для повернення інвестицій.

На першому етапі здійснюється розрахунок теперішньої вартості інвестицій (*PV*), вкладених у наукову розробку.

Величина початкових інвестицій PV , які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$\text{PV} = k_{інв} \times ЗВ \tag{4.17}$$

$k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=3;

$ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо $518484{,}2$ грн.

$$\text{PV} = 3 \times 518484{,}2 = 1\,555452{,}6$$

На другому етапі проводиться розрахунок очікуваного зростання прибутку (ΔПі), який отримає підприємство (організація) від впровадження результатів наукової розробки. Розрахунок здійснюється для кожного року, розпочинаючи з першого року впровадження. Раніше ми вже розрахували такий приріст прибутку, який складає:

$$\Delta П_1 = 1\,396861 \ (\text{грн.})$$
$$\Delta П_2 = 3\,285855 \ (\text{грн.})$$
$$\Delta П_3 = 6\,119345 \ (\text{грн.})$$

На третьому етапі будується вісь часу, на якій відображаються всі фінансові транзакції, що відбуваються протягом виконання науково-дослідної роботи та впровадження її результатів.

Рисунок 4.1 характеризує рух платежів (інвестицій та додаткових прибутків).

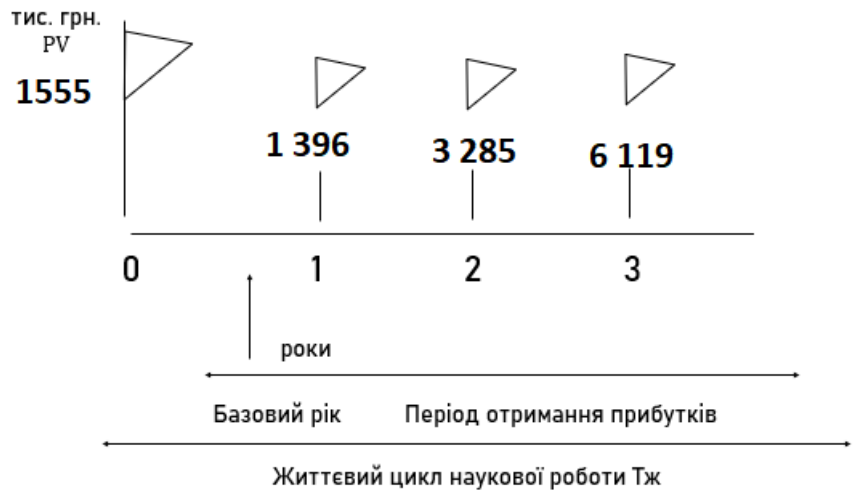


Рисунок 4.1 – Вісь часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів НДДКР

На четвертому етапі проведемо розрахунок абсолютної ефективності вкладених інвестицій ($E_{абс}$) за визначеною формулою:

$$E_{абс} = (ПП - PV), (грн.) \tag{4.18}$$

ПП – приведена вартість всіх чистих прибутків, що їх отримає підприємство (організація) від реалізації результатів наукової розробки, грн.;

PV – теперішня вартість інвестицій, грн.

Приведена вартість всіх чистих прибутків ПП розраховується за формулою:

$$ПП = \sum_{1}^{T} \frac{\Delta П_1}{(1+\tau)^t}, (грн) \tag{4.19}$$

$\Delta П_1$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн,;

Т – період часу, протягом якого виявляються результати впровадженої НДДКР, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні – 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки «0»;

$$ПП = \frac{1\ 396861}{(1+0{,}1)^1} + \frac{3\ 285855}{(1+0{,}1)^2} + \frac{6\ 119345}{(1+0{,}1)^3} = 8583010{,}7\ (грн.)$$

$$E_{абс} = 8583010{,}7 - 1\ 555452{,}6 = \ 7\ 027558{,}1\ (грн.)$$

З огляду на те, що значення абсолютної ефективності вкладених інвестицій ($Е_{абс}$) є позитивним, це свідчить про те, що проведення наукових досліджень для розробки програмного продукту та їх подальше впровадження принесе прибуток. Це підтверджує доцільність проведення досліджень. Проте важливо врахувати, що це лише свідчення ефективності процесу і не гарантує зацікавленість інвестора у фінансуванні даної програми. На п'ятому етапі здійснимо розрахунок відносної (щорічної) ефективності вкладених в наукову розробку інвестицій ($Е_в$) за встановленою формулою.

$$Е_в = \sqrt[Т_ж]{1 + \frac{Е_{абс}}{PV}} - 1 \quad (4.20)$$

$Е_{абс}$ – абсолютна ефективність вкладених інвестицій, грн.;

PV – теперішня вартість інвестицій, грн.;

$Т_ж$ – життєвий цикл наукової розробки, роки.

$$Е_в = \sqrt[3]{1 + \frac{7\,027558{,}1}{1\,555452{,}6}} - 1 = 0{,}77 \text{ або } 77\%$$

Порівняємо значення $Е_в$ з мінімальною (бар'єрною) ставкою дисконтування τmin, яка визначає ту найнижчу рівну дохідність, нижче якої інвестиції не будуть здійснюватися. У загальному висловленні мінімальна (бар'єрна) ставка дисконтування τ_{min} визначається за встановленою формулою:

$$\tau_{min} = d + f \quad (4.21)$$

d – середньозважена ставка за депозитними операціями в комерційних банка;

f – показник, що характеризує ризикованість вкладень; f = 0,3.

d = 0,11.

$$\tau_{min} = 0{,}11 + 0{,}3 = 0{,}41$$

Оскільки $Е_в$ становить 77%, що перевищує мінімальний поріг τ_{min} у 41%, це свідчить про потенційний інтерес інвестора до фінансування даної наукової розробки. На 6-му етапі проведемо розрахунок періоду окупності вкладених інвестицій у реалізацію наукового проекту за встановленою формулою:

$$Т_{ок} = \frac{1}{Е_в}, \text{рік}$$

(4.22)

$$T_{ок} = \frac{1}{0{,}77} = 1{,}29 \text{ (року)}$$

Оскільки період повернення інвестицій у впровадження наукового проекту складає менше трьох років, можна визначити, що фінансування нової розробки є обґрунтованим і виправданим.

Висновки до розділу

Було проведено комплексне оцінювання комерційного потенціалу нового програмного засобу для захищеного консолідованого інформаційного ресурсу системного аналізу безпеки транспортної інфраструктури регіону. Використовуючи експертний підхід, технологічний аудит був проведений з участю трьох незалежних експертів. Результати оцінювання свідчать про високу якість та конкурентоспроможність розробки на ринку.

Витрати на проведення науково-дослідної, дослідно-конструкторської та технологічної роботи оцінені у суму $\mathbf{518484,2}$ грн. Абсолютна ефективність вкладених інвестицій складає $\mathbf{7\,027558,1}$ грн., що свідчить про потенційний прибуток від комерціалізації розробленого програмного продукту.

