Efficient lightweight image compression approach for Multimedia Internet of Things

Sara Kadhum Idrees[1,4], Raphaël Couturier[2], Joseph Azar[2], Ali Kadhum Idrees[3], Franck Gechter[1,5]

[1]*Université Marie et Louis Pasteur - UTBM - CIAD (UR 7533), F-90000 Belfort, France*
[2]*Université Marie et Louis Pasteur, CNRS, institut FEMTO-ST, F-90000 Belfort, France*
[3]*Smart Edge Lab, Faculty of Computer Science, University of Applied Science and Arts, Dortmund, Germany*
[4]*Dept. of Information Networks, College of Information Technology, University of Babylon, Babylon, Iraq*
[5]*LORIA (UMR 7503), SIMBIOT Team, F-54506, Vandoeuvre-lès-Nancy, France*
Emails: {sara.idrees,franck.gechter}@utbm.fr, {raphael.couturier,joseph.azar}@univ-fcomte.fr, ali.idrees@fh-dortmund.de

***Abstract*—The development of the Internet of Things (IoT) and communication technologies has led to the emergence of the multimedia IoT. The multimedia IoT contributes to a variety of multimedia applications, that can capture and transmit large amounts of images or video sequences over an IoT network. Due to restricted resources, multimedia IoT devices must transmit less multimedia data while retaining accuracy. This can save energy and memory use and increase multimedia IoT network performance. This paper proposes an efficient lightweight image compression approach (ELiCA) for Multimedia Internet of Things. The proposed ELiCA starts dividing the captured image into equal-sized blocks and the Discrete Cosine Transform (DCT) is applied to each of them to produce the DCT coefficients. Then, the DCT coefficients are quantized. The output of the quantization process is reordered using zigzag process. Finally, the output files of the zigzag are compressed using the error-bounded lossy compression technique known as SZ developed specifically for High-Performance Computing (HPC) systems. The compressed file will be transmitted by the MIoT device. The proposed ELiCA is evaluated using real sensed images and based on real IoT devices to show the efficiency of the proposed approach. The ELiCA presents better results compared with JPEG in terms of compression ratio, PSNR, and SSIM.**

***Index Terms*—Multimedia Internet of Things, SZ compression, Embedded IoT, DCT, Quantization, Lightweight Data Compression.**

I. Introduction

The Multimedia Internet of Things (MIoT) represents a progression beyond the conventional IoT model, integrating the transmission and analysis of multimedia data (such as audio, video, and photos) in addition to sensor data. This can be done through several multimedia applications, such as wireless endoscopies [1], traffic monitoring [2], and smart glasses [3]. The MIoT devices used in these applications face significant hurdles. They have limited resources in terms of their processing capabilities, memory capacity, and battery longevity. The MIoT Network bandwidth can be significantly burdened by the large amount of multimedia data, particularly in real-time scenarios such as remote surgery or autonomous vehicles. Moreover, processing and transmitting large amount of MIoT data lead to increase the Latency in the MIoT network [4]. Therefore, it is necessary to reduce transmitted MIoT data by removing redundant ones. In this case, utilizing image compression techniques can help to reduce the amount of storage space required to save the MIoT data and extend the battery lifetime of limited power MIoT devices [5]. Usually it is considered that IoT devices have enough space of memory and processing resources to run complex and effective algorithms [6]. However, with the purpose to attain low-cost structures, IoT platforms depend often on useful resource confined microprocessors with only some kilobytes of storge [6]. Because of this, low-complexity and low-memory compression techniques are typically recommended, if it is not mandatory [7]. There are numerous state-of-the-art high performance image compression standards , such as HEVC [8], JPEG2000 [9], baseline JPEG [10], and H.264 [11]. However, the memory demands of these compression standards do not align with the memory constraints of numerous IoT devices. Image compression is broadly classified into two categories, namely lossless and lossy, depending on whether the original image can be recovered with full mathematical precision from the compressed image [12]. Lossy compression techniques vary from lossy techniques in that they introduce gradually controlled distortion in the reconstructed image, allowing much higher compression ratios. This trade-off between image quality and compression efficiency has been a major principle of research in the domain of image processing [13]. While there are several existing lossy image compression techniques, most of them are not suitable for devices with limited storage and computational resources [14]. Thus, the trade-off between computational complexity and the achievable compression ratio needs to be further taken into account in the case of resource-constrained devices.

