SUST Journal of Science and Technology Vol. 35(2) 2025, 55 -71



SUST Journal of Science and Technology

journal homepage: https://journals.sust.edu



Low-Complexity Colour Image Compression for Resource-Constrained Devices Using 4:2:0 Chroma Subsampling and a DPCM-Conditional RLE-Huffman Cascade

Md. Nasir Uddin[1], Md. Mehedi Hasan[1], and A. K. M. Fakhrul Hossain[1*]

[1]*Department of Computer Science and Engineering, Shahjalal University of Science and Technology, Sylhet – 3114, Bangladesh*

ARTICLE INFO

Keywords:

Colour image compression
Chroma subsampling
DPCM
Run-length encoding
Huffman coding
Resource-constrained devices

ABSTRACT

Colour images captured by drones, IoT cameras and embedded sensors must often be stored or transmitted by devices that cannot host transform-based or learned codecs. Lightweight run-length-based schemes suit such devices, but they compress R, G and B independently, ignore the lower sensitivity of human vision to chrominance, and perform well only on images with long runs of identical pixels. This paper presents a low-complexity lossy compression pipeline in which each stage targets one class of redundancy: BT.601 YCbCr conversion separates luminance from chrominance, 4:2:0 subsampling removes perceptual chrominance redundancy, and a per-channel cascade of first-order DPCM, conditional run-length encoding and canonical Huffman coding removes inter-pixel, run-length and symbol-frequency redundancy. The entropy-coding cascade is bit-exact, so all distortion is confined to chroma subsampling and colour-transform rounding. The bitstream is fully specified. On a corpus of 113 images in 5 disjoint content categories, the pipeline reduces images to a mean of 23.27% of their raw 24-bit size (5.58 bits per pixel) at a mean PSNR of 45.0 dB, SSIM of 0.9999 and CIEDE2000 colour difference of 0.61. An ablation study shows that Huffman coding contributes the largest share of the gain in the lossless domain and that the conditional RLE rule prevents expansion on run-poor content. In a same-image comparison on the public Kodak dataset with PNG, JPEG-LS, QOI, WebP, JPEG XL, JPEG and JPEG 2000, the pipeline produces the smallest file of any low-complexity codec tested, and it requires no dictionary or trained model, but only sparse Huffman codebooks. Compressed size correlates strongly with the mean absolute DPCM residual ($r = 0.81$) and with luminance entropy ($r = 0.64$), making performance predictable from image content.

1. Introduction

Consider a fixed IoT surveillance camera or a reconnaissance drone capturing 4K colour frames. Each uncompressed 3840×2160 RGB frame occupies roughly 24.9 MB; at even one frame per second, a 30-minute mission produces about 45 GB of raw pixel data. A typical 1–2 Mbps LTE or RF uplink would need more than two full days to transmit what the camera captured in half an hour, and the on-board flash of an embedded platform would be exhausted long before the mission ends. Surveys of compression for wireless sensor networks and edge computing identify this imbalance between data generation and link or storage capacity as a central constraint of IoT deployments [1, 2]. The gap between what such devices capture and what they can store or transmit is not narrowing; it is widening with every generation of sensor. For this class of platform, compression is not an optimisation; it is the difference between a deployable system and an inoperable one.

The raw material for closing this gap is abundant, because image data is saturated with redundancy of three

* Corresponding author.
E-mail address: fakhrul-cse@sust.edu (A. K. M. F. Hossain)

https://doi.org/10.63512/sustjst.2025.2006

Copyright© 2025 SUST. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

distinct kinds. Pixel values within a channel change slowly, producing long runs of similar values (spatial redundancy) and strong correlation between adjacent samples (inter-pixel redundancy). And the human visual system is markedly more sensitive to brightness than to colour detail, so the chrominance content of an image carries perceptual redundancy that can be discarded at little visible cost [3, 4]. A compression method earns its keep by how completely it harvests these three sources.

Measured against that standard, the lightweight methods most suited to constrained devices fall short in a structural way. Plain run-length encoding (RLE) and its optimised variants [5, 6, 7] harvest only one kind of redundancy, literal repetition, and so live or die on image content: an image without long identical runs compresses poorly or even expands. They process R, G and B independently, leaving cross-channel and perceptual redundancy untouched. Most tellingly, the literature around these methods rarely articulates which redundancy each processing stage is meant to remove, so when a pipeline performs well on one image and badly on the next, there is no framework for explaining why and no principled path to fixing it.

Our motivation stems from a practical gap between two classes of methods. Standardised codecs such as JPEG [8], JPEG 2000 [9], WebP [10] and JPEG XL [11] offer excellent rate–distortion performance, but their transforms, context models and dictionaries impose code-size, memory and energy costs that are significant on microcontroller-class hardware [12], and learned codecs [13, 14, 15] additionally require accelerator hardware. At the other extreme, low-complexity schemes such as JPEG-LS [16], QOI [17] and RLE-based methods [5, 6, 7, 18] are inexpensive, but they are lossless and therefore cannot exploit the perceptual redundancy of chrominance, and RLE-based schemes are further limited by their dependence on long runs. There is, therefore, a need for a method that uses only simple integer operations, requires neither a dictionary nor a trained model, keeps distortion bounded and attributable to a single stage, and still exploits chrominance redundancy.

The objective of this work is to design, fully specify and experimentally evaluate such a method, and to quantify the contribution of each of its stages. To this end, we compose five classical stages so that each removes one class of redundancy and prepares the signal for the next, and we evaluate the resulting pipeline on a documented corpus and a public benchmark against lossless, lossy codecs, and discuss learned codecs as well. The main contributions of this paper are as follows:

1. **A redundancy-matched pipeline.** YCbCr conversion, 4:2:0 chroma subsampling, DPCM, conditional RLE and canonical Huffman coding are each justified against a named redundancy class and ordered so that each stage produces the statistical structure that its successor exploits (Section 3, Table 2).
2. **A fully specified, bit-exact codec.** The bitstream format and size accounting are specified completely, and the reconstruction path inverts every coding stage exactly, confining all distortion to chroma subsampling and colour-transform rounding (Section 4).
3. **A controlled evaluation.** A stage-wise ablation (Section 6.3), a same-image comparison with lossless, near-lossless, lossy (and discussed learned) codecs on the Kodak dataset (Section 6.4), multiple fidelity metrics (PSNR, SSIM, CIEDE2000 and VIF; Section 6.2), and measured runtime and memory (Section 6.8).

The remainder of this paper is organised as follows. Section 2 reviews related work on standardised, low-complexity, RLE and Huffman-based, and learned image compression, and summarises the research gap. Section 3 explains the background and design rationale of each pipeline stage. Section 4 presents the compression and reconstruction paths, the bitstream format and the implementation. Section 5 describes the datasets, baselines and evaluation protocol. Section 6 reports the results, including the ablation study, the benchmark comparison, perceptual quality, and runtime and memory measurements. Section 7 discusses the findings and limitations, and Section 8 concludes the paper and outlines future work.

2. Related Work

This section reviews four families of methods relevant to the target setting, namely devices without accelerators and with limited memory, and closes by stating the research gap (Table 1).

2.1. Standardised and general-purpose codecs

JPEG [8] converts RGB to YCbCr, optionally subsamples the chrominance planes, and codes 8×8 DCT blocks using quantisation, run-length coding of zero coefficients and Huffman coding. It is therefore the closest standardised relative of the present pipeline, which differs in replacing the transform and quantisation stages with lossless prediction, so that luminance is preserved to within rounding. JPEG 2000 [9] replaces the DCT with a discrete wavelet transform and context-adaptive arithmetic coding, while WebP [10] and JPEG XL [11] add intra prediction, adaptive transforms and context modelling, achieving considerably better rate–distortion performance. General-purpose lossless formats such as PNG combine per-row prediction filters with DEFLATE, that is, LZ77 dictionary matching followed by Huffman coding [19, 20].

These codecs are the natural reference for compression efficiency, but their transforms, context models and dictionary windows carry code-size, memory and energy costs that are significant on ultra-low-power processors; even a baseline JPEG encoder required dedicated parallelisation and optimisation to run energy-efficiently on such a platform [12]. Recent work evaluating visually lossless compression across codecs and imaging modalities [21] further illustrates that visual losslessness must be established for each codec and type of content, rather than inferred from a fixed PSNR threshold [22].

2.2. Low-complexity and embedded image compression

JPEG-LS, based on the LOCO-I algorithm [16], is the standard low-complexity lossless and near-lossless codec. It uses a median edge-detecting predictor, simple context modelling and Golomb–Rice coding, and its near-lossless mode bounds the per-sample error by a user-set tolerance. QOI [17] trades compression for speed with a byte-

oriented format that combines run-length, index and difference codes. LiCo [23] is a recent lossless compressor designed for both high compression and high throughput. RAGE [18] targets embedded graphics: it combines generalised deduplication with a run-length stage, supports pixel-level random access on an ARM Cortex-M33 microcontroller and offers a lossy variant; its authors report that it outperforms JPEG on discrete-tone graphics, whereas JPEG remains better on most continuous-tone natural images. Surveys of compression for sensor networks [1] and edge computing [2] emphasise that such methods are selected as much for their memory and energy budgets as for their compression ratio. Common to the lossless members of this family is that they cannot exploit the perceptual redundancy of chrominance, which is where the present pipeline obtains a large and predictable share of its gain.

2.3. RLE- and Huffman-based hybrids

Run-length encoding is a classic lossless method that has been repeatedly applied to image compression and subsequently optimised [5], but its effectiveness is strongly content-dependent: images without long identical runs compress poorly and can even expand, and plain RLE captures neither spatial correlation nor edge continuity nor frequency-domain structure. Spatial variants that trace paths of similar pixels before applying RLE improve matters for grayscale content [6], yet they are neither designed for multi-channel colour images nor benchmarked against modern codecs. Hybrid schemes combining RLE with extended-ASCII character replacement report strong lossless ratios on standard colour test images [7], but they process R, G and B independently, leaving cross-channel and perceptual redundancy unexploited, and can require large temporary dictionaries. Earlier run-length work on binary and document images [24] inherits the same channel-blind limitation. Huffman-based schemes have also been refined for lossless image coding [25, 26] and, more recently, applied to edge-node data to reduce IoT network traffic [27]; these schemes improve entropy coding but likewise leave chrominance redundancy untouched.

2.4. Learned image compression

End-to-end learned codecs now define the state of the art in rate–distortion performance. The scale-hyperprior model [13] introduced a learned side-information prior for entropy coding, and subsequent work added discretised Gaussian-mixture likelihoods with attention modules [14] and efficient space–channel context models [15]. Learned lossless and near-lossless schemes combine a lossy learned layer with residual coding to bound the reconstruction error [28]. These models, however, contain millions of parameters and require GPU- or NPU-class hardware for practical encoding speed, which places them outside the target setting of this work; we include one of them as a rate–distortion reference in Section 6.4.

2.5. Research gap

Taken together, existing methods fall into two groups with a gap between them. Transform-based and learned codecs exploit perceptual redundancy, but at a complexity that is high for microcontroller-class devices, while low-complexity codecs are either lossless, and therefore cannot remove chrominance redundancy, or depend on long natural runs. None composes complementary stages so that perceptual, spatial and statistical redundancy are each removed by the stage best suited to it, while keeping the distortion attributable to a single, identifiable operation. Table 1 summarises this gap.

