

Evaluasi Kompresi Lossless Huffman Statis pada Sampel Teks dan Citra Komik

Wira Adiguna Sutawa^{1✉}, Nur Hamzah^{2*}, Ridho Sholehurrohman^{3✉}

¹Jurusan Statistika dan Sains Data, Sekolah Sains Data, Matematika dan Informatika,
Institut Pertanian Bogor, Indonesia

²Jurusan Matematika, Fakultas Matematika dan Ilmu Pengetahuan Alam,
Universitas Lampung, Indonesia

³Jurusan Ilmu Komputer, Fakultas Matematika dan Ilmu Pengetahuan Alam,
Universitas Lampung, Indonesia

Abstrak: Kompresi lossless diperlukan untuk mengurangi ukuran representasi data tanpa kehilangan informasi asli. Penelitian ini mengevaluasi implementasi Huffman statis pada satu sampel teks dan satu citra komik digital yang tersedia dalam naskah. Implementasi disusun kembali berdasarkan pseudocode dan diuji menggunakan teks UTF-8 sebesar 28 byte serta citra RGB beresolusi 720×1.235 piksel dengan ukuran mentah 2.667.600 byte. Evaluasi mencakup validasi dekompresi, ukuran keluaran lengkap termasuk overhead, waktu kompresi dan dekompresi, serta penggunaan memori proses. zlib level 6 digunakan sebagai pembanding teks dan PNG level 6 sebagai pembanding citra. Waktu pemrosesan diukur sebanyak 30 kali setelah satu pemanasan dan dilaporkan menggunakan median serta rentang antarkuartil. Seluruh keluaran dekompresi identik dengan masukan berdasarkan perbandingan langsung byte dan hash SHA-256; pada citra diperoleh nol piksel berubah dan mean squared error sebesar nol. Huffman menghasilkan berkas teks 71 byte dan zlib menghasilkan 36 byte sehingga keduanya memperbesar sampel teks pendek. Pada citra, Huffman menghasilkan 1.853.603 byte dengan penghematan 30,51%, sedangkan PNG menghasilkan 528.345 byte dengan penghematan 80,19% terhadap data RGB mentah. Temuan menunjukkan bahwa keberhasilan rekonstruksi lossless tidak selalu diikuti penghematan ukuran berkas karena dipengaruhi karakteristik data dan overhead. Hasil dibatasi pada sampel serta implementasi yang diuji dan belum membuktikan peningkatan waktu akses, efisiensi bandwidth, penghematan kuota, atau pengalaman pengguna aplikasi Webtoon.

Kata Kunci: Huffman statis; kompresi lossless; dekompresi; komik digital.

Abstract: Lossless compression aims to reduce the size of data representations while preserving the original information. This study evaluates a static Huffman implementation using one text sample and one digital comic image available in the manuscript. The implementation was reconstructed from the documented pseudocode and tested on a 28-byte UTF-8 text sample and a $720 \times 1,235$ -pixel RGB image containing 2,667,600 raw bytes. The evaluation covers decompression validation, complete output size including overhead, compression and decompression time, and process memory usage. Level-6 zlib and level-6 PNG serve as the text and image baselines, respectively. Processing time was measured over 30 repetitions following one warm-up run and summarized using the median and interquartile range. All decompressed outputs are identical to their inputs according to direct byte comparisons and SHA-256 hashes; image validation yields zero changed pixels and a mean squared error of zero. Huffman

produces a 71-byte text file and zlib a 36-byte stream, both exceeding the original short text size. For the image, Huffman produces 1,853,603 bytes, saving 30.51% relative to raw RGB data, whereas PNG produces 528,345 bytes, saving 80.19%. These findings show that lossless reconstruction does not necessarily imply a reduction in complete file size because data characteristics and overhead affect compression efficiency. The results are limited to the tested samples and implementations and do not establish improvements in application loading time, bandwidth efficiency, mobile data consumption, or user experience.

Keywords: static Huffman; lossless compression; decompression; digital comics.

1. PENDAHULUAN

Komik digital menggabungkan informasi visual dan teks sehingga ukuran citra menjadi salah satu faktor kebutuhan penyimpanan [1]. Webtoon digunakan sebagai media membaca digital [2], [3], berkembang dari distribusi komik cetak menuju platform daring [4], [5], dan mendukung partisipasi pembaca serta transformasi karya dalam ekosistem media [6], [7]. Dalam konteks tersebut, kompresi lossless relevan ketika rekonstruksi data yang identik perlu dipertahankan.

Huffman memberi kode berdasarkan frekuensi simbol. Pengujiannya pada dokumen RTF telah dibandingkan dengan Levenstein [8], sementara kompresi teks juga dikaji menggunakan Sequitur [9] dan Adaptive Huffman pada teks terenkripsi [10]. Keberhasilan mengurangi panjang payload belum tentu menghasilkan berkas yang lebih kecil karena keluaran perlu menyimpan informasi untuk dekompresi. Oleh sebab itu, evaluasi harus membedakan ukuran payload dari ukuran berkas lengkap [11].