This paper presents the following contributions:

- we propose an efficient lightweight image compression approach (ELiCA) for the Multimedia Internet of Things. ELiCA partition the input image into equal size blocks and applies the DCT, quantization, and zigzag for each block. Finally, ELiCA applies the SZ MIoT compressor to the resulting files to reduce the size of compressed

MIoT data.

- The proposed ELiCA was evaluated using real experiments that were implemented on real Portenta H7 Lite microcontroller with real image datasets from Kodak dataset [15] and the BOSSbase dataset [16]. The conducted results show a very efficient compression ratio compared with the standard baseline JPEG for low image quality while maintaining the same degree of image quality.

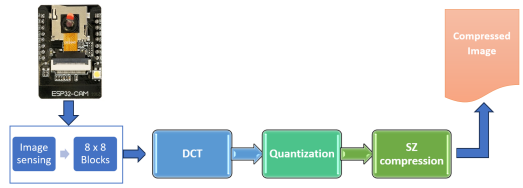
II. Related works

The image compression techniques can be categorized into lossless and lossy techniques. Lossless compression strategies prioritize preserving the visible content integrity, ensuring that the decompressed photograph is the same as the original one. On the other hand, lossy compression sacrifices a few picture facts to obtain a higher compression ratio, making it ideal for decreasing bandwidth usage in networks. Typically, the compression ratio inversely impacts the image best, in which a lower compression ratio corresponds to a better-perceived image quality, and conversely, a better compression ratio consequence in a lower image quality. Over the years, numerous systems and techniques have been developed and implemented for lossy image compression. One of the broadly followed techniques for lossy compression is the utilization of transform-based methods [17]. These methods involve transforming the data of the image into a different domain like the discrete cosine transform (DCT) or the discrete wavelet transform (DWT), followed by quantization and then executing reversible coding algorithms like Huffman Coding [18] on the quantized data to achieve the compression.

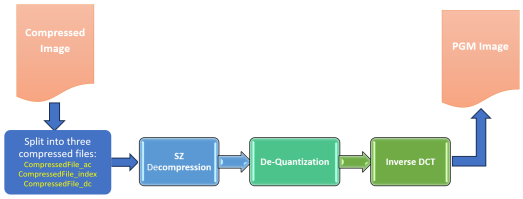
In current years, learning-based techniques have earned a significant concentration in the domain of lossy image compression. To develop efficient representations of images, these techniques are represented in neural network designs such as generative adversarial networks (GANs) [19], variational auto-encoders (VAEs) [20], convolutional auto-encoders (CAEs) [21], and super-resolution with denoising DL [22]. These techniques can perform better than conventional compression algorithms in terms of compression efficiency and visual quality preservation by utilizing machine learning. Additionally, studies have looked into lossy compression based on region of interest (ROI), in which the compression is tailored to emphasize the quality of imaging areas within the image. This focused approach is especially helpful in medical imaging applications where it's crucial to maintain the clarity and accuracy of the diagnostic data [23], [24].

III. The ELiCA Approach

The MIoT network consists of a high number of edge devices. These devices process and transmit multimedia data (images) to the data center, either via multi-hop or star wireless connections. The MIoT devices should compress multimedia data before transmitting it to the application server for decompression to recover visual content. Unfortunately, due to resource and computational limits, some of these MIoT devices are unable to execute lossy image compression. To reduce the needed resource overhead, MIoT devices should employ an effective data reduction approach with minimal memory and processing requirements in order to reduce the sent image size. As a result, it may control both the bandwidth and energy consumption required to transmit the images. This section will introduce the proposed efficient lightweight image compression approach (ELiCA) for the Multimedia Internet of Things in more detail. Figure 1 refers to the architecture of the proposed ELiCA.



(a) Image Compression Process



(b) Image Decompression Process

Fig. 1: The ELiCA Approach.

The ELiCA is inspired from JPEG compression with some modifications. The main modification is to replace the lossless Huffman encoding with lossy lightweight SZ compression. Algorithm 1 refers to the compression process of the ELiCA approach.

