Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет кібербезпеки
(повна назва)

Кафедра інформаційно-мережної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

рівень вищої освіти *другий(магістерський)*

Дослідження алгоритмів стиснення зображень для передачі в інфокомунікаційних мережах
(тема)

Виконав:
здобувач 2 року навчання,
групи ІМІм-24-2
Олександр ЗЕМСЬКИЙ
(власне ім'я, прізвище)

Спеціальність 172 Електронні комунікації та радіотехніка
(код і повна назва спеціальності)
Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)
Освітня програма Інформаційно-мережна інженерія
(повна назва освітньої програми)

Керівник доц. Наталія ХАРЧЕНКО
(посада, власне ім'я, прізвище)

Допускається до захисту

Зав. кафедри _____ Микола МОСКАЛЕЦЬ.
(підпис) (власне ім'я, прізвище)

2026 р.

Не містить відомостей, заборонених до відкритого публікування

Здобувач _____ / Олександр Земський___ /
(підпис) (власне ім'я, прізвище)

Керівник _____ / Наталія Харченко /
(підпис) (власне ім'я, прізвище)

Харківський національний університет радіоелектроніки

Факультет кібербезпеки

Кафедра інформаційно-мережної інженерії

Рівень вищої освіти другий (магістерський)

Спеціальність 172 Електронні комунікації та радіотехніка

Тип програми освітньо-наукова
(код і повна назва)

Освітня програма інформаційно-мережна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Земському Олександру Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження алгоритмів стиснення зображень для передачі в інфокомунікаційних мережах
затверджено наказом університету від 23 березня 2026р . №232 Ст
2. Термін подання студентом роботи до екзаменаційної комісії 27 травня 2026р .
3. Вихідні дані до роботи
Розглянути основні методи стиснення зображень.
Дослідити класичні та сучасні методи стиснення зображень, проаналізувати принцип роботи алгоритмів. Проаналізувати різні методи кодування, як саме вони працюють та в яких випадках кращі за інші. Виконати порівняльний аналіз алгоритмів стиснення зображень та їх методи кодування, визначити сильні та слабі сторони, провести порівняння сучасних алгоритми та визначити в яких задачах вони ефективніші.

4. Перелік питань, що потрібно опрацювати в роботі
Вступ
1. Огляд процесу стиснення та форматів зображень
2. Аналіз алгоритмів стиснення зображень
3. Порівняльний аналіз методів стиснення
Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) _____

Слайди у форматі Power Point (назва та мета роботи, класифікація методів стиснення зображень, методи кодування та їх особливості у стисненні цифрових зображень, порівняльний аналіз алгоритмів стиснення та їх методів кодування, дослідження ліпших випадків для застосування, дослідження ефективності та витрачення ресурсів для реалізації стиснення зображень за допомогою AVIF, JPEG XL, JPEG, висновки)

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Ознайомлення із завданням. Уточнення ТЗ.	23.03.2026	вик.
2	Підбір літератури за темою роботи	26.03 -1.04.2026	вик.
3	Основні поняття стиснення зображень	2.04. -20.04.2026	вик.
4	Аналіз класичних алгоритмів та методів кодування для стиснення зображень	20.04. -5.05.2026	вик.
5	Аналіз результатів досліджень застосування сучасних алгоритмів та методів кодування у різних випадках стиснення зображень	7.05. -11.05.2026	вик.
6	Оформлення презентаційного матеріалу	11.05 - 19.05.2026	вик.
	підготовка до захисту у ЕК		

Дата видачі завдання 23 березня 2026 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Наталія Харченко
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 99 с., 21 рис., 2 табл., 2 додатки, 11 джерел.

Об'єкт роботи – алгоритми стиснення зображень.

Предмет дослідження – оцінювання класичних та сучасних алгоритмів та методів кодування у задачі стиснення зображень.

Основною тенденцією, останнім часом, стало використання нових класів зображень. Старі алгоритми перестали задовольняти вимогам, що пред’являються до стиснення. Багато зображень практично не стискалися, хоча і мали явну надмірність. Це привело до створення нового типу алгоритмів - стискаючих з втратою інформації. Одна з серйозних проблем машинної графіки полягає в тому, що досі не знайдений адекватний критерій оцінки втрат якості зображення. А втрачається вона постійно - при оцифровуванні, при переведенні в обмежену палітру кольорів, при переведенні в іншу систему представлення кольорів для друку, і, що для нас особливо важливе, при стисненні з втратами. Краще всього втрати якості зображень оцінювати візуально, за допомогою зору. Відмінною вважається стиснення, при якому неможливо розрізнити первинне і відновлене зображення. Тому задача полягає у розгляданні класичних та нових методів стиснення зображень, що дозволять досягти цієї мети.

СТИСНЕННЯ ЗОБРАЖЕНЬ, ВТРАТИ ЯКОСТІ ТА ЧІТКОСТІ, МЕТОДИ КОДУВАННЯ, JPEG, JPEG XL, AVIF

ABSTRACT

Abstract: 99 pages, 21 figures, 2 tables, 2 appendices, 11 references.

Subject of the work – algorithms image compression.

Research topic – evaluation of classical and modern algorithms and coding methods in the problem of image compression.

Recently, the use of new classes of images has become a major trend. Old algorithms no longer meet the requirements for compression. Many images were practically uncompressed, even though they clearly contained redundancy. This led to the creation of a new type of algorithm—lossy compression. One of the major problems in computer graphics is that an adequate criterion for evaluating image quality loss has not yet been found. And quality is constantly lost—during digitization, when converting to a limited color palette, when converting to a different color representation system for printing, and, most importantly for us, during lossy compression. The best way to assess image quality loss is visually, using the human eye. Compression is considered excellent when it is impossible to distinguish between the original and the reconstructed image. Therefore, the task is to examine classical and new image compression methods that will allow us to achieve this goal.

IMAGE COMPRESSION, LOSS OF QUALITY AND SHARPNESS, CODING METHODS, JPEG, JPEG XL, AVIF.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

AVIF – AV1 Image File Format

BMP – Bitmap Image File

BRISQUE – Blind/Referenceless Image Spatial Quality Evaluator

CNN – Convolutional Neural Network (згорткова нейронна мережа)

FSIM – Feature Similarity Index

GIF – Graphics Interchange Format

HEIF – High Efficiency Image File Format

HEVC – High Efficiency Video Coding

JPEG – Joint Photographic Experts Group

JPEG XL – Joint Photographic Experts Group XL

LZ77 – алгоритм Лемпеля–Зіва 1977 року

LZW – алгоритм Лемпеля–Зіва–Велча

MSE – Mean Squared Error

MS-SSIM – Multi-Scale Structural Similarity

PIQE – Perceptual Image Quality Evaluator

PNG – Portable Network Graphics

PSNR – Peak Signal-to-Noise Ratio

QOI – Quite OK Image

RGB – Red Green Blue

RGBA – Red Green Blue Alpha

RLE – Run Length Encoding

SSIM – Structural Similarity Index

SVM – Support Vector Machine

ВСТУП

Першими для стиснення зображень стали застосовуватися звичні алгоритми, що використовуються в системах резервного копіювання, при створенні дистрибутивів. Ці алгоритми архівували інформацію без змін. Проте основною тенденцією, останнім часом, стало використання нових класів зображень. Старі алгоритми перестали задовольняти вимогам, що пред'являються до стиснення. Багато зображень практично не стискалися, хоча і мали явну надмірність. Це привело до створення нового типу алгоритмів - стискаючих з втратою інформації. Як правило, коефіцієнт стиснення і, отже, величину втрат якості в них можна задавати. При цьому досягається компроміс між розміром і якістю зображень. Одна з серйозних проблем машинної графіки полягає в тому, що досі не знайдений адекватний критерій оцінки втрат якості зображення. Втрачається вона постійно - при оцифровуванні, при переведенні в обмежену палітру кольорів, при переведенні в іншу систему представлення кольорів для друку, і, що для нас особливо важливе, при стисненні з втратами. Краще всього втрати якості зображень оцінювати візуально, за допомогою зору. Відмінною вважається стиснення, при якому неможливо розрізнити первинне і відновлене зображення. Доброю - коли сказати, яке із зображень піддавалося стисненню, можна тільки порівнюючи дві картинки, що знаходяться поруч. При подальшому збільшенні степені стискання, як правило, стають помітні побічні ефекти, характерні для цього алгоритму. Навіть при відмінному збереженні якості, в зображення можуть бути внесені регулярні специфічні зміни. Тому алгоритми стиснення з втратами не рекомендується використовувати при стисненні зображень, які надалі передбачається друкувати з високою якістю, або обробляти програмами розпізнавання образів.

1 ОГЛЯД ПРОЦЕСУ СТИСНЕННЯ ТА ФОРМАТІВ ЗОБРАЖЕНЬ

Постійна тенденція збільшення роздільної здатності зображення в більшості комерційних і професійних застосунків вимагає постійного розвитку набору рішень, які полегшують ефективне кодування зображень, зменшуючи їх розмір без шкоди для якості. Від найсучаснішого відео 4K і 8K до програм із наднизьким енергоспоживанням, співвідношення між розміром необроблених даних зображення та розміром закодованого зображення вважається одним із основних показників для визначення ефективності кожного формату, оскільки це вирішальний фактор для вимог програми до зберігання та пропускної здатності у випадку обробки в реальному часі. Стиснення зображень – це метод, який використовується для зменшення розміру зображень, що може покращити швидкість завантаження та загальну продуктивність веб-сайту. Методи, які використовуються для стиснення файлів зображень, зазвичай, належать до однієї з двох категорій: із втратами та без втрат [1]. Стиснення із втратами зменшує розмір файлу зображення, остаточно видаляючи менш важливу інформацію, зокрема зайві дані. Стиснення з втратами може значно зменшити розмір файлу, але також може знизити якість зображення до рівня спотворення, особливо якщо зображення надто стиснене. Однак якість можна зберегти, якщо стиснення застосовано ретельно. Ступінь стиснення може відрізнятися залежно від формату зображення, а також тип даних зображення. Вибір відповідного формату для цільового типу даних може стати вирішальним для продуктивності всієї програми. З появою нових форматів зображень, таких як QOI, формат Quite OK Image, разом із безперервним удосконаленням існуючих форматів і реалізацій, вимірювання ефективності кожного формату для наборів даних, що відображають значну дисперсію типів зображень, є корисний посібник для

визначення правильного кодека для сполучення з кожною програмою. Однією з проблем стиснення з втратами є його незворотність [1].

Після того, як його було застосовано до зображення, це зображення ніколи не можна відновити до початкового стану. Якщо стиснення з втратами багаторазово застосовується до того самого зображення, воно дедалі більше спотворюється. Тим не менш, стиснення з втратами даних виявилося цінною стратегією для Інтернету, де часто допускається помірне погіршення якості зображення. Найпоширенішим прикладом стиснення з втратами є JPEG, формат стиснення зображень, який широко використовується в Інтернеті та цифровій фотографії. Одна з великих переваг JPEG полягає в тому, що він дозволяє дизайнеру точно налаштувати ступінь стиснення. Це призводить до кращої якості зображення при правильному використанні, а також призводить до найменшого прийнятного розміру файлу. Оскільки JPEG використовує стиснення з втратами, зображення, збережені в цьому форматі, схильні до «артефактів», коли можна побачити пікселізацію та дивні ореоли навколо певних частин зображення. Найчастіше вони трапляються в областях зображення, де є різкий контраст між кольорами. Як правило, що більший контраст має зображення, то вища якість зображення має бути збережена, щоб отримати гідне кінцеве зображення. Цей загальновизнаний формат підтримується численними інструментами та програмами. Крім того, стиснення можна застосовувати в градусах, що дає змогу використовувати стиснення JPEG, яке найкраще забезпечує баланс між розміром файлу та якістю [1].

Рисунок 1.1 – Стиснене зображення у форматі JPEG

Інший підхід до стиснення зображення називається стиснення без втрат. Цей метод застосовує стиснення без видалення критичних даних або зниження якості зображення, що призводить до стисненого зображення, яке можна відновити до початкового стану без погіршення чи спотворення. Однак стиснення без втрат не зменшує розмір файлу майже так само, як стиснення з втратами, пропонуючи невелику перевагу з точки зору місця для зберігання, пропускної здатності мережі або швидкості завантаження. Стиснення без втрат, зазвичай, використовується в ситуаціях, коли якість зображення важливіша за простір на диску чи продуктивність мережі, наприклад для зображень продуктів або для демонстрації творів мистецтва. Одним із найпоширеніших форматів без втрат є PNG, широко використовуваний формат, який зменшує розмір файлу шляхом визначення шаблонів і стиснення цих шаблонів разом. PNG (Portable Network Graphics) — ще один формат растрового зображення, який використовує стиснення даних без втрат і був створений для заміни формату зображення GIF. Формат PNG тривалий час практично не підтримувався Internet Explorer, через що він використовувався рідше, ніж формати GIF і JPEG, хоча зараз він належним чином підтримується всіма основними браузерами. Файли PNG підтримують кольори на основі палітри (24-розрядний RGB або 32-розрядний RGBA), градації сірого, колірні простори RGBA та RGB. Однією з найбільших переваг PNG є те, що він підтримує низку параметрів прозорості, включаючи прозорість альфа каналу [1].

Рисунок 1.2 – Стиснене зображення у форматі PNG

Хоча файли PNG, як правило, більші за файли JPEG, веб-сайти широко використовують їх, коли потрібно більше деталей зображення, наприклад для логотипів, значків, скріншотів або зображень із текстом. Іншим знайомим форматом без втрат є BMP, власний підхід до стиснення зображень, запроваджений Microsoft і використовується переважно для продуктів Microsoft, зокрема для комп‘ютерів Windows. GIF — це формат стиснення, який відноситься до категорії без втрат, хоча існує певна плутанина щодо того, чи це формат стиснення, чи без втрат. GIF розшифровується як Graphics Interchange Format і є форматом растрових зображень, представленим CompuServe у 1987 році. Він підтримує до 8 біт на піксель, тобто зображення може мати до 256 різних кольорів RGB. Однією з найбільших переваг формату GIF є те, що він дозволяє анімовані зображення. Зображення GIF обмежені 256 кольорами, тому перетворення зображення з більшою кількістю кольорів у GIF призводить до втрати якості, що іноді пояснюється стисненням із втратами. Але алгоритми стиснення, які використовує GIF, є без втрат. Якщо якість втрачається, це пов‘язано з проблемами, пов‘язаними з перетворенням файлу. В даний час формат GIF використовується переважно для простих відео та анімацій [1].

Стиснення flate/deflate Алгоритм стиснення deflate без втрат базується на двох інших алгоритмах стиснення: кодуванні Хаффмана та стисненні LZ77. Потрібно розуміти як працюють ці два алгоритми, щоб зрозуміти стиснення deflate. Deflate — це розумний алгоритм, який адаптує спосіб стиснення даних до самих фактичних даних. Компресор має три режими стиснення: – Зовсім не стиснутий. Це розумний вибір для даних, які вже стиснено. Дані, що зберігаються в цьому режимі, трохи розширяться, але не так сильно, як якщо б вони вже були стиснуті та використали один із інших методів стиснення. – Стиснення, спочатку з LZ77, а потім з трохи модифікованою версією кодування Хаффмана. Дерева, які використовуються для стиснення в цьому режимі, визначаються самою

специфікацією deflate, тому для зберігання цих дерев не потрібно займати додатковий простір. – Стиснення, спочатку за допомогою LZ77, а потім за допомогою дещо модифікованої версії кодування Хаффмана з деревами, які компресор створює та зберігає разом із даними. Дані розбиваються на «блоки», і кожен блок використовує один режим стиснення. Якщо компресор бажає перейти від нестиснутого сховища до стиснення за допомогою дерев, визначених специфікацією, або до стиснення за допомогою вказаних дерев Хаффмана, або до стиснення за допомогою іншої пари дерев Хаффмана, поточний блок має бути завершено та розпочато новий [1].

JPEG означає Joint Photographic Experts Group, яка є комітетом стандартизації. Це також розшифровується як алгоритм стиснення, винайдений цим комітетом. Алгоритм JPEG був розроблений для максимального зменшення розміру файлу природних фотографічних зображень у справжніх кольорах, не впливаючи на якість зображення, яке відчуває сенсорна система людини. Стиснення JPEG відбувається з втратами: коли розпаковується стиснене зображення JPEG і порівнюється результат із вихідним зображенням, яке було стиснено, у розпакованому зображенні будуть відмінності та втрата деталей і різкості. Інші алгоритми стиснення, такі як LZW, працюють без втрат: розпаковане зображення ідентичне вихідному зображенню. Людина легше сприймає невеликі зміни яскравості, ніж невеликі зміни кольору. Людська зорова система також не дуже добре може впоратись з високочастотними даними – якщо деталі зображення змінюються дуже швидко в частині зображення, мозок не може розрізняти чітко всі ці зміни. Ці два аспекти сприйняття використовуються в стисненні JPEG для зменшення розміру файлу зображення. Стиснення JPEG не використовує єдиний фіксований спосіб стиснення даних. Алгоритм може сильно стискати дані зображення зі значною втратою деталей або обмежити втрату деталей шляхом зниження коефіцієнта стиснення. У таких програмах, як Adobe

Photoshop, повзунок дозволяє вибрати ступінь стиснення. Поєднуючи кілька алгоритмів стиснення, JPEG досягає чудових коефіцієнтів стиснення. Можна легко стиснути файл до однієї п‘ятої його початкового розміру. Для веб-публікацій або обміну електронною поштою можна досягти ще кращих співвідношень до 20 до 1. Декомпресія JPEG підтримується в протоколах RIP PostScript рівня 2 і 3. Це означає, що файли меншого розміру можна надсилати через мережу до RIP, що швидше звільняє станцію надсилання, мінімізує витрати на сервер та прискорює RIP [1].

1.1 Алгоритми стиснення

LZ77 LZ77 був оголошений у 1977 році та названий основою багатьох інших алгоритмів стиснення без втрат. Зазвичай він використовує метод «ковзного вікна». У цій методології він піклується про словник, який використовує трійки для представлення: – Зсув – це можна назвати фактичним початком фрази та початком файлу. – Довжина – визначається як кількість символів, які допомагають у створенні фрази. – Відхиляючі символи – це маркетологи, які вказують нову фразу [1].

Він містить вказівку на те, що використана фраза повністю відповідає вихідній фразі, а також визначає, чи є інший символ. Під час аналізу файлу словник динамічно оновлюється для відображення вмісту та розміру стиснутих даних.

Стиснення LZW замінює рядки символів окремими кодами. Він не аналізує вхідний текст. Замість цього він просто додає кожен новий рядок символів, який бачить, до таблиці рядків. Стиснення відбувається, коли замість рядка символів виводиться один код. Код, який виводить алгоритм LZW, може мати будь-яку довільну довжину, але він повинен містити більше бітів, ніж один символ. Перші

256 кодів (при використанні восьмибітних символів) за замовчуванням призначаються стандартному набору символів. Решта кодів призначаються рядкам під час виконання алгоритму. Зразок програми працює, як показано, з дванадцятибітовими кодами. Це означає, що коди 0-255 стосуються окремих байтів, тоді як коди 256-4095 стосуються підрядків [1].