Pengolahan citra digital mencakup representasi kanal dan metode kompresi [12]. Penelitian terkait meliputi kompresi PNG menggunakan Huffman berbasis Java [13], prinsip kompresi lossless [14], penerapan Huffman pada teks [15], dan metode pembentukan kode dengan redundansi minimum [16]. Aplikasi lain mencakup kompresi teks [17], data pada muatan roket [18], gambar pada jaringan 6LoWPAN [19], serta analisis kompresi file teks [20]. Rujukan tersebut memberi konteks, tetapi hasilnya tidak langsung menentukan kinerja pada sampel komik yang diuji di sini.

Permasalahan evaluasi pada naskah ini adalah perlunya kesesuaian antara istilah algoritma, simbol yang dikodekan, identitas sampel, dan cara menghitung ukuran keluaran. Penyebutan kompresi citra perlu disertai penjelasan apakah simbol merupakan nilai kanal atau kombinasi RGB. Klaim lossless juga memerlukan hasil dekompresi, bukan hanya ukuran payload atau kesamaan visual. Kesenjangan yang ditangani adalah kelengkapan evaluasi pada studi kasus ini, bukan klaim bahwa seluruh penelitian terdahulu tidak melakukan validasi tersebut.

Penelitian menggunakan Huffman statis dengan penghitungan frekuensi sebelum encoding dan kode tetap selama pengodean setiap masukan. Kontribusinya adalah evaluasi yang dapat ditelusuri dari representasi byte masukan hingga berkas lengkap dan hasil rekonstruksi, disertai perbandingan zlib untuk teks dan PNG untuk citra. Implementasi disusun kembali dari pseudocode yang tersedia dan menggunakan kode kanonik untuk menyimpan informasi dekompresi. Penelitian ini tidak mengusulkan algoritma Huffman baru.

Tujuan penelitian adalah menilai identitas hasil dekompresi, ukuran keluaran termasuk overhead, waktu kompresi-dekompresi, dan penggunaan memori pada satu panel komik. Dialog dan citra dari panel yang sama diperlakukan sebagai dua representasi masukan, bukan sampel sumber independen. Hasil dibatasi pada studi kasus yang tersedia dan tidak dimaksudkan sebagai bukti efektivitas umum atau implementasi langsung pada platform Webtoon.

2. METODE PENELITIAN

Penelitian menggunakan evaluasi eksperimental kompresi lossless pada sampel teks dan citra komik digital. Tahapan penelitian pada Gambar 1 mencakup identifikasi sampel, kompresi dengan Huffman dan pembandingan, dekompresi, validasi rekonstruksi, serta pengukuran ukuran, waktu, dan memori. Evaluasi tambahan dilaksanakan pada 2 Oktober 2026.



Gambar 1. Tahapan penelitian dan evaluasi kompresi lossless

2.1 Data dan representasi masukan

Dataset merupakan sampel kemudahan yang dipilih dari materi komik yang telah tersedia dalam naskah untuk menguji alur kompresi-dekompresi secara dapat direplikasi. Unit sumbernya adalah satu panel komik; dari panel yang sama digunakan satu citra C1 dan satu transkripsi dialog T1. Kedua representasi tersebut bukan dua sumber independen dan tidak dipilih melalui sampling acak atas episode Webtoon. Pemilihan sampel didasarkan pada ketersediaan citra dan dialog yang dapat diperiksa kembali, bukan untuk mewakili populasi konten Webtoon. Karakteristik masukan dirangkum pada Tabel 1.

Citra C1 pada Gambar 2 diekstraksi tanpa perubahan dari gambar yang tertanam dalam dokumen sumber. Atribusi pada naskah awal menyebut Webtoon, tetapi judul komik, nomor episode, tanggal pengambilan, dan tautan asal belum terdokumentasi. Karena itu, provenance sumber terbatas pada artefak yang tersedia dalam naskah. Citra yang diuji merupakan PNG RGB 8 bit per kanal berukuran 720×1.235 piksel; ukuran berkas PNG-

nya 547.127 byte. Berkas JPEG asli yang tampak pada tangkapan layar eksperimen terdahulu tidak tersedia untuk diverifikasi.

Teks T1 merupakan transkripsi dialog "IYA.. AKU TIDAK PUNYA TEMAN." yang disimpan sebagai satu baris UTF-8 tanpa BOM dan tanpa karakter baris baru. Teks terdiri atas lima kata, 21 huruf kapital, empat spasi, dan tiga tanda titik, sehingga berjumlah 28 karakter ASCII atau 28 byte; terdapat 13 simbol byte unik. Dua titik setelah IYA dan satu titik setelah TEMAN dipertahankan. Citra C1 digunakan utuh, termasuk balon percakapan, tanpa pemotongan, perubahan resolusi, konversi grayscale, atau kuantisasi warna. Penggunaan dialog dan citra dari satu panel tidak menambah jumlah sampel sumber independen.