In Algorithm 1, the sensed image is partitioned into submatrices, or blocks, that are all the same size (8*8) as explained in step 1. In steps 2-4, the empty files are intialized. The steps 5-12, the DCT, quantization, and zigzag are applied for each block in the image [25]. In step 6, a frequency transformation is performed on every image block. This transformation breaks down each block into sub-bands of high, middle, and low frequencies, represented by DCT coefficients. This process yields a total of 64 coefficients for each block. Every coefficient obtained contains unique information about the converted signal. A quantization matrix is used in step 7 to quantize the DCT coefficients. The compression ratio (image quality) determines the quantization matrix. Many of the medium and high frequency AC coefficients will be disregarded in this step. The number of DCT coefficients that will be rounded to zeros is determined by the quantization and rounding step. In step 8, the zigzag is used to reorder the quantized DCT coefficients. For each block in the image, the first DCT coefficient is saved in DC file $file_{dc}$. The remaining DCT coefficients are saved in the AC file $file_{ac}$. Thus, DC comprises the fundamental lowest frequency. The DC coefficient and the first AC coefficient correspond to the low frequencies, whilst the remaining AC coefficients correspond to the medium and high frequencies. The indices of the blocks which are brought

Algorithm 1 *ELiCA Image Compression Algorithm*

Require: I : *sensed image*
Ensure: C*: the final compressed file*

1: Convert I into 8 x 8 Blocks (I_1,..., I_ρ) ; // ρ is the number of blocks
2: $file_{dc} \leftarrow []$; //$file_{dc}$ is empty file
3: $file_{index} \leftarrow []$; //$file_{index}$ is empty file
4: $file_{ac} \leftarrow []$; //$file_{ac}$ is empty file
5: **for** $j \leftarrow 1$ to ρ **do**
6: $\Psi \leftarrow DCT(I_j)$; //Ψ is the DCT coefficients vector
7: $\theta \leftarrow quantize(\Psi)$; //$\theta$ is the quantized DCT coefficients vector
8: $\delta \leftarrow Zigzag(\theta)$; //$\delta$ is the Reordered quantized DCT coefficients vector
9: $file_{dc} \leftarrow Append(\delta_1)$;
10: $file_{index} \leftarrow Append(GetBlockIndex(I_j))$
11: $file_{ac} \leftarrow Append((\delta_2, ..., \delta_\rho)$;
12: **end for**
13: $C \leftarrow []$; //$file_{dc}$ is empty file
14: $CompressedFile_{dc} \leftarrow SZ_Compressor(file_{dc})$;
15: $CompressedFile_{index} \leftarrow SZ_Compressor(file_{index})$
16: $CompressedFile_{ac} \leftarrow SZ_Compressor(file_{ac})$;
17: $C \leftarrow CompressedFile_{ac} + CompressedFile_{index} + CompressedFile_{ac}$;
18: return C;

by the GetBlockIndex() function refer to the positions of AC coefficients' locations and saved in the index file $file_{index}$. The steps 9-11 explain the result of zigzag process.

SZ is a lossy compression method with error bounds that is commonly employed by high-performance computers (HPCs) to efficiently process large volumes of data. The SZ algorithm has been implemented on various computing platforms, including CPU, GPU, and FPGA [26]. It can be used to compress many sorts of data, including single-precision and double-precision, as well as diverse shapes of scientific data. The initial code is designed for a 64-bit processor, and due to its extensive size, it requires modification to operate on IoT devices. This work introduces the SZ_compressor library, which provides a compact implementation of the SZ lossy compressor specifically designed for multimedia IoT devices. The adaptation involved adjusting the variables and pointers to be compatible with the compiler of multimedia IoT devices that use 32-bit pointers. Due to their limited storage capacity, multimedia IoT devices have undergone a process of removing any unnecessary components. Multiple buffers were minimised within the code to optimise its execution on multimedia IoT chips.

In Algorithm 2, the abbreviations QCF, LCF, and PNF stand for Quadratic-Curve Fitting, Linear-Curve Fitting, and Preceding Neighbour Fitting, respectively. SZ_compressor uses ZLib compressor for lossless compression of lossy compression outputs. In step 1, the algorithm allocates 2β bits of memory to record the predictability of each value. Next, the algorithm

Algorithm 2 SZ_Compressor

Require: $X = \{x_1, \ldots, x_\beta\}$: 1-D array data, β: size of array, ϵ: the relative error bound
Ensure: ν, α: compressed data, where ν and α are used to record the predictability of values and the unpredictable data respectively.

