

Repositório ISCTE-IUL

Deposited in *Repositório ISCTE-IUL*:

2026-10-01

Deposited version:

Accepted Version

Peer-review status of attached file:

Peer-reviewed

Citation for published item:

Sezavar, A., Brites, C., Ascenso, J. & Ebrahimi, T. (2025). A learning-based lossless event data compression for computer vision applications. In Tescher A.G., Ebrahimi T. (Ed.), Proceedings of SPIE - The International Society for Optical Engineering. SPIE. <https://doi.org/10.1117/12.3068095>

Further information on publisher's website:

[10.1117/12.3068095](https://doi.org/10.1117/12.3068095)

Publisher's copyright statement:

This is the peer reviewed version of the following article: Sezavar, A., Brites, C., Ascenso, J. & Ebrahimi, T. (2025). A learning-based lossless event data compression for computer vision applications. In Tescher A.G., Ebrahimi T. (Ed.), Proceedings of SPIE - The International Society for Optical Engineering. SPIE. <https://doi.org/10.1117/12.3068095>, which has been published in final form at <https://dx.doi.org/10.1117/12.3068095>. This article may be used for non-commercial purposes in accordance with the Publisher's Terms and Conditions for self-archiving.

Use policy

Creative Commons CC BY 4.0

The full-text may be used and/or reproduced, and given to third parties in any format or medium, without prior permission or charge, for personal research or study, educational, or not-for-profit purposes provided that:

- a full bibliographic reference is made to the original source
- a link is made to the metadata record in the Repository
- the full-text is not changed in any way

The full-text must not be sold in any format or medium without the formal permission of the copyright holders.

A learning-based lossless event data compression for computer vision applications

Ahmadreza Sezavar^{ac}, Catarina Brites^{bc}, João Ascenso^{ac}, Touradj Ebrahimi^{cd}

^aInstituto Superior Técnico (IST), Instituto de Telecomunicações (IT), Lisboa, Portugal

^bInstituto Universitário de Lisboa (ISCTE-IUL), Instituto de Telecomunicações (IT), Lisboa, Portugal

^cRayShaper SA, Crans-Montana Valais, Switzerland, ^dEPFL, Lausanne, Vaud, Switzerland

ABSTRACT

Event-based computer vision is becoming very popular. With progress in sensing events, the volume of data produced has increased manyfold, and there is a need for compression. This paper introduces a novel deep-learning-based lossless event data compression codec. The idea is to represent the events as a point cloud with spatial dimensions x and y and temporal dimension t as its coordinates. Then, an adaptive octree structure is created to better compact the latter without introducing any loss by coding the occupancy map. The binary representation of the octree structure, which corresponds to a denser representation of the event data, is then entropy-coded with a learning-based model. The latter is based on using a deep neural network to obtain the probability model of a hyperprior-based arithmetic coder. The proposed hyperprior network architecture includes two neural networks following an auto-encoder structure, which allows the capture of the source statistics effectively.

Keywords: event cameras, event compression, hyperprior, learning-based, octree, arithmetic coding

1. INTRODUCTION

Event-based imaging, also known as neuromorphic imaging, is a recent visual information modality that relies on emerging bio-inspired sensors that try to mimic the sensing behavior of the biological retina, the so-called event-based vision sensors. Contrary to conventional cameras, where all sensor pixels acquire visual information simultaneously at regular time intervals, in event cameras each sensor pixel is in charge of controlling the light acquisition process asynchronously and independently, according to the dynamics of the visual scene (e.g., when lighting conditions change or due to scene/camera motion), thus producing a variable data rate output.

Generally, event cameras follow a differential visual sampling model in which time-domain changes in the incoming light intensity (i.e., temporal contrast) are pixel-wise detected and compared to a threshold, triggering a so-called event if it exceeds the threshold. This visual sampling model allows event cameras to provide interesting advantages over conventional cameras, such as high temporal resolution (small time interval at which a sensor pixel can react to scene dynamics), very high dynamic range, low latency, and low power consumption. These are rather compelling properties in scenarios particularly challenging to conventional cameras, such as scenes with high-speed motion and/or uncontrolled lighting conditions. Therefore, event cameras may benefit several computer vision applications, such as autonomous driving, drones, robotics, etc., where conventional cameras usually fail to provide good performance. Moreover, event cameras are also advantageous in industrial automation, visual surveillance, augmented reality, and mobile devices, where fast response, high dynamic range, or low power consumption is critically needed.