Tabel 1. Karakteristik dataset yang diuji

Karakteristik	Teks T1	Citra C1
Jumlah masukan	1 dialog dari panel C1	1 panel komik
Representasi	UTF-8 tanpa BOM atau baris baru	PNG sumber; masukan codec berupa RGB 8 bit per kanal
Ukuran data mentah	28 byte atau 224 bit	2.667.600 byte atau 21.340.800 bit
Resolusi	Tidak berlaku	720 × 1.235 piksel
Jumlah simbol unik	13 nilai byte	256 nilai byte gabungan R, G, B
Karakteristik	5 kata; 21 huruf kapital; 4 spasi; 3 titik	889.200 piksel; 3 kanal; rentang 0–255 per kanal
Unit sumber	Satu panel yang sama dengan C1	Satu panel; bukan korpus beberapa episode



Gambar 2. Sampel citra komik yang digunakan dalam evaluasi

2.2 Implementasi Huffman statis dan format keluaran

Algoritma Huffman yang digunakan adalah Huffman statis dengan dua lintasan atas masukan. Lintasan pertama menghitung frekuensi seluruh simbol dan membangun pohon serta panjang kode [16], [21], [22]. Lintasan kedua mengodekan seluruh masukan menggunakan satu tabel kode tetap. Pohon dan panjang kode tidak diperbarui setelah setiap simbol selama encoding. Dengan demikian, implementasi ini tidak menerapkan Adaptive Huffman; konversi panjang kode menjadi kode kanonik hanya mengatur representasi kode dan tidak menjadikan algoritma bersifat adaptif. Implementasi evaluasi disusun kembali dari pseudocode yang tersedia, bukan reproduksi persis program pada tangkapan layar lama.

Simbol citra adalah setiap nilai kanal 8 bit dalam rentang 0–255. Citra RGB dibaca berurutan per baris dari kiri ke kanan dan dari atas ke bawah, dengan urutan R, G, B untuk setiap piksel. Sebagai contoh, piksel (255, 255, 255) menyumbang tiga kemunculan simbol 255, bukan satu simbol berupa triplet RGB. Satu histogram gabungan dipakai untuk seluruh kanal; tidak dibuat tiga pohon terpisah. Sebanyak 889.200 piksel menghasilkan 2.667.600 simbol byte. Semua nilai 0–255 muncul pada sampel ini, sehingga terdapat 256 simbol unik. Pada teks, simbol adalah setiap byte UTF-8.

Implementasi menggunakan perhitungan frekuensi seluruh masukan, antrean prioritas untuk penggabungan dua node berfrekuensi terendah, dan penelusuran pohon untuk menentukan panjang kode setiap simbol. Panjang kode tersebut diubah menjadi kode Huffman kanonik yang disimpan bersama simbolnya. Tabel kode berlaku tetap sepanjang pengodean satu masukan dan dibangun ulang ketika berkas masukan berbeda. Langkah operasional dirangkum pada Tabel 2.

Tabel 2. Prosedur Huffman statis dan rekonstruksi

Tahap	Prosedur
Penyiapan masukan	Bentuk urutan byte UTF-8 atau RGB. Simpan panjang asli dan, untuk citra, dimensi serta mode RGB.
Lintasan pertama	Hitung frekuensi seluruh byte. Gabungkan dua node berfrekuensi terendah sampai terbentuk satu pohon.
Pembentukan kode	Tentukan panjang kode; bentuk kode kanonik dan tabel pasangan simbol–panjang kode.
Lintasan kedua	Ganti setiap byte dengan kode tetap. Kemas bit menjadi byte dan tambahkan padding bila diperlukan.
Serialisasi	Simpan header, tabel kode, dan payload berpading sebagai satu berkas HUF1.
Dekompresi	Baca HUF1; bangun kode dari tabel; pulihkan byte sesuai panjang asli; susun RGB sesuai dimensi.

Keluaran menggunakan format eksperimen HUF1. Header berukuran 32 byte terdiri atas identitas format (4 byte), versi (1 byte), penanda kanal (1 byte), panjang masukan asli (8 byte), lebar citra (4 byte), tinggi citra (4 byte), jumlah simbol unik (2 byte), dan panjang payload dalam bit (8 byte). Tabel kode menyimpan simbol dan panjang kodenya masing-masing 1 byte, sehingga memerlukan 2K byte untuk K simbol unik. Format ini menyimpan panjang kode kanonik, bukan tabel frekuensi atau seluruh struktur pohon; informasi tersebut cukup untuk membangun kembali kode saat dekompresi. Sisa byte terakhir diisi nol sebagai padding. Decoder membedakan bit data dan padding menggunakan panjang payload dalam header.