1: Construct an array of bits ν of β items, each one includes 2 bits
2: vrs $\leftarrow$ (Max(X_i) - Min(X_i)) // calculate the size of value range
3: **for** $i \leftarrow 1$ to β **do**
4: $Y_i^{PNF} \leftarrow Y_{i-1}$;
5: $Y_i^{LCF} \leftarrow 2Y_{i-1}$ - Y_{i-2} ;
6: $Y_i^{QCF} \leftarrow 3Y_{i-1}$ - $3Y_{i-2}$ + Y_{i-3} ;
7: BestSol $\leftarrow \text{argmin}_{M=\{Y_i^{PNF}, Y_i^{LCF}, Y_i^{QCF}\}}(|Y_i^M - X_i|)$
8: **if** $(|Y_i^{BestSol} - X_i|) satisfy \epsilon)$ **then**
9: **switch** Φ **do**
10: **case** PNF **do**
11: $\nu_i \leftarrow PrecedingNeighborFitting()$;
12: **case** LCF **do**
13: $\nu_i \leftarrow LinearCurveFitting()$;
14: **case** QCF **do**
15: $\nu_i \leftarrow QuadraticCurveFitting()$;
16: **end switch**
17: $Y_i \leftarrow Y_i^{bestsol}$ // store the predicted value for next prediction
18: **else**
19: $\nu_i \leftarrow UnpredictableData$;
20: Compress X_i using binary representation;
21: $\alpha_{k++} \leftarrow X_i^*$ // decompressed binary value of X_i;
22: $Y_i \leftarrow X_i^*$;
23: **end if**
24: **end for**
25: $return(\nu, \alpha)$;

calculates the magnitude of the range value vrs (refer to step 2), which can be used to compare the prediction errors to the bounds' error. The algorithm's phases (steps 3-24) entail individually scrutinising each value in the 1-D array X. More specifically, the programme will initially select the optimal model for curve fitting and store it in the parameter "bestsol" (refer to steps 4-7). It will next assess whether the prediction error meets the user's given relative error constraints. If the prediction error value falls below the user's specified relative error limits, the bestsol will be recorded in a bit-array ν (refer to steps 9-17). Otherwise, the most recent data value will be compressed by employing a binary-representation method. The time complexity of Algorithm 2 is $\mathcal{O}(\beta)$ while it needs $\mathcal{O}(\beta)$ of storage.

During the image decompression process in ELiCA (see Figure 1 b), the compressed image data file is partitioned into three compressed files: $CompressedFile_{dc}$, $CompressedFile_{index}$, and $CompressedFile_{ac}$. These compressed files are decompressed using our IoT SZ decompres-

sor. Then, the dezigzag process is applied to the output of the IoT SZ decompressor. The resulted data are de-quantized by consulting the quantization table. Subsequently, we execute the Inverse Discrete Cosine Transform (DCT) to restore the image to its original pixel values. Finally, the image back to its original format.

IV. Performance Evaluation

This section conducts multiple practical experiments using the Arduino IDE to assess the performance of ELiCA on the IoT microcontroller-based Portenta H7 Lite. The Arduino HAT Carrier Board is combined with the Portenta H7 Lite to use the image dataset stored on the SD card on the Arduino HAT Carrier Board.

We use two types of datasets when evaluating the proposed ELiCA. These datasets include images from the Kodak dataset [15] and the BOSSbase dataset [16], which were converted to PGM grayscale images. We crop the obtained images to reduce their size to 250x250, which is suitable for the microcontroller memory size. This paper evaluates the performance of the proposed ELICA in terms of compression ratio, PSNR, and SSIM.

A. Compression ratio evaluation

This experiment aimed to examine the impact of the dataset size on the compression ratio. Figure 2 displays the compression ratio for the proposed ELiCA compared with JPEG for different datasets and JPEG quality. The Compression Ratio (CR) is defined as follows:

$$CR = \frac{Orig_{ImgSize}}{Comp_{DataSize}} \quad (1)$$

where $Orig_{ImgSize}$ refers to the size of original image before compression, and $Comp_{DataSize}$ refers to the size of the data after using the compression approach.