Переваги і недоліки:

- Стиснення LZW найкраще працює для файлів, які містять багато даних, що повторюються. Це часто трапляється з текстом і монохромними зображеннями. Файли, які стиснуті, але взагалі не містять жодної повторюваної інформації, можуть навіть збільшуватися;

- Стиснення LZW відбувається швидко;

- LZW є досить старою технікою стиснення. Усі останні комп‘ютерні системи мають потужність, щоб використовувати більш ефективні алгоритми.

Деякі версії стиснення LZW захищені авторським правом. Їх власник Unisys вимагає роялті від будь-якої компанії, що використовує їхній алгоритм. Це серйозно вплинуло на популярність стиснення LZW, і в довгостроковій перспективі, ймовірно його замінять менш дорогі або безкоштовні алгоритми. Кінцевий користувач не повинен хвилюватися, оскільки лише виробники програмного забезпечення мають платити ліцензійні збори [2].

1.2 Алгоритм стиснення RLE

RLE означає Run Length Encoding. Це алгоритм без втрат, який пропонує пристойні коефіцієнти стиснення лише для певних типів даних. RLE є, мабуть, найпростішим алгоритмом стиснення. Він замінює послідовності однакових значень даних у файлі числом підрахунку та одним значенням. Припустимо, слід стиснути наступний рядок даних (17 байт): ABBBBBBBBBCDEEEEF

Використовуючи стиснення RLE, стиснутий файл займає 10 байт і може виглядати так: A *8B CD *4E F. Як бачимо, рядки даних, що повторюються, замінюються контрольним символом (*), за яким іде кількість повторюваних символів і сам символ, що повторюється. Контрольний характер не є фіксованим, він може відрізнятися від реалізації до реалізації. Якщо сам контрольний символ з‘являється у файлі, кодується один додатковий символ. Кодування RLE є ефективним, лише якщо є послідовності з 4 або більше символів, що повторюються, оскільки три символи використовуються для проведення RLE, тому кодування двох символів, що повторюються, навіть призведе до збільшення розміру файлу. Важливо знати, що існує багато різних схем кодування довжини серії. Наведений вище приклад використовувався для демонстрації основного принципу кодування RLE. Іноді реалізацію RLE адаптують до типу даних, які стискаються. Цей алгоритм дуже простий у реалізації та не вимагає великої потужності процесора. Стиснення RLE ефективне лише для файлів, які містять багато даних, що повторюються. Це можуть бути текстові файли, якщо вони містять багато проміжків для відступів, але штрихові зображення, які містять великі білі або чорні області, є набагато придатнішими. Кольорові зображення, створені комп‘ютером (наприклад, архітектурні креслення), також можуть забезпечувати справедливі коефіцієнти стиснення [1].

1.3 Гібридні та адаптивні методи

Сучасні підходи часто поєднують елементи обох підходів. Наприклад:

- PDF – текстові частини стискаються без втрат, а зображення – з втратами (JPEG);

- HEIF / HEVC – використовують складні моделі перетворень, які адаптивно змінюють параметри стиснення залежно від типу сцени чи змісту.

Класифікація за іншими критеріями

Крім базового поділу на lossless/lossy, у науковій літературі виділяють інші критерії класифікації:

- Статистичні та словникові алгоритми;
- Ентропійні, трансформаційні, прогнозні моделі;
- Блокові та потокові методи (за організацією обробки);
- Символьні, побітові, гібридні (за рівнем обробки).

Кожна з цих класифікацій дає змогу точніше описати принципи роботи алгоритму та його сферу застосування. Наприклад, алгоритм JPEG є трансформаційним, блоковим і з втратами, а Хаффман – статистичний, символьний і без втрат [2].

2 АНАЛІЗ АЛГОРИТМІВ СТИСНЕННЯ ЗОБРАЖЕНЬ

У багатьох випадках, використання сучасного або застарілого алгоритму стиснення зображень, змінюється ефективність вирішення складнощів, котрі виникають у процесі. Фундаментальні засади стиснення даних були закладені в рамках теорії інформації Клода Шеннона. Проте, крім історичного значення, ці ідеї мають безпосереднє практичне застосування при розробці сучасних алгоритмів стиснення. Щоб зрозуміти механізми роботи методів, необхідно розглянути ключові поняття більш глибоко: ентропію, надлишковість, префіксність, структурну повторюваність, а також межі ефективності кодування.

2.1 Ентропія

Ентропія (H) – це математична міра середньої кількості інформації, яку несе окремий символ повідомлення. Якщо всі символи з’являються з однаковою ймовірністю, ентропія максимальна, і повідомлення — найменш стиснене. Формально вона визначається як:

$$H(X) = -\sum p(x) \log_2 p(x), \qquad (2.1)$$

де p(x) — ймовірність появи символу x. Ця формула визначає нижню межу, до якої можливо стиснути дані без втрат інформації [2].

2.2 Надлишковість та повторюваність

Надлишковість – це частина інформації, яка не є абсолютно необхідною для відновлення змісту.

Саме вона є «мішенню» більшості алгоритмів стиснення. Надлишковість може бути:

- статистичною (часті повторення одного символу чи послідовностей — наприклад, «AAAAA»);

- структурною (повторювані шаблони або фрагменти — напр., «the cat and the cat»);

- перцептивною (елементи, які можна ігнорувати з точки зору людського сприйняття — JPEG, MP3).

Наприклад, словникові методи (як-от LZW) не оперують імовірностями, а шукають фрагменти, що повторюються, і замінюють їх кодами. Це ефективно при наявності закономірностей, навіть якщо частоти окремих символів не є показовими [2].

2.3 Коефіцієнт стиснення

Основними технічними характеристиками процесів стиснення та результатів їх роботи є:

- ступінь стиснення об'ємів вхідного і вихідного потоків;

- швидкість стиснення – час, необхідний для стиснення певної кількості інформації вхідного потоку до отримання з нього еквівалентного вихідного потоку;

- якість стиснення – значення, яке показує, наскільки вихідний потік упакований шляхом застосування до нього повторного стиснення за тим самим або іншим алгоритмом [2].

Основною характеристикою алгоритму стиснення даних є коефіцієнт стиснення, тобто величина, яка визначає різницю між обсягом початкових даних і стислих даних і може бути розрахована за такою формулою:

$$k = \frac{S_0}{S_c}, \qquad (2.2)$$

де k - ступінь стиснення, S_0 – обсяг вихідних даних, S_c – обсяг стислих даних.

Очевидно, що вищий ступінь стиснення означає більшу ефективність алгоритму; k=1 означає, що алгоритм не стискає, а k<1 означає, що алгоритм шкідливий, тобто обсяг результуючого повідомлення більший, ніж початковий.

2.4 Визначення якості зображень математичним шляхом

Якість зображення визначається комплексом його характеристик, таких як рівень шуму, деталізація, розмиття та точність кольору. Ці аспекти впливають на загальне враження від зображення та можуть бути суб’єктивними. Однак, для об’єктивних методів оцінки якості зображень, що не потребують експертних оцінок, важливо мати метрики, які відображають сприйняття інформації на зображенні людиною, що допоможе створити автономні системи, які можуть автоматично оцінювати якість зображень. Розробка методів об’єктивної оцінки якості зображень призведе до створення нових алгоритмів обробки зображень, які можуть враховувати різні аспекти людського сприйняття інформації. Ці

методи дозволять ефективніше визначати якість, що, в свою чергу, спростить подальшу обробку зображень [11].

У сфері оцінювання якості зображень існує необхідність в розробці ефективних методів, які б дозволили об’єктивно порівнювати зображення зняті на пристрої з використанням різних сенсорів та різних частот. Особливу увагу слід звернути на розвиток методів, які забезпечують комплексну оцінку якості зображень, враховуючи їхні специфічні особливості, такі як яскравість, тепловий контраст та просторова роздільна здатність. У цій роботі пропонується дослідити ефективність двох методів оцінювання якості зображень – BRISQUE (Blind/Referenceless Image Spatial Quality Evaluator) та PIQE (Perceptual Image Quality Evaluator) – з метою їхнього використання в контексті мультимодального комплексування графічної інформації [11].

2.4.1 Методи оцінювання якості зображень

Метрики оцінювання якості у порівнянні з еталонним зображенням. Оцінювання якості у порівнянні з еталонним зображенням – це процес визначення ступеня схожості або відмінності між двома зображеннями: тестовим (або обробленим) і еталонним (або оригінальним). Цей процес може бути важливим у багатьох областях, включаючи комп’ютерний зір, обробку зображень, а також в галузі оцінки якості відео та зображення [2]. Основні методи оцінки якості зображень включають у себе:

- Метрики розмірності: ці метрики базуються на порівнянні розмірностей або розподілів пікселів між тестовим і еталонним зображеннями;

Прикладами таких метрик є MSE (Mean Squared Error), PSNR (Peak Signal-to-Noise Ratio), SSIM (Structural Similarity Index), MS-SSIM (MultiScale Structural Similarity) і FSIM (Feature Similarity Index) [11];

- Метрики чутливості до спотворень: ці метрики враховують специфічні аспекти спотворень, які можуть бути присутніми в зображенні. Наприклад, у медичній діагностиці може використовуватися метрика, яка приділяє увагу різним типам артефактів, таким як розмиття, шум або артефакти стискання;

- Метрики на основі моделей: деякі методи використовують складні моделі, які шукають взаємозв'язок між еталонним і тестовим зображеннями. Ці моделі можуть бути навчені на основі набору даних з пар зображень, для яких відома якість.

Математично ці метрики можуть бути описані різними способами в залежності від конкретної метрики. Так, MSE може бути визначено як середнє значення квадрата різниці між значеннями пікселів тестового і еталонного зображень. PSNR обчислюється як $10 \cdot \log_{10}(I^{\wedge}(2)_{max}/\text{MSE})$, де Imax – максимальне значення пікселя, відповідає можливому діапазону значень пікселів (наприклад, 255 для 8-бітного зображення) [11].

Взагалі, математичні вирази для цих метрик зазвичай базуються на порівнянні значень пікселів, структурних особливостей та врахуванні внутрішніх параметрів зображення. Використання правильних метрик і розуміння їх математичних особливостей дозволяє об'єктивно оцінити якість зображення порівняно з еталоном.

2.4.2 Метрика BRISQUE

Метод BRISQUE (Blind/Referenceless Image Spatial QUality Evaluator) – це алгоритм оцінювання якості зображень без використання посилань на еталонне зображення. Він базується на аналізі статистичних властивостей зображення та використовує машинне навчання для оцінки його якості [11]. Основний принцип роботи методу BRISQUE наступний:

- Витягування ознак: спочатку зображення проходить через кілька етапів підготовки, таких як конвертація в чорно-біле зображення та розбиття на невеликі блоки. Потім витягуються ознаки з цих блоків, що відображають статистичні особливості зображення;

- Створення вектора ознак: ознаки з кожного блоку об’єднуються в один вектор, який представляє всю інформацію про статистичні властивості зображення;

- Машинне навчання: отриманий вектор ознак використовується для тренування моделі машинного навчання. Зазвичай для цього використовуються алгоритми класифікації, такі як Support Vector Machine (SVM) або регресія, які навчаються передбачати якість зображення на основі його статистичних характеристик;

- Оцінювання якості: після навчання моделі машинного навчання вона може бути використана для передбачення якості нових зображень. Для кожного зображення витягується вектор ознак, після чого модель визначає його якість.

Перевагою методу BRISQUE є: можливість оцінювання якості зображень без наявності еталонного зображення.

Недоліками є: обмежена універсальність (метод ефективний для певного діапазону типів зображень і спотворень) та потреба у великій кількості даних для навчання [11].

Він може бути використаний для автоматичного оцінювання якості зображень у реальному часі, що робить його корисним інструментом у таких областях, як комп’ютерне бачення, обробка зображень та медичне зображення.

2.4.3 Метрика PIQE

Метод PIQE (Perceptual Image Quality Evaluator) – це ще один метод оцінки якості зображень без використання еталонних зображень, який базується на сприйнятті людиною якості зображень. Він використовує моделі сприйняття людиною для прогнозування якості зображень.

Принцип роботи методу PIQE такий:

- Витягування ознак: зображення проходить стадію підготовки, під час якої витягуються різноманітні ознаки, що відображають його специфічні характеристики. Ці ознаки можуть включати структурні, текстурні та колірні характеристики зображення;

- Створення моделі сприйняття людиною: наступним кроком є створення моделі, яка відображає сприйняття людиною якості зображення на основі витягнутих ознак. Ця модель може бути побудована на основі психофізичних досліджень, які визначають, які аспекти зображення найбільше впливають на сприйняття його якості;

- Оцінювання якості: після побудови моделі сприйняття людиною вона може бути використана для прогнозування якості нових зображень.

Для кожного зображення витягуються ознаки, після чого модель оцінює його якість, враховуючи специфічні характеристики зображення.

Деякі переваги методу PIQE включають: він не потребує навченої моделі для оцінювання якості. Крім того, він може бути ефективним для різноманітних типів зображень і спотворень [11].

Проте метод PIQE також має свої обмеження: такі як обмежена універсальність і можливість перекосу результатів через специфічні аспекти моделі сприйняття людиною. Також метод має вдвічі більший час обробки ніж методи з тренованою моделлю.

Порівняння методів BRISQUE та PIQE: Дослідження проводилось на наявному у вільному доступі наборі парних зображень LLVIP у видимому (при слабкому вечірньому освітленні) та інфрачервоному (8-14 мкм) діапазонах, отриманому за допомогою високочутливої двоспектральної камери HIKVISION DS-2TD8166BJZFY-75H2F/V2, яка може виявляти об’єкти на відстані до 7 м та працювати в широкому діапазоні температур. Алгоритми було реалізовано за допомогою бібліотеки PyTorch мови програмування Python. Результати роботи алгоритмів показано на рисунках 2.1 та 2.2. Для оцінки роботи алгоритмів, брався градієнт яскравості зображення, який в свою чергу несе інформацію про імовірні контури загальної картини. Оцінювання зображень в залежності від контурів показані на рисунках 2.3 та 2.4. Проаналізувавши графіки, можна сказати що оцінка за допомогою BRISQUE хаотична в залежності від контурів, що відповідає реальному сприйняттю людиною зображення. Водночас метод PIQE дає монотонне зростання оцінки в залежності від контурів, що може бути кориснішим з технічного погляду [11].

Методи надають об’єктивну оцінку якості зображень без порівняння з еталоном. BRISQUE базується на аналізі статистичних властивостей зображення (оцінених заздалегідь людиною), тому його оцінку можна назвати суб’єктивною, тоді як PIQE використовує моделі сприйняття людиною і дає виключно об’єктивну оцінку. Хоча обидва методи досить ефективні, PIQE, використовуючи моделі сприйняття людиною, надає більш об’єктивне оцінювання якості, оскільки метод враховує особливості сприйняття людьми зображень. BRISQUE може бути менш вимогливим до ресурсів, оскільки він використовує попередньо обчислені ознаки і прості моделі машинного навчання, тоді як PIQE, особливо використовуючи складні моделі сприйняття людиною, може вимагати більше обчислювальних ресурсів. Отже, вибір між цими методами може залежати від конкретного використання та вимог до точності, обчислювальних ресурсів та

універсальності. В обох випадках важливо розуміти обмеження кожного методу і враховувати їх при оцінюванні якості зображень [11].

Рисунок 2.1 – Найкращий результат роботи алгоритмів

Рисунок 2.2 – Найгірший результат роботи алгоритмів

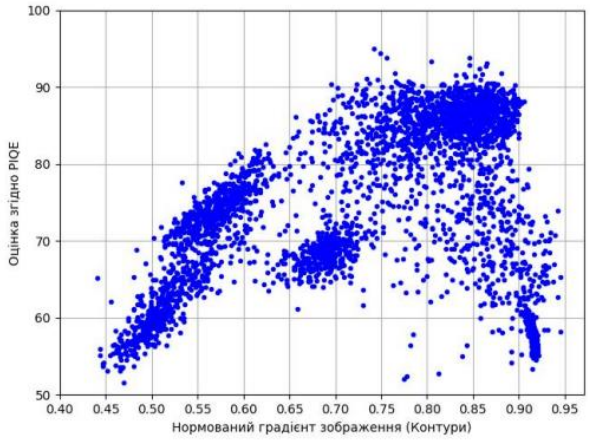

Рисунок 2.3 – BRISQUE

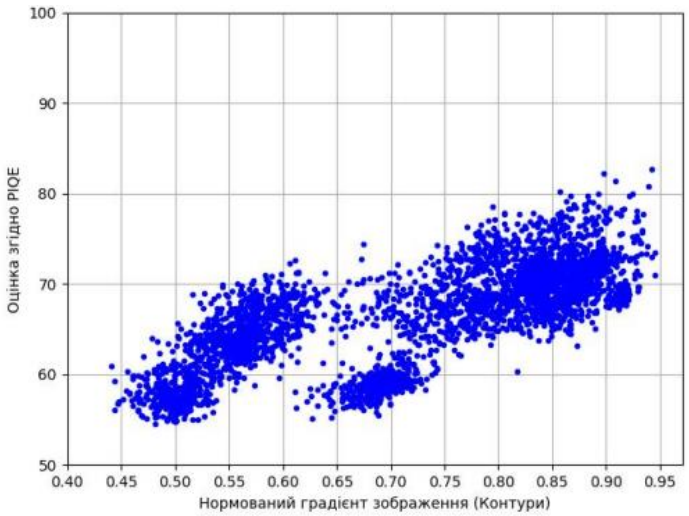

Рисунок 2.4 – PIQE

2.5 Кодування стиснення без втрат

Як вже зазначалось, алгоритми стиснення без втрат (lossless compression) забезпечують збереження даних із повним відновленням оригінальної інформації. Розглянемо принципи роботи двох найбільш поширених методів стиснення без втрат – кодування Хаффмана та алгоритм Лемпеля Зіва–Велча (LZW).

2.5.1 Кодування Хаффмана

Кодування Хаффмана, запропонований у 1952 році Девідом Хаффманом, базується на частотному розподілі символів у вхідних даних і забезпечує мінімальну середню довжину коду для заданих ймовірностей появи символів. Він належить до класу статистичних ентропійних методів і будує так зване префіксне бінарне дерево, де символи з більшою частотою мають коротші коди. Властивості коду Хаффмана [6]:

- Префіксність – жоден з кодів не є префіксом іншого, що забезпечує однозначне декодування без необхідності роздільників;

- Оптимальність – для заданого частотного розподілу код Хаффмана є найкоротшим можливим у межах префіксних кодів (Wayner, 1999);

- Обмеження – ефективність знижується при малій довжині вхідного тексту або рівномірному розподілі символів [2].