Ukuran keluaran lengkap dihitung sebagai $32 + 2K + \text{ceil}(B/8)$ byte, dengan B adalah jumlah bit payload. Fungsi ceil membulatkan ke atas, sehingga $\text{ceil}(B/8)$ mencakup padding 0–7 bit. Perhitungan ini membedakan payload murni dari berkas yang dapat didekompresi secara mandiri. Format HUF1 belum dioptimalkan untuk meminimalkan overhead pada pesan sangat pendek; overhead yang dilaporkan merupakan karakteristik format eksperimen ini.

Tahap dekompresi membaca header dan tabel kode, membentuk kembali kode kanonik, lalu menelusuri payload untuk memulihkan seluruh byte sesuai panjang aslinya. Decoder tidak menggunakan tabel frekuensi atau pohon dari memori proses kompresi. Untuk citra, byte dipulihkan sebagai urutan RGB dengan dimensi yang tersimpan dalam header. Implementasi juga lulus sepuluh pemeriksaan round-trip tambahan yang mencakup masukan kosong, satu simbol, seluruh nilai byte, dan masukan acak deterministik; pemeriksaan tersebut bukan sampel tambahan dalam evaluasi kinerja.

2.3 Pembandingan dan metrik evaluasi

Evaluasi ukuran dan kualitas rekonstruksi melengkapi konteks penelitian kompresi teks dengan Elias Gamma [23] serta perbandingan kualitas kompresi citra digital [24]. Pembandingan untuk teks adalah zlib level 6. Pembandingan citra adalah PNG, format kompresi citra lossless dengan penyaringan baris piksel dan DEFLATE, menggunakan Pillow pada `compress_level=6` dan `optimize=False`. Kedua metode citra menerima byte RGB yang sama. Ukuran PNG mencakup seluruh berkas, sedangkan ukuran zlib mencakup seluruh stream. Perbandingan ini menilai keluaran implementasi Huffman byte-per-byte terhadap format praktis yang relevan.

Compression ratio (CR) dan persentase penghematan ruang (P) dibedakan secara eksplisit. S_{asli} adalah ukuran byte masukan dan S_{kompresi} adalah ukuran keluaran lengkap, termasuk overhead. Kedua metrik dihitung sebagai berikut.

$$CR = \frac{S_{\text{asli}}}{S_{\text{kompresi}}} \quad (1)$$

$$P = \left(1 - \frac{S_{\text{kompresi}}}{S_{\text{asli}}}\right) \times 100\% \quad (2)$$

Nilai $CR > 1$ atau $P > 0$ menunjukkan pengurangan ukuran, sedangkan $P < 0$ menunjukkan pembesaran data. Baseline citra adalah 2.667.600 byte RGB mentah, bukan ukuran PNG sumber. Dengan demikian, hasil penghematan tidak boleh ditafsirkan sebagai pengurangan ukuran berkas PNG yang telah terkompresi.

2.4 Validasi dekompresi dan sifat lossless

Keluaran setiap metode didekompresi dan dibandingkan langsung dengan masukan pada seluruh posisi byte. SHA-256 juga dihitung terhadap byte asli dan hasil dekompresi. Pada citra, dimensi dan mode RGB dipertahankan, kemudian setiap piksel sebelum dan sesudah kompresi-dekompresi dibandingkan. Metrik tambahan adalah jumlah piksel berubah, mean squared error (MSE), dan selisih absolut maksimum antarnilai kanal.

$$MSE = \frac{1}{3N} \sum_{j=1}^{3N} (x_j - \hat{x}_j)^2 \quad (3)$$

Pada (3), N adalah jumlah piksel, x berindeks j adalah nilai kanal asli, dan $\hat{x}$ bertopi berindeks j adalah nilai kanal hasil dekompresi. Indeks j mencakup seluruh 3N nilai kanal RGB. Rekonstruksi dinyatakan identik apabila seluruh byte sama, tidak terdapat piksel berubah, $MSE = 0$, dan selisih absolut maksimum = 0. Hash citra dihitung terhadap urutan byte RGB mentah, bukan terhadap berkas PNG. Validasi tidak mencakup identitas container atau metadata berkas, maupun pemulihan informasi yang hilang sebelum citra dimasukkan ke naskah.

2.5 Lingkungan dan prosedur pengukuran kinerja

Pengujian dijalankan pada Windows 11 menggunakan prosesor Intel Core Ultra 7 155H dan memori fisik sekitar 15,50 GiB, dengan Python 3.12.14, Pillow 12.3.0, dan zlib 1.3.2. Waktu diukur menggunakan `perf_counter_ns`. Setiap pasangan sampel dan metode menjalani satu pemanasan, kemudian 30 pengulangan pada masukan yang sama. Median dan interquartile range (IQR) dilaporkan; pengulangan ini bukan 30 sampel independen.

