BLADE: Real-Time Lossless Compression of Bayer CFA Images on GPU Platforms

Joan Bartrina-Rapesta, Sebastià Mijares, Joan Serra-Sagristà *Senior Member, IEEE*, Natàlina Blasco Andreo, Xavier Fernández Mellado and Pau Quintas-Torra

***Abstract*—Modern imaging systems generate large volumes of raw data under strict throughput and memory constraints. In Bayer Color Filter Array (CFA) sensors, conventional pipelines typically compress demosaiced RGB images, introducing redundancy and increasing computational cost. This motivates compression directly in the sensor domain. We propose BLADE, a GPU-oriented lossless compression framework that operates directly on CFA data. The method combines CFA-aware prediction with a block-and-stripe decomposition that enables fine-grained parallelism while preserving statistical efficiency. A lightweight global entropy model is employed to avoid synchronization overhead and ensure scalable parallel execution. Experimental results on high-resolution and high-bit-depth datasets show that BLADE achieves competitive or superior compression performance compared to state-of-the-art CFA methods, while significantly reducing execution time with respect to both conventional and GPU-based codecs. The method operates close to the Pareto-optimal frontier, achieving low bitrate and high throughput. System-level analysis under continuous acquisition demonstrates sustained real-time operation within a well-defined operating region. The method exhibits predictable behavior across resolutions, input rates, and power constraints, including on embedded GPU platforms. These results indicate that BLADE effectively bridges compression efficiency and real-time processing requirements, enabling high-throughput edge imaging systems.**

***Index Terms*—Lossless image compression, Bayer CFA, RAW imaging, predictive coding, high-throughput processing.**

I. Introduction

Modern high-resolution imaging systems generate massive volumes of data under strict throughput, memory, and energy constraints. In many applications, including remote sensing and airborne platforms, data must be compressed in real time to enable continuous acquisition, storage, and transmission.

Most imaging pipelines rely on single-sensor architectures equipped with Bayer Color Filter Arrays (CFA), where each pixel captures a single spectral component. Full-color images are typically reconstructed through demosaicing prior to compression, increasing data volume and computational cost. From a data representation perspective, CFA data contains one sample per pixel, whereas a demosaiced RGB image represents each pixel with up to three components. For an image with N pixels, this corresponds to N samples in the CFA domain and up to $3N$ samples after demosaicing [1], [2]. This expansion increases memory bandwidth, storage, and computational cost, motivating compression directly in the CFA domain.

Demosaicing can instead be deferred to the decompression stage, preserving true lossless compression with respect to sensor measurements. In contrast, demosaicing prior to compression alters the signal through interpolation, so the compressed data no longer corresponds to the original CFA samples.

Most lossless methods transform the signal into prediction residuals, which are entropy coded. The coding rate is therefore governed by the entropy of these residuals, making accurate prediction essential.

In practical systems, compression must satisfy real-time constraints. Let R_{in} denote the input rate (in MP/s) and R_{proc} the processing throughput. Sustained operation requires

$$R_{\text{proc}} \geq R_{\text{in}}, \tag{1}$$

which favors parallel-friendly designs with minimal dependencies.

Lossless compression of CFA images has been widely studied using CFA-aware predictive and transform-based methods [3]–[9]. General-purpose standards such as JPEG-LS and JPEG 2000 have also been applied to CFA data [10], [11], but do not exploit CFA-specific dependencies and often rely on sequential or weakly parallel processing. High-throughput approaches such as JPEG XS [12], [13] prioritize parallelism but operate in the lossy domain.

Despite extensive research, existing approaches fail to jointly achieve high compression efficiency, fine-grained parallelism, and sustained real-time performance on modern GPU platforms. This limitation arises from a fundamental tension between statistical dependency and parallel execution, which has not been explicitly addressed in prior CFA compression frameworks.

To address this challenge, we propose BLADE (Bayer Lossless Compression for High-Throughput Airborne Data on Edge GPUs), a GPU-native compression framework that operates directly in the CFA domain. The proposed method combines CFA-aware prediction with a block-and-stripe decomposition that enables fine-grained parallelism while preserving most of the statistical efficiency.

This trade-off can be formalized as:

$$\min_{\theta} \ \mathbb{E}[L(X;\theta)] \quad \text{s.t.} \quad R_{\text{proc}}(\theta) \geq R_{\text{in}}, \tag{2}$$

Joan Bartrina-Rapesta, Sebastià Mijares, Joan Serra-Sagristà, Natàlina Blasco Andreo, Xavier Fernández Mellado and Pau Quintas-Torra are with Universitat Autònoma de Barcelona, Spain.

This work was supported in part by the Spanish Ministry of Science and Innovation (MICINN) and by the European Regional Development Fund (FEDER), funded by MCIN/AEI/10.13039/501100011033, UE, under grant PID2024-156292OB-I00; also by the Catalan Government under grant SGR2021-00643; and also by the "Data Compression and Machine Learning for Earth Observation Satellites" project under the Institute for Space Studies of Catalonia (IEEC) and the Catalan Government in the framework of the New Space Strategy of Catalonia 2024.

TABLE I: Comparison of representative image compression methods in terms of publication year, CFA-aware modeling, lossless capability, sensor-level (RAW) losslessness, parallelism, and execution platform.

Method	Year	CFA-aware modeling	Lossless	Sensor-level (RAW)	Parallelism	Processing
JPEG-LS [10]	2000	✗	✓	✗	■	CPU
JPEG 2000 [11]	2002	✗	✓	✗	■■	CPU/GPU
JPEG-XL [14]	2019	✗	✓	✗	■■	CPU
JPEG XS [13]	2019	✗	✗	✗	■■■	CPU/GPU/FPGA
CFA (JPEG XS-based) [15]	2021	✓	✗	✗	■■■	CPU/GPU
CFA methods [3]–[5], [9], [16], [17]	2006–2017	✓	✓	✓	■	CPU
Refined CFA methods [6]–[8]	2014–2022	✓	✓	✓	■	CPU
Learning-based [18]–[22]	2018–2022	✗	✗	✗	■	GPU
Lossless neural [23], [24]	2022–2023	✗	✓	✗	■	GPU
GPU compression [25], [26]	2021–2022	✗	✓	✗	■■■■	GPU
GPU codecs [27], [28]	2020–2023	✗	✓	✗	■■■	GPU
Scientific GPU compression [29], [30]	2020–2023	✗	✓	✗	■■■	GPU
BLADE (proposed)	**2026**	✓	✓	✓	■■■■	**GPU-native**

where θ denotes the set of design parameters controlling prediction, decomposition, and coding.

The main contributions of this work are as follows:

- We introduce BLADE, a GPU-native lossless compression framework for CFA data that operates directly in the sensor domain, preserving exact RAW measurements while enabling high-throughput processing.
- We propose a block-and-stripe decomposition that resolves the tension between parallelism and statistical efficiency, enabling fine-grained parallel execution without introducing cross-region dependencies.
- We design a CFA-aware predictive and entropy coding scheme tailored for massively parallel architectures, achieving effective redundancy reduction while avoiding synchronization overhead.
- We provide a system-level analysis under continuous acquisition, establishing a clear operating model based on the relation between input rate and processing throughput.
- We demonstrate that the proposed method operates close to the Pareto-optimal frontier of rate and throughput, achieving competitive or superior compression performance while significantly reducing execution time.

Overall, BLADE establishes a practical framework for bridging the gap between compression efficiency and real-time processing in modern edge imaging systems.

The remainder of this paper is organized as follows. Section III describes the proposed method. Section V presents experimental results and system-level evaluation. Finally, Section VI concludes the paper. The implementation of the proposed method is publicly available in [31].

II. Related Work

Lossless image compression has been extensively studied through predictive and transform-based approaches. JPEG-LS [10] relies on low-complexity context modeling, while JPEG 2000 [11] and JPEG-XL [14] exploit spatial decorrelation and advanced entropy coding.

JPEG XS [13] targets ultra-low latency and high-throughput compression through simplified coding tools and parallel-friendly designs, enabling real-time processing at 4K and 8K resolutions. Extensions to CFA data have also been proposed [15]. However, these approaches are visually lossless and do not preserve exact sensor measurements.

Most existing standards assume fully sampled RGB or grayscale signals and do not explicitly model the structural dependencies of CFA data. Their pipelines often include sequential operations that limit scalability on massively parallel architectures.

CFA-specific lossless methods [3]–[5], [16], [17] exploit the Bayer structure through prediction and context modeling, with refinements based on hierarchical prediction [6]–[8]. These methods achieve *sensor-level lossless* compression by operating directly on raw CFA data. However, their adaptive and context-based models introduce inter-sample dependencies that limit parallel scalability.

Learning-based compression methods [18]–[22] achieve strong rate-distortion performance in RGB data, with extensions to lossless compression [23], [24]. However, their computational complexity, memory requirements, and sequential inference limit their applicability to sustained real-time sensor-domain compression under the target constraints.