Розрахунки показують, що річна ефективність вкладених інвестицій у наукову розробку становить 77%, перевищуючи мінімальну бар'єрну ставку дисконтування у 41%. Це свідчить про значний інтерес інвесторів до фінансування цього інноваційного проекту. Термін окупності вкладених інвестицій у реалізацію проекту складає 1,29 року, що підкреслює високу ефективність фінансування нової розробки.

ВИСНОВКИ ПО РОБОТІ

У рамках даної дипломної роботи було показано реалізацію концепції стеганографічних методів, що використовують математичне перетворення зображення (ДКП, вейвлет-перетворення, та ін.) на прикладі перетворення Уолша-Адамара.

Було розроблено програму для стиснення зображень на основі Уолша-Адамара та приховування текстової інформації в цьому ж зображенні за допомогою модифікованих алгоритмів впровадження. Для досягнення цієї мети були виконані наступні завдання:

- проаналізовано методи впровадження інформації в зображення різних форматів;
- удосконалено метод стиснення зображення для подальшого його перетворення;
- удосконалено метод впровадження інформації в зображення;
- розроблено конкурентоспроможну програму на основі розробленого методу;
- протестувано розроблений програмний додаток.

Також були описані алгоритми нанесення стенографії на нерухомі зображення, до кожного з алгоритмів наведено блок-схеми для більш повного донесення викладеного матеріалу у межах розділу. Представлено приклад реалізації моделі нейронної мережі, що виконує приховування текстового повідомлення у графічні файли. Модель реалізована з використанням потужного фреймворку PyTorch та платформи CUDA.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Мельник С.В. Світові тенденції розвитку цифрової стеганографії в контексті завдань забезпечення інформаційної безпеки держави / С.В.Мельник, С.В.Кондакова // Актуальні проблеми управління інформаційною безпекою держави : зб. матер. наук.-практ. конф. – К. : Наук.-вид. відділ НА СБ України, 2010. – С. 134-138.
2. Конахович Г.Ф. Комп’ютерна стегано-графія. Теорія и практика / Г.Ф.Конахович, А.Ю.Пузиренко. – К. : МК-Пресс, 2006. – 288 с.
3. Грибунін В.Г. Цифрова стеганографія / В.Г.Грибунін, И.Н.Оков, И.В.Туринцев. – М. : СОЛОН-Пресс, 2002. – 272 с.
4. Биков С.Ф., Мотуз О.В. Основи стегоанализу.// Захист інформації. Конфидент. – СПб.: 2000, № 3. – С. 38-41.
5. Грибунін В.Г., Оков И.Н., Туринцев И.В. Цифрова стеганографія. – М.: Солон-Пресс, 2002. – 272 с.
6. Елтишева Е.Ю., Фіонов А.Н. Побудова стегосистеми на базі растрових зображень статистики молодших біт // Вісник СибГУТИ. – 2009. № 1. – С. 67-84.
7. Конахович Г.Ф., Пузиренко А.Ю. Компьютерна стеганографія. Теорія и практика. – К.: МК-Пресс, 2006. – 288 с.
8. Жилкін М. Ю. Стегоаналіз графічних даних на основі методов стиснення // Вісник СибГУТИ. – 2008. № 2. – С. 62–66.
9. Кувшинів С.С. Методі та алгоритми сокритія великих об’емів даних на основі стеганографії / Диссертація степені кандидата техничних наук. – Санкт-Петербург. 2010. – 116 с.
10. Бернет С., Пейн С.: Криптографія. Официйне руководство RSA Security – М. «Бином», 2012. – 325 с.
11. Венбо Мао. Сучасна криптографія: теорія и практика = Modern Cryptography: Theory and Practice. – М.: «Вильямс», 2005. – С. 768.

12. Ростовцев А.Г. , Михайлова Н.В. Методи криптоаналізу класичних шифрів. – К.: «Наука», 2012. – С. 142.
13. Саломан А. Криптографія з відкритим ключем. – К.: «Наука», 2013. – 342 с.
14. Серов Р.Е., Гончаров В.В., Основи криптографії – Москва, Горяча лінія – Телеком, 2011. – 443 с.
15. Столлингс В. Криптографія и захист мереж: теорія та практика. М: Вильямс. 2001. Пер. с англ. – 235 с.
16. Чмора А.Л. Сучасна прикладна криптографія. 2-е вид., Стер. – М.: Геліос АРВ, 2012. – 256 с.
17. Шнайер, Брюс. Прикладна криптографія. Протоколі, алгоритми, тексти на мові Сі – М.: ТРИУМФ, 2002 – 816 с.
18. Antonini M., Barlaud M., Mathieu P., Daubechies I. Image Coding Using Wavelet transform / / IEEE Transactions On Image Processing, 1992. – Vol. 1, № 2. – P. 205-220
19. Аграновский А.В. Основи компьютерної стеганографії / А.В. Аграновский, П.Н. Девянин, Р.А. Хади, А.В. Черемушкин. – М: Радио и связь, 2003. – 152 с.
20. В.Ф. Жирков. – Владимир: Владим. гос. ун-та, 2007. – 192 с.
21. Іванов . Стеганографія . ун-та 16-19 мая 2011 г.: в 3-х ч.: ч. 1. – Минск, 2011. – С. 117–120
22. Защелкин К.В. Рішення проблеми классификації блоків контейнера при jpeg-атаці на стеганографичний метод Бенгама-Мемона-Эо-Юнг / К.В.
23. Кустов В.Н. Методи вбудови скритих повідомлень / В.Н. Кустов, А.А. Федчук // Конфидент. – 2000. – № 3. – С.34–37.
24. Li F. JPEG steganalysis with high-dimensional features and bayesian ensemble classifier / F. Li, X. Zhang, B. Chen, G. Feng //IEEE signal processing letters. – 2013. – Vol. 20, № 3. – P. 233–236.
25. Яне Б. Цифрова обробка зображень / Б. Яне. – М.: Техносфера, 2007. – 583с.
26. Паралельно-ієрархічне перетворення як системна модель оптико-електронних засобів штучного інтелекту. Монографія/ Кол. авторів під заг. ред. Кожем'яко В.П. - Вінниця: УНІВЕРСУМ-Вінниця, 2003. - 324 с.