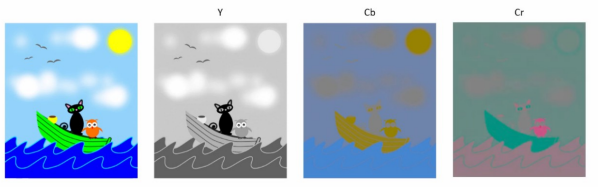


Figure 1. RGB decomposition and YCbCr transform of a colour image: the R, G and B channels are recombined into a luminance plane (Y') and two chrominance planes (Cb, Cr).

3. Background and Design Rationale

The pipeline's effectiveness comes not from any individual technique, since all five are classical, but from their composition: each stage is selected because it removes one specific redundancy class and, in doing so, reshapes the signal into precisely the form the next stage needs. This section introduces each block and its rationale, and Table 2 distils the argument. Each rationale is tested experimentally by the ablation study in Section 6.3.

3.1. Colour-space conversion (YCbCr)

The YCbCr transform separates an image into a luminance component (Y') and two chrominance components (Cb, Cr). Because human vision is far more sensitive to brightness than to colour detail, this separation allows the colour channels to be compressed more aggressively than luminance [3]. For 8-bit R, G, B values in the range 0–255, the conversion follows the ITU-R BT.601 digital standard [4]:

$$Y' = 0.299R + 0.587G + 0.114B \quad (1)$$

$$Cb = 128 - 0.168736R - 0.331264G + 0.5B \quad (2)$$

$$Cr = 128 + 0.5R - 0.418688G - 0.081312B \quad (3)$$

The results are rounded to the nearest integer and clipped to $[0, 255]$. The implementation rounds these coefficients to three decimal places; the resulting deviation is below the 8-bit quantisation step and does not alter any reported metric. Because the luminance coefficients sum to one and each set of chrominance coefficients sums to zero, achromatic pixels ($R = G = B$) map exactly to $Y' = R$ and $Cb = Cr = 128$, so that, with round-to-nearest arithmetic, they are reconstructed without error.

Design rationale. RGB channels are mutually correlated in a way that does not align with human perceptual sensitivity, so any method that compresses R, G and B independently, as the RLE family does, is structurally unable to protect the information viewers actually notice while discarding what they do not. YCbCr is the minimal fix:

Table 1. Positioning of this work relative to existing methods.

Method [ref.]	Year	Colour handling	Exploits chroma redundancy	Dictionary / model memory	Complexity	Main limitation for the target setting
RLE + optimisation [5]	2019	Per-channel RGB	No	None	Very low	Depends on natural runs; can expand run-poor data
Spatial path-based RLE [6]	2021	Grayscale only	No	None	Low	No native colour support
RLE + character replacement [7]	2011	Independent RGB	No	Large symbol dictionary	Low	Dictionary memory; channel-blind
PNG / DEFLATE [19, 20]	1996–97	RGB with row filters	No	32 KB window	Low–moderate	Lossless only
JPEG-LS [16]	2000	Per component	No	Small context tables	Low	Lossless / near-lossless only
QOI [17]	2021	RGB(A) pixel-wise	No	64-entry index	Very low	Lossless only
LiCo [23]	2024	RGB pixel	No	Multistage pipeline	Low	Lossless only
RAGE [18]	2024	RGB pixel chunks	No (bit-level pruning in lossy mode)	Base dictionary	Low	Tuned to discrete-tone graphics
JPEG, JPEG 2000, WebP, JPEG XL [8, 9, 10, 11]	1992–2019	YCbCr (JPEG, WebP) or component transforms	Yes	Transform and context state	Moderate–high	Code size, memory and energy cost on MCUs [12]
Learned codecs [13, 14, 15, 28]	2018–2024	Learned latent space	Implicitly	Millions of parameters	Very high	Requires GPU/NPU
This work	–	YCbCr 4:2:0	Yes	No dictionary or trained model; sparse Huffman codebooks and 3 KB of tables	Very low	Single operating point

a fixed linear transform, exactly invertible in real arithmetic and invertible to within one 8-bit quantisation step in the integer implementation, costing a handful of multiply–adds per pixel, with none of the per-image overhead of adaptive decorrelating transforms. Yet it establishes the luma/chroma separation on which every downstream decision in this pipeline rests. Figure 1 illustrates the decomposition: the RGB channels of a colour image are recombined into one luminance plane carrying nearly all structural detail and two low-detail chrominance planes.

3.2. Chroma subsampling

Following conversion, the chrominance channels are spatially subsampled so that fewer chroma samples are stored than luma samples. The widely adopted 4:2:0 scheme halves Cb and Cr resolution both horizontally and vertically; for every 2×2 pixel block, the retained chrominance sample is the block average [3]:

$$\tilde{C}(m,n) = \text{round}\left(\frac{1}{4}\sum_{i=0}^{1}\sum_{j=0}^{1} C(2m+i, 2n+j)\right) \quad (4)$$

where $C \in \{Cb, Cr\}$, (m,n) indexes the 2×2 blocks and round$(\cdot)$ denotes round to nearest integer. Planes with an odd width or height are first padded by replicating the last column or row. The block samples are written $C(\cdot,\cdot)$ rather than with single letters to avoid a clash with the blue channel B in Eqs. (1)–(3).

Design rationale. This is the pipeline's only designed lossy operation, and its placement is deliberate: it sits immediately after colour conversion, before any entropy coding, because it is the cheapest large reduction available anywhere in the pipeline: a guaranteed 4× cut in chrominance data, targeting the information that the human visual system resolves least well, with no content-adaptive machinery. Figure 2 contrasts the 4:4:4, 4:2:2 and 4:2:0 schemes: chroma resolution falls progressively while the luma grid is untouched. Subsampling first also means that the entropy-coding stages never spend effort compressing chrominance detail that would not have been perceptually missed. As a second-order benefit, block averaging pushes each retained sample toward the local mean, smoothing the chroma planes and strengthening the inter-sample correlation on which the next stage relies. Because PSNR computed over RGB underweights exactly the chrominance error that this step introduces, its cost is quantified in Section 6.2 with colour-difference and per-channel metrics.

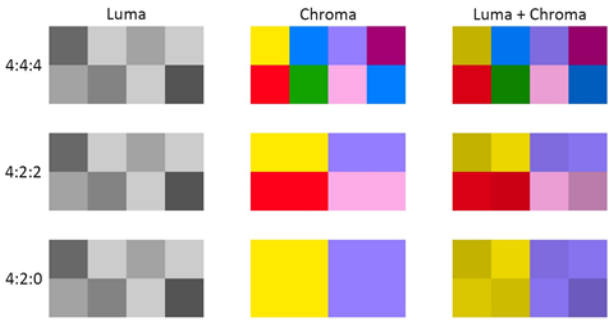


Figure 2. Chroma subsampling schemes (4:4:4, 4:2:2, 4:2:0): progressive reduction in colour sampling while luminance detail is fully preserved.

3.3. Differential pulse-code modulation (DPCM)

DPCM predicts each sample from its neighbour and stores only the residual difference [29, 30]. Each channel is processed in row-major raster order, continuing from the end of one row to the start of the next, and residuals are computed modulo 256 and remapped to the signed range $[-128, 127]$:

$$d_i = ((p_i - p_{i-1} + 128) \bmod 256) - 128; \quad p_{-1} = 0 \quad (5)$$

where p_i is the i-th sample of the channel in raster order. The decoder recovers the samples exactly by

$$p_i = (p_{i-1} + d_i) \bmod 256. \quad (6)$$

This keeps the residual alphabet at 256 symbols. It halves the worst-case Huffman codebook relative to the naïve $[-255, 255]$ difference range, and the mapping remains exactly invertible by Eq. (6). Because adjacent image pixels are highly correlated, the residuals cluster tightly around zero, lowering both entropy and dynamic range. Figure 3 shows a worked example: a smooth pixel run collapses into a residual stream dominated by small values.

Pixels: [100, 101, 102, 103, 99, 98, 99, 99]

DPCM: [100, -1, -1, -1, -4, 1, 1, 0]

Figure 3. DPCM example: the first value is stored relative to the initial predictor; each subsequent value is the difference from its predecessor, producing small, near-zero residuals from smooth pixel runs.

Design rationale. This is the pivotal stage of the cascade. RLE and Huffman coding are both frequency-driven: they compress well only when a signal exhibits long identical runs or a skewed symbol distribution, properties that raw photographic pixels rarely have. DPCM manufactures those properties. A one-tap predictor (the previous pixel) converts a wide-range, high-entropy pixel stream into a narrow residual distribution concentrated near zero, at the cost of one subtraction per pixel and with perfect reversibility. The ablation in Section 6.3 quantifies the effect of removing it. DPCM is, in effect, the translator between the statistics images naturally have and the statistics entropy coders need.

3.4. Conditional run-length encoding (RLE)

RLE replaces consecutive runs of the same value with a single value–count pair. Applied to raw pixels it is unreliable; applied to DPCM residuals, which contain long runs of identical near-zero values, it is highly effective. This pipeline applies RLE conditionally: for each channel, the encoder estimates the size of the RLE representation, at 2 bytes per value plus 1 byte per count, that is, 3 bytes per value–count pair, against the size of the raw residual stream at 2 bytes per residual, and keeps whichever is smaller. The 16-bit value field reflects the signed $[-128, 127]$ alphabet of Eq. (5) and records the choice in a mode flag in the channel header (Table 3). Run counts are limited to $[1, 255]$; longer runs are split. The rule is evaluated on the residual stream before entropy coding, so it is a size estimate rather than an exact post-Huffman comparison.

Design rationale. RLE is a structural match for the signal that DPCM emits, but only when runs actually exist. The conditional rule converts RLE's classic weakness, expansion on run-poor content, into a non-event: by construction, the stage never increases the size of the stream it receives. Empirically, it engages on 176 of the 339 channel streams of our corpus, and far more often on chrominance than on luminance: 30 of 113 luma streams against 146 of 226 chroma streams. Luminance residuals contain too few exact repeats to pay for a count field, whereas the smoothed, subsampled chrominance planes frequently do. The split by content is sharper still: the rule engages on only 1 of the 72 streams of the high-chroma-variance set against 51 of 72 for the low-chroma-variance set and all 48 chroma streams of the grayscale set, which is the mechanism analysed in Section 6.6. On the Kodak benchmark, whose content is uniformly photographic, it engages on 20 of 72 streams and never on luminance. The effect of the rule itself is isolated by configurations A5 and A6 of the ablation (Table 6). Dictionary-based schemes such as LZW [31] can capture repeated patterns that RLE misses, but they carry dictionary-storage overhead, a real cost on memory-constrained targets; Section 6.4 compares the pipeline against DEFLATE and PNG, which follow this approach. Conditional RLE harvests the bulk of the remaining run redundancy with no dictionary and no risk of expansion, which is why it appears here rather than as the front-line coder it is usually, and unreliably, deployed as.

Table 2. Module design rationale: the redundancy each stage removes and the structure it hands to its successor.

