Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

Розробка методу та програмних засобів ущільнення зображень на основі карти Кохонена

Виконав: студент II курсу
групи 1ПІ-20м спеціальності
121 – Інженерія програмного забезпечення
Педченко Ярослав Володимирович

Керівник: к.т.н., доц. каф. ПЗ Майданюк В.П.

«____»_____2021 р.

Опонент: д.т.н.,проф.каф.КН Василевський О.М.

«____»_____2021 р.

Допущено до захисту
Завідувач кафедри ПЗ
д.т.н., проф. Романюк О. Н.
(прізвище та ініціали)

«____»_____2021 р.

ВНТУ – 2021

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
« 13 » вересня 2021 р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Педченку Ярославу Володимировичу

1. Тема роботи – Розробка методу та програмних засобів ущільнення зображень на основі карти Кохонена.

Керівник роботи: Майданюк Володимир Павлович, д.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від « 24 » вересня 2021 р. № 277.

2. Строк подання студентом роботи
1 грудня 2021 р.

3. Вихідні дані до роботи: метод ущільнення зображень – двовимірна карта Кохонена; коефіцієнт ущільнення – 3-4; час кодування зображення 512x512 - не більше 10 сек.; мова програмування – C#.

4. Зміст розрахунково-пояснювальної записки: вступ; аналіз та обґрунтування вибору методу розробки та постановка задачі дослідження; розробка структури та алгоритмів роботи програмного продукту; розробка програми для ущільнення зображень; тестування програми; економічна частина; висновки; додатки.

5. Перелік графічного матеріалу: мета МКР, наукова новизна МКР, нейрона мережа, загальна структура системи, головне вікно додатку, вікно з завантаженим зображенням, порівняння JPEG з картою Кохонена..

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Майданюк В. П., к.т.н., доц. каф. ПЗ		
5	Ратушняк О.Г., к.т.н., доц. каф. ЕПВМ		

7. Дата видачі завдання ___14 вересня 2021 р.___

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної Роботи	Строк виконання етапів роботи	Примітка
1	Аналіз та обґрунтування вибору методу розробки та постановка задачі дослідження	15.09. 2021 – 27.09.2021	Вик.
2	Розробка структури та алгоритмів роботи програмного продукту	28.09.2021 – 07.10.2021	Вик.
3	Розробка програми для ущільнення зображень	11.10.2021 - 27.10.2021	Вик.
4	Тестування програми	28.10.2021 – 16.11.2021	Вик.
5.	Економічна частина	17.11.2021 - 30.11.2021	Вик.

Студент _____ **Педченко Я. В.**

(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ **Майданюк В. П.**

(підпис) (прізвище та ініціали)

АНОТАЦІЯ

УДК. 004.4:004.92

Педченко Я. В. Магістерська кваліфікаційна робота зі спеціальності 121 – інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2021. с.

На укр. мові. Бібліогр.: 26 назв; рис.: 19; табл. 12.

У магістерській кваліфікаційній роботі розроблено засіб для ущільнення зображень з використанням нейроподібних методів, а саме, двовимірної карти Кохонена.

Розробка виконана мовою програмування C# в Unity, характеризуються зручністю та зрозумілістю інтерфейсу користувача, швидкістю та точністю опрацювання даних, що забезпечує всі вимоги користувача щодо ущільнення зображень.

У роботі проведено детальний аналіз методів і засобів ущільнення зображень. Сформульовано мету досліджень – підвищення швидкодії та коефіцієнта ущільнення зображень за допомогою карти Кохонена.

Запропоновано метод використання двовимірної карти Кохонена для ущільнення зображень. Розроблено базову архітектуру мережі на основі карти Кохонена. Запропоновано використання арифметичного кодування для ущільнення даних, отриманих після векторного квантування з використанням карти Кохонена. Запропоновано проводити обчислення блоків паралельно для підвищення швидкодії. Розроблено загальну схему кодування. Мовою програмування C# в Unity розроблено додаток для виконання ущільнення зображень запропонованим методом.

Ключові слова: векторне квантування, кодування зображень, карта Кохонена, ущільнення зображень.

ANNOTATION

UDC. 004.4:004.92

Pedchenko Y. V. Master's degree in specialty 121 – software engineering, educational program – software engineering. Vinnytsia: VNTU, 2021

In Ukr. Language. Rebligr.: 26 titles; Figs.: 19; Table. 12.

In the master`s thesis, developed a tool for compressing images using neural methods, namely, two-dimensional Kohen’s map.

Developed software products written using C# programming language in Unity is characterized by user interface usability, speed and accuracy of data processing, which provides all the requirements of the user for image compression.

There is a detailed analysis of image compression methods and means. The purpose of the research is to increase the image compression coefficient using the Kohen’s map.

The method of using a two-dimensional card for compressing images is proposed. The basic architecture of the network that based on Kohen’s maps was developed. Was decided to use arithmetic coding to compress data received after vector quantization using Kohen’s map. It is proposed to calculate blocks in parallel to improve performance. Was decided to use parallel calculation for beter generic encoding scheme was developed. Application made by using C# programming language to perform image compression by the proposed method.

Keywords: vector quantization, image coding, Kohonen map, image compression.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. У наші дні зображення використовуються повсюдно і в великих кількостях. Іноді використовують завеликі зображення, які займають багато дискового простору, тому існує необхідність в ущільненні зображень. Наразі існує багато різних методів і засобів ущільнення зображень [1-3].

Підвищення коефіцієнту ущільнення зображень при збереженні високої якості та покращення швидкодії їх обробки є пріоритетним напрямком досліджень, що проводяться в галузі кодування зображень. Одним з варіантів, який забезпечує вирішення цих задач є застосування штучних нейронних мереж.

У науковій літературі розглядаються різні підходи до застосування нейронних мереж при ущільненні зображень[1], проте на особливу увагу заслуговують підходи, що ґрунтуються на принципах векторного квантування зображень. Ці підходи забезпечують високу швидкодію при ущільнення і при цьому дозволяють зберегти відносно високу якість відновлюваного зображення. При векторному квантуванні зображення розділяється на квадратні блоки які розглядаються як вектори в 4-х вимірному, 16-и вимірному або 64-х вимірному просторі. Потім з цього простору вибирається обмежена кількість векторів, які з найбільшою точністю апроксимують вектори вилучені з вхідного зображення. Обрані вектори записуються в кодову книгу. Завдяки цьому використовується менше біт для представлення номера вектора ніж для початкового вектору, кількість векторів в кодовій книзі значно менша від кількості векторів в початковому зображенні, саме це дає можливість досягти хорошого рівня ущільнення.

Ідеальним варіантом для вирішення цих задач є самоорганізуючі нейронні мережі, наприклад самоорганізуюча мережа у вигляді двовимірної карти Кохонена, яка була запропонована фінським ученим Т. Кохоненом. Карта

Кохонена має властивості, які використовуються при ущільненні зображень методами векторного квантування. По-перше, вона дуже подібна до інших методів векторного квантування, які застосовують при ущільненні зображень з втратами, а по-друге близьким кластерам вхідних векторів відповідають близько розташовані нейрони, що збільшує ефективність ущільнення без втрат, яке застосовується на наступному етапі ущільнення [2].

Однак, програмна реалізація ущільнення зображень з використанням карти Кохонена практично відсутня, тому актуальною є розробка додатку ущільнення зображень з використанням нейроподібних методів на основі двовимірної карти Кохонена.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою роботи є підвищення швидкодії та коефіцієнта ущільнення зображень за допомогою двовимірної карти Кохонена.

Основними задачами дослідження є:

- провести аналіз існуючих методів і засобів ущільнення зображень для обґрунтування доцільності розробки;
- розробити або удосконалити метод та алгоритм ущільнення зображень з використанням двовимірної карти Кохонена;
- розробити додаток для виконання ущільнення зображень з використанням двовимірної карти Кохонена;
- провести експериментальні дослідження розробленого програмного засобу ущільнення зображень;
- оцінка комерційного потенціалу розробки.

Об'єкт дослідження – процес ущільнення зображень.

Предмет дослідження – методи та програмні засоби ущільнення зображень з використанням двовимірної карти Кохонена.

Методи дослідження. У процесі досліджень використовувались: теорія інформації та кодування, теорія нейронних мереж, методи обробки зображень для розробки моделей та методів ущільнення зображень; комп'ютерне моделювання для аналізу та перевірки отриманих теоретичних положень.

Наукова новизна отриманих результатів.

1. Подальшого розвитку отримав метод ущільнення зображень на основі векторного квантування, у якому, на відміну від існуючих, використано двовимірну карту Кохонена в якості векторного квантувача, що дозволило підвищити коефіцієнт ущільнення зображень на 10 % без значної втрати якості зображення.

2. Подальшого розвитку отримав метод ущільнення зображень на основі двовимірної карти Кохонена, який відрізняється від існуючих застосуванням паралельних обчислень та багатопотоковості, що дозволило підвищити швидкість кодування у 3 рази.

Практична цінність отриманих результатів. Практична цінність одержаних результатів полягає в тому, що на основі отриманих в магістерській кваліфікаційній роботі теоретичних положень запропоновано алгоритми та розроблено програмні засоби ущільнення зображень для комп'ютерних систем.

Особистий внесок здобувача. Усі наукові результати, викладені у магістерській кваліфікаційній роботі, отримані автором особисто. У друкованих працях, опублікованих у співавторстві, автору належать такі результати: узагальнена схема ущільнення зображень на основі карти Кохонена [4]; векторний кантувач на основі карти Кохонена [5]; алгоритм навчанні мережі [6].

Апробація матеріалів роботи. Основні положення магістерської кваліфікаційної роботи доповідалися та обговорювалися на міжнародних та всеукраїнських конференціях: Весняні наукові зібрання — 2020, XLV Міжнародна науково-практична інтернет-конференція (м. Суми, 22 травня 2020 року); L Науково-технічна конференція підрозділів Вінницького національного

технічного університету (2021); Міжнародної науково-практична Інтернет конференція «Електронні інформаційні ресурси: створення, використання, доступ» (9-10 листопада 2021 р., Суми/Вінниця).

Публікації. Основні результати досліджень опубліковано в 3 наукових працях у матеріалах конференцій.

Структура та обсяг роботи. Робота містить вступ, 5 розділів, висновки, перелік посилань, додатки. У першому розділі розглянуто сучасний стан питання ущільнення зображень, сформульовано основні завдання, які вирішуються в роботі. У другому розділі розроблено загальну схему ущільнення зображень, яка включає векторний кантувач на основі карти Кохонена та арифметичний кодер для ущільнення квантованих даних. У третьому розділі обґрунтовано вибір мови програмування, яка буде використовуватися при розробці програмного продукту, розроблено алгоритми роботи модулів програми та код модулів програми ущільнення зображень на основі двовимірної карти Кохонена. У четвертому розділі виконано тестування програми ущільнення зображень на основі карти Кохонена, розроблено інструкцію користувача. У п’ятому розділі виконано оцінку комерційного потенціалу розробки.

Перелік посилань - 26 джерел. У додатках міститься технічне завдання на роботу, лістинг коду та ілюстративна частина до захисту роботи.

1 АНАЛІЗ ТА ОБҐРУНТУВАННЯ ВИБОРУ МЕТОДУ РОЗРОБКИ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Аналіз стану ущільнення зображень

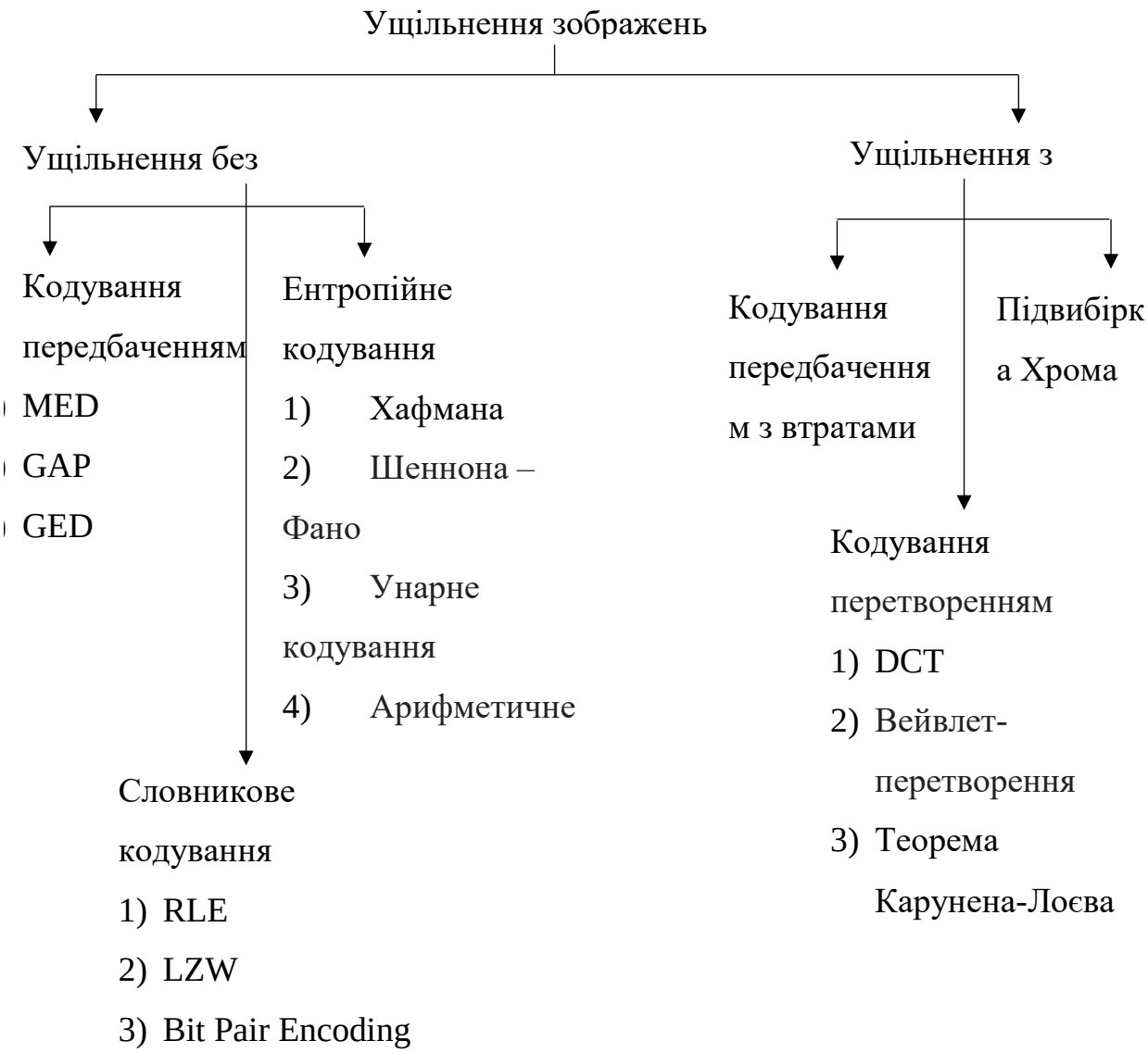
Недоліком растрових зображень є те, що вони досить великі. Якщо упустити заголовки файлів та рештою неграфічних даних, то розмір файлу пропорційний кількості пікселів у зображенні та кількості бітів шо представляють кожен піксель.

Використовуючи різні методи ущільнення зображень, можна значно зменшити фізичний розмір графічного файлу в пам’яті. Ущільнення інформації є одним зі способів її кодування. При ущільнення графічної інформації використовуються методи що дають змогу скоротити кількість байтів, необхідних для представлення зображення. А вже якість та місце пам’яті зайняте на носії інформації стисненого зображення залежить від використаного методу ущільнення та розміру і вмісту зображення. Графічний файл стискується в пам'ять і більш разів, існують методи, що стискають ще сильніше, але з втратами якості – при відновленні зображення губиться деяка частина інформації. Розпакована картинка може стати злегка розмитою і знебарвленою [7].

Існує два типи методів стиснення зображень – з втратами і без втрат. Обидві техніки працюють по-різному. Стиснення без втрат - це метод, який використовується для зменшення розміру файлу зображення, зберігаючи при цьому його якість, як і раніше. Він схожий на DSLR-камеру, яка пропонує можливість зберігати фотографії в різних форматах, таких як JPEG або RAW. Файли JPEG займають менше місця і не заповнять жорсткий диск швидко, але під час процесу перетворення ви можете втратити деякі дані. Файли у форматі RAW не мають стиснення і відмінно підходять, якщо ви професійний користувач. Метод стиснення з втратами - це ще один спосіб стиснення

зображень, який передбачає різання певної частини зображення для створення ще менших розмірів файлів. Одним із способів практики техніки стиснення з втратами є зменшення колірного простору зображення до найпоширеніших кольорів. Він часто практикується в GIF і кілька разів у зображеннях PNG для створення файлів набагато менших розмірів. При правильній практиці і в поєднанні з тремтіння, це може призвести до зображень, які майже ідентичні оригінальним зображенням.

Типи на які поділяється ущільнення зображень зображені на рисунку 1.1.



Рисунку 1.1 – типи ущільнення зображень

Існують такі види стандарти стиснення зображень:

– GIF – з його повною формою як формат графічного обміну, це растровий стандарт зображення, який був введений CompuServe в 1987 році. Цей формат підтримує до 8 біт на піксель, маючи на увазі, що зображення може включати до 256 різних кольорів RGB. Основною перевагою формату GIF є те, що він дозволяє анімовані зображення, які інші формати або стандарти не зможуть запропонувати вам.

– PNG – повна форма цього стандарту зображення - портативна мережева графіка. Це bitmapped стандарт зображення, який використовує без втрат метод стиснення зображень. Він був введений в першу чергу для заміни формату GIF. Однак цей формат не підтримується браузером Internet Explorer, і з цієї причини це рідкість у порівнянні з форматами JPEG і GIF. Цей стандарт зображення підтримує кольори на основі палітри, колірні простори RGB, RGBA та градацій сірого. Однією з помітних переваг, яку пропонує цей стандарт стиснення зображень, є те, що він підтримує кілька варіантів прозорості, включаючи прозорість альфа-каналу.

– BMP – це ще один стандарт стиснення зображень, який відомий як растровий пристрій зображення або файл, незалежний від растрового малюнка. Це більш швидкий графічний формат зображення, який корисний для зберігання растрових цифрових зображень і самостійно відображає зображення на пристрої. Цей метод стиснення використовується особливо в операційних системах OS/2 і Microsoft Windows.

– RAW – формат зображення RAW – це необроблене зображення, яке містить необроблену інформацію про зображення. Ця інформація про необроблене зображення зберігається у форматі RAW і корисна для надсилання або спільного використання між кількома особами. Деякі користувачі вважають за краще RAW над JPEG, оскільки RAW містить більше деталей у порівнянні з

файлами JPEG. Крім того, вони мають кращу якість у порівнянні з файлами зображень у форматі JPEG.

– JPEG це один з найбільш часто використовуваних форматів стиснення зображень. Він відомий як Joint Photographic Experts Group і є форматом стиснення з втратами, який використовується для створення зображень надзвичайно менших розмірів файлів. Однією з головних переваг цього стандарту є те, що він дозволяє дизайнерам точно налаштувати обсяг стиснення. В результаті дизайнери можуть досягти кращої якості зображення при використанні точно і в той же час досягти найменшого розміру зображення. Однак цей стандарт є стисненням з втратами і, отже, зображення, збережені в цьому форматі, можуть призвести до артефактів, де ви можете спостерігати пікселі та дивні ореоти навколо певних областей зображення. Він дуже корисний на зображеннях, що мають різкий контраст між кольорами. Це тому, що такі зображення, як правило, зберігають свою якість, і ви можете, отже, мати приємний вигляд остаточного зображення.

Незважаючи на постійне збільшення ємності пристроїв для зберігання інформації, також зростає об'єм візуальної інформації, що використовується в комп'ютерних системах і цей об’єм випереджає постійне збільшення ємності пристроїв для зберігання інформації, що робить вирішення задачі ущільнення зображень завжди актуальним. Мета кодування зображень – це зменшення числа біт, необхідних для зберігання або передачі зображень. Кодування пропорційно залежить від об'єму цифрового сигналу, який описує зображення з заданою точністю. Об'єм цифрового сигналу характеризується кількістю біт на елемент (відлік) зображення. Вважається, що для високоякісного відтворення зображення, необхідно, щоб на 1 елемент зображення припадало не менше 8 біт [8]. За допомогою деяких методів кодування вдається зменшити цю величину в декілька разів і забезпечити достатню якість відновленого зображення з цифрового сигналу об'ємом до 1 біта на елемент зображення. Відповідно до цього зменшується кількість пам’яті яка необхідна для зберігання таких

зображень і також це знижує необхідну пропускну здатність каналу для передачі зображень.

1.2 Аналіз методів розв’язання поставленої задачі

Для того щоб обрати оптимальний варіант вирішення задачі проведемо порівняння двох підходів які забезпечують приблизно однакові коефіцієнти ущільнення та якість відновленого зображення:

- фрактальне ущільнення зображень;
- ущільнення з використанням карти Кохонена.

Фрактальне стиснення зображень – це алгоритм стиснення зображень з втратами, заснований на застосуванні систем ітерованих функцій (IFS, які, як правило, є афінними перетвореннями) до зображень. Даний алгоритм відомий тим, що в деяких випадках дозволяє отримати дуже високі коефіцієнти стиснення для реальних фотографій природних об'єктів, що неможливо для інших алгоритмів стиснення зображень. Через складну ситуацію з патентуванням широкого розповсюдження алгоритм не отримав.

Самоорганізована карта – це нейронна мережа, яка виконує завдання візуалізації та кластеризації.

Порівняємо обчислювальні витрати які потребують дані підходи для своєї реалізації.

Для виконання фрактального ущільнення необхідна певна кількість операцій множення і ділення (1.1):

$$L_f = 8(4n^{k+1}(n^{k-1} - 3) + 9n^2) * n^{2k-2}, \qquad (1.1)$$

де n – розмір рангового блоку і розміри сторін зображення рівні $N=M=n^k$ [ ].

Для виконання ущільнення з використанням карти Кохонена з розмірами mxm і розміру фрагменту зображення, що піддається векторному квантуванню

nxn при тих же розмірах зображення кількість операцій множення необхідних для виконання виконання ущільнення.

Для одного фрагменту необхідно виконати фіксовану кількість операцій множення з урахуванням двох проходів (1.2):

$$m^2*n^2*2 \quad (1.2)$$

З урахуванням того, що загальна кількість фрагментів розміром nxn в зображенні складе n^{2k-2} кількість операцій множення визначається виразом (1.3):

$$L_k = 2*n^{2k}*m^2 \quad (1.3)$$

Взявши відношення L_f/L_k, отримаємо (1.4):

$$L_f/L_k \approx n^{2k-1} \quad (1.4)$$

Тобто виконання ущільнення зображень вимагає значно менше операцій множення і ділення. Зведемо отримані дані в таблицю 1.1.

Таблиця 1.1 – Порівняльні характеристики ущільнення зображень фрактальним методом і методом карти Кохонена

Метод ущільнення	Якість відновленого зображення	Коефіцієнт ущільнення	Кількість арифметичних операцій	Можливість масштабування зображення
Фрактальний методод	Висока	Високий	Більша в $>n^{2k-1}$	Так
Карта Кохонена	Висока	Високий	Менша	Ні

Аналіз таблиці 1.1 показує, що застосування карти Кохонена має переваги за більшістю параметрів у порівнянні з фрактальним методом. Тому в якості базового для подальших досліджень ущільнення зображень вибираємо карту Кохонена.

1.3 Порівняльний аналіз аналогів

Методи ущільнення растрових зображень поділяються на дві великі категорії: ущільнення з втратами та ущільнення без втрат. Методи ущільнення без втрат дають більш точні відновлення даних після декодування і може використовуватися для ущільнення будь-якої інформації. Методи з втратами мають вищий ступінь ущільнення, порівняно з методами без втрат, але допускає деякі відхилення декодованих даних від похідних[9].

Метод стиснення з втратами використовується, в основному, для графіки (JPEG), звука (MP3), відео (MPEG), тобто в тих місцях, де незначні відхилення від оригіналу непомітні або не суттєві, а ступінь стиснення, вважаючи на розмір файлу, дуже важлива. Стиснення без втрат використовується коли потрібно зберегти максимально точну інформацію без спотворень для таких файлів як тексти, високоякісний звуку Raw формат для фото і відео, тощо.