GPU-based techniques enable high-throughput processing through massive parallelism. Libraries such as nvCOMP [25] and GDeflate [26] operate on generic data without exploiting image structure. In contrast, GPU-accelerated codecs such as nvJPEG2000 [27] implement full coding pipelines, inheriting sequential dependencies that limit scalability. GPU-based scientific compression methods [29], [30] further highlight the importance of parallel-friendly designs.

Overall, existing approaches prioritize either compression efficiency (CFA-specific methods) or computational performance (GPU-based methods), but rarely both. Furthermore, most lossless methods do not operate at the sensor level, as they rely on demosaiced representations.

Table I summarizes this landscape. Only a limited number of methods achieve sensor-level lossless compression, and these

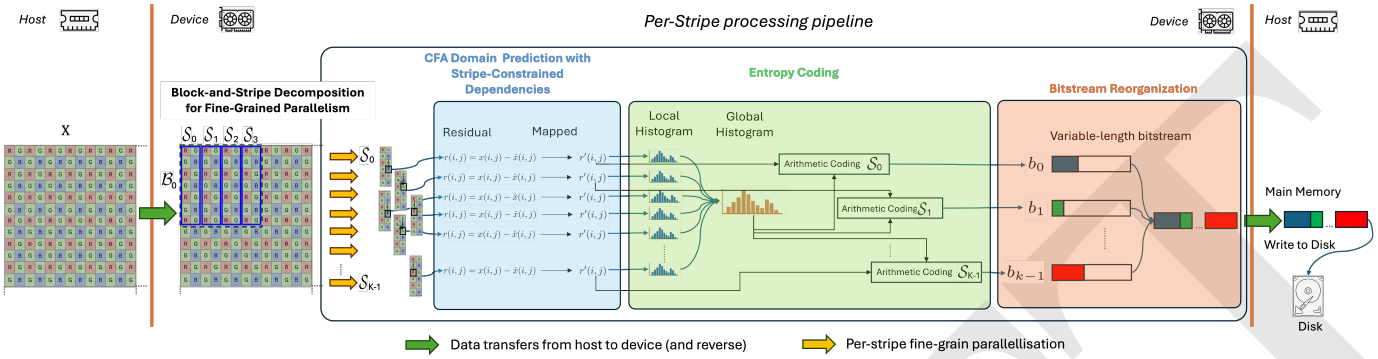


Fig. 1: Overview of the proposed BLADE compression pipeline. The CFA image is partitioned into blocks and stripes, enabling independent prediction, entropy modeling, and encoding. Each stripe produces an independent bitstream, which is later compacted into the final compressed representation. The example illustrates a configuration with block size 8×8 and stripe width $S = 2$.

are not designed for massively parallel execution. Conversely, GPU-based approaches provide high throughput but do not preserve sensor-level fidelity.

This gap motivates compression frameworks that jointly achieve sensor-level lossless coding and high-throughput parallel execution, as addressed by the proposed BLADE method.

III. Proposed Method

Fig. 1 illustrates the overall pipeline. The method operates directly in the sensor domain, avoiding demosaicing and preserving exact RAW measurements.

The proposed framework consists of four main stages: (i) block-and-stripe decomposition, (ii) CFA-domain prediction, (iii) entropy coding, and (iv) bitstream reorganization.

Let $X \in \{0, \dots, 2^b - 1\}^{H \times W}$ denote the input CFA image with bit depth b, where $x_{i,j}$ represents the sample at position (i, j). For clarity, a summary of the main notation is provided in Appendix A.

The image is partitioned into non-overlapping blocks $\{\mathcal{B}_u\}$, which are further subdivided into vertical stripes $\{\mathcal{S}_k\}$. Each stripe is processed independently and produces a partial bitstream, later concatenated into the final compressed output. The details of each stage are described in the following subsections.

A. Block-and-Stripe Decomposition

To enable fine-grained parallelism under real-time constraints, the image is decomposed into independent processing units that can be handled concurrently without introducing data dependencies.

The image X is partitioned into a set of non-overlapping square blocks $\{\mathcal{B}_u\}$ of size $B \times B$, which form a tiling of the image domain and define the primary units of parallel processing. The total number of blocks is

$$N_B = \left\lceil \frac{H}{B} \right\rceil \left\lceil \frac{W}{B} \right\rceil . \tag{3}$$

Each block $\mathcal{B}_u$ is further subdivided into vertical stripes of width S, defining a collection of regions $\{\mathcal{S}_k\}$. Each stripe $\mathcal{S}_k \subset \mathcal{B}_u$ constitutes an independent coding unit, and the regions are mutually disjoint:

$$\mathcal{S}_k \cap \mathcal{S}_m = \emptyset, \quad k \neq m, \tag{4}$$

ensuring that each sample is processed exactly once.

The number of stripes per block is

$$N_S = \left\lceil \frac{B}{S} \right\rceil , \tag{5}$$

and the total number of stripes is therefore

$$K = N_B \cdot N_S. \tag{6}$$

This decomposition exposes fine-grained parallelism, as each stripe can be processed independently without cross-stripe dependencies.

Each stripe produces an independent bitstream b_k, which are concatenated to form the final compressed representation. This design enables scalable high-throughput compression while explicitly controlling the trade-off between parallelism, compression efficiency, and metadata overhead.

B. CFA-Domain Prediction

A CFA spatial sampling pattern used in single-sensor imaging systems captures only one color component per pixel. The most common Bayer pattern follows a 2×2 periodic structure composed of one red (R), one blue (B), and two green samples:

$$\begin{bmatrix} R & G_1 \\ G_2 & B \end{bmatrix} . \tag{7}$$

This pattern defines four interleaved sampling grids corresponding to channels R, G_1, G_2, and B. Each spatial location (i, j) belongs to exactly one channel, denoted by $c(i, j) \in \{R, G_1, G_2, B\}$, and the CFA image X contains a single sample $x_{i,j}$ per pixel.

Prediction is performed independently within each channel using only previously processed samples of the same type. The effective predictor support is defined as

$$\begin{aligned} \mathcal{N}^*(i,j) = \{(p,q) : (p,q) \prec (i,j), \\ (p,q) \in \mathcal{S}_k, \\ c(p,q) = c(i,j)\}, \end{aligned} \tag{8}$$

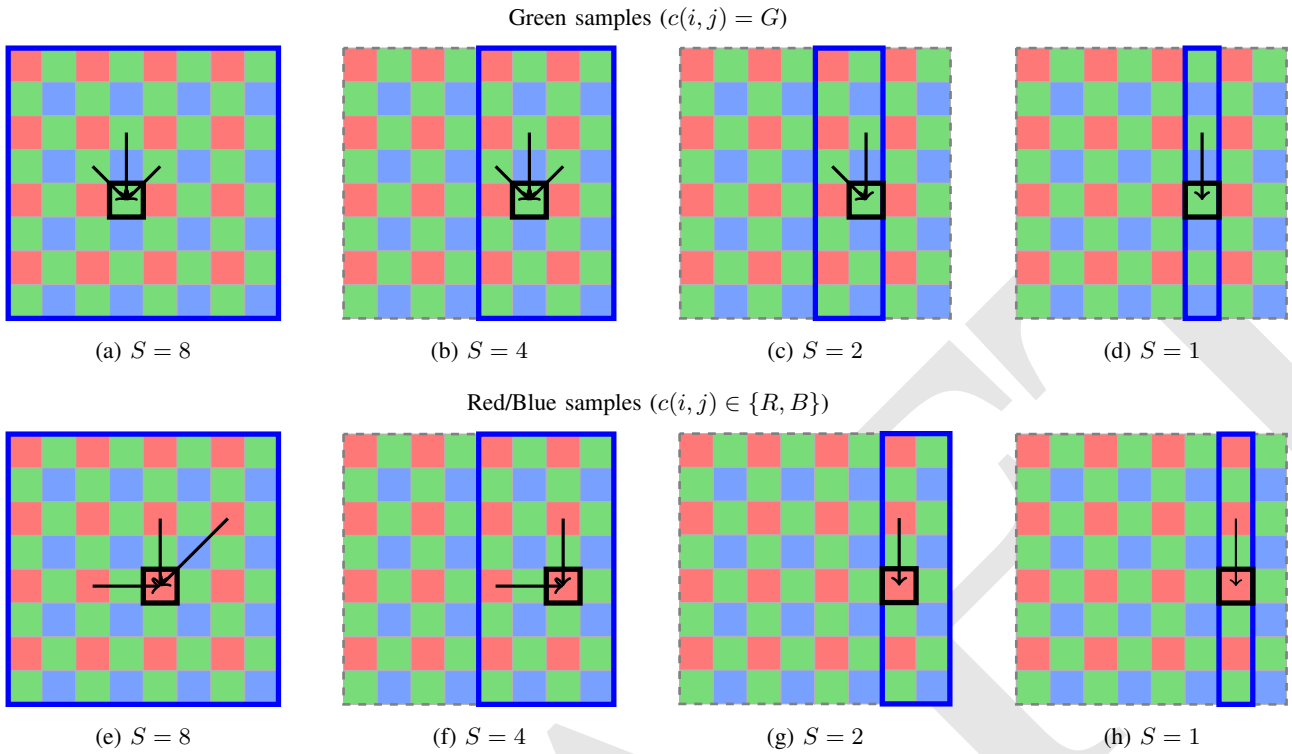


Fig. 2: Prediction support under stripe-constrained dependencies for green (top) and red/blue (bottom) samples. Green samples exhibit a denser neighborhood, while red/blue samples rely on more sparsely distributed same-channel neighbors. As the stripe width S decreases, horizontal dependencies are progressively removed, leading to predominantly vertical support.