A standardized event data coding framework is currently not yet available, which is an obstacle to the deployment of some applications where this type of visual information acquisition (sensing) might be profitable. Indeed, standardization can offer a well-defined format for representing visual information captured by event cameras, allowing sensor manufacturers to interoperate with other technology providers to exploit event data efficiently. For this reason, the JPEG committee believes that the time has come to conduct work towards the standardization of event-based imaging formats.

Further author information: send correspondence to Ahmadreza Sezavar: ahmadreza.sezavar@lx.it.pt, Catarina Brites: catarina.brites@lx.it.pt, João Ascenso: joao.ascenso@lx.it.pt, Touradj Ebrahimi, touradj@rayshapersa.ch

In this paper a deep-learning-based lossless compression technique for event data submitted to the JPEG XE Call for Proposals [1] is presented, expanding on some initial work presented in [2]. The approach relies on organization into a 3D representation defined by the x and y spatial coordinates and the temporal dimension t of each event. This 3D representation is then adaptively partitioned using an octree, a hierarchical structure that subdivides the space into partitions based on event density, optimizing the representation of sparse and dense regions alike. The resulting binary representation of the octree, which efficiently encodes the event data into a more compact form, is subsequently compressed using an entropy encoder guided by a learning-based entropy model. A significant contribution of this approach lies in its use of a deep neural network to develop a probability model for a hyperprior-based arithmetic coding, enhancing the compression efficiency by accurately predicting the statistical distribution of the encoded data.

2. LEARNING-BASED LOSSLESS EVENT COMPRESSION

The proposed learning-based lossless event data compression architecture, hereafter referred to as the LLEC, relies on two key lossless operations: i) adaptive partitioning of the space-time representation of events via an octree; and ii) entropy coding with a learning-based entropy model derived from an auxiliary latent representation (hyperprior).

Octree structures are particularly well-suited to sparse data formats, such as event-based streams, due to their ability to adaptively partition space by focusing only on regions containing data. Rather than uniformly subdividing the entire spatiotemporal event volume, the octree recursively divides only those regions where events are present, thereby avoiding unnecessary overhead from empty regions. The process begins by representing the event data in three dimensions — spatial coordinates (x, y) and temporal coordinate (t) — and dividing this space into eight equal partitions, or voxels. Each voxel is classified as either empty or non-empty, depending on whether it contains events. Non-empty voxels are recursively subdivided into eight smaller voxels until a predefined maximum depth is reached. The occupancy of each voxel can be encoded as an 8-bit binary sequence, where each bit indicates whether a corresponding child voxel contains events. Depending on the selected depth, the resulting binary octree structure can support either lossless or lossy representations.

To compress the octree structure, specifically its occupancy information, a learned entropy coding model is employed. This model leverages a hyperprior (auxiliary side information) to estimate the probability distribution of the occupancy symbols. By integrating the hyperprior, the probability model can be dynamically updated during the encoding process, allowing the system to adapt to the evolving statistics of the input data. This adaptive behavior improves compression efficiency compared to static models. Additionally, the learned entropy model captures complex dependencies within the data, outperforming conventional context models that rely solely on neighboring or previously decoded nodes. As a result, the proposed LLEC codec achieves more effective and compact lossless compression of event data.

2.1. Overall Architecture and Dataflow

This sub-section describes the overall architecture of the LLEC and its dataflow. Prior to compression as depicted in Figure 1, the input event stream may undergo several pre-processing operations: First, if the event stream contains polarity information, it is divided into two sub-streams based on their polarity — one for positive polarity (S^+) and one for negative polarity (S^-). Next, each sub-stream may be divided into temporally independent segments, each with a fixed time interval length T_s , denoted as S_i^+, S_i^- , where i identifies a specific segment. Finally, within each segment (S_i^+, S_i^-), the timestamp (t) of every event is adjusted by subtracting the minimum timestamp in that segment, resulting in timestamp values ranging from 0 to T_s for each segment (S_i^+, S_i^-). After this preprocessing, each segment — regardless of polarity — is independently compressed using the proposed LLEC core coding engine.