Ефективне кодування звичайно виконується в три етапи:

1 Знаходяться представлення сигналу зображення, наприклад у вигляді деякого набору коефіцієнтів перетворення. Така операція, як правило, зворотна.

2 Знижується точність представлення, але так, щоб виконувались задані вимоги до якості зображення. Така операція не зворотна.

3 Ліквідується статистична надлишковість, наприклад за допомогою коду Хаффмана. Ця операція зворотна.

При виконанні пп. 1, 2 найбільш широке застосування знаходять методи кодування на основі двомірних ортогональних перетворень [10]. Зокрема, стандарти JPEG та MPEG [12] передбачають застосування дискретного косинусного перетворення (ДКП), яке потребує меншу кількість арифметичних операцій в порівнянні із звичайним перетворенням Фур'є. Однак, навіть в цьому випадку потрібні надзвичайно великі обчислювальні потужності. Це свідчить про те, що необхідне детальне дослідження інших методів стискання, які дозволяли б вирішувати задачі кодування зображень при менших обчислювальних і, відповідно, апаратних витратах.

Отже, тема кодування зображень є актуальною і потребує пошуку нових методів кодування. Підвищення коефіцієнта ущільнення зображень при збереженні високої якості є пріоритетним напрямком досліджень, які виконуються в цій галузі. Підхід із застосування штучних нейронних мереж, що ґрунтуються на принципах векторного квантування зображень є одним із варіантів, який надає можливість вирішення даних задач, оскільки цей підхід забезпечує високий коефіцієнт ущільнення при збереженні високої якості відновленого зображення. Метою роботи є розробка, реалізація і підвищення швидкодії алгоритму кодування зображень за допомогою карти Кохонена.

1.4 Постановка задачі для ущільнення зображень

Після аналізу питання методів ущільнення зображень визначено наступні завдання, які необхідно виконати для розробки програмного продукту:

- визначити найбільш ефективний метод ущільнення зображень;
- розробити нейрону мережу для підвищення ефективності обраного методу;
- розробити інтерфейс програмного продукту, який буде легким для розуміння та інтуїтивним користувачу;
- розробити програмний продукт, призначений для вирішення проблеми;
- провести тестування програмного продукту для перевірки всіх можливих варіантів використання.

1.5 Висновки

У першому розділі розглянуто сучасний стан питання ущільнення зображень. Розглянуто аналоги та виконано їх порівняння між собою та програмою, яка розробляється, що дозволило обґрунтувати доцільність розробки програмного засобу. Крім того, проаналізовано можливі методи вирішення питання, сформульовано основні завдання, які вирішуються в роботі.

2 РОЗРОБКА СТРУКТУРИ ТА АЛГОРИТМІВ РОБОТИ ПРОГРАМНОГО ПРОДУКТУ

2.1 Аналіз інформаційного забезпечення програми

Імпульсно-кодова модуляція – процес при якому через певні інтервали часу беруться відліки аналогового сигналу і незалежно один від одного квантуються і кодуються цифрами[11].

Для запобігання появи фальшивих контурів необхідно використовувати не менш 50 рівнів квантування для одної складової кольору, що відповідає 6-8 розрядному слову на кожен елемент колірної складової зображення. Через великі обсяги інформації ІКМ використовується лише при внутрішньо-студійній передачі телевізійних сигналів рівнобіжним кодом. ІКМ є базовим, канонічним представленням зображення в цифровій формі [12].

Існує цілий ряд методів кодування що використовує передбачення. Серед таких методів найбільш дослідженою є диференційно-імпульсна кодова модуляція (ДІКМ). Цей метод працює наступним чином: на основі лінійної комбінації значень яскравості попередніх елементів робиться передбачення значень яскравості наступного елемента зображення.

Отримана в результаті передбачення оцінка віднімається від істинного значення яскравості сигналу і різницевий відлік квантується невеликим числом рівнів, завдяки цьому досягається скорочення обсягу даних [1,9]. Використання методів з передбаченням дає змогу ущільнити зображення в 2-2,5 рази при відносно простій технічній реалізації. Але недоліками цих методів є: наявність погрішності в місцях де є різкої зміни яскравості; досить низька завадостійкість.

Класифікація методів кодування наведена на рисунку 2.1.

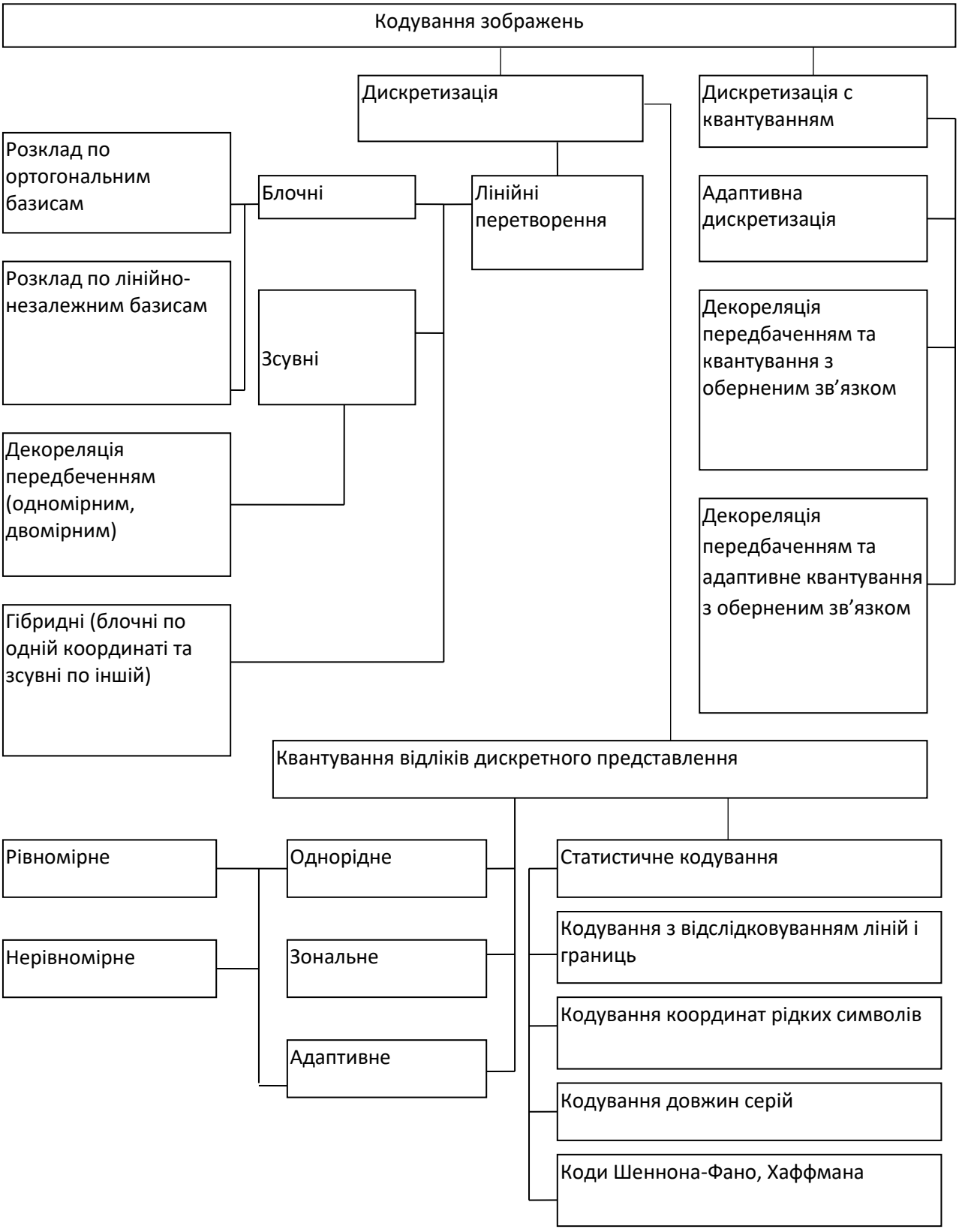


Рисунок 2.1- Класифікація методів кодування зображень

Один з важливих класів методів кодування зображень – це методи кодування зображень на основі перетворень, що є непрямими методами. Зображення проходить через унітарне математичне перетворення після чого отримані в результаті обробки коефіцієнти перетворення квантуються та кодуються для подальшого здійснення їхньої передачі по каналах зв’язку чи запису у файл. Для більшості зображень значення більшої частини коефіцієнтів перетворення порівняно малі, тому ці коефіцієнти часто можна відкинути чи виділити для їхнього кодування невелике число двійкових розрядів.

Найбільшого поширення одержали методи на основі двовимірних ортогональних перетворень. Це методи перетворення такі як: перетворення Карунена-Лоєва, дискретне перетворення Фур'є (ДПФ), дискретне косинусное перетворення (ДКП), перетворення Уолша-Адамара [13] і т.д. Коефіцієнти стиску отримані за допомогою даних методів можуть складати 6-8 раз, саме завдяки цьому деякі з цих методів знайшли широке використання та навіть були прийняті як промислові стандарти для кодуванню зображень. Загальний недолік що присутній в цих методах являється висока обчислювальна складність, а також неможливість рішення ряду задач на обмеженому обсязі даних.

Серед статистичних методів ущільнення зображень широке застосування мають блокові методи кодування зображень. За цими методами блоки зображення розміром MN елементів кодуються відповідно до імовірності їхньої появи в графічному файлі. Для найбільш вірогідних блоків, що найчастіше зустрічаються зображенні, використовують короткі кодові слова, а для менш вірогідних, які рідко зустрічаються в зображенні, використовують довші кодові слова (алгоритм Гаффмана), у результаті чого досягається стиск даних [14]. Коефіцієнти ущільнення отримані за допомогою таких методів ущільнення зображень можуть складати зменшення файлу в 4-5 раз.

Ще один важливим класом методів кодування зображень являються інтерполяційні методи ущільнення зображень. Вони базуються на чисельних

методах апроксимації, за допомогою яких в виді послідовності чи двовимірного масиву відліків яскравостей приблизно передаються через неперервні функції. При цьому кодування проводиться лише для деяких відліків, а сусідні з ними відліки одержують свої значення в результаті інтерполяції поліномами кодованих значень, поліноми не вище третього ступеню. Коефіцієнт ущільнення при використанні таких методів може досягати ущільнення у 5-6 разів.

Практично всі розглянуті методи кодування зображень можуть бути упорядковані в схему, яка наведена на рисунку 2.1. Більшість із них передбачають триступеневу процедуру кодування: роздільну дискретизацію, квантування відліків отриманого дискретного представлення та послідуюче статистичне кодування.

Фрактальне кодування напівтонових зображень засновано на гіпотезі, відповідно до якої в будь-якому зображенні можна знайти локальну самоподобу різних його частин. Існуючі алгоритми фрактального ущільнення, як правило, дотримують наступної схеми кодування [10]. Зображення що кодується розбивається на безліч блоків, що не перекриваються їх називають ранговою областю, потім для кожного блоку шукають у межах всього зображення, блок більшого розміру, пікселі якого шляхом деякого перетворення, що задається декількома коефіцієнтами, переводилися б у пікселі рангової області. Для пошуку оптимальної відповідності рангових областей і доменів необхідно провести повний перебір варіантів. Відображення, що переводить зображення в зображення формується з виконаних перетворень, що переводять домени в рангові області. Кодом зображення при цьому будуть виступати коефіцієнти перетворень і розміри рангових областей, а також місце розташування зображення. В результаті цього кількість біт, необхідних для опису коду, буде і меншою кількості біт, які необхідних для опису вихідного зображення. У відомих фрактальних методах стиску зображень значення коефіцієнта стиску може досягати 100 при прийнятній якості відновлення [10].

Недоліком фрактального методу ущільнення ущільнення є дуже низька швидкодія ущільнення зображень.

Отримати великий коефіцієнт ущільнення даних, використовуючи лише один метод (крім фрактального) практично не є можливим, тому набагато ефективне проводити кодування зображень користуючись декількома методами в декількох етапах. Саме така концепція покладена в основу стандартів, на яких розроблено міжнародною організацію по стандартизації (ISO). Цими стандартами являються стандарти JPEG та MPEG.

JPEG одержав свою назву від абревіатури об'єднаної групи експертів в області фотографії, що його і розробила[15]. Основна ідея методу розділити інформацію зображення за рівнем важливості, і відкинути менш важливу частину інформації задля зменшення загального обсягу даних які необхідно зберігати. Такий ефект досягається завдяки перетворенню матриці колірних значень на матрицю амплітуд, що відповідають визначеним частотам розкладання зображення. JPEG відкидає частину високочастотних компонентів зображення тому, що людське око менш чутливе до високочастотних варіацій кольору. Залишаються компоненти з низькими частотами. Значення пікселя, отримане при відновленні зображення, трохи відрізняється від вихідного значення, бо частина інформації загублена.

При кодуванні рухомих зображень у відповідності до стандарту MPEG використовуються два типи ущільнення: внутрішньокадрове кодування (подібно JPEG) і міжкадрове кодування. Міжкадрове кодування засноване на кодуванні з передбаченням і інтерполяції. Кадри рухомого зображення містять багато ідентичних даних. Якщо використовувати цей факт, то в результаті можна набагато збільшити ступінь стиску.

Кодування JPEG і MPEG містить наступні недоліки:

- недостатньо високі коефіцієнти стиску складних зображень;
- відсутність можливості зміни роздільної здатності;
- втрата вірогідності відновлених зображень [12].

Однак, незважаючи на ці недоліки, дані методи поки що мають найкращі реально досягнуті характеристики такі як коефіцієнт ущільнення, якість зображення та швидкодію алгоритмів і при цьому ці формати підтримуються основними виробниками комп'ютерної техніки і техніки зв'язку.

У науковій літературі розглядаються різні підходи до застосування нейронних мереж при ущільненні зображень[2], проте особливу увагу заслуговують підходи, що ґрунтуються на принципах векторного квантування зображень. Такі підходи забезпечують високу швидкодію при ущільнення і при цьому забезпечують збереження високої якості відновлюваного зображення. При обробці зображення методами векторного квантування зображення розбивається на квадратні блоки які надалі розглядаються як вектори в чотирьохвимірному, шістнадцятивимірному чи шістьдесятчотирьохвимірному просторі. Потім із цього простору вибирається обмежена кількість векторів, що мають найточніше апроксимувати вектори вилучені з вхідного зображення. Обрані вектори утворюють кодову книгу. Далі вже ця книга передаються по каналах зв’язку або записується в файл, номери векторів з кодової книги, які знаходяться найближче до векторів вилучених з початкового зображення і сама кодова книга. Оскільки кількість векторів записаних в кодовій книзі значно менша від загальної кількості векторів у початковому зображенні, то для представлення векторів з книги витрачається менше біт ніж для представлення початкових векторів. За рахунок цього і досягається ущільнення.

Ідеальним рішенням для вирішення таких задач є використання самоорганізуючих нейронних мереж, а саме, самоорганізуючих мереж у вигляді двовимірної карти Кохонена, що була запропонована фінським ученим Т. Кохоненом. Карта Кохонена має властивості, що використовують при ущільненні зображень методами векторного квантування: по-перше, вона подібна до інших методів векторного квантування, які застосовують при ущільненні зображень з втратами, а по-друге близьким кластерам вхідних

векторів відповідають близько розташовані нейрони, що збільшує ефективність ущільнення без втрат, яке застосовується на наступному етапі ущільнення[2].

2.2 Розробка структури системи

Схема ущільнення зображень за допомогою використання карти Кохонена приведена на рисунку 2.2. Після перетворення блоків зображення в вектори, виконується векторне квантування за допомогою карти Кохонена. Вихідні дані після векторного квантувача надходять до арифметичного кодера, який виконує кодування зображення без втрат. Декодування збереженого зображення виконується в зворотному порядку спочатку дані розкодовуються арифметичним кодером. Розкодовані дані або кодова книга разом з навчальними векторами передається векторному квантувачу який відновлює незбережені пікселі в кожному блоці зображення.

У вихідний файл записується квантові значень вхідних векторів і кодова книга. Розмір книги значно менший якщо порівнянювати з вхідним зображенням тому це не впливає на коефіцієнт ущільнення. В ролі векторного кантувача виступає карта Кохонена з розміром 16x16 або більшим. Вибір розміру карти Кохонена пов’язаний з тим, що зображення представляється з точністю 8-біт на елемент зображення для напівтонових зображень, або 24 біти для кольорового зображення[3].

Застосування арифметичного кодера на етапі ущільнення без втрат забезпечує найбільший коефіцієнт ущільнення в порівнянні з іншими методами кодування без втрат [12].

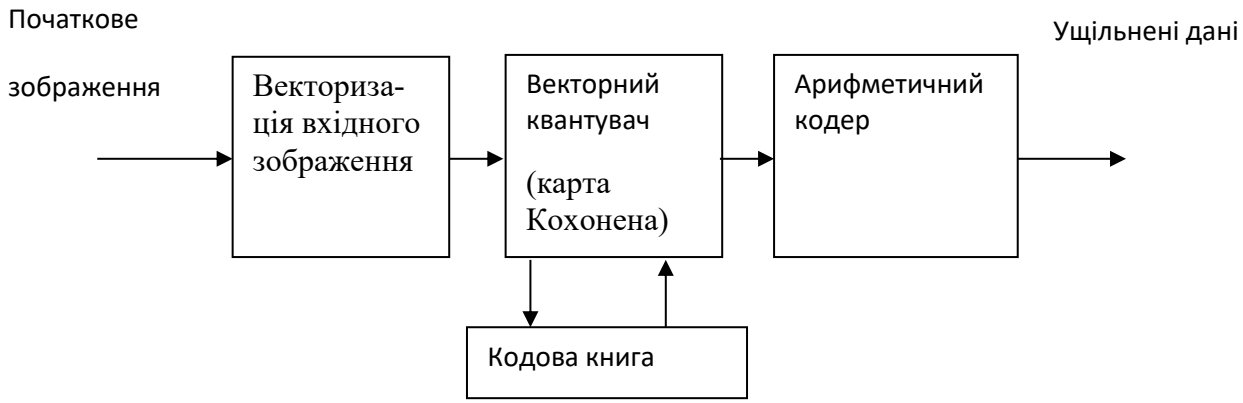


Рисунок 2.2 – Загальна схема кодування (кодер)

2.3 Розробка дизайну інтерфейсу

Дизайн інтерфейсу користувача – це принципи та відповідний процес розробки інтерфейсу користувача для машин та програмного забезпечення (таких як комп'ютери, побутові, мобільні та інші електронні пристрої) при якому досягають максимально зручний спосіб їх використання користувачами[16].

Існують такі види інтерфейсі: графічний інтерфейс користувача, сенсорний графічний інтерфейс користувача, інтерфейс на основі меню, інтерфейс командного рядка, розмовний інтерфейс користувача. Кожен має свої переваги та недоліки проаналізуємо їх для вибору найбільш підходящого виду інтерфейсу.

Переваги графічного інтерфейсу користувача:

- підходить для нетехнічних користувачів;
- складність дій прихована від користувачів;
- покращені привабливими візуальними ефектами;
- негайний візуальний зворотний зв'язок;
- використовуйте моделі та зображення з реального світу;
- дозволяє використовувати кілька пристроїв вводу.

Недоліки графічного інтерфейсу користувача:

- потрібні ресурси живлення та пам'яті;
- може мати низьку видимість ;
- може перевантажити користувачів зростаючою кількістю елементів контролю;
- приховані команди потрібно шукати навмисно.

Переваги сенсорного графічного інтерфейсу користувача:

- простіше і швидше, ніж маніпулювання мишею або введення тексту;
- уникнення зовнішніх пристроїв, таких як клавіатура або миша;
- можливість додавання різних рухових дій;
- доступно для дітей і старійшин;
- рухи збільшення сприяють доступності для людей з вадами зору;
- адаптований до широкого спектру пристроїв.

Недоліки сенсорного графічного інтерфейсу користувача:

- розмір елементів керування обмежений розміром мобільного дисплея;
- додаткові рухи може бути нелегко виявити;
- може бути без необхідності активована бродячими дотиками.

Переваги інтерфейсу, керованого меню:

- зручно для початківців комп'ютерів і початківців користувачів;
- низьке когнітивне навантаження на користувачів;
- знайомий інтерфейс на різних платформах;
- ви відповідаєте за створення замовлення та ієрархії для шляхів користувача;
- більше контролю над взаємодією з користувачами;
- простий в реалізації в різного роду пристроях.

Недоліки інтерфейсу, керованого меню:

- обмежені параметри меню;
- підменю може бути важко знайти;

– ризики зайняти багато місця на екрані або бути занадто маленьким;
– вимагає непотрібних дій для простого завдання.

Переваги інтерфейсу командного рядка:

– швидше, ніж інші типи інтерфейсу користувача;
– менше вимог до обробки процесора;
– працює з меншою роздільною здатністю;
– легко ваги в розмірах;
– можливість перетворення повторюваних завдань в одну команду;
– можливість запускати взаємодії між додатками для виконання складних дій.

Недоліки інтерфейсу командного рядка:

– вимагає досвіду та / або навичок програмування;
– друкарські помилки в синтаксисі команд призводять до помилок;
– зазвичай приймає тільки тип вводу клавіатури;
– не інтуїтивно зрозумілий – вимагає читання посібника перед його використанням.

Переваги розмовного інтерфейсу користувача:

– універсальний в додатках;
– не потрібно вчитися новим навичкам;
– голос забезпечує реалістичне відчуття;
– зв'язок із користувачами на особистому рівні;
– відповідає контекстом для створення взаємодій;
– адаптація до статі, тону, акценту та темпу мовлення;
– може бути інтегрований в існуючі додатки.

Недоліки розмовного інтерфейсу користувача:

– обмежена кількість візуальних і текстових підказок;
– формулювання команд може бути складним.

Проаналізувавши різні види інтерфейсів, їх переваги та недоліки було вирішено використовувати графічний інтерфейс при розробці додатку через його переваги в простоті роботі з ним користувача.

Дизайн головного вікна зображено на рисунку 2.3.

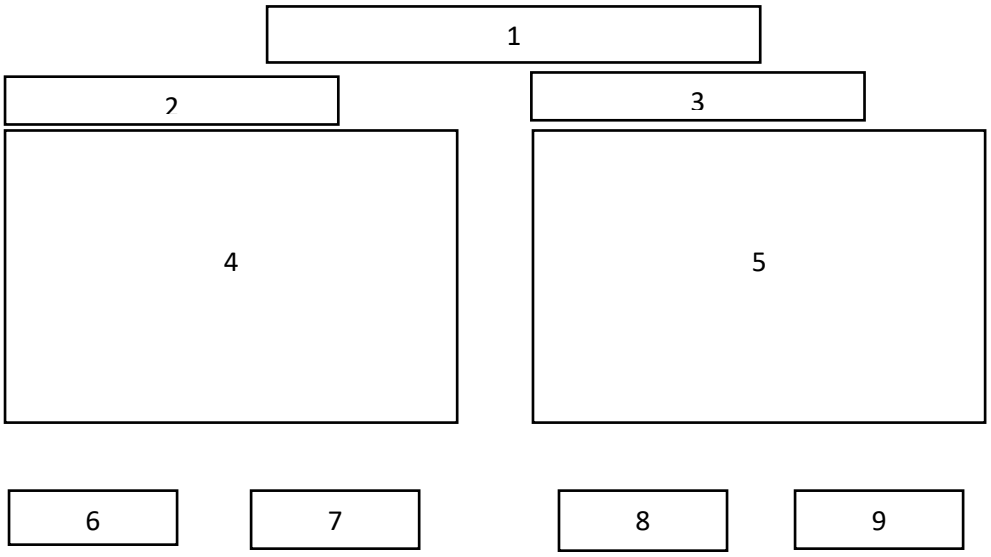


Рисунок 2.3 – Вигляд головного вікна програми

У вікні додатку знаходиться:

1) Назва відкритого зображення
2) Назва лівого вікна Відкрите\Оригінальне зображення
3) Оброблене зображення
4) Безпосередньо саме відкрите зображення
5) Вигляд зображення після обробки
6) Кнопка для відкриття звичайних зображень
7) Кнопка для відкриття збережених оброблених зображень
8) Кнопка для запуску обробки відкритого зображення
9) Кнопка для зберігання обробленого зображення

2.4 Розробка алгоритму векторного квантування зображень

Для розробки алгоритму векторного квантування було вирішено використовувати двовимірну карту Кохонена адже це дозволить підвищити коефіцієнт ущільнення зображень на 10 % без значної втрати якості зображення.