Waktu kompresi Huffman mencakup perhitungan frekuensi, pembentukan pohon dan kode, serta serialisasi keluaran. Waktu dekompresi mencakup pembacaan tabel kode dan pemulihan seluruh data. Pada PNG, pembentukan objek citra dari RGB dan pengembalian hasil decoding menjadi byte RGB termasuk dalam interval pengukuran. Ekstraksi sampel, akses berkas, dan validasi hash berada di luar interval pengukuran.

Memori diukur secara terpisah pada tiga proses baru untuk setiap pasangan sampel, metode, dan arah operasi. Metrik yang digunakan adalah median peak working set proses berdasarkan `PeakWorkingSetSize` pada Windows, dalam MiB. Nilai tersebut mencakup interpreter, pustaka, masukan, dan keluaran, bukan hanya memori algoritma. Huffman diimplementasikan dalam Python, sedangkan zlib dan PNG memakai pustaka terkompilasi; karena itu, waktu pemrosesan mencerminkan implementasi dan lingkungan yang diuji, bukan perbedaan kompleksitas algoritma secara umum.

3. HASIL DAN PEMBAHASAN

3.1 Validasi dekompresi

Seluruh metode menghasilkan data dekompresi yang identik dengan masukan, seperti dirangkum pada Tabel 3. Seluruh 28 byte teks pulih tanpa perubahan. Pada citra, seluruh 2.667.600 nilai kanal yang merepresentasikan 889.200 piksel juga pulih secara identik. Huffman dan PNG menghasilkan nol piksel berubah, $MSE = 0$, dan selisih absolut maksimum antarnilai kanal = 0. Hasil ini mendukung sifat lossless pada sampel dan implementasi yang diuji.

Tabel 3. Hasil validasi dekompresi

Sampel dan metode	Byte identik	SHA-256 sama	Piksel berubah	MSE
Teks Huffman	Ya	Ya	—	—
Teks zlib	Ya	Ya	—	—
Citra Huffman	Ya	Ya	0	0
Citra PNG	Ya	Ya	0	0

Hash SHA-256 teks asli dan hasil dekompresi untuk kedua metode:

49349c4642e1ef25e0649ecae1307ab8

3fcb82ad3bf90e7adfd8ab355498ca1

Hash SHA-256 RGB asli dan hasil dekompresi untuk kedua metode:

7b5a365831faaf18d814a76eb740bc67

c4d4702803195ab3431a3bc3bb2955b6

3.2 Ukuran hasil kompresi dan overhead

Tabel 4 menyajikan ukuran keluaran lengkap. Pada teks, payload Huffman terdiri atas 99 bit. Setelah penambahan 5 bit padding, payload memerlukan 13 byte. Penambahan header 32 byte dan tabel kode 26 byte menghasilkan berkas 71 byte, lebih besar daripada masukan 28 byte. zlib menghasilkan stream 36 byte. Penghematan negatif pada kedua metode menunjukkan bahwa kompresi tidak mengurangi ukuran akhir sampel teks pendek ini.

Tabel 4. Ukuran keluaran lengkap dan penghematan ruang

Sampel dan metode	Asli (byte)	Hasil (byte)	CR	P (%)
Teks Huffman	28	71	0,3944	-153,57
Teks zlib	28	36	0,7778	-28,57
Citra Huffman	2.667.600	1.853.603	1,4391	30,51
Citra PNG	2.667.600	528.345	5,0490	80,19

CR = $S_{\text{asli}}/S_{\text{kompresi}}$; P = $(1 - S_{\text{kompresi}}/S_{\text{asli}}) \times 100\%$. Nilai negatif pada P menunjukkan pembesaran ukuran. Baseline citra adalah byte RGB mentah.

Rincian pada Tabel 5 menunjukkan bahwa ukuran payload murni berbeda dari ukuran berkas lengkap. Ukuran 512 bit menjadi 380 bit pada dokumentasi awal tidak disertai identifikasi masukan dan rincian overhead yang dapat diverifikasi, sehingga tidak dapat dipastikan sebagai payload atau berkas lengkap. Angka tersebut, bersama ukuran teks lama 616 bit, tidak digunakan dalam evaluasi tambahan. Hasil baru memakai masukan yang teridentifikasi pada Tabel 1 dan ukuran keluaran lengkap yang dapat didekompresi secara mandiri.

Pada citra, payload Huffman berukuran 14.824.465 bit dan memerlukan 1.853.059 byte setelah penambahan 7 bit padding. Header 32 byte dan tabel kode 512 byte menghasilkan ukuran akhir 1.853.603 byte, dengan penghematan 30,51% terhadap RGB mentah. PNG menghasilkan 528.345 byte dengan penghematan 80,19% terhadap baseline yang sama. PNG menghasilkan keluaran lebih kecil pada sampel ini.