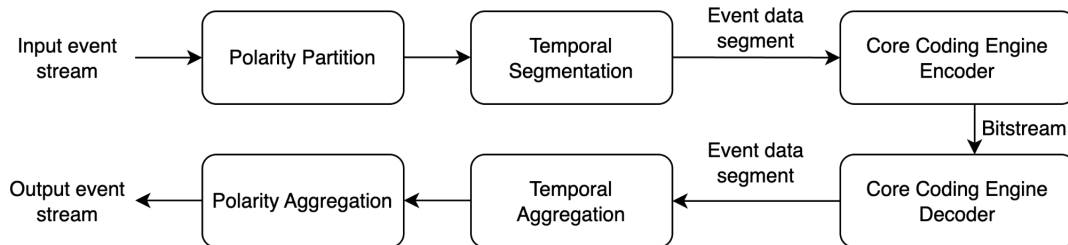


Figure 1. LLEC overall architecture.

The core coding engine architecture is presented in Figure 2. The following operations are performed at the encoder:

1. **Stream to Octree Conversion:** An octree is constructed from the pre-processed input data (S_i^+, S_i^-) at the encoder side. The depth of the octree, denoted as D , is calculated as $D = \log_2(\max(x_{\max}, y_{\max}, t_{\max}))$, where $x_{\max}$, $y_{\max}$ and $t_{\max}$ represent the maximum possible values for the spatial coordinates (x and y) and the timestamp (t), respectively. This computed depth D ensures the process is lossless, meaning that each leaf node uniquely corresponds to a specific spatiotemporal location without ambiguity.
2. **Octree Encoding:** On the encoder side, the octree structure is encoded by traversing each node and extracting its occupancy information - a stream of occupancy bytes, where '1' represents a non-empty node and '0' represents an empty node of the octree. Additionally, at the encoder, the occupancy stream is grouped into sets of tiles, each with a size of $N = 512$, for subsequent processing, forming the occupancy tiles.
3. **Hyperprior Encoder:** The hyperprior encoder generates a compact latent hyperprior code (z) of size M for each occupancy tile. This latent code captures spatial dependencies among the elements within the occupancy tile and is transmitted to the decoder (after quantization) as side information. Experimental results indicate that a size of $M = 8$ strikes an effective balance between the compactness of the latent code and the overall performance of the codec.
4. **Quantization:** The scalar uniform quantization method, detailed in Section 3.3, is applied to the latent code produced by the hyperprior encoder. The resulting quantized latent symbols (z') are sent to the decoder without entropy coding, as the quantized latent code is sufficiently compact and requires only minimal bitrate. The number of levels of the quantizer is 64.
5. **Hyperprior Decoder:** The hyperprior decoder generates the probability model for each occupancy tile using the quantized latent symbols. This step is performed identically on both the encoder and decoder sides to ensure the same probability model is used at both ends.
6. **Arithmetic Encoder:** An arithmetic encoder, utilizing the probability model computed in the previous step, is employed to encode the occupancy tiles. This arithmetic codec is implemented using the torchac library ([3][4]), which supports encoding, generating, parsing, and decoding the bitstream based on the cumulative distribution of the probability model.

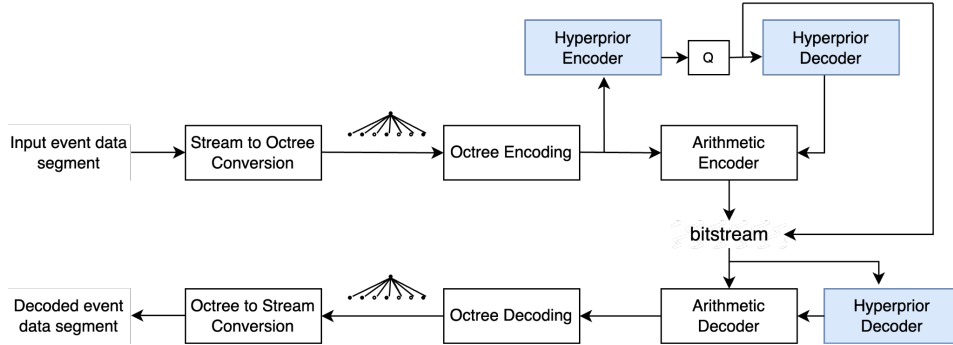


Figure 2. LLEC core coding engine.

At the decoder side, the inverse operations are performed. The first operation is to parse the quantized hyperprior latent code (z') which is used by the hyperprior decoder to obtain the same probability model that was obtained at the encoder, but this time at the decoder. After arithmetic decoding is performed and the occupancy stream obtained (partitioned into tiles), after octree decoding is performed creating the octree which is then traversed until all leaf nodes are reached, reconstructing the decoded event data identical to the original input.