27. Мюррей Дж.Д., Райпер У. Энциклопедия форматов графических файлов/Пер. с англ. - К.: ВНУ, 1997. - 672 с.
28. Міжнародний стандарт JPEG ISO/IEC 10918.
29. Ватолін Д., Ратушняк А., Смирнов М., Юкин В. Методи стиснення даниї Стиснення зображень. - М.: ДИАЛОГ-МИФИ, 2002. - 384 с. :ил.
30. Кожем'яко В.П., Майданюк В.П., Жуков К.М. Аналіз та перспективи розвитку кодування зображень// Вісник ВГП. - 1999. - № 3. - С. 42-48.
31. Круглов В.В., Борисов В.В. Штучні нейронні мережі. Теорія и практика. - М.: Гаряча лінія Щ Телеком, 2001. - 382 с: ил.
32. Каллан Р. Основні концепції нейроних мереж.: Пер. с англ. - М.: «Вильяме», 2001. -286 с: ил.
33. Александров К В. , Горский Е Д. Обробка зображень: Рекурсивний підхід. Л.: Наука, 1985, 192с.
34. Кожемяко В. П. , Натрошвили О. Г. , Тимченко ЛИ. и др. Оптоэлект-ронні параллельні пристрої. Тбилиси, Изд-во Тбилиского университета, 1985, 241с.
35. Прзтт У. Цифроваа обробка зображень. В 2-х книгах. М.: Изд. Мир, 1982.
36. Агаян С. С. , ШЕглазарян К О., Бабаян Е А. Класс ортогональних пертворень// Тез. докл. І Всесоюзной конференции.: нові інформаційні технології. Шнек: 1991.
37. Кожемяко В. Е , Тимченко Л. И. Ієрархія та паралелізм в диалектиці розвитку. В научном журнале изд. "Вища школа" - Киев, 1992.
38. Боянов К., Белчева О. Методи та алгоритми компрессії зображень // Автоматика і інформатика.-1994.-28, №3.-стр.3-14.
39. Голубов Б.И., Ефимів А.В., Скворцов В.А. Ряди і перетворення Уолша: Теорія і примінення. - М.: Наука, 1987. - 344 с
40. Сачанюк-Кавецька Н.В., Кожемяко В.П. «Модель абстрактного КВ-автомата в логіко-часовому середовищі». Оптико-електронні інформаційно-енергетичні технології №2(12) – 2006р.
41. В.П. Кожемяко, Н.В. Сачанюк-Кавецька. Елементи око-процесорної обробки зображень в логіко-часовому середовищі.
42. Майданюк В. П. Методи і засоби комп'ютерних інформаційних технологій. Кодування зображень. - Вінниця: ВДТУ, 2001 - 63 с.

43. Кожемяко В.П. и др. Паралельні обчислювальні метода та засоби пірамідальної обробки інформації. - К.: 1С ДО, 1994. - 256 с.
44. Кулик А.Я, канд. техн. наук; Кириченко О.В. «Використання функцій Уолша-Адамара та Пелі для перетворення даних під час передачі». Оптико-електронні інформаційно-енергетичні технології. № 1(7) 2004 р. Вінницький національний технічний університет.
45. Васюра А.С, Кулик А.Я., Кириченко О. В. „Аналіз методів стиснення інформації". Вісник технологічного університету Поділля. №1 2005 р. Хмельницький державний університет.
46. Тези доповідей. О. Кириченко (ВНТУ). Алгоритм стиснення і відновлення одноколірних напівтонових зображень. КУСС – 2005
47. А.С. Васюра, к.т.н., проф.; Кулик А.Я., к.т.н; Кириченко О. В., «Оцінка ймовірних спектральних характеристик для різних видів дискретних перетворень при стисненні зображень». Оптико-електронні інформаційно-енергетичні технології. № (10) 2006 р.
48. В. П. Кожем'ко, А. С. Васюра, Н. В. Сачанюк-Кавецька, О. В. Кириченко. Принципи ущільнення та перетворення зображення : монографія / за заг. ред.В. П. Кожем'яко. – Вінниця : ВНТУ, 2011. – 242 с.
49. https://arxiv.org/pdf/1901.03892.pdf
50. https://habr.com/ru/post/498168/
51. https://pytorch.org/docs/stable/index.html
52. https://pytorch.org/get-started/locally/
53. https://docs.nvidia.com/cuda/index.html
54. https://data.vision.ee.ethz.ch/cvl/DIV2K/DIV2K_valid_HR.zip
55. https://data.vision.ee.ethz.ch/cvl/DIV2K/DIV2K_train_HR.zip

ДОДАТКИ

Додаток А. Технічне завдання

Вінницький національний технічний університет
Факультет менеджменту та інформаційної безпеки
Кафедра менеджменту та безпеки інформаційних систем

ЗАТВЕРДЖУЮ
Голова секції "Управління інформаційною безпекою" кафедри МБІС
д.т.н., професор
_____ Юрій ЯРЕМЧУК
"20" березня 2023 р.

ТЕХНІЧНЕ ЗАВДАННЯ
до магістерської кваліфікаційної роботи на тему:
Дослідження нейроквантронних систем перетворення для збільшення стеганоконтейнера під час передавання потокової конфіденційної інформації
08-72.МКР.009.00.000.ТЗ

Керівник магістерської кваліфікаційної роботи
к.т.н., доцент
Сачанюк-Кавецька Н.В._____

Вінниця – 2023 р.

1. Найменування та область застосування

Дослідження нейроквантронних систем перетворення для збільшення стеганоконтейнера під час передавання потокової конфіденційної інформації

2.Підстава для розробки

Розробка виконується на основі наказу ректора ВНТУ №203 від 14. 09. 2022 р.

3. Мета та призначення розробки

3.1 Мета розробки: розробити програму, яка покращить швидкість виконання перетворень та приховання інформації у зображенні для передачі потокової конфіденційної інформації.

3.2 Призначення: приховання інформації у зображенні для передачі потокової конфіденційної інформації.

4. Джерела розробки

4.1 Паралельно-ієрархічне перетворення як системна модель оптико-електронних засобів штучного інтелекту. Монографія/ Кол. авторів під заг. ред. Кожем'яко В.П. - Вінниця: УНІВЕРСУМ-Вінниця, 2003. - 324 с.

4.2 Майданюк В. П. Методи і засоби комп'ютерних інформаційних технологій. Кодування зображень. - Вінниця: ВДТУ, 2001 - 63 с.

4.3 В. П. Кожем’ко, А. С. Васюра, Н. В. Сачанюк-Кавецька, О. В. Кириченко. Принципи ущільнення та перетворення зображення : монографія / за заг. ред.В. П. Кожем’яко. – Вінниця : ВНТУ, 2011. – 242 с.

4.4 Кулик А.Я, канд. техн. наук; Кириченко О.В. «Використання функцій Уолша-Адамара та Пелі для перетворення даних під час передачі». Оптико-електронні інформаційно-енергетичні технології. № 1(7) 2004 р. Вінницький національний технічний університет.

5. Вимоги до програми

5.1 Вимоги до функціональних характеристик:

5.1.1 Програмний засіб повинен мати зручний, легкий у використанні інтерфейс користувача;

5.1.2 Реалізація методу не повинна вимагати спеціальних ліцензійних програмних додатків;

5.2 Вимоги до надійності:

5.2.1 Програмний засіб повинен працювати без помилок, у випадку виникнення критичних ситуацій необхідно передбачити виведення відповідних повідомлень;

5.2.2 Бази даних повинні бути налаштовані на автоматичне створення резервних копій;

5.2.3 Програмний засіб повинен виконувати свої функції.

5.3 Вимоги до складу і параметрів технічних засобів:

– процесор – Pentium 1500 МГц і подібні до них;

– оперативна пам’ять – не менше 1024 Mb;

– середовище функціонування – операційна система сімейство Windows;

– вимоги до техніки безпеки при роботі з програмою повинні відповідати існуючим вимогам та стандартам з техніки безпеки при користуванні комп’ютерною технікою.

