A Deep Learning Scheme for Efficient Multimedia IoT Data Compression

Hassan N. Noura[1], Joseph Azar[1], Ola Salman[2], Raphaël Couturier[1], and Kamel Mazouzi[1]

[1]Univ. Bourgogne Franche-Comté (UBFC), FEMTO-ST Institute, France
[2]American University of Beirut, Beirut, Lebanon

***Abstract*—Multimedia Internet of Things (MIoT) devices and networks will face many power and communication overhead constraints given the volume of multimedia sensed data. One classic approach to overcoming the difficulty of large-scale data is to use lossy compression. However, current lossy compression algorithms require a limited compression rate to maintain acceptable perceived image quality. This is commonly referred to as the image quality-compression ratio trade-off. Motivated by current breakthroughs in computer vision, this article proposes recovering