where $(p, q) \prec (i, j)$ denotes a causal ordering.
Each sample is predicted from $\mathcal{N}^*(i, j)$, yielding

$$\hat{x}_{i,j}, \quad r_{i,j} = x_{i,j} - \hat{x}_{i,j}. \tag{9}$$

To enable efficient entropy coding, residuals are mapped to non-negative integers using a bijective transformation. Let $x_{\min}$ and $x_{\max}$ denote the representable range. Define

$$\theta_{i,j} = \min\left(\hat{x}_{i,j} - x_{\min},\ x_{\max} - \hat{x}_{i,j}\right), \tag{10}$$

and map residuals as

$$r'_{i,j} = \begin{cases} 2r_{i,j}, & 0 \le r_{i,j} \le \theta_{i,j}, \\ -2r_{i,j} - 1, & -\theta_{i,j} \le r_{i,j} < 0, \\ \theta_{i,j} + |r_{i,j}|, & \text{otherwise.} \end{cases} \tag{11}$$

This mapping is bijective and ensures exact recovery of the original residuals. It follows the range-adaptive principle used in predictive lossless compression [10], [32], [33].
To enable parallel execution, prediction is restricted to stripe-local causal neighborhoods $\mathcal{N}^*(i, j)$, avoiding cross-stripe dependencies and synchronization across processing units.
The predictor adapts to the CFA structure and the available support. The effect of stripe-constrained prediction is illustrated in Fig. 2.

Green samples ($c(i, j) = G$):

$$\hat{x}_{i,j} = \frac{1}{3}\left(x_{i-1,j-1} + x_{i-1,j+1} + x_{i-2,j}\right), \tag{12}$$

with reduced-support variants near boundaries.

Red and Blue samples ($c(i, j) \in \{R, B\}$):

$$\hat{x}_{i,j} = \frac{1}{3}\left(x_{i,j-2} + x_{i-2,j} + x_{i-2,j+2}\right), \tag{13}$$

also adapting to available neighbors when necessary.
This design ensures that all predictor inputs belong to the same channel and lie within the same stripe, enabling fully parallel execution while preserving most of the statistical dependencies of CFA data.

C. Entropy Coding

The mapped residuals $r'_{i,j}$ are encoded using an entropy coding stage designed to balance compression efficiency and parallel scalability.
A global statistical model shared across all stripes is employed. Local histograms are first computed independently for each stripe $\mathcal{S}_k$ in parallel:

$$H_k(v) = \left|\{(i, j) \in \mathcal{S}_k : r'_{i,j} = v\}\right|. \tag{14}$$

These local histograms are aggregated into a global histogram,

$$H(v) = \sum_{k=1}^{K} H_k(v), \tag{15}$$

from which a probability model $P(v)$ is derived.
Let $L(v)$ denote the code length associated with symbol v. Under this model, it is given by

$$L(v) = -\log_2 P(v) + \epsilon, \tag{16}$$

where ϵ accounts for coding overhead.

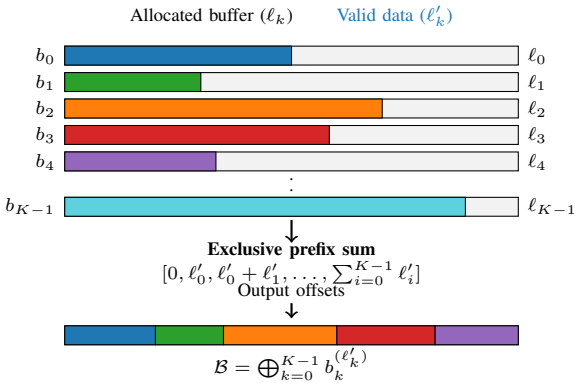


Fig. 3: Bitstream reorganization from independently encoded stripe buffers. Each stripe produces a variable-length bitstream stored in a preallocated buffer, where only a prefix of length ℓ'_k contains valid data. An exclusive prefix sum determines the output offsets, enabling compact assembly of the final bitstream without synchronization.

The global model is then used to encode all stripes independently. This decouples model estimation from encoding, enabling fully parallel processing across stripes without inter-thread synchronization.

Coding is performed at the symbol level, encoding each mapped residual as a single event. This reduces the number of coding operations and avoids fine-grained dependencies associated with bit-level adaptive schemes.

The use of a global model represents a deliberate trade-off: while local or adaptive models could improve compression efficiency, they introduce synchronization and irregular memory access patterns that significantly degrade parallel performance on GPU architectures.

In practice, residual distributions remain sufficiently consistent across stripes, allowing the global model to retain most of the statistical efficiency while enabling scalable high-throughput execution.

D. Bitstream Reorganization

The final stage assembles the global output bitstream by concatenating the independently generated stripe bitstreams, as illustrated in Fig. 3. The final bitstream is denoted by $\mathcal{B}$ to avoid confusion with the block size B.

To enable fully parallel encoding, each stripe $\mathcal{S}_k$ is assigned a preallocated buffer of size ℓ_k, representing an upper bound on its encoded length. During encoding, each stripe produces a bitstream b_k with valid length $\ell'_k \leq \ell_k$, which is written into its corresponding buffer.

The lengths $\{\ell'_k\}$ are recorded and used to compute prefix sums that determine the final memory offsets of each stripe bitstreams. The global output is then obtained as

$$\mathcal{B} = \bigoplus_{k=1}^{K} b_k, \tag{17}$$

where b_k denotes the bitstream generated by stripe $\mathcal{S}_k$, considering only its valid portion of length ℓ'_k.

TABLE II: Summary of datasets used in the experiments. "Type" indicates RGB or CFA data. Synthetic datasets follow standard UHD resolutions (4K–16K). Entropy is reported in bits per sample (bps).

Dataset (Purpose)	Type	Res.	Bit	Size	Entropy
Real Images					
Kodak (*Benchmark*)	CFA (simulated)	–	8	1024×1536	7.31
Kodak (*Benchmark*)	CFA (simulated)	–	8	1536×1024	7.31
Sony A7R IV (*Sensor*)	CFA	–	16	6376×9600	10.96
Synthetic					
Random (*Scalability*)	CFA	8K	8	4320×7680	8.00
Random (*Scalability*)	CFA	8K	12	4320×7680	12.00
Random (*Scalability*)	CFA	8K	16	4320×7680	15.99
Random (*Scalability*)	CFA	12K	8	6480×11520	8.00
Random (*Scalability*)	CFA	12K	12	6480×11520	12.00
Random (*Scalability*)	CFA	12K	16	6480×11520	15.99
Random (*Scalability*)	CFA	16K	8	8640×15360	8.00
Random (*Scalability*)	CFA	16K	12	8640×15360	12.00
Random (*Scalability*)	CFA	16K	16	8640×15360	16.00

This design decouples local bitstream generation from global memory layout, avoiding dynamic allocation and synchronization during encoding. Bitstream generation is fully parallel, followed by a lightweight reorganization step based on prefix sums.

The scheme is well suited for GPU execution, as both prefix-sum operations and memory compaction can be efficiently implemented in parallel, enabling high-throughput bitstream assembly with minimal synchronization.

E. GPU Mapping: Blocks, Stripes, and Warps

The proposed BLADE pipeline is explicitly designed to align with the execution model of modern GPU architectures. The block-and-stripe decomposition exposes multiple levels of parallelism that map naturally to CUDA.

At the coarse level, image blocks $\mathcal{B}_u$ of size $B \times B$ are processed independently and mapped to CUDA thread blocks, enabling parallel execution without inter-block synchronization. Multiple blocks can be scheduled concurrently on each Streaming Multiprocessor (SM), contributing to high occupancy.

Within each block, stripes define the units of fine-grained parallelism. Each stripe $\mathcal{S}_k$ is processed by one or more warps operating on contiguous pixel segments.