6. Вимоги до програмної документації

6.1 Обов'язкова поетапна інструкція для майбутніх користувачів, наведена у пункті 3.4

7. Вимоги до технічного захисту інформації

7.1 Необхідно забезпечити захист розроблюваного програмного засобу від несанкціонованого використання.

7.2 Неможливість отримання доступу незареєстрованих користувачів до інформаційних ресурсів.

8. Техніко-економічні показники

8.1 Цінність результатів використання даного проекту повинна перевищувати витрати на його реалізацію.

8.2 Має бути реалізований таким чином, щоб підходити для використання широкого загалу.

9. Стадії та етапи розробки

№	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи		Примітка
1	Визначення напрямку магістерської роботи, формулювання теми	20.09.2023	31.09.2023	
2	Аналіз предметної області обраної теми	01.10.2023	15.10.2023	
3	Розробка роботи	16.10.2023	26.10.2023	
4	Написання магістерської роботи на основі розробленої теми	27.10.2023	15.11.2023	
5	Передзахист магістерської кваліфікаційної роботи	16.11.2023	24.11.2023	
6	Виправлення, уточнення, корегування магістерської кваліфікаційної роботи	27.11.2023	04.12.2023	
7	Захист магістерської кваліфікаційної роботи	11.12.2023	17.12.2023	

10. Порядок контролю та прийому

10.1 До приймання магістерської кваліфікаційної роботи надається:

– ПЗ до магістерської кваліфікаційної роботи;
– програмний додаток;
– презентація;
– відзив керівника роботи;
– відзив опонента

Технічне завдання до виконання прийняв _____ Кириченко О.В.