Самоорганізаційна карта Кохонена – це нейронна мережа, яка використовується для конструювання багатовимірного простору в простір з нижчою розмірністю (найчастіше, двовимірний), створює дискретне представлення вхідних просторів навчальних вибірок, які називаються картою, і тому використання цього типу нейронної мережі є методом для зниження розмірності[17]. Базова архітектура мережі SOFM наведена на рисунку 2.4.

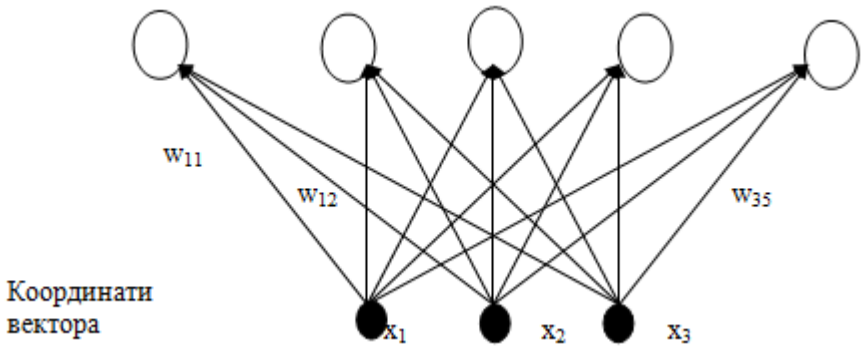


Рисунок 2.4 – Базова архітектура мережі SOFM

Вихідні елементи називаються кластерними елементами або кодовими словами, які розміщуються в виді одно або двовимірного масиву. Кількість кластерних елементів менша, якщо порівняннювати з кількістю навчальних зразків, оскільки метою є отримання спрощеної характеристики вхідних даних. Саме це дає можливість для використання SOFM як векторного квантувача [4].

Кожний нейрон представляється ваговими коефіцієнтами w_{ij} . Алгоритм навчання мережі працює таким чином:

1. Ініціалізація вагових коефіцієнтів випадковими значеннями.

2. Обчислення відстані до навчального вектора для кожного кластерного елемента (2.1):

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

3. Пошук кластерного елементу j для якого d_j є мінімально.

4. Оновлення вагових коефіцієнти для кластерних елементів із круга заданого радіуса з центром в j елементі за формулою (2.2):

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

де η - норма навчання, x_i – координата навчального вектора.

5. Оновлення норми навчання η і радіусу при необхідності і повторення пунктів 1-5 для наступного навчального вектора.

Норма навчання змінюється з часом., наприклад, спочатку мати значення 0,9, а потім змінюватись лінійно до деякого фіксованого значення, наприклад 0,01, після чого залишатися незмінною. Радіус вибирається достатньо великим, щоб обновлялись всі елементи, але з часом радіус зменшується і в кінці повинен обновлятись тільки один елемент[6].

Це дозволить після проведення навчання досить точно визначати пікселі зображення, що найкраще апроксимують кожну область. На великих зображеннях такі зміни будуть непомітні, але розміри файлів значно зменшаться. Точність визначення залежить від кількості етапів навчання, але прийнятні результати можна отримати вже після 25 етапів навчання, тому вирішено проводити саме 25 етапів навчання перед обробкою зображення.

Є кілька моментів, щоб мати на увазі тут:

Самоорганізаційна карта Кохонена зберігає взаємозв'язок і структуру вхідного набору даних.

Самоорганізаційна карта Кохонена проходить весь цей процес саме для того, щоб максимально наблизитися до набору даних. Для цього він повинен прийняти топологію набору даних.

Самоорганізаційна карта Кохонена виявляє кореляції, які не були б в іншому випадку легко ідентифіковані. Якщо є набір даних із сотнями або тисячами стовпців, було б практично неможливо скласти кореляції між усіма цими даними. Самоорганізаційна карта Кохонена обробляє це, аналізуючи весь набір даних і фактично зіставивши його, щоб можна було легко читати.

Самоорганізаційна карта Кохонена класифікує дані без необхідності перегляду. Самоорганізовані карти є неконтрольованою формою глибокого навчання. Не потрібно надавати свою карту з мітками для категорій класифікації даних. Мережа розвиває свої класи сама.

Самоорганізаційна карта Кохонена не вимагає цільових векторів і не проходить процес зворотного розповсюдження. Як відомо з теорії штучних нейронних мереж, мережа повинна бути забезпечена цільовим вектором. Потім дані проходять через мережу, витягають результати, порівнюють їх з цільовим вектором, виявляють будь-які помилки, а потім процес зворотного розповсюдження, щоб оновити ваги. Оскільки у нас немає цільового вектора, то і не буде помилок для карти і відповідно зворотного розповсюдження.

Існують бічні зв'язки між вихідними вузлами. Єдине з'єднання, яке виникає між вихідними вузлами в самоорганізаційній карті Кохонена, - це з'єднання push-and-pull між вузлами та IMB на основі радіуса навколо кожного BMU. Немає функції активації, як зі штучними нейронними мережами. Іноді вузли вишикуються в сітці, але єдина функція для цієї сітки полягає в тому, щоб уточнити, що ці вузли є частиною самоорганізаційної карти Кохонена, і акуратно організувати їх. Сітка не з'єднує вузли.

2.5 Арифметичне кодування

Арифметичне кодування – алгоритм ентропійного стиснення, який на відміну від алгоритму Гаффмана не має жорсткої постійної відповідності вхідних символів - групам біт вихідного потоку[18]. Основні його принципи розроблені наприкінці 70-х років. Арифметичне кодування використовує імовірність появи символу у файлі, як основу технологію ущільнення. У результаті виконання арифметичного кодування символьна послідовність замінюється дійсним числом більше нуля і менше одиниці [12].

Перша буква слова одержує інтервал з нижньою границею $\beta_0^{\,l}$ і з верхньої - $\beta_0^{\,h}$. Нижня границя інтервалу і стає першою значущою цифрою коду. Потім виконується розрахунок границь підінтервалів для кожної наступної букви за виразами (2.3):

$$\begin{aligned} \beta_n^l &= \beta_{n-1}^l + (\beta_{n-1}^h - \beta_{n-1}^l) P_n^l \\ \beta_n^h &= \beta_{n-1}^l + (\beta_{n-1}^h - \beta_{n-1}^l) P_n^h, \end{aligned} \qquad (2.3)$$

де β^l, β^h – нижня і верхня границя кодового інтервалу, P^l і P^h – нижня і верхня границі інтервалу імовірності для символу.

Арифметична кодування зіставляє рядок символів даних (джерел), кодовий рядок таким чином, що вихідні дані можуть бути відновлені з рядка коду. Алгоритми кодування та декодування виконують арифметичні операції на коді рядка. Одна рекурсія алгоритму обробляє один символ даних. Арифметичне кодування насправді є сімейство кодів, які поділяють властивість розглядати рядок коду як величину. Поняття компресійних систем фіксує ідею про те, що дані може бути перетворено в щось, що закодовано і передається до

місця призначення, а потім перетворюється назад у вихідних даних. Будь-який підхід до стиснення даних, будь то використання арифметичного кодування, кодів Гаффмана або будь-якої іншої техніки кодування, має модель, яка робить деякі припущення про дані та події.

Сам код може бути незалежним від моделі. Наприклад система, яка стискає форми хвиль (наприклад, оцифрована мова) може спрогнозувати наступне значення та закодувати помилку. У цій моделі закодована помилка, а не фактичні дані. Як правило, на кодер компресійної системи, дані, які потрібно стиснути, подаються модельним блоком. Модель визначає події, які повинні бути закодованими, і оцінку відносної частоти (ймовірність) подій. Кодер приймає події та деяке значення її відносної частоти та генерує рядок коду.

Проста модель - це безпам'ятна модель, де дані самі символи кодуються за єдиним кодом. Ще однією моделлю є марковська модель першого порядку, яка використовує попередній символ як контекст для поточного символу. Розглянемо, наприклад, стиснення англійських речень. Якщо символ даних (в даному випадку буква) "q" - це попередня буква, Ми очікуємо, що наступний символ буде "u". Марковська модель є залежною моделлю; у нас є різні очікування для кожного символу (або в прикладі кожної літери), Залежно від контексту. Контекст, у певному сенсі, є станом, що регулюється минулою послідовністю символів. Мета контексту полягає в тому, щоб забезпечити розподіл ймовірностей, або статистику, для кодування (декодування) наступного символу.

Для виконання арифметичного кодування спочатку потрібно визначити правильну модель. Пам'ятайте, що функція моделі полягає в забезпеченні ймовірності даного персонажа в повідомленні. Концептуальна ідея арифметичної моделі кодування полягає в тому, що кожен символ буде володіти своїм унікальним сегментом числової лінії дійсних чисел від 0 до 1. Важливо відзначити, що існує багато різних способів моделювання характеру ймовірності. Деякі моделі статичні, ніколи не змінюються. Інші оновлюються

після обробки кожного символу. Тільки дві речі, які мають для нас значення, це те, що 1) модель намагається точно передбачити ймовірність появи символа, а 2) кодер і декодер мають однакові моделі в будь-який час.

Як приклад, ми можемо почати з кодера, який може кодувати лише алфавіт зі 100 різних символів. У простій статичній моделі почнемо з великих літер, потім з букв нижнього регістру. Це означає, що перший символ, "A", буде володіти числовим рядком від 0 до 0.01, "B" буде володіти відбою від 0,01 до 0.02 і так далі. (У всіх випадках це строго напівзакритий інтервал, тому діапазон ймовірності для "A" насправді > = 0 і < 0.01.). За допомогою цієї моделі мій кодер може представляти одну букву "B", вивівши число з плаваючою комою, яке менше 0.02 і більше або дорівнює 0.01. Так, наприклад, арифметичний кодер, який хотів створити цю одну букву, може вивести 0.15 і зробити це. Кодер, який просто виводить окремі символи, не дуже використовується. Кодування рядка символів передбачає трохи складніший процес. У цьому процесі перший символ визначає діапазон числової лінії, який відповідає інтервал, призначеному йому моделлю. Для символу "B" це означає, що повідомлення знаходиться між 0.01 і 0.02.

Арифметичне кодування забезпечує високий ступінь ущільнення даних, коли зустрічаються дані з частотою появи різних символів, що сильно відрізняється одна від одної. Хоча процедура арифметичного кодування вимагає потужних обчислювальних ресурсів, що може привести до затримки при передачі даних, проте варто очікувати високих коефіцієнтів ущільнення при його застосуванні для кодування зображень.

2.6 Паралельне обчислення блоків зображення

Паралельні обчислення – це форма обчислень, в яких кілька дій проводяться одночасно. Ґрунтуються на тому, що великі задачі можна

розділити на кілька менших, кожну з яких можна розв'язати незалежно від інших[19].

Існує декілька рівнів паралельних обчислень таких як: бітовий, інструкційний, даних, задач. Зазвичай вони використовується в високо продуктивних обчисленнях але останнім часом інтерес до них зріс через обмеження в рості частоти обробки даних. В наші дні всі вже мають прилади які мають багатоядерні процесори навіть смартфони вже мають по декілька ядер в процесорі. Тому використання паралельних обчислень в проекті в наші дні вже можна вважати необхідністю а не можливістю.

Кількість апаратного забезпечення збільшується при паралельній обробці і разом з нею значення системи поліпшується. Але технологічні розробки знизили витрати на апаратне забезпечення до такої міри, що паралельні методи обробки економічно можливі. Паралельну обробку можна розглядати з декількох рівнів складності. На найнижчому рівні ми класифікуємо паралельні та послідовні операції за використовуваним методом регістрів. Регістри Shift виконують серійно потроху за раз, в той час як регістри з функцією паралельного навантаження з усіма бітами слова разом.

Паралельна обробка на більшому рівні складності може управлятися, маючи безліч функціональних одиниць, які реалізують ідентичні або різні операції разом. Паралельна обробка створюється шляхом присвоєння інформації кільком функціональним одиницям. Наприклад, операції арифметики, логіки і зсуву можна розділити на три одиниці, а операнди перенаправляються на кожну одиницю під наглядом блоку управління. На малюнку відображається один можливий спосіб розбиття блоку реалізації на вісім функціональних блоків, що працюють паралельно. Операнди в регістрах використовуються для одного з одиниць, що спирається на операцію, зокрема пов'язану з інструкціями з операндами.

Як правило, комп'ютерний вчений буде розділити складне завдання на кілька частин за допомогою програмного інструменту і призначити кожну

частину процесору, потім кожен процесор вирішить свою частину, а дані збираються програмним інструментом для зчитування рішення або виконання завдання. Кожен процесор буде працювати в звичайному режимі і буде виконувати операції паралельно, як вказано, витягуючи дані з пам'яті комп'ютера. Процесори також будуть покладатися на програмне забезпечення для спілкування один з одним, щоб вони могли залишатися в синхронізації щодо змін значень даних. Припускаючи, що всі процесори залишаються в синхронізації один з одним, в кінці завдання програмне забезпечення буде відповідати всім частинам даних разом. Комп'ютери без декількох процесорів все ще можуть бути використані в паралельній обробці, якщо вони об'єднані в мережу для формування кластера.

Існує кілька типів паралельної обробки, два з найбільш часто використовуваних типів включають SIMD і MIMD. SIMD, або одна інструкція з декількома даними, є формою паралельної обробки, в якій комп'ютер буде мати два або більше процесорів, які слідують одному набору інструкцій, в той час як кожен процесор обробляє різні дані. SIMD зазвичай використовується для аналізу великих наборів даних, заснованих на тих же заданих тестах. MIMD, або кілька інструкцій декількох даних, є ще однією поширеною формою паралельної обробки, яка має два або більше своїх власних процесорів і буде отримувати дані з окремих потоків даних. Інший, менш використовуваний, тип паралельної обробки включає в себе MISD, або кілька інструкцій одинарних даних, де кожен процесор буде використовувати інший алгоритм з однаковими вхідними даними.

Якщо паралельна обробка може виконувати кілька завдань за допомогою двох або більше процесорів, послідовна обробка (також називається послідовною обробкою) буде виконувати лише одне завдання за один раз, використовуючи один процесор. Якщо комп'ютеру потрібно виконати кілька призначених завдань, він виконає одне завдання за один раз. Точно так само, якщо комп'ютер, що використовує послідовну обробку, повинен виконати

складне завдання, то це займе більше часу в порівнянні з паралельним процесором.

Індивідуально кожен процесор працює так само, як і будь-який інший мікропроцесор. Процесори діють за інструкціями, написаними мовою асемблера. На основі цих інструкцій процесори виконують математичні операції над даними, витягнутими з пам'яті комп'ютера. Процесори також можуть переміщати дані в інше місце пам'яті. У послідовній системі це не проблема, якщо значення даних змінюються в результаті роботи процесора. Процесор може включати нове значення в майбутні процеси і продовжувати. У паралельній системі зміни значень можуть бути проблематичними. Якщо кілька процесорів працюють з однаковими даними, але значення даних змінюються з плином часу, конфліктні значення можуть призвести до спаду системи або аварійного завершення роботи. Щоб запобігти цьому, багато паралельних систем обробки використовують певну форму обміну повідомленнями між процесорами. Процесори покладаються на програмне забезпечення для надсилання та отримання повідомлень. Програма дозволяє процесору передавати інформацію іншим процесорам. Обмінюючись повідомленнями, процесори можуть регулювати значення даних і залишатися в синхронізації один з одним. Це важливо, тому що як тільки всі процесори завершать свої завдання, процесор повинен зібрати всі окремі рішення в загальне рішення для оригінальної обчислювальної проблеми. Подумайте про це як про головоломку - якщо всі процесори залишаються синхронізованими, шматочки головоломки легко поєднуються. Якщо процесори не синхронізовані, частини головоломки можуть взагалі не вміститися разом.

Є два основних фактори, які можуть вплинути на продуктивність системи: затримка і пропускна здатність. Затримка відноситься до кількості часу, необхідного для передачі результатів процесору назад в систему. Це не добре, якщо процесору потрібно менше часу для запуску алгоритму, ніж для передачі отриманої інформації назад в загальну систему. У таких випадках

послідовна комп'ютерна система була б більш доречною. Пропускна здатність відноситься до того, скільки даних процесор може передавати за певний проміжок часу. Хороша паралельна система обробки буде мати як низьку затримку, так і високу пропускну здатність. Іноді паралельна обробка не швидше, ніж послідовні обчислення. Якщо процесору комп'ютера потрібно занадто багато часу, щоб зібрати всі окремі паралельні рішення процесора, кращий вибір може бути послідовним комп'ютером. Оскільки комп'ютерні вчені вдосконалюють методи паралельної обробки, а програмісти пишуть ефективне програмне забезпечення, це може стати меншою проблемою.

Існують різні архітектури паралельного обчислення такі як: багатоядерні обчислення, симетрична багатопроцесорна обробка, розподілені обчислення, масові паралельні обчислення.

Багатоядерний процесор - це комп'ютерний процесор інтегральної схеми з двома або більше окремими ядрами обробки, кожен з яких виконує програмні інструкції паралельно.

Симетрична багатопроцесорна багатопроцесорна комп'ютерна апаратна та програмна архітектура, в якій два або більше незалежних, однорідних процесорів управляються одним екземпляром операційної системи, який однаково ставиться до всіх процесорів, і підключається до єдиної, спільної основної пам'яті з повним доступом до всіх загальних ресурсів і пристроїв.

Розподілені компоненти системи розташовані на різних мережевих комп'ютерах, які координують свої дії, спілкуючись за допомогою чистих з'єднувачів HTTP, RPC та черг повідомлень.

Масово паралельні обчислення: відноситься до використання численних комп'ютерів або комп'ютерних процесорів для одночасного виконання набору обчислень паралельно.

В нашому випадку буде доречно використати архітектуру багатоядерного процесору.

Хоча використання паралельних обчислень піднімають складність написання програмних продуктів але взамін дають можливість значно підняти швидкість їхнього виконання.

Потенційне прискорення алгоритму при збільшенні числа процесорів задається законом Амдала, що вперше був сформульований Джином Амдалем у 1960-тих. [19]. Даний закон стверджує що якщо наявна частина програми яку неможливо розділити на паралельні процеси тоді ця частина буде обмежувати загальне прискорення програми. Адже великі задачі можуть містити в собі декілька підзадач що можуть виконуватись паралельно і кількох задач які можуть виконуватись лише послідовно одна за одною даний зв’язок задається рівнянням(2.4):

$$S = \frac{1}{1 - P}, \tag{2.4}$$

де S – це прискорення програми, P – частина програми що можна виконувати паралельно.

Прискорення при різних відсотках паралелізації процесів в програмному додатку продемонстровано на рисунку 2.5.

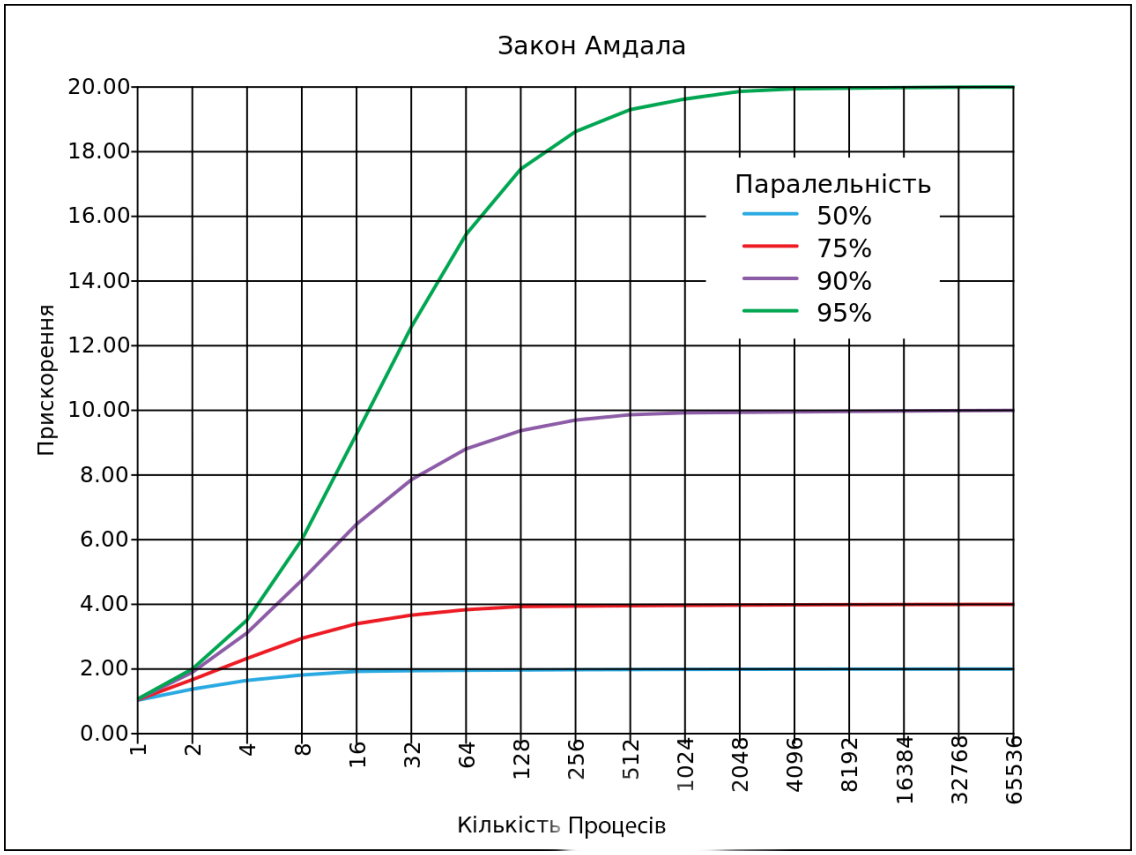


Рисунок 2.5 – Прискорення за законом Амдала

2.7 Висновки

1. Аналіз методів кодування зображень показав перспективність ущільнення зображень методом векторного квантування на основі двовимірної карти Кохонена, оскільки цей метод забезпечує достатній коефіцієнт ущільнення при прийнятній якості відновленого зображення.

2. Розроблено загальну схему ущільнення зображень, яка включає векторний кантувач на основі карти Кохонена та арифметичний кодер для ущільнення квантованих даних. Арифметичний кодер забезпечує генерацію найоптимальнішого коду серед відомих методів ущільнення без втрат.

3. Розроблено прототип інтерфейсу програмного продукту, який характеризується високим рівнем юзабельності і буде використаний у фінальній версії програмного продукту.

3 РОЗРОБКА ПРОГРАМИ ДЛЯ УЩІЛЬНЕННЯ ЗОБРАЖЕНЬ

3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програми

Для розробки будь-якого програмного продукту вибір мови програмування є дуже важливим, оскільки від цього залежить простота реалізації алгоритму роботи, а також ефективність і швидкість роботи програми. Для реалізації даного програмного засобу необхідно обрати таку мову програмування яка працює найшвидше, тому що розпаковані дані будуть декодуватись і чим більший об’єм даних необхідно декодувати тим довше програма буде запускатись.

Існує багато мов програмування, таких як C, C++, C# .Visual Basic, Basic, Pascal, Java і т.д., однак для вирішення завдань було C#.

Мова програмування C# є візуальною, тобто при розробці програмного забезпечення не потрібно малювати кнопки, панелі та інші об’єктів, звичні для нашого ока. Синтаксис C# близький до C++ і Java[20]. Мова має строгу статичну типізацію. Мова містить підтримку поліморфізму, перевантаження операторів, вказівників на функції-члени класів, атрибутів, подій, властивостей, винятків, коментарів у форматі XML. Перейнявши багато що від своїх попередників – мов C++, Object Pascal, Модула і Smalltalk – C#, спираючись на практику їхнього використання, виключає деякі моделі, які зарекомендували себе як проблематичні при розробці програмних систем, наприклад множинне спадкування класів присутнє в C++.