2.2. Hyperprior Network Architecture

The hyperprior network architecture of LLEC is based on an auto-encoder structure, designed to process and transform occupancy tiles of size N through multiple neural network layers, effectively capturing intricate relationships within the input data. Both the hyperprior encoder (shown in Figure 3a) and the hyperprior decoder (shown in Figure 3b) are composed of a sequence of fully connected (FC) layers, each followed by batch normalization (BN) and exponential linear unit (ELU) activation layers. In the hyperprior encoder, a hyperbolic tangent (tanh) activation function is used to produce latent code elements constrained within the range $[-1, 1]$. In the hyperprior decoder, a softmax activation function is applied as the final layer to generate the probability distribution model.

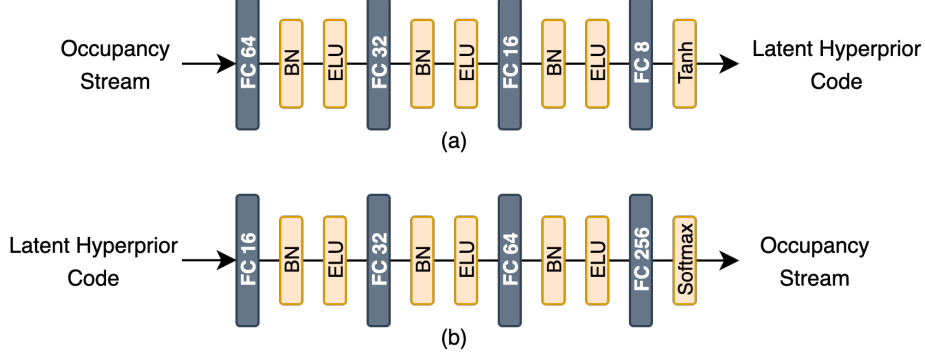


Figure 3. Proposed hyperprior network architecture: (a) encoder, (b) decoder.

2.3. Quantization

The scalar quantization approach outlined in [3] is utilized to quantize the output (z) of the hyperprior encoder. Given a set of quantization levels $L = \{l_1, l_2, \dots, l_L\} \subset \mathbb{R}$, each latent code element z_i is quantized by assigning it to the nearest neighboring quantization level, as defined in (1).

$$z'_i = Q(z_i) := \arg \min_j |z_i - l_j| \quad (1)$$

However, the quantization operation is inherently non-differentiable, posing a challenge to conventional gradient-based optimization training methods, which require all operations to be differentiable. To overcome this and enable gradient computation during the backward pass, a differentiable soft quantization approach is employed, as described in (2).

$$Q(z_i) = \sum_{j=1}^L \frac{e^{-\sigma_q |z_i - l_j|}}{\sum_{l=1}^L e^{-\sigma_q |z_i - l_l|}} l_j \quad (2)$$

In (2), σ_q is a hyperparameter that controls the "softness" of the quantization process. In this implementation, the number of levels (L) is set to 64, with levels evenly distributed across the range $[-1, 1]$, and σ_q is assigned a value of 2.

2.4. Training Procedure

During preprocessing, the time interval T_s was set to 2048 for sequences with a spatial resolution of 1280×780 and to 1024 for sequences with a resolution of 640×480 ; these T_s values are also applied during inference. The validation sequences were utilized to fine-tune hyperparameters, including the number of layers, batch size, and learning rate. The Adam optimizer was employed with a learning rate of 10^{-4} . Additionally, a scheduler with a decay rate of 0.1 every 5 epochs is implemented. The proposed hyperprior used a latent code size of $M = 8$ elements and a batch size of 512 elements, as these values have been experimentally determined to yield optimal performance. Training was conducted over 100 epochs, with an early-stopping mechanism featuring a patience of 10 epochs to prevent overfitting. The hyperprior network training was executed on a machine equipped with an NVIDIA GeForce RTX 3090 GPU. The loss function aims to minimize the expected bitstream length (rate), as defined in (3).

$$R = \mathbb{E}_{x \sim p(x)} [-\log_2 \hat{p}(x)] \quad (3)$$

Here, x represents the input occupancy tile, $p(x)$ denotes the unknown distribution of input occupancy tiles, and $\hat{p}(x)$ is the discrete entropy model estimated by the hyperprior decoder. Equation (3) expresses the cross-entropy between the marginal distribution of the latent code and the learned entropy model, serving as the loss function during training. This cross-entropy, or loss, is minimized when the two distributions align perfectly.