Let $W = 32$ denote the warp size. Choosing stripe widths aligned with the warp size ($S \approx W$) establishes a direct mapping between algorithmic and hardware execution units. Under this configuration, each warp processes a contiguous region, maximizing memory coalescing and minimizing control divergence.

The number of warps required to process a stripe is

$$N_{\text{warps}} = \left\lceil \frac{S}{W} \right\rceil, \tag{18}$$

while the number of stripes per block is

$$N_{\text{stripes}} = \left\lceil \frac{B}{S} \right\rceil. \tag{19}$$

The total number of warps per block can be approximated as

$$N_{\text{warps per block}} \approx \left\lceil \frac{B}{S} \right\rceil \left\lceil \frac{S}{W} \right\rceil. \tag{20}$$

This directly impacts SM occupancy. Larger stripe widths increase the number of threads and warps per block, which may limit the number of concurrently resident blocks due to register and shared memory constraints. Conversely, smaller stripe widths reduce per-block resource usage, enabling higher occupancy.

However, narrower stripes reduce the available spatial context for prediction, degrading compression efficiency. As shown in Section V-F, optimal configurations balance these effects, achieving high throughput while preserving sufficient predictor support.

Overall, the proposed mapping establishes a direct correspondence between algorithmic units (blocks and stripes) and hardware execution units (thread blocks and warps), enabling efficient GPU utilization.

This mapping follows directly from the independence structure induced by the proposed decomposition and does not rely on implementation-specific optimizations.

IV. Experimental Setup

A. Datasets

The proposed method is evaluated on three complementary datasets covering: (i) *Benchmark*: coding efficiency under standard benchmark conditions, enabling comparison with state-of-the-art lossless CFA-aware methods; (ii) *Sensor*: performance on real sensor data, reflecting behavior on a high-end commercial imaging sensor; and (iii) *Scalability*: computational scalability under controlled statistical regimes, evaluating throughput under near worst-case conditions using minimally compressible random data. Table II summarizes the main characteristics of the datasets, including their entropy expressed in bits per sample (bps).

Together, these datasets span a wide range of signal characteristics, from highly compressible natural images to near-incompressible data, enabling a joint assessment of compression efficiency, robustness, and scalability.

Fig. 4 illustrates representative samples from these datasets, including both RGB visualizations and CFA-domain details.

1) Kodak Dataset (Simulated CFA): The Kodak dataset consists of 6 natural images originally in RGB format, from which Bayer CFA images are generated using an RGGB sampling pattern. This protocol is widely adopted in prior works [4], [7], [8], enabling fair and reproducible comparison with state-of-the-art CFA compression methods.

Following established practice, we use the same image subset as in [9], ensuring consistency with recent literature and enabling direct comparison.

The resulting 8-bit CFA images exhibit entropy values around 7.31 bits/sample. This dataset is therefore used to evaluate coding efficiency.

2) High Bit-Depth CFA (Real Sensor Data): To assess performance under realistic acquisition conditions, we use a dataset of RAW Bayer images captured with a Sony A7R IV camera (Sony IMX455 sensor) [34]. The images have a resolution of 9600×6376 pixels and 16-bit depth.

This dataset preserves key sensor-domain characteristics such as noise, high dynamic range, and native CFA sampling, providing a challenging and representative scenario for real-world deployment. The measured entropy ranges from approximately 9.01 to 10.96 bps.

This dataset is therefore used to evaluate performance on real sensor data under practical operating conditions.

3) Synthetic CFA (Controlled Worst-Case Scenario): Synthetic data are used to evaluate scalability and upper-bound performance under controlled statistical conditions. We consider multiple resolutions from 8K to 16K and bit depths of 8, 12, and 16 bits.

The synthetic dataset consists of random CFA images, where samples are independently drawn, resulting in entropy values close to the theoretical maximum (8, 12, and 16 bps).

Due to the absence of spatial correlation, this dataset represents a worst-case scenario for lossless compression. It is therefore used to characterize computational throughput and to stress the entropy coding stage, rather than to evaluate coding efficiency.

Together, these datasets provide a comprehensive evaluation framework spanning from highly compressible natural images to near-incompressible signals, enabling a joint assessment of compression efficiency, robustness, and real-time performance.

B. Hardware Configuration

Experiments are conducted on two GPU platforms: a high-end NVIDIA RTX 5000 and an embedded NVIDIA Jetson Orin NX (16GB, DSBOARD-ORNX).

The RTX 5000 ADAis used to evaluate computational performance and scalability, and to enable comparison with GPU-accelerated baselines such as nvJPEG2000 and GDeflate.

The Jetson Orin NX represents an edge-computing scenario, allowing evaluation under constrained power, memory, and computational resources.

These platforms provide complementary insights into both high-performance and resource-constrained deployments.

C. Evaluation Metrics

Compression performance is measured in bits per sample (bps), defined as

$$\mathrm{bps} = \frac{|\mathcal{B}|}{HW}, \tag{21}$$

where $|\mathcal{B}|$ denotes the length (in bits) of the final bitstream, and $H \times W$ is the number of CFA samples.

Computational performance is evaluated in terms of processing time (ms per image) and throughput (megapixels per second, MP/s), providing a representation-independent measure of efficiency.

Real-time operation is assessed by comparing the encoding time T_{encode} with the acquisition period T_{acq}, requiring

$$T_{\mathrm{encode}} \leq T_{\mathrm{acq}}. \tag{22}$$

System behavior under sustained operation is further analyzed through buffer occupancy and service throughput (Section V-G), characterizing stability in streaming scenarios.

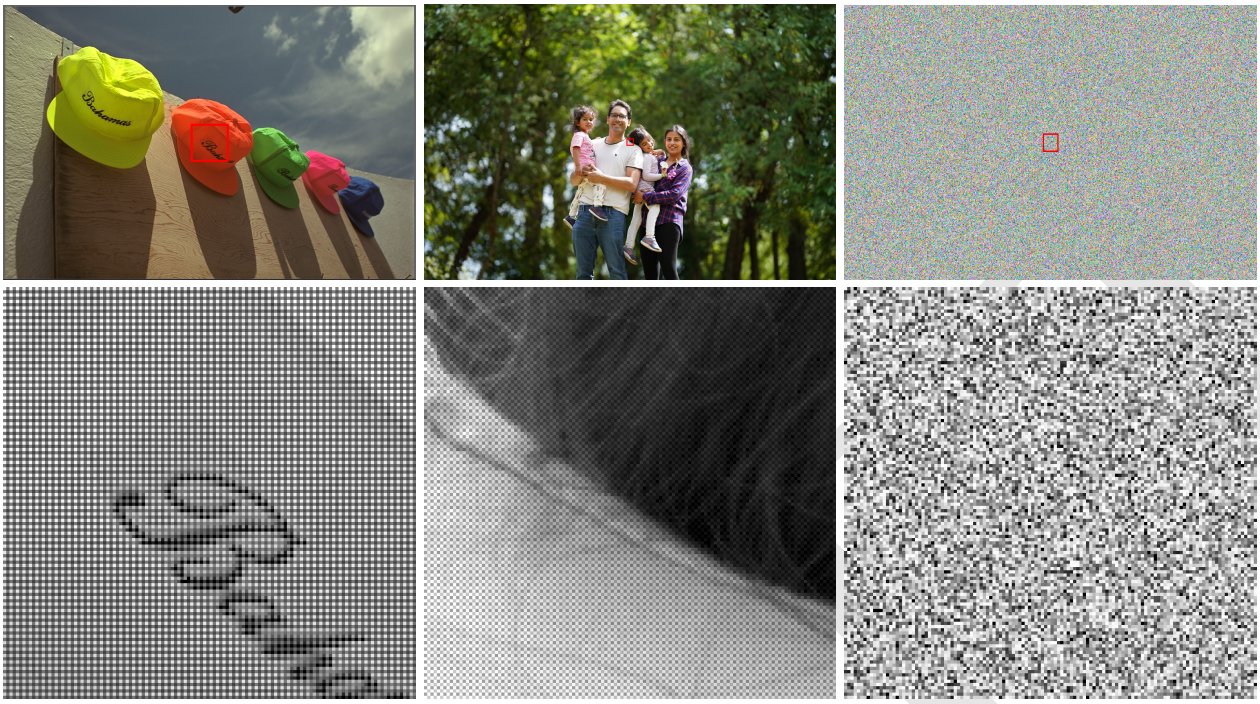


Fig. 4: Representative samples from the evaluated datasets. Top row: RGB visualization for interpretability. Bottom row: corresponding CFA zooms highlighting the Bayer sampling pattern and local statistical structure. Columns correspond to Kodak (left), Sony A7R IV (center), and synthetic data (right). Red boxes in the RGB images indicate the regions extracted for the CFA zooms, illustrating the underlying sensor-domain data processed by the proposed method.

Two complementary comparison strategies are used to evaluate both coding efficiency and computational performance.