Module	Redundancy removed	What it hands the next stage	Tested by
RGB → YCbCr	Cross-channel perceptual entanglement	A perceptually ordered signal: critical luma isolated from tolerant chroma	A1
4:2:0 chroma subsampling	Perceptual redundancy in chrominance (4× reduction)	Smoothed chroma planes with increased inter-sample correlation	A2, A3
DPCM	Inter-pixel (spatial) correlation	A narrow, near-zero residual distribution rich in long identical runs	A4
Conditional RLE	Run redundancy in the residual stream	A compact value–count symbol stream with pronounced frequency skew	A5, A6
Huffman encoding	Remaining symbol-frequency skew	The final prefix-coded bitstream	A7

3.5. Huffman encoding

Huffman encoding assigns shorter prefix-free codes to more frequent symbols, achieving the minimum average code length among symbol-wise prefix codes [32]. We use the canonical form: because canonical codes are fully determined by their lengths, each channel's codebook is serialised as code lengths alone, shrinking header overhead and simplifying decoder-side reconstruction [25, 26], both directly relevant on the memory-constrained targets motivating this work.

Design rationale. Even after DPCM and RLE, the surviving symbol stream retains a skewed frequency distribution, since certain residual values and run lengths dominate. Huffman coding removes this final layer of statistical redundancy in linear time, without the adaptive probability modelling that arithmetic or range coding requires. That trade is intentional: arithmetic coding [33] can approach the empirical entropy of the symbol stream more closely, at the cost of adaptive probability modelling and higher implementation complexity. Rather than estimating this gain, we measure it: Section 6.3 reports the gap between the average Huffman code length and the zeroth-order empirical entropy of the final symbol stream, which bounds what any zeroth-order arithmetic coder could gain. Over our corpus, that gap averages 2.1% of the coded payload per image (median 1.0%, maximum 7.6%; 1.6% when all images are pooled), and 1.3% on Kodak, so the headroom for a stronger zeroth-order coder is small. Huffman coding is the point on the ratio-versus-cost curve that this pipeline is designed to occupy.

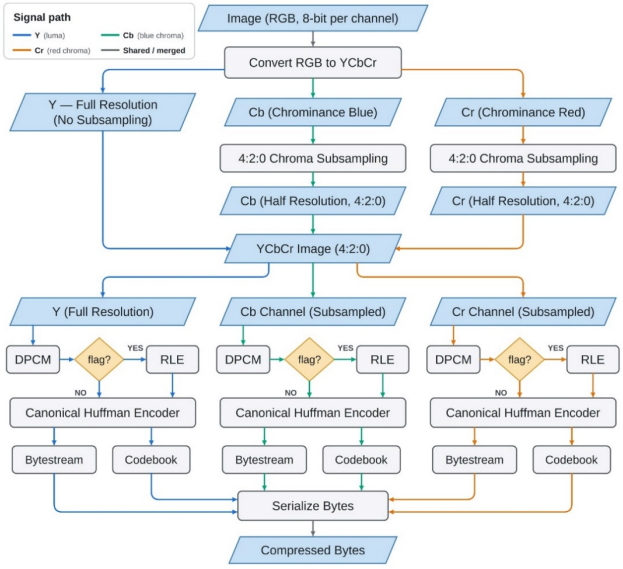


Figure 4. Compression architecture.

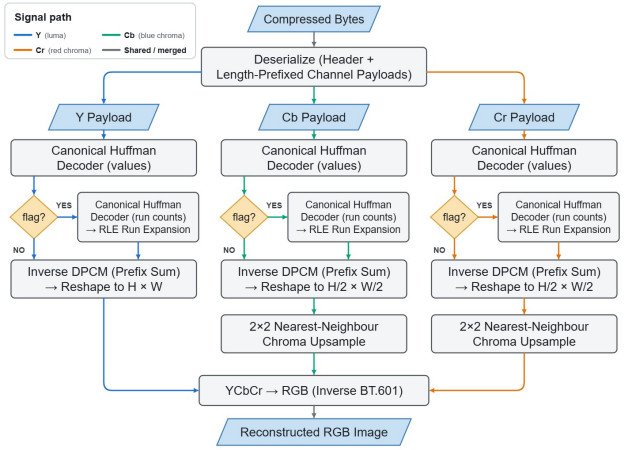


Figure 5. Decompression (reconstruction) architecture.

4. Proposed Method

The system is an end-to-end pipeline: a raw colour image enters, a compact serialised bitstream leaves, and a matching reconstruction path recovers the image from that bitstream alone. The compression flow is shown in Figure 4; reconstruction inverts it stage by stage (Section 4.2, Figure 5).

4.1. Compression path

Each RGB pixel is first converted to YCbCr via Eqs. (1)–(3), and the chrominance planes are 4:2:0-subsampled via Eq. (4) (Sections 3.1–3.2). The Y', Cb and Cr channels are then compressed independently through the three-stage lossless cascade of DPCM, conditional RLE and Huffman coding described in Sections 3.3–3.5. Every coding stage is exactly reversible, and the three compressed channels are concatenated and serialised into a single bitstream whose format is given in Section 4.3. Single-plane grayscale images bypass the colour transform and subsampling and are coded as one plane, so that the pipeline is lossless for them. Algorithm 1 summarises the compression path.

4.2. Reconstruction path

Reconstruction inverts each stage in reverse order (Figure 5). The bitstream is deserialised into its per-channel sub-streams; each passes through Huffman decoding, RLE expansion (if the channel's mode flag indicates RLE) and inverse DPCM (Eq. (6)), recovering the subsampled YCbCr image bit-exactly. The Cb and Cr planes are then upsampled to full resolution, the single point at which reconstruction differs from the original, since the four chrominance samples averaged during compression cannot be individually recovered. The image is finally converted back to RGB by the inverse BT.601 transform, with rounding to the nearest integer and clipping to $[0, 255]$:

$$R = Y' + 1.402(Cr - 128) \tag{7}$$

$$G = Y' - 0.344136(Cb - 128) - 0.714136(Cr - 128) \tag{8}$$

$$B = Y' + 1.772(Cb - 128) \tag{9}$$

Chroma upsampling uses nearest-neighbour replication: each stored sample is copied into its 2×2 block, and any padding added before subsampling is removed. Reconstruction error therefore has exactly two sources: 4:2:0 subsampling, which dominates, and the 8-bit rounding of the colour transform, whose contribution alone is measured by ablation configuration A2. The entropy-coding backbone contributes no error at all: decoding recovers the subsampled YCbCr planes bit-exactly, a property we verify on every image (Section 6.2). Algorithm 2 summarises the reconstruction path.

4.3. Bitstream format and size accounting

Table 3 specifies every field of the compressed file, as produced by the implementation. All sizes reported in this paper are the length of this file in bytes, including every header, codebook and padding bit; the arithmetic was verified byte-for-byte against the serialised output on all 113 corpus images and all 24 Kodak images. The raw size used as the reference is $W \times H \times 3$ bytes (24 bits per pixel); it does not depend on the size of the source file (PNG or JPEG) from which the pixels were decoded.

4.4. Implementation

The pipeline is implemented as independent, swappable modules for colour conversion, subsampling and each entropy-coding stage. Any component (the entropy coder, the predictor, the subsampling scheme) can be replaced without touching the rest, which makes the extensions discussed in Section 8 (arithmetic coding, adaptive subsampling) drop-in upgrades rather than redesigns. The reference implementation is written in Python 3.12.3 (NumPy 2.4.4, Pillow 12.1.1, scikit-image 0.26.0, pandas 3.0.2); Algorithms 1 and 2 give the compression and reconstruction procedures.

Table 3. Bitstream format of the proposed codec. Sizes in bytes; all integers little-endian.

Element	Size (B)	Contents
File header	38	Magic "ENTR" (4); format version (1); image width and height (4 each, uint32); subsampling mode (1: 0 = 4:4:4, 1 = 4:2:2, 2 = 4:2:0); the six plane dimensions Y', Cb, Cr (2 each, uint16); the three channel block lengths (4 each, uint32).
Channel order	—	Y', then Cb, then Cr, each a self-contained block.
Channel header	13 (RAW) / 17 (RLE)	Mode flag (1: 0 = RAW, 1 = RLE); number of reconstructed samples (4); number of RLE pairs (4); length of the values block (4); in RLE mode, length of the counts block (4).
Huffman block header	13 per block	Symbol count (4, uint32); padding bits in the final byte (1); payload length (4); codebook length (4). One block in RAW mode; two in RLE mode, since values and run counts are coded with separate codebooks.
Codebook	$4 + 3n$	Symbol count (4), then per symbol a signed 16-bit symbol value and a 1-byte code length; canonical codes are rebuilt from the lengths. The alphabet is sparse and signed, so lengths are stored per present symbol rather than as a fixed 256-entry table: n averages 101 symbols per stream over our corpus (range 2–257), and the longest code observed is 23 bits.
RAW-mode symbols	—	One symbol per DPCM residual, over the alphabet $[-128, 127]$ of Eq. (5).
RLE-mode symbols	—	A values stream over the same alphabet and a counts stream over $[1, 255]$, each Huffman-coded independently; runs longer than 255 are split.
Payload	⌈bits/8⌉	Canonical codewords concatenated most-significant-bit first and zero-padded to a byte boundary; the pad length is stored in the block header.
Padding and rounding	—	Odd chrominance dimensions padded by replicating the last row or column before averaging; round-to-nearest with clipping to $[0, 255]$ in both colour transforms and in the 2×2 chroma mean.

Algorithm 1: Compression

Input: image I of size $W \times H$; mode (colour or grayscale). **Output:** bitstream S.

1. write file header (magic, version, W, H, subsampling mode, plane sizes) to S
2. $(Y', Cb, Cr) \leftarrow$ BT.601(I) by Eqs. (1)–(3), round to nearest, clip to $[0, 255]$
3. pad Cb, Cr to even size by edge replication; $Cb \leftarrow$ Sub420(Cb); $Cr \leftarrow$ Sub420(Cr) by Eq. (4)
4. planes $\leftarrow [Y', Cb, Cr]$
5. **for** each plane X in planes **do** (6–11)
6. $\quad d \leftarrow$ DPCM(X) by Eq. (5), row-major raster order, first sample stored literally
7. $\quad (v, c) \leftarrow$ RLE(d), counts split at 255
8. $\quad$ **if** $3|v| < 2|d|$ **then** (mode, streams) $\leftarrow$ (RLE, $[v, c]$) **else** (mode, streams) $\leftarrow$ (RAW, $[d]$)
9. $\quad$ write channel header (mode, $|X|$, $|v|$, block lengths) to S
10. $\quad$ **for** each stream in streams **do** (11)
11. $\quad\quad$ build canonical Huffman codebook; write block header, payload and codebook to S
12. **return** S

Algorithm 2: Reconstruction

Input: bitstream S. **Output:** reconstructed image $\hat{I}$.