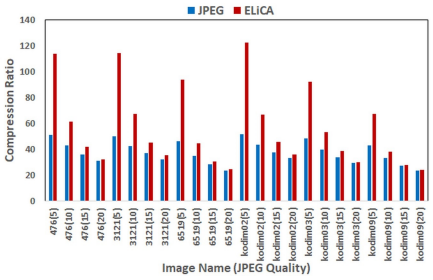


Fig. 2: Compression ratio

The findings depicted in Figure 2 demonstrate that the ELiCA algorithm achieves a better compression ratio across the IoT device compared with JPEG for different datasets and JPEG quality. It can be observed from the results in Figure 2 that the proposed ELiCA increases the compression ratio from 24.6 up to 122.3 compared to JPEG, that introduced a compression ratio from 23.6 up tp 51.8. The SZ compressor setting and the quality of the JPEG affect the compression ratio of the proposed ELiCA. Using low image quality in the JPEG and choosing an absolute error bound mode equal to 0E0 can achieve a high compression ratio. It is worth mentioning that we compress the AC and DC coefficients in the int16 type and the index coefficients in the int8 type to get the best compression ratio.

B. Peak Signal to Noise Ratio evaluation

This section evaluates the effectiveness of the proposed ELiCA in terms of the fidelity of reconstructed or compressed images compared to the original uncompressed version using the Peak Signal to Noise Ratio (PSNR). It offers a numerical gauge of the accuracy of the compressed images compared to the original data. Figure 3 refers to the PSNR for different datasets and JPEG quality.

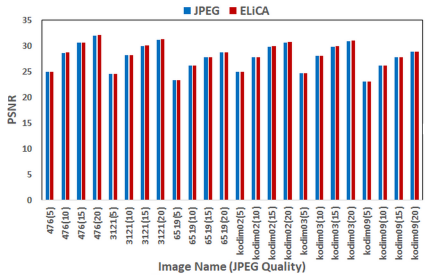


Fig. 3: PSNR

It can be seen from the results in Figure 3 that the PSNR values are similar or very close for both the proposed ELiCA and JPEG for different datasets and JPEG quality. The proposed ELiCA and JPEG introduced a PSNR from 23 up to 32 and from 23 up to 31, respectively. Hence, the ELICA introduces better performance compared with JPEG, where it provides a higher compression ratio while maintaining a PSNR equal to or slightly better than JPEG.

C. Structural Similarity Index Measure evaluation

This experiment compares the proposed ELiCA to JPEG in terms of the Structural Similarity Index Measure (SSIM) for different datasets and JPEG quality. This quantitative measure assesses the similarity between two images, taking into account both their luminance (brightness) and structural characteristics. Figure 4 refers to the SSIM of the ELiCA compared with JPEG for different datasets and JPEG quality.

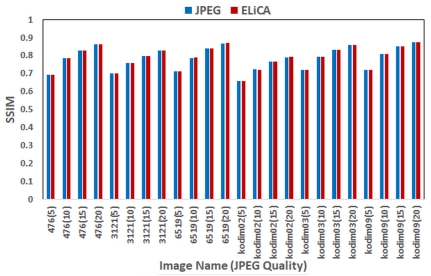


Fig. 4: SSIM

It can be noticed from the results in Figure 4 that the SSIM values are similar or very close for both the proposed ELiCA and JPEG for different datasets and JPEG quality. The ELiCA and JPEG presented a SSIM from 0.66 up to 0.87 and from 0.66 up to 0.87, respectively. Hence, even though ELICA introduces better performance in terms of compression ratio compared with JPEG, it still provides similar or very close SSIM values compared with JPEG.

D. Compression time

Figure 5 shows the compression time of ELiCA compared with JPEG for different datasets and JPEG quality. It can be

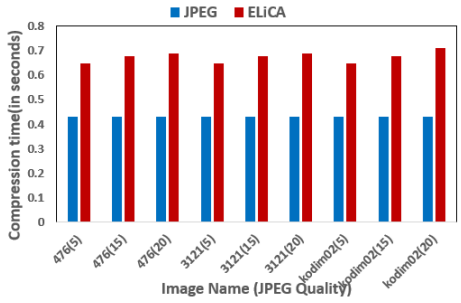


Fig. 5: Compression time

observed from the results in Figure 5 that ELiCA consumes a little bit more time in compressing the data compared with JPEG, but it introduced a higher compression ratio (see Figure 2).