- CFA methods: Compression performance (bps) is compared against state-of-the-art CFA lossless methods reported in the literature. Since GPU implementations are not publicly available, this comparison is limited to coding efficiency.
- Available implementations: We also compare against widely used lossless compression libraries, including JPEG-LS [10], JPEG 2000 [11], JPEG XL [14], and GPU-based implementations. In particular, we consider nvCOMP, which includes and several codecs available through the nvCOMP library [35], including nvJPEG2000 [27], zstandard (nv-zstd), gdeflate (nv-gdeflate), deflate (nv-deflate), and bitcomp (nv-bitcomp). Although these methods operate on generic image representations rather than CFA data, they provide a reference for throughput and system-level performance. All methods are evaluated using publicly available implementations to ensure reproducibility.

This dual evaluation reflects the requirements of real-time systems, where both compression efficiency and throughput must be jointly considered.

V. Results and Discussion

This section evaluates the proposed method in terms of compression efficiency (bps) and computational complexity. We analyze the impact of key design parameters, assess compression performance on CFA data, and study the trade-off between coding efficiency and execution time. Additional experiments evaluate scalability through resolution-dependent behavior, as well as the contribution of individual components.

A. Parameter Analysis

Fig. 5 shows the impact of block size and stripe width on throughput. Performance is not monotonic with respect to these parameters.

Small block sizes increase the number of independent work units, improving load balancing and latency hiding, but introduce overhead due to fragmentation and kernel scheduling. Conversely, large blocks reduce the number of concurrent thread blocks per SM, limiting occupancy and reducing throughput.

A similar trade-off is observed for stripe width. Narrow stripes increase parallelism and scheduling flexibility, while wider stripes increase per-thread workload but reduce the number of independent regions.

As a result, optimal performance is achieved at intermediate configurations that balance computational granularity and hardware utilization.

Based on this analysis, the configuration $\mathrm{BLADE}_{512,2}$ is selected as a reference operating point, as it provides a favorable balance between throughput and computational efficiency.

These trends are consistent with the GPU execution model, where performance is governed by occupancy, memory access patterns, and warp-level parallelism, rather than purely empirical tuning.

B. Auxiliary Data Overhead

In addition to compression efficiency and computational performance, the proposed method introduces auxiliary data

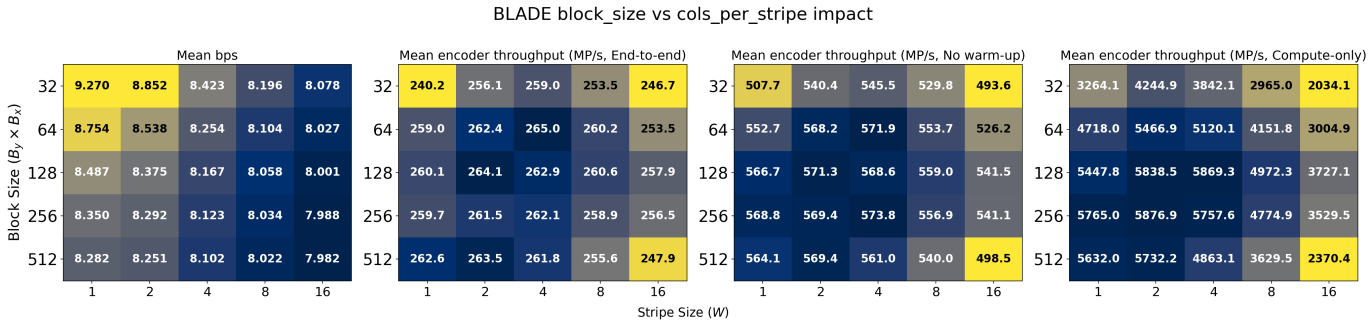


Fig. 5: Throughput (MP/s) as a function of block size and stripe width. Optimal performance is achieved at intermediate configurations due to the balance between parallelism and computational granularity.

TABLE III: Theoretical auxiliary data size (KB) for the Sony A7R IV dataset. Rows: stripe width. Columns: block size.

Stripe \ Block	32	64	128	256	512
1	3750	1875	938	469	244
2	1875	938	469	234	122
4	938	469	234	117	61
8	469	234	117	59	30
16	234	117	59	30	15

associated with the block-and-stripe decomposition. This metadata includes per-stripe information required for decoding, such as cumulative frequencies and structural descriptors.

Let K denote the total number of stripes, as defined in Section III.

Assuming a fixed cost of $C = 2$ bytes per stripe, the total auxiliary data size is given by

$$\text{AuxData} = \frac{K \cdot C}{1024} \quad \text{(KB)}. \tag{23}$$

This expression provides a deterministic estimate of the metadata overhead for any (B, S) configuration.

Table III reports the resulting auxiliary data size for the Sony A7R IV dataset under different block and stripe configurations. In practice, the auxiliary data represents a small fraction of the total compressed size for typical configurations, and does not significantly impact overall compression efficiency.

Table III shows that the auxiliary data decreases monotonically with both block size B and stripe width S, as both parameters reduce the number of independently encoded stripes.

The overhead ranges from approximately 3750 KB for $(B = 32, S = 1)$ to 15 KB for $(B = 512, S = 16)$, corresponding to a reduction of more than two orders of magnitude.

This reduction comes at the cost of decreased local adaptivity, as fewer independent regions are available to capture spatial variations. As a result, a trade-off emerges between metadata overhead and statistical modeling accuracy, which directly impacts compression efficiency.

This behavior is consistent with previous observations: smaller (B, S) configurations increase adaptivity and parallelism but incur higher overhead, whereas larger regions reduce metadata at the expense of modeling flexibility.

In practical configurations, the auxiliary data remains negligible relative to the compressed bitstream size. For instance, for $(B = 256, S = 4)$, the overhead is on the order of a few hundred kilobytes, representing a small fraction of the total compressed data.

Overall, the proposed method provides a low and predictable metadata overhead that can be explicitly controlled through (B, S) without significantly affecting compression performance.

C. Compression Performance

Unless otherwise stated, results correspond to the $\text{BLADE}_{512,2}$ configuration, selected based on the scalability analysis in the previous subsection.

Table IV compares the proposed method against state-of-the-art CFA lossless compression techniques on simulated Bayer images derived from the Kodak dataset.

The proposed $\text{BLADE}_{512,2}$ configuration achieves the best overall performance, reaching an average bitrate of 5.019 bps. It provides the lowest bitrate in four out of six images, indicating consistent performance across diverse spatial statistics.

Compression performance is strongly correlated with the spatial structure of the signal. As shown in Fig. 6, images with high spatial regularity, such as *Wall* and *Landscape*, exhibit lower residual entropy due to improved prediction, resulting in reduced bitrate.

In contrast, highly textured images, such as *Boat* and *Lighthouse*, present broader residual distributions and higher entropy, which limits compression efficiency. In these cases, methods with more globally adaptive models can achieve slightly lower bitrates.

The degradation observed for $\text{Blade}_{512,16}$ (5.579 bps, +11.1%) highlights the dependence of optimal context size on signal statistics.

This behavior is explained by the characteristics of simulated CFA data. Since these images are generated from RGB content, they exhibit strong local correlations, particularly in the green channel. Under these conditions, local predictors are highly effective.

Increasing the stripe width introduces a broader context that may smooth these local dependencies, resulting in suboptimal residual modeling.

These results indicate that the optimal prediction context depends on the underlying statistical properties of the signal. In particular, localized contexts are better suited for highly

TABLE IV: Average lossless compression results in bps for 6 simulated mosaic images. Best results in bold. Second-best results in italic. Percentages indicate relative difference with respect to the best result in each row.

Image	[3]	[4]	[5]	[6]	[9]	$\text{Blade}_{512,16}$	$\text{Blade}_{512,2}$
Boat	5.028 (+4.9%)	*4.881* (+1.8%)	4.984 (+4.0%)	**4.793** (0.0%)	5.170 (+7.9%)	5.393 (+12.5%)	4.947 (+3.2%)
Fence	4.823 (+7.2%)	4.711 (+4.7%)	4.886 (+8.6%)	*4.649* (+3.4%)	4.663 (+3.7%)	5.033 (+11.9%)	**4.498** (0.0%)
Landscape	6.243 (+11.7%)	6.138 (+9.9%)	6.279 (+12.4%)	6.072 (+8.7%)	*6.047* (+8.2%)	6.172 (+10.5%)	**5.588** (0.0%)
Lighthouse	4.867 (+16.4%)	4.803 (+14.9%)	4.864 (+16.3%)	*4.699* (+12.4%)	**4.181** (0.0%)	5.070 (+21.3%)	4.711 (+12.7%)
Wall	5.650 (+8.0%)	5.478 (+4.7%)	5.750 (+9.9%)	*5.438* (+4.0%)	6.044 (+15.6%)	5.840 (+11.7%)	**5.230** (0.0%)
Windows	5.725 (+14.9%)	5.570 (+11.8%)	**4.984** (0.0%)	*5.506* (+10.5%)	6.270 (+25.8%)	5.971 (+19.8%)	5.142 (+3.2%)
Average	5.389 (+7.4%)	5.264 (+4.9%)	5.291 (+5.4%)	*5.193* (+3.5%)	5.396 (+7.5%)	5.579 (+11.1%)	**5.019** (0.0%)

(a) Lighthouse

(b) Wall

(c) Landscape

(d) Boat

Fig. 6: Representative image content illustrating the impact of spatial structure on compression performance. The Lighthouse image (left) exhibits complex textures and high variability, while Wall and Landscape (right) show strong spatial regularity. Boat presents intermediate behavior.