Tabel 5. Rincian ukuran berkas Huffman lengkap

Komponen	Teks T1	Citra C1
Payload murni B (bit)	99	14.824.465
Padding (bit)	5	7
Payload dan padding (byte)	13	1.853.059
Header (byte)	32	32
Simbol unik K	13	256
Tabel kode 2K (byte)	26	512
Overhead header dan tabel (byte)	58	544
Ukuran berkas lengkap (byte)	71	1.853.603

Hasil tersebut konsisten dengan perbedaan pendekatan: implementasi Huffman yang diuji menggunakan frekuensi byte, sedangkan PNG melakukan penyaringan piksel sebelum DEFLATE [26]. Namun, kontribusi masing-masing tahap PNG tidak dipisahkan melalui

eksperimen ablasi. Temuan pada teks juga menunjukkan bahwa pengurangan jumlah bit payload tidak selalu menghasilkan penghematan berkas setelah overhead diperhitungkan. Besarnya overhead Huffman dipengaruhi format HUF1 yang digunakan dan bukan batas terbaik seluruh implementasi Huffman.

Keluaran Huffman untuk citra lebih besar daripada berkas PNG sumber yang berukuran 547.127 byte. Oleh karena itu, penghematan 30,51% terhadap RGB mentah tidak mendukung klaim bahwa Huffman memperkecil PNG yang sudah terkompresi. Pengujian ini juga tidak membuktikan penurunan trafik unduhan konten Webtoon yang telah menggunakan format terkompresi.

3.3 Waktu kompresi dan dekompresi

Tabel 6. Median dan IQR waktu pemrosesan

Sampel dan metode	Kompresi (ms)	IQR (ms)	Dekompresi (ms)	IQR (ms)
Teks Huffman	0,0189	0,0025	0,0172	0,0025
Teks zlib	0,0043	0,0003	0,0011	0,0001
Citra Huffman	464,41	37,64	1.349,44	103,10
Citra PNG	47,94	11,83	10,94	3,53

IQR adalah selisih kuartil ketiga dan pertama pada 30 pengukuran setelah satu pemanasan. Nilai waktu tidak mencakup akses berkas atau validasi hash.

Tabel 6 menunjukkan bahwa pada citra, median kompresi dan dekompresi Huffman masing-masing 464,41 ms dan 1.349,44 ms, sedangkan PNG masing-masing 47,94 ms dan 10,94 ms. PNG lebih cepat pada implementasi serta lingkungan ini. Perbedaan bahasa implementasi dan optimasi pustaka membatasi generalisasi hasil. Waktu pengolahan teks yang sangat singkat juga sensitif terhadap overhead pemanggilan fungsi dan kondisi sistem.

3.4 Penggunaan memori

Tabel 7. Median peak working set proses

Sampel dan metode	Kompresi (MiB)	Dekompresi (MiB)
Teks Huffman	23,19	23,20
Teks zlib	23,28	23,22
Citra Huffman	29,15	31,57
Citra PNG	31,55	33,09

Nilai merupakan median tiga proses baru per operasi dan mencakup interpreter, pustaka, masukan, serta keluaran. Satu MiB sama dengan 1.048.576 byte.

Tabel 7 menunjukkan peak working set citra Huffman tercatat 29,15 MiB saat kompresi dan 31,57 MiB saat dekompresi, sedangkan PNG masing-masing 31,55 MiB dan 33,09 MiB. Angka tersebut menggambarkan jejak memori keseluruhan proses pada lingkungan

ini. Karena komponen runtime dan pustaka ikut terukur, perbedaan kecil tidak dapat diartikan sebagai penggunaan memori eksklusif algoritma.

3.5 Keterbatasan dan implikasi

Evaluasi ini merupakan studi kasus satu panel komik dengan dua representasi masukan: citra C1 dan transkripsi dialog T1. Sampel dipilih berdasarkan ketersediaan bahan, bukan melalui sampling yang mewakili episode, genre, kreator, atau distribusi warna seluruh konten Webtoon. Tiga puluh pengulangan waktu hanya mengukur variasi pemrosesan pada masukan yang sama, bukan memperbesar dataset. Identitas judul dan episode sumber belum terdokumentasi, dan citra tertanam belum dipastikan identik dengan JPEG masukan pada eksperimen lama. Keterbatasan ini mencegah generalisasi efektivitas metode ke seluruh platform.

Penelitian belum menguji integrasi Android, waktu loading, bandwidth, konsumsi kuota, energi perangkat, atau pengalaman pengguna. Perluasan dataset pada penelitian lanjutan harus mencakup beberapa episode dan variasi resolusi, format, kompleksitas warna, serta panjang dialog. Setiap sampel perlu disertai identitas komik, nomor episode, tautan, tanggal pengambilan, ukuran sumber, dan aturan praproses. Hasil sebaiknya dilaporkan per sampel beserta ringkasan antarsampel. Pengujian tersebut belum dilakukan dan tidak dinyatakan sebagai hasil penelitian ini.