ДОДАТОК Б

ТЕКСТ МОДУЛІВ ПРОГРАМИ КОДУВАННЯ НА ОСНОВІ ПЕРЕТВОРЕННЯ УОЛША-АДАМАРА

Перетворення Уолша-Адамара

```
unit UAD;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Menus, ExtCtrls, ExtDlgs;

const
      ni_i=40;
      njj =320;
      Matr8_8   : array [1..8,1..8] of integer =
                    ((8 ,8 ,8 ,8 ,8 ,8 ,8 ,16),
                    (8 ,8 ,8 ,8 ,8 ,8 ,16,16),
                    (8 ,8 ,8 ,16,16,16,16,16),
                    (8 ,8 ,16,8 ,16,16,16,16),
                    (8 ,8 ,16,16,16,16,16,16),
                    (8 ,8 ,16,16,16,16,16,32),
                    (8 ,16,16,16,16,16,32,32),
                    (16,16,16,16,16,32,32,32));

                    {((8 ,8 ,8 ,8 ,8 ,8 ,8 ,8 ),
                    (8 ,8 ,8 ,8 ,8 ,8 ,8 ,8 ),
                    (8 ,8 ,8 ,8 ,8 ,8 ,8 ,8 ),
                    (8 ,8 ,8 ,8 ,8 ,8 ,8 ,8 ),
                    (8 ,8 ,8 ,8 ,8 ,8 ,8 ,8 ),
                    (8 ,8 ,8 ,8 ,8 ,8 ,8 ,8 ),
                    (8 ,8 ,8 ,8 ,8 ,8 ,8 ,8 ),
                    (8 ,8 ,8 ,8 ,8 ,8 ,8 ,8 ));}
type
    din1       = array[1..ni_i,1..njj] of integer;
    din1ptr    = ^din1;
    din2       = array[1..ni_i,1..njj] of byte;
    din2ptr    = ^din2;
    din3       = array[1..ni_i,1..njj] of shortint;
    din3ptr    = ^din3;
    buf = array [0..199, 0..319] of byte;
Type
  Dlina=string[39];
var
  Regime        : byte;
  Ch            : char;
   i,j,ii,n,k,s2         : integer;
   R                     : array [0..3,0..8] of integer;
   X1,KK1,XX1,KKK1,KKK   : din1ptr;
   H1,KK2                : din2ptr;
   DDD                   : din2ptr;
   ff1,ff2               : file;
   f1,ff                 : file;
   first_i,first_j       : integer;
   last_i,last_j         : integer;
   pj8,pi8,ch1,ch2,ch3,ch4,ch5   : integer;
   p1,p2,p3,pp           : byte;
```

```
   p4,p0,tt             : integer;
   stt,name1,st1        : string;
    buf1: buf;
type
  TTextViewer = class(TForm)
    MainMenu1: TMainMenu;
    File1: TMenuItem;
    Open1: TMenuItem;
    Close1: TMenuItem;
    N1: TMenuItem;
    Exit1: TMenuItem;
    SaveDialog1: TSaveDialog;
    decod1: TMenuItem;
    Image1: TImage;
    ImageChange1: TMenuItem;
    Image2: TImage;
    SavePictureDialog1: TSavePictureDialog;
    OpenDialog1: TOpenDialog;
    Label1: TLabel;
    Label2: TLabel;
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure Open1Click(Sender: TObject);
    procedure decod1Click(Sender: TObject);
    procedure Close1Click(Sender: TObject);
    procedure Exit1Click(Sender: TObject);
    procedure ImageChange1Click(Sender: TObject);
  private
    { Private declarations }
    Filename: string;
  public
    { Public declarations }
    Procedure Open (Const AFilename: String);
  end;

var
  TextViewer: TTextViewer;

implementation
uses viewmain;
{$R *.DFM}
Procedure TTextViewer.Open (const AFilename: string);
Begin
Filename:=AFilename;
Image1.Picture.LoadFromFile (Filename);
Caption:=Filename;
end;

procedure TTextViewer.FormClose(Sender: TObject; var Action: TCloseAction);
begin
Action:=caFree;
end;
Procedure Ex_var;
var
   s: integer;
begin

R[1,0]:=R[0,0]+R[0,4];
R[1,1]:=R[0,1]+R[0,5];
R[1,2]:=R[0,2]+R[0,6];
R[1,3]:=R[0,3]+R[0,7];
```

```
R[1,4]:=R[0,0]-R[0,4];
R[1,5]:=R[0,1]-R[0,5];
R[1,6]:=R[0,2]-R[0,6];
R[1,7]:=R[0,3]-R[0,7];

R[2,0]:=R[1,0]+R[1,2];
R[2,1]:=R[1,1]+R[1,3];
R[2,2]:=R[1,0]-R[1,2];
R[2,3]:=R[1,1]-R[1,3];

R[2,4]:=R[1,4]+R[1,6];
R[2,5]:=R[1,5]+R[1,7];
R[2,6]:=R[1,4]-R[1,6];
R[2,7]:=R[1,5]-R[1,7];


R[3,0]:=R[2,0]+R[2,1];
R[3,1]:=R[2,0]-R[2,1];

R[3,2]:=R[2,2]+R[2,3];
R[3,3]:=R[2,2]-R[2,3];

R[3,4]:=R[2,4]+R[2,5];
R[3,5]:=R[2,4]-R[2,5];

R[3,6]:=R[2,6]+R[2,7];
R[3,7]:=R[2,6]-R[2,7];
end;
Procedure First_pr40_320;
Begin
  last_i:=ni_i-(ni_i mod 8);
  last_j:=njj-(njj mod 8);
  first_i:=1;

Repeat

  first_j:=1;
 repeat
   ch3:=first_i;
   ii:=first_i;
   ch2:=first_j-1;
  while ii<=(ch3+7) do
   begin
     ch1:=0;
    for j:=first_j to (first_j+7) do
     begin
       R[0,ch1]:=H1^[ii,j];
       inc(ch1);
     end;
      Ex_var;

      X1^[ii,ch2+1]:=R[3,0];
      X1^[ii,ch2+2]:=R[3,4];
      X1^[ii,ch2+3]:=R[3,6];
      X1^[ii,ch2+4]:=R[3,2];
      X1^[ii,ch2+5]:=R[3,3];
      X1^[ii,ch2+6]:=R[3,7];
      X1^[ii,ch2+7]:=R[3,5];
      X1^[ii,ch2+8]:=R[3,1];
```

```
      ii:=ii+1;
    end;
    inc(first_j,8);
  until (first_j>last_j);
     inc(first_i,8);
Until (first_i>last_i);
  first_j:=1;
  ii:=1;

Repeat
  first_i:=1;
 repeat
    ch3:=first_j;
    ii:=first_j;
    ch2:=first_i-1;
     ch5:=1;
  while ii<=(ch3+7) do
    begin
       ch1:=0;
      for j:=first_i to (first_i+7) do
       begin
       R[0,ch1]:=X1^[j,ii];
        inc(ch1);
       end;
         Ex_var;
      KK1^[ch2+1,ii]:=round((R[3,0]/8)/Matr8_8[1,ch5]);
      KK1^[ch2+2,ii]:=round((R[3,4]/8)/Matr8_8[2,ch5]);
      KK1^[ch2+3,ii]:=round((R[3,6]/8)/Matr8_8[3,ch5]);
      KK1^[ch2+4,ii]:=round((R[3,2]/8)/Matr8_8[4,ch5]);
      KK1^[ch2+5,ii]:=round((R[3,3]/8)/Matr8_8[5,ch5]);
      KK1^[ch2+6,ii]:=round((R[3,7]/8)/Matr8_8[6,ch5]);
      KK1^[ch2+7,ii]:=round((R[3,5]/8)/Matr8_8[7,ch5]);
      KK1^[ch2+8,ii]:=round((R[3,1]/8)/Matr8_8[8,ch5]);
       inc(ch5);
       ii:=ii+1;
   end;
      inc(first_i,8);
  until (first_i>last_i);
    inc(first_j,8);
Until (first_j>last_j);
End;


Procedure Second_obr40_320;
Begin
  last_i:=ni_i-(ni_i mod 8);
  last_j:=njj-(njj mod 8);
  first_i:=1;

Repeat
  first_j:=1;
 repeat
   ch3:=first_i;
   ii:=first_i;
   ch2:=first_j-1;
   ch5:=1;
  while ii<=(ch3+7) do
   begin
      ch1:=0;
```

```
    for j:=first_j to (first_j+7) do
     begin
       R[0,ch1]:=KKK^[ii,j]*Matr8_8[ch5,ch1+1];
       inc(ch1);
     end;
      Ex_var;

      XX1^[ii,ch2+1]:=R[3,0];
      XX1^[ii,ch2+2]:=R[3,4];
      XX1^[ii,ch2+3]:=R[3,6];
      XX1^[ii,ch2+4]:=R[3,2];
      XX1^[ii,ch2+5]:=R[3,3];
      XX1^[ii,ch2+6]:=R[3,7];
      XX1^[ii,ch2+7]:=R[3,5];
      XX1^[ii,ch2+8]:=R[3,1];
      inc(ch5);
      ii:=ii+1;
    end;
    inc(first_j,8);
  until (first_j>last_j);
     inc(first_i,8);
Until (first_i>last_i);
   first_j:=1;
   ii:=1;

Repeat
   first_i:=1;
 repeat
    ch3:=first_j;
    ii:=first_j;
    ch2:=first_i-1;
   while ii<=(ch3+7) do
    begin
       ch1:=0;