1. read file header (W, H, mode flags)
2. **for** each channel header in S **do** (3–5)
3. $\quad$ read block header(s) and codebook(s); decode the values stream, and the counts stream in RLE mode
4. $\quad$ **if** mode = RLE **then** $d \leftarrow$ expand(values, counts) **else** $d \leftarrow$ values
5. $\quad X \leftarrow$ inverse DPCM of d by Eq. (6)
6. $Cb, Cr \leftarrow$ nearest-neighbour 2×2 upsampling; crop to (H, W)
7. $I \leftarrow$ inverse BT.601 by Eqs. (7)–(9), round to nearest, clip to $[0, 255]$
8. **return** I

5. Experimental Setup

5.1. Datasets

The evaluation uses a frozen corpus of 113 unique images organised into five disjoint sets (Table 4), together with one public benchmark. Every image belongs to exactly one set, so the set sizes sum to the corpus total, and every table and figure in Section 6 states the set(s) and number of images it uses. Set A is a heterogeneous collection of photographs, graphics and screen content, deliberately chosen to stress the pipeline across heterogeneous visual data. Sets B and C contain images with low and high chroma variance, respectively, and probe the content-dependence hypothesis of Section 6.6: Set B (sky, studio backgrounds, sparse graphics) is expected to compress well because of long DPCM/RLE runs, and Set C (saturated, finely textured photographs) is expected to compress poorly because of short runs and weak symbol skew. Set D1 and D2 contain the same 24 images in colour and in grayscale to draw comparison between colour and grayscale image compression (Section 6.5).

To enable comparison with published results, all experiments are repeated on the Kodak Lossless True Color Image Suite (24 images, 768×512) [34], which is distributed in a lossless format. This matters for fidelity: images decoded from JPEG files have usually already been chroma-subsampled at 4:2:0 by their original encoder, since 4:2:0 is the common default in consumer encoding practice [35], so re-applying 4:2:0 subsampling to them loses little additional information and would inflate the measured fidelity. We therefore report fidelity separately for lossless-source and JPEG-source images.

Sets D1 and D2 contain the same 24 scenes in colour and in grayscale, so the corpus contains 89 distinct scenes in 113 coded images. All 24 grayscale images are exactly achromatic ($R = G = B$). Two further images elsewhere in the corpus are also achromatic, giving 26 in total.

5.2. Baseline methods

All baselines are run on exactly the same images, on the same machine. They fall into five groups:

- **Lossless:** PNG optimised with Pillow [36], raw DEFLATE [19], JPEG-LS via CharLS [16, 37], QOI [17], lossless WebP [10] and lossless JPEG XL (libjxl, distance 0) [11, 38].
- **Near-lossless:** JPEG-LS with error tolerance NEAR = 1 and NEAR = 2 [16], the closest low-complexity com-

Table 4. Dataset composition. Sets A, B, C, D1 and D2 are disjoint.

Set	Description	n	Colour/gray	Source format	Resolution range	Used in
A. Heterogeneous	Photographs, graphics and screen content	17	16/1	10 JPEG / 7 PNG	220 × 148 to 4000 × 4000	All aggregate results
B. Low chroma variance	Sky, studio backgrounds, sparse graphics	24	23/1	17 JPEG / 7 PNG	200 × 193 to 3000 × 4000	Aggregate; Section 6.6
C. High chroma variance	Saturated, finely textured photographs	24	24/0	23 JPEG / 1 PNG	225 × 179 to 512 × 512	Aggregate; Section 6.6
D1. Colour versions	Colour originals of the paired set	24	24/0	14 JPEG / 10 PNG	512 × 402 to 1536 × 2048	Subset; Section 6.5
D2. Grayscale versions	The same 24 scenes in grayscale	24	0/24	11 JPEG / 13 PNG	512 × 402 to 1536 × 2048	Subset; Section 6.5
Total (own corpus)	Disjoint union of A, B, C, D1 and D2	113	87/26	75 JPEG / 38 PNG	200 × 193 to 4000 × 4000	—
Kodak [34]	Public benchmark, lossless source	24	24/0	PNG	768 × 512	Sections 6.1–6.4

petitor with bounded error.

- **Lossy transform codecs:** JPEG via libjpeg-turbo with 4:2:0 and 4:4:4 sampling [8, 39], lossy WebP [10], JPEG 2000 via OpenJPEG [9, 40] and lossy JPEG XL [11, 38].
- **Prior RLE methods:** the methods of [5] and [7], reimplemented from their published descriptions. The method of [6] is not reproduced as it is applicable only for grayscale images, but we intend to include colour images in our experimental domain.
- **Learned reference:** ELIC [15] with pre-trained weights from CompressAI [41], reported with its parameter count and hardware, but no run or implementation.

5.3. Metrics and evaluation protocol

Compression is reported primarily as retained size, the compressed size as a percentage of the raw size (lower is better), and as bits per pixel (bpp):

$$\mathrm{RS} = 100 \cdot S_c / S_{raw}, \qquad \mathrm{bpp} = 8 \cdot S_c / (W \cdot H) \tag{10}$$

where S_c is the compressed file size and S_{raw} the raw size in bytes. The percentage reduction, $100 - \mathrm{RS}$, is given only in parentheses. Fidelity is reported as PSNR computed over all three RGB channels,

$$\mathrm{PSNR} = 10 \log_{10}(255^2/\mathrm{MSE}), \tag{11}$$

together with per-channel PSNR on Y', Cb and Cr; SSIM [42] computed on luminance; the CIEDE2000 colour difference ΔE_{∞} [44], computed per pixel in CIELAB (D65) and reported as the image mean and 99th percentile, which directly measures the chrominance error introduced by subsampling; and the visual information fidelity, VIF [45], which measures how much of the image information survives coding. Following [22] and [43], PSNR is not used as a proxy for perceptual losslessness.

Because the proposed pipeline has a single operating point, lossy codecs are compared in two ways. At iso-rate, the quality parameter of each codec is found by binary search as the setting that produces the largest file not exceeding the size of our output for the same image, and fidelity is then compared. At iso-quality, the setting is the one that produces the smallest file whose PSNR (and, separately, mean ΔE_{∞}) is at least that of our output, and bpp is then compared. Aggregate statistics are reported as mean ± standard deviation, median and range, with 95% confidence intervals. Bit-exactness of the entropy-coding cascade is checked by comparing the decoded subsampled YCbCr planes with the encoder's input planes on every image.

5.4. Hardware and software

All measurements were made single-threaded on Linux (Ubuntu 24.04.4 LTS) Kernel 6.18.44 (glib 2.39), Intel Xeon @2.10 GHz using Python 3.12.3, NumPy 2.4.4, Pillow 12.1.1, scikit-image 0.26.0, pandas 3.0.2. Baseline codecs were used in the following versions: libjpeg-turbo 3.2.0 (API 8.0), CharLS 2.4.4 (JPEG-LS), OpenJPEG 2.5.4 (JPEG 2000), libwebp 1.6.0, libjxl 0.12.0 (JPEG XL), QOI commit 4461cc3, libpng 1.6.58, libdeflate 1.25, zlib 1.3, zopfli 1.0.3. Runtime is reported as throughput in megapixels per second (MP/s), averaged over the whole set.

6. Results

6.1. Compression performance

Table 5 and Figure 6 report retained size by set. Across the full corpus of 113 images, the compressed output averages 23.27% of the raw size (SD 8.71, median 23.89), corresponding to 5.58 bpp and a mean reduction of 76.73%. Per-image values range from 4.06% to 38.78%. On Kodak, the mean is 26.85% (6.44 bpp), higher than the corpus mean because Kodak is uniformly photographic, whereas the corpus also contains grayscale and low-chroma-variance content. The spread between sets is large and systematic: the low-chroma-variance set averages 16.44% and the high-chroma-variance set 33.73%, a gap of more than a factor of two that Section 6.6 traces to the residual statistics.

PSNR is averaged over the 87 images with finite PSNR; the 26 achromatic images reconstruct bit-exactly and are excluded from that mean (Section 6.2).

Figure 7 shows the stage-wise data volume. Chroma subsampling alone reduces data volume to exactly 50% of the original (the luma plane is untouched and each chroma plane falls to one quarter, so $(1 + \frac{1}{4} + \frac{1}{4})/3 = \frac{1}{2}$). The DPCM → RLE → Huffman cascade then drives the final sizes to 4.1–38.8% of the original, a mean of 23.27%. Neither half could achieve the combined result alone: subsampling cannot touch luminance redundancy, and the entropy

Table 5. Compression and fidelity by set (mean ± SD; retained size as % of raw 24-bit size, lower is better).

Set	n	Retained %	Range %	bpp	PSNR (dB)	SSIM	Mean ΔE_{∞}	VIF
A. Heterogeneous	17	22.21 ± 6.65	11.52–32.72	5.33	48.63	1.0000	0.38	0.923
B. Low chroma var.	24	16.44 ± 5.92	6.26–27.17	3.95	52.19	0.9999	0.32	0.929
C. High chroma var.	24	33.73 ± 2.95	25.85–38.78	8.10	39.67	0.9999	1.29	0.833
D1. Colour versions	24	24.66 ± 8.03	6.56–34.21	5.92	40.83	0.9999	0.96	0.832
D2. Grayscale versions	24	18.98 ± 6.77	4.06–26.18	4.56	∞ (bit-exact)	1.0000	0.00	1.000
Full corpus	113	23.27 ± 8.71	4.06–38.78	5.58	44.95	0.9999	0.61	0.902
Kodak	24	26.85 ± 3.27	21.99–33.03	6.44	43.89	0.9999	0.93	0.886

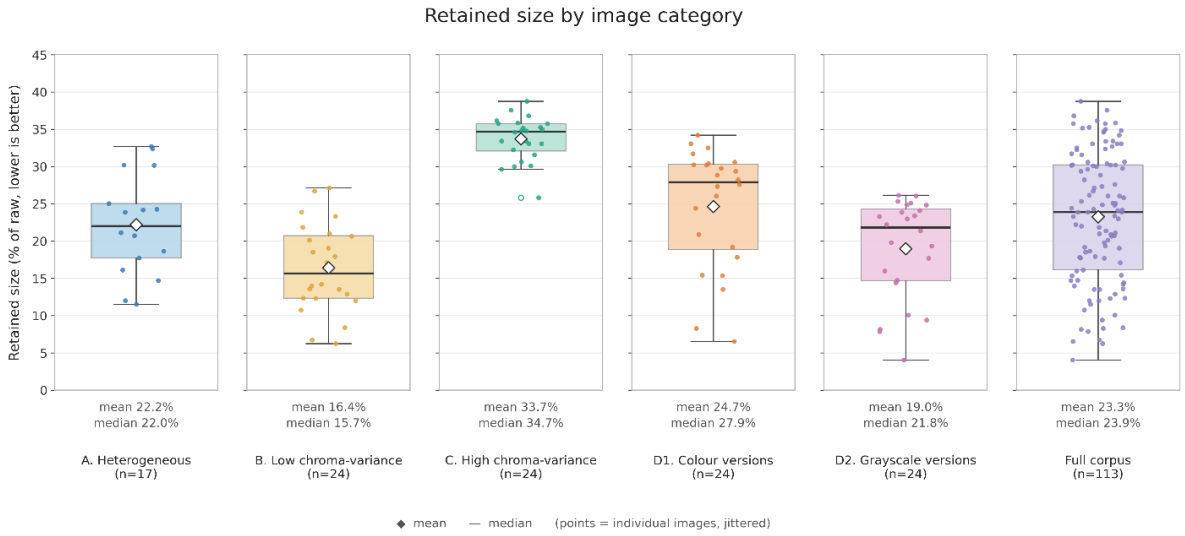


Figure 6. Retained size by set, as a percentage of the raw 24-bit size (lower is better).

cascade applied to raw RGB faces the content dependence that limits plain RLE (configuration A1 in Section 6.3 quantifies this). For the texture-rich images of set C, where the cascade leaves 33.7% against 50% that subsampling alone achieves; noisy or texture-rich images, the entropy cascade adds relatively little beyond subsampling, because their DPCM residuals contain few exploitable runs, although the conditional-RLE rule of Section 3.4 guarantees that the RLE stage itself never expands the stream.