Основні етапи побудови коду Хаффмана:

- Підрахунок частоти кожного символу у вхідному повідомленні;

- Створення пріоритетної черги з вузлів, що містять символи та їхні частоти;

- Ітеративне об'єднання двох вузлів з найменшими частотами у новий вузол;

- Побудова бінарного дерева, де ліве ребро позначається "0", праве – "1";

- Присвоєння кожному символу коду, сформованого шляхом від кореня до відповідного листа. На дереві кодування зображено послідовну побудову бінарного дерева Хаффмана для п’яти символів з різною частотою появи у вхідному повідомленні. Алгоритм розпочинається з формування пріоритетної черги з вузлів, кожен з яких містить символ і його частоту. На кожній ітерації з черги вилучаються два вузли з найменшими значеннями частоти, які об'єднуються у новий вузол. Частота нового вузла є сумою частот його дочірніх вузлів. Побудова дерева триває доти, поки в черзі не залишиться лише один вузол – корінь дерева. На схемі показано (рис. 2.1):

- частоти символів на листках дерева (A, B, C, D, E);

- об’єднання пар вузлів із найменшими частотами;

- значення суми частот у внутрішніх вузлах;

- присвоєння бітів кожному ребру дерева (лівий — 0, правий — 1);

- отримані бінарні коди для кожного символу, які формуються шляхом проходження від кореня дерева до відповідного листа.

Так, наприклад, символ A, що має найвищу частоту, отримує найкоротший код – 0, а символи D та E, частота яких становить лише 1, отримують найдовші коди – 1110 та 1111 відповідно. Завдяки префіксності побудованого дерева, жоден з кодів не є префіксом іншого, що дозволяє однозначно розкодовувати дані без роздільників [2].

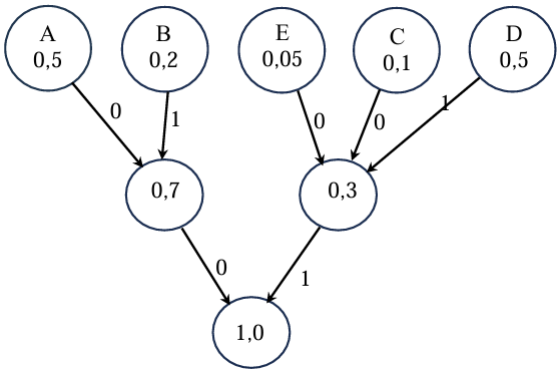

Рисунок 2.5 – Дерево кодування за алгоритмом Хаффмана (5 символів)

2.5.2 Кодування Лемпеля-Зіва-Велча (LZW)

Алгоритм LZW, представлений у 1984 році, є розвитком словникових методів LZ77 та LZ78. Його головна ідея — формування динамічного словника у процесі стиснення, без необхідності попереднього аналізу частот. Це дозволяє забезпечити ефективне кодування при повторенні патернів без втрати інформації.

Переваги алгоритму:

- Адаптивність – словник створюється на льоту, що дозволяє працювати з новими типами даних;

- Ефективність – добре підходить для стиснення текстів, у яких зустрічаються повторювані послідовності;

- Універсальність – не потребує передачі кодової таблиці, як у Хаффмана.

Етапи роботи алгоритму LZW:

- Ініціалізація словника всіма можливими одиничними символами (ASCII);

- Читання вхідного потоку символів та формування ланцюжка (стрічки);

- Якщо ланцюжок є у словнику – продовжити читання;

- Якщо ланцюжок відсутній – записати код попереднього ланцюжка та додати новий до словника;

- Продовження, доки не буде оброблено весь потік.

На рисунку 2.6 зображено загальну схему функціонування алгоритму стиснення без втрат LZW, яка демонструє його основні етапи. Схема побудована на основі теоретичного опису алгоритму, викладеного у класичних джерелах та з адаптацією до логіки, реалізованої в рамках даної роботи. У даній блок-схемі використано такі позначення:

C – символ, зчитаний з вхідного потоку;

W – поточний ланцюжок символів;

W + C – новий ланцюжок, утворений додаванням символа C до ланцюга W;

EOF (End Of File) – ознака завершення вхідного потоку [2].

Схема починається з ініціалізації словника, який містить усі можливі одиничні символи (наприклад, символи таблиці ASCII). Далі зчитується перший символ вхідного потоку C, після чого на кожному кроці формується новий ланцюжок W + C. Якщо цей ланцюжок уже є у словнику, він розширюється, і зчитується наступний символ. Якщо ж W + C відсутній у словнику, до вихідного потоку записується код W, новий ланцюжок додається до словника, а W оновлюється як C. Процес повторюється до досягнення кінця вхідного потоку EOF, після чого виводиться останній код, і алгоритм завершується [2].

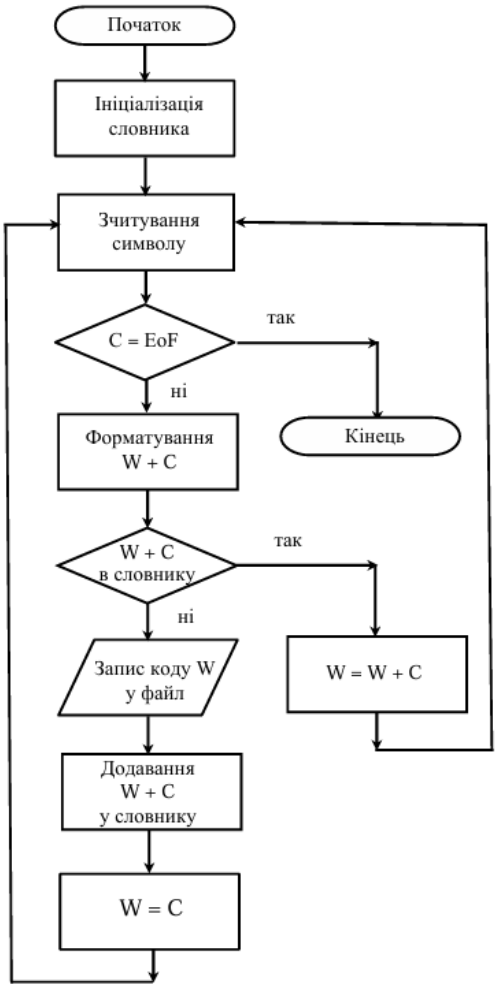

Рисунок 2.6 – Схема роботи алгоритму LZW

2.5.3 Арифметичне кодування

На відміну від кодів змінної довжини, арифметичне кодування генерує неблокові коди. В арифметичному кодуванні, яке бере свій початок у роботах Еліаса, однозначної відповідності між символами вихідного сигналу та кодовими словами не існує [6].

Натомість усій послідовності символів вихідного сигналу (або повідомленню) присвоюється одне арифметичне кодове слово. Саме кодове слово визначає інтервал дійсних чисел від 0 до 1. У міру збільшення кількості символів у повідомленні інтервал, що використовується для його представлення,

стає меншим, а кількість одиниць інформації (скажімо, бітів), необхідних для представлення інтервалу, стає більшою. Кожен символ повідомлення зменшує розмір інтервалу відповідно до ймовірності його появи.

Оскільки ця техніка не вимагає, як підхід Хаффмана, щоб кожен символ джерела перетворювався на ціле число кодових символів (тобто щоб символи кодувалися по одному), вона досягає (але тільки в теорії) межі, встановленої теоремою про безшумне кодування [5].

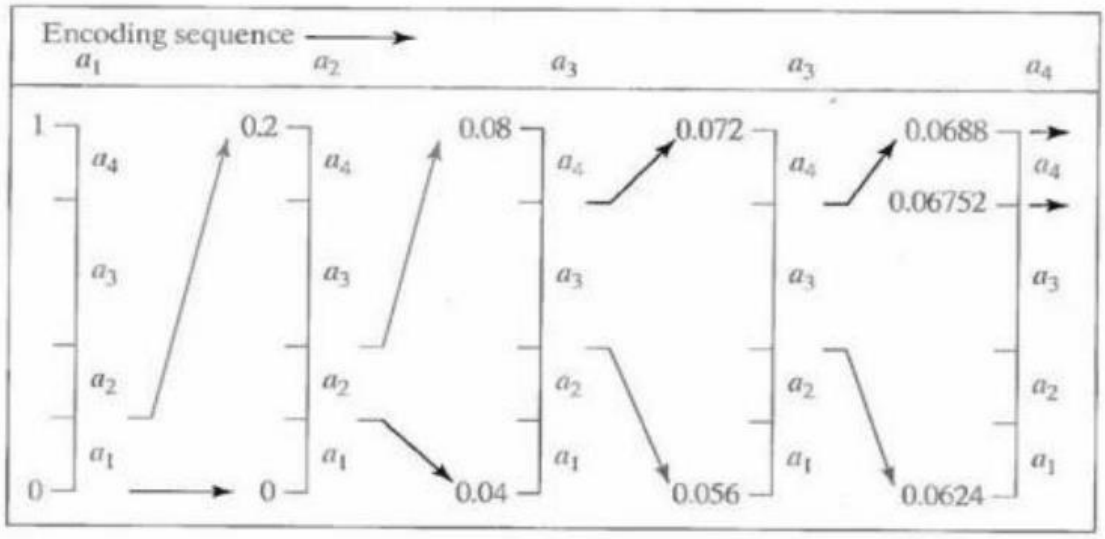

Рисунок 2.7 – Процедура арифметичного кодування

На рисунку зображено основний процес арифметичного кодування. Тут кодується послідовність із п’яти символів, або повідомлення, a1a2a3a3a4, отримане з джерела, що складається з чотирьох символів.

На початку процесу кодування припускається, що повідомлення займає весь напіввідкритий інтервал [0, 1). Цей інтервал спочатку поділяється на чотири області на основі ймовірностей кожного символу джерела.

Наприклад, символ ах пов'язаний з підінтервалом [0, 0,2). Оскільки це перший символ повідомлення, що кодується, інтервал повідомлення спочатку звужується до [0, 0,2).

Таким чином, на рис. [0, 0,2) розширюється до повної висоти фігури, а його кінцеві точки позначаються значеннями звуженого діапазону. Потім звужений діапазон поділяється відповідно до початкових ймовірностей символів джерела, і процес продовжується з наступним символом повідомлення [5].

Source Symbol	Probability	Initial Subinterval
a_1	0.2	[0.0, 0.2)
a_2	0.2	[0.2, 0.4)
a_3	0.4	[0.4, 0.8)
a_4	0.2	[0.8, 1.0)

Рисунок 2.8 – Приклад арифметичного кодування

Таким чином, символ а2 звужує підінтервал до [0,04; 0,08), а3 – ще більше, до [0,056; 0,072) і так далі. Останній символ повідомлення, який має бути зарезервований як спеціальний індикатор кінця повідомлення, звужує діапазон до [0,06752; 0,0688). Звичайно, будь-яке число в межах цього підінтервалу – наприклад, 0,068 — може бути використане для представлення повідомлення. В арифметично закодованому повідомленні три десяткові цифри використовуються для представлення повідомлення з п’яти символів.

Це перетворюється на 3/5 або 0,6 десяткових цифри на символ джерела і вигідно відрізняється від ентропії джерела, яка становить 0,58 десяткових цифри або 10-арі одиниць/символ. У міру збільшення довжини послідовності, що

кодується, отриманий арифметичний код наближається до межі, встановленої теоремою про безшумне кодування [5].

На практиці два фактори призводять до того, що ефективність кодування не досягає цієї межі:

- додавання індикатора кінця повідомлення, необхідного для відокремлення одного повідомлення від іншого;

- використання арифметики з кінцевою точністю. Практичні реалізації арифметичного кодування вирішують останню проблему шляхом впровадження стратегії масштабування та стратегії округлення (Langdon і Rissanen [1981]).

Стратегія масштабування перенормує кожен підінтервал до діапазону (0, 1) перед його поділом відповідно до ймовірностей символів. Стратегія округлення гарантує, що обрізання, пов'язані з арифметикою скінченної точності, не заважають точному представленню підінтервалів кодування [6].

2.5.4 Кодування зі змінною довжиною

Найпростіший підхід до безпомилкового стиснення зображень полягає у зменшенні лише кодової надмірності. Кодова надмірність зазвичай присутня в будь-якому природному двійковому кодуванні рівнів сірого в зображенні. Її можна усунути шляхом кодування рівнів сірого. Для цього потрібно побудувати код зі змінною довжиною, який присвоює найкоротші можливі кодові слова найімовірнішим рівням сірого. На практиці символами джерела можуть бути або рівні сірого зображення, або результат операції відображення рівнів сірого (різниці пікселів, довжини послідовностей тощо).

2.5.5 Кодування бітових площин

Ефективним методом зменшення міжпіксельної надмірності зображення є окрема обробка бітових площин зображення. Цей метод, який називається кодуванням бітових площин, ґрунтується на концепції розкладання багаторівневого (монохромного або кольорового) зображення на серію бінарних зображень та стиснення кожного бінарного зображення за допомогою одного з декількох відомих методів бінарного стиснення [5].

Розклад на бітові площини: Рівні сірого m-бітного зображення в сірій гамі можна представити у вигляді полінома з основою 2. Виходячи з цієї властивості, простим методом розкладу зображення на набір бінарних зображень є розділення m коефіцієнтів полінома на m 1-бітних бітових площин. Бітова площина нульового порядку генерується шляхом збору a0 бітів кожного пікселя, тоді як бітова площина (m - 1)-го порядку містить am-1 бітів або коефіцієнтів. Загалом, кожна бітова площина нумерується від 0 до m-1 і будується шляхом прирівнювання її пікселів до значень відповідних бітів або коефіцієнтів полінома з кожного пікселя вихідного зображення. Недоліком цього підходу є те, що невеликі зміни рівня сірого можуть мати значний вплив на складність бітових площин.

Якщо, наприклад, піксель з інтенсивністю 127 (01111111) межує з пікселем з інтенсивністю 128 (10000000), то кожна бітова площина міститиме відповідний перехід від 0 до 1 (або від 1 до 0). Наприклад, оскільки найстарші біти двох двійкових кодів для 127 і 128 різні, бітова площина 7 міститиме піксель із нульовим значенням поруч із пікселем із значенням 1, створюючи перехід від 0 до 1 (або від 1 до 0) у цій точці [5].

Альтернативний підхід до розкладу (який зменшує вплив невеликих коливань рівня сірого) полягає в тому, щоб спочатку представити зображення за

допомогою m-бітного коду Грея. m-бітний код Грея gm-1...g2g1g0, що відповідає поліному у рівнянні вище, можна обчислити з

$$g_i = a_i \oplus a_{i+1} \quad 0 \le m-2$$
$$g_{m-1} = a_{m-1}. \qquad (2.3)$$

Тут позначає операцію «або» (XOR). Цей код має унікальну властивість: послідовні кодові слова відрізняються лише в одній бітовій позиції. Таким чином, незначні зміни рівня сірого рідше впливають на всі m бітових площин. Наприклад, коли рівні сірого 127 і 128 є сусідніми, лише 7-й бітний рівень міститиме перехід від 0 до 1, оскільки коди Грея, що відповідають 127 і 128, становлять 11000000 та 01000000 відповідно [5].

2.5.6 Безвтратне прогнозне кодування

Підхід до стиснення без втрат не вимагає розкладання зображення на набір бітових площин. Цей підхід, який зазвичай називають прогнозним кодуванням без втрат, базується на усуненні міжпіксельної надмірності для пікселів, розташованих близько один до одного, шляхом виокремлення та кодування лише нової інформації в кожному пікселі. Нова інформація пікселя визначається як різниця між фактичним і прогнозованим значенням цього пікселя. На малюнку показано основні компоненти системи прогнозного кодування без втрат.

Система складається з кодера та декодера, кожен з яких містить ідентичний прогнозувач. Коли кожен наступний піксель вхідного зображення, позначений fn, подається до кодера, прогнозувач генерує очікуване значення цього пікселя на основі певної кількості попередніх вхідних даних [5].

Вихідні дані предиктора потім округлюються до найближчого цілого числа, позначаються як f^n і використовуються для формування різниці або помилки прогнозування, яка кодується за допомогою коду змінної довжини (символьним кодером) для генерації наступного елемента потоку стиснених даних.

$$e_n = f_n - \hat{f}_n,$$

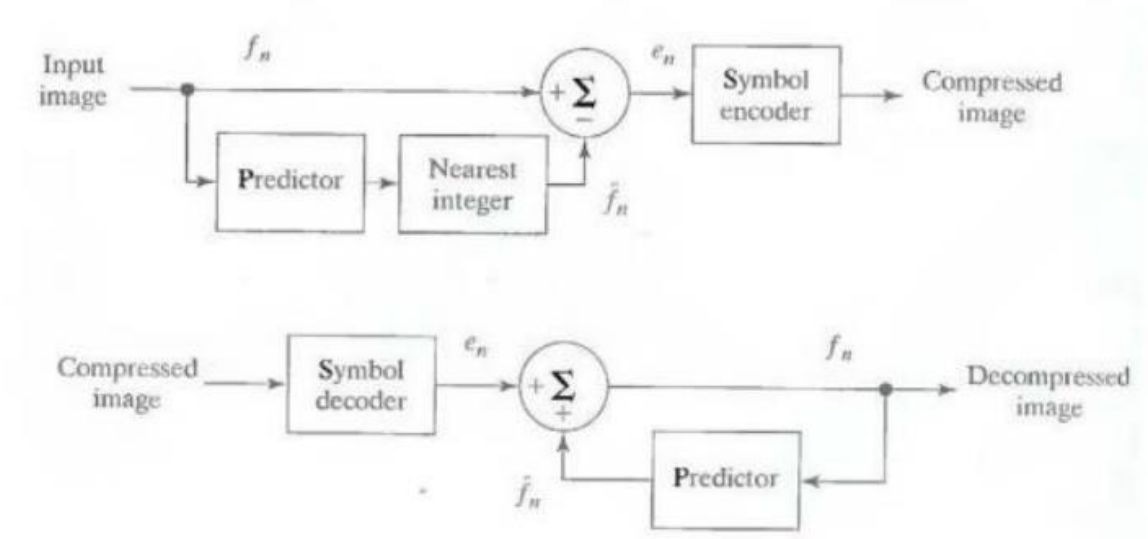

Рисунок 2.9 – Модель кодування з прогнозуванням без втрат: (а) кодер; (б) декодер

Декодер відтворює сигнал en на основі отриманих кодових слів змінної довжини та виконує обернену операцію.

$$f_n = c_n + \hat{\hat{f}}_n. \qquad (2.4)$$

Для обчислення f^n можна використовувати різні локальні, глобальні та адаптивні методи. Однак у більшості випадків прогноз формується як лінійна комбінація m попередніх пікселів. Тобто:

$$\hat{f}_n = round\,[\sum_{i=1}^{m} a_i f_{n-i}], \qquad (2.5)$$

де m – порядок лінійного предиктора, round – функція, що використовується для позначення операції округлення або знаходження найближчого цілого числа, а αi для i = 1, 2, ..., m – коефіцієнти прогнозування. У додатках растрового сканування індекс n індексує вихідні дані предиктора відповідно до часу їх виникнення. Тобто fn, f^n та en у рівняннях вище можна замінити на більш явні позначення f (t), f^(t) та e (t), де t позначає час. В інших випадках n використовується як індекс просторових координат та/або номера кадру (у часовій послідовності зображень) зображення. Наприклад, у 1-вимірному лінійному прогнозному кодуванні рівняння вище можна записати як

$$\hat{f}_n(x,y) = round\,[\sum_{i=1}^{m} a_i f(x,y-i)], \qquad (2.6)$$

де кожна змінна з індексом тепер явно виражається як функція просторових координат x та y. Рівняння вказує, що одновимірне лінійне прогнозування f(x, y) є функцією лише попередніх пікселів на поточній лінії. У двовимірному прогнозному кодуванні прогноз є функцією попередніх пікселів при скануванні зображення зліва направо та зверху вниз [5].