      for j:=first_i to (first_i+7) do
       begin
       R[0,ch1]:=XX1^[j,ii];
        inc(ch1);
       end;
         Ex_var;
      KKK1^[ch2+1,ii]:=round(R[3,0]/8);
      KKK1^[ch2+2,ii]:=round(R[3,4]/8);
      KKK1^[ch2+3,ii]:=round(R[3,6]/8);
      KKK1^[ch2+4,ii]:=round(R[3,2]/8);
      KKK1^[ch2+5,ii]:=round(R[3,3]/8);
      KKK1^[ch2+6,ii]:=round(R[3,7]/8);
      KKK1^[ch2+7,ii]:=round(R[3,5]/8);
      KKK1^[ch2+8,ii]:=round(R[3,1]/8);
       ii:=ii+1;
   end;
      inc(first_i,8);
  until (first_i>last_i);
     inc(first_j,8);
Until (first_j>last_j);

End;


Procedure Uolsh_Adamar;
```

```
BEGIN


{-------------Прямое преобразование----------------------}
  {Assign(ff1,'C:\tp7\bin\DATA1.DAT');}
  AssignFile(ff1,'d:\UAD\DATA1.DAT');
  Reset(ff1);
  AssignFile(f1,'d:\UAD\DAT2.DAT');
  Rewrite(f1);

  ch4:=1;


Repeat
      Getmem(DDD,sizeof(din2));
      Getmem(H1,sizeof(din2));
      Getmem(X1,sizeof(din1));
      Getmem(KK1,sizeof(din1));

     BlockRead(ff1,H1^,100);

     First_pr40_320;
   for i:=1 to ni_i do begin
     for j:=1 to njj do
       begin
         DDD^[i,j]:=KK1^[i,j];
       end;
       end;
     BlockWrite(f1,DDD^,100);
     Freemem(DDD,sizeof(din2));
     Freemem(H1,sizeof(din2));
     Freemem(X1,sizeof(din1));
     Freemem(KK1,sizeof(din1));
     inc(ch4);
Until  (ch4>(ni_i div 8));

   CloseFile(ff1);
   CloseFile(f1);

{-------------Обратное преобразование-----------}


  AssignFile(f1,'d:\UAD\DAT2.DAT');
  Reset(f1);
  AssignFile(ff2,'d:\UAD\DATA3.DAT');
  Rewrite(ff2);
  ch4:=1;


Repeat
      Getmem(DDD,sizeof(din2));
      Getmem(KKK,sizeof(din1));
      Getmem(KK2,sizeof(din2));
      Getmem(XX1,sizeof(din1));
      Getmem(KKK1,sizeof(din1));
      BlockRead(f1,ddd^,100);
pi8:=1;
pj8:=1;
   for i:=1 to ni_i do begin
     for j:=1 to njj do
```

```
        begin
if (j=pj8) and (i=pi8) then begin
        KKK^[i,j]:=DDD^[i,j];
       pj8:=pj8+8;
       end else begin
if DDD^[i,j]>128 then KKK^[i,j]:=-(not(ddd^[i,j])+1)
else kkk^[i,j]:=ddd^[i,j];
       end;
        end;
         pj8:=1;
if i=pi8 then pi8:=pi8+8;
        end;
     Second_obr40_320;
   for i:=1 to ni_i do begin
     for j:=1 to njj do
        begin
        if KKK1^[i,j]<0 then
          KKK1^[i,j]:=0;
        if KKK1^[i,j]>255 then
          KKK1^[i,j]:=255;
        KK2^[i,j]:=Lo(KKK1^[i,j]);
        end;
        end;
        BlockWrite(ff2,KK2^,100);
    Freemem(DDD,sizeof(din2));
     Freemem(KKK,sizeof(din1));
     Freemem(KK2,sizeof(din2));
     Freemem(XX1,sizeof(din1));
     Freemem(KKK1,sizeof(din1));
     inc(ch4);
Until  (ch4>(ni_i div 8));

   Closefile(f1);
   Closefile(ff2);

END;


procedure TTextViewer.Open1Click(Sender: TObject);
var
i,j:integer;
k: integer;
str:string;
filename: Pchar;
begin
if SaveDialog1.Execute then begin

for j:=0 to 319 do
for i:=0 to 199 do

buf1[i,j]:=Image1.Canvas.Pixels[j,i];
  AssignFile(f1,'d:\UAD\DATA1.DAT');
  Rewrite(f1);
  BlockWrite(f1,buf1,500);
   Closefile(f1);


      Uolsh_Adamar;

   AssignFile(f1,'d:\UAD\DATA3.DAT');
```

```
  Reset(f1);
  BlockRead(f1,buf1,500);
   Closefile(f1);

   for j:=0 to 319 do
      for i:=0 to 199 do     begin
     k:= buf1[i,j];
     k:=k shl 8;
     k:=k or buf1[i,j];
     k:=k shl 8;
     k:=k or buf1[i,j];
   Image2.Canvas.Pixels[j,i]:=k{buf1[i,j]};


    end;

    AssignFile(f1, 'd:\UAD\dat2.dat');
    Rename(f1,'d:\UAD\ComprUAD');
    WinExec('d:\UAD\arj.exe a -e d:\UAD\ComprUAD.arj d:\UAD\ComprUAD',SW_HIDE);

    Sleep (2000);
     AssignFile(f1, 'd:\UAD\ComprUAD');
       Erase(f1);

     {AssignFile(f1, SaveDialog1.FileName);
     Erase (f1);
     CloseFile (f1);

     AssignFile(f1, 'd:\UAD\ComprUAD.arj');
     Rename(f1,SaveDialog1.FileName);}

     CopyFile ('d:\UAD\ComprUAD.arj', PChar (SaveDialog1.FileName), FALSE);
end;
end;

procedure TTextViewer.decod1Click(Sender: TObject);
var
k: integer;
str:string;
begin
if OpenDialog1.Execute then begin

{-------------Обратное преобразование-----------}
  str:=OpenDialog1.FileName+#0;
  CopyFile(Pchar(str),'d:\UAD\dat2.arj',false);


 WinExec('d:\UAD\arj.exe e -y d:\UAD\dat2.arj d:\UAD\',SW_HIDE);
 Sleep (1000);
 AssignFile(f1, 'd:\UAD\ComprUAD');
    Rename(f1,'d:\UAD\dat2.dat');

   for j:=0 to 319 do
      for i:=0 to 199 do
   Image2.Canvas.Pixels[j,i]:=0{buf1[i,j]};

  AssignFile(f1,'d:\UAD\DAT2.DAT');
  Reset(f1);
  AssignFile(ff2,'d:\UAD\DATA3.DAT');
  Rewrite(ff2);
```

```
  ch4:=1;


Repeat
      Getmem(DDD,sizeof(din2));
      Getmem(KKK,sizeof(din1));
      Getmem(KK2,sizeof(din2));
      Getmem(XX1,sizeof(din1));
      Getmem(KKK1,sizeof(din1));
      BlockRead(f1,ddd^,100);
pi8:=1;
pj8:=1;
   for i:=1 to ni_i do begin
     for j:=1 to njj do
       begin
if (j=pj8) and (i=pi8) then begin
        KKK^[i,j]:=DDD^[i,j];
       pj8:=pj8+8;
       end else begin
if DDD^[i,j]>128 then KKK^[i,j]:=-(not(ddd^[i,j])+1)
else kkk^[i,j]:=ddd^[i,j];
       end;
        end;
         pj8:=1;
if i=pi8 then pi8:=pi8+8;
        end;
     Second_obr40_320;
   for i:=1 to ni_i do begin
     for j:=1 to njj do
        begin
        if KKK1^[i,j]<0 then
          KKK1^[i,j]:=0;
        if KKK1^[i,j]>255 then
          KKK1^[i,j]:=255;
        KK2^[i,j]:=Lo(KKK1^[i,j]);
        end;
        end;
        BlockWrite(ff2,KK2^,100);
    Freemem(DDD,sizeof(din2));
     Freemem(KKK,sizeof(din1));
     Freemem(KK2,sizeof(din2));
     Freemem(XX1,sizeof(din1));
     Freemem(KKK1,sizeof(din1));
     inc(ch4);
Until  (ch4>(ni_i div 8));

   Closefile(f1);
   Closefile(ff2);
   Erase (f1);

   AssignFile(f1,'d:\UAD\DATA3.DAT');
  Reset(f1);
  BlockRead(f1,buf1,500);
   Closefile(f1);

   for j:=0 to 319 do
      for i:=0 to 199 do     begin
     k:= buf1[i,j];
     k:=k shl 8;
     k:=k or buf1[i,j];
```

```
      k:=k shl 8;
      k:=k or buf1[i,j];
    Image2.Canvas.Pixels[j,i]:=k{buf1[i,j]};
       end;

     end;
end;

procedure TTextViewer.Close1Click(Sender: TObject);
begin
close;
end;

procedure TTextViewer.Exit1Click(Sender: TObject);
begin
close;
Application.Terminate;


end;

procedure TTextViewer.ImageChange1Click(Sender: TObject);
begin
if SavePictureDialog1.Execute then begin
Image2.Picture.SaveToFile (SavePictureDialog1.FileName);

end;
end;

end.
```