Unity – багатоплатформовий інструмент для розробки відеоігор і застосунків, і рушій, на якому вони працюють. Створені за допомогою Unity програми працюють на настільних комп'ютерних системах, мобільних пристроях та гральних консолях у дво- та тривимірній графіці, та на пристроях

віртуальної чи доповненої реальності. Застосунки, створені за допомогою Unity, підтримують DirectX та OpenGL[21].

Редактор Unity має інтерфейс, що складається з різних вікон, які можна розташувати на свій розсуд. Завдяки цьому можна проводити налагодження гри чи застосунка прямо в редакторі. Головні вікна – це оглядач ресурсів проєкту, інспектор поточного об'єкта, вікно попереднього перегляду, оглядач сцени та оглядач ієрархії ресурсів.

Проєкт в Unity складається з сцен, що є окремими файлами і містять свої набори об'єктів, сценаріїв та налаштувань. Сцени можуть містити в собі як об'єкти-моделі (ландшафт, персонажі, предмети довкілля тощо), так і порожні ігрові об'єкти без якогої-небуть моделі вигляду, проте задають поведінку інших об'єктів (тригери подій, точки збереження прогресу тощо). Ці невидимі об’єкти дозволяється переміщати, повертати, масштабувати та прикріпляти до них скрипти. Вони можуть містити назву (в Unity допускається наявність двох і більше об'єктів з однаковими назвами), також може бути тег (мітка) та шар, на якому об’єкт розміщений та буде відображатись. Так, будь-який предмет що розміщується на сцені обов'язково присутній компонент Transform – цей компонент відповідає за зберігання в собі координат місця розташування, куту повороту об’єкта і також містить ййого розміри по всіх трьох осях. У об'єктів з видимою геометрією також за умовчанням присутній компонент Mesh Renderer, що робить модель видимою. Різні моделі можуть об'єднуватися в набори (ассети) для швидкого доступу до них.

Скриптова система Unity зроблена на Mono – вільний відкритий проєкт з реалізації .NET Framework. Програмісти можуть використовувати UnityScript (власна скриптова мова, подібна до JavaScript та ECMAScript), C# або Boo (мова програмування, подібна до Python). Починаючи з версії 3.0, до Unity входить перероблена версія MonoDevelop для налагодження скриптів. З виходом версії 5.2 у 2015 році передбачена вбудована можливість редагувати скрипти у середовищі Microsoft Visual Studio.

Скрипти в юніті це окремі класи одного чи різних namespace-ів які зазвичай розподіляють на дві групи скрипти Core та UI. Перша група відповідає за будь які обчислення, збереження і передачу даних. Друга ж група відповідає за взаємодію та масштабування та вигляд елементів в меню і подальшу передачу керування до скриптів першої групи. Зазвичай «асети» які містять в собі скрипти мають свій окремий «namespace» підключивши який і створивши посилання на потрібний клас можна звертатись до його публічних полів та функцій. Така структура дозволяє вести досить швидку розробку продукту працюючи над ним по частинам функціоналу які можна одразу протестувати на коректність до завершення роботи над всім проектом, але це забирає можливості вносити зміни туди пізніше. Ці особливості дозволяють прцювати в команді і вести розробку одразу декількох видів функціоналу який не зв’язай чи слабко зв’язаний між собою без проблем сумісності.

При написані скриптів для юніті послуговуються принципа розробки «SOLID» тобто принципи:

- S - Single-responsiblity Principle – це значить що кожен скрипт має містити лише функції одного типі з єдиним призначенням.
- O - Open-closed Principle – це значить що класи мають мати поля і функції до яких можна доступитись поза межами цього класу так і ті які можуть викликатись лише в межах цього класу.
- L - Liskov Substitution Principle – це значить що кожен клас має наслідуватись від одного базового класу чи батьківського класу.
- I - Interface Segregation Principle – принцип що дозволяє реалізовувати схожі чи однакові функції в різних класах обов’язково з однаковою назвою і тимиж аргументами що приймаються і типом отримального результату.
- D - Dependency Inversion Principle – цей принцип відповідаю за відношення класів між собою, а саме як класи взаємодіють від низькорівневих до високорівневих.

Також при розробці часто використовують такі принципи як:

- DRY - DON’T REPEAT YOURSELF – цей принци означає що якщо якась частина яка буде використовуватись в декількох місцях то замість того що прописувати таку частину кожен раз варто винести такий код в окрему функцію і взамін викликати її.
- KISS - KEEP IT SIMPLE STUPID – цей принцип означає що не потрібно ускладнювати систему незрозумілим кодом якщо є простіший шлях розв’язання поставленої задачі.
- YAGNI - You aren't gonna need it – цей принцип означає що непотрібно вести розробку функціоналу чи модулів які не будуть використовуватись в програмному продукті.

Для розробки було обрано середовище Unity черз простоту в розробці інтерфейсу та через те що можливо використовувати мову програмування C# з усім його функіоналом а також з можливостями що надаються завдяки використання Unity. Також є можливість написання коду в інших середовищах розробки таких як MonoDevelop, Visual Studio, Visual Studio Code.

MonoDevelop – це відкрите інтегроване середовище розробки передусім націлене на розробку програм, які використовують як Mono так і Microsoft .NET framework[22]. MonoDevelop включає можливості подібні до NetBeans та Microsoft Visual Studio, такі як автоматичне доповнення, інтеграція контролю коду, графічний користувацький інтерфейс і веб-дизайнер. MonoDevelop містить в собі інтегрований Gtk# GUI дизайнер під назвою Stetic.

Microsoft Visual Studio – це серія продуктів фірми Майкрософт, які включають інтегроване середовище розробки програмного забезпечення та низку інших інструментальних засобів. Ці продукти дозволяють вести розробку консольних програм, програм з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, а також веб-сайтів, веб-застосунків, веб-служб як в рідному, так і в керованому кодах для всіх платформ, що

підтримуються Microsoft Windows, Windows Mobile, Windows Phone, Windows CE, .NET Framework, .NET Compact Framework та Microsoft Silverlight. [23]. Це середовище містить велику кількість вбудованих інструментів та бібліотек що значно полегшує розробку також є однією з найпопулярніших IDE.

Visual Studio Code – засіб для створення, редагування та зневадження сучасних веб-застосунків і програм для хмарних систем. Visual Studio Code розповсюджується безкоштовно і доступний у версіях для платформ Windows, Linux і OS X[24]. Visual Studio Code потребує встановлення плагінів для розширення функціоналу.

Отже, для розробки програмного забезпечення для кодування зображень методом SOFM буде використовуватись Unity та мова програмування C#, оскільки програмний продукт, створений на цій мові повністю задовольняє сучасним вимогам до швидкодії і якості інтерфейсу користувача. А середовищем було обрано Microsoft Visual Studio через його функціонал та простоту використання його конструктора форм і також відсутність потреби налаштування середовища перед початком роботи.

3.2 Розробка алгоритму роботи модуля векторного квантування на основі SOFM

Для розробки алгоритму необхідно вирішити декілька питань:

- Який розмір карти Кохонена вибрати.
- Яку вибрати розмірність вхідного вектора.
- Яка розрядність вагових коефіцієнтів карти Кохонена.

Розрядність вагових коефіцієнтів вибирається з урахуванням розрядності вхідних даних. Оскільки зображення звичайно представляються в форматі 8x8x8, тобто складові кольорів RGB представляються одним байтом, то і розрядність вагових коефіцієнтів вибираємо рівною 8-и бітам.

Розмірність вхідних векторів вибирається з урахуванням кореляції вхідних даних. Відомо, що найбільшими кореляційними зв'язками характеризуються сусідні відліки зображення, які можуть мати близькі значення [1]. Тому зображення розбивається на примикаючі квадрати розміром 2x2, кожний з яких розглядається як вектор в 4-вимірному просторі.

Розмір карти Кохонена визначається мінімальною кількістю розрядів, яка представляє елемент зображення і забезпечує достатню 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. До того ж збільшення розміру карти призводить до значної втрати швидкодії навчання мережі. Швидкість навчання прямо пропорційна квадрату розміру сторони карти , тобто(3.1):

$$Тн = kN2 , \quad (3.1)$$

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

З урахуванням вище сказаного розмір карти Кохонена, який задовольняє цим суперечливим вимогам, вибираємо 16x16. Оскільки розмір вхідного вектора рівний чотирьом, така карта забезпечує при векторному квантуванні таку кількість біт на елемент зображення(3.2):

$$M = \log2\ N2\ /n = 8/4 = 2 \text{ біти/ел.} \quad (3.2)$$

Що є прийнятним як з точки зору швидкодії, так і з точки зору забезпечення високої якості відновленого зображення.

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

– навчання мережі;

– векторне квантування.

Причому навчальними векторами є всі фрагменти зображення з розмірами 2х2.

Таким чином алгоритм кодування буде таким:

1. етап навчання;
2. ініціалізація коефіцієнтів вагових нейронів випадковими значеннями.
3. вибір з зображення першого фрагменту 2х2
4. представити обраного фрагменту у вигляді навчального вектора;
5. для кожного кластерного елемента карти обчислити відстань до навчального вектора(3.3):

$$d_j = \sum_{i=0}^{3} (w_{ij} - x_i)^2 \qquad (3.3)$$

6. пошук кластерного елементу j для якого dj мінімально;
7. для знайденого кластерного елементу оновити вагові коефіцієнти згідно формули(3.4):

$$w_{ij}(n+1) = w_{ij}(n) + \eta(n)[x_i - w_{ij}(n)], \qquad (3.4)$$

де η - норма навчання, xi – координата навчального вектора;

8. Обновити норму навчання η і обрати з зображення наступний фрагмент 2х2 і повторювати пункти 3-6 для наступних навчальних векторів до тих пір поки не будуть вибрані всі фрагменти.

Векторне квантування включає такі етапи:

1. Вибір з зображення першого фрагменту 2х2.
2. Представлення обраного фрагменту у вигляді навчального вектора.

3. Обчислення для кожного кластерного елемента карти відстані до навчального вектора(3.5):

$$d_j = \sum_{i=0}^{3} (w_{ij} - x_i)^2 \tag{3.5}$$

4. Записування в вихідний файл номеру кластерного елемента з мінімальним dj.
5. Вибір з зображення наступного фрагменту 2x2 і повторення пунктів 8-11 до тих пір поки не будуть вибрані всі фрагменти.
6. Записування в вихідний файл значень коефіцієнтів wij – всього 1024 байти.
7. Записування в вихідний файл розміру початкового файлу.
8. Ущільнення отриманого файлу методом арифметичного кодування.

Декодування виконується значно швидше і включає такі етапи:

1. Декодувати ущільнений файл арифметичним методом.
2. Прочитати з вхідного файлу і записати в масив значення номерів кластерних елементів, коефіцієнтів wij і розмір початкового зображення.
3. Вибрати з масиву номер кластерного елементу для першого фрагменту 2x2.
4. Коефіцієнти wij даного кластерного елемента (4 коефіцієнти) записати в вихідне зображення як значення елементів відповідного фрагменту 2x2.
5. Вибрати наступний номер кластерного елементу і повторяти пункти 4-5 до тих пір поки не будуть відновлені всі фрагменти зображення.

3.3 Розробка програмних модулів векторного квантування на основі SOFM

Для реалізації указаного алгоритму використовуються такі візуальні компоненти:

- Компонент для виведення повідомлень оператору про хід виконання завдання.
- Два візуальні компоненти PictureBox для демонстрації зображень в вікні програми як відновленого після кодування так і початкового зображення.
- Візуальний компонент MenuStrip для створення меню додатку.
- Компонент для завантаження початкового зображення в форматах, що підтримуються.
- Компонент для збереження зображення в форматах, що підтримуються.
- Компонент для збереження зображення у власному форматі.
- Компонент для завантаження зображення у власному форматі.

Даний алгоритм реалізує такі процедури:

1. Процес векторного квантування з використанням мережі SOFM реалізує. Для цього використовується функція, яка в залежності від значення змінної виконує як корекцію вагових коефіцієнтів при навчанні так і простий пошук найближчого кластерного елемента при квантуванні.

2. Деквантування виконує, яка реалізує відповідні пункти алгоритму.

3. завантажує вхідне зображення в компоненти Bitmap, а виконує запис ущільненого файлу на диск.

4. призначена для завантаження ущільненого файлу. В даній процедурі релізовано ряд функцій необхідних для декодування ущільненого файлу.

Крім перерахованих процедур програма реалізує ряд допоміжних функцій, необхідних при дослідженні зображення: обчислює оцінку середньо-

квадратичного відхилення відновленого після кодування і початкового зображення.

Структура програми має вигляд (рис. 3.1).

Рисунок 3.1 – Структура програми

На рисунку 3.2 показано функцію яка відповідає за обробку зображення «ProcessImage».

```
public void ProcessTheImage(DataContainer data, Texture2D image)
{
    NuralNetwork nn = new NuralNetwork();
    data.Withdraw = image1.width - 1;
    data.Height = image1.height - 1;
    data.W8 = data.Withdraw / 8;
    data.H8 = data.Height / 8;
    data.W4 = data.Withdraw / 4;
    data.H4 = data.Height / 4;
    data.W2 = data.Withdraw / 2;
    data.H2 = data.Height / 2;
    image1 = image;
    nn.InitNuralNetwork(data, image);
    X08 = nn.CalculateImage(data);
    DecodeImage(data, nn.GetWeight());
}
```

Рисунок 3.2 – Функція обробки зображень

В даній функції проводиться спочатку ініціалізація нейроної мережі початковими заначення і її навчання на відкритому зображенні і в подальшому після обробки зображення навченою мережею проводиться декодування для демонстрації результату обробки алгоритму.

Фунція яка відповідає за навчання мережі має наступний вигляд рисунок 3.3. На вхід подається контейнер для даних і безпосередньо зображення і потім дана функція обробляє зображення по блокам і обновлює вагові коефіцієнти. Навчання триває доти поки значення коефіцієнів залишаються майже не змінними. Після чого навчання припиняється.

```
private int[] winer(float[] localsaple, int index1, int index2)
{
    float D0 = 0;
    float D1 = 0;
    float D2 = 0;
    float D3 = 0;
    int[] res;

    for (int i = 0; i < localsaple.Length; ++i)
    {
        D0 = Mathf.Pow((localsaple[i] - weights[index1][index2][i]), 2);
        D1 = Mathf.Pow((localsaple[i] - weights[index1][index2 + 1][i]), 2);
        D2 = Mathf.Pow((localsaple[i] - weights[index1 + 1][index2][i]), 2);
        D3 = Mathf.Pow((localsaple[i] - weights[index1 + 1][index2 + 1][i]), 2);
    }


    if (D0 > D1 && D0 > D2 && D0 > D3)
    {
        res = new int[] { index1, index2 };
        return res;
    }

    else if (D1 > D0 && D1 > D2 && D1 > D3)
    {
        res = new int[] { index1, index2 + 1 };
        return res;
    }

    else if (D2 > D0 && D2 > D1 && D2 > D3)
    {
        res = new int[] { index1 + 1, index2 };
        return res;
    }

    else
    {
        res = new int[] { index1 + 1, index2 + 1 };
        return res;
    }
}

private void updatew(int J, int I, float[] localsaple)
{
    for (int i = 0; i < 3; ++i)
        weights[J][I][i] = weights[J][I][i] + alfa * (localsaple[i] - weights[J][I][i]);
}
```

Рисунок 3.3 – Функція навчання мережі

За оновлення вагових векторів при навчані відповідає функція “Calculate” вона має наступний вигляд:

```
public byte Calculate(float x00, float x01, float x10, float x11, bool ln)
        {
            double d;
            byte ci = 0, cj = 0;
            double dmin = 2147483647;

            //Пошук нейрона с dmin
            for (int j = 0; j < 15; ++j)
                for (int i = 0; i < 15; ++i)
                {
                    d = Mathf.Sqrt(x00 - w1[2 * i, 2 * j]) + Mathf.Sqrt(x01 - w1[2 * i + 1, 2
* j])
                        + Mathf.Sqrt(x10 - w1[2 * i, 2 * j + 1])
                        + Mathf.Sqrt(x11 - w1[2 * i + 1, 2 * j + 1]);

                    if (d < dmin)
                    {
                        dmin = d;
                        ci = (byte)i;
                        cj = (byte)j;
                    }

                    d = 0;
                }

            if (ln)
            {
                //Обновлення вагових векторів
```

```
            w1[2 * ci, 2 * cj] = w1[2 * ci, 2 * cj] + (lsp[ci, cj] * (x00 - w1[2 * ci, 2 * cj]));
            w1[2 * ci + 1, 2 * cj] = w1[2 * ci + 1, 2 * cj] + (lsp[ci, cj] * (x01 - w1[2 * ci + 1, 2 * cj]));
            w1[2 * ci, 2 * cj + 1] = w1[2 * ci, 2 * cj + 1] + (lsp[ci, cj] * (x10 - w1[2 * ci, 2 * cj + 1]));
            w1[2 * ci + 1, 2 * cj + 1] = w1[2 * ci + 1, 2 * cj + 1] + (lsp[ci, cj] * (x11 - w1[2 * ci + 1, 2 * cj + 1]));

            if (lsp[ci, cj] > 0.01) lsp[ci, cj] = lsp[ci, cj] - 0.01f;

            return 0;
        }

        return (byte)((ci * Mathf.Pow(2,4)) + cj);
    }
```

Цей клас приймає значення кольорів пікселів обраного блоку і прапорець чи ці дані для обробки чи навчання. Якщо дані для навчання то проводиться навчання і в результаті обновлюються вагові вектори. Якщо ж даня для обробки то вони просто оброблюються використовую чи вектори що вже є в системі.

3.4 Розробка паралельних обчислень

Задля підвищення швидкодії, а саме, під час навчання було вирішено обчислювати блоки зображень паралельно один з одним для цього буде використано метод паралельних обчислень.

Зображення буде поділене на чотири рівні частини і буде запущено алгоритм навчання для всіх частин одночасно це дозволить скоротити час

навчання і в результаті час необхідний для обробки зображення в 3 рази порівнянні з звичайним послідовним навчанням.

Такий метод дозволить обновлювати вагові вектори без ризику конфліктів при перезаписі вагових векторів, на відміну, якщо б ми обробляли повністю все зображення просто в декількох потоках, оскільки такий підхід може викликати збій програми або запис некоректних даних, коли декілька потоків спробують оновити одні й ті ж самі комірки пам’яті при оновлені вагових векторів.

3.5 Висновки

У третьому розділі обґрунтовано вибір мови програмування, яка буде використовуватися при розробці програмного продукту та наведено основні її переваги. Розроблені алгоритм роботи модуля векторного квантування на основі двовимірної карти Кохонена та код модуля векторного квантування на основі двовимірної карти Кохонена.

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Тестування програмного забезпечення

Тестування програмного забезпечення – це процес технічного дослідження, призначений для виявлення інформації про якість продукту відносно контексту, в якому він має використовуватись [25]. Процес тестування включає в себе пошук помилок чи інших дефектів та випробування програмних складових з метою їх оцінки.

Існують різні типи тестування такі як:

- тестування доступності – це практика забезпечення роботи ваших мобільних та веб-додатків та використання для користувачів без інвалідності та з обмеженими можливостями, такими як порушення зору, порушення слуху та інші фізичні або когнітивні умови;
- приймальне тестування гарантує, що кінцевий користувач (клієнти) може досягти цілей, встановлених в бізнес-вимогах, що визначає, чи є програмне забезпечення прийнятним для доставки чи ні. Він також відомий як тестування прийому користувачів (UAT);
- тестування чорного ящика передбачає тестування системи, де код і шляхи невидимі;
- завершення тестування – це метод, який перевіряє робочий процес програми від початку до кінця, щоб переконатися, що все функціонує належним чином;
- функціональне тестування перевіряє додаток, веб-сайт або систему, щоб переконатися, що він робить саме те, що він повинен робити;

– інтерактивне тестування також відоме як ручне тестування, інтерактивне тестування дозволяє тестувальникам створювати і полегшувати ручні тести для тих, хто не використовує автоматизацію і збирає результати зовнішніх тестів;
– інтеграційне тестування гарантує, що ціла інтегрована система відповідає набору вимог. Він виконується в інтегрованому апаратному та програмному середовищі для забезпечення належної роботи всієї системи;
– тестування навантаження цей тип нефункціонального процесу тестування програмного забезпечення визначає, як поводиться програмне забезпечення під час доступу до декількох користувачів одночасно;
– нефункціональне тестування перевіряє готовність системи за нефункціональними параметрами (продуктивність, доступність, UX і т.д.) які ніколи не розглядаються функціональним тестуванням;
– тестування продуктивності перевіряє швидкість, стабільність, надійність, масштабованість та використання ресурсів програмного забезпечення під заданим робочим навантаженням;
– регресійне тестування проводиться, щоб визначити, чи порушують модифікації коду програму або споживають ресурси;
– тестування розсудливості виконується після виправлення помилок, тестування розсудливості визначає, що помилки виправлені, і що до цих змін не вводяться подальші проблеми.;
– тестування безпеки розкриває вразливості системи, щоб гарантувати, що програмна система та програма вільні від будь-яких загроз або ризиків. Ці тести спрямовані на виявлення будь-яких потенційних недоліків і недоліків у програмній системі, які

можуть призвести до втрати даних, доходу або репутації на одного співробітника або за межами компанії;

– тестування продуктивності одного користувача перевіряє, що програма, що тестується, виконує штраф відповідно до вказаного порогу без будь-якого завантаження системи. Цей орієнтир може бути використаний для визначення реалістичного порогу, коли система знаходиться під навантаженням;
– тестування диму даний вид тестування програмного забезпечення перевіряє стабільність програмного забезпечення, він виконується на початковій збірці програмного забезпечення для забезпечення роботи критичних функцій програми;
– стрес-тестування – це діяльність тестування програмного забезпечення, яка перевіряє понад нормальну оперативну здатність для перевірки результатів;
– модульне тестування – це процес перевірки невеликих фрагментів коду, щоб переконатися, що окремі частини програми працюють належним чином самостійно, прискорюючи стратегії тестування та зменшуючи даремно витрачені тести.;
– тестування білої коробки включає тестування базової структури, архітектури та коду продукту для перевірки потоку вводу-виводу та підвищення дизайну, зручності використання та безпеки.

Багато з цих видів тестування можуть бути зроблені вручну - або вони можуть бути автоматизовані.

Програмний продукт може бути оцінений за такими критеріями як:

– відповідність до вимог, якими керувалися проектувальники та розробники;
– правильністю відповідей для усіх можливих вхідних даних даний вид тестування програмного забезпечення перевіряє стабільність програмного забезпечення, він виконується на початковій збірці

програмного забезпечення для забезпечення роботи критичних функцій програми;

- виконанням функцій за прийнятний час;
- практичність;
- сумісність з різним програмним забезпеченням та іншими операційними системами;
- відповідністю що до задач замовника.

На сьогодні тестування програмного забезпечення – один з найбільш дорогих етапів життєвого циклу програмного забезпечення, на нього відводиться від 50% до 65% загальних витрат[25].

Існує 3 види проводження тестування програмного продукту:

1. Ручне тестування є найбільш практичним типом тестування і використовується кожною командою в якийсь момент. Звичайно, в сучасному швидко розвивається життєвому циклі розробки програмного забезпечення ручне тестування важко масштабувати.
2. Автоматизоване тестування використовує тестові скрипти та спеціалізовані інструменти для автоматизації процесу тестування програмного забезпечення.
3. Безперервне тестування йде ще далі, застосовуючи принципи автоматизованого тестування масштабованим, безперервним способом для досягнення найнадійнішого тестового покриття для підприємства.

Файловий склад програми:

1. ImageCompress.exe
2. UnityPlayer.dll
3. Папка ImageCopmress_Data
4. Папка MonoBleedingEdge

Вимоги до графічних файлів. Графічний файл може вміщувати як чорно-біле, так і кольорове зображення в форматі до 24bit.

Файли формату .atc представляють собою закодовані та стиснені із використанням запропонованого методу зображення. При ущільненні використовувався алгоритм стиснення арифметичного кодування, який є безкоштовним та забезпечує найкращі результати при ущільненні закодованих даних.