У тривимірному випадку воно базується на цих пікселях та попередніх пікселях попередніх кадрів. Вираз вище не можна обчислити для перших m пікселів кожного рядка, тому ці пікселі необхідно кодувати за допомогою інших засобів (таких як код Хаффмана) і розглядати як накладні витрати процесу прогнозного кодування. Аналогічне зауваження стосується випадків з більшою кількістю вимірів [5].

2.6 Порівняльна характеристика методів кодування

Для кращого розуміння доцільності застосування тих чи інших методів стиснення, а також їхньої ефективності в конкретних умовах, варто здійснити порівняльний аналіз алгоритмів без втрат і з втратами, зокрема – Хаффмана, LZW та JPEG. Ці три алгоритми були обрані як основа дослідження у межах роботи через їхню поширеність, практичне значення та відмінності у принципах побудови.

Таблиця 2.1 – Порівняння методів кодування

Кодування	Тип стиснення	Принцип	Втрати	Застосування
Хаффмана	Без втрат	Статистичне префіксне дерево	Ні	Текст, код, дані з повтореннями
LZW	Без втрат	Словникове динамічне кодування	Ні	Текстові файли, GIF, TIFF
Арифметичне	З втратами ; Без втрат	Адаптивне дискретне косинусове перетворення / модульний режим кодування	В залежності від режиму	Цифрові зображення, фото
Бітових площин	Без втрат	Розкладання зображення на бітові площини та окреме стиснення кожної бінарної площини.	Ні	Растрові зображення, JPEG 2000, BMP, TIFF
Прогнозне	Без втрат	Усунення міжпіксельної надмірності для пікселів, розташованих близько один до одного	Ні	Супутникові зображення, медичні знімки, PNG, JPEG-LS

Алгоритм Хаффмана показує найкращі результати при стисканні текстів або даних із відомою статистикою символів. Він гарантує оптимальну довжину коду за наявності точного частотного розподілу. Проте його ефективність знижується при стисканні коротких текстів або вже стислих даних. До того ж, при рівномірному розподілі символів алгоритм майже не дає виграшу.

LZW не вимагає попереднього аналізу частот і будує словник під час обробки потоку.

Завдяки цьому він добре працює з повторюваними структурами, особливо у текстах і форматах, де повтори поширені (наприклад, HTML, XML, JSON). Саме адаптивність LZW зробила його базовим для таких форматів, як GIF і TIFF.

JPEG принципово відрізняється від Хаффмана та LZW. Це алгоритм з втратами, який перетворює зображення у частотну область, і шляхом квантування відкидає незначущі високочастотні компоненти.

Саме здатність до компресії без видимих спотворень зробила JPEG стандартом для зберігання цифрових зображень. Однак, у випадках, коли потрібне точне відновлення (наприклад, у медичній візуалізації), JPEG є непридатним.

3 ПОРІВНЯЛЬНИЙ АНАЛІЗ МЕТОДІВ СТИСНЕННЯ

3.1 Алгоритми стиснення зображень

Методи стиснення з втратами (lossy compression) застосовуються там, де допустиме певне зниження якості даних в обмін на істотне скорочення їх обсягу. У цьому підрозділу детально розглянуто алгоритм JPEG як один із найпоширеніших стандартів стиснення зображень із втратами.

JPEG є класичним прикладом алгоритму з втратами, який поєднує перетворення, квантування та ентропійне кодування, що дозволяє розглядати його як модель для розуміння принципів стиснення зображень.

По друге, алгоритм широко використовується на практиці – від цифрових фотоапаратів до веб-сайтів і PDF-документів.

По-третє, реалізація JPEG передбачає роботу з матрицями, частотними перетвореннями та статистичними моделями, що створює добру основу для поєднання теорії та практики [2].

Структура та етапи роботи алгоритму JPEG: JPEG базується на блочному перетворенні зображення і включає такі основні етапи:

- Перетворення кольору з RGB до YCbCr – з метою відокремлення яскравісної компоненти (Y) від кольорових (Cb, Cr);

- Даунсемплінг кольорових компонент – зменшення роздільної здатності Cb та Cr, що мінімально впливає на сприйняття, наприклад, у форматі 4:2:0;

- Розбиття зображення на блоки 8×8 пікселів;

- Дискретне косинусне перетворення (DCT) – перетворення просторового сигналу в частотний, що дозволяє виділити основні частоти зображення;

- Квантування – округлення коефіцієнтів DCT з урахуванням таблиць чутливості, втрачаючи деталі, які складно помітити візуально;

- Зигзаг-сканування – перетворення блоку в одновимірну послідовність, де високочастотні нулі групуються;

- Кодування довжин серій (RLE) – заміна послідовностей нулів на компактні представлення;

- Хаффманівське кодування – ентропійне стиснення з використанням префіксних кодів;

- Формування бітового потоку – запис усіх закодованих даних у файл JPEG. У результаті досягається ефективне стиснення (5–20 разів), при якому втрати інформації залишаються непомітними для людського ока [2].

Схема роботи алгоритму JPEG

Рисунок 3.1 – Схема роботи алгоритму JPEG

3.2 Сучасний алгоритм стиснення JPEG XL

Формат JPEG XL був розроблений з метою досягти або перевершити всі попередні растрові формати зображень як за функціональністю, так і за ефективністю стиснення. Щодо функціональності, JPEG XL може робити все те саме, що й JPEG, PNG, GIF, TIFF, BMP, WebP, AVIF та HEIC, а також набагато більше. Фактично, це усуває необхідність вибору відповідного формату зображення для конкретного завдання. Він також підходить як кодек корисного навантаження в різних форматах для конкретних застосувань, включаючи DNG, DICOM та формати GIS [10].

Що стосується стиснення, JPEG XL пропонує найсучаснішу продуктивність, як без втрат, так і з втратами, у широкому діапазоні якості, скорочуючи витрати на зберігання та пропускну здатність удвічі порівняно з JPEG [3].

Унікальною особливістю JPEG XL є його здатність без втрат перекомпресувати існуючі зображення JPEG, зменшуючи розміри файлів — а отже, і витрати на передачу та зберігання — приблизно на 20%, дозволяючи при цьому точне відтворення, байт за байтом, оригінального файлу JPEG. Це можливо, оскільки інструменти кодування JPEG XL є надмножиною інструментів, доступних у JPEG. Це допомагає зберегти нашу цифрову спадщину, оскільки в процесі переходу від JPEG до JPEG XL не відбувається втрати якості (додаткові артефакти стиснення, спричинені транскодуванням із втратами).

Серед основних переваг JPEG XL:

-покращена точність і відтворення кольорів: широка колірна гама, високий динамічний діапазон без обмежень щодо точності;

-додаткові канали: підтримка прозорості альфа-каналу, карт глибини, CMYK, плашкових кольорів, масок виділення, мультиспектральних зображень;

-багатошарові зображення, анімації та багатосторінкові зображення;

-істотно покращене стиснення: економія близько 50 % як при стисненні зображень із втратами, так і без втрат;

-рекомпресія JPEG без втрат, економія близько 20% при перенесенні існуючих зображень, повністю без ризику [10].

Мета JPEG XL досить амбітна: він покликаний стати універсальним кодеком зображень без ліцензійних відрахувань, який може стати універсальним форматом обміну для всіх випадків використання нерухомих зображень, починаючи від зйомки та створення зображень і закінчуючи їх розміщенням у мережі. Що стосується функціональних можливостей та стиснення, він має на меті об’єднати та замінити різні існуючі формати зображень, включаючи JPEG, JPEG 2000, JPEG XR, PNG, GIF, WebP, AVIF, HEIC, TIFF та OpenEXR.

На відміну від WebP та AVIF, які зосереджуються на конкретному випадку використання – веб-доставці з «веб-якістю», JPEG XL орієнтований на широкий спектр випадків використання. Це включає стиснення без втрат та дуже високої якості, як того вимагають робочі процеси, де передбачається подальше редагування. Він також орієнтований на друк (наприклад, CMYK та плашкові кольори), наукові та медичні застосування (наприклад, мультиспектральні та високоточні зображення), великі зображення та різні компроміси між зусиллями кодування та стисненням [10].

Формат JPEG XL визначено у стандарті ISO/IEC 18181. Цей стандарт складається з чотирьох частин:

- 18181-1: Основний потік коду;

- 18181-2: Формат файлу;

- 18181-3: Тестування на відповідність;

- 18181-4: Еталонне програмне забезпечення [3].

Потік коду

Основний потік коду містить усі дані, необхідні для декодування та відображення нерухомого зображення (можливо з декількома шарами) або анімації. Окрім самих даних пікселів, це також включає основні метадані, такі як розміри зображення, інформація про колірний простір, орієнтація, підвищення частоти дискретизації, змішування кадрів або шарів тощо. Варто зазначити, що в цьому відношенні JPEG XL відхиляється від традиційних конструкцій JPEG, де потік коду лише відтворює значення зразків на регулярній сітці дискретизації, не несучи жодної інтерпретації. JPEG та JPEG 2000 дотримуються цієї архітектури, де, наприклад, інформація про колірний простір надається лише на рівні формату файлу. Однак у JPEG XL кодовий потік передає не лише значення зразків, а й їх колориметричну інтерпретацію (див. також розділ 4) та будь-які інші метадані, необхідні для правильного відображення зображення [3].

Формат файлу

Формат файлу JPEG XL може мати дві форми:

«сирий» потік коду: у цьому випадку зберігаються лише самі дані зображення, а додаткові метадані не можуть бути включені. Такий файл починається з байтів 0xFF0A (маркер JPEG, що означає «початок потоку коду JPEG XL»).

Контейнер типу ISOBMFF: це контейнерний формат на основі блоків, який містить блок кодового потоку JPEG XL (jxlc) і може опціонально містити інші блоки з додатковою інформацією, такою як метадані Exif. У цьому випадку файл починається з байтів 0x0000000C4A584C20 0D0A870A.

Архітектура кодека

JPEG XL чітко розмежовує дані зображення та метадані. Все, що потрібно для коректного відображення зображення (або анімації), вважається даними зображення і є частиною основного потоку коду [3].

Сюди входять елементи, які традиційно вважалися «метаданими», такі як профілі ICC та орієнтація Exif, оскільки вони є необхідними для коректного відображення зображення. Будь-який інший вид метаданих (такий як заяви про авторські права або GPS-координати) вважається не частиною даних зображення і не частиною основного кодового потоку. Метою є зменшення неоднозначності та ймовірності неправильних реалізацій, які можуть бути спричинені наявністю «чорного ящика» потоку коду, що містить лише числові дані про пікселі, залишаючи за програмами завдання з’ясувати, як правильно інтерпретувати дані (тобто застосовувати перетворення кольорів, збільшення роздільної здатності, орієнтацію, змішування, обрізання тощо).

Включаючи цю функціональність у сам кодовий потік, декодер може за замовчуванням надавати вихідні дані у нормалізованому вигляді (наприклад, у форматі RGBA, з уже застосованою орієнтацією, злиті та об’єднані кадри), спрощуючи завдання правильного відтворення зображень з точки зору додатка шляхом обробки схильних до помилок підзадач, таких як злиття кадрів або застосування орієнтації зображення, у бібліотеці декодування, а не залишаючи їх для реалізації на рівні додатка. Знання колориметричної інтерпретації зразка даних також дозволяє кодеру з втратами приймати кращі рішення та отримувати більш послідовні результати для даного налаштування якості [3].

Решта метаданих, наприклад Exif або XMP, може зберігатися у форматі контейнера, але це не впливає на основне відтворення зображення (хоча може описувати альтернативні варіанти відтворення).

У випадку орієнтації Exif це поле повинно ігноруватися додатками, оскільки орієнтація в кодовому потоці завжди має пріоритет (і вже буде прозоро застосована декодером).

Це також означає, що видалення метаданих може здійснюватися без впливу на зображення, що відображається.

На рисунку наведено 3.2 навведено загальний огляд архітектури кодека JPEG XL. У верхній частині цієї діаграми показано сторону кодера; сторона декодера знаходиться внизу і здебільшого опущена для стислості, оскільки складається з тих самих компонентів, що застосовуються у зворотному порядку: наприклад, ентропійне декодування, потім застосування оберненого DCT, відтворення характеристик зображення, скасування колірного перетворення [3].

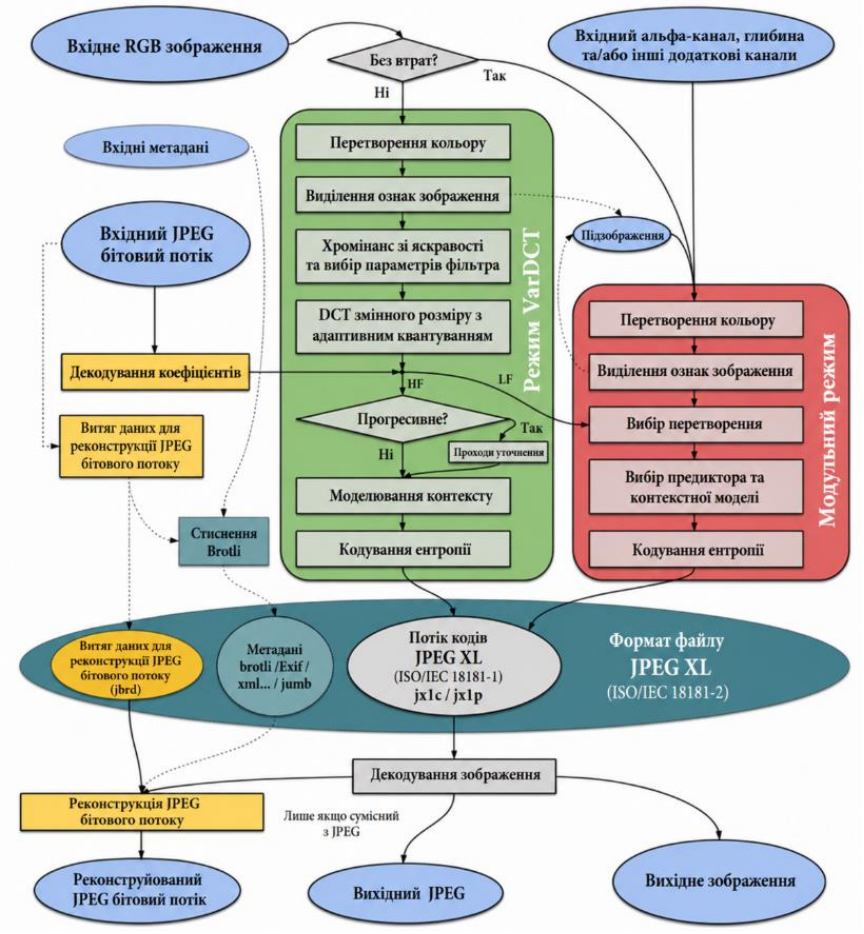

Рисунок 3.2 – Огляд архітектури кодека JPEG XL.

Функції потоку коду

Потік коду містить одне зображення, яке має певні розміри (до 2^30 × 2^30) та складові: основні дані зображення мають або одну (у відтінках сірого), або три (кольорові) складові, а також може містити до 4096 додаткових складових, відомих як «додаткові канали». Розміри зображення визначають розмір (у пікселях) прямокутного полотна.

Зображення складається з одного або декількох кадрів, що відтворюються на цьому полотні. Кадри можуть представляти кадри анімації, шари складеного нерухомого зображення або окремі сторінки багатосторінкового зображення. Кожен кадр містить дані пікселів (вибірки для кожного компонента), які кодуються або у режимі VarDCT, або у модульному режимі. Дані поділяються на групи, які можна кодувати незалежно, що дозволяє здійснювати паралелізацію та декодування областей інтересу.

Кадри

Потік коду JPEG XL містить один або кілька кадрів. У разі анімації ці кадри мають тривалість і можуть повторюватися (нескінченно або певну кількість разів). Можливі кадри з нульовою тривалістю, які представляють різні шари зображення. Кадри з максимальною тривалістю вказують на «розриви сторінок» у багатосторінковому зображенні.

Кадри мають режим накладання (Замінити, Додати, Альфа-накладання, Множення тощо) і можуть використовувати будь-який попередній кадр як основу. Вони можуть бути меншими за полотно зображення, у цьому випадку пікселі за межами обрізки копіюються з базового кадру. Кадри можна розміщувати з довільним зміщенням від полотна зображення. Це зміщення також може бути від’ємним, а кадри можуть бути більшими за полотно зображення, у цьому випадку частини кадру будуть невидимими, і відображатиметься лише перетин із полотном зображення [3].

Також можливі невидимі кадри «ReferenceOnly», які не візуалізуються, але можуть використовуватися як основа для змішування кадрів або як джерело для патчів.

За замовчуванням декодери змішують і об'єднують кадри, створюючи лише один вихідний кадр, коли є наступні кадри з нульовою тривалістю. В результаті всі вихідні кадри мають однаковий розмір (розмір полотна зображення) і мають або нульову тривалість (у випадку нерухомого зображення), або ненульову тривалість (у випадку анімації). Це призводить до того, що багатошарові зображення відображаються в програмах перегляду як одне об'єднане зображення, обрізане до розмірів полотна зображення, тоді як інструменти створення все ще можуть декодувати окремі шари. Це також спрощує реалізацію програм перегляду анімації, оскільки їм не потрібно реалізовувати злиття кадрів на рівні програми.

Дані пікселів

Кожен кадр містить дані пікселів, закодовані в одному з двох режимів:

-Режим VarDCT: у цьому режимі застосовуються перетворення DCT зі змінним розміром, а дані зображення кодуються у формі коефіцієнтів DCT. Цей режим завжди є втратним. У разі, якщо потік коду JPEG XL було створено шляхом перекодування зображення, закодованого у форматі JPEG, у синтаксис JPEG XL, використовується лише підмножина функцій JPEG XL, наприклад, застосовується лише DCT 8x8. Такий кодовий потік JPEG XL дозволяє відновити оригінальне зображення JPEG, на основі якого він був створений. Оскільки JPEG XL забезпечує вдосконалені методи ентропійного кодування, кодовий потік JPEG XL, як правило, матиме менший розмір, навіть якщо він представляє точно те саме зображення;

-Модульний режим: У цьому режимі використовується лише цілочисельна арифметика, що забезпечує стиснення без втрат [3].