Лістинг програми по внесенню інформації в зображення на основі нейронної мережі

Клас реалізації BasicEncoder успадковано від модуля nn бібліотеки pytorch

```
class BasicEncoder(nn.Module):
    """
    The BasicEncoder module takes an cover image and a data tensor and combines
    them into a steganographic image.
    Input: (N, 3, H, W), (N, D, H, W)
    Output: (N, 3, H, W)
    """
    add_image = False
    def _conv2d(self, in_channels, out_channels):
        return nn.Conv2d(
            in_channels=in_channels,
            out_channels=out_channels,
            kernel_size=3,
            padding=1
        )
    def _build_models(self):
        self.features = nn.Sequential(
            self._conv2d(3, self.hidden_size),
            nn.LeakyReLU(inplace=True),
            nn.BatchNorm2d(self.hidden_size),
        )
        self.layers = nn.Sequential(
            self._conv2d(self.hidden_size + self.data_depth, self.hidden_size),
            nn.LeakyReLU(inplace=True),
            nn.BatchNorm2d(self.hidden_size),
            self._conv2d(self.hidden_size, self.hidden_size),
            nn.LeakyReLU(inplace=True),
            nn.BatchNorm2d(self.hidden_size),
            self._conv2d(self.hidden_size, 3),
            nn.Tanh(),
        )
        return self.features, self.layers
    def __init__(self, data_depth, hidden_size):
        super().__init__()
```

```
            self.version = '1'
            self.data_depth = data_depth
            self.hidden_size = hidden_size
            self._models = self._build_models()
        def upgrade_legacy(self):
            """Transform legacy pretrained models to make them usable with new code versions."""
            # Transform to version 1
            if not hasattr(self, 'version'):
                self.version = '1'
        def forward(self, image, data):
            x = self._models[0](image)
            x_list = [x]
            for layer in self._models[1:]:
                x = layer(torch.cat(x_list + [data], dim=1))
                x_list.append(x)
            if self.add_image:
                x = image + x
            return x
```

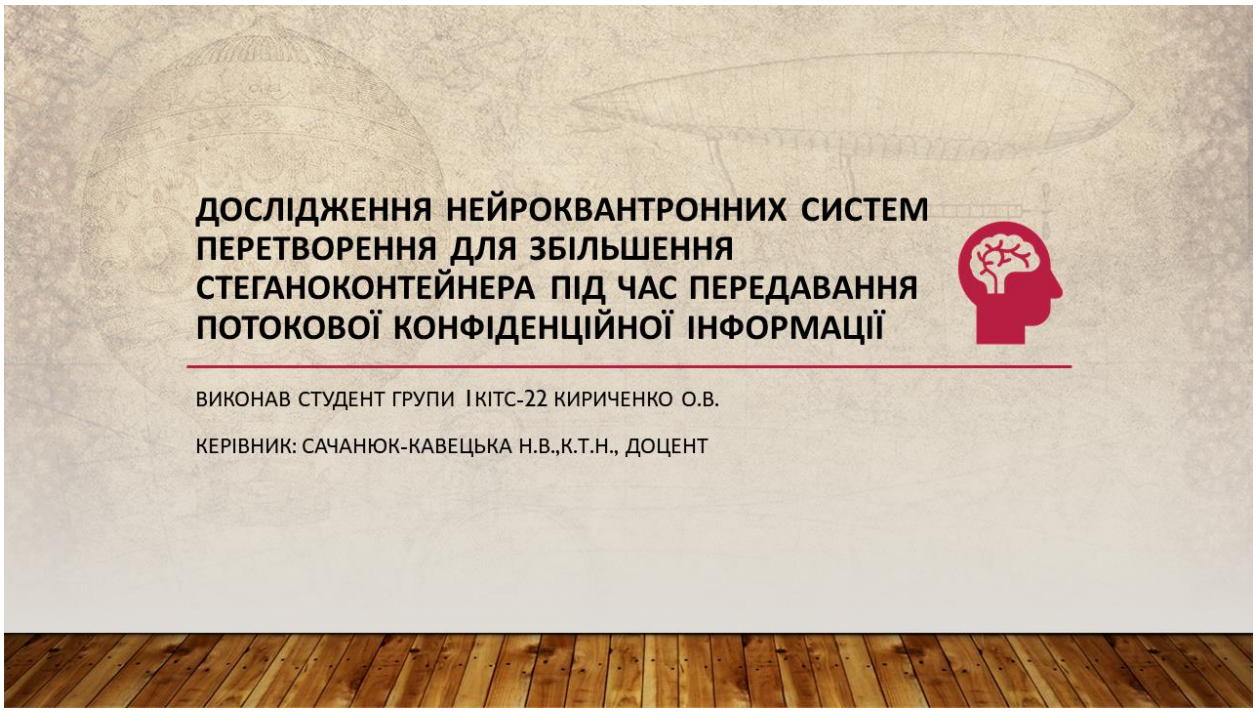
модифікацію з використанням DenseNet.

