LiLoCA: Lightweight Lossy Compression Approach for Edge-Enabled Multimedia IoT Applications

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

[1]*Université Marie et Louis Pasteur-UTBM-CIAD (UR 7533), Belfort, France*

[2] *Smart Edge Lab, Faculty of Computer Science, University of Applied Science and Arts, Dortmund, Germany*

[3] *CNRS, Institut FEMTO-ST, Université Marie et Louis Pasteur, Belfort, France*

[4]*Dept. of Information Networks, College of Information Technology, University of Babylon, Babylon, Iraq*

Emails: {sara.idrees, franck.gechter}@utbm.fr, {ali.idrees, rolf.schuster}@fh-dortmund.de, {joseph.azar, raphael.couturier}@univ-fcomte.fr

***Abstract*—The advancement of the Internet of Things (IoT) and communication technologies has given rise to the multimodal IoT. The multimedia IoT facilitates a wide range of multimedia applications that are capable of capturing and transmitting substantial quantities of images or video sequences across an IoT network towards the edge server. Given limited resources, multimedia IoT devices need to communicate a reduced amount of multimedia data while maintaining precision and decreasing the end-to-end latency. This can optimize energy consumption, memory utilization, and end-to-end latency while enhancing the performance of multimedia IoT networks. This paper presents a Lightweight Lossy Compression Approach (LiLoCA) for Edge-Enabled Multimedia IoT Applications. The suggested LiLoCA partitions the collected image into blocks of equal size and applies the Discrete Cosine Transform (DCT) to each block to get the DCT coefficients. Next, the DCT coefficients undergo quantization. The result of the quantization process is rearranged using the zigzag method. The output files of the zigzag are compressed using the Zlib lossless compression technique. The MIoT device will transmit the compressed file to the edge server. The effectiveness of the suggested LiLoCA is assessed based on real-sensed photos and IoT devices, and the results demonstrate the efficiency of the proposed LiLoCA in terms of compression ratio, Peak Signal-to-Noise Ratio (PSNR), and Structural Similarity Index Measure (SSIM) compared with some other exciting methods. Furthermore, LiLoCA decreased the end-to-end latency efficiently.**

***Index Terms*—Multimedia IoT, Edge computing, Zlib compression, Embedded IoT, DCT, Quantization, Lightweight lossy Data Compression.**

I. Introduction

The Multimedia Internet of Things (MIoT) is an advanced version of the traditional IoT framework that incorporates the transmission and analysis of multimedia data, including audio, video, and images, alongside sensor data. Various multimedia applications, including wireless endoscopies, traffic monitoring, and smart glasses, can be utilized for this purpose [1]–[3].

The MIoT devices utilized in these applications encounter substantial obstacles. Their processing capabilities, memory capacity, and battery longevity are constrained by restricted resources. The MIoT Network's bandwidth can be heavily strained by the substantial volume of multimedia data, especially in time-sensitive situations like remote surgery or autonomous vehicles. Furthermore, the processing and transmission of substantial volumes of MIoT data result in heightened latency inside the MIoT network [4]. Hence, it is imperative to minimize the transmission of redundant MIoT data. Using image compression techniques can decrease the storage space needed to store MIoT data and prolong the battery life of low-power MIoT devices [5].

Typically, it is believed that IoT devices possess sufficient memory and processing capabilities to execute complicated and efficient algorithms [6]. However, in order to achieve cost-effective infrastructures, IoT platforms generally rely on resource-constrained microprocessors with limited storage capacity of just a few kilobytes [6]. As a result, it is generally advisable to utilize low-complexity and low-memory compression techniques, unless they are required.

Several state-of-the art efficient image compression techniques exist, including HEVC (Ram, 2017), JPEG2000 (ISO15444-1), baseline JPEG (Zimbico, 2014), and H.264 (ISO14496-10). Nevertheless, the memory requirements of these compression techniques do not correspond to the memory limitations of many IoT devices.