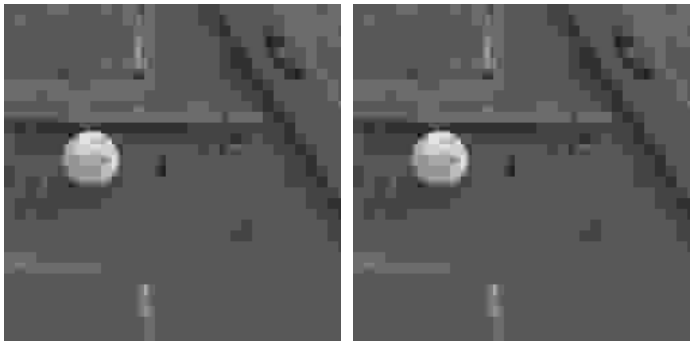
E. Compression Effect on Visual Image Quality

In this section, we examine the impact of compression on visual image quality, taking into account various datasets and JPEG quality. The compressed image quality improves as the compression ratio decreases, and vice versa. The specific multimedia IoT application targeting the JPEG image quality parameter determines its configuration. Increasing the JPEG image quality parameter results in a decrease in the compression ratio, and vice versa. Figures 6 and 7 refer to the reconstructed images using ELiCA and JPEG with kodak dataset and BOSSbase dataset, respectively.

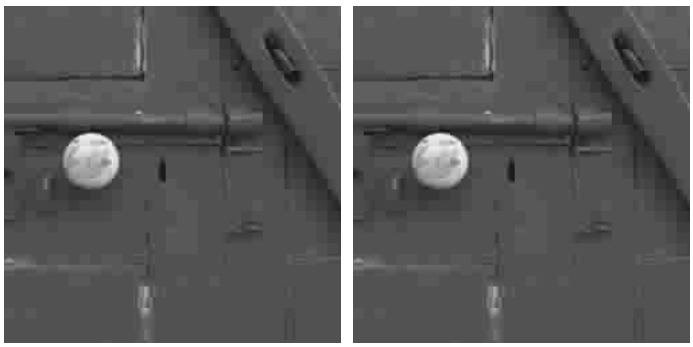
It can be observed from the results in Figures 6 and 7 that the visual image quality of the reconstructed image is improved when the JPEG image quality parameter is increased for both ELiCA and JPEG using different datasets. The proposed ELiCA introduced competitive results compared with JPEG in terms of visual image quality, but despite that, it outperformed JPEG from the compression ratio point of view.

V. Conclusion and Perspectives

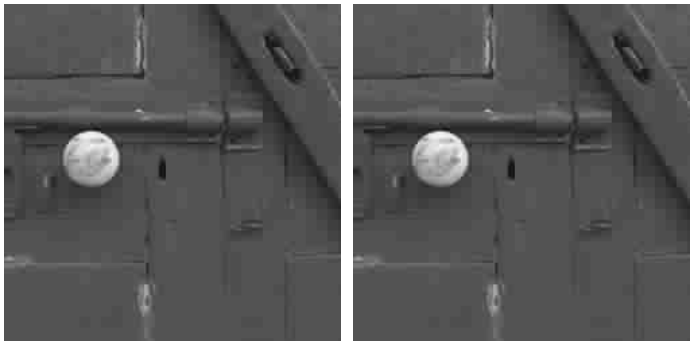
The Multimedia Internet of Things (MIoT) plays a significant role in a wide range of multimedia applications that can capture and send large numbers of photos or video sequences across the MIoT network. To adapt to the restricted resources of multimedia IoT devices, it is necessary to decrease the amount of multimedia data being transmitted while still ensuring that the accuracy of this data remains adequate on