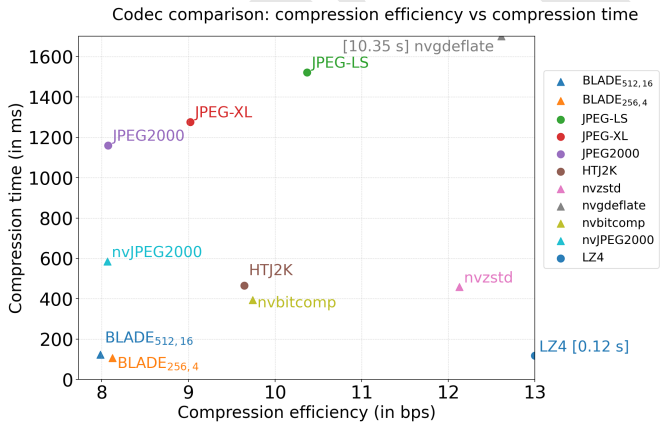


Fig. 7: Rate–time trade-off for lossless compression of CFA images. BLADE operates close to the Pareto frontier, achieving low bitrate and reduced execution time compared to both classical and GPU-based codecs.

correlated data, while larger contexts are expected to be beneficial for natural sensor data with more complex statistics. Overall, the proposed method achieves state-of-the-art compression performance on CFA data while maintaining robustness across different image structures.

This behavior is further analyzed in the following subsection through rate–complexity trade-offs.

D. Rate-Complexity Trade-off

Fig. 7 shows the Pareto frontier between compression efficiency (bps) and execution time (ms) for the evaluated codecs. Experiments are conducted using Sony A7R IV RAW images from the dataset described above. The evaluated methods include BLADE, JPEG-LS, JPEG XL, JPEG 2000, HTJ2K, LZ4, and nvCOMP.

Employing the following configuration parameters. BLADE is evaluated using $\text{BLADE}_{512,16}$ and $\text{BLADE}_{256,4}$. JPEG-LS operates in strict lossless mode. JPEG XL is configured in lossless mode with `effort=7` and automatic threading. JPEG 2000 uses lossless settings, with 5 DWT levels and `ALL_CPUs` threading. HTJ2K is configured with the same settings as JPEG 2000. LZ4 uses `level=1` with maximum threading.

GPU-based methods are evaluated using nvCOMP framework. Within nvCOMP, the evaluated codecs are nvJPEG2000, in lossless mode with 5 DWT levels; zstandard (nvzstd), deflate (nvgdeflate), , and bitcomp (nvbitcomp). In nvzstd, the default backend configuration is used without specifying an explicit compression level. For nvgdeflate and nvdeflate, the compression level can be controlled in the range 0–5; experiments either fix this level for strict reproducibility or use the default nvCOMP configuration to maintain neutrality. For nvbitcomp, the chunk size is set to balance throughput and compression efficiency (typically 8 MiB). This configuration ensures a consistent comparison across techniques while preserving exact lossless reconstruction.

BLADE configurations lie on the Pareto frontier, achieving the best trade-off between compression efficiency and execution time. In particular, $\text{BLADE}_{256,4}$ attains the lowest execution time among all evaluated codecs, while $\text{BLADE}_{512,16}$ provides improved compression efficiency with a moderate increase in runtime.

Conventional codecs such as JPEG-LS, JPEG XL, and JPEG 2000 achieve competitive compression efficiency but at significantly higher computational cost. HTJ2K reduces runtime compared to JPEG 2000, yet remains slower than BLADE.

GPU-based methods exhibit heterogeneous behavior. LZ4 achieves the lowest execution time but at the expense of

substantially reduced compression efficiency. Other nvCOMP codecs, including nvzstd and nvgdeflate, show both higher runtime and lower compression performance compared to BLADE, placing them outside the Pareto frontier.

Overall, the proposed method operates in the lower-left region of the plot, demonstrating a favorable balance between rate and computational complexity. This behavior results from combining accurate CFA-domain prediction, which reduces residual entropy, with a highly parallel design that sustains high throughput.

E. Scalability Analysis

While the previous results focus on rate–complexity trade-offs at a fixed resolution, we next analyze the scalability of the proposed method with respect to input size and signal characteristics.

Scalability is evaluated across multiple image resolutions (8K, 12K, and 16K) and bit depths (8, 12, and 16 bits) using synthetic CFA data with controlled statistical properties.

Fig. 8b shows the compute-only throughput as a function of input resolution. The results indicate that throughput remains in the same overall range across resolutions for each bit depth, with only moderate variation. In particular, throughput is around 2.3–3.1 GP/s for 8-bit data, 2.0–2.6 GP/s for 12-bit data, and 1.1–1.4 GP/s for 16-bit data.

This behavior indicates that the computational cost scales approximately linearly with the number of input pixels, with no observable degradation due to scheduling or synchronization overhead as the problem size increases.

Although some variability is observed across resolutions, the overall throughput remains stable enough to confirm that the proposed block-and-stripe decomposition efficiently exploits parallel resources over a wide range of input sizes.

Across different bit depths, throughput decreases as the symbol range and entropy coding cost increase, which is consistent with the additional computational effort required to process higher-precision data. However, these differences do not alter the overall scalability trend with respect to resolution.

Compression performance remains stable across resolutions, with bitrates close to the theoretical entropy of the synthetic data (approximately 8, 12, and 16 bps), confirming that the statistical behavior of the residuals is preserved across scales.

Overall, these results show that the proposed method maintains predictable performance across a wide range of resolutions and bit depths, supporting its applicability to high-throughput imaging scenarios without requiring parameter retuning.

1) Throughput vs Input Rate: While buffer occupancy provides a temporal view of system behavior, it does not directly characterize the operating limits. To address this, we analyze the relationship between input rate and sustained processing throughput.

Fig. 8a shows the sustained service throughput as a function of the input rate for different configurations. The dashed diagonal represents the ideal condition $R_{\text{proc}} = R_{\text{in}}$.

Points above the diagonal correspond to configurations where processing throughput exceeds the input rate, resulting in stable operation with bounded buffer occupancy. Points near the diagonal indicate near-capacity operation, where the system remains stable but operates close to its limits.

Points below the diagonal indicate insufficient processing throughput, leading to backlog accumulation and eventual buffer saturation, consistent with the behavior observed in Fig. 9.

In practice, the diagonal represents an idealized boundary. Due to the discrete and batched nature of processing, throughput is measured as a sustained average, while the input rate corresponds to a continuous arrival process. As a result, configurations close to or slightly below the diagonal may still exhibit stable behavior, as transient imbalances do not necessarily lead to unbounded backlog growth.

Overall, this representation provides a concise characterization of the operating limits, identifying stable, near-capacity, and overload regimes, and complementing the temporal analysis of buffer occupancy.

F. Ablation Study

This section evaluates the contribution of the main components of the proposed method, namely prediction and entropy modeling.

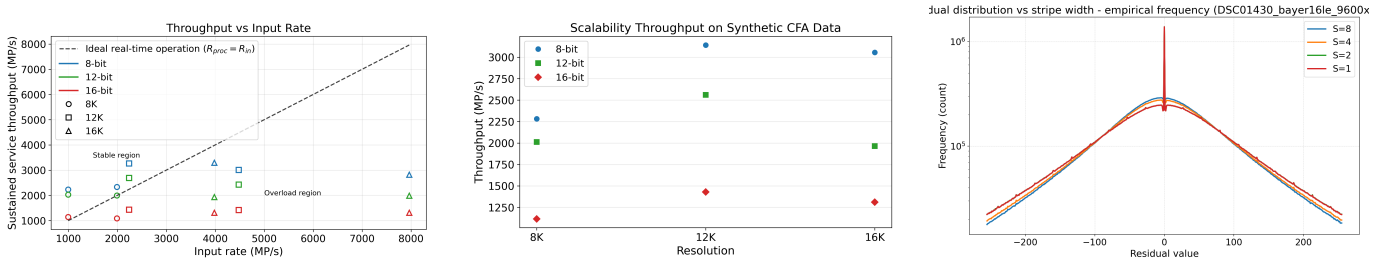
Fig. 8c shows the empirical distributions of prediction residuals for different stripe widths on real CFA data. Increasing the stripe width leads to a higher concentration of residuals around zero, indicating improved prediction accuracy and reduced conditional entropy. This effect is particularly visible in the central peak of the distribution, while differences in the tails remain more moderate.