Image compression is categorized into two main types: lossless and lossy. The classification is based on whether the compressed image can be fully restored to its original form with mathematical precision [7].

Lossy compression techniques differ from lossless approaches in that they deliberately introduce controlled distortion in the rebuilt image, which enables far higher compression ratios. The balance between image quality and compression efficiency has been a fundamental focus of study in the field of image processing [8].

Although there are numerous lossy image compression methods available, the majority of them are unsuitable for devices that have limited storage and computational capabilities [9]. Therefore, it is necessary to consider the balance between computational complexity and the compression ratio that can be achieved when dealing with devices that have limited resources.

This paper presents the following contributions:

- we propose an efficient lightweight lossy compression

approach (LiLoCA) for Multimedia Internet of Things networks. LiLoCA partition the input image into equal size blocks and applies the DCT, quantization, and zigzag for each block. Finally, LiLoCA applies the zlib compressor to the resulting files to reduce the size of compressed MIoT data.

- The proposed LiLoCA was evaluated using real experiments that were implemented on real Portenta H7 Lite microcontroller with real image datasets from kaggle DIV2K and Car License Plate Detection dataset. 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

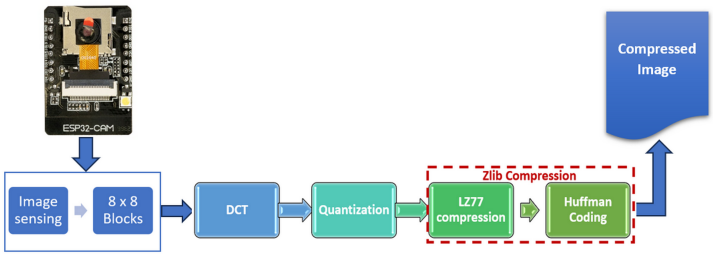
Image compression can be divided into two categories: lossless and lossless methods. Lossless compression techniques prioritize maintaining the integrity of the optical information, assuring that the extracted image is identical to the original image. Conversely, lossy compression distorts some of the image content to achieve a higher compression ratio, making it better suited to reduce the amount of bandwidth used in networks Compression ratio is often related to the quality of the image; it is the opposite, i.e., a lower compression ratio makes the image feel better quality But a higher compression ratio lowers the image quality Over time, a wide variety of algorithms and techniques have been developed and implemented for imaging loss.

Transformation-based algorithms are widely used for lossy compression [10]. These techniques involve transforming the image data to a different domain, such as discrete cosine transform (DCT) or discrete wavelet transform (DWT) followed by compressing the quantized data using quantization and reversible coding algorithms such as Huffman coding [**?**].

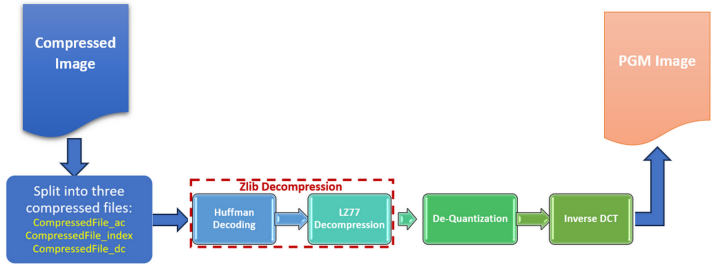
In recent years, learning-based algorithms have received dramatic attention in image loss compression. Neural network structures such as generation anti-networks (GANs) [11] , fractional auto-encoders (VAEs) [12], convolutional auto-encoders (CAEs) Other techniques have been developed to achieve lossless compression, including Huffman encoding, run-length encoding, and Lempel-Ziv-Welch encoding. The evaluation of these strategies is conducted by analyzing their compression ratios and effectiveness across many criteria [13]. Advancements in the field have introduced novel methods such as 'bits back with asymmetric numeral systems' (BB-ANS), which utilize latent variable models to achieve optimal compression rates. These approaches have demonstrated significant improvements in compressing complex data, such as images. [14]. In addition, the practical usefulness of lossless algorithms in medical diagnostics is demonstrated by specific applications, such as real-time compression of photoplethysmographic data. These methods achieve large compression ratios while minimizing distortion [15].