6.2. Reconstruction fidelity and perceptual quality

Fidelity decomposes cleanly along the architecture's fault line. The DPCM–conditional RLE–Huffman backbone is mathematically lossless, and the bit-exactness check of Section 5.3 confirms it: decoding recovers the subsampled YCbCr planes bit-exactly on all 113 images and on all Kodak images. Residual error therefore originates in the designed-in lossy step, 4:2:0 subsampling, plus the rounding of the 8-bit colour transform (Section 4.2).

Twenty-six of the 113 corpus images are exactly achromatic, including the whole grayscale set, and every one of them is reconstructed **bit-exactly** (PSNR $= \infty$, $\Delta E_{\infty} = 0$), confirming the analysis of Section 3.1: for $R = G = B$ the chrominance planes are constant, so 4:2:0 subsampling discards nothing and the round trip is exact. Over the remaining 87 images the mean PSNR is 44.95 dB (median 43.21; range 25.61–87.05 dB), the mean SSIM is 0.9999, and the mean ΔE_{∞} is 0.78 with a mean 99th percentile of 3.63 (Table 5, Figure 8); 69 of the 87 exceed 40 dB. Per-channel PSNR shows the luminance plane preserved at 69.4 dB, essentially to within transform rounding, while the error is concentrated in Cb (48.7 dB) and Cr (50.2 dB), as the design intends. Restricting the comparison to chromatic images, JPEG-originated images reach 46.08 dB against 41.98 dB for PNG-originated ones, so sources that were already chroma-subsampled do not depress our figures; the difference tracks set composition, since the chromatic PNG group is dominated by the saturated images of Set C. On Kodak, the pipeline attains a mean SSIM of 0.9999 and a VIF of 0.886.

Figure 9 shows zoomed crops of the four images with the two largest and one smallest mean ΔE_{∞}, and of one representative image, alongside the JPEG output at the same file size. The characteristic 4:2:0 artefact is visible in the worst cases as colour bleeding across sharp edges between saturated colours and around small coloured text, where the averaged chroma sample spans two differently coloured regions. In smooth and natural regions, no difference is visible even at 300% magnification. We therefore do not claim that the output is visually lossless in general; the pipeline is intended for applications that tolerate this documented chrominance error.

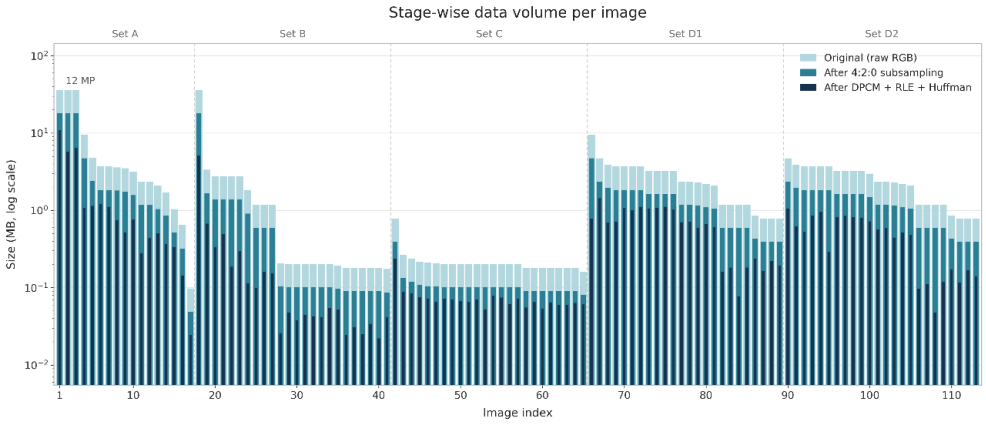


Figure 7. Stage-wise data volume per image (full corpus, $n = 113$): raw RGB, after 4:2:0 chroma subsampling, and after the DPCM + RLE + Huffman cascade.

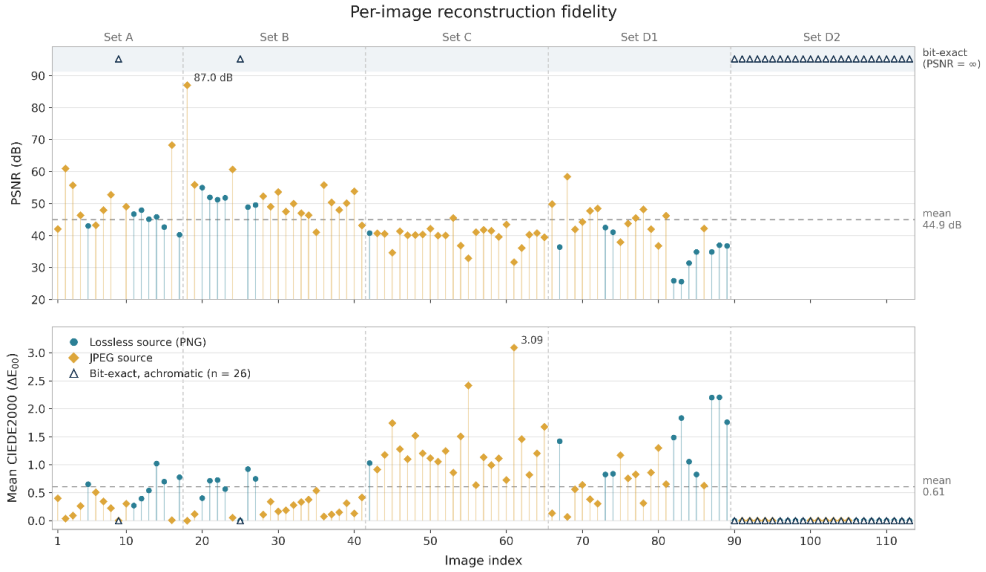


Figure 8. Per-image reconstruction fidelity: PSNR and mean CIEDE2000 colour difference, with lossless-source and JPEG-source images marked.

6.3. Ablation study

Table 6 isolates the contribution of each stage. Each configuration removes or replaces one stage while all other stages remain unchanged; results are given for the full corpus. Configurations A4–A7 alter only lossless stages, so their fidelity equals that of A0 by construction; this equality was verified experimentally. Configuration A1 (RGB, no subsampling) is lossless, and A2 (YCbCr 4:4:4) isolates the error due to colour-transform rounding alone.

Huffman coding is the single largest contributor among the lossless stages (+24.60 pt without it), then DPCM (+8.74 pt), then RLE (+0.63 pt). The conditional rule is worth +2.06 pt against forcing RLE (A6), and it never hurts: A6 is larger on 83 of 113 images and equal on the rest, worst case +4.90 pt. Meanwhile, A5 shows RLE itself is nearly free in aggregate (+0.63 pt) but matters per image — it helps on 67 images and hurts on 46, by up to −1.84 pt, which is exactly why the rule exists.

A2 at 69.00 dB isolates colour-transform rounding cleanly, confirming that 4:2:0 causes essentially all the visible loss (A0 sits at 44.95 dB). A3 offers a real middle point: +4.62 pt of size for +2.6 dB and ΔE_{∞} 0.431 versus 0.605.

A1 is lossless at 56.23% retained, achieved with nothing but DPCM, RLE and Huffman.

6.4. Comparison with existing methods

Table 7 compares the pipeline with lossless and near-lossless low-moderate-complexity and moderate-high-complexity codecs and with the re-implemented RLE methods, and Table 8 compares it with lossy codecs at iso-rate and iso-quality on the public Kodak dataset [34]. Figure 10 shows the Kodak results in the rate–quality plane.

All rows were measured in one session on the hardware of Section 5.4, single-threaded, as the best of three runs, and losslessness was verified image by image. † marks pure-Python reference implementations, whose compressed sizes are comparable but whose throughput is not, as the Python takes longer runtime; the remaining rows use C or C++ libraries. The character-replacement method is reported with the guard that stores a channel verbatim when

Table 6. Ablation study on the full corpus ($n = 113$).

ID	Configuration	Retained %	Δ vs A0	bpp	PSNR (dB)	Mean ΔE_{∞}	Enc. (MP/s)	RLE-sel. streams
A0	Full pipeline (YCbCr 4:2:0 + DPCM + cond. RLE + Huffman)	23.27	–	5.58	44.95	0.605	1.49	176/339
A1	RGB instead of YCbCr, no subsampling	56.23	**+32.97**	13.50	∞ (lossless)	0	0.71	86/339
A2	YCbCr 4:4:4 (no subsampling)	33.47	**+10.20**	8.03	69.00	0.064	0.91	203/339
A3	YCbCr 4:2:2 instead of 4:2:0	27.89	**+4.62**	6.69	47.55	0.431	1.25	175/339
A4	No DPCM (RLE + Huffman on samples)	32.00	**+8.74**	7.68	=A0	=A0	2.33	188/339
A5	No RLE (DPCM + Huffman)	23.90	**+0.63**	5.74	=A0	=A0	2.09	0/339
A6	Unconditional RLE	25.32	**+2.06**	6.08	=A0	=A0	1.28	339/339
A7	No Huffman (DPCM + cond. RLE, fixed-length symbols)	47.87	**+24.60**	11.49	=A0	=A0	2.55	176/339

Table 7. Comparison with lossless, near-lossless and RLE-based methods on identical images on the Kodak dataset (retained % of raw size, lower is better).