Although the overall change is subtle, the trend is consistent across the residual range and explains the reduction in bitrate observed for wider stripes.

Conversely, narrower stripes limit the available prediction context, resulting in broader residual distributions and higher entropy.

Table V quantifies the impact of removing individual components: (1) removing prediction increases the bitrate from 7.988 to 10.997 bps (+37.7%), confirming its key role in exploiting spatial redundancy. This modification has limited impact on computational cost, indicating that the predictor is lightweight. (2) replacing the entropy model with a uniform distribution leads to a more severe degradation, increasing the bitrate to 16.040 bps (+100.8%) and significantly increasing execution time. (3) the full configuration achieves the best trade-off, combining low bitrate with competitive execution time. These results demonstrate that prediction and entropy modeling are complementary: prediction reduces residual entropy, while accurate modeling enables efficient coding of the resulting distribution. This confirms that the reduction in bitrate is primarily driven by improved residual concentration rather than changes in entropy coding efficiency.

Overall, this analysis confirms that both components are essential to achieve high compression efficiency without compromising throughput, confirming that the proposed architecture is not only efficient, but also well-balanced, with each component contributing significantly to the overall performance.



(a) Throughput vs input rate. (b) Throughput vs resolution. (c) Residual distributions vs stripe width.

Fig. 8: Performance and statistical behavior of the proposed method. (a) Relationship between input rate and sustained throughput, showing operating limits. (b) Compute-only throughput across resolutions, confirming near-linear scalability. (c) Residual distributions for different stripe widths, illustrating entropy reduction with wider stripes.

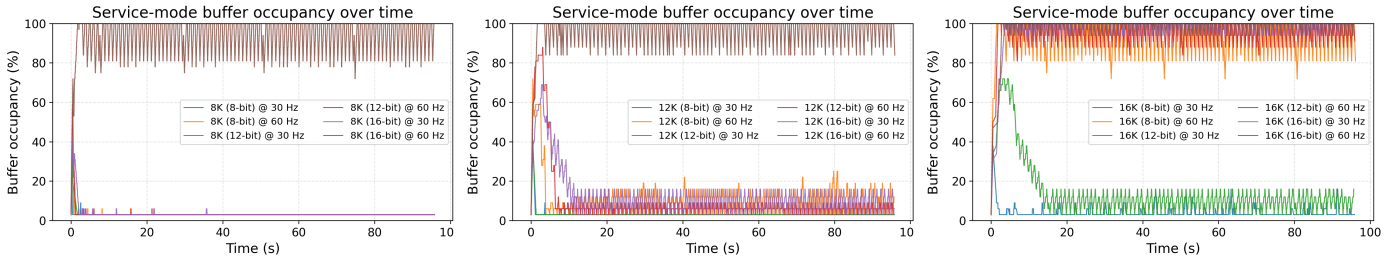


Fig. 9: Buffer occupancy over time for different resolutions under continuous acquisition, illustrating the transition from stable operation to saturation.

TABLE V: Ablation study of BLADE (Sony A7R IV).

Configuration	bps	End-to-end (ms)	No warm-up (ms)	Compute-only (ms)
Without prediction	10.997	248.960	124.388	16.696
Uniform entropy model	16.040	326.766	200.798	80.090
Full BLADE	7.988	242.705	116.664	17.352

G. Real-Time Performance

While previous sections focus on algorithmic performance, this section evaluates the proposed method under continuous acquisition, reflecting realistic deployment conditions.

The system operates in an I/O-bound regime, where data transfer dominates execution time, enabling efficient pipelined processing.

1) System Model: We analyze system behavior in a streaming pipeline where a producer continuously fills a shared-memory buffer with CFA images, while the compressor asynchronously processes available frames and writes compressed outputs. This setup captures system-level effects such as buffering, scheduling, and I/O interactions, which are not reflected in per-image benchmarks.

Experimental setup. Input streams are generated at fixed resolution and bit depth, with controlled input rates. Experiments run over sustained intervals (60–120 seconds) to ensure steady-state behavior.

Metrics. We monitor:

- Input rate (MP/s)
- Sustained throughput (MP/s)
- Buffer occupancy (%)
- Frame drops

TABLE VI: Summary of steady-state buffer behavior under continuous acquisition.

Resolution	Rate (Hz)	Buffer occupancy (%)	Time at saturation (s)
8K (16-bit)	30	3.0	0.0
8K (16-bit)	60	90.9	44.0
12K (8-bit)	30	3.0	0.0
12K (8-bit)	60	18.6	1.3
12K (12-bit)	30	3.0	0.0
12K (12-bit)	60	91.6	27.6
12K (16-bit)	30	89.6	10.2
12K (16-bit)	60	98.1	79.8
16K (8-bit)	30	3.1	0.0
16K (8-bit)	60	88.4	28.4
16K (12-bit)	30	6.4	0.0
16K (12-bit)	60	94.3	47.7
16K (16-bit)	30	98.9	80.5
16K (16-bit)	60	99.1	87.4

The system can be modeled as a discrete-time service system with rate R_{proc} and an arrival rate R_{in}. Stable operation occurs when:

$$R_{\text{proc}} \geq R_{\text{in}}, \tag{24}$$

ensuring bounded buffer occupancy. Otherwise, backlog accumulates, leading to saturation and potential frame drops.

2) Buffer Behavior: Table VI summarizes steady-state buffer occupancy and saturation time, providing a direct

TABLE VII: Performance comparison across GPU platforms. End-to-end and compute-only throughput are reported for both high-end (RTX 5000) and embedded (Jetson Orin NX) devices.

	RTX 5000				Jetson Orin NX			
	End-to-end		Compute-only		End-to-end		Compute-only	
Configuration	MP/s	ms	MP/s	ms	MP/s	ms	MP/s	ms
$\text{BLADE}_{512,4}$	569.2	107.529	4837.9	12.652	125.8	486.724	201.9	303.195
$\text{BLADE}_{512,2}$	555.6	110.175	5726.4	10.689	116.7	524.299	208.6	293.405
$\text{BLADE}_{256,4}$	567.3	107.887	5773.9	10.601	123.2	496.642	207.4	295.125
$\text{BLADE}_{256,2}$	569.4	107.507	5844.5	10.473	120.1	509.693	208.8	293.143

TABLE VIII: Performance–energy trade-off on Jetson Orin NX. Impact of power profiles on throughput and execution time.

Profile	Throughput (MP/s)	Time (ms)
10W	115.1	531.966
15W	125.8	486.7
25W	135.1	452.971
MAXN	201.1	304.310

characterization of system stability. Fig. 9 shows the temporal evolution of buffer occupancy.

Near saturation, the buffer exhibits periodic oscillations due to the mismatch between continuous data arrival and discrete frame-based processing. This results in alternating filling and partial draining phases, characteristic of steady-state operation close to system capacity.

Stable operation is observed when $R_{\text{proc}} \geq R_{\text{in}}$, while higher input rates lead to persistent high occupancy and eventual saturation. At 8K, real-time operation is sustained at 30 Hz, with saturation occurring at higher rates. At 12K, stable operation is maintained for lower bit depths at 30 Hz, while higher bit depths operate near system limits. At 16K, saturation appears earlier, particularly for high bit depths.

These results highlight that system stability is not solely determined by average throughput, but also by the interaction between processing granularity and input dynamics. In particular, even when operating near the theoretical limit $R_{\text{proc}} \approx R_{\text{in}}$, the system can sustain stable behavior provided that transient imbalances remain bounded.

Overall, these observations define a clear transition between stable and overload regimes, establishing the practical operating limits of the system under sustained load and reinforcing the importance of system-level analysis beyond per-image performance.

3) Edge Deployment: To assess edge deployment feasibility, we evaluate performance on an embedded NVIDIA Jetson Orin NX platform, representative of resource-constrained onboard processing scenarios.

Table VII compares performance across high-end and embedded GPU platforms, highlighting the gap between computational capability and system-level constraints.

On the RTX 5000, end-to-end throughput exceeds 550 MP/s across configurations, while compute-only performance reaches several GP/s, indicating that system-level performance is primarily limited by data movement rather than computation.

On the Jetson Orin NX, throughput reaches up to 125 MP/s, corresponding to approximately 3–4 Hz for 8K images, enabling sustained real-time operation under moderate input rates.

The condition $R_{\text{proc}} \geq R_{\text{in}}$ defines the boundary between stable operation and backlog accumulation, with reduced hardware capabilities shifting this operating region.