III. The LiLoCA approach

The MIoT network consists of a high number of MIoT devices. These devices process and transmit multimedia data (images) to the edge devices then to the cloud data center. The MIoT devices should compress multimedia data before transmitting it to the edge devices 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 to the edge devices. As a result, it may control both the bandwidth and energy consumption required to transmit the images to these devices. This section will introduce the proposed efficient lightweight Lossy compression approach (LiLoCA) for the Multimedia Internet of Things network in more detail. Figure 1 refers to the architecture of the proposed LiLoCA.



(a) Image Compression Process



(b) Image Decompression Process

Fig. 1: The LiLoCA Approach.

Algorithm 1 refers to the compression process of the LiLoCA 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 [16]. 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

Algorithm 1 *LiLoCA Image Compression Algorithm*

Require: IMG : *Captured image*
Ensure: CM*: the final compressed file*
1: Transform IMG into 8 x 8 Blocks (IMG_1,..., IMG_{NB}) ; // NB is the number of blocks
2: $F_{dc} \leftarrow []$; //F_{dc} is empty file
3: $F_{index} \leftarrow []$; //F_{index} is empty file
4: $F_{ac} \leftarrow []$; //F_{ac} is empty file
5: **for** $j \leftarrow 1$ to NB **do**
6: $DCT_{coeff} \leftarrow DCT(IMG_j)$; //$DCT_{coeff}$ is the DCT coefficients vector
7: $QDCT_{coeff} \leftarrow quantize(DCT_{coeff})$; //$QDCT_{coeff}$ is the quantized DCT coefficients vector
8: $RQDCT_{coeff} \leftarrow Zigzag(QDCT_{coeff})$; //$RQDCT_{coeff}$ is the Reordered quantized DCT coefficients vector
9: $F_{dc} \leftarrow Append(RQDCT_{coeff_1})$;
10: $F_{index} \leftarrow Append(FindBlockIndex(IMG_j))$
11: $F_{ac} \leftarrow Append((RQDCT_{coeff_2}, ..., RQDCT_{coeff_{NB}})$;
12: **end for**
13: $CM \leftarrow []$; //CM is empty file
14: $CompressedF_{dc} \leftarrow Zlib_Compressor(file_{dc})$;
15: $CompressedF_{index} \leftarrow Zlib_Compressor(file_{index})$
16: $CompressedF_{ac} \leftarrow Zlib_Compressor(file_{ac})$;
17: $CM \leftarrow CompressedF_{ac} + CompressedF_{index} + CompressedF_{ac}$;
18: return CM;

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 F_{dc}. The remaining DCT coefficients are saved in the AC file F_{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 by the FindBlockIndex() function, refer to the positions of AC coefficients' locations and saved in the index file F_{index}. The steps 9-11 explain the result of zigzag process.