Для роботи програми необхідно:

- комп’ютер з процесором класу Intel I3;
- кольоровий монітор SVGA;
- щонайменше 1GB оперативної пам’яті;
- на комп’ютері повинна бути встановлена операційна система Windows (XP/Vista/7/8/8.1,10).

Даний програмний продукт досить простий у використанні, тому працювати з ним можуть користувачі ЕОМ навіть без спеціальної підготовки.

Тестування програмного продукту показало повну відповідність поставленому технічному завданню.

При проведенні експериментальних досліджень вхідне зображення представлене у форматі

Якість відновленого зображення оцінювалась методом експертних оцінок за шкалою, наведеною в табл. 4.1.

Таблиця 4.1 – Шкала оцінювання якості відновлюваного зображення

Якість	Погіршення
Відмінна	Непомітно
Добра	Помітно, але не заважає
Задовільна	Трохи заважає
Погана	Заважає
Дуже погана	Дуже заважає

А також виконувався розрахунок середньоквадратичного відхилення (СКВ) відповідно з виразом(4.1):

$$\sigma = \frac{1}{M * N} \sqrt{(\mathrm{X}(\mathrm{m}_1, m_2) - \bar{X}(m_1, m_2))^2} \ , \qquad (4.1)$$

де $\mathrm{X}(\mathrm{m}_1,\mathrm{m}_2)$ – значення відліку початкового зображення, а $\bar{\mathrm{X}}(\mathrm{m}_1,\mathrm{m}_2)$ – значення відліку відновленого зображення.

Проведено порівняльні дослідження залежності коефіцієнту ущільнення та якості відновленого зображення від характеристик карти Кохонена, а саме:

- залежність коефіцієнта ущільнення від розміру карти (об’єму кодової книги);
- залежність середньоквадратичного відхилення від розміру карти (об’єму кодової книги).

Результати роботи алгоритму для файлу.BMP при 4-вимірних вхідних векторах і розмірах карти від 8x8 до 16x16 представлені в табл.4.2, а також на рис. 4.1.

Аналіз приведених результатів показує, що векторне квантування забезпечує достатньо високі характеристики. Цілий ряд експериментів з

різними типами зображень показав, що коефіцієнти ущільнення можуть знаходитися в межах 10 – 30. Для деяких зображень коефіцієнт ущільнення перевершує стандарт JPEG при тій же якості зображення.

Таблиця 4.2 – Експериментальні дані по файлам

Метод ущільнення	Розмір початкового файлу, байт	Розмір стиснутого файлу, байт	Коефіці-єнт ущільнення	Середньо-квадратична помилка	Візуальна оцінка якості
JPEG	192 054	13106	14,7	0,019	Відмінна
Карта Кохонена (16x16)	192054	11740	16,4	0,02	Відмінна
Карта Кохонена (14x14)	192 054	10979	17,5	0,023	добра
Карта Кохонена (11x11)	192054	9245	20,8	0,026	Добра
Карта Кохонена (8x8)	192054	6954	27,6	0,03	Задовільна

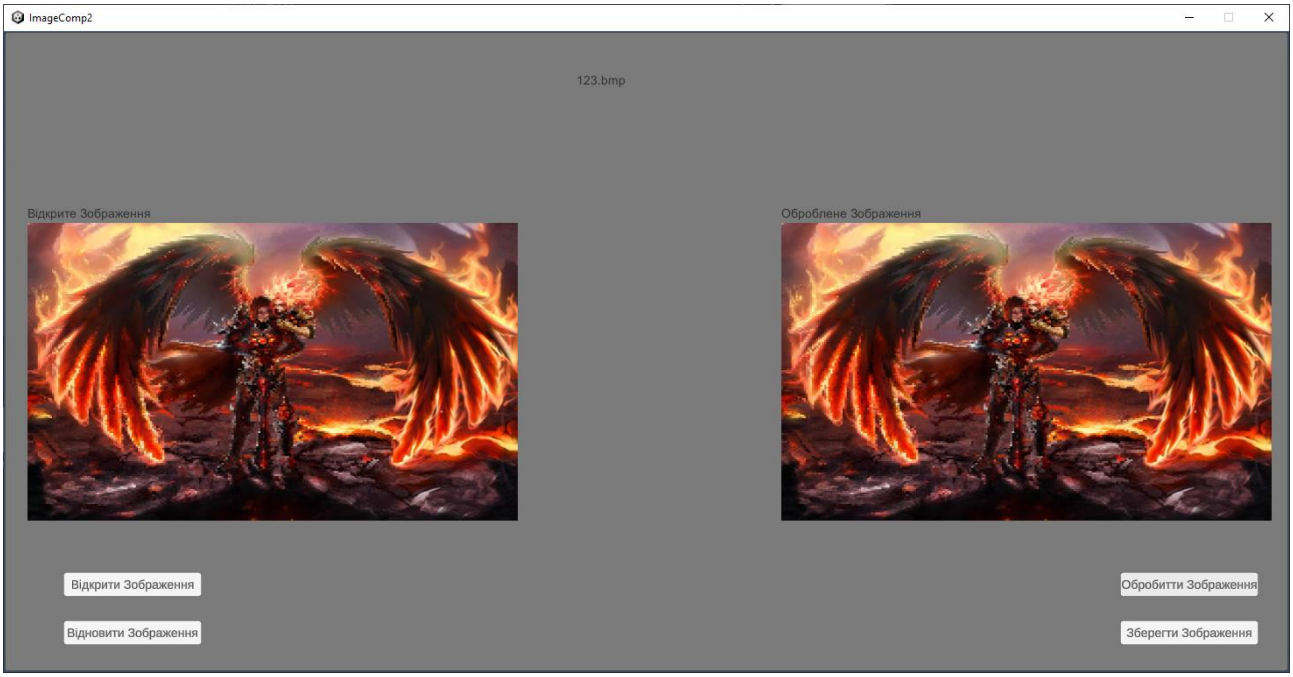


Рисунок 4.1 – Зображення після кодування

Результати порівняння швидкодії при послідовному та при паралельному процесі виконання алгоритму показало невелику різниця при маленькому розширені зображень, але при збільшенні розміру зображення різниця стає все більш помітною і досягає різниці при опрацюванні в 3 рази на великих зображеннях з високим розширенням.

Система повинна працювати на IBM-сумісних персональних комп’ютерах. У таблиці 4.3 та 4.4 наведено мінімальну та рекомендовану конфігурацію персонального комп’ютера для запуску та функціонування програми.

Таблиця 4.3 – Мінімальна конфігурація:

Тип процесора	32-розрядний (x86) або 64-розрядний (x64) процесор з тактовою частотою 1 ГГц
Об’єм оперативної пам’яті	2 ГБ для 32-розрядної системи і 3 ГБ для 64-разрядної системи
Розмір жорсткого диску	17 ГБ для 32-розрядної системи і 21 ГБ для 64-розрядної системи
Графічний пристрій	графічний пристрій DirectX11

Таблиця 4.4 – Рекомендована конфігурація:

Тип процесора	32-розрядний (x86) або 64-розрядний (x64) процесор з тактовою частотою 2 ГГц
Об’єм оперативної пам’яті	4 ГБ для 32-розрядної системи і 6 ГБ для 64-разрядної системи
Розмір жорсткого диску	20 ГБ для 32-розрядної системи і 25 ГБ для 64-розрядної системи
Графічний пристрій	графічний пристрій DirectX11

Програма повинна працювати під управлінням сімейства операційних систем Windows (7(SP1+)/8/8.1,10).

4.2 Розробка інструкції користувача

Посібник користувача повинен бути написаний для аудиторії - тих, хто буде купувати ваш продукт або послугу і читати керівництво користувача. Аналіз аудиторії розповість вам, хто буде вашою основною або цільовою аудиторією, і буде керувати вашим письмом. Перед розробкою інструкції користувача слід поговорити з людьми, які будуть використовувати ваш

пристрій. Запропонуйте тестовим користувачам прототипи пристрою та чернетку посібника користувача в контрольованих умовах. Запитуйте відгуки цих тестових користувачів про речі, які не є очевидними або заплутаними в вказівках користувачів, і включіть зміни в посібник користувача на основі цього відгуку.

Для початку роботи з програмою необхідно її розархівувати помістивши всі файли з архіву в одну папку після чого можна починати роботу.

Після цього користувач може запускати використавши файл «ImageCompress.exe» і почати працювати з програмним продуктом.

Для того щоб завантажити зображення в додаток необхідно натиснути на кнопку «Відкрити Зображення» (рисунок 4.2).

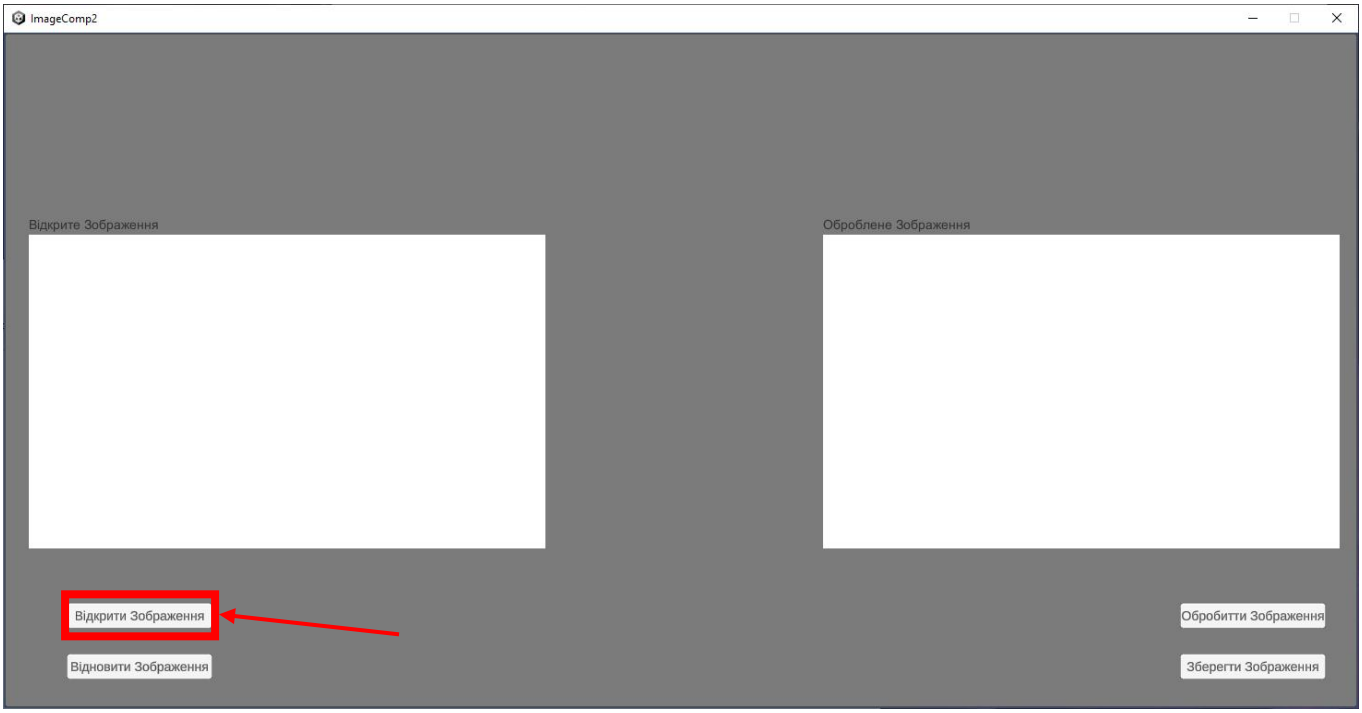


Рисунок 4.2 – Відкрити зображення

Після чого з’явиться діалогове вікно в якому необхідно обрати зображення з розширенням .bmp і натиснути «Завантажити» (рисунок 4.3).

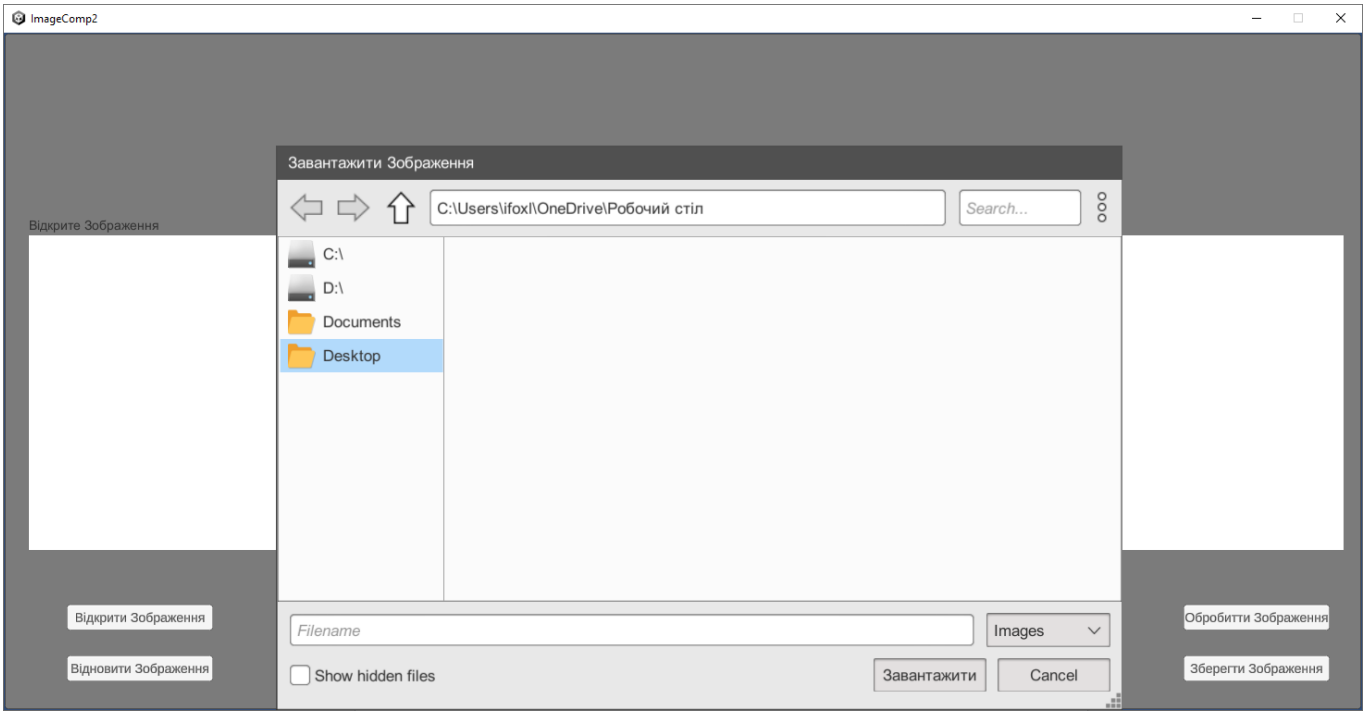


Рисунок 4.3 – вікно вибору зображення

Після чого обране зображення зявиться в лівому слоті програми (рисунок 4.4).

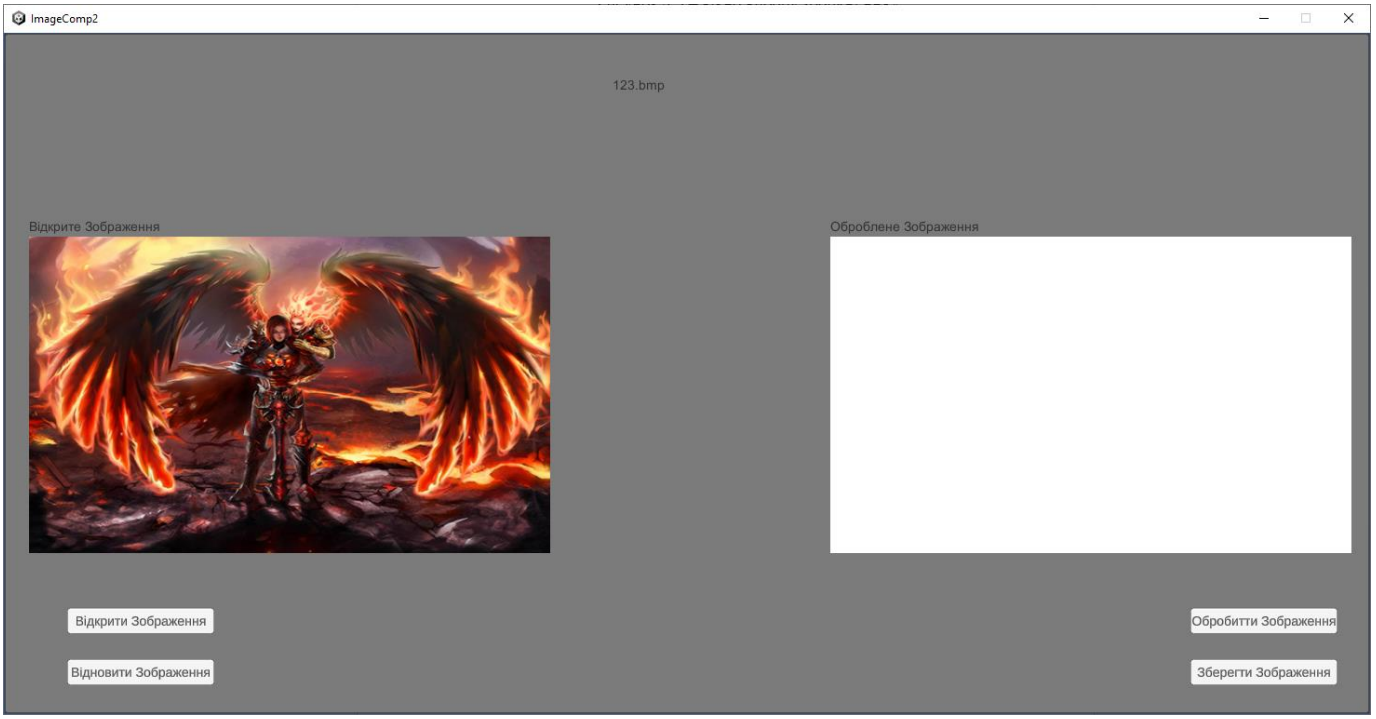


Рисунок 4.3 – вигляд відкритого зображення в додатку

При натисненні на кнопку «Обробити Зображення» зображення буде опрацьоване за алгоритмом і відновлена його версія буде продемонстрована в правому слоті програми (рисунок 4.5).

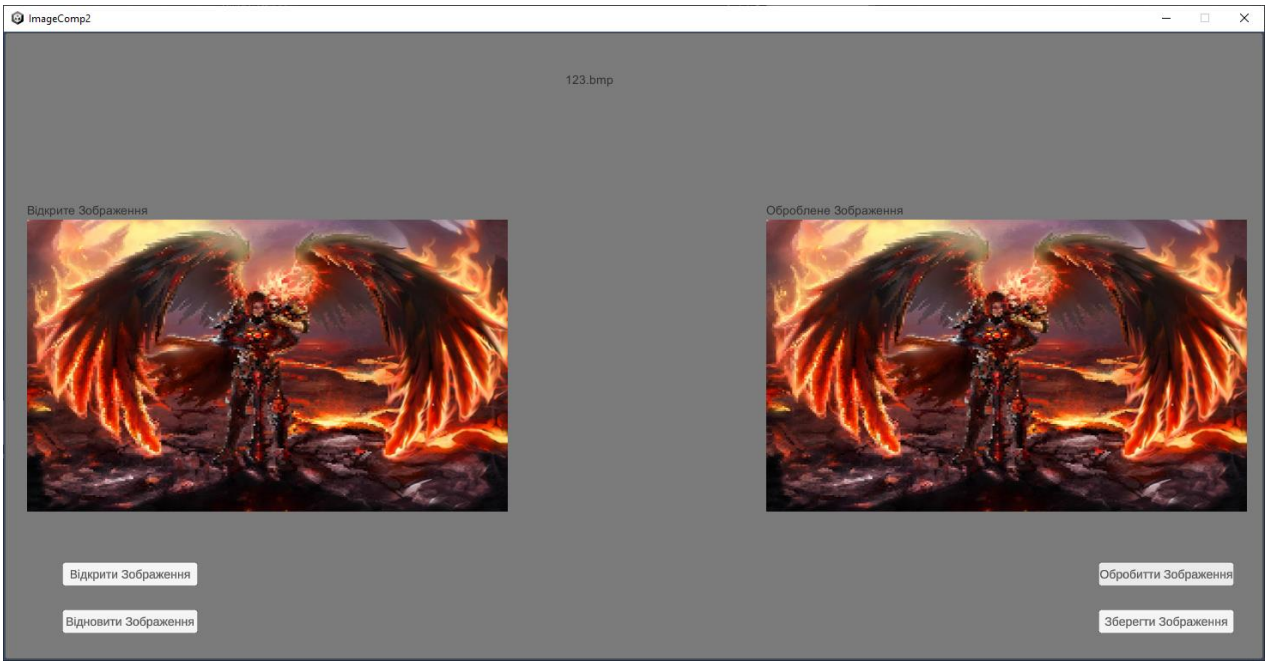


Рисунок 4.5 – вигляд обробленого зображення в додатку

Для збереження обробленого зображення необхіндо натиснути кнопку «Зберегти Зображення» (рисунок 4.6).

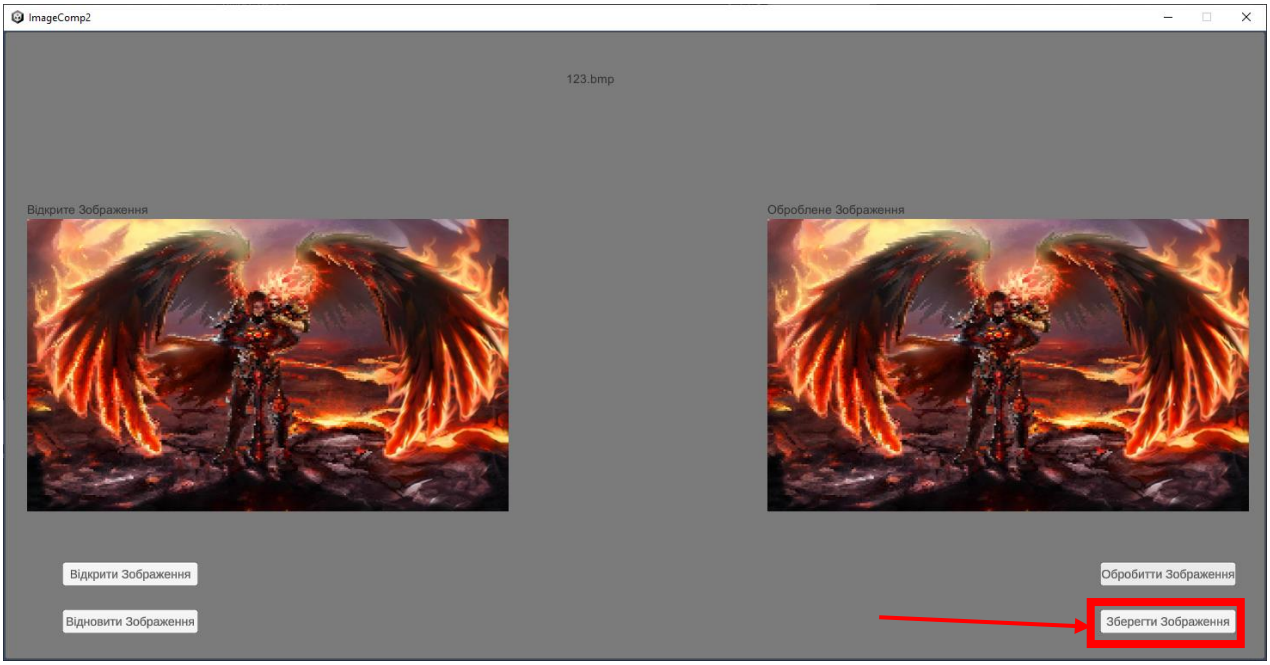


Рисунок 4.6 – Зберегти зображення

Після чого відкриється вікно для збереження зображення, де необхідно обрати місце для зберігання та ввести бажану назву файлу та натиснути «Зберегти» (рисунок 4.7).