3. BITSTREAM SYNTAX

The bitstream syntax produced by the proposed LLEC codec is organized hierarchically to ensure efficient compression, ease of access, and robustness against data corruption. It is composed of three main layers: a global header, a set of event segments, and one or more tiles per segment. This organization facilitates independent decoding across temporal and polarity partitions.

The global header, which initiates the bitstream, has a fixed size of 10 bytes and begins with a reserved marker value (0xFF01). It encodes two key pieces of metadata: the minimum timestamp observed for OFF-polarity events and the minimum timestamp for ON-polarity events. These global timing references are used to normalize timestamps during decoding and ensure synchronization across segments.

Each event segment corresponds to a time-constrained group of events of a single polarity (ON or OFF). Segment boundaries are marked by specific identifiers—0xFF02 for OFF-polarity segments and 0xFF03 for ON-polarity segments. A segment includes metadata such as the number of octree blocks, represented as a 16-bit integer, and a 32-bit local time offset, which records the minimum timestamp within that segment. This allows accurate temporal reconstruction within each polarity group.

Within each segment, data is further divided into tiles, each encoding a portion of the spatiotemporal octree volume. Each tile includes a 32-bit field that records the length of the compressed bitstream, followed by 48 bits encoding latent variables, used to reconstruct the entropy model parameters on the decoder side. The remaining portion of each tile is the compressed bitstream, produced by an arithmetic coder guided by a learned hyperprior model.

4. PERFORMANCE EVALUATION

4.1. Dataset Description

The JPEG XE reference dataset, as adopted by the JPEG XE CTC [5], comprises an extensive collection of raw event sequences that encompass a diverse range of use cases, environmental conditions, durations, sensors, and sensor configurations. This dataset is encoded in the Canonical format (defined in JPEG XE CTC), with events arranged in chronological order according to their timestamps. For this proposal, as shown in Table 1, sequences from Prophesee were selected for training and validation, with the dataset featuring four distinct sequences: Spinner, 195_falling_particles, Traffic_monitoring, and Hand_spinner. These sequences maintain a consistent 640×480 pixel resolution, with three sequences used for training and one for validation. The sequences exhibit varying durations and event rates, ranging from 0.032 to 28 seconds, and event rates from 0.64 to 632.52 Mev/s, providing a diverse dataset for comprehensive model development.

Table 1. Event sequences selected for training and validation.

Sequence	Train/validation	Resolution	Duration	Event-rate (Mev/s)
Spinner	Train	640×480	5	10.83
195_falling_particles	Train	640×480	0.032	632.52
traffic_monitoring	Train	640×480	28	0.64
hand_spinner	Validation	640×480	5	2.39

Table 2. Event sequences selected for inference.

Sequence name	Resolution	Duration (s)	Event-rate (Mev/s)
activemarkers_handheld	1280×720	10.1	0.60
deblur_street	1280×720	2.4	64.6
depthsensing_highspeedlaser	1280×720	3.0	38.8
eyetracking_right	1280×720	9.9	5.75
industrial_counting	640×480	6.3	0.73
industrial_fluidflow	1280×720	5.6	1.88
industrial_spray	640×480	4.3	1.30
localization_cube	640×480	10.0	8.29
surveillance_startracking	1280×720	12.3	1.84

4.2. Performance Metrics

To evaluate performance, two metrics adopted by the JPEG XE CTC [2] are employed: the compression ratio (CR) and the average compressed event size in bits (S). The CR metric quantifies the size reduction of the compressed bitstream produced by the proposed LLEC solution (measured in bits) relative to the size (in bits) of the input sequence in the canonical raw format, as defined in (4). Notably, the compressed rate accounts for both the bits required to transmit the hyperprior and the arithmetic-coded symbols.

$$CR = \frac{\text{size}(\text{input sequence})}{\text{size}(\text{compressed bitstream})} \quad (4)$$

The S metric provides an alternative measure of compression efficiency by calculating the average number of bits used by the proposed LLEC solution to represent a single event. It is computed as the ratio of the compressed bitstream size (in bits) to the total number of events in the input sequence, expressed in bits per event, according to (5).

$$S = \frac{\text{size}(\text{compressed bitstream})}{\text{number of input events}} \quad (5)$$

4.3. Experimental Results

For inference, sequences from the JPEG Common Test Conditions were used, and their statistical properties are shown in Table 2. They are evaluated using an NVIDIA GeForce RTX 3090 GPU paired with an 11th Gen Intel Core i9-11900K CPU, an x86_64 architecture processor featuring 16 threads across 8 cores. To ensure accuracy, results are cross-verified using an NVIDIA GeForce RTX 4090 GPU with an AMD Ryzen Threadripper PRO 5955WX 16-Cores CPU. The experimental results obtained are reported in Table 3.