Zlib is a lossless compression technique utilised for the compression and decompression of data. The DEFLATE algorithm is used, incorporating the LZ77 compression technique (a sliding window method) and Huffman coding (a form of entropy encoding). The zlib presents a compression approach that guarantees no data is lost during the compression process, allowing the compressed data to be completely restored to its original form [17]–[19]. The Zlib compressor in each step of 14-16 in Algorithm 1 will be fed by the appropriate data to compress it. First, Zlib will use the LZ77 Compression that consists of three steps: sliding window, finding matches, and literal data. The LZ77 operates by systematically analysing the incoming data through the utilisation of a sliding window. The window is typically composed of two components: a search buffer and a lookahead buffer. The method searches for byte sequences (substrings) in the lookahead buffer that correspond to sequences in the search buffer (located earlier in the data stream). Upon finding a match, it is substituted with a reference to the preceding instance, represented by a pair of values: distance and length. If a match is not found, the data is encoded exactly as it is, without any modifications. In the Huffman Coding of the Zlib compression consists of threes steps: frequency analysis, Creating a Huffman tree, and encoding. Frequency analysis is achieved on the data after it has been processed by the LZ77 algorithm. The produced sequence, which includes both literals and pointers, is then compressed using Huffman coding. The process of creating a Huffman tree involves constructing a tree structure using the repetition count of each byte (or symbol) included in the given data. Shorter codes are given to more frequent bytes, whereas longer codes are allocated to less frequent bytes. Finally, the Huffman tree encodes the data, producing the compressed output. The decompression process by the LiLoCA decompressor will implement the Huffman decoding, LZ77 decompression, de-quantization, and inverse DCT to produce the original data. This work introduces the LiLoCA compressor library, which is specifically designed for multimedia IoT devices.

1) Comparison results: : This experiment will evaluate and contrast our suggested lossy compression approach with several established lossy compression techniques documented in the literature, such as E-ABCS [**?**] , CMT-ABCS [**?**], WDR [**?**]. The practical experiments are conducted under the Linux operating system.

Table I shows the compressed image size and space savings for several medical images, such as CT Scan , MRI, and X-ray for several sampling rates and compression quality factors. The space-saving is evaluated according to the following equation:

$$\text{Saving Space}(\%) = \left(1 - \frac{\text{Comp}_{\text{Img}}}{\text{Org}_{\text{Img}}}\right) \times 100 \qquad (1)$$

Where Comp_{Img} represents the size of the compressed medical image and Org_{Img} represents the size of the original medical image.

TABLE I: Saving space comparison for different lossy methods.

		original img size (byte) = 77900	original img size (byte) = 63500	original img size (byte) = 58000			
Methods	SR/QF	CT Scan	MRI	X-ray	CT Scan	MRI	X-ray
	SR	comp size			Saving space %		
E-ABCS	0.1	34587.6	27750	24882	55.6	56.3	57.1
CMT-ABCS	0.1	25473.3	19749	17458	67.3	68.9	69.9
WDR	0.1	36690.9	28258	25172	52.9	55.5	56.6
E-ABCS	0.3	46584.2	36894	32828	40.2	41.9	43.4
CMT-ABCS	0.3	31082.1	23686	21170	60.1	62.7	63.5
WDR	0.3	47908.5	37846	33756	38.5	40.4	41.8
E-ABCS	0.5	51803.5	41021	37004	33.5	35.4	36.2
CMT-ABCS	0.5	32873.8	25337	22388	57.8	60.1	61.4
WDR	0.5	54452.1	42672	38454	30.1	32.8	33.7
	QF						
LiLoCA	75	7183	6817	6635	90.77	89.26	88.56
LiLoCA	85	9092	8729	9555	88.32	86.25	83.52
LiLoCA	93	12197	11458	14363	84.34	81.95	75.23

Table II presents a comparison of SSIM and PSNR across various lossy methods. The data in Table II indicates that

the suggested LiLoCA compression method surpasses other approaches regarding PSNR and SSIM metrics.

TABLE II: SSIM and PSNR comparison for different compression methods.

		SSIM			PSNR		
Method	SR/QF	CT Scan	MRI	X-ray	CT Scan	MRI	X-ray
E-ABCS	0.1	0.63	0.71	0.71	27.57	59.01	35.19
CMT-ABCS	0.1	0.91	0.89	0.96	36.82	34.64	44.38
WDR	0.1	0.65	0.69	0.80	26.43	47.81	39.24
E-ABCS	0.3	0.86	0.96	0.97	38.95	42.02	42.04
CMT-ABCS	0.3	0.95	0.97	0.98	43.41	43.99	45.20
WDR	0.3	0.93	0.81	0.91	27.10	48.82	40.55
E-ABCS	0.5	0.94	0.96	0.98	43.67	45.76	44.72
CMT-ABCS	0.5	0.96	0.98	0.98	46.61	48.69	46.27
WDR	0.5	0.93	0.96	0.97	41.40	46.52	43.80
QF							
LiLoCA	75	0.96	0.97	0.88	38.07	36.34	34.38
LiLoCA	85	0.93	0.93	0.90	39.96	38.43	35.52
LiLoCA	93	0.98	0.99	0.93	42.97	42.97	37.13