4. KESIMPULAN

Pengujian tambahan menunjukkan bahwa implementasi Huffman statis dapat merekonstruksi sampel teks dan citra secara identik. Validasi didukung oleh kesamaan seluruh byte, kesamaan hash SHA-256, dan pemeriksaan setiap piksel citra yang menghasilkan nol piksel berubah serta MSE = 0. Kesimpulan ini berlaku pada sampel dan implementasi yang diuji.

Setelah overhead diperhitungkan, teks berukuran 28 byte menghasilkan berkas Huffman 71 byte sehingga tidak memberikan penghematan ruang. Pada citra RGB berukuran 2.667.600 byte, Huffman menghasilkan 1.853.603 byte dengan penghematan 30,51%, sedangkan PNG menghasilkan 528.345 byte dengan penghematan 80,19%. Efisiensi bergantung pada karakteristik data, representasi masukan, dan overhead format keluaran. Hasil ini tidak mendukung klaim keunggulan umum Huffman atas format citra lossless yang sudah digunakan secara praktis.

Temuan dibatasi pada studi kasus satu panel komik yang tersedia dan belum mewakili keragaman konten Webtoon. Penelitian ini tidak membuktikan peningkatan loading, penghematan kuota atau bandwidth, maupun pengalaman pengguna. Pengujian lanjutan perlu memperluas dataset lintas episode, melengkapi provenance sumber, dan menilai format terkompresi yang digunakan aplikasi pada perangkat serta jaringan nyata.

PERNYATAAN KONTRIBUSI PENULIS

Wira Adiguna Sutawa: konseptualisasi, metodologi, pengembangan perangkat lunak, analisis formal, dan penulisan draf awal. **Nur Hamzah:** supervisi, peninjauan metodologi, serta peninjauan dan penyuntingan naskah. **Ridho Sholehurrohman:** validasi hasil serta peninjauan dan penyuntingan naskah. Pembagian kontribusi ini merupakan usulan penyusunan dan belum menyatakan peran yang telah dikonfirmasi.

PERNYATAAN KONFLIK KEPENTINGAN

Para penulis menyatakan tidak memiliki konflik kepentingan finansial maupun nonfinansial yang dapat memengaruhi penelitian dan publikasi artikel ini.

PERNYATAAN PENDANAAN

Penelitian ini tidak menerima hibah khusus dari lembaga pendanaan di sektor publik, komersial, atau nirlaba.

PERNYATAAN PENGGUNAAN AI GENERATIF DAN TEKNOLOGI BERBANTUAN AI DALAM PROSES PENULISAN

Dalam penyiapan dan revisi naskah ini, ChatGPT digunakan untuk membantu penyesuaian format jurnal, perapian bahasa, penyusunan abstrak, penyeragaman gaya sitasi, pemeriksaan metadata referensi, dan penyusunan draf pernyataan penulis.

REFERENCES

- [1] H. Lubis, D. B. Srisulistiowati, and A. I. Ramdhani, "Huffman coding pada image compression," *Informatics, Science, and Technologies Journal (Jurnal GERBANG)*, vol. 12, no. 1, pp. 16–25, 2022.
- [2] A. A. S. Darmawanti, "Aplikasi Webtoon sebagai media pembelajaran membaca," *Jurnal Pendidikan dan Pembelajaran Bahasa Indonesia*, vol. 11, no. 2, pp. 201–209, 2022, doi: 10.23887/jurnal_bahasa.v11i2.773.
- [3] D. Suhada, S. G. Fauziyyah, M. D. Kurnia, and C. Hasanudin, "Pemanfaatan aplikasi Webtoon untuk meningkatkan minat baca bagi peserta didik," *Jubah Raja: Jurnal Bahasa, Sastra, dan Pengajaran*, vol. 1, no. 2, pp. 9–16, 2022, doi: 10.30734/jr.v1i2.2870.
- [4] R. S. Hanum and F. Kurniawan, "Pemanfaatan Webtoon sebagai media adaptasi dari komik cetak," *CoverAge: Journal of Strategic Communication*, vol. 14, no. 1, pp. 25–36, 2023, doi: 10.35814/coverage.v14i1.5327.
- [5] A. F. Lestari and Irwansyah, "LINE Webtoon sebagai industri komik digital," *SOURCE: Jurnal Ilmu Komunikasi*, vol. 6, no. 2, pp. 134–148, 2020, doi: 10.35308/source.v6i2.1609.
- [6] J. Nam and Y. Jung, "Exploring fans' participation in digital media: Transcreation of webtoons," *Telecommunications Policy*, vol. 46, no. 10, Art. no. 102407, 2022, doi: 10.1016/j.telpol.2022.102407.
- [7] H. Cho, "The platformization of culture: Webtoon platforms and media ecology in Korea and beyond," *The Journal of Asian Studies*, vol. 80, no. 1, pp. 73–93, 2021, doi: 10.1017/S0021911820002405.
- [8] D. Asdini and D. P. Utomo, "Analisis perbandingan kinerja algoritma Huffman dan algoritma Levenstein dalam kompresi file dokumen format .RTF," *KOMIK (Konferensi Nasional Teknologi Informasi dan Komputer)*, vol. 6, no. 1, pp. 87–99, 2022, doi: 10.30865/komik.v6i1.5739.
- [9] C. A. N. Ginting, "Penerapan algoritma Sequitur pada kompresi file teks," *KOMIK (Konferensi Nasional Teknologi Informasi dan Komputer)*, vol. 6, no. 1, pp. 333–339, 2022, doi: 10.30865/komik.v6i1.5723.