Method	Type	Retained %	bpp	PSNR (dB)	Enc./dec. (MP/s)
This work	Lossy (4:2:0)	**26.85**	6.44	43.89	1.33 / 1.20†
JPEG-LS, NEAR=2	Near-lossless	28.76	6.90	45.31	12.09 / 20.36
JPEG XL (lossless, effort 7)	Lossless	35.79	8.59	∞	0.63 / 3.10
JPEG-LS, NEAR=1	Near-lossless	36.19	8.68	49.99	11.41 / 18.73
WebP (lossless, method 6)	Lossless	39.75	9.54	∞	1.03 / 42.55
JPEG-LS	Lossless	54.48	13.08	∞	14.21 / 18.88
PNG (Pillow, optimize)	Lossless	55.84	13.40	∞	1.86 / 32.98
QOI	Lossless	58.29	13.99	∞	92.06 / 124.96
Raw DEFLATE	Lossless	62.47	14.99	∞	7.44 / 52.90
RLE + optimisation [5] (re-impl.)	Lossless	96.55	23.17	∞	1.42 / 13.90†
RLE + character replacement [7] (re-impl.)	Lossless	100.02	24.00	∞	1.46 / 2.37†

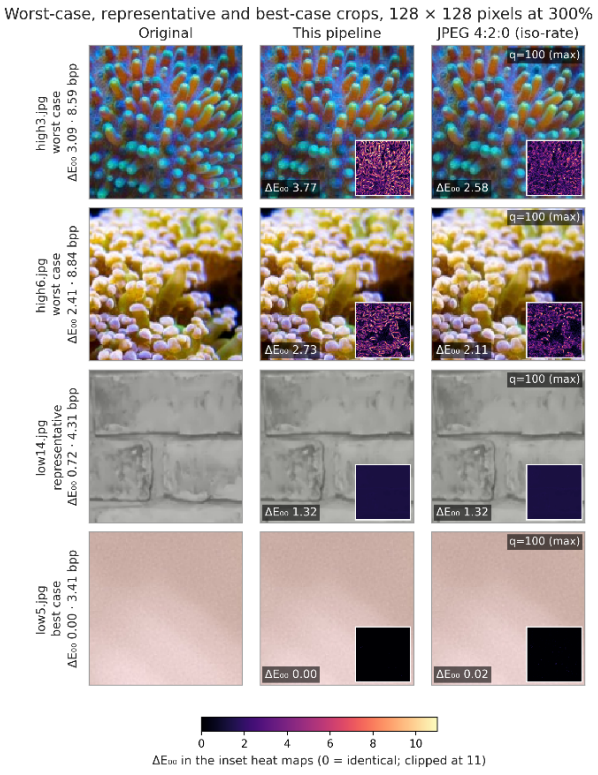


Figure 9. Qualitative comparison on worst-case, best-case and representative images: original, proposed pipeline and JPEG at the same file size.

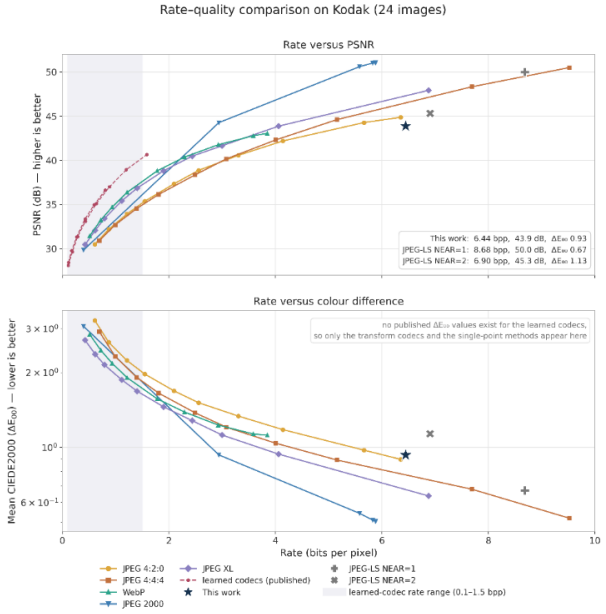


Figure 10. Rate–quality comparison on Kodak. Curves show lossy codecs over their quality range; points show single-operating-point methods.

its token stream is larger; without the guard it expands Kodak images to 240% of raw size. PSNR is ∞ where the decoded image is bit-identical to the original, verified per image.

All rows were measured in one session, matching every codec to this pipeline image by image. "Not matched" counts images for which no setting of the codec could match the corresponding target. Published results for ELIC span roughly 0.1–1.5 bpp on Kodak [15], so it cannot spend a 6.44 bpp budget and cannot reach 43.9 dB with released weights; it is therefore reported in Table **??** as a complexity reference rather than matched here.

The two tables tell a consistent story. Against low-complexity lossless codecs (Table 7) the pipeline produces the smallest files: 26.85% retained against 35.79% for loss-

Table 8. Comparison with lossy codecs on Kodak (mean over 24 images) at iso-rate (same file size as this work) and at iso-quality (same PSNR as this work).

Method	Iso-rate PSNR (dB)	Iso-rate SSIM	Iso-rate mean ΔE_{00}	Iso-rate bpp	Iso-quality bpp (PSNR)	Iso-quality bpp (ΔE_{00})	Not matched (rate/PSNR/ΔE)
This work	43.89	0.9999	0.931	6.444	6.444 (ref.)	6.444 (ref.)	—/—/—
JPEG, 4:2:0	44.63	0.9994	0.922	6.097	5.659	6.350	0/1/4
JPEG, 4:4:4	46.13	0.9961	0.805	5.881	5.129	5.337	0/0/0
WebP (lossy)	43.04	0.9940	1.118	3.849	3.516	not reached	0/20/24
JPEG 2000	51.01	0.9985	0.506	5.853	2.995	3.107	0/0/0
JPEG XL (lossy, effort 7)	45.64	0.9968	0.806	5.144	4.613	4.714	0/0/0
ELIC [15] (learned; GPU)	outside operating range	—	—	—	not reached	not reached	24/24/24

Table 9. Colour versus grayscale compression (representative subset from Set D1 and D2). Retained = compressed/raw × 100 (lower is better); reduction = 100− retained.

Image	Size	Raw RGB (B)	Colour (B)	Colour ret.%	Colour red.%	Gray (B)	Gray ret.% (3-ch base)	Gray red.% (3-ch base)	Gray ret.% (1-ch base)	Gray red.% (1-ch base)
image1	512 × 512	786,432	165,862	21.1	78.9	114,992	14.6	85.4	43.9	56.1
image2	512 × 512	786,432	222,045	28.2	71.8	168,583	21.4	78.6	64.3	35.7
image3	748 × 1024	2,297,856	601,859	26.2	73.8	444,687	19.4	80.6	58.1	41.9
image4	512 × 512	786,432	190,461	24.2	75.8	139,347	17.7	82.3	53.2	46.8
image5	1024 × 712	2,187,264	669,104	30.6	69.4	510,748	23.4	76.6	70.1	29.9
image6	1024 × 768	2,359,296	705,290	29.9	70.1	563,758	23.9	76.1	71.7	28.3
image7	1024 × 683	2,098,176	606,768	28.9	71.1	482,727	23.0	77.0	69.0	31.0
image8	1024 × 768	2,359,296	714,719	30.3	69.7	585,535	24.8	75.2	74.5	25.5
Mean	—	1,707,648	484,514	**27.4**	**72.6**	376,297	**21.0**	**79.0**	**63.1**	**36.9**

less JPEG XL, 39.75% for lossless WebP and 54.48% for JPEG-LS, and it also edges past near-lossless JPEG-LS at NEAR = 2 (28.76%). That margin is the perceptual budget: lossless codecs must preserve chrominance exactly, while this pipeline spends resolution where vision is least sensitive. Against the re-implemented RLE methods the gap is far larger — 96.55% and 100.02% retained — because photographic content contains almost no exact runs, which is precisely the failure mode that motivated the cascade.

Against lossy transform codecs (Table 8) the pipeline does not lead. At iso-rate, JPEG 2000 reaches 51.0 dB and JPEG XL 45.6 dB against our 43.9 dB, and both attain lower colour error; only lossy WebP falls behind, and it could not reach our PSNR at any setting on 20 of the 24 images. At iso-quality the gap is clearest: JPEG 2000 needs 3.00 bpp and JPEG XL 4.61 bpp to match a reconstruction that costs us 6.44 bpp, so a mature transform codec reaches the same fidelity in roughly half the bits. We report this plainly because it defines what the contribution is and is not. The pipeline is not competitive in rate–distortion terms with codecs built on wavelets, context modelling and adaptive entropy coding; its claim is to reach the region below every low-complexity lossless codec while using only integer arithmetic, table look-ups, no dictionary and no trained model (Section 6.8), and while keeping distortion confined to one identifiable, perceptually motivated stage.

6.5. Colour versus grayscale compression

Table 9 isolates the effect of colour by compressing the same representative eight sample images of Set D1 and D2: once in full colour, and once as a single grayscale plane. The grayscale images obtain 66.7% reduction simply by storing one channel instead of three, so Table 9 reports the grayscale result against its own single-plane baseline, and the column labels state explicitly whether they give retained size or reduction.

Against the three-channel baseline, the grayscale version appears to compress more on every image (mean reduction 79.0% versus 72.6% for colour), but this comparison is not like for like. Against its own single-plane baseline, a grayscale plane is reduced only to 63.1% of its raw size on average (43.9–74.5%). The colour images, whose data are halved by subsampling, are reduced by the cascade to 27.4% of the original image on average. Colour images compress well largely because their subsampled chroma planes are smooth and highly compressible; this is the mechanism that Sections 3.2–3.3 anticipate.

6.6. Why performance varies: compressed size versus image complexity

The spread in per-image retained size (4.1–38.8%) is not noise; it reflects the pipeline's mechanics. The controlling quantity is DPCM residual variance. Low local pixel variance yields small, tightly clustered residuals, long RLE runs, a heavily skewed Huffman symbol distribution and strong compression. High colour variance, fine texture, sensor noise and complex gradients yield large, near-uniform residuals that starve RLE of repeats and leave Huffman coding little skew to exploit. Every stage of this causal chain is inspectable, and it is quantitatively confirmed: across the full corpus ($n = 113$), per-image retained size correlates with luminance entropy at $r = 0.641$ (95% CI 0.563–0.712; Spearman $\rho = 0.647$) and, more strongly, with the mean absolute DPCM residual at $r = 0.809$ (95% CI 0.756–0.855; $\rho = 0.843$) (Figure 11). That the residual is the better predictor is itself evidence for the mechanism: what governs the final size is not how much information the image carries but how well the predictor removes it. The controlled sets sharpen the picture: chroma variance predicts retained size at $r = 0.342$ ($\rho = 0.508$) across the corpus, with Set C retaining 33.73% ± 2.95 against 16.44% ± 5.92 for Set B, and conditional RLE selecting the raw path on 71 of the 72 channel streams of Set C against 21 of 72 for Set B, as the mechanism predicts. The colour and grayscale comparison of Section 6.5 points the same way.

6.7. PNG versus JPEG source formats

As a check, we also compared retained size grouped by the images' original container format: PNG ($n = 38$) and JPEG ($n = 75$), which together account for all 113 corpus images (Figure 12). PNG-sourced images retained a mean 18.93% ± 8.12 (median 19.23%), against 25.46% ± 8.21 (me-

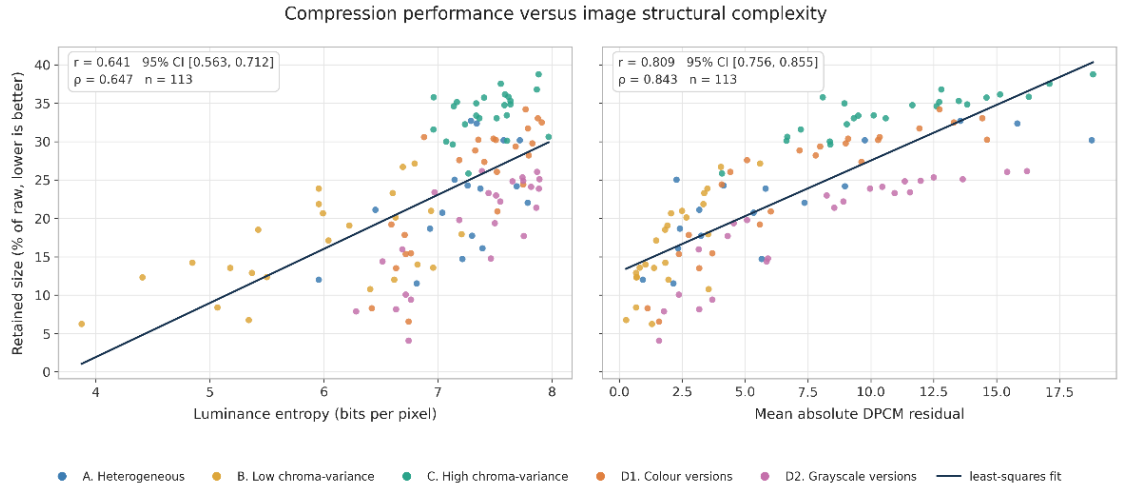


Figure 11. Compression performance as a function of image structural complexity: retained size versus per-image luminance entropy (left) and mean absolute DPCM residual (right), full corpus ($n = 113$).