IV. Performance Evaluation

This section carries out various practical experiments utilizing the Arduino Integrated Development Environment (IDE) to evaluate the efficiency of LiLoCA on the Internet of Things (IoT) microcontroller-based Portenta H7 Lite. The Arduino HAT Carrier Board is utilized in conjunction with the Portenta H7 Lite to access the picture dataset stored on the SD card of the Arduino HAT Carrier Board. Figure 2 depicts the integration of the IoT microcontroller-based Portenta H7 Lite with an Arduino HAT Carrier Board.

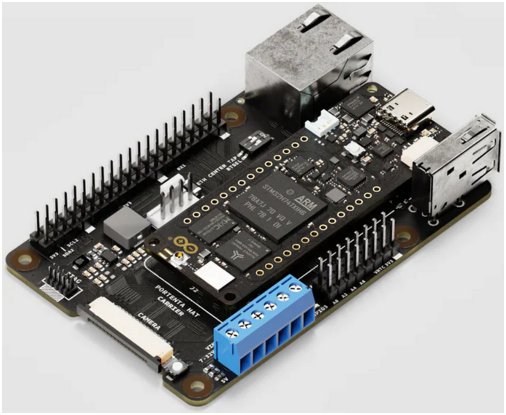


Fig. 2: Portenta H7 Lite combined with Arduino HAT Carrier Board

When analyzing the proposed LiLoCA, we utilize two distinct types of datasets. The datasets consist of images obtained from the DIV2K and Car License Plate Detection dataset. The images were transformed into PGM grayscale format. We resize the acquired images to dimensions of 256x256 in order to decrease their size, which is compatible with the memory capacity of the microcontroller. This section assesses the efficiency of the proposed LiLoCA by measuring its compression ratio, PSNR (Peak Signal-to-Noise Ratio), and SSIM (Structural Similarity Index).

A. Compression ratio

This experiment intended to investigate the effect of dataset size on compression ratio. Figure 3 shows the compression ratio of the proposed LiLoCA versus JPEG for various datasets and JPEG quality. The Compression Ratio (CR) is defined as the following:

$$CR = \frac{Orig_{ImgSize}}{Comp_{DataSize}} \tag{2}$$

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.

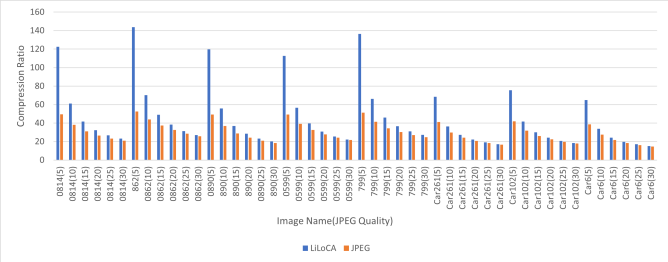


Fig. 3: Compression ratio

The findings depicted in Figure 3 demonstrate that the LiLoCA 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 3 that the proposed LiLoCA increases the compression ratio from 15.276 up to 143.752 compared to JPEG, which introduced a compression ratio from 14.75 up tp 52.651.

B. Peak Signal to Noise Ratio evaluation

This section reviews the efficacy of the proposed LiLoCA by measuring the fidelity of reconstructed or compressed images in comparison to the original uncompressed version, using the Peak Signal to Noise Ratio (PSNR). It provides a quantitative measure of the fidelity of the compressed images in relation to the original data. Figure 4 refers to the PSNR for different datasets and JPEG quality.