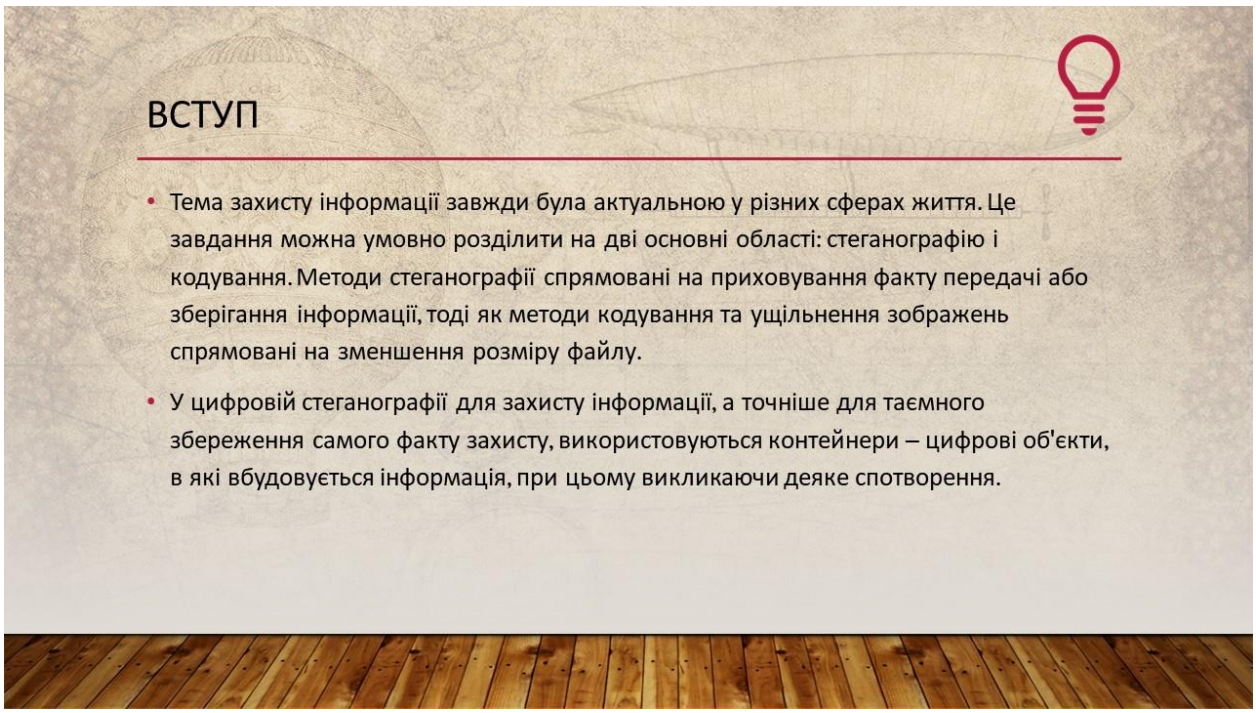
```
    class BasicDecoder(nn.Module):
        """
        The BasicDecoder module takes an steganographic image and attempts to decode
        the embedded data tensor.
        Input: (N, 3, H, W)
        Output: (N, D, H, W)
        """
        def _conv2d(self, in_channels, out_channels):
            return nn.Conv2d(
                in_channels=in_channels,
                out_channels=out_channels,
                kernel_size=3,
                padding=1
            )
        def _build_models(self):
            self.layers = nn.Sequential(
                self._conv2d(3, self.hidden_size),
                nn.LeakyReLU(inplace=True),
```

```
            nn.BatchNorm2d(self.hidden_size),
            self._conv2d(self.hidden_size, self.hidden_size),
            nn.LeakyReLU(inplace=True),
            nn.BatchNorm2d(self.hidden_size),
            self._conv2d(self.hidden_size, self.hidden_size),
            nn.LeakyReLU(inplace=True),
            nn.BatchNorm2d(self.hidden_size),
            self._conv2d(self.hidden_size, self.data_depth)
        )
        return [self.layers]

    def __init__(self, data_depth, hidden_size):
        super().__init__()
        self.version = '1'
        self.data_depth = data_depth
        self.hidden_size = hidden_size
        self._models = self._build_models()
    def upgrade_legacy(self):
        """Transform legacy pretrained models to make them usable with new code versions."""
        # Transform to version 1
        if not hasattr(self, 'version'):
            self._models = [self.layers]
            self.version = '1'
    def forward(self, x):
        x = self._models[0](x)
        if len(self._models) > 1:
            x_list = [x]
            for layer in self._models[1:]:
                x = layer(torch.cat(x_list, dim=1))
                x_list.append(x)
        return x

class DenseDecoder(BasicDecoder):
    """
    The DenseDecoder module takes an steganographic image and attempts to decode
    the embedded data tensor.
    Input: (N, 3, H, W)
```

```
Output: (N, D, H, W)
"""
def _build_models(self):
    self.conv1 = nn.Sequential(
        self._conv2d(3, self.hidden_size),
        nn.LeakyReLU(inplace=True),
        nn.BatchNorm2d(self.hidden_size)
    )
    self.conv2 = nn.Sequential(
        self._conv2d(self.hidden_size, self.hidden_size),
        nn.LeakyReLU(inplace=True),
        nn.BatchNorm2d(self.hidden_size)
    )
    self.conv3 = nn.Sequential(
        self._conv2d(self.hidden_size * 2, self.hidden_size),
        nn.LeakyReLU(inplace=True),
        nn.BatchNorm2d(self.hidden_size)
    )
    self.conv4 = nn.Sequential(self._conv2d(self.hidden_size * 3, self.data_depth))
    return self.conv1, self.conv2, self.conv3, self.conv4
def upgrade_legacy(self):
    """Transform legacy pretrained models to make them usable with new code versions."""
    # Transform to version 1
    if not hasattr(self, 'version'):
        self._models = [
            self.conv1,
            self.conv2,
            self.conv3,
            self.conv4
        ]
        self.version = '1'
```

Додаток В. Ілюстративний матеріал





АКТУАЛЬНІСТЬ ТЕМИ

- Найчастіше методи стеганографії та методи ущільнення використовуються разом, утворюючи потужну систему захисту інформації. Ще одним важливим фактором у використанні перетворень зображень є розробка більш стійких до стегоаналізу алгоритмів для внесення інформації з метою приховування та використання різних форматів як контейнерів і стегоповідомлень. Це стосується не лише зображень і текстової інформації, але й файлів у будь-яких можливих форматах

ОСНОВНІ ЗАВДАННЯ

- Усі стеганографічні методи, що використовують математичні перетворення (ДКП, вейвлет-перетворення, перетворення Уолша-Адамара та ін.) вони обчислювально складніші, ніж «прямі» стеганографічні методи. Тому пропонується рішення, яке дозволить збільшити стеганоконтейнер і тим самим підвищити швидкість виконання перетворень, що у стаганографічних методах є найбільш складною складовою.
- Реалізований алгоритм для стеганографічних методів, що використовують математичного перетворення зображення (перетворення Уолша-Адамара та ін.) на основі штучної нейронної мережі.

АЛГОРИТМ РЕАЛІЗАЦІЇ

- Алгоритм ущільнення зображення на основі нейронної мережі типу карти Кохонена.

РОЗРОБКА ТА ДОСЛІДЖЕННЯ АЛГОРИТМУ І ПРОГРАМИ ДЛЯ ПРИХОВУВАННЯ ІНФОРМАЦІЇ В ЗОБРАЖЕННЯ НА ОСНОВІ НЕЙРОННОЇ МЕРЕЖІ

- Після ущільнення зображення виконаєм розробку програми для приховування інформації в зображення.
- Розглянемо модель нейронної мережі, що виконує приховування текстового повідомлення у графічні файли. Модель реалізована з використанням потужного фреймворку PyTorch та платформи CUDA та зображена на рисунку

Платформа CUDA

3760кб - Оригінал

4094кб - Із текстовим повідомленням

ВИСНОВКИ ПО РОБОТІ

- Виконані експериментальні дослідження показали, що застосування двовимірних карт Кохонена для векторного квантування коефіцієнтів перетворення Уолша-Адамара дозволяє підвищити коефіцієнт ущільнення порівняно з методами, які не використовують нейронні мережі. Коефіцієнт ущільнення може знаходитися у межах 3-15.
- У роботі представлено модель нейронної мережі, що виконує приховування текстового повідомлення у графічні файли. Модель реалізована з використанням потужного фреймворку PyTorch та платформи CUDA. Для навчання оригінальної мережі використано перший датасет та 2000 зображень із другого датасету. Датасети складаються із зображень середньої-високої якості. Тому для зменшення навантаження на наші обчислювальні засоби ми здійснили конвертацію обраних датасетів із зменшенням об'єму зображень, що також дозволить прискорити навчання мережі.

ДЯКУЮ ЗА УВАГУ

Додаток Г. Протокол перевірки на антиплагіат

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА
НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Дослідження нейроквантронних систем перетворення для збільшення стеганоконтейнера під час передавання потокової конфіденційної інформації

Тип роботи: магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ: Кафедра менеджменту та безпеки інформаційних систем
Факультет менеджменту та інформаційної безпеки
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 82 % Схожість 18 %

Аналіз звіту подібності (відмітити потрібне):

1. Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

2. Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

3. Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ (підпис) Коваль Н.П. (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ (підпис) Кириченко О.В. (прізвище, ініціали)

Керівник роботи _____ Сачанюк-Кавецька Н.В.