(a) ELiCA:kodim02(5),CR=122.339 (b) JPEG:kodim02(5),CR=51.88



(c) ELiCA:kodim02(15),CR=45.933 (d) JPEG:kodim02(15),CR=37.888

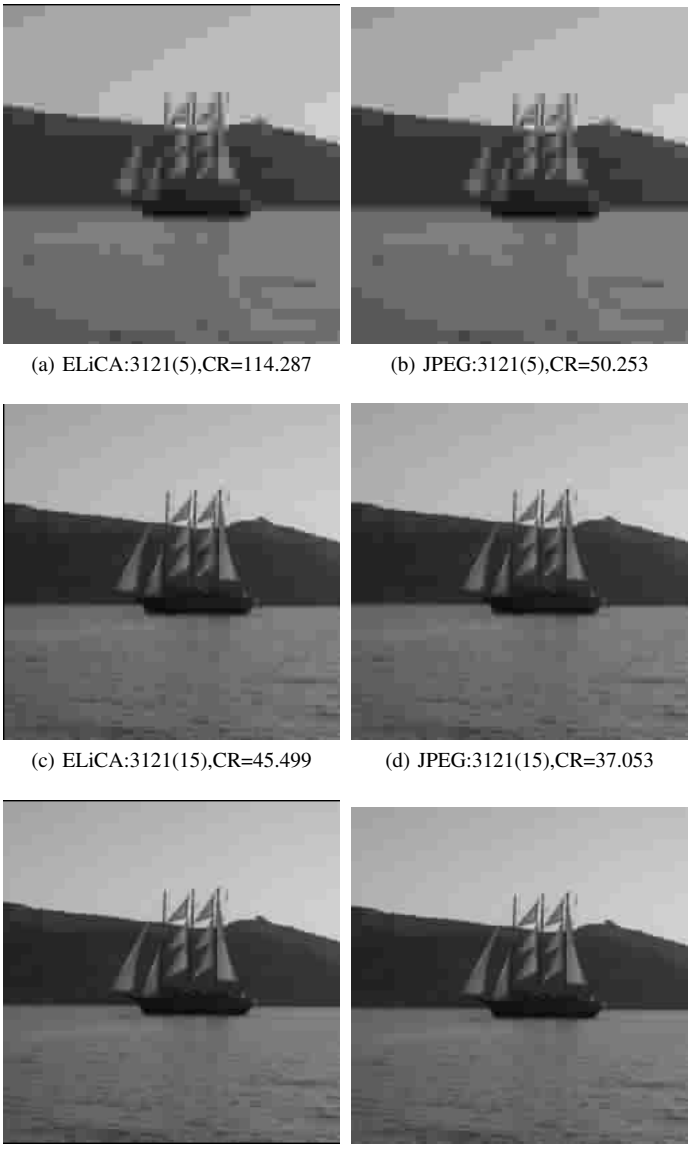


(e) ELiCA:kodim02(20),CR=35.908 (f) JPEG:kodim02(20),CR=33.147

Fig. 6: The reconstructed images using ELiCA and JPEG with Kodak24 dataset.

the receiving end. This can lead to energy and memory conservation, as well as enhanced performance of the MIoT network. This study presents a method called ELiCA (Efficient Lightweight Image Compression Approach) for compressing images in the context of MIoT. The goal of ELiCA is to reduce the size of transmitted images while maintaining image quality.

The proposed ELiCA starts partitioning the acquired image into blocks of the same size, and then applies the DCT on each block to obtain the DCT coefficients. We quantize the DCT coefficients and then reorder them using the zigzag method. Furthermore, we apply the SZ lossy compression technique to compress the output of the quantization step. The MIoT device will transfer the compressed file. We assess the efficacy of the proposed ELiCA by applying actual sensed photos to real IoT devices. The ELiCA outperforms JPEG in terms of



(a) ELiCA:3121(5),CR=114.287 (b) JPEG:3121(5),CR=50.253

(c) ELiCA:3121(15),CR=45.499 (d) JPEG:3121(15),CR=37.053

(e) ELiCA:3121(20),CR=35.419 (f) JPEG:3121(20),CR=32.158

Fig. 7: The reconstructed images using ELiCA and JPEG with BOSSbase dataset.

compression ratio, PSNR, and SSIM. We plan to enhance the decompressed images using deep neural networks. In addition, we plan to adapt the ELiCA to deal with compressing the video on the MIoT devices.

REFERENCES

[1] R. Hamza, K. Muhammad, Z. Lv, and F. Titouna, "Secure video summarization framework for personalized wireless capsule endoscopy," *Pervasive and Mobile Computing*, vol. 41, pp. 436–450, 2017.

[2] A. Celesti, A. Galletta, L. Carnevale, M. Fazio, A. Ĺay-Ekuakille, and M. Villari, "An iot cloud system for traffic monitoring and vehicular accidents prevention based on mobile sensor data processing," *IEEE Sensors Journal*, vol. 18, no. 12, pp. 4795–4802, 2017.

[3] E. De Buyser, E. De Coninck, B. Dhoedt, and P. Simoens, "Exploring the potential of combining smart glasses and consumer-grade eeg/emg headsets for controlling iot appliances in the smart home," 2016.

[4] Y. B. Zikria, M. K. Afzal, and S. W. Kim, "Internet of multimedia things (iomt): Opportunities, challenges and solutions," *Sensors*, vol. 20, no. 8, p. 2334, 2020.