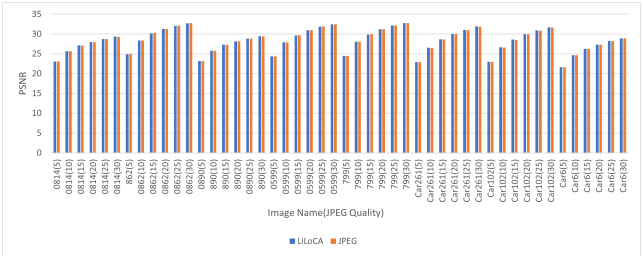


Fig. 4: PSNR

It can be seen from the results in Figure 4 that the PSNR values are similar or very close for both the proposed LiLoCA and JPEG for different datasets and JPEG quality. The proposed LiLoCA and JPEG introduced a PSNR from 21.62 up to 32.74 and from 21.62 up to 34.72, respectively. Hence, the

LiLoCA 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 evaluates the performance of LiLoCA, a suggested method, in comparison to JPEG, using the Structural Similarity Index Measure (SSIM) as the metric. The evaluation is conducted on several datasets and varying levels of JPEG quality. This quantitative metric evaluates the resemblance between two photographs by considering their luminance (brightness) and structural attributes. Figure 5 refers to the SSIM of the LiLoCA compared with JPEG for different datasets and JPEG quality.

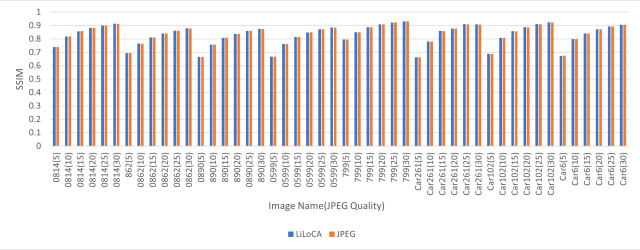


Fig. 5: SSIM

It can be noticed from the results in Figure 5 that the SSIM values are similar for the proposed LiLoCA and JPEG for different datasets and JPEG quality. The LiLoCA and JPEG presented a SSIM from 0.66 up to 0.93 and from 0.66 up to 0.93, respectively. Hence, even though LiLoCA introduces better performance in terms of compression ratio compared with JPEG, it still provides similar SSIM values compared with JPEG.

D. Compression time

Figure 6 shows the compression time of LiLoCA compared with JPEG for different datasets and JPEG quality. It can be

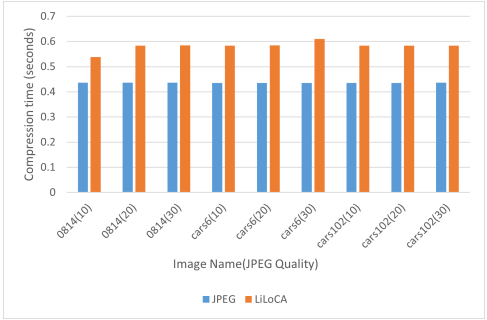


Fig. 6: Compression time

observed from the results in Figure 6 that LiLoCA consumes a little bit more time in compressing the data compared with JPEG, but it introduces a higher compression ratio (see Figure 3).

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 7 and 8 refer to the reconstructed images using LiLoCA and JPEG with DIV2K dataset and Car License Plate Detection dataset, respectively.



(a) LiLoCA:890(5),CR=119.837



(b) JPEG:890(5),CR=49.212



(c) LiLoCA:890(15),CR=36.993



(d) JPEG:890(15),CR=28.915



(e) LiLoCA:890(30),CR=20.108



(f) JPEG:890(30),CR=18.486

Fig. 7: The reconstructed images using LiLoCA and JPEG with DIV2K dataset.