To analyze the performance–energy trade-off, we evaluate different power configurations on the Jetson Orin NX.

Table VIII quantifies the impact of power constraints on throughput and latency.

Throughput increases with available power, reaching 201.1 MP/s in MAXN mode, with sublinear scaling, indicating that memory bandwidth becomes the limiting factor.

Overall, the proposed method maintains predictable performance across hardware platforms and power budgets, enabling flexible deployment under diverse throughput and energy constraints.

VI. Conclusion

This paper presented BLADE, a GPU-oriented lossless compression framework for Bayer CFA images designed for real-time high-resolution imaging systems. The proposed method combines CFA-aware prediction with a block-and-stripe decomposition, enabling efficient exploitation of spatial redundancy while preserving fine-grained parallelism on GPU architectures.

Experimental results show that BLADE achieves competitive or superior compression performance compared to existing CFA methods, while significantly reducing execution time. The method operates close to the empirical Pareto frontier, achieving an effective balance between bitrate and throughput.

Beyond per-image evaluation, system-level analysis under continuous acquisition demonstrates sustained real-time operation within a well-defined operating region, determined by the relation between input rate and processing throughput. The method exhibits predictable behavior across resolutions, input rates, hardware platforms, and power constraints, including embedded GPU devices.

Overall, BLADE bridges compression efficiency and real-time processing requirements, making it a strong candidate for onboard and edge high-throughput imaging systems. Future work will explore more adaptive modeling strategies while preserving parallel scalability.

Although designed for Bayer CFA data, the proposed framework generalizes to other structured sampling patterns where

prediction and parallel execution must be jointly optimized. This highlights parallelism-aware statistical modeling as a fundamental design principle for lossless compression under high-throughput constraints.

APPENDIX A
NOTATION

This appendix summarizes the notation used throughout the paper for clarity and reference.

Symbol	Description
X, $x_{i,j}$	Input CFA image and sample at position (i, j)
b	Bit depth (bits per sample)
H, W	Image height and width
B	Block size
S	Stripe width
$\mathcal{B}_u$	Block u
$\mathcal{S}_k$	Stripe k
$\mathcal{B}$	Final bitstream
K	Total number of stripes
$\hat{x}_{i,j}$	Predicted sample value
$r_{i,j}$	Prediction residual
$r'_{i,j}$	Mapped (non-negative) residual
$\theta_{i,j}$	Range-adaptive mapping parameter
$H_k(v), H(v)$	Local and global histograms
$P(v)$	Probability model
b_k	Bitstream generated by stripe k
ℓ_k, ℓ'_k	Allocated and valid bitstream lengths
W	GPU warp size

REFERENCES

[1] D. Menon, S. Andriani, and G. Calvagno, "Demosaicing with directional filtering and a posteriori decision," *IEEE Transactions on Image Processing*, vol. 16, no. 1, pp. 132–141, 2007.

[2] H. S. Malvar, L. wei He, and R. Cutler, "High-quality linear interpolation for demosaicing of bayer-patterned color images," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2004, pp. 485–488.

[3] N. Zhang and X. Wu, "Lossless compression of color mosaic images," *IEEE Transactions on Image Processing*, vol. 15, no. 6, pp. 1379–1388, 2006.

[4] K.-H. Chung and Y.-H. Chan, "A lossless compression scheme for bayer color filter array images," *IEEE Transactions on Image Processing*, vol. 17, no. 2, pp. 134–144, 2008.

[5] A. Bazhyna and K. Egiazarian, "Lossless and near lossless compression of real color filter array data," *IEEE Transactions on Consumer Electronics*, vol. 54, no. 4, pp. 1492–1500, 2008.

[6] S. Kim and N. I. Cho, "Lossless compression of color filter array images by hierarchical prediction and context modeling," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 6, pp. 1040–1046, 2014.

[7] K. M. M. e. a. Rahman, "A low complexity lossless bayer cfa image compression," *Signal, Image and Video Processing*, vol. 15, pp. 1767–1775, 2021.

[8] H. Chen *et al.*, "Lossless white balance for improved lossless cfa image compression," *IEEE Transactions on Image Processing*, 2022.

[9] M. Hernández-Cabronero, M. W. Marcellin, I. Blanes, and J. Serra-Sagristà, "Lossless compression of color filter array mosaic images with visualization via jpeg 2000," *IEEE Transactions on Multimedia*, vol. 20, no. 2, pp. 257–270, 2018.

[10] M. J. Weinberger, G. Seroussi, and G. Sapiro, "The loco-i lossless image compression algorithm: Principles and standardization into jpeg-ls," *IEEE Transactions on Image Processing*, vol. 9, no. 8, pp. 1309–1324, 2000.

[11] D. Taubman and M. Marcellin, *JPEG2000: Image Compression Fundamentals, Standards and Practice*. Springer, 2002.

[12] *ISO/IEC 21122-1:2019, Information technology – JPEG XS low-latency lightweight image coding system – Part 1: Core coding system*, ISO/IEC Std., 2019.

[13] D. Taubman, P. Schelkens, and T. Ebrahimi, "Jpeg xs: A new standard for visually lossless low-latency lightweight image compression," *IEEE Signal Processing Magazine*, vol. 36, no. 6, pp. 105–114, 2019.

[14] J. Alakuijala *et al.*, "Jpeg xl next-generation image compression architecture and coding tools," in *SPIE Applications of Digital Image Processing*, 2019.

[15] T. Richter, S. Fößel, A. Descampe, and G. Rouvroy, "Bayer cfa pattern compression with jpeg xs," *IEEE Transactions on Image Processing*, vol. 30, pp. 6557–6569, 2021.

[16] X. Liu and N. Ahuja, "Lossless compression of bayer pattern images," *IEEE Transactions on Image Processing*, 2010.

[17] C. Lee and Y. Kim, "Lossless compression of cfa images using hierarchical prediction," *IEEE Transactions on Circuits and Systems for Video Technology*, 2011.

[18] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, "Variational image compression with a scale hyperprior," in *International Conference on Learning Representations (ICLR)*, 2018. [Online]. Available: https://openreview.net/forum?id=rkcQFMZRb

[19] D. Minnen, J. Ballé, and G. Toderici, "Joint autoregressive and hierarchical priors for learned image compression," *NeurIPS*, 2018.

[20] F. e. a. Mentzer, "High-fidelity generative image compression," *NeurIPS*, 2020.

[21] Z. e. a. Cheng, "Learned image compression with discretized gaussian mixture likelihoods," *CVPR*, 2020.

[22] L. e. a. Theis, "Lossy image compression with neural networks," *IEEE Transactions on Computational Imaging*, 2022.

[23] R. e. a. Yang, "Lossless image compression with neural networks," *IEEE Transactions on Circuits and Systems for Video Technology*, 2022.

[24] C. e. a. Schröder, "Neural lossless image compression: A review," *IEEE Access*, 2023.

[25] A. B. et al., "nvcomp: High-performance lossless data compression on gpus," in *IEEE High Performance Extreme Computing Conference (HPEC)*, 2022.

[26] NVIDIA Corporation, "Gdeflate: A gpu-friendly deflate compression format," in *NVIDIA Technical Report / Developer Blog*, 2021.

[27] ——, "nvJPEG2000 Library," https://docs.nvidia.com/cuda/nvjpeg2000/index.html, 2023, accessed: 2026.

[28] C. D. Cea-Dominguez, J. C. Moure-Lopez, J. Bartrina-Rapesta, and F. Aulí-Llinàs, "Gpu-oriented architecture for an end-to-end image/video codec based on jpeg2000," *IEEE Access*, vol. 8, pp. 68 474–68 487, 2020.

[29] J. Z. et al., "High-performance lossless compression on gpus for scientific data," *IEEE Transactions on Parallel and Distributed Systems*, 2023.

[30] S. Di and F. Cappello, "Fast error-bounded lossy hpc data compression with sz on gpus," in *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2020.

[31] J. Bartrina-Rapesta, J. Serra-Sagristà, and S. Mijares i Verdú, "BLADE: GPU-Oriented Lossless Compression of Bayer CFA Images," 2026, zenodo.

[32] X. Wu and N. Memon, "Context-based, adaptive, lossless image coding," *IEEE Transactions on Communications*, vol. 45, no. 4, pp. 437–444, 1997.

[33] Consultative Committee for Space Data Systems (CCSDS), "Low-complexity lossless and near-lossless multispectral and hyperspectral image compression," CCSDS, Tech. Rep. 123.0-B-3, 2022.

[34] Sony Semiconductor Solutions, "IMX455 Full-Frame CMOS Image Sensor," https://www.sony-semicon.com/en/products/is/camera/, 2019, accessed: 2026.

[35] NVIDIA Corporation, "nvCOMP Library," https://developer.nvidia.com/nvcomp, 2023, accessed: 2026.