[5] O. Ghorbel, W. Ayedi, M. W. Jmal, and M. Abid, "Images compression in wsn: Performance analysis," in *2012 IEEE 14th International Conference on Communication Technology*. IEEE, 2012, pp. 1363–1368.

[6] A. Kassie and M. Mohammed, "Privacy preservation of end-to-end communication for resource-constrained devices in iot," in *2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI)*. IEEE, 2018, pp. 1–9.

[7] S.-W. Lee and H.-Y. Kim, "An energy-efficient low-memory image compression system for multimedia iot products," *EURASIP Journal on Image and Video Processing*, vol. 2018, no. 1, p. 87, 2018.

[8] C. Ram and S. Panwar, "Performance comparison of high efficiency video coding (hevc) with h. 264 avc," in *2017 13th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*. IEEE, 2017, pp. 303–310.

[9] ISO/IEC, "Information technology – jpeg 2000 image coding system: Core coding system," ISO/IEC Standard 15 444–1, 2016.

[10] A. Zimbico, F. Schneider, and J. Maia, "Comparative study of the performance of the jpeg algorithm using optimized quantization matrices for ultrasound image compression," in *5th ISSNIP-IEEE Biosignals and Biorobotics Conference (2014): Biosignals and Robotics for Better and Safer Living (BRC)*. IEEE, 2014, pp. 1–6.

[11] ISO/IEC, ITU-T, "Iso/iec 14496-10, itu-t h.264 - h.264: Advanced video coding for generic audiovisual services," ISO/IEC Standard, ITU-T Standard 14 496–10, 2008.

[12] K. Sayood, *Introduction to data compression*. Morgan Kaufmann, 2017.

[13] E. Setyaningsih and A. Harjoko, "Survey of hybrid image compression techniques," *International Journal of Electrical and Computer Engineering*, vol. 7, no. 4, p. 2206, 2017.

[14] Y. Huang, B. Chen, J. Zhang, Q. Han, and S.-T. Xia, "Compressive sensing based asymmetric semantic image compression for resource-constrained iot system," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, 2022, pp. 877–882.

[15] Kodak, "Kodak photocd dataset," https://r0k.us/graphics/kodak/.

[16] Heyongchong, "Bossbase 256-4," https://www.kaggle.com/datasets/heyongchong/bossbase-256-4.

[17] F. Bellard, "Bpg image format." [Online]. Available: URLhttps://bellard.org/bpg,2015

[18] D. Kosolobov and O. Merkurev, "Optimal skeleton huffman trees revisited," in *International Computer Science Symposium in Russia*. Springer, 2020, pp. 276–288.

[19] E. Agustsson, F. Mentzer, M. Tschannen, L. Cavigelli, R. Timofte, L. Benini, and L. V. Gool, "Soft-to-hard vector quantization for end-to-end learning compressible representations," *Advances in neural information processing systems*, vol. 30, 2017.

[20] L. Theis, W. Shi, A. Cunningham, and F. Huszár, "Lossy image compression with compressive autoencoders," in *International conference on learning representations*, 2022.

[21] G. Toderici, D. Vincent, N. Johnston, S. Jin Hwang, D. Minnen, J. Shor, and M. Covell, "Full resolution image compression with recurrent neural networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017, pp. 5306–5314.

[22] H. N. Noura, J. Azar, O. Salman, R. Couturier, and K. Mazouzi, "A deep learning scheme for efficient multimedia iot data compression," *Ad Hoc Networks*, vol. 138, p. 102998, 2023.

[23] M. Kaur and V. Wasson, "Roi based medical image compression for telemedicine application," *Procedia Computer Science*, vol. 70, pp. 579–585, 2015.

[24] K.-H. Jin and et al., "Roi-based image compression for efficient medical image storage and transmission," *IEEE Journal of Biomedical and Health Informatics*, vol. 24, no. 12, pp. 3475–3485, 2020.

[25] G. K. Wallace, "The jpeg still picture compression standard," *Communications of the ACM*, vol. 34, no. 4, pp. 30–44, 1991.

[26] S. Di and F. Cappello, "Fast error-bounded lossy hpc data compression with sz," in *2016 ieee international parallel and distributed processing symposium (ipdps)*. IEEE, 2016, pp. 730–739.