It can be observed from the results in Figures 7 and 8 that the visual image quality of the reconstructed image is improved when the JPEG image quality parameter is increased for both LiLoCA and JPEG using different datasets. The



(a) LiLoCA:car261(5),CR=68.425 (b) JPEG:car261(5),CR=41.098

(c) LiLoCA:car261(15),CR=27.222 (d) JPEG:car261(15),CR=24.197

(e) LiLoCA:car261(30),CR=17.237 (f) JPEG:car261(30),CR=16.786

Fig. 8: The reconstructed images using LiLoCA and JPEG with Car License Plate Detection dataset.

proposed LiLoCA 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.

F. End-to-End Latency

In this section, we will show the effect of the proposed LiLOCA approach on the end-to-end latency. Suppose we consider a scenario in which a smart traffic camera (IoT device) grasps a 2 MB image and transmits it to a proximate edge server for license plate recognition. The server analyses the image and returns a brief result (e.g., identified plate number).

$$T_{\text{total}} = T_{\text{compress}} + T_{\text{Trans}}^{\text{Uplink}} + T_{\text{prop}}^{\text{RTT}} + T_{\text{queue}} + T_{\text{compress}} + T_{\text{proc}} + T_{\text{tx}}^{\text{Downlink}}$$

$$T_{\text{Trans}}^{\text{Uplink}} = \frac{UplinkDataSize}{UplinkBandwidth}$$

$$T_{\text{prop}}^{\text{RTT}} = 2 * \frac{DistancetoEdgeServer}{PropagationSpeed}$$

$$T_{\text{queue}} = QD_{Uplink} + QD_{Downlink}$$

$$T_{\text{tx}}^{\text{Downlink}} = \frac{DownlinkDataSize(Result)}{DownlinkBandwidth}$$

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 the receiving end. This can lead to energy and memory conservation, as well as enhanced performance of the MIoT network. This article presents an Efficient Lightweight Lossy Compression Approach (LiLoCA) for compressing images in the context of MIoT. The goal of LiLoCA is to reduce the size of transmitted images while maintaining image quality. The proposed LiLoCA 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 Zlib lossless compression technique to compress the output of the quantization step. The MIoT device will transfer the compressed file to the edge device. We assess the efficacy of the proposed LiLoCA by applying actual sensed photos on real IoT devices. The LiLoCA outperforms JPEG in terms of compression ratio, PSNR, and SSIM. In future, we plan to enhance the decompressed images using deep neural networks. In addition, we plan to adapt the LiLoCA 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] K. Sayood, *Introduction to data compression*. Morgan Kaufmann, 2017.
[8] 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.
[9] 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.
[10] F. Bellard, "Bpg image format." [Online]. Available: URLhttps://bellard.org/bpg,2015
[11] 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.
[12] L. Theis, W. Shi, A. Cunningham, and F. Huszár, "Lossy image compression with compressive autoencoders," in *International conference on learning representations*, 2022.
[13] U. Ijaz, A. Ijaz, A. Iqbal, F. Gillani, and M. Hayat, "Comparative analysis of lossless image compression algorithms," 2023.
[14] J. Townsend, "Lossless compression with latent variable models," *arXiv preprint arXiv:2104.10544*, 2021.
[15] R. Gupta, "Lossless compression technique for real-time photoplethysmographic measurements," *IEEE Transactions on Instrumentation and Measurement*, vol. 64, no. 4, pp. 975–983, 2014.
[16] G. K. Wallace, "The jpeg still picture compression standard," *Communications of the ACM*, vol. 34, no. 4, pp. 30–44, 1991.
[17] P. Deutsch, "Rfc1951: Deflate compressed data format specification version 1.3,rfc 1951," Available: https://tools.ietf.org/html/rfc1951., 1996.
[18] P. Deutsch and J.-L. Gailly, "Zlib compressed data format specification version 3.3,rfc 1950," Available: https://tools.ietf.org/html/rfc1950., 1996.
[19] J. Gailly and M. Adler, "zlib: A massively spiffy yet delicately unobtrusive compression library, version 1.2.11," Available: https://zlib.net., 1996.