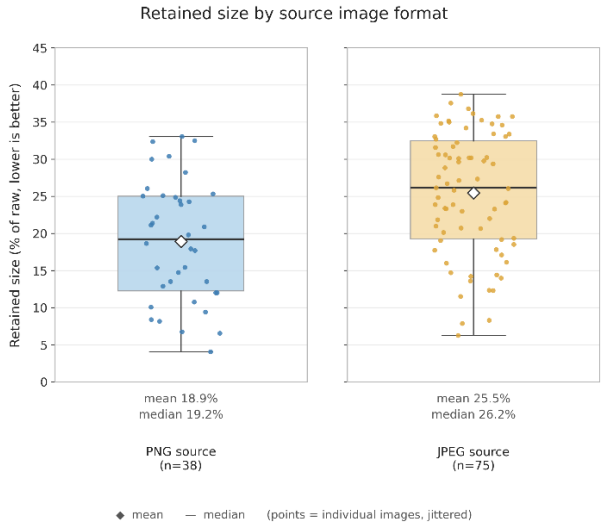


Figure 12. Retained size by source image format (full corpus, $n = 113$; lower is better). The overlap indicates that compressibility is driven by image content rather than container format.

dian 26.18%) for JPEG-sourced images. We do not read this as a format effect on compression: the pipeline operates on decoded RGB pixels and has no visibility into the original container, so it cannot respond to whether a source image began as PNG or JPEG. The gap instead reflects dataset composition: the high-chroma-variance set, which compresses worst, is 23 of 24 JPEG, while the PNG group is weighted towards graphics and grayscale content, which compress best. Source format could in principle affect fidelity, because JPEG-source images may already carry 4:2:0 chroma from their original encoding, but the measured difference runs the other way here (46.08 dB for chromatic JPEG sources against 41.98 dB for chromatic PNG sources, Section 6.2), again reflecting set composition rather than the container.

6.8. Runtime and memory

Table 10 reports mean per-stage throughput on the hardware of Section 5.4. Two patterns stand out, and both align with the design. First, the perceptual front end is inexpensive: YCbCr conversion and 4:2:0 subsampling together account for 10.3% of the total time, so the perceptual redundancy reduction of Sections 3.1–3.2 is obtained at low computational cost. Second, 89.7% of the budget goes to the lossless backbone of DPCM, RLE and Huffman coding, with Huffman coding the largest single item at both ends (36.9% of compression and 77.2% of reconstruction) and chroma upsampling the cheapest at 0.3%. Throughput of the implementation is 1.49 MP/s encoding and 1.56 MP/s decoding on the corpus, and 1.33 MP/s and 1.20 MP/s on Kodak, on the hardware of Section 5.4. The absolute values reflect the Python reference implementation, whose per-sample loops are much slower than an equivalent C implementation, and should be read as relative costs.

Table 11 compares throughput and coding state with the baseline codecs on the same machine on the Kodak dataset. Measured peak resident memory tells a different story from the coding state, and the difference is instructive: coding one 0.39 MP Kodak image raises the process footprint by 55.6 MB for our reference implementation, against 2.6 MB for libjpeg-turbo, 3.5 MB for QOI and zlib, 5.0 MB for PNG, and 16.1 MB for OpenJPEG. Our figure is an artefact of the implementation, not of the algorithm: each channel is materialised as a Python list of integers, at roughly 8 bytes of pointer per sample plus object overhead, where the C codecs stream through line or block buffers. The algorithm itself needs the image buffers plus the $\approx$ 3 KB of tables in Table 11, which is what the analytic model below quantifies and what a compiled implementation would exhibit.

† Throughput of the Python reference implementation; the other rows use C or C++ libraries, so the comparison is of implementations rather than of algorithms. Coding-state figures for this work follow from the format of Section 4.3; those for the other codecs are the persistent structures their specifications and libraries require. ELIC was not run here (no GPU, and the pretrained weights were not reachable from the test environment) and is listed for its model and hardware requirements only.

Measured peak memory depends on implementation lan-

Table 10. Per-stage runtime (ms per megapixel, mean over the full corpus, $n = 113$).

Stage	Compression (ms/MP)	Share in compression	Reconstruction (ms/MP)	Share in reconstruction
YCbCr conversion / inverse	51.0	7.6%	35.3	5.5%
Chroma subsampling / upsampling	18.1	2.7%	1.9	0.3%
DPCM / inverse DPCM	179.9	26.8%	90.4	14.1%
RLE / expansion	173.8	25.9%	18.6	2.9%
Huffman encoding / decoding	247.7	36.9%	494.9	77.2%
Total (throughput)	670.5 (1.49 MP/s)	100%	641.1 (1.56 MP/s)	100%

Table 11. Complexity comparison on the 24 Kodak images. Throughput is the best of three single-threaded runs; lossy codecs run at the per-image iso-rate settings of Table 8, so their retained sizes match that comparison. The coding-state column counts the persistent state a codec needs beyond the image buffers themselves, which is the quantity that decides whether a method fits a constrained target.

Method	Retained %	Enc. (MP/s)	Dec. (MP/s)	Coding state, excl. image buffers	Implementation
This work	26.85	1.33†	1.20†	≈ **3** KB: 256-entry histogram, 256 code lengths and canonical decode tables. No dictionary, no trained model.	Python (NumPy + list stages)
PNG (Pillow, optimize)	55.84	1.86	32.98	32 KB LZ77 window; ≈ **256** KB encoder and ≈ **40** KB decoder state for zlib at these settings	C (libpng, zlib)
Raw DEFLATE	62.47	7.44	52.90	As above, without PNG filters or chunk structure	C (zlib)
JPEG-LS (CharLS)	54.48	14.21	18.88	≈ **2** KB of context counters (365 contexts) plus one line of context	C++ (CharLS 2.4.4)
QOI	58.29	92.06	124.96	256 B: a 64-entry pixel index plus one running pixel	C (reference)
JPEG (libjpeg-turbo)	25.40	26.52	72.73	≈ **2** KB of quantisation and Huffman tables plus MCU row buffers	C with SIMD (3.2.0)
JPEG 2000 (OpenJPEG)	24.39	5.17	6.36	MQ-coder and code-block state, plus wavelet coefficient buffers that scale with the tile	C (2.5.4)
ELIC [15] (learned)	not run	not run	not run	Trained weights: millions of parameters, tens of megabytes	Python/PyTorch, GPU

guage and buffering, so we complement it with an analytic model of the working memory that the algorithm itself requires. Beyond the image buffers, the codec state consists of a 256-entry histogram (1 KB with 32-bit counters), a 256-entry code-length table and the canonical encoding and decoding tables, about 3 KB in total; no dictionary or trained model is needed. The current implementation buffers whole channels, because the conditional-RLE decision and the Huffman statistics are computed per channel, so its working memory grows linearly with image size: about $3WH$ bytes for the input and $1.5WH$ bytes for the YCbCr planes. DPCM itself needs only one sample of context and 4:2:0 subsampling needs two rows, so a stripe-based variant that codes s rows at a time with per-stripe codebooks would need approximately

$$M_{stripe} \approx 4.5 \cdot W \cdot s \text{ bytes} + 3 \text{ KB}, \tag{12}$$

that is, about 273 KB for 16-row stripes of a 3840-pixel-wide frame and about 48 KB for a 640-pixel-wide frame, at the cost of more frequent codebook overhead.

7. Discussion

7.1. Discussion

Four properties distinguish this pipeline, and each is traceable to a deliberate design decision. First, it is the composition, not any single component, that delivers the result: perceptual, spatial and statistical redundancy are each removed by the stage matched to it (Table 2), and the ablation in Section 6.3 shows that removing any one stage increases the retained size (Table 6). Second, loss is architecturally quarantined: the entropy backbone is lossless by construction and bit-exact on every tested image, so all error in the output can be attributed to one identified, perceptually motivated operation and to transform rounding, an auditability property that matters wherever compressed imagery becomes evidence or scientific record. Third, colour is handled as perception suggests rather than as storage convention dictates: YCbCr separation lets the pipeline spend fidelity where the eye is most sensitive, something channel-independent RGB methods structurally cannot do, as Section 6.5 shows directly. Fourth, the modularity is functional, not cosmetic: the extensions of Section 8 are component swaps, not redesigns.

These properties also define the pipeline's place relative to existing work. It does not match the rate–distortion efficiency of modern transform or learned codecs (Section 6.4); its contribution is to extend the class of methods that need no transforms, dictionaries or trained models so that they can exploit chrominance redundancy, with a distortion that is bounded, localised in chrominance and predictable from image content. It is best suited to devices on which a standard transform codec is too costly and to applications that tolerate a documented chrominance error, such as transmission of monitoring imagery from drones and fixed IoT cameras.

7.2. Limitations

We state the trade-offs explicitly:

- **Content dependence.** Fine texture, noise and complex gradients shorten residual runs and weaken compression (Section 6.6). The dependence is characterised and predictable, but not eliminated; it is intrinsic to redundancy-driven coding.
- **Chrominance is not exactly preserved.** 4:2:0 subsampling [3] is inappropriate where exact chrominance matters, such as medical imaging, scientific visualisation and archival storage, and it produces visible

colour bleeding at sharp saturated edges (Figure 9). Subsampling can be disabled, running the backbone in strictly lossless mode at the cost of ratio (configuration A1/A2, Table 6).

- **Single operating point.** Unlike transform codecs, the pipeline has no quality parameter for rate control; a rate-controlled variant would require adding quantisation of the DPCM residuals.
- **Rate–distortion efficiency.** At the same file size, JPEG 2000 and JPEG XL achieve 7.1 dB and 1.75 dB higher PSNR, and they reach our fidelity using 3.00 and 4.61 bpp against our 6.44 (Table 8); the pipeline's advantage lies in simplicity and memory, not in rate–distortion performance.
- **Whole-channel buffering.** The current implementation buffers entire channels.
- **No embedded-hardware measurement.** Runtime and memory were measured on a desktop CPU with a Python reference implementation; no microcontroller or GPU or pretrained model measurement were yet available. The embedded suitability of the method is supported by its operation count and memory model, not by an on-device deployment.
- **Fixed pre-processing cost.** The multistage design adds a constant time cost that can offset gains for very small images or ultra-low-latency loops.