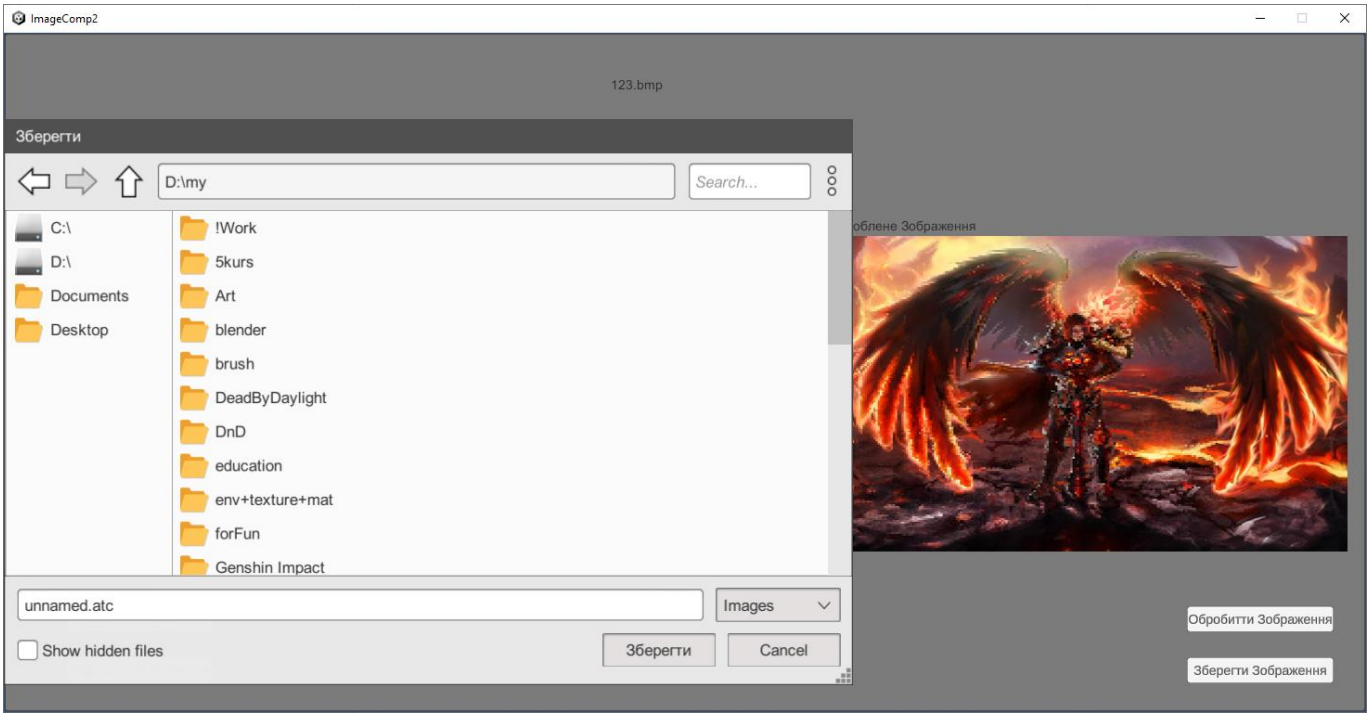


Рисунок 4.7 – вигляд вікна для зберігання зображення

Для завантаження збереженого обробленого зображення необхідно натиснути кнопку «Відновити Зображення» (рисунок 4.8).

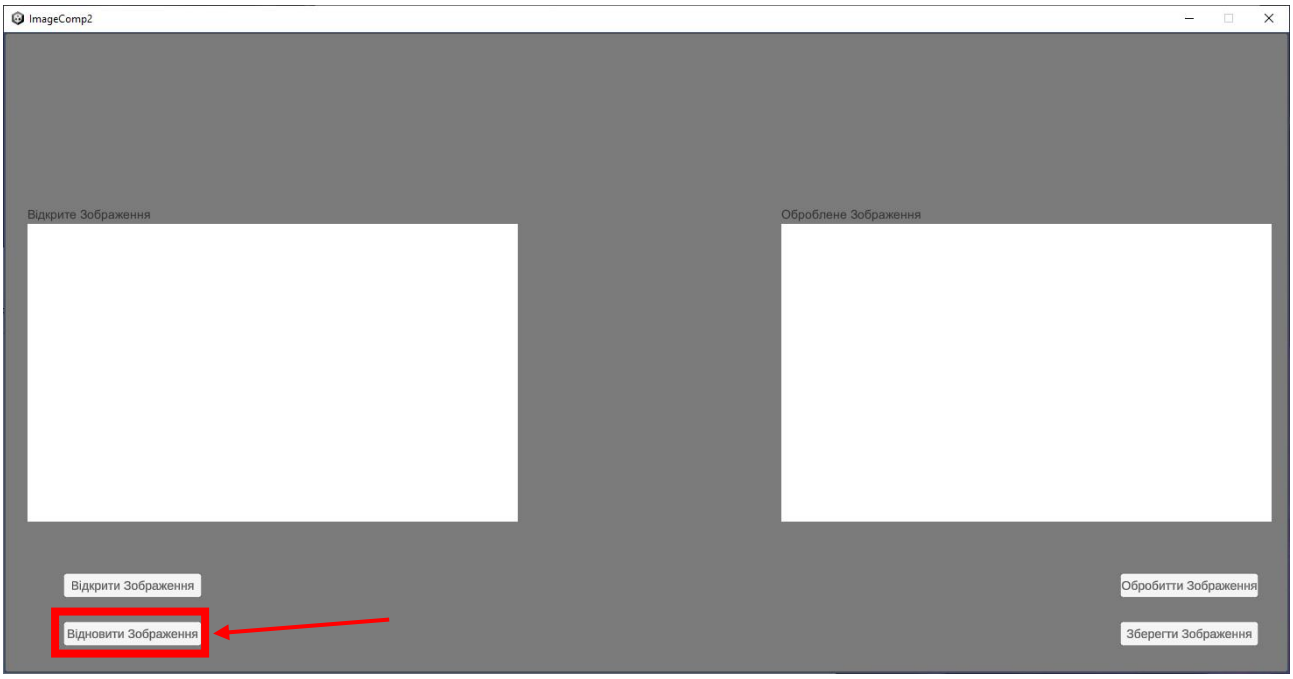


Рисунок 4.8 – Відновити зображення

Після чого відкриється вікно де необхідно вибрати збережений файл і натиснути кнопку «Завантажити» (рисунок 4.9).

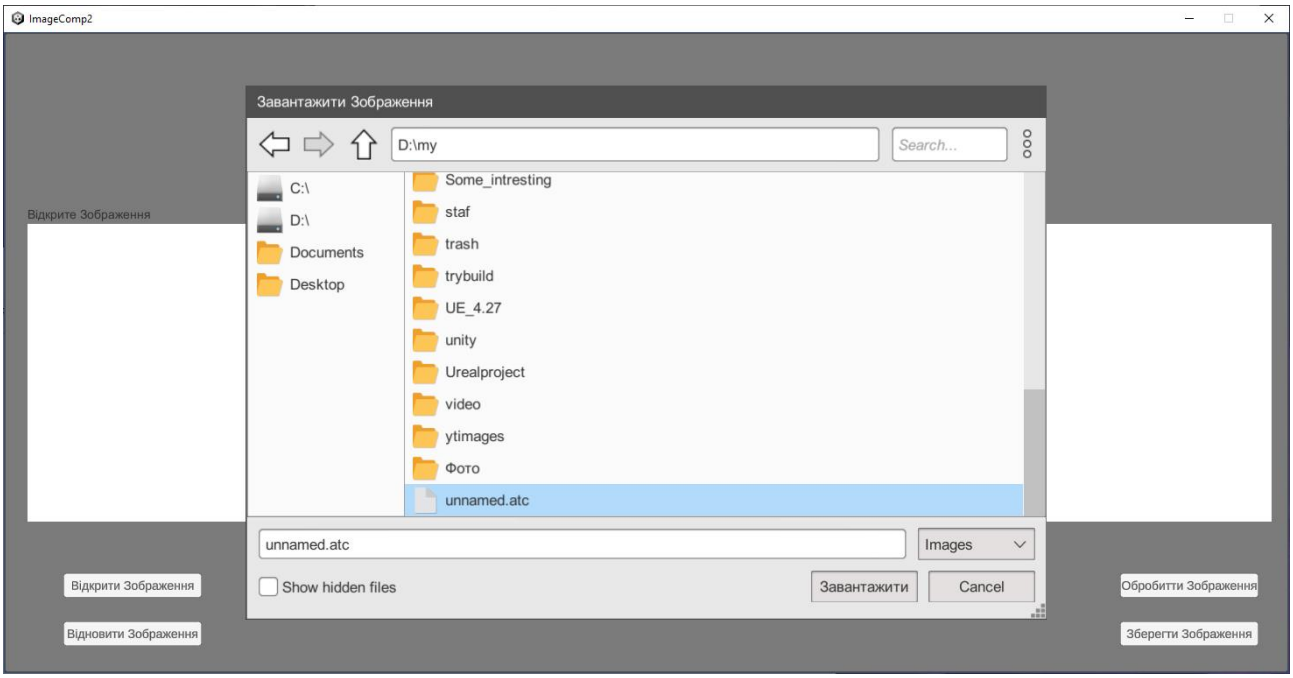


Рисунок 4.9 – вигляд вікна для відкритя відновлюваного файлу

Після відновлення зображення буде показано в лівому слоті додатку (рисунок 4.10).

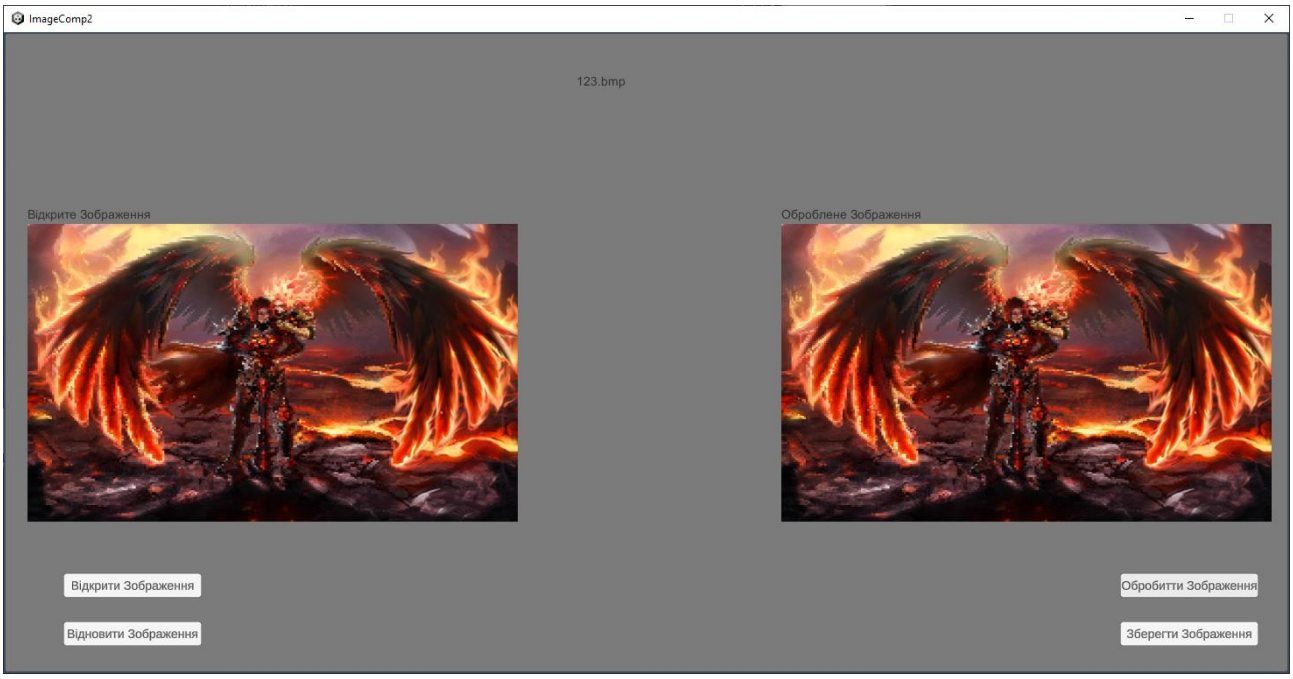


Рисунок 4.10 – вигляд датку з відновленим зображенням

4.3Висновки

Тестування програми показало повну її працездатність та відповідність поставленому технічному завданню. Розроблено інструкцію користувача по встановленню програмного продукту.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Оцінювання комерційного потенціалу розробки

Метою проведення комерційного та технологічного аудиту є оцінювання комерційного потенціалу впровадження програмних засобів ущільнення зображень на основі карти Кохонена.

Для проведення технологічного аудиту було залучено 3-х незалежних експертів Вінницького національного технічного університету кафедри програмного забезпечення: к.т.н., доцент Ракитянська Г. Б., к.т.н., доцент Черноволик Г. О., к.т.н., доцент Катєльніков Д. І. Для проведення технологічного аудиту було використано таблицю 5.1 [26] в якій за п’ятибальною шкалою використовуючи 12 критеріїв здійснено оцінку комерційного потенціалу.

Таблиця 5.1 – Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	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	Експлуатаційні витрати дещо	Експлуатаційні витрати на рівні	Експлуатаційні витрати трохи	Експлуатаційні витрати значно

	вищі, ніж в аналогів	вищі, ніж в аналогів	експлуатаційних витрат аналогів	нижчі, ніж в аналогів	нижчі, ніж в аналогів

Продовження табл. 5.1

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 пові-домлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Таблиця 5.2 – Рівні комерційного потенціалу розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0-10	Низький
11-20	Нижче середнього
21-30	Середній
31-40	Вище середнього
41-48	Високий

В таблиці 5.3 наведено результати оцінювання експертами комерційного потенціалу розробки.

Таблиця 5.3 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта		
	Ракитянська Г. Б.	Черноволик Г.О.	Катєльніков Д.І.
	Бали, виставлені експертами:		
1	2	2	2
2	2	2	2
3	2	2	3
4	2	2	2
5	1	2	2
6	2	2	3
7	2	2	2
8	2	2	2
9	2	2	2
10	2	2	2
11	2	3	3
12	2	2	2
Сума балів	$СБ_1 = 23$	$СБ_2 = 25$	$СБ_3 = 27$
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{3} = \frac{23+25+27}{3} = \frac{75}{3} = 25,00.$		

Середньоарифметична сума балів, розрахована на основі висновків експертів склала 25 балів, що згідно таблиці 5.2 вважається, що рівень комерційного потенціалу проведених досліджень є середнім.

Порівняємо нову розробку з аналогами, які є на ринку. В якості аналога для розробки було обрано фрактальне ущільнення зображень. Основними недоліками аналога є велика кількість арифметичних операцій. Для виконання фрактального ущільнення необхідна певна кількість операцій множення і ділення. У розробці дана проблема вирішується використанням карти Кохонена що має меншу кількість арифметичних операцій. Також система випереджає аналог за такими параметрами як якість відновлюваного зображення та швидкодія.

Проведемо оцінку якості і конкурентоспроможності нової розробки порівняно з аналогом. В таблиці 5.4 наведені основні техніко-економічні показники аналога і нової розробки.

Таблиця 5.4 – Основні параметри нової розробки та товару-конкурента

Показник	Варіанти		Відносний показник якості	Коефіцієнт вагомості параметра
	Базовий (товар-конкурент)	Новий (інноваційне рішення)		
1	2	3	4	5
Якість відновленого зображення, %	70	85	1.21	35 %
Коефіцієнт ущільнення	16	16	1	35 %
Швидкодія, мс	7000	3000	2,3	30 %

Проведемо оцінку якості продукції, яка є найефективнішим засобом забезпечення вимог споживачів та порівняємо її з аналогом.

Визначимо відносні одиничні показники якості по кожному параметру за формулами (5.1) та (5.2) і занесемо їх у відповідну колонку табл. 5.5.

$$q_i = \frac{P_{Hi}}{P_{Бi}} \tag{5.1}$$

або

$$q_i = \frac{P_{Бi}}{P_{Нi}} \tag{5.2}$$

де $P_{Нi}$, $P_{Бi}$ – числові значення *i*-го параметру відповідно нового і базового виробів.

$$q_1 = \frac{85}{70} = 1{,}21;$$

$$q_2 = \frac{16}{16} = 1;$$

$$q_3 = \frac{7000}{3000} = 2{,}3.$$

Відносний рівень якості нової розробки визначаємо за формулою:

$$К_{я.в.} = \sum_{i=1}^{n} q_i \cdot \alpha_i, \tag{5.3}$$

$$К_{я.в.} = 1{,}21 \cdot 0{,}35 + 1 \cdot 0{,}35 + 2{,}3 \cdot 0{,}3 = 1{,}4$$

Відносний коефіцієнт показника якості нової розробки більший одиниці, отже нова розробка якісніший базового товару-конкурента.

Наступним кроком є визначення конкурентоспроможності товару. Конкурентоспроможність товару є головною умовою конкурентоспроможності підприємства на ринку і важливою основою прибутковості його діяльності.

Однією із умов вибору товару споживачем є збіг основних ринкових характеристик виробу з умовними характеристиками конкретної потреби покупця. Такими характеристиками найчастіше вважають нормативні та технічні параметри, а також ціну придбання та вартість споживання товару.

В табл. 5.5 наведено технічні та економічні показники для розрахунку конкурентоспроможності нової розробки відносно товару-аналога, технічні дані взяті з попередніх розрахунків.

Таблиця 5.5 – Нормативні, технічні та економічні параметри нової розробки і товару-виробника

Показники	Варіанти	
	Базовий (товар-конкурент)	Новий (інноваційне рішення)
1	2	3
1. Нормативно-технічні показники		
Якість відновленого зображення, %	70	85
Коефіцієнт ущільнення	16	16
Швидкодія, мс	7000	3000
2. Економічні показники		
Ціна придбання, грн.	60000	55000

Загальний показник конкурентоспроможності інноваційного рішення (К) з урахуванням вищезазначених груп показників можна визначити за формулою:

$$K = \frac{I_{m.n.}}{I_{e.n.}}, \tag{5.4}$$

де $I_{m.n.}$ – індекс технічних параметрів; $I_{e.n.}$ – індекс економічних параметрів.

Індекс технічних параметрів є відносним рівнем якості інноваційного рішення. Індекс економічних параметрів визначається за формулою (5.5)

$$I_{e.n.} = \frac{\sum_{i=1}^{n} P_{Hei}}{\sum_{i=1}^{n} P_{Бei}}, \tag{5.5}$$

де P_{Hei}, $P_{Бei}$ – економічні параметри (ціна придбання та споживання товару) відповідно нового та базового товарів.

$$\text{I}_{\text{е.п.}} = \frac{55000}{60000} = 0{,}9;$$

$$\text{К} = \frac{1{,}4}{0{,}9} = 1{,}6.$$

Зважаючи на розрахунки, можна зробити висновок, що нова розробка буде конкурентоспроможніше, ніж конкурентний товар.

5.2 Прогнозування витрат на виконання науково-дослідної роботи

Витрати, пов'язані з проведенням науково-дослідної роботи групуються за такими статтями: витрати на оплату праці, витрати на соціальні заходи, матеріали, паливо та енергія для науково-виробничих цілей, витрати на службові відрядження, програмне забезпечення для наукових робіт, інші витрати, накладні витрати.

1. Основна заробітна плата кожного із дослідників $З_О$, якщо вони працюють в наукових установах бюджетної сфери визначається за формулою:

$$З_О = \frac{М}{Т_р} * t \text{ (грн)} \quad (5.6)$$

де $М$ – місячний посадовий оклад конкретного розробника (інженера, дослідника, науковця тощо), грн.;

$Т_Р$ – число робочих днів в місяці; приблизно $Т_Р \approx 21...23$ дні;

t – число робочих днів роботи дослідника.

Для розробки необхідно залучити програміста з посадовим окладом 10000 грн. Кількість робочих днів у місяці складає 40, а кількість робочих днів програміста складає 22. Зведемо сумарні розрахунки до таблиця 5.6.

Таблиця 5.6 – Заробітна плата дослідника в науковій установі бюджетної сфери

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату грн.
Керівник	12000	545,5	5	2727
Програміст	13000	590,9	24	14182
Всього				16909

2. Розрахунок додаткової заробітної плати робітників

Додаткова заробітна плата $З_д$ всіх розробників та робітників, які приймали устають в розробці нового технічного рішення розраховується як 10 - 12 % від основної заробітної плати робітників.

На даному підприємстві додаткова заробітна плата начисляється в розмірі 10% від основної заробітної плати.

$$З_д = (З_о + З_р) * \frac{Н_{дод}}{100\%} \quad (5.7)$$

$$З_д = 0{,}11 * 16909 = 1860 \text{ (грн)}$$

3. Нарахування на заробітну плату $Н_{ЗП}$ дослідників та робітників, які брали участь у виконанні даного етапу роботи, розраховуються за формулою (5.8):

$$Н_{ЗП} = (З_о + З_д) * \frac{\beta}{100} \text{ (грн)} \quad (5.8)$$

де $З_о$ – основна заробітна плата розробників, грн.;

$З_д$ – додаткова заробітна плата всіх розробників та робітників, грн.;

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

Дана діяльність відноситься до бюджетної сфери, тому ставка єдиного внеску на загальнообов'язкове державне соціальне страхування буде складати 22%, тоді**:**

$$Н_{ЗП} = (16909 + 1860\ ) * \frac{22}{100} = 4129{,}2 \text{ (грн)}$$

4. Витрати на комплектуючі вироби, які використовують при виготовленні одиниці продукції, розраховуються, згідно їх номенклатури, за формулою:

$$К=\sum_{i=1}^{n} Н_i \cdot Ц_i \cdot К_i \quad , \tag{5.9}$$

де $Н_i$ – кількість комплектуючих *i*-го виду, шт.;

$Ц_i$ – покупна ціна комплектуючих *i*-го найменування, грн.;

$К_i$ – коефіцієнт транспортних витрат (1,1…1,15).

Таблиця 5.7 – Комплектуючі, що використані на розробку

Найменування матеріалу	Ціна за одиницю, грн.	Витрачено	Вартість витраченого матеріалу, грн.
Папір	130	1	130
Ручка	20	1	20
CD-диск	15	1	15
Флешка	200	1	200
Всього			365
З врахуванням коефіцієнта транспортування			401,5

5. Програмне забезпечення для наукової роботи включає витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення необхідного для проведення дослідження.

Для створення програмного продукту використовувались безкоштовні версії Microsoft Visual Studio Community Edition та Unity Personal Edition.

6. Амортизація обладнання, комп’ютерів та приміщень, які використовувались під час виконання даного етапу роботи

Дані відрахування розраховують по кожному виду обладнання, приміщенням тощо.

$$A = \frac{Ц \cdot T}{Tкор \cdot 12} \quad [грн], \tag{5.10}$$

де Ц – балансова вартість даного виду обладнання (приміщень), грн.;

$Т_{кор}$ – час користування;

Т – термін використання обладнання (приміщень), цілі місяці.

Згідно пункта 137.3.3 Податкового кодекса амортизація нараховується на основні засоби вартістю понад 2500 грн. В нашому випадку для написання магістерської роботи використовувався персональний комп'ютер вартістю 24000 грн.

$$\text{А} = \frac{22000 \cdot 1}{2 \cdot 12} = 916{,}67$$

7. До статті «Паливо та енергія для науково-виробничих цілей» відносяться витрати на всі види палива й енергії, що безпосередньо використовуються з технологічною метою на проведення досліджень.

$$\text{В}_\text{е} = \sum_{\text{і}=1}^{n} \frac{W_{yt} \cdot t_i \cdot \text{Ц}_\text{е} \cdot \text{К}_{\text{впі}}}{\eta_\text{і}} \qquad (5.11)$$

де W_{yt} – встановлена потужність обладнання на певному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$\text{Ц}_\text{е}$ – вартість 1 кВт-години електроенергії, грн;

$\text{К}_{\text{впі}}$ – коефіцієнт, що враховує використання потужності, $\text{К}_{\text{впі}} < 1$;

$\eta_\text{і}$ – коефіцієнт корисної дії обладнання, $\eta_\text{і} < 1$.