Однак цей режим також можна використовувати для стиснення з втратами. Для покращення стиснення або отримання інших бажаних ефектів можна використовувати кілька перетворень: оборотні колірні перетворення (RCT), (дельта) палітрові перетворення та модифіковане нелінійне перетворення Гаара, яке називається Squeeze, що полегшує (але не вимагає) стиснення з втратами та забезпечує прогресивне декодування.

Ці режими кодування не є взаємовиключними. Три основні колірні компоненти (або компонента відтінків сірого) кодуються в одному з двох режимів; будь-які додаткові компоненти (відомі як «додаткові канали»), такі як прозорість альфа, завжди кодуються в модульному режимі. Крім того, внутрішньо режим VarDCT використовує модульні підпотоки для різних допоміжних зображень: «LF-зображення» (зменшена у співвідношенні 1:8 версія зображення, що містить коефіцієнти постійної складової DCT8x8 та низькочастотні коефіцієнти більших DCT-перетворень) та ваги для адаптивного квантування [3].

В обох режимах додаткові, окремо кодовані «особливості зображення» накладаються на декодоване зображення:

-Патчі: прямокутники з попередньо декодованого кадру (зазвичай кадру ReferenceOnly) можна накласти за допомогою одного з режимів накладання на поточний кадр.

Це дозволяє кодеру ідентифікувати повторювані візерунки (такі як літери тексту) і кодувати їх лише один раз, використовуючи патчі для вставки візерунка в декількох місцях. Ці візерунки кодуються в попередньому кадрі, що дозволяє поєднувати пікселі, закодовані за допомогою Modular, з кадром, закодованим за допомогою VarDCT, або навпаки. Лука Версарі розробив підсистему патчів.

-Сплайни: можна кодувати доцентрові сплайни Катмулла-Рома, з кольором і товщиною, що можуть змінюватися вздовж довжини дуги кривої.

Хоча сучасні кодери поки що не використовують цю функцію потоку бітів, ми вважаємо, що вона може бути корисною для доповнення даних, закодованих за допомогою DCT, оскільки тонкі лінії важко точно відтворити за допомогою DCT. Самі Букортт і Юркі Алакуйяла спільно розробили підсистему сплайнів. Наприклад, окремі пасма волосся, тонкі гілки дерева або лінійні малюнки можна кодувати ефективніше за допомогою сплайнів;

-Шум: до зображення можна додати синтетичний шум із модуляцією яскравості, наприклад, для імітації фотонного шуму, таким чином, щоб уникнути поганої компресії через високочастотні коефіцієнти DCT. Йіркі Алакуйяла розробив підсистему фотонного шуму. Підсистема фотонного шуму JPEG XL відрізняється від більш складних інструментів «синтезу зернистості плівки», оскільки фотонний шум працює через лапласівський фільтр [3].

Це унеможливлює введення підсистемою фотонного шуму низькочастотних особливостей або випадкових скупчень шуму. У фізичних вимірюваннях шум знімка пов’язаний із квадратним коренем сили сигналу.

Однак, з огляду на психовізуальне стиснення, це може означати, що інтенсивність шуму є вкрай нелінійною. У JPEG XL ми вирішуємо цю проблему, надаючи невелику таблицю відповідності для вираження залежності інтенсивності шуму. У стандартах AVIF та MPEG подібні функції називаються «синтезом зернистості плівки». У JPEG XL не було зроблено жодних спроб моделювання зернистості плівки, але модель шуму є просто більш простим шумом знімка. І синтез зернистості плівки, і модель фотонного шуму сприяють стисненню, роблячи плоскі або рівні ділянки зображення більш природними.

Нарешті, обидва режими також можуть опціонально застосовувати два методи фільтрації до декодованого зображення, які мають на меті зменшення блокових артефактів та дзвінкості:

-перетворення типу Габора («Gaborish»): невелике (3x3) розмиття, яке застосовується поперек меж блоків та груп, зменшуючи блоковість.

Кодер застосовує обернене перетворення різкості перед кодуванням, ефективно отримуючи переваги перетворень з накладенням без недоліків;

-фільтр із збереженням країв («EPF»): подібний до двостороннього фільтра, цей згладжувальний фільтр запобігає розмиванню країв, одночасно зменшуючи дзвін. Сила цього фільтра задається локально. У порівнянні зі згладжувальними фільтрами, що зазвичай доступні в стандартах MPEG та інших відео кодеках, таких як AV1, EPF у JPEG XL є дещо більш обчислювально витратним, має більш тонкий ефект згладжування та може бути локально налаштований у більш детальний спосіб [3].

Стиснення зображень через JPEG XL без втрат

Загалом неможливо оцінити «властиві» характеристики кодека з точки зору швидкості, ступеня стиснення або якості. Можна лише протестувати конкретні реалізації кодерів та декодерів. Це дає уявлення про те, на що здатний кодек, але не гарантує, що отримані результати відображають найкращі можливості кодека. Можливо, існують потенційні поліпшення в реалізації кодера або декодера, які ще не використані. Це особливо актуально для таких кодеків, як JPEG XL, які залишають багато свободи для реалізації кодера. Як для стиснення з втратами, так і для стиснення без втрат, у багатьох аспектах процесу кодування доводиться робити вибір, що призводить до величезного простору пошуку. Вичерпно дослідити цей простір пошуку обчислювально неможливо. Лише у випадку стиснення без втрат зрозуміло, що саме потрібно оптимізувати (розмір стисненого файлу). У випадку стиснення з втратами можна використовувати об’єктивний показник для оптимізації, але це може бути лише наближенням до фактичної візуальної якості, яка зазвичай має значення.

Як результат, кодер зазвичай реалізує різні евристичні алгоритми та процедури часткового пошуку.

З цієї причини сучасні кодеки часто мають кодери з налаштуванням швидкості (або зусилля), що дозволяє робити різні компроміси між швидкістю кодування та ефективністю стиснення. Навіть найповільніші налаштування зазвичай не є вичерпними, і навіть найшвидші налаштування не обов'язково є найшвидшими швидкостями кодування, які можливі. Майбутні вдосконалення завжди можуть привести до швидшого або кращого стиснення.

Декодери зазвичай не мають або майже не мають ступенів свободи: функціонально вони (майже) повністю визначаються специфікаціями стандарту. Оскільки немає вибору, який потрібно зробити, або простору пошуку, який потрібно досліджувати, декодери часто на порядок швидші за кодери.

Точна швидкість декодування, звичайно, все ще залежить від реалізації: високооптимізований, написаний вручну SIMD-код може бути значно швидшим за просту наївну реалізацію на мові програмування високого рівня; декодер, що реалізує спеціалізовані «швидкі шляхи» для типових особливих випадків, може бути швидшим за декодер, що реалізує лише загальні методи [3].

Безвтратне стиснення – яке іноді називають «математично безвтратним», щоб уникнути плутанини з «візуально безвтратним» стисненням із втратами — найчастіше використовується під час створення та архівування матеріалів, хоча є також випадки, коли воно може бути корисним для передачі зображень кінцевим користувачам.

Основний компроміс полягає між швидкістю обробки (часом кодування та декодування) та щільністю стиснення, яку можна виразити як швидкість передачі даних (біт на піксель, або bpp), де чим нижче, тим краще, або як коефіцієнт стиснення Нестиснене : Стиснене, де чим вище, тим краще. Зазвичай кодування

є обчислювально більш вимогливим, ніж декодування, але це не обов'язково завжди так [3].

У деяких кодеках процес кодування є повністю або майже повністю детермінованим, у тому сенсі, що кодер не має значних варіантів вибору. У цьому випадку кодек пропонує лише одну точку компромісу; щільність стиснення, яку він досягає, є фіксованою, а швидкість залежить лише від апаратної та програмної реалізації. Інші кодеки, особливо JPEG XL, є дуже виразними з точки зору синтаксису бітового потоку, що залишає кодерам багато простору для дослідження більшої чи меншої частини простору пошуку. Це може призвести до цілого ряду точок компромісу між швидкістю кодування та щільністю стиснення.

У libjxl це можна налаштувати,встановивши опцію «effort», діапазон якої становить від 1 до 10, де більші числа вказують на те, що буде використано більше часу (і, можливо, пам'яті).

Для вимірювання часу кодування використовувався MacBook Pro листопада 2023 року з процесором M3 Pro ARMv8 з 6 ядрами продуктивності на частоті 4,1 ГГц та 6 ядрами ефективності на частоті 2,7 ГГц та 36 ГБ оперативної пам'яті LPDDR5. Вказаний час – це загальний час, витрачений на кодування всіх зображень по черзі.

Для кожного набору зображень, що охоплює різні типи вмісту зображень та глибини бітів, JPEG XL пропонує кілька парето-оптимальних компромісів між швидкістю кодування та стисненням. Ці результати узгоджуються з попередніми незалежними порівняннями ефективності стиснення без втрат.

Повторне стиснення JPEG без втрат: Окрім стиснення пікселів без втрат, JPEG XL також може повторно стискати файли JPEG. Очікуваний виграш залежить від кодера JPEG, піддискретизації кольоровості та налаштувань якості. Для файлів JPEG, які зазвичай створюються камерою, виграш становить близько 20% [3].

Стиснення JPEG XL з втратами

При стисненні зображень із втратами головний компроміс полягає не між швидкістю та ступенем стиснення, а між якістю зображення та ступенем стиснення. Швидкість обробки також важлива, особливо швидкість кодування. Зокрема, у сучасних кодеках кодери мають багато ступенів свободи, або, іншими словами, величезний простір пошуку, що може зробити кодування дуже повільним. Однак головним параметром, що впливає на розмір стисненого файлу, є налаштування якості. Мета полягає в тому, щоб досягти найкращої якості зображення при заданому розмірі файлу або найменшого можливого розміру файлу для заданого рівня якості [3].

3.3 Формат AVIF

AVIF – це відкритий формат зображень нового покоління, що не потребує сплати роялті та забезпечує високоякісне стиснення й дуже малий розмір файлів порівняно з традиційними форматами, такими як JPEG. Його назва розшифровується як «AV1 Image File Format»: його можна спрощено описати як версію популярного відеоформату AV1 для зображень. Іншими словами, це стандарт для зберігання нерухомих та анімованих зображень, який, на думку команди, що його просуває, відкриває нову еру у світі цифрових форматів зображень, оскільки він розроблений як наступник традиційних форматів зображень, таких як JPEG, PNG та GIF, пропонуючи чудове стиснення та кращу якість зображення порівняно з ними [7].

Заснований на кодуванні відео AV1, розробленому Alliance for Open Media – консорціумом, до якого входять такі технологічні гіганти, як Google, Netflix та Mozilla, – формат AVIF був представлений у 2019 році і поступово поширюється

в Інтернеті, що призвело до розширення його підтримки на рівні браузерів, CDN для зображень, плагінів WordPress, а також інструментів кодування.

Це стало можливим завдяки його функціям, розробленим та спроектованим для революції у способі обробки зображень в Інтернеті: він поєднує дуже високу якість зображення та ефективне стиснення, що дозволяє значно зменшити розмір файлів без втрати чіткості або глибини кольорів [7].

Механізм AV1, на якому базується формат AVIF, забезпечує більш ефективне стиснення зображень, ніж традиційні формати JPEG або PNG, завдяки передовим технологіям внутрішньокадрового стиснення. Іншими словами, він зменшує розмір файлу без втрати візуальної якості, яку бачить користувач.

Це означає швидше завантаження сторінок і позитивний вплив на користувацький досвід (UX) — два фактори, що стають дедалі важливішими для оптимізації сайтів у пошукових системах, таких як Google.

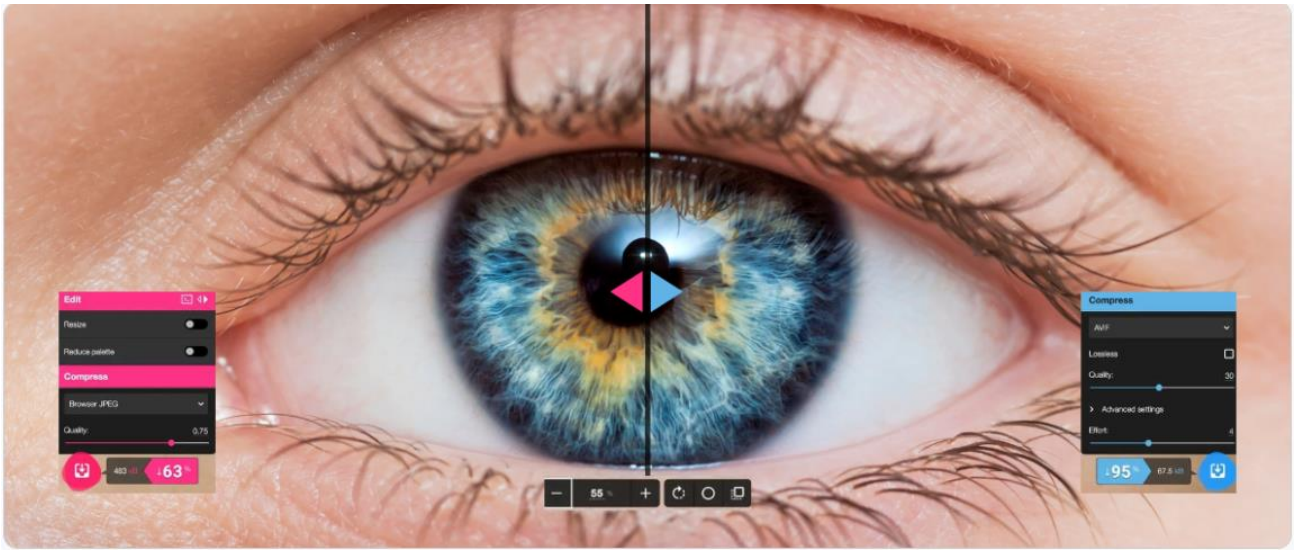

Рисунок 3.3 – Стиснення через AVIF

Через трохи більше п’яти років після дебюту — перша версія AVIF вийшла в лютому 2019 року — майже всі браузери тепер підтримують AVIF, як показано на графіку caniuse.

Більше того, згідно з новинами кількох тижнів тому, з вересня 2024 року AVIF офіційно є підтримуваним типом файлів у Google Search, Google Images та будь-де, де використовуються зображення в Google Search; це також означає, що файли AVIF індексуються Google без особливих труднощів.

Формат зображень AVIF і його сновні особливості

AVIF може бути чудовим вибором для розміщення зображень у мережі, пишуть експерти web.dev: цей формат швидко кодується та декодується, забезпечує якісне відтворення та ефективне стиснення розміру файлів, а також наразі є найефективнішим способом доставки анімації в мережі [7].

Висока швидкість кодування та висока візуальна якість мають вирішальне значення для впровадження стиснення зображень у великих масштабах, а технічні особливості роблять формат AVIF чудовим варіантом для використання у веб-дизайні та інших цифрових додатках.

По-перше, AVIF забезпечує чудову якість зображення при зменшеному розмірі файлу: зображення у форматі AVIF зберігають високий рівень деталізації та чіткості, при цьому їхній розмір значно менший, ніж у традиційних форматів зображень. Таке чудове стиснення особливо корисне у веб-дизайні, де розмір файлів зображень може суттєво впливати на продуктивність веб-сайту. Зокрема, AVIF підтримує дуже ефективне стиснення з втратами та без втрат для створення зображень високої якості, і після стиснення AVIF досягає кращих результатів, ніж популярні на сьогодні у мережі формати (JPEG, WebP, JPEG 2000 тощо); зокрема, деякі тести показують, що зображення можуть бути вдесятеро меншими за файли JPEG з подібною візуальною якістю, тоді як практичні дослідження продемонстрували, що AVIF забезпечує 50-відсоткову економію розміру файлу порівняно з JPEG-файлами подібної сприйманої якості [9].

Крім того, AVIF підтримує ширшу колірну гаму та більшу глибину кольору, ніж традиційні формати зображень, з відтворенням більш тонких

колірних переходів – іншими словами, зображення є реалістичнішими та детальнішими, з покращеною сприйманою якістю та меншим розміром файлу, ніж у JPEG або WebP. AVIF також підтримує прозорість, подібно до PNG та GIF, що робить його придатним для використання в різноманітних дизайнерських додатках.

Ще однією важливою особливістю AVIF є підтримка технології High Dynamic Range (HDR), яка дозволяє передавати на зображенні широкий діапазон яскравості — від найтемніших до найяскравіших деталей, що забезпечує більш реалістичні та візуально вражаючі зображення [7].

Це робить формат AVIF чудовим вибором для зображень з високою роздільною здатністю та додатків для потокового відео, де якість зображення має першочергове значення.

Технічні властивості AVIF: кодування, глибина кольору, анімація та метадані Таким чином, формат AVIF є значним кроком вперед у світі цифрових зображень завдяки унікальному поєднанню технічних особливостей, що роблять його надзвичайно універсальним та ефективним.

Почнемо з кодування: AVIF базується на відеокодеку AV1, який є результатом співпраці консорціуму провідних технологічних компаній, зокрема Google, Microsoft та Amazon. Кодек AV1 відомий своєю ефективністю стиснення, оскільки здатний створювати зображення найвищої якості, зберігаючи при цьому неймовірно малі розміри файлів. Це особливо важливо у веб-середовищі, де швидкість завантаження сторінок є надзвичайно важливою.

Однією з відмінних рис формату AVIF є його здатність підтримувати глибину кольору до 12 бітів на канал, що перевищує класичне 8-бітне обмеження формату JPEG. Завдяки такій більшій глибині AVIF може відображати значно ширший динамічний діапазон, забезпечуючи більш насичене та детальне відтворення кольорів, що дає змогу отримувати реалістичніші зображення з

більшою градацією, особливо в зонах з високим контрастом, таких як глибокі тіні або яскраві світлі ділянки. Для тих, хто працює з зображеннями HDR (High Dynamic Range), AVIF пропонує передачу кольорів, що відповідає вимогам сучасних екранів, забезпечуючи яскраві та точні кольори.

Окрім статичної якості, AVIF також підтримує анімацію. На відміну від таких форматів, як JPEG, AVIF може містити послідовності анімованих зображень в одному файлі, що відкриває нові творчі можливості. Це робить його придатним для створення інтерактивного та динамічного контенту без необхідності використання окремих або більш складних форматів, таких як GIF або відео MP4.

Анімації в AVIF можуть підтримувати ефективне стиснення, зберігаючи плавність і чіткість, що ідеально підходить для рекламних банерів, мікроінтеракцій та інших сучасних застосувань веб-дизайну.

AVIF не ігнорує важливість вбудовування метаданих. Цей формат підтримує широкий спектр вбудованих метаданих, таких як профілі ICC для управління кольором, що забезпечують точне відтворення зображень на різних пристроях. Серед інших підтримуваних метаданих — географічні координати та інформація про авторські права, що робить AVIF універсальним вибором не тільки для поширення зображень, а й для управління та архівування візуального контенту, який потребує більш розширеної класифікації та захисту [7].