8. Conclusion and Future Work

The central claim of this paper is architectural: lightweight image compression benefits less from finding a better single coding technique than from composing classical stages so that each removes the redundancy class it is best suited to, and from making that matching explicit. The pipeline presented here puts this claim into practice. YCbCr conversion and 4:2:0 subsampling spend the perceptual budget where vision is least sensitive; DPCM translates natural image statistics into the run-rich, skewed form that entropy coders require; conditional RLE and Huffman coding then exploit that structure with about 3 KB of coding tables and no dictionary. A matching reconstruction path inverts every coding stage exactly, confining all loss to a single perceptually motivated operation and to transform rounding, and the pipeline is lossless for single-plane grayscale content.

On a documented corpus of 113 images and on the Kodak benchmark, the pipeline reduces colour images to 23.3% and 26.9% of their raw size on average (5.58 and 6.44 bpp) at a mean PSNR of 45.0 dB and 43.9 dB and a mean CIEDE2000 of 0.61 and 0.93, and it reconstructs achromatic images bit-exactly. The ablation study shows that every stage contributes, with Huffman encoding contributing most in the lossless part, and the conditional RLE rule prevents expansion on run-poor content. Compared on identical images, the pipeline produces smaller files than every low-complexity lossless codec tested and than near-lossless JPEG-LS, while modern transform codecs reach the same fidelity in roughly half the bits at substantially higher complexity. Its performance is explained, not merely observed: per-image results track the mean absolute DPCM residual ($r = 0.81$) and luminance entropy ($r = 0.64$) through an inspectable causal chain.

The modular architecture turns future work into component swaps: arithmetic or range coding in place of Huffman coding [33] where the measured entropy gap of Section 6.3 justifies it; transform-based or context-model prediction, such as the median edge detector of LOCO-I [16], in place of one-tap DPCM for stronger decorrelation; adaptive subsampling that spends chrominance resolution where local content demands it; quantisation of DPCM residuals to provide rate control. Because the three channels are coded independently, the cascade is also amenable to parallel hardware; GPU or FPGA implementation [46] is left to future work and is not claimed here.

Benchmarking Data Availability

The Kodak dataset is publicly available from their original sources [34].

References

[1] K. L. Ketshabetswe, A. M. Zungeru, B. Mtengi, C. K. Lebekwe, and S. Prabaharan, "Data compression algorithms for wireless sensor networks: A review and comparison," *IEEE Access*, vol. 9, pp. 136872–136891, 2021.

[2] X. Kong, Y. Wu, H. Wang, and F. Xia, "Edge computing for internet of everything: A survey," *IEEE Internet Things J.*, vol. 9, no. 23, pp. 23472–23485, 2022.

[3] P. Pokorny, "Lossy compression in the chroma subsampling process," *WSEAS Trans. Comput.*, vol. 15, pp. 76–82, 2016.

[4] ITU-R, "Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios," Rec. ITU-R BT.601-7, Int. Telecommun. Union, Geneva, Switzerland, Mar. 2011.

[5] A. Birajdar, H. Agarwal, M. Bolia, and V. Gupte, "Image compression using run length encoding and its optimisation," in *Proc. Global Conf. Advancement Technol. (GCAT)*, Bangalore, India, 2019, pp. 1–6, doi: 10.1109/GCAT47503.2019.8978464.

[6] A. R. Idrais, I. Aljarrah, and O. AlKhaleel, "A spatial image compression algorithm based on run length encoding," *Bull. Electr. Eng. Inform.*, vol. 10, no. 5, pp. 2607–2616, 2021, doi: 10.11591/eei.v10i5.2563.

[7] D. Chakraborty and S. Banerjee, "Efficient lossless colour image compression using run length encoding and special character replacement," *Int. J. Comput. Sci. Eng.*, vol. 3, no. 7, pp. 2719–2725, 2011.

[8] G. K. Wallace, "The JPEG still picture compression standard," *IEEE Trans. Consum. Electron.*, vol. 38, no. 1, pp. xviii–xxxiv, 1992.

[9] A. Skodras, C. Christopoulos, and T. Ebrahimi, "The JPEG 2000 still image compression standard," *IEEE Signal Process. Mag.*, vol. 18, no. 5, pp. 36–58, 2001.

[10] Google, "WebP: An image format for the Web," and "libwebp" codec library. [Online]. Available: https://developers.google.com/speed/webp

[11] J. Alakuijala *et al.*, "JPEG XL next-generation image compression architecture and coding tools," in *Proc. SPIE 11137, Applications of Digital Image Processing XLII*, 2019, Art. no. 111370K.

[12] T. Polonelli, D. Battistini, M. Rusci, D. Brunelli, and L. Benini, "An energy optimized JPEG encoder for parallel ultra-low-power processing-platforms," in *Applications in Electronics Pervading Industry, Environment and Society (ApplePies 2019)*. Cham, Switzerland: Springer, 2020, pp. 125–133.

[13] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, "Variational image compression with a scale hyperprior," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2018.

[14] Z. Cheng, H. Sun, M. Takeuchi, and J. Katto, "Learned image compression with discretized Gaussian mixture likelihoods and attention modules," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2020, pp. 7939–7948.

[15] D. He, Z. Yang, W. Peng, R. Ma, H. Qin, and Y. Wang, "ELIC: Efficient learned image compression with unevenly grouped space-channel contextual adaptive coding," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2022, pp. 5718–5727.

[16] M. J. Weinberger, G. Seroussi, and G. Sapiro, "The LOCO-I lossless image compression algorithm: Principles and standardization into JPEG-LS," *IEEE Trans. Image Process.*, vol. 9, no. 8, pp. 1309–1324, 2000.

[17] D. Szablewski, "QOI: The Quite OK Image Format," 2021. [Online]. Available: https://qoiformat.org/

[18] C. D. Rask and D. E. Lucani, "RAGE for the machine: Image compression with low-cost random access for embedded applications," arXiv:2402.05974, 2024.

[19] P. Deutsch, "DEFLATE compressed data format specification version 1.3," RFC 1951, May 1996.

[20] T. Boutell, "PNG (Portable Network Graphics) specification version 1.0," RFC 2083, Mar. 1997.

[21] B. Elmeligy, T. Richter, R. R. R. Rao, S. Fößel, and A. Raake, "Evaluating visually lossless compression of JPEG XS, JPEG 2000, HEVC and AV1 in selected medical imaging modalities," in *Proc. 16th Int. Conf. Quality Multimedia Experience (QoMEX)*, 2024, pp. 221–227.

[22] Q. Huynh-Thu and M. Ghanbari, "Scope of validity of PSNR in image/video quality assessment," *Electron. Lett.*, vol. 44, no. 13, pp. 800–801, 2008, doi: 10.1049/el:20080522.

[23] N. Azami, R. Lawson, and M. Burtscher, "LICO: An effective, high-speed, lossless compressor for images," in *Proc. Data Compression Conf. (DCC)*, 2024, pp. 462–471.

[24] F. J. Merkl, "Binary image compression using run length encoding and multiple scanning techniques," M.S. thesis, Dept. Comput. Sci., Rochester Inst. Technol., Rochester, NY, USA, 1988.

[25] Y.-C. Hu and C.-C. Chang, "A new lossless compression scheme based on Huffman coding scheme for image compression," *Signal Process. Image Commun.*, vol. 16, no. 4, pp. 367–372, 2000, doi: 10.1016/S0923-5965(99)00064-8.

[26] X. Liu, P. An, Y. Chen, and X. Huang, "An improved lossless image compression algorithm based on Huffman coding," *Multimed. Tools Appl.*, vol. 81, no. 4, pp. 4781–4795, 2022, doi: 10.1007/s11042-021-11017-5.

[27] A. Nasif, Z. A. Othman, N. S. Sani, M. K. Hasan, and Y. Abudaqqa, "Huffman deep compression of edge node data for reducing IoT network traffic," *IEEE Access*, vol. 12, pp. 122988–122997, 2024, doi: 10.1109/ACCESS.2024.3452669.

[28] Y. Bai, X. Liu, K. Wang, X. Ji, X. Wu, and W. Gao, "Deep lossy plus residual coding for lossless and near-lossless image compression," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2024.

[29] R. R. S. Tomar and K. Jain, "Lossless image compression using differential pulse code modulation and its application," in *Proc. Int. Conf. Comput. Intell. Commun. Netw. (CICN)*, 2015, pp. 397–400, doi: 10.1109/CICN.2015.84.

[30] S. Mun and J. E. Fowler, "DPCM for quantized block-based compressed sensing of images," in *Proc. 20th Eur. Signal Process. Conf. (EUSIPCO)*, 2012, pp. 1424–1428.

[31] T. A. Welch, "A technique for high-performance data compression," *Computer*, vol. 17, no. 6, pp. 8–19, 1984, doi: 10.1109/MC.1984.1659158.

[32] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proc. IRE*, vol. 40, no. 9, pp. 1098–1101, 1952, doi: 10.1109/JRPROC.1952.273898.

[33] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic coding for data compression," *Commun. ACM*, vol. 30, no. 6, pp. 520–540, 1987, doi: 10.1145/214762.214771.

[34] Eastman Kodak Company, "Kodak Lossless True Color Image Suite." [Online]. Available: https://r0k.us/graphics/kodak/

[35] C. Poynton, *Digital Video and HD: Algorithms and Interfaces*, 2nd ed. Burlington, MA, USA: Morgan Kaufmann, 2012.

[36] H. van Kemenade, A. Murray, wiredfool, A. Clark, *et al.*, "Pillow: The friendly PIL fork (Python Imaging Library)," version 12.1.1, Zenodo, 2026. [Online]. Available: https://doi.org/10.5281/zenodo.596518

[37] CharLS: A C++ JPEG-LS library implementation. [Online]. Available: https://github.com/team-charls/charls

[38] libjxl: JPEG XL reference implementation. [Online]. Available: https://github.com/libjxl/libjxl

[39] libjpeg-turbo. [Online]. Available: https://libjpeg-turbo.org/

[40] OpenJPEG: An open-source JPEG 2000 codec. [Online]. Available: https://www.openjpeg.org/

[41] J. Bégaint, F. Racapé, S. Feltman, and A. Pushparaja, "CompressAI: A PyTorch library and evaluation platform for end-to-end compression research," arXiv:2011.03029, 2020.

[42] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, 2004, doi: 10.1109/TIP.2003.819861.

[43] Z. Wang and A. C. Bovik, "Mean squared error: Love it or leave it? A new look at signal fidelity measures," *IEEE Signal Process. Mag.*, vol. 26, no. 1, pp. 98–117, Jan. 2009, doi: 10.1109/MSP.2008.930649.

[44] G. Sharma, W. Wu, and E. N. Dalal, "The CIEDE2000 color-difference formula: Implementation notes, supplementary test data, and mathematical observations," *Color Res. Appl.*, vol. 30, no. 1, pp. 21–30, 2005.

[45] H. R. Sheikh and A. C. Bovik, "Image information and visual quality," *IEEE Trans. Image Process.*, vol. 15, no. 2, pp. 430–444, 2006.

[46] M. Hameed, A. Khmag, F. Zaman, and A. R. Ramli, "A new lossless method of Huffman coding for text data compression and decompression process with FPGA implementation," *J. Eng. Appl. Sci.*, vol. 11, no. 3, pp. 402–407, 2016.