- [10] R. Syahputra, "Implementasi algoritma Adaptive Huffman Code dalam kompresi file teks terenkripsi algoritma Loki97," *Journal of Computing and Informatics Research*, vol. 1, no. 1, pp. 23–28, 2021. [Online]. Available: <https://journal.fkpt.org/index.php/comforch/article/view/114>
- [11] D. Salomon, *A Concise Introduction to Data Compression*. London, U.K.: Springer, 2008, doi: 10.1007/978-1-84800-072-8.
- [12] R. C. Gonzalez and R. E. Woods, *Digital Image Processing, Global Edition, 4th ed.* Pearson, 2017, ISBN: 978-1-292-22304-9.
- [13] L. Hermawan and Yunitasari, "Kompresi data citra PNG menggunakan algoritma Huffman coding berbasis JAVA," *Jurnal Teknologi Informasi*, vol. 7, no. 2, pp. 239–244, 2023, doi: 10.36294/jurti.v7i2.3767.
- [14] K. Sayood, *Introduction to Data Compression, 4th ed.* Waltham, MA, USA: Morgan Kaufmann, 2012.
- [15] M. L. Imani, R. R. Muhima, and S. Agustini, "Penerapan metode Huffman dalam kompresi data," in *Prosiding Seminar Nasional Sains dan Teknologi Terapan IX*, 2021, pp. 457–462. [Online]. Available: <https://ejurnal.itats.ac.id/sntekpan/article/view/2253>
- [16] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, 1952, doi: 10.1109/JRPROC.1952.273898.
- [17] R. Jannah, I. Y. Putra, H. Fikri, and S. Bilad, "Penggunaan algoritma Huffman dalam mengkompres data (file text)," *Jurnal Pendidikan Teknologi Informasi dan Komputer*, vol. 2, no. 1, pp. 33–38, 2023. [Online]. Available: <https://journal.grahamitra.id/index.php/petik/article/view/72>
- [18] M. L. Bukhori and E. E. Prasetyo, "Sistem kompresi data pada muatan roket menggunakan kode Huffman," *Teknika STTKD: Jurnal Teknik, Elektronik, Engine*, vol. 7, no. 2, pp. 270–280, 2021.
- [19] K. Y. Dewi and H. Nurwasito, "Implementasi algoritma Huffman untuk kompresi gambar pada jaringan 6LoWPAN," *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, vol. 6, no. 4, pp. 2008–2017, 2022. [Online]. Available: <https://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/10968>
- [20] L. Sodikin, T. F. Putri, and T. Hidayat, "Analisa kompresi file teks menggunakan Algoritma Huffman," *Jurnal ICTEE*, vol. 3, no. 1, pp. 10–19, 2022, doi: 10.33365/jictee.v3i1.1271.
- [21] Jamaluddin, "Analisis perbandingan kompresi data dengan fixed-length code, variable-length code dan algoritma Huffman," *Majalah Ilmiah METHODODA*, vol. 3, no. 2, pp. 41–47, 2013.
- [22] A. Pahdi, "Algoritma Huffman dalam pemampatan dan enkripsi data," *Indonesian Journal on Networking and Security*, vol. 6, no. 3, 2017.
- [23] D. Cahayati, A. M. H. Pardede, and H. Khair, "Implementasi algoritma Elias Gamma kompresi pada file teks," *ALGORITMA: Jurnal Ilmu Komputer dan Informatika*, vol. 6, no. 1, pp. 159–166, 2022, doi: 10.30829/algoritma.v6i1.11867.
- [24] W. Y. Sulistyio, M. Rosidin, M. F. Gustafi, M. A. Aziz, and A. I. Putra, "Analisis perbandingan kualitas kompresi citra digital pada media sosial," *Jurnal Riset Teknologi Informasi dan Komputer (JURISTIK)*, vol. 2, no. 1, pp. 1–6, 2022, doi: 10.53863/juristik.v2i1.473.