Compression ratios (CR) across the sequences range from 2.01 to 4.26, with an average CR of 3.08, indicating that the LLEC codec reduces the size of event data by approximately 67% on average. The highest compression ratios are observed for sequences such as deblur_street (4.26) and activemarkers_handheld (4.23), suggesting that LLEC performs particularly well on both scenes with high and low temporal redundancy. On the other hand, lower CR values, such as those seen in surveillance_startracking (2.01) and industrial_fluidflow (2.05), may reflect either lower spatial-temporal correlation in the event stream or reduced sparsity in those scenes.

The average bits per event (S) metric provides further insight into the per-symbol efficiency of the codec. The values range from 11.80 to 24.95, with an overall average of 17.69 bits per event. This indicates that the LLEC codec is able to encode each event with a relatively compact bit representation, despite the requirement for exact lossless reconstruction. Event

sequences with lower S values, such as `deblur_street` and `activemarkers_handheld`, again highlight scenarios where the model is particularly well adapted to the data statistics.

Table 3. Coding efficiency experimental results for all test sequences.

Sequence name	Compressed size (byte)	Original size (byte)	CR	S
<code>activemarkers_handheld</code>	12.2 MB	36.4 MB	4.23	11.89
<code>deblur_street</code>	240.8 MB	930 MB	4.26	11.80
<code>depthsensing_highspeedlaser</code>	204.5 MB	698 MB	3.83	13.13
<code>eyetracking_right</code>	164.8 MB	341 MB	2.41	20.87
<code>industrial_counting</code>	13.5 MB	27.5 MB	3.02	16.63
<code>industrial_fluidflow</code>	37.7 MB	63 MB	2.05	24.28
<code>industrial_spray</code>	22.3 MB	33.7 MB	2.27	22.09
<code>localization_cube</code>	160.4 MB	495.7 MB	3.69	13.62
<code>surveillance_startracking</code>	83.3 MB	136 MB	2.01	24.95
AVERAGE	89.6 MB	307.4 MB	3.08	17.69

Overall, these results validate the codec’s design, especially its ability to adapt to varying event densities and statistical properties via hyperprior-guided entropy modeling. Although some sequences remain more challenging, the LLEC codec consistently provides substantial compression gains, making it a strong candidate for efficient lossless event representation in real-world computer vision pipelines. Further improvements could be expected with complexity optimizations and scene-adaptive parameter tuning.

5. CONCLUSIONS

This paper introduced a deep learning-based lossless compression method for event-based data, using an adaptive octree structure and a hyperprior-guided entropy model. The proposed LLEC codec effectively captures spatiotemporal sparsity and achieves high compression efficiency, with a compression ratio of 3.08 (average) on the JPEG XE dataset. Results demonstrate strong performance across diverse sequences, confirming the codec’s potential for efficient event data transmission in computer vision applications. However, current limitations include high computational cost due to the octree structures and a Python-based implementation, which is clearly an opportunity for future optimization and acceleration.

ACKNOWLEDGEMENTS

The authors would like to acknowledge financial support from Innosuisse Swiss Innovation Agency under Innovation project Eurostars Project E!2610 EVASION in the framework of Eureka Eurostars.

REFERENCES

- [1] ISO/IEC JTC 1/SC29/WG1 N100888, REQ "Final Call for Proposals on JPEG XE", 104th Meeting, Sapporo, Japan, July 2024.
- [2] A. Sezavar, C. Brites and J. Ascenso, "Learning-based Lossless Event Data Compression," IEEE International Conference on Visual Communications and Image Processing (VCIP), Tokyo, Japan, 2024, pp. 1-5, doi: 10.1109/VCIP63160.2024.10849853.
- [3] F. Mentzer, E. Agustsson, M. Tschannen, R. Timofte, and L. Van Gool, "Practical Full Resolution Learned Lossless Image Compression", IEEE Conference on Computer Vision and Pattern Recognition (CVPR), California, USA, 2019.
- [4] "torchac: Fast Arithmetic Coding for PyTorch", <https://github.com/fab-jul/torchac>
- [5] ISO/IEC JTC 1/SC29/WG1 N100891, CPM "Common Test Conditions for JPEG XE v2.0", 104th Meeting, Sapporo, Japan, July 2024.