Як працює формат AVIF і чому він забезпечує кращу компресію

На відміну від традиційних форматів, AVIF використовує технологію компресії, що базується на методі прогнозного внутрішньокадрового кодування кодека AV1, який використовує просторову кореляцію між різними частинами зображення для зменшення розміру файлу.

У результаті отримуємо файл меншого розміру, що зберігає вражаючу якість зображення, порівнянну з якістю нестисненого зображення.

Іншим ключовим аспектом є стиснення з втратами та без втрат. З AVIF ми маємо можливість вибирати між стисненням із втратами, яке жертвує деякими деталями для значного зменшення розміру файлу, та стисненням без втрат, яке зберігає кожну деталь оригінального зображення, одночасно зменшуючи розмір файлу порівняно з традиційними форматами. Це є великою перевагою в ситуаціях, де якість зображення є критично важливою, наприклад, у висококласній фотографії або в галузі охорони здоров'я.

Для чого призначений новий формат і чому він корисний

Зображення є найпопулярнішим типом ресурсів у мережі, і часто вони є найбільшими та найважчими: користувачі цінують високоякісні зображення, але адміністратори сайтів повинні подбати про те, щоб усі зображення — чи то головні ілюстрації, фотографії товарів чи меми — завантажувалися якомога ефективніше та швидше [9].

Якщо розглянути, наприклад, лише показники Core Web Vitals від Google, то цікаво знати, що зображення становлять близько 42 відсотків показника Largest Contentful Paint для веб-сайтів, і загалом дуже часто на ці та інші ключові показники, орієнтовані на користувача, сильно впливають розмір, кількість, розміщення та пріоритет завантаження зображень на сторінці — саме тому ми приділяємо велику увагу технікам оптимізації зображень також і для SEO.

Таким чином, формат AVIF був створений, щоб надати можливість публікувати високоякісні зображення з невеликим розміром файлу для різноманітних цифрових додатків завдяки ряду інноваційних та вигідних функцій, які відрізняють його від інших популярних форматів зображень.

Як уже згадувалося, однією з найочевидніших особливостей AVIF є його здатність забезпечувати чудову якість зображення при зменшеному розмірі файлу, що стало можливим завдяки використанню відеокодека AV1, який забезпечує чудове стиснення порівняно з кодеками, що використовуються в

традиційних форматах зображень, і дозволяє зображенням AVIF зберігати високий рівень деталізації та чіткості при значно меншому розмірі файлу.

Ще однією інноваційною особливістю AVIF є підтримка ширшої колірної гами та більшої глибини кольору: тоді як традиційні формати зображень, такі як JPEG і PNG, підтримують глибину кольору 8 біт, AVIF підтримує глибину кольору 10 або 12 біт, що дозволяє відтворювати більш тонкі колірні переходи, завдяки яким зображення стають реалістичнішими та деталізованішими, що має вирішальне значення в ситуаціях, де якість зображення є надзвичайно важливою.

Крім того, AVIF підтримує HDR – функцію, яка не підтримується більшістю традиційних форматів зображень, – що дозволяє фіксувати в зображенні широкий діапазон яскравості, від найтемніших до найяскравіших деталей, що може призвести до створення більш реалістичних і візуально вражаючих зображень [7].

Тому цей формат є особливо корисним у веб-дизайні, де розміри файлів зображень можуть мати значний вплив на продуктивність веб-сайту, оскільки, як уже згадувалося, він дозволяє зберегти високий рівень якості зображення при одночасному зменшенні розмірів файлів, що може призвести до швидшого завантаження сторінок та кращого користувацького досвіду.

Підсумовуючи, можна сказати, що, окрім чудового стиснення, AVIF також пропонує інші корисні та цікаві функції:

-Він підтримує анімацію, Live Photos та інші функції завдяки багатошаровим зображенням, що зберігаються у вигляді послідовностей кадрів.

-Він забезпечує кращу підтримку графічних елементів, логотипів та інфографіки, де формат JPEG має обмеження.

-Забезпечує краще стиснення без втрат, ніж JPEG.

-Підтримує 12-бітну глибину кольору, що дозволяє створювати зображення з високим динамічним діапазоном (HDR) та широкою кольоровою гамою (WCG) з кращими світлими та темними тонами та ширшим діапазоном яскравості.

-Включає підтримку монохромних зображень та багатоканальних зображень, включаючи прозорі зображення з використанням альфа-каналу.

Коли і навіщо використовувати формат AVIF: практичні міркування

Одним із ключових аспектів управління веб-сайтом є вибір найбільш підходящого формату зображень. Знання того, коли і навіщо віддавати перевагу AVIF, може стати вирішальним фактором у забезпеченні високої продуктивності сайту. У таких випадках, як електронна комерція, новинні портали або блоги з переважною кількістю зображень, використання AVIF настійно рекомендується з двох основних причин: чудове стиснення та покращена якість зображення на сучасних пристроях. Це означає, що зображення у форматі AVIF дозволяють веб-сторінкам завантажуватися швидше та зменшують споживання даних, тим самим покращуючи користувацький досвід, особливо на мобільних пристроях.

AVIF особливо корисний, коли потрібно оптимізувати швидкість завантаження сторінки без втрати якості зображення. Наприклад, на платформах із сильним візуальним компонентом, таких як соціальні мережі або сайти з графічним дизайном та фотографією, де якість зображення має вирішальне значення для утримання уваги користувачів, AVIF забезпечує виняткову продуктивність із файлами, які важать на 50 відсотків менше, ніж JPEG.

Для тих, хто веде сайт, орієнтований виключно на мобільну аудиторію, AVIF може стати справжнім проривом. Покращена швидкість завантаження може мати прямий вплив на SEO, що призведе до збільшення органічного трафіку. Однак важливо уважно розглянути потреби нашого сайту, щоб визначити, чи дійсно AVIF є оптимальним рішенням для кожного зображення.

Отже, в принципі, формат AVIF може бути корисним для широкого кола користувачів і застосувань, а особливо — для веб-дизайнерів та розробників веб-сайтів, які можуть використовувати цей формат для підвищення продуктивності сайтів та покращення користувацького досвіду.

Завдяки використанню зображень у форматі AVIF веб-сайти, що містять велику кількість зображень — такі як сайти електронної комерції, блоги про подорожі чи новинні сайти — можуть отримати особливу вигоду від зменшення розміру файлів зображень, що призведе до прискорення завантаження сторінок та покращення користувацького досвіду.

Крім того, з огляду на свої технічні характеристики, AVIF може бути корисним для додатків потокового відео, для обробки коротких анімацій та для зображень з високою роздільною здатністю, де якість зображення має першочергове значення; завдяки підтримці HDR та ширшої колірної гами він може забезпечити більш реалістичні та детальні зображення, ніж традиційні формати зображень [7].

Переваги AVIF перед іншими форматами зображень

Чому варто віддати перевагу AVIF перед іншими форматами, такими як JPEG, PNG або WebP? По-перше, AVIF забезпечує вищу якість зображення при меншому розмірі файлу. Це призводить до відчутного підвищення продуктивності сайту, зокрема скорочення часу завантаження, що, у свою чергу, впливає на позиції в пошукових системах.

Що стосується управління кольорами, AVIF підтримує широку гаму розширених кольорів, включаючи кольори з високим динамічним діапазоном (HDR), що дозволяє отримувати більш насичені та яскраві зображення. Уявіть собі інтернет-магазин модного одягу, який може надзвичайно точно відображати тканини та кольори без уповільнення роботи сайту.

Для порівняння, формати JPEG та PNG мають обмеження як щодо розміру, так і щодо якості кольорів, особливо на сучасних пристроях з екранами високої роздільної здатності.

Але AVIF – це не лише синонім покращеної продуктивності: це також відкритий стандарт, повністю безроялтітний формат, що відповідає тенденції відкритого та спільного веб-простору.

Ця особливість має велике значення, оскільки дозволяє уникнути додаткових витрат та технологічних перешкод, на відміну від формату JPEG 2000, який стикався з проблемами впровадження саме через ліцензування.

Вибір між AVIF і JPEG часто є одним із перших питань, з якими ми стикаємося, коли йдеться про оптимізацію зображень. JPEG, з його багаторічною історією та універсальною сумісністю, здається природним вибором. Однак, якщо розглянути це детальніше, стає зрозуміло, що AVIF має переваги, з якими JPEG не може зрівнятися. З AVIF ми отримуємо вищу якість зображення при значно меншому розмірі файлу, що прискорює завантаження сторінки [7].

Зокрема, AVIF підтримує ширший динамічний діапазон, забезпечуючи більшу точність у відтворенні кольорів та візуальних деталей, ніж JPEG. Це надзвичайно корисно для такого контенту, як професійні фотографії, детальні ілюстрації та зображення, де чіткість має вирішальне значення. З іншого боку, JPEG може бути корисним у ситуаціях, коли універсальна сумісність та підтримка старих версій мають вирішальне значення, або коли потрібна висока швидкість рендерингу на застарілому обладнанні.

Саме це і називається «колірним об’ємом» — технічним терміном, що позначає повний діапазон кольорів, які може відтворювати або сприймати такий пристрій, як монітор, принтер, фотоапарат або певна система відображення. У цифровій сфері та в контексті зображень колірний об’єм визначає діапазон усіх

кольорів, які здатна відтворити конкретна технологія (наприклад, екран або формат зображення).

Сказати, що формат або технологія підтримує ширший «гамут», означає, що вони здатні відтворювати багатший і різноманітніший діапазон кольорів.

Це особливо важливо для застосувань, де точність кольорів має вирішальне значення, таких як графічний дизайн, професійна фотографія або контент із високим динамічним діапазоном (HDR), де необхідно відображати яскравіші, темніші або більш насичені колірні гами з більшою точністю, ніж у більш обмежених форматах.

У конкретному випадку формату AVIF ширша гама кольорів вказує на те, що цей формат може обробляти багатше та точніше відтворення кольорів, ніж старіші формати, такі як JPEG, що дає більш реалістичні зображення з більшою візуальною деталізацією, особливо у сценах з високим контрастом або широким розмаїттям кольорів. Це не означає, що нам слід повністю відмовитися від формату JPEG.

Вибір між цими двома форматами залежить від сукупності факторів, зокрема від типу візуального контенту, цільової аудиторії та бажаної продуктивності. Найкращим підходом може бути застосування диверсифікованої стратегії: використання AVIF для високоякісного контенту та JPEG для більш компактного контенту, який легше завантажувати.

Ефективність AVIF у порівнянні з іншими форматами зображень

Головною причиною створення AVIF стало прагнення подолати обмеження традиційних форматів зображень, таких як JPEG, PNG та GIF, які, незважаючи на широке поширення, мають низку недоліків, зокрема відносно великий розмір файлів та обмежену колірну гаму.

Наразі ми маємо досить широкий вибір форматів зображень для візуалізації в Інтернеті, суттєва відмінність яких полягає в кодеках зображень, що

використовуються для кодування або декодування кожного типу зображення, які відрізняються між собою.

Кодек зображення – це алгоритм, що використовується для стиснення та кодування зображень у певний тип файлу, а також для їх декодування з метою відображення на екрані. Ефективність та результативність кодека можна оцінити, проаналізувавши такі параметри, як ступінь та якість стиснення, а також швидкість кодування та декодування: однак, нагадуємо, що ми повинні прагнути оцінити конфігурацію якості та формати, які найкраще відповідають нашим потребам, можливо, доручивши частину цього завдання CDN для зображень, якщо у нас бракує часу або досвіду [7].

Отже, з практичної точки зору ми можемо порівняти продуктивність та вихідні дані формату AVIF з основними та найпопулярнішими форматами зображень, і при цьому виявити низку суттєвих переваг.

Наприклад, в принципі AVIF забезпечує вищу якість зображення та менший розмір файлу, ніж зображення у форматі JPEG — за розміром файлу зображення у форматі AVIF можуть бути на 50 відсотків меншими, ніж зображення у форматі JPEG аналогічної якості, — при цьому зберігаючи високий рівень деталізації та чіткості навіть при високих рівнях стиснення.

Підтримка функцій	JPEG	PNG	WebP	AVIF
Підтримка програмного забезпечення	★★★★★	★★★★★	★★★	★★
Стиснення (фотографічні зображення)	★★	✕	★★	★★★★
Стиснення (не фотографічні зображення)	★	★★	★★★	★★★
Швидкість	★★★★	★★★	★★	★
Підтримка анімації	✕	✕	✅	✅
Втратне стиснення 4:4:4	✅	✅	✕	✅
Субдискретизація кольору 4:2:0	✅	✕	✅	✅
Субдискретизація кольору 4:2:2	✅	✕	✕	✅
Широка колірна гама	✅	✅	✕	✅
HDR	✕	✅	✕	✅
Прогресивне декодування	★★★	★	✕	★
Прозорість альфа-каналу	✕	✅	✅	✅
Накладки (оверлеї)	✕	✕	✕	✅

Рисунок 3.4 – Порівняння AVIF з іншими форматами

Позитивні результати також були зафіксовані у порівнянні з PNG — ще одним поширеним форматом зображень, відомим своєю здатністю підтримувати прозорість та забезпечувати високу якість зображень, однак за рахунок збільшення розміру файлу. Натомість AVIF може забезпечити аналогічну або вищу якість зображення при значно меншому розмірі файлу, водночас підтримуючи прозорість, що робить його придатним для використання в різноманітних дизайнерських додатках.

У порівнянні з WebP, форматом зображень, нещодавно розробленим Google, AVIF пропонує вищу якість зображення та ефективніше стиснення: хоча WebP забезпечує краще стиснення порівняно з JPEG та PNG, AVIF йде далі, пропонуючи ще ефективніше стиснення та вищу якість зображення. Однак WebP має ширшу підтримку і може використовуватися для візуалізації звичайних зображень, де не потрібні розширені функції, такі як широка колірна гама або накладення тексту [7].

Підсумовуючи, як також видно з порівняльної таблиці, складеної smashingmagazine у 2023 році, AVIF є надійним першим вибором, якщо прийнятне стиснення з втратами та низькою точністю, а економія пропускної здатності є головним пріоритетом, за умови, що швидкість кодування/декодування відповідає нашим потребам. На даний момент WebP має кращу підтримку та пропонує краще стиснення, ніж JPEG або PNG, тому це може бути ефективним рішенням для оптимізації зображень на нашому веб-сайті. Рекомендується оцінити, чи відповідає AVIF нашим потребам, і впровадити його як поступове вдосконалення, щоб ми могли по ходу роботи спостерігати, як і наскільки добре цей формат приймається різними браузерами та платформами.

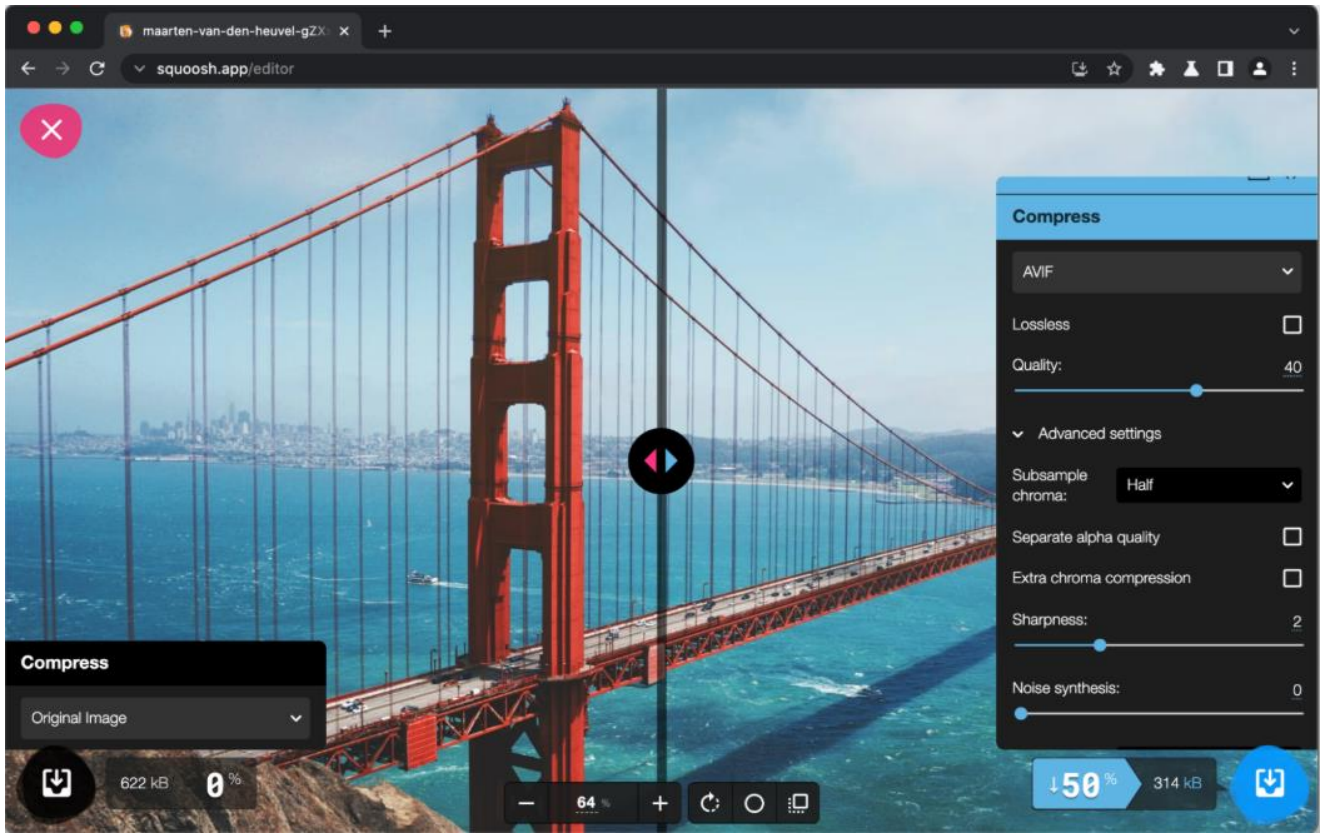

Рисунок 3.5 – Стиснення через AVIF

Ключовим аспектом вибору найбільш підходящого формату зображення є контекст використання та конкретні вимоги до сумісності й продуктивності кожного проекту. Формати AVIF, JPEG, PNG та WebP є компромісом між візуальною якістю, ступенем стиснення та сумісністю, тому вибір формату слід здійснювати з чітким розумінням переваг та обмежень кожного з них.

Для проектів, орієнтованих переважно на візуальне оповідання — таких як цифрові портфоліо, фотогалереї або сайти про мистецтво — AVIF, безсумнівно, пропонує найкращу візуальну якість із найкращим співвідношенням стиснення та якості. Однак для проектів із низьким рівнем сумісності, таких як корпоративні сайти, що мають підтримувати застарілі платформи, JPEG може бути

безпечнішим вибором, хоча й із компромісом щодо деталізації та загальної якості зображення.

WebP стає серйозним конкурентом AVIF, особливо для тих, хто шукає компроміс між ефективним стисненням і широкою підтримкою, оскільки цей формат добре розпізнається сучасними браузерами, але не вимагає такого рівня сумісності, як AVIF. З іншого боку, PNG залишається надійним вибором для зображень, що вимагають точної прозорості і де стиснення з втратами може спричинити небажані артефакти.

Важливо розуміти, що вибір формату не має бути жорстко фіксованим; часто гібридний мультиформатний підхід, за якого основні зображення зберігаються у форматі AVIF, а резервні — у WebP або JPEG, дозволяє досягти максимальної візуальної якості та ступеня стиснення, зберігаючи при цьому належну універсальну доступність. Така стратегія забезпечує візуальну узгодженість та зручність користування на будь-якому пристрої, гарантуючи при цьому швидке завантаження сторінок та оптимальний рейтинг у пошукових системах [7].

Відмінності між AVIF та WebP

І AVIF, і WebP – це два сучасні формати зображень, розроблені для заміни застарілих JPEG та PNG більш ефективними рішеннями з точки зору стиснення та якості. Перша відмінність полягає у стисненні: тоді як WebP пропонує значно краще стиснення, ніж JPEG, з підтримкою зображень із втратами та без втрат, AVIF йде далі, пропонуючи ще ефективніше стиснення завдяки використанню кодування на основі AV1. Це призводить до того, що файли AVIF часто на 30–50% менші за WebP, зберігаючи при цьому вищу візуальну якість.

Що стосується глибини кольору, AVIF підтримує до 12 бітів на канал, забезпечуючи вищий динамічний діапазон і точність передачі кольорів, ніж WebP, який обмежується 8 бітами на канал.

Хоча WebP доступний і підтримується вже довше, нещодавнє впровадження AVIF провідними браузерами та платформами швидко скорочує цей розрив, роблячи AVIF кращим вибором для тих, хто шукає найкраще поєднання стиснення та якості.

Відмінності між AVIF та JPEG

Якщо порівняти AVIF із JPEG, відмінності стають очевидними одразу, особливо в плані стиснення та візуальної якості. JPEG десятиліттями домінував як фактичний стандарт для зображень у мережі, пропонуючи прийнятне стиснення з втратами, яке зменшує розмір файлу за рахунок якості, з переважанням візуальних артефактів, типових для надмірно спрощеного стиснення. З іншого боку, AVIF підриває це лідерство, пропонуючи значно вищий коефіцієнт стиснення, що суттєво зменшує розмір файлу без шкоди для візуальної якості. Завдяки кодуванню на основі кодека AV1, AVIF краще обробляє деталі зображення та підтримує глибину кольору до 12 бітів на канал, порівняно з 8 бітами у JPEG, що візуально перетворюється на плавніші переходи кольорів та реалістичніше зображення, особливо у зображеннях з високим динамічним діапазоном. Крім того, тоді як JPEG не підтримує прозорість, AVIF добре з нею справляється, забезпечуючи більшу гнучкість для веб-дизайну [7].

Обмеження формату AVIF

Однак, незважаючи на всі ці позитивні аспекти, залишається одна фундаментальна проблема, на яку ми вже згадували раніше: AVIF і WebP можуть забезпечити вищу якість зображень і значно менший розмір файлів, але в Інтернеті, в основному через зручність використання, як і раніше домінують такі формати, як JPEG, GIF або PNG.

Насправді, порівняно з будь-яким новим кодуванням зображень, ці формати мають міцну традицію і, практично протягом десятиліть, працювали

ефективно, безпечно та гарантовано у всіх браузерах, тоді як нові формати не є загальноприйнятими в Інтернеті [7].

Протягом останніх кількох років величезну кількість часу та зусиль було присвячено розробці нових форматів зображень, спрямованих на поліпшення як якості, так і розміру при передачі: такі формати, як WebP, AVIF та JPEG XL (не підтримується жодним браузером), по-різному прагнуть стати універсальним рішенням для растрових зображень у Інтернеті, як SVG для векторних; інші, такі як JPEG 2000 (підтримується лише в Safari), мали на меті задовольнити всі ті самі випадки використання, що й базовий JPEG, але з покращеними методами стиснення, щоб забезпечити візуально схоже, але набагато менший розмір зображення.

Хоча деякі з цих нових форматів мають у назві слово «JPEG», їхні кодування принципово відрізняються так само, як JavaScript відрізняється від Java: браузер, який не підтримує певне кодування, взагалі не зможе обробити цей файл зображення; він завантажить дані зображення, спробує їх обробити, а в разі невдачі видалить їх, не відобразивши. Джерело зображення, яке не відображається поза сучасними браузерами, стане величезною проблемою для нашого контенту та для Інтернету загалом: пошкоджене зображення та марна трата пропускної здатності для величезної кількості користувачів по всьому світу. Тому, як підтверджує web.dev, діє одне правило: ми не повинні жертвувати більш стійкою мережею заради кращої продуктивності.

Повертаючись до AVIF, з технічної точки зору також існують деякі відомі обмеження: якщо найбільшим недоліком формату на даний момент є відсутність єдиної підтримки серед браузерів, то при детальному розгляді виявляються й інші критичні моменти.

Наприклад, AVIF може не мати можливості стискати нефотографічні зображення так само добре, як це роблять PNG або WebP без втрат. Крім того, є

й інші аспекти, в яких AVIF не відповідає ідеальним стандартам сучасного формату файлів, такі як:

-зворотна сумісність. Сучасні версії Chrome (Chrome 94+) підтримують прогресивну візуалізацію AVIF, тоді як старіші версії — ні, і на даний момент не існує кодера, який би міг легко створювати такі зображення. Прогресивне кодування дозволяє відображати попередній перегляд зображення низької якості, поки завантажується решта зображення;

-кодування та створення зображень AVIF займає більше часу. Це може стати проблемою для сайтів, які динамічно створюють файли зображень або потребують обробки великої кількості зображень у реальному часі. Однак команда AVIF працює над покращенням швидкості кодування. Учасники проекту AVIF у Google також повідомили про певне підвищення продуктивності. З 1 січня 2021 року кодування AVIF показало покращення часу транскодування приблизно на 47 відсотків (це швидкість 6, поточне значення за замовчуванням для libavif), а протягом календарного року — на 73 відсотки; з наступного липня також відбулося покращення часу транскодування на 72 відсотки на швидкості 9 (кодування «на льоту»);

-Декодування зображень AVIF для відображення також може вимагати більшої потужності процесора, ніж інші кодеки, хоча менший розмір файлів може це компенсувати;

-Хоча AVIF пропонує чудову якість зображення при меншому розмірі файлу, стиснення може призвести до втрати якості зображення в деяких випадках, особливо якщо ступінь стиснення дуже високий. Це може стати проблемою для програм, які вимагають зображень високої якості з дуже дрібними деталями;

-Деякі CDN досі не підтримують формат AVIF за замовчуванням у режимах автоматичного форматування, оскільки при першому запиті час його генерації може бути навіть довшим [7].

3.4 Покроковий опис стиснення зображення через AVIF

Формат AVIF використовує відеокодек AV1 для високоефективного стиснення, а потім застосовує контейнер HEIF (High-Efficiency Image File) для упорядкування та зберігання даних зображення. Кодек працює наступним чином:

-спочатку він розділяє зображення на крихітні блоки (від 4×4 до 64×64 пікселів), щоб мати змогу аналізувати та стискати кожну частину окремо, що забезпечує більш ефективне кодування та менший розмір файлів;

-потім кодек шукає візерунки, кольори та краї в кожному блоці та прогнозує значення на основі сусідніх блоків. Замість того, щоб зберігати всі вихідні дані, він зберігає лише різницю між прогнозованими та фактичними значеннями, тим самим заощаджуючи простір;

-дані зображення перетворюються на візерунки (наприклад, хвилі), щоб полегшити стиснення. Потім він видаляє менш важливі деталі (у стисненні з втратами), щоб заощадити ще більше місця;

-кодек упаковує дані щільніше, присвоюючи короткі коди поширеним візерункам і довші коди менш поширеним. Цей крок робить файл якомога меншим;

-після стиснення зображення зберігається в контейнері HEIF. Цей контейнер упорядковує дані зображення разом із додатковою інформацією, такою як налаштування камери, місце розташування та деталі кольору, в один акуратний файл AVIF [8].

3.5 Порівняння AVIF, JPEG XL та JPEG

Якість зображення та артефакти: Зображення, стиснуті у форматі AVIF, містять менше артефактів, ніж у форматі JPEG. Погляньте на цей приклад векторного файлу SVG, перетвореного як у формат JPEG (ліворуч), так і у формат AVIF (праворуч). Роздільна здатність у обох випадках однакова [4].

Рисунок 3.6 – JPEG vs AVIF

Формат JPEG зазвичай погано справляється з чіткими краями та високим контрастом, що добре видно на скріншоті. Файл AVIF праворуч виглядає набагато плавніше. Розмір файлу JPEG становить 15,5 КБ, а AVIF – 5,5 КБ.

Підтримка прозорості та альфа-каналу: На відміну від JPEG, AVIF підтримує прозорість, що робить його придатним для графіки та логотипів [4].

Підтримка браузерами та програмним забезпеченням: Станом на жовтень 2025 року AVIF підтримується всіма основними браузерами в їхніх останніх версіях. Windows підтримує AVIF за допомогою безкоштовного розширення, яке встановлюється з Microsoft Store, а macOS 13 і новіші версії пропонують вбудовану підтримку AVIF. Для мобільних пристроїв вбудована підтримка AVIF починається з iOS 16 та Android 12. Крім того, завжди є можливість відкрити файл AVIF у браузері [4].

3.5.1 Коли варто віддати перевагу AVIF перед JPEG

Дискусія щодо вибору між AVIF та JPEG часто виникає під час обговорення веб-сайтів. Зазвичай більшість відвідувачів веб-сайтів користуються сучасними браузерами, тому перехід на AVIF є цілком безпечним. Проте для впевненості варто перевірити аналітику вашого веб-сайту, щоб дізнатися, якими саме браузерами користуються відвідувачі. Також не забувайте передбачити альтернативні варіанти за допомогою HTML-елементів <picture>. Для електронних листів та застарілих веб-систем використовуйте JPEG.

Однак є одне застереження – цифри щодо зменшення розміру файлів, які ви можете побачити в Інтернеті, зазвичай є усередненими. Тому, перш ніж повністю перейти на новий формат, протестуйте конвертацію зображень за допомогою таких додатків, як ImageOptim та Squoosh – це допоможе вам прийняти обґрунтоване рішення, виходячи з ваших конкретних умов.

У ситуаціях, коли використання AVIF не дає істотної різниці, розгляньте можливість використання Progressive JPEG – зображення спочатку відображається цілком у низькій роздільній здатності, а деталі з’являються у міру завантаження [4].

3.5.2 AVIF проти JPEG XL

AVIF і JPEG XL – це два основні претенденти на заміну формату JPEG у сферах застосування поза браузерами.

Стиснення та розмір файлу: При однаковому рівні візуального стиснення за сприйняттям формат JPEG XL може випереджати AVIF на 25 %. Це особливо актуально для фотографій, стиснутих із збереженням високої якості [4].

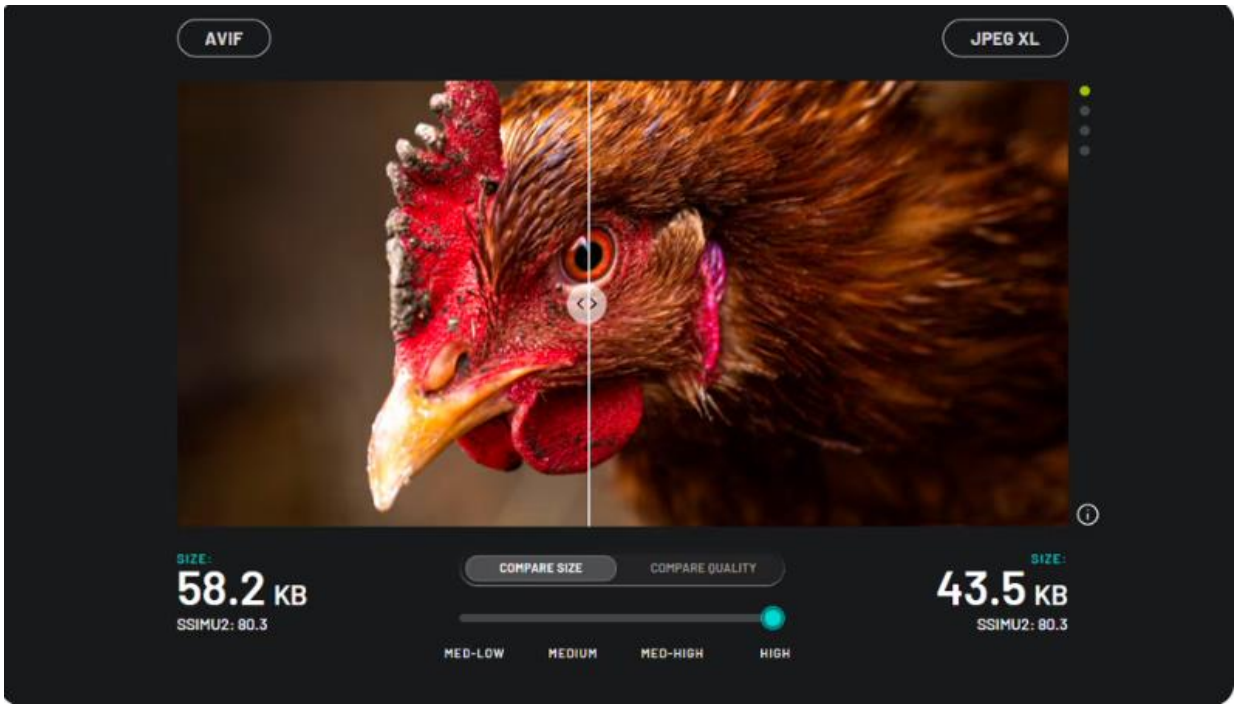

Рисунок 3.7 – AVIF у порівнянні з JPEG XL при однаковому індексі схожості

При стисненні для виведення на екран з низькою роздільною здатністю формат AVIF виходить на перше місце – він робить зображення розмитими по краях, приховуючи артефакти стиснення. На скріншоті нижче порівняно два формати при налаштуваннях низької якості – хмари та зірки виглядають краще на зображенні у форматі AVIF, незважаючи на нижчий показник SSIMU2. AVIF, ймовірно, є найкращим вибором для сильно стиснутих логотипів та діаграм [4].

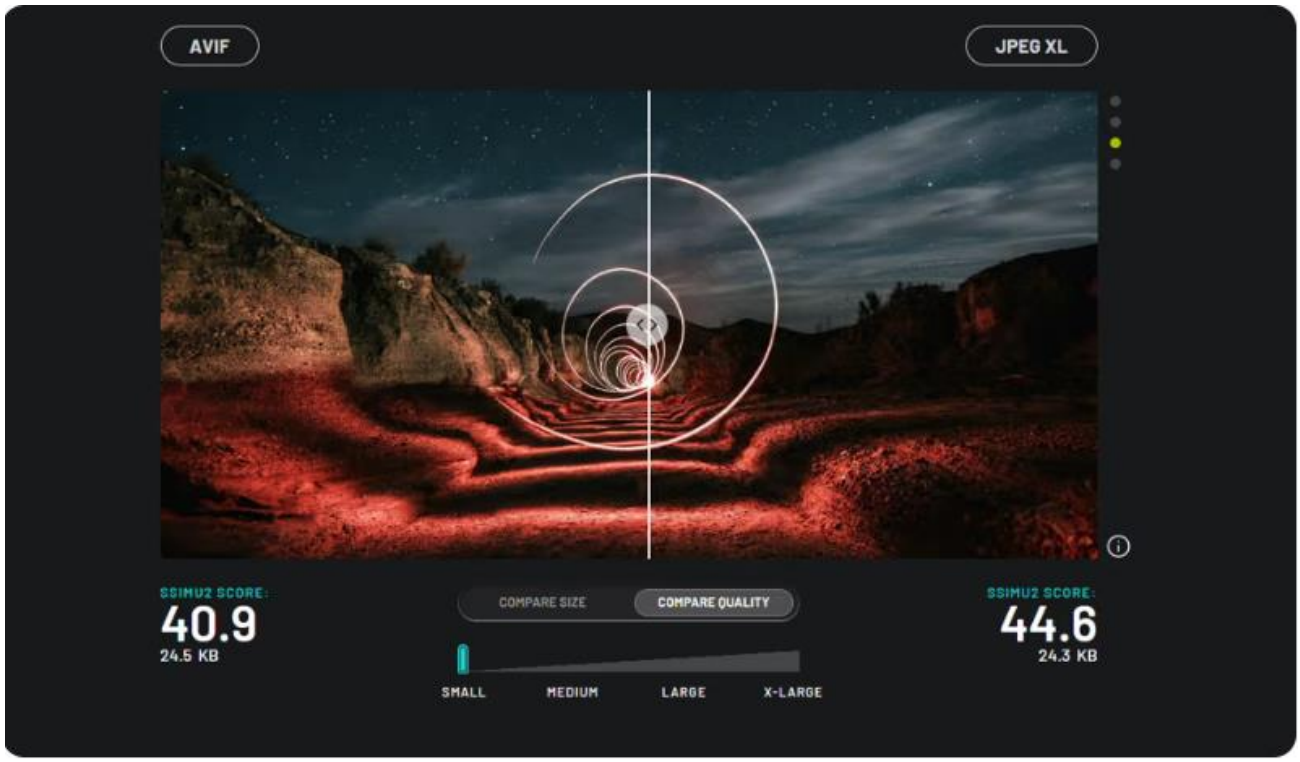

Рисунок 3.8 – AVIF у порівнянні з JPEG XL, стисненим для досягнення мінімального розміру

Показники якості сприйняття, такі як SSIMU2, не є універсальними. Для спеціалізованих застосувань може знадобитися проаналізувати результати стиснення на основі власного набору зображень.

Варіанти стиснення без втрат та зі втратами

Обидва формати підтримують стиснення без втрат, причому JPEG XL дещо випереджає конкурента завдяки своєму походженню – JXL підтримує транскодування JPEG без втрат без повторного кодування. Іншими словами, він «обгортає» звичайний файл JPG, стискаючи його більш сучасним способом на 16–22 %. За результатами цього експерименту JPEG XL значно перевершує AVIF. Щодо стиснення з втратами, дивіться скріншоти вище.

Глибина кольору, HDR та розрядність

Формат JPEG XL підтримує 32 біти на канал, на відміну від 12 бітів у форматі AVIF, що робить перший кращим вибором для галузей, які значною мірою покладаються на плавні колірні переходи. Для порівняння: стандартний формат JPEG підтримує 8 бітів на канал. Обидва формати кодування підтримують HDR та широку колірну гаму [4].

Швидкість кодування та декодування

Згідно з порівняннями, проведеними компанією Google, формат AVIF перевершує JPEG XL за швидкістю декодування в браузері Chrome з розширенням JPEG XL. Щодо швидкості кодування, то в ході одного з експериментів було виявлено, що процес кодування у форматі JPEG XL може бути до 3 разів швидшим, ніж у форматі AVIF, при однакових налаштуваннях якості зображення. Що стосується рендерингу, JPEG XL підтримує прогресивне завантаження: зображення з’являється цілим, а деталі з’являються по мірі завантаження. AVIF не підтримує прогресивне завантаження.

Підтримка браузерами та платформами у 2025 році

Як уже зазначалося, AVIF є безперечним лідером у цьому питанні – його підтримують усі основні браузери. У 2025 році JPEG XL фактично не вважається прийнятним варіантом для веб-використання. Наразі рівень його підтримки на сайті caniuse становить лише 10%.

Стиснення AVIF є ефективнішим для зображень низької та середньої якості, тоді як стиснення JPEG XL краще працює для зображень високої якості та без втрат.

Приклади застосування: AVIF – це найкращий вибір для будь-яких веб-завдань, тоді як JPEG XL найкраще підходить для промислових застосувань та ситуацій, де головну роль відіграють кольори.

Продуктивність веб-сайтів та Core Web Vitals: Core Web Vitals (LCP, CLS, INP) – це кількісні показники від Google, що відображають якість користувацького досвіду (UX) сайту. Значне поліпшення цих показників збільшує ймовірність того, що відвідувач залишиться на сайті довше [4].

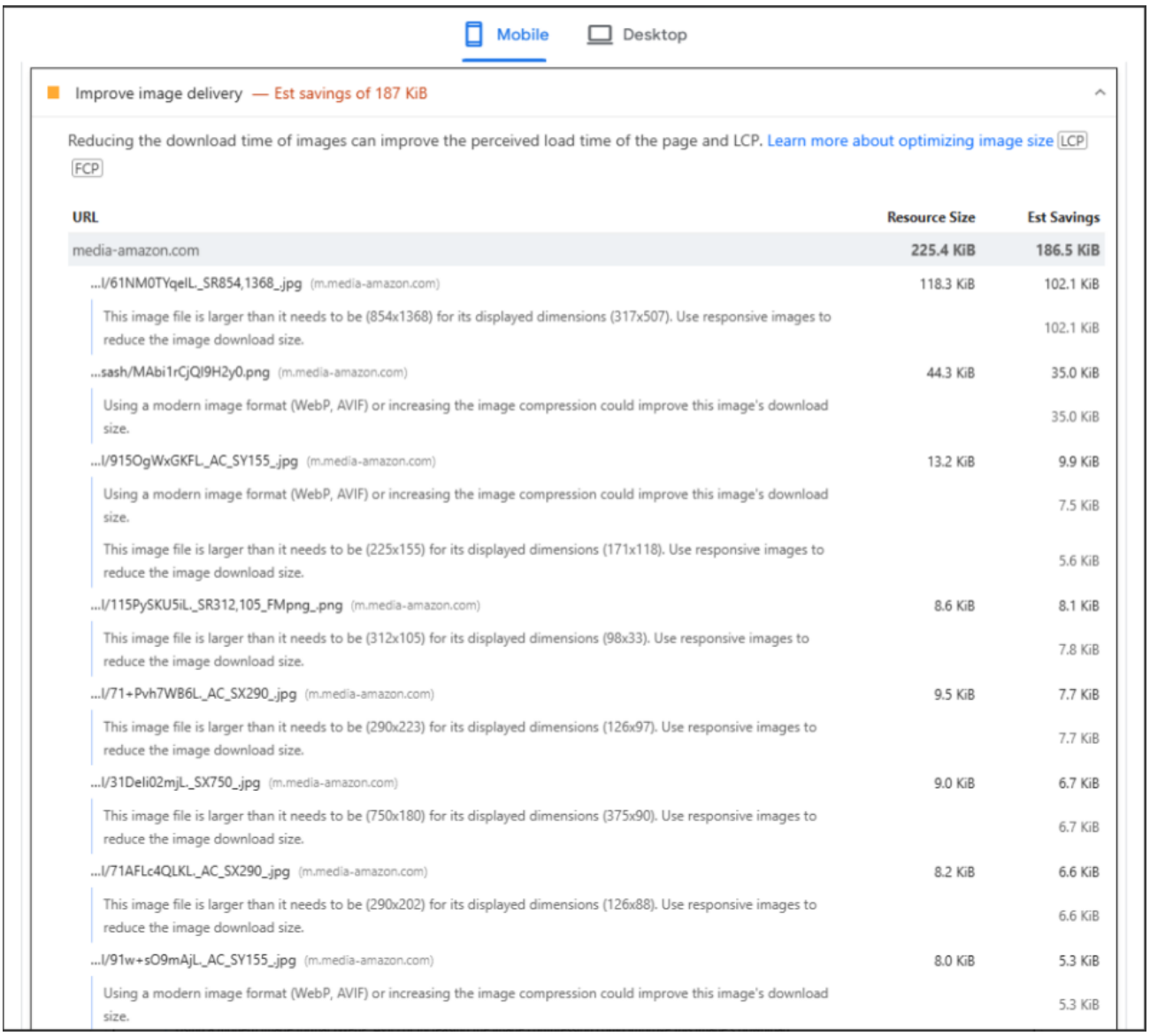

Рисунок 3.9 – Звіт PageSpeed Insights щодо доставки зображень

Ефективно використовуючи формат AVIF для заміни об’ємних зображень на веб-сайті, ви можете заощадити пропускну здатність і прискорити

завантаження всіх необхідних елементів сторінки, покращивши показники Largest Contentful Paint та Interaction to Next Paint.

Електронна комерція та фотографії товарів

Зображення становлять значну частину сторінок інтернет-магазинів, тому використання AVIF замість JPG скорочує час завантаження, але це варто перевіряти в кожному конкретному випадку.

Крім того, використання HDR краще зберігає дрібні деталі фотографій товарів без артефактів, характерних для JPG, що особливо актуально для тканин і текстур. Ви можете розраховувати на ще більшу економію при використанні зображень у форматі AVIF у функції 360-градусного огляду товару [4].

Цифрове мистецтво та прозорість

Завдяки високій ефективності стиснення без появи кольорових смуг, а також підтримці прозорості та можливостям анімації, як AVIF, так і JPEG XL є чудовими варіантами для роботи з цифровим мистецтвом та дизайном. Крім того, підтримка HDR стає у нагоді для збереження яскравих кольорів.

Додатково: JPEG XL може зберігати сплайни та точки, які не «запікаються» в оригінальне зображення, а зберігаються як окремий шар. Вони не піддаються JPEG DCT-перетворенням і не спотворюються під час стиснення. Обидва формати можуть зберігати шум окремо. У звичайному JPEG шум (навіть художній) важко стиснути.

JPEG XL може зберігати фотонний шум, подібний до цифрових камер, а AVIF додає шум, схожий на зернистість плівки – це властивість кодека AV1, на якому базується AVIF.

Якщо ваш проект із нерухомими зображеннями призначений як для веб-використання, так і для офлайн-використання, використовуйте AVIF для вбудовування у веб-сторінки та JXL для завантаження.

Для коротких відеороликів із циклічним відтворенням або кінематографів AVIF працює краще – коротко кажучи, це відеокодек із такими функціями, як оцінка руху, перероблений для зображень.

Вимоги до архівування та безвтратного стиснення

Архівування вимагає безвтратного стиснення, оскільки архівні файли часто використовуються як еталонні копії. Якщо обирати між форматами JPEG XL та AVIF, то JPEG XL буде кращим варіантом завдяки своїм кращим можливостям безвтратного стиснення [4].

Якщо вихідні зображення мають формат JPEG, їх перекодування у формат JPEG XL забезпечить зменшення розміру приблизно на 20 % без перекодування самих даних зображення.

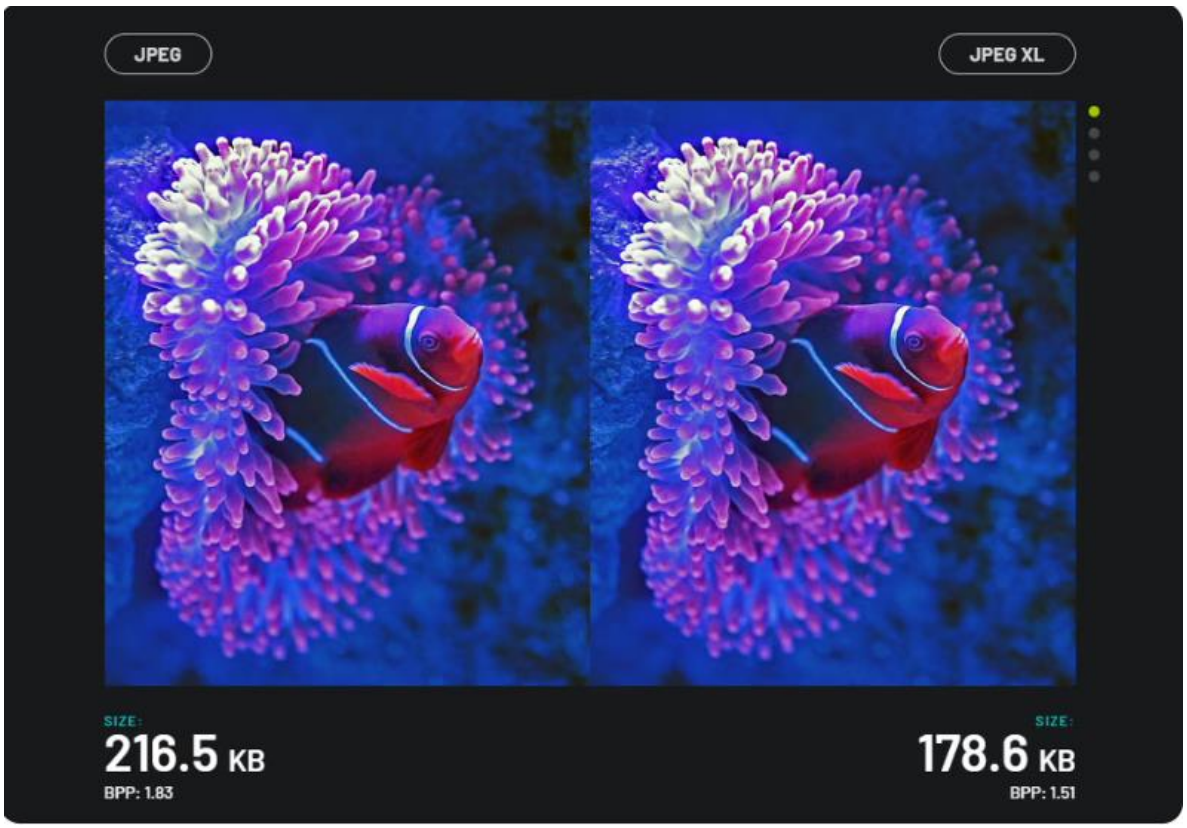

Рисунок 3.10 – Перекодування у формат JPEG XL без втрат якості

Друкована медіа та фотографія

Для друку необхідна підтримка CMYK, висока розрядність та точність — JXL підтримує 32-бітну розрядність і CMYK. AVIF не підтримує CMYK, що робить його несумісним із професійною поліграфічною галуззю. JPEG XL також підтримує шари та до 4099 каналів для зберігання інформації, такої як маски виділення та плашкові кольори [4].

Конвертація JPEG у AVIF або JPEG XL

Хоча найкращим варіантом є збереження файлу безпосередньо у форматі AVIF або JPEG XL. Ось як конвертувати зображення JPEG у нові формати: Онлайн-конвертери та інструменти. Онлайн-додатки, такі як CloudConvert і Squoosh, можуть допомогти вам конвертувати наявні зображення як у формат AVIF, так і в JPEG XL.

Squoosh навіть пропонує базовий редактор, доступний після завантаження зображення, щоб ви могли налаштувати параметри стиснення, обертати зображення та переглянути результат. Ви також можете скористатися Image CDN від Uploadcare для обслуговування файлів AVIF на різних пристроях і платформах.

Конвертацію у формат AVIF можна виконати за допомогою FFmpeg – одного з найпопулярніших інструментів командного рядка для конвертації відео. Для конвертації у формат AVIF за допомогою FFmpeg необхідно встановити бібліотеку aom та включити її до збірки FFmpeg. Параметри можуть відрізнятися, проте існують основні кроки для виконання базової конвертації у формат AVIF:

ffmpeg -i image.png -c:v libaom-av1 -still-picture 1 image.avif

FFmpeg також може конвертувати у формат JPEG XL, за умови, що бібліотека libjxl також попередньо встановлена:

ffmpeg -i image.png -c:v libjxl image.jxl

Інтеграція плагінів та CMS

У версії 6.5 WordPress запровадив вбудовану підтримку формату AVIF. Ви можете налаштувати WordPress так, щоб завантажені зображення у форматі JPG автоматично конвертувалися у формат AVIF.

Shopify також запровадив підтримку формату AVIF. Система автоматично визначає можливості пристрою користувача та за потреби відображає зображення у форматі AVIF [4].

Таблиця 3.1 – Порівняння форматів JPEG, AVIF та JPEG XL

Параметр	**JPEG**	**AVIF**	**JPEG XL**
Підтримка стиснення без втрат	-	+	+
Прозорість	-	+	+
Глибина кольорів	8-bit	12-bit	32-bit
HDR / Gamut	- / sRGB	+ / Rec.2020	+ / Rec.2020
Анімація	-	+	+
Підтримка браузерів	100%	94%, всі популярні браузери	~10%, не готовий для веб-сайтів
Найкращий у	Legacy	Web / E-комерція	Архівування та професійне застосування

JPEG є класичним і найбільш поширеним форматом стиснення зображень. Він використовує стиснення з втратами та забезпечує хороше зменшення розміру файлу при високій швидкості обробки. Основними перевагами JPEG є універсальна підтримка та простота використання. Проте при сильному стисненні помітно погіршується якість зображення та з’являються артефакти.

AVIF є сучасним форматом, який використовує більш складні методи кодування на основі кодека AV1. Він дозволяє отримати значно менший розмір файлу при кращій якості зображення порівняно з JPEG. Формат підтримує HDR, прозорість та одночасно режим із втратами та без втрат. Недоліком AVIF є високе навантаження на процесор і повільніше кодування.

JPEG XL поєднує високу якість зображення, ефективне стиснення та швидшу роботу, ніж AVIF. Формат підтримує HDR, прозорість і стиснення без втрат. Його головною перевагою є хороший баланс між якістю, швидкістю та ступенем стиснення. Недоліком поки що залишається обмежена підтримка деякими програмами та браузерами.

Отже:

-JPEG найкраще підходить для сумісності та простого використання;

- AVIF для максимального стиснення та веб-технологій;

-JPEG XL для зберігання зображень високої якості з хорошим стисненням без помітної втрати деталей. Його доцільно використовувати для професійної графіки, фотографій, архівування зображень та у тих випадках, коли важливі одночасно якість, невеликий розмір файлу та швидке відкриття зображень [4].

ВИСНОВКИ

Досліджено класичні та сучасні алгоритми стиснення зображень, які використовуються для передачі цих зображень в інфокомунікаційних мережах.

JPEG залишається одним із універсальних варіантів завдяки постійній підтримці, в той же час він помітно відстає від нових форматів. Нові JPEG XL та AVIF демонструють значно кращі результати при зменшенні розміру зображення і збереженні його якості.

JPEG XL добре працює як зі стисненням без втрат, так і з втратами, підтримує високу розрядність кольору, HDR, CMYK та дозволяє змінювати старі файли JPEG шляхом зменшення їхнього розміру приблизно на 20%. AVIF же є кращим варіантом для веб-сайтів, файли виходять меншими, при цьому якість і чіткість залишаються на високому рівні, є підтримка прозорості та анімації. Він підходить для звичайних сайтів і швидкого завантаження. Для професійної фотографії, архівів та друку доречніше вибрати JPEG XL. Розглянуті класичні методи кодування (Хаффмана, LZW, арифметичне, прогнозне тощо).

Отже, перехід на сучасні формати стиснення дозволяє суттєво зменшити обсяг зображень без помітної втрати якості. Таким чином, розвиток алгоритмів стиснення зображень залишається актуальним і важливим напрямком, який безпосередньо впливає на ефективність функціонування багатьох веб-сайтів, програм та різних систем, котрі працюють із цифровими зображеннями та фотографіями.

ПЕРЕЛІК ПОСИЛАНЬ

1. Дзяман Є. В. СИСТЕМА СТИСНЕННЯ ЗОБРАЖЕНЬ НА БАЗІ АЛГОРИТМУ QOI / МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ Чорноморський національний університет імені Петра Могили. Миколаїв, 2023. 60 с.

2. Свіріна Д. Алгоритми та методи стиснення інформації / Міністерство освіти і науки України НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ». Київ, 2025. 62 с.

3. The JPEG XL Image Coding System History, Features, Coding Tools, Design Rationale, and Future / Jon Sneyers та ін. 1 Cloudinary Research, Petah Tikva, Israel 2 Google Research, Zurich, Switzerland ¨ 3 Moving Picture Technologies, Fraunhofer IIS, Erlangen, Germany 4 Department of Electronic Systems Engineering, Takushoku University, Tokyo, Japan, 2025. 73 с.

4. AVIF vs JPEG XL vs JPEG: Which image format is best for you?. *AVIF vs JPEG XL vs JPEG: Which image format is best for you?*. *Uploadcare*. 24.10.2025. URL: https://uploadcare.com/blog/avif-vs-jpeg-comparison/?utm_source.

5. Chakole V. V. Unit 4: Image Coding and Compression. *Digital Image Processing*. Nagpur : Department of Electronics Engineering, KDKCE, 2026. С. 32.

6. Evaluation of Huffman and Arithmetic Algorithms for Multimedia Compression Standards. Iran : Department of Computer Engineering, Faculty of Engineering, University of Guilan, 2026. 11 с.

7. AVIF: what it is, how to use it, and what advantages this image format has. *Seozoom*. *SEO*. 12.09.2024. URL: https://www.seozoom.com/avif/.

8. What is an AVIF file? Everything you need to know about the AVIF format. *Uploadcare*. *Blog*. 20.11.2025. URL: https://uploadcare.com/blog/avif-image-format/?utm_source.

9. AVIF for Next-Generation Image Coding. *Medium*. *Netflix TechBlog*. 13.02.2020. URL: https://netflixtechblog.com/avif-for-next-generation-image-coding-b1d75675fe4.

10. Jyrki Alakuijala, Ruud van Asseldonk та ін. JPEG XL next-generation image compression architecture and coding tools. Zurich : Google Research Zuurich, Switzerland. Cloudinary, Israel, 2026. 13 с.

11. Горбачов С., Гордійко Н. ОЦІНЮВАННЯ ЯКОСТІ ЗОБРАЖЕНЬ СИСТЕМ МУЛЬТИСПЕКТРАЛЬНОГО МОНІТОРИНГУ БЕЗ ПОРІВНЯННЯ З ЕТАЛОНОМ. *Актуальні проблеми сучасної фізики*. Навчально-науковий Фізико-технічний інститут, 2026. С. 3.