Для написання магістерської роботи використовується персональний комп'ютер для якого розрахуємо витрати на електроенергію.

$$\text{В}_\text{е} = \frac{0{,}3 \cdot 150 \cdot 4{,}1 \cdot 0{,}5}{0{,}8} = 115{,}31$$

Витрати на службові відрядження, витрати на роботи, які виконують сторонні підприємства, установи, організації та інші витрати в нашому досліджені не враховуються оскільки їх не було.

Накладні (загальновиробничі) витрати Внзв охоплюють: витрати на управління організацією, оплата службових відряджень, витрати на утримання, ремонт та експлуатацію основних засобів, витрати на опалення, освітлення, водопостачання, охорону праці тощо. Накладні (загальновиробничі) витрати

Внзв можна прийняти як (100…150)% від суми основної заробітної плати розробників та робітників, які виконували дану МКНР, тобто:

$$В_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (5.12)$$

де $Н_{нзв}$ – норма нарахування за статтею «Інші витрати».

$$В_{нзв} = 16909 \cdot \frac{100}{100\%} = 16909 \text{ грн}$$

Сума всіх попередніх статей витрат дає витрати, які безпосередньо стосуються даного розділу МКНР

$$В = 16909+1860+4129,2+401,5+916,67+115,31+16909 = 41240,9$$

Прогнозування загальних втрат ЗВ на виконання та впровадження результатів виконаної МКНР здійснюється за формулою:

$$ЗВ = \frac{В}{\eta}, \quad (5.13)$$

де η – коефіцієнт, який характеризує стадію виконання даної НДР.

Оскільки, робота знаходиться на стадії науково-дослідних робіт, то коефіцієнт β = 0,9.

Звідси:

$$ЗВ = \frac{41240,9}{0,9} = 45823,18 \text{ грн.}$$

5.3 Розрахунок економічної ефективності науково-технічної розробки

У даному підрозділі кількісно спрогнозуємо, яку вигоду, зиск можна отримати у майбутньому від впровадження результатів виконаної наукової роботи. Розрахуємо збільшення чистого прибутку підприємства $\Delta П_i$, для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, за формулою

$$\Delta П_i = \sum_1^n \left(\Delta Ц_o \cdot N + Ц_o \cdot \Delta N\right)_i \cdot \lambda \cdot \rho \cdot \left(1 - \frac{\upsilon}{100}\right) \quad (5.14)$$

де $\Delta Ц_0$ – покращення основного оціночного показника від впровадження результатів розробки у даному році.

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки:

$Ц_o$ – основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки:

л – коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт *л* = 0,8333.

р – коефіцієнт, який враховує рентабельність продукту. р = 0,25;

х – ставка податку на прибуток. У 2021 році – 18%.

Припустимо, що при впроваджені результатів наукової розробки покращується якість програмного продукту для ущільнення зображень на основі карти Кохонена. Припустимо, що ціна від зросте на 1000 грн. Кількість одиниць реалізованої продукції також збільшиться: протягом першого року на 20 шт., протягом другого року – на 15 шт., протягом третього року на 10 шт. Реалізація продукції до впровадження розробки складала 1 шт., а її ціна до складає 55000 грн. Розрахуємо прибуток, яке отримає підприємство протягом трьох років.

$$\Delta П_1 = [1000 \cdot 1 + (55000 + 1000) \cdot 20] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 + \frac{18}{100}\right) = 191496{,}51 \text{ грн.}$$

$$\Delta П_2 = [1000 \cdot 1 + (55000 + 1000) \cdot (20 + 15)] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 + \frac{18}{100}\right) = 335819{,}94 \text{ грн.}$$

$$\Delta П_3 = [1000 \cdot 1 + (55000 + 1000) \cdot (20 + 15 + 10)] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 + \frac{18}{100}\right) = 431482{,}78 \text{ грн.}$$

5.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності

Розрахуємо основні показники, які визначають доцільність фінансування наукової розробки певним інвестором, є абсолютна і відносна ефективність вкладених інвестицій та термін їх окупності.

Розрахуємо величину початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки.

$$PV = k_{\text{інв}} \cdot \text{ЗВ}, \tag{5.15}$$

$k_{\text{інв}}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо ($k_{\text{інв}} = 2…5$).

$$PV = 2 \cdot 45823{,}18 = 91646{,}36$$

Розрахуємо абсолютну ефективність вкладених інвестицій $E_{абс}$ згідно наступної формули:

$$E_{абс} = (ПП - PV) \tag{5.16}$$

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

$$ПП = \sum_{1}^{T} \frac{\Delta П_i}{(1+\tau)^t},$$

(5.17)

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДЦКР, грн.;

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

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,2;

t – період часу (в роках).

$$ПП = \frac{191496,51}{(1+0,2)^1} + \frac{335819,94}{(1+0,2)^2} + \frac{431482,78}{(1+0,2)^3} = 643650,8 \text{ грн.}$$

$$Е_{абс} = (643650,8 - 91646,36) = 552004,44 \text{грн.}$$

Оскільки $E_{абс} > 0$ то вкладання коштів на виконання та впровадження результатів НДДКР може бути доцільним.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$. Для цього користуються формулою:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \qquad (5.18)$$

$T_ж$ – життєвий цикл наукової розробки, роки.

$$Е_в = \sqrt[3]{1 + \frac{552004,44}{91646,36}} - 1 = 1,35 = 135\%$$

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою:

$$\tau = d + f,$$

(5.19)

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2021 році в Україні d= (0,14…0,2);

f−показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,1).

$$\tau_{\min} = 0,18 + 0,05 = 0,23.$$

Так як $E_в > \tau_{\min}$ то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Розрахуємо термін окупності вкладених у реалізацію наукового проекту інвестицій за формулою:

$$T_{ок} = \frac{1}{E_в} \tag{5.20}$$

$$\mathrm{T_{ок}} = \frac{1}{1,35} = 0,7 \text{ роки}$$

Так як $T_{ок} \leq$ 3...5-ти років, то фінансування даної наукової розробки в принципі є доцільним.

5.5 Висновки до економічного розділу

Було проведено оцінку комерційного потенціалу розробка методів та програмного засобу ущільнення зображень на основі карти Кохонена, який є на середньому рівні. При порівнянні нової розробки з аналогом виявлено, що вона є якіснішою і конкурентоспроможнішою відносно аналога, а також краще по технічним і економічним показникам.

Прогнозування витрат на виконання науково-дослідної роботи по кожній з статей витрат складе 41240,9 грн. Загальна ж величина витрат на виконання та впровадження результатів даної НДР буде складати 45823,18 грн.

Вкладені інвестиції в даний проект окупляться через 7 місяців при прогнозованому прибутку 643650, грн. за три роки.

ВИСНОВКИ

У роботі подальшого розвитку отримав метод ущільнення зображень на основі двовимірної карти Кохонена, який відрізняється від існуючих застосуванням паралельних обчислень та багатопотоковості, що дозволило підвищити швидкість кодування у 3 рази. Мовою програмування C# в середовищі розробки Unity з використанням Microsoft Visual Studio Community edition 2019 розроблено програмний засіб для ущільнення зображень на основі карти Кохонена.

Основні результати отримані під час виконання даної роботи такі:

1. Подальшого розвитку отримав метод ущільнення зображень на основі векторного квантування, у якому, на відміну від існуючих, використано двовимірну карту Кохонена в якості векторного квантувача, що дозволило підвищити коефіцієнт ущільнення зображень на 10 % без значної втрати якості зображення.
2. Проведені дослідження показали, що застосування карти Кохонена є перспективним для виконання ущільнення методами векторного квантування. Ряд експериментів з різними типами зображень показав, що коефіцієнти ущільнення можуть знаходитися в межах 6 – 30. Для деяких зображень коефіцієнт ущільнення перевершує стандарт JPEG при тій же якості зображення.
3. Напрямок подальших досліджень пов’язаний з застосування карт Кохонена в комбінації з смуговими методами і двовимірними ортогональними перетвореннями

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Прэтт У. Цифровая обработка зображений: Пер. с англ. / У. Прэтт – М.: Мир, 1982. – Кн.2. – 480 с.
2. Каллан Р. Основные концепции нейронных сетей.: Пер. с англ. / Р. Каллан – М.: «Вильямс», 2001. – 286 с.: ил.
3. Майданюк В. П. Весняні наукові зібрання – 2020 / В. П. Майданюк, Я. В. Педченко – Суми,2020 – 80с.
4. Збірник матеріалів Міжнародної науково-практичної Інтернет конференції 9-10 листопада 2021 р. – Суми/Вінниця: НІКО/ВНТУ, 2021. – 223 с.
5. Матеріали L науково-технічної конференції підрозділів Вінницького національного технічного університету (НТКП ВНТУ–2021) : збірник доповідей. – Вінниця : ВНТУ, 2021.
6. Міжнародної науково-практична Інтернет конференція «Електронні інформаційні ресурси: створення, використання, доступ» (9-10 листопада 2021 р., Суми/Вінниця).
7. Мюррей Дж.Д. Энциклопедия форматов графических файлов: Пер. с англ. / Дж.Д. Мюррей, У. Райпер - К.: ВНУ, 1997. - 672 с.
8. Круглов. Искусственные нейронные сети. Теория и практика./ Круглов., В.В. Борисов – М.: Горячая лия – Телеком, 2001. – 382 с.: ил.
9. Стиснення даних [Електронний ресурс] // Вікіпедія – режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Стиснення_даних
10. Ватолин Д.С. Алгоритмы сжатия изображений./ Д.С. Ватолин – М.: Издательский отдел факультета Вычислительной Математики и Кибернетики МГУ им. М.В.Ломоносова (лицензия ЛР № 040777 от 23.07.96), 1999 г.-76 с.

11. Імпульсно-кодова модуляція [Електронний ресурс] // Вікіпедія – режим доступу до ресурсу: https://uk.wikipedia.org/wiki/ Імпульсно-кодова_модуляція
12. Майданюк В.П. Методи і засоби комп'ютерних інформаційних технологій. Кодування зображень. / В.П. Майданюк – Вінниця: ВДТУ, 2001. – 63 с.
13. Птачек М. Цифровое телевидение. Теория и техника: пер. с чешск. / М. Птачек - М.: Радио и связь, 1990. - 528 с.
14. Код Хаффмана [Електронний ресурс] // Вікіпедія – режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Код_Гаффмана
15. JPEG [Електронний ресурс] // Вікіпедія – режим доступу до ресурсу: https://uk.wikipedia.org/wiki/JPEG
16. Дизайн інтерфейсу користувача [Електронний ресурс] // Вікіпедія – режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Дизайн_інтерфейсу_користувача
17. Самоорганізаційна карта Кохонена [Електронний ресурс] // Вікіпедія – режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Самоорганізаційна_Карта_Кохонена
18. Арифметичне кодування [Електронний ресурс] // Вікіпедія – режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Арифметичне_кодування
19. Паралельне обчислення [Електронний ресурс] // Вікіпедія – режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Паралельні_обчислення
20. Amdahl, G. (April 1967) «The validity of the single processor approach to achieving large-scale computing capabilities». In Proceedings of AFIPS Spring Joint Computer Conference, Atlantic City, N.J., AFIPS Press, pp. 483-85.

21. C# [Електронний ресурс] // Вікіпедія – режим доступу до ресурсу: https://uk.wikipedia.org/wiki/C_Sharp
22. Unity [Електронний ресурс] // Вікіпедія – режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Unity_(рушій_гри)
23. MonoDevelop [Електронний ресурс] // Вікіпедія – режим доступу до ресурсу: https://uk.wikipedia.org/wiki/MonoDevelop.
24. Microsoft Visual Studio [Електронний ресурс] // Вікіпедія – режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Microsoft_Visual_Studio
25. Visual Studio Code [Електронний ресурс] // Вікіпедія – режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Visual_Studio_Code
26. Пєтух А.М. Інформаційно-вимірювальні системи відновлення і ущільнення зображень: монографія / А.М. Пєтух, О. М. Рейда, В. П. Майданюк, В. П. Кожем'яко. – Вінниця: ВНТУ, 2011. – 144 с.
27. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с

Додаток А

(обов’язковий)

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп’ютерної інженерії

ЗАТВЕРДЖУЮ

д.т.н., проф. О. Н. Романюк

"____ "_____2021 р.

Технічне завдання

на магістерську кваліфікаційну роботу «Розробка методу та програмних засобів ущільнення зображень на основі карти Кохонена» за спеціальністю 121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:

_____к.т.н., доц. каф. ПЗ В.П. Майданюк

"____"_____2021 р.

Виконав:

_____студент гр.2ПІ-17м Я. В. Педченко

"____"_____2021 р.

Вінниця – 2021 року

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Розробка методу та програмних засобів ущільнення зображень на основі карти Кохонена».

Галузь застосування - системи комп'ютерної графіки.

2. Підстава для розробки.

Підставою для виконання магістерської кваліфікаційної роботи (МКР) є індивідуальне завдання на МКР та наказ № ректора по ВНТУ про закріплення тем МКР.

3. Мета та призначення розробки.

Метою роботи є підвищення швидкодії та коефіцієнта ущільнення зображень за допомогою двовимірної карти Кохонена.

Призначення роботи – розробка методу та програмних засобів ущільнення зображень на основі карти Кохонена.

3 Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись МКР.

1. Майданюк В.П. Методи і засоби комп'ютерних інформаційних технологій. Кодування зображень. / В.П. Майданюк – Вінниця: ВДТУ, 2001. – 63 с.
2. Ватолин Д.С. Алгоритмы сжатия изображений./ Д.С. Ватолин – М.: Издательский отдел факультета Вычислительной Математики и Кибернетики МГУ им. М.В.Ломоносова (лицензия ЛР № 040777 от 23.07.96), 1999 г.-76 с.
3. Майданюк В. П. Весняні наукові зібрання – 2020 / В. П. Майданюк, Я. В. Педченко – Суми,2020 – 80с.

4. Збірник матеріалів Міжнародної науково-практичної Інтернет конференції 9-10 листопада 2021 р. – Суми/Вінниця: НІКО/ВНТУ, 2021. – 223 с.
5. Матеріали L науково-технічної конференції підрозділів Вінницького національного технічного університету (НТКП ВНТУ–2021) : збірник доповідей. – Вінниця : ВНТУ, 2021.

4. Технічні вимоги

Методи квантування зображень за домогою самоорганізовуючої двовимірної карти Кохонена.

5. Конструктивні вимоги.

Конструкція пристрою повинна відповідати естетичним та ергономічним вимогам, повинна бути зручною в обслуговуванні та керуванні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

6. Перелік технічної документації, що пред'являється по закінченню робіт:

– пояснювальна записка до МКР;

– технічне завдання;

– лістинги програми.

7. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

8. Стадії та етапи розробки:

№ з/п	Назва етапів магістерської кваліфікаційної Роботи	Строк виконання етапів роботи
1	Аналіз та обґрунтування вибору методу розробки та постановка задачі дослідження	15.09. 2021 – 27.09.2021
2	Розробка структури та алгоритмів роботи програмного продукту	28.09.2021 – 07.10.2021
3	Розробка програми для ущільнення зображень	11.10.2021 - 27.10.2021
4	Тестування програми	28.10.2021 – 16.11.2021
5	Економічна частина	17.11.2021 - 30.11.2021

9. Порядок контролю та прийняття.

Виконання етапів магістерської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи. Прийняття магістерської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком

Додаток Б
(обов'язковий)

ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи: Розробка методу та програмних засобів ущільнення зображень на основі карти Кохонена

Тип роботи: кваліфікаційна робота

Підрозділ : кафедра програмного забезпечення, ФІТКІ, 1ПІ – 20м

Науковий керівник: к.т.н. доц. Майданюк В. П.

Unicheck	
Оригінальність	**93,8%%**
Схожість	6,2%

Аналіз звіту подібності

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

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

□ 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.

Заявляю, що ознайомлена з повним звітом подібності, який був згенерований Системою щодо роботи «Розробка методу та програмних засобів ущільнення зображень на основі карти Кохонена».

Автор _____ Педченко Ярослав Володимирович

Опис прийнятого рішення: **допустити до захисту**

Особа, відповідальна за перевірку _____ Черноволик Г. О.
(підпис) (прізвище, ініціали)

Експерт _____ _____
(за потреби) (підпис) (прізвище, ініціали, посада)

ДОДАТОК В
(обов’язковий)

Лістинг програми для ущільнення зображень на основі карти Кохонена.

```
//Модуль кластеризації зображення на основі карти кохонена
    using System.Collections;
    using System.Collections.Generic;
    using UnityEngine;
    using UnityEngine.UI;

    namespace ImageComprace
    {
       public class ProcessImage : MonoBehaviour
       {
          [SerializeField] private RawImage _pictureBox1;
          [SerializeField] private RawImage _pictureBox2;
          [SerializeField] private DataContainer data;
          Texture2D image1;
          Texture2D image2;
          float alfa = 0.01f;
          int epochs = 25;
          int[][][] winers;
          float[][][] sample;
          float[][][] weights;

          public void proseccimage()
          {
             image1 = (_pictureBox1.texture as Texture2D);
```

```
if (image1.height % 2 != 0)
{
   winers = new int[(image1.height / 2) + 1][][];
   weights = new float[image1.height + 2][][];
   sample = new float[image1.height + 2][][];

}
else
{
   winers = new int[image1.height / 2][][];
   weights = new float[image1.height + 2][][];
   sample = new float[image1.height + 2][][];
}

for (int i = 0; i < image1.height+2; ++i)
{
   if (image1.width % 2 != 0)
   {
      weights[i] = new float[image1.width + 2][];
      sample[i] = new float[image1.width + 2][];
   }
   else
   {

      weights[i] = new float[image1.width + 2][];
      sample[i] = new float[image1.width + 2][];
   }
}
```

```
for (int i = 0; i < winers.Length; ++i)
{
   if (image1.width % 2 != 0)
      winers[i] = new int[(image1.width / 2) + 1][];

   else
      winers[i] = new int[image1.width / 2][];

   for (int j = 0; j < winers[i].Length; ++j)
      winers[i][j] = new int[2];
}

for (int i = 0; i < image1.height+2; ++i)
{
   for (int j = 0; j < image1.width+2; ++j)
   {
      Color color;
      if (i < image1.height && j < image1.width)
         color = image1.GetPixel(i, j);
      else
         color = new Color(0f, 0f, 0f);

      sample[i][j] = new float[3] { color.r, color.g, color.b };
      weights[i][j] = new float[3] { 0.01f, 0.01f, 0.01f };
   }
}

int[] w = new int[2];
```

```
            for (int e = 0; e < epochs; ++e)
            {
               for (int i = 0; i < image1.height-1; ++i)
                  for (int j = 0; j < image1.width-1; ++j)
                  {
                     w = winer(sample[i][j], i, j);
                     updatew(w[0], w[1], sample[i][j]);
                  }
            }

            for (int u = 0; u < winers.Length; ++u)
               for (int j = 0; j < winers[u].Length; ++j)
                        winers[u][j] = winer(sample[u*2][j*2], u*2, j*2);

            updateimage();
            data.Width = image1.width;
            data.Height = image1.height;
            data.SetSize(winers.Length, winers[0].Length);
            for (int u = 0; u < winers.Length; ++u)
               for (int j = 0; j < winers[u].Length; ++j)
               {
                  data.weights[u][j][0] = weights[u][j][0];
                  data.weights[u][j][1] = weights[u][j][1];
                  data.weights[u][j][2] = weights[u][j][2];
                  data.color[u, j] = image1.GetPixel(winers[u][j][0],
winers[u][j][1]);
               }
         }
```

```
private int[] winer(float[] localsaple, int index1, int index2)
{
    float D0 = 0;
    float D1 = 0;
    float D2 = 0;
    float D3 = 0;
    int[] res;

    for (int i = 0; i < localsaple.Length; ++i)
    {
        D0 = Mathf.Pow((localsaple[i] - weights[index1][index2][i]), 2);
        D1 = Mathf.Pow((localsaple[i] - weights[index1][index2 + 1][i]), 2);
        D2 = Mathf.Pow((localsaple[i] - weights[index1 + 1][index2][i]), 2);
        D3 = Mathf.Pow((localsaple[i] - weights[index1 + 1][index2 + 1][i]),
2);
    }


    if (D0 > D1 && D0 > D2 && D0 > D3)
    {
        res = new int[] { index1, index2 };
        return res;
    }

    else if (D1 > D0 && D1 > D2 && D1 > D3)
    {
        res = new int[] { index1, index2 + 1 };
        return res;
    }
```

```
            else if (D2 > D0 && D2 > D1 && D2 > D3)
            {
                res = new int[] { index1 + 1, index2 };
                return res;
            }

            else
            {
                res = new int[] { index1 + 1, index2 + 1 };
                return res;
            }
        }

        private void updatew(int J, int I, float[] localsaple)
        {
            for (int i = 0; i < 3; ++i)
                weights[J][I][i] = weights[J][I][i] + alfa * (localsaple[i] -
weights[J][I][i]);
        }


        private void updateimage()
        {
            image2 = image1;//new Texture2D(image1.width, image1.height);

            for (int i = 0; i < winers.Length; ++i)
            {
                for (int j = 0; j < winers[i].Length; ++j)
```

```
                {
                    image2.SetPixel(i * 2, j * 2, image1.GetPixel(winers[i][j][0], winers[i][j][1]));
                    image2.SetPixel(i * 2, (j * 2)+1, image1.GetPixel(winers[i][j][0], winers[i][j][1]));
                    image2.SetPixel((i * 2) + 1, j * 2, image1.GetPixel(winers[i][j][0], winers[i][j][1]));
                    image2.SetPixel((i * 2) + 1, (j * 2) + 1, image1.GetPixel(winers[i][j][0], winers[i][j][1]));
                    Debug.Log("raw "+ i*2 + "  pixel " + j *2 +"  1Color "+ image1.GetPixel(winers[i][j][0], winers[i][j][1]) + "  2 Color "+ image2.GetPixel(winers[i][j][0], winers[i][j][1]));
                }
            }

            image2.Apply();
            _pictureBox2.texture = image2;
        }
    }
}
//Модуль кешування
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

namespace ImageComprace
{
   public class DataContainer : MonoBehaviour
```

```
    {
        public Color[,] color;

        public float[][][] weights;

        public int Width;
        public int Height;

        public void SetSize(int height, int length)
        {
            color = new Color[height, length];
            weights = new float[height][][];
            for(int i = 0; i < height; ++i)
            {
                weights[i] = new float[length][];
                for (int j = 0; j < length; ++j)
                    weights[i][j] = new float[3];
            }
        }
    }
}
//Модуль відкриття зоюраження у форматі BMP
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.Networking;
using System.IO;
using SimpleFileBrowser;
```

```
namespace ImageComprace
{
   public class LoadImage : MonoBehaviour
   {
      [SerializeField] private RawImage rawImage;
      [SerializeField] private Text _imageName;
      private string _path;

      public void OpenImage()
      {
         FileBrowser.SetFilters(true, new FileBrowser.Filter("Images", ".bmp",
".tif"));
         FileBrowser.SetDefaultFilter(".bmp");
         StartCoroutine(ShowLoadDialogCoroutine());
      }

      IEnumerator ShowLoadDialogCoroutine()
      {
         yield return
FileBrowser.WaitForLoadDialog(FileBrowser.PickMode.FilesAndFolders, false,
null, null, "Завантажити Зображення", "Завантажити");

         Debug.Log(FileBrowser.Success);

         if (FileBrowser.Success)
         {
            for (int i = 0; i < FileBrowser.Result.Length; i++)
               Debug.Log(FileBrowser.Result[i]);
```

```
            _path = FileBrowser.Result[0];
            StartCoroutine(GetTexture());
        }
    }

    IEnumerator GetTexture()
    {
        UnityWebRequest www = UnityWebRequestTexture.GetTexture("file:///" +
_path);

        yield return www.SendWebRequest();

        if (www.isNetworkError || www.isHttpError)
            Debug.Log("request faild");

        Texture texture =
((DownloadHandlerTexture)www.downloadHandler).texture;
        _imageName.text = Path.GetFileName(_path);
        rawImage.texture = texture;
    }
  }
}

//Модуль запису ущільнених даних у файл
using System.Collections;
using System.Collections.Generic;
using System.IO;
using UnityEngine;
```

```
using SimpleFileBrowser;

namespace ImageComprace
{
    public class SaveImage : MonoBehaviour
    {
        [SerializeField] private DataContainer data;
        string saveFile;

        public void Save()
        {
            saveFile = string.Empty;
            saveFile += data.Height + "  " + data.Width + " " + data.weights.Length + " "
+ data.weights[0].Length + " ";

            for (int u = 0; u < data.weights.Length; ++u)
                for (int j = 0; j < data.weights[u].Length; ++j)
                {
                    saveFile += "  " + data.color[u, j].r + "  " +
                    data.color[u, j].g + "  " +
                    data.color[u, j].b;
                }

            FileBrowser.SetFilters(true, new FileBrowser.Filter("Images", ".atc"));
            FileBrowser.SetDefaultFilter(".atc");
            StartCoroutine(ShowSaveDialogCoroutine());
        }

        IEnumerator ShowSaveDialogCoroutine()
```

```
        {
            yield return FileBrowser.WaitForSaveDialog(FileBrowser.PickMode.Files,
false, "C:\\", "unnamed.atc", "בונודעט", "בונודעט");

            Debug.Log(FileBrowser.Success);

            if (FileBrowser.Success)
                FileBrowserHelpers.WriteTextToFile(FileBrowser.Result[0], saveFile);
        }
    }
}
```

Додаток Г

(обов’язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

Розробка методу та програмних засобів ущільнення зображень на основі карти Кохонена

(Назва магістерської кваліфікаційної роботи)

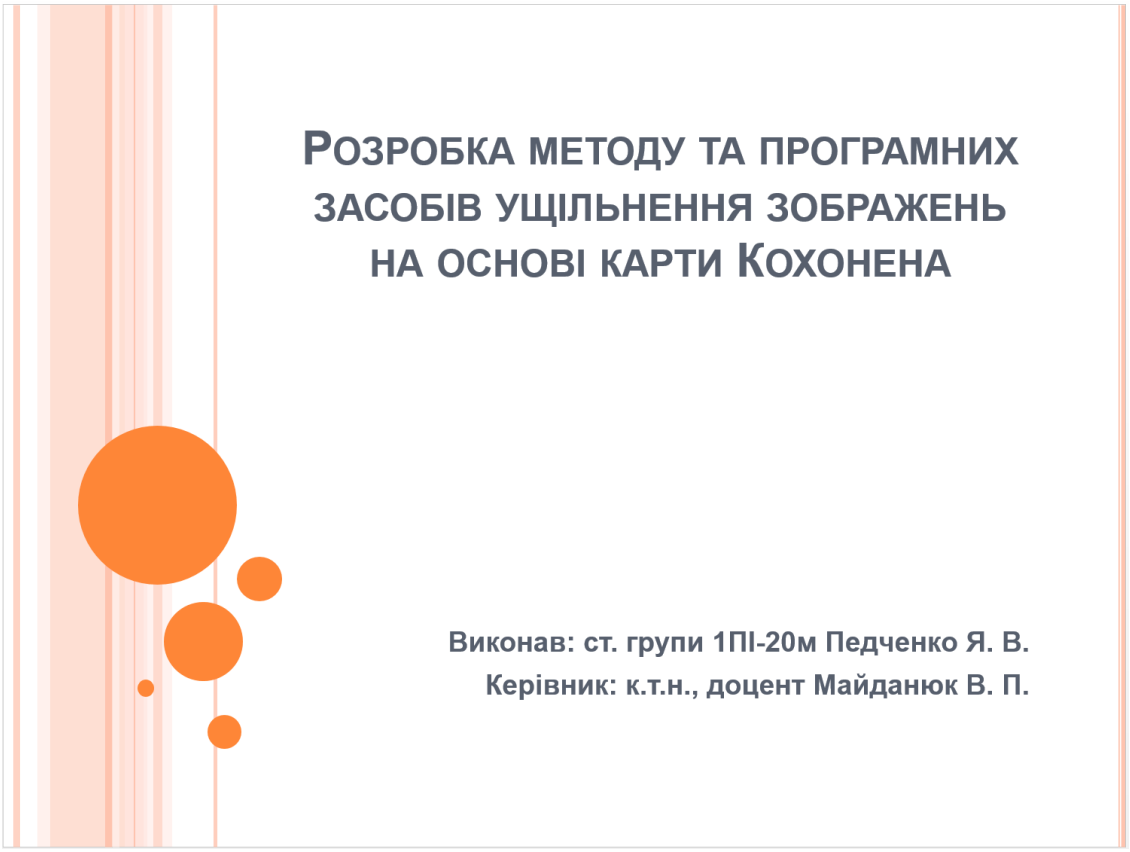


Рисунок Г.1 – Титульний слайд

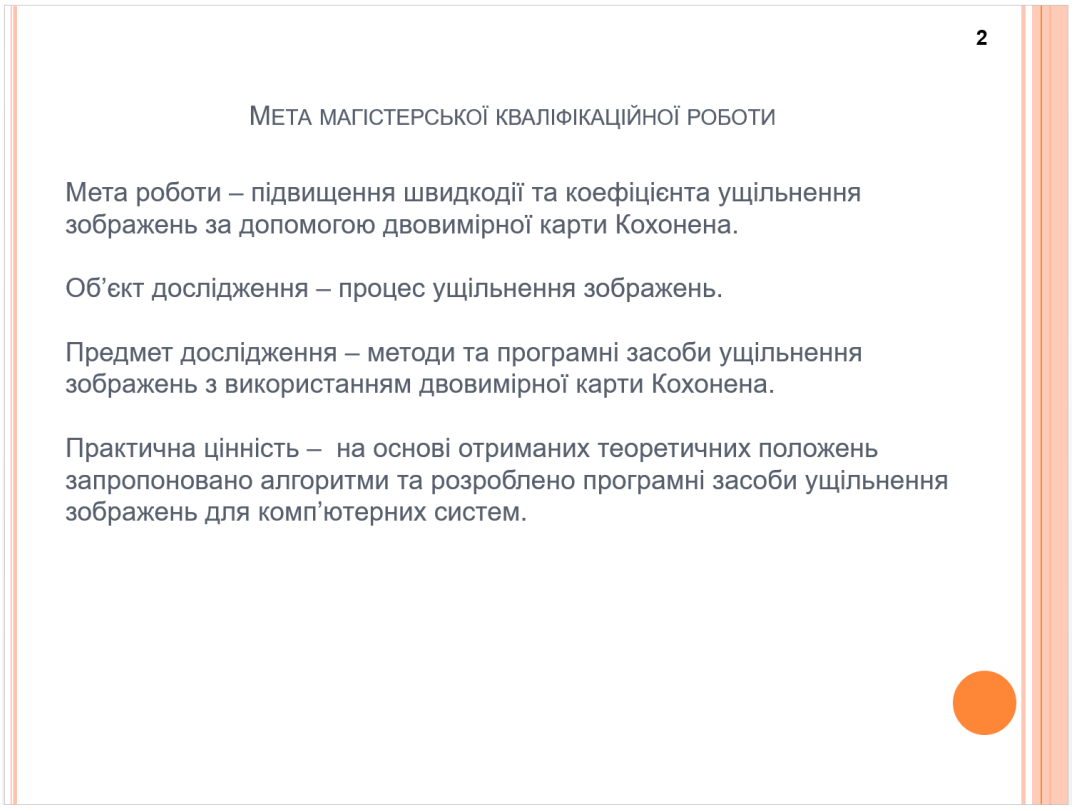


Рисунок Г.2 – Мета, об’єкт та предмет дослідження

3

НАУКОВА НОВИЗНА

Здобув подальший розвиток метод ущільнення зображень на основі двовимірної карти Кохонена.

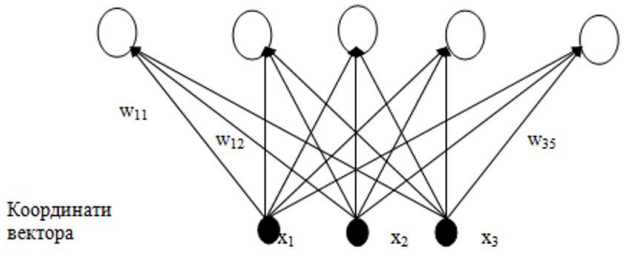
Особливість якого полягає у підвищеній швидкодії в 3 рази порівняно з попередньою реалізацією за рахунок паралельного обчислення декількох частин одночасно.

Що дає можливість скоротити час необхідний для ущільнення зображень.

Рисунок Г.3 – Наукова новизна одержаних результатів

4

НЕЙРОННА МЕРЕЖА



Навчальний вектор $d_j = \sum_i (w_{ij} - x_i)^2$

Оновлення вагових коефіцієнтів $w_{ij}(n+1) = w_{ij}(n) + \eta(n)[x_i - w_{ij}(n)]$

η – норма навчання

x_i – координата навчального вектора.

Рисунок Г.4 – Будова нейроної мережі

Рисунок Г.5 – Схема структури додатку

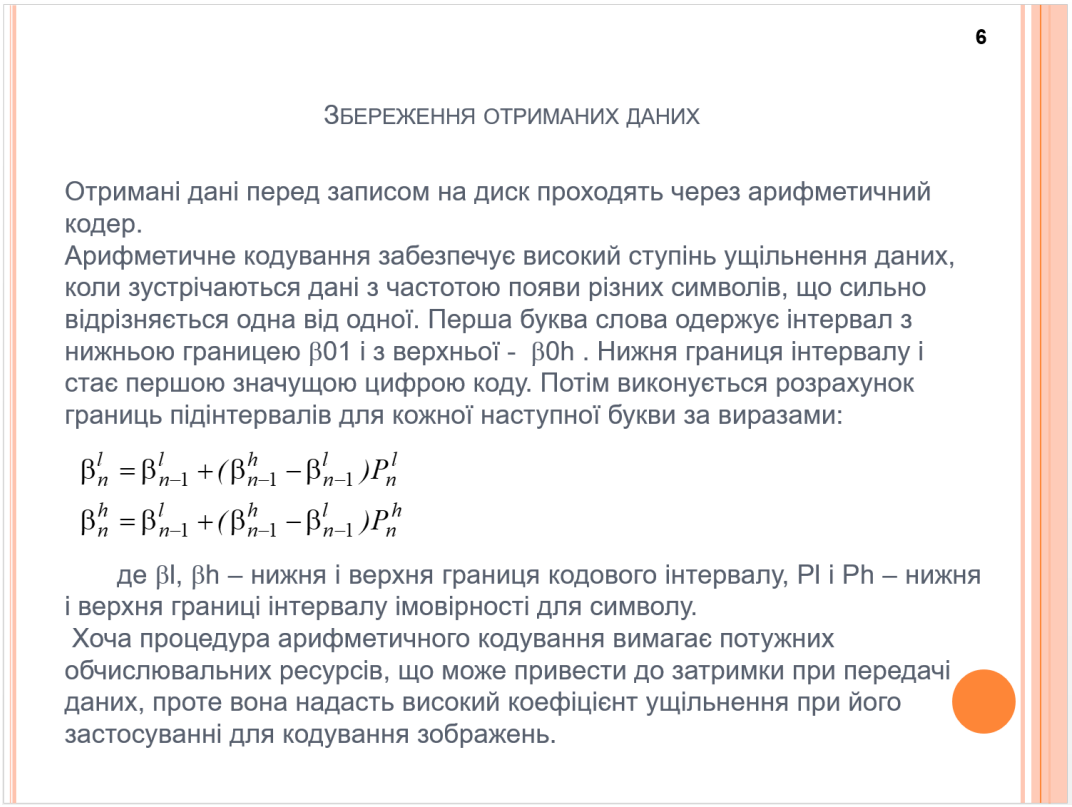


Рисунок Г.6 – Метод збереження отриманих даних

8

Вигляд обробленого зображення

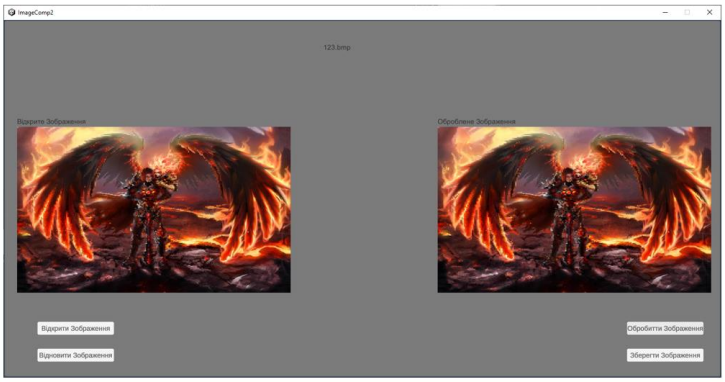


Рисунок Г.7 – Вигляд обробленого зображення

9

Порівняння JPEG з картою Кохонена

Метод ущільнення	Розмір початкового файлу, байт	Розмір стиснутого файлу, байт	Коефіці-єнт ущільнення	Середньо-квадратична помилка	Візуальна оцінка якості
JPEG	192 054	13106	14,7	0,019	Відмінна
Карта Кохонена (16x16)	192054	11740	16,4	0,02	Відмінна
Карта Кохонена (14x14)	192 054	10979	17,5	0,023	Добра
Карта Кохонена (11x11)	192054	9245	20,8	0,026	Добра
Карта Кохонена (8x8)	192054	6954	27,6	0,03	Задовільна

Рисунок Г.8 – Порівняння JPEG з картою Кохонена

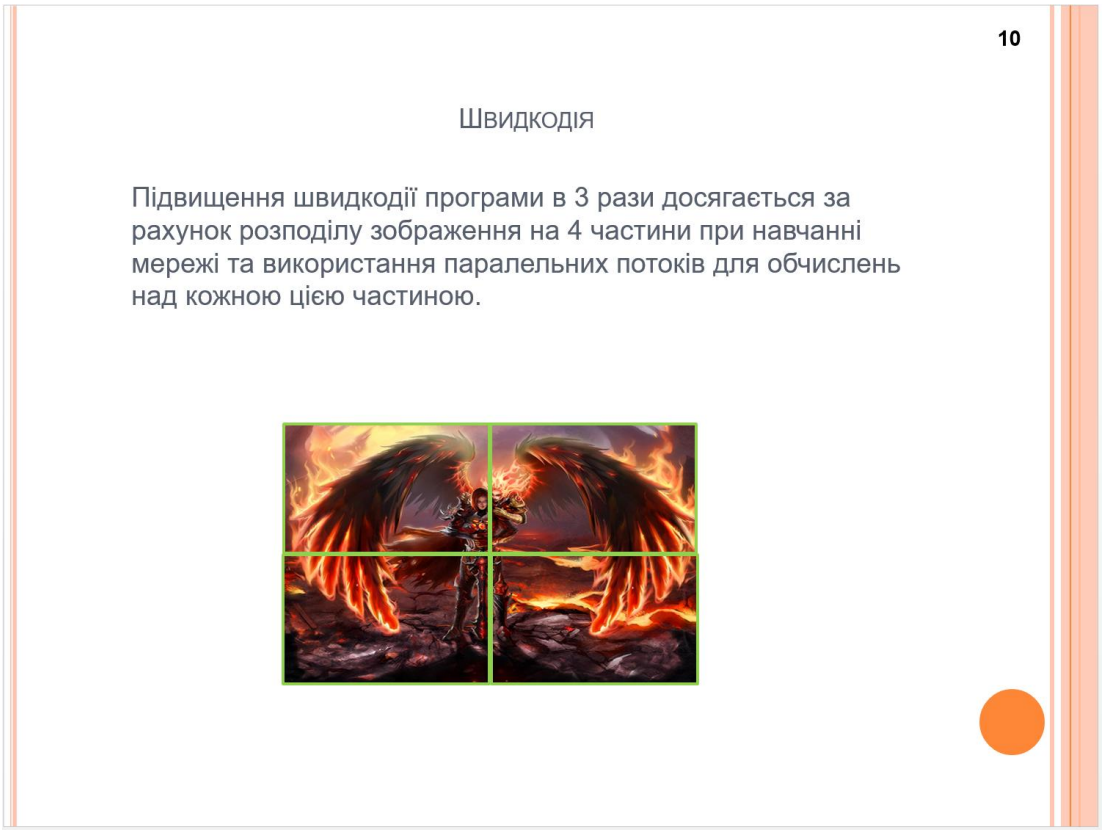


Рисунок Г.9 – Швидкодія системи

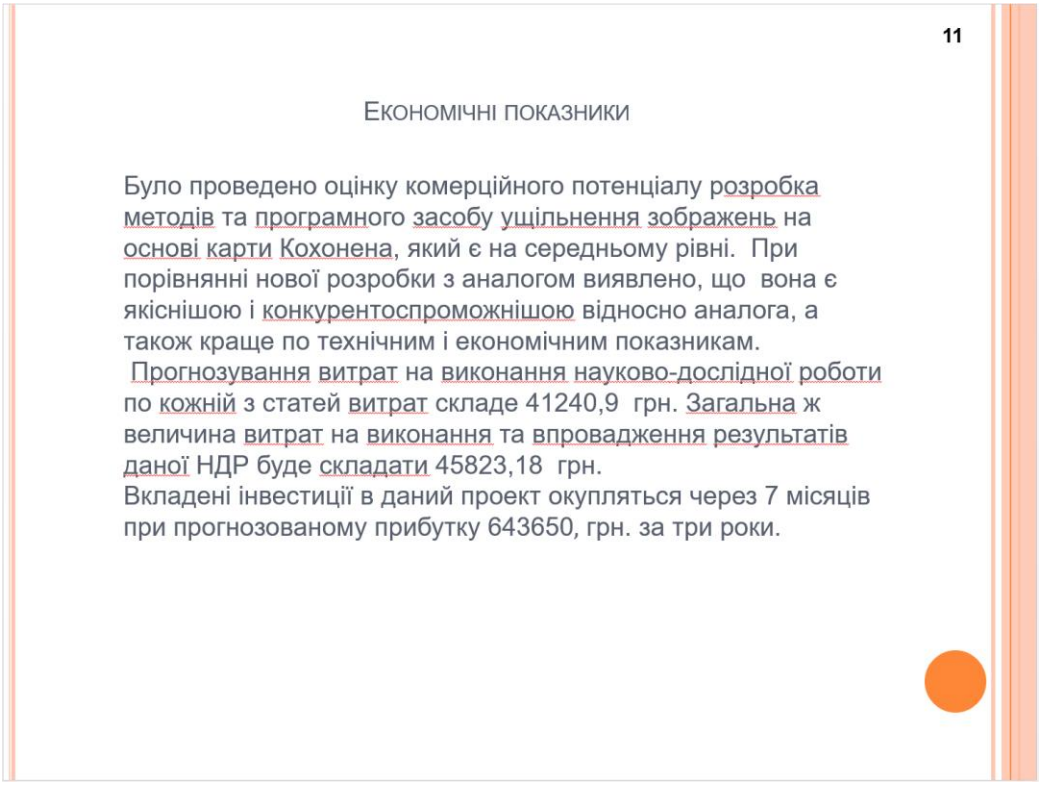


Рисунок Г.10 – Економічна частина

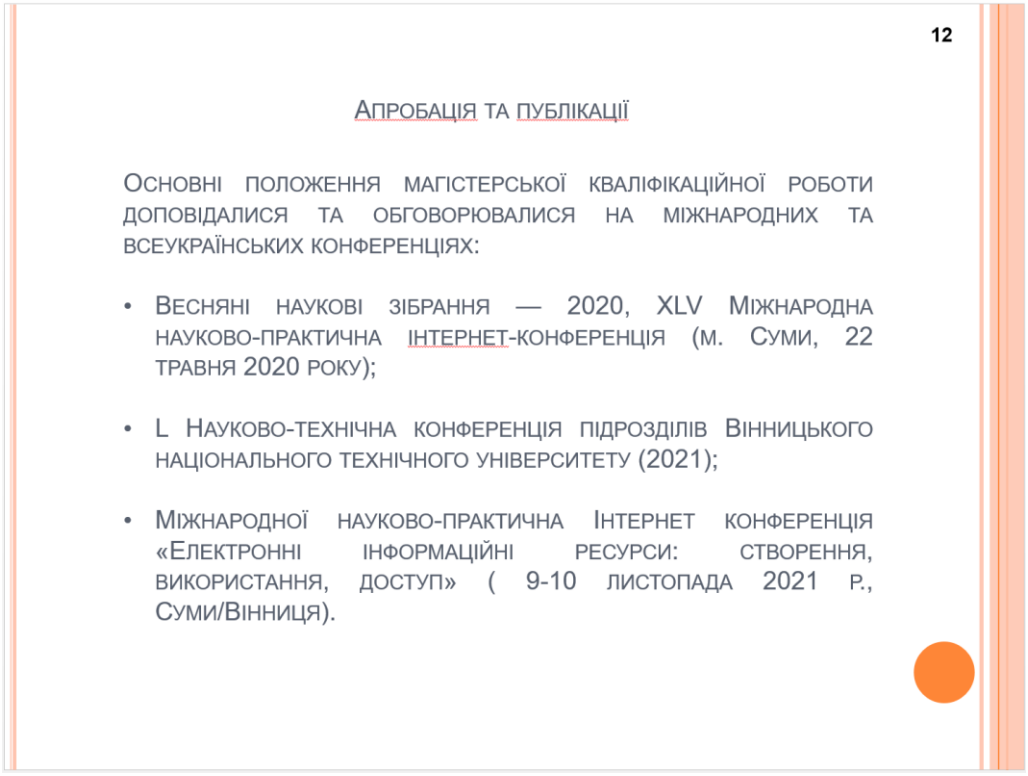


Рисунок Г.11 – Апробація та публікації

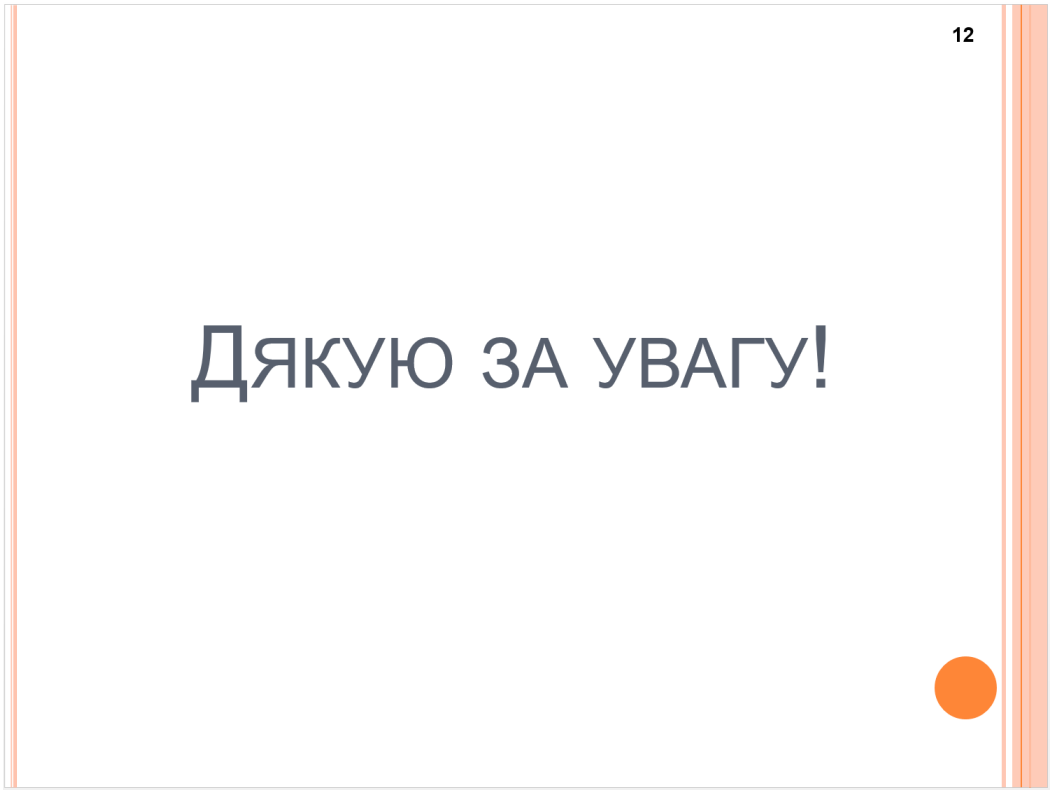


Рисунок Г.12 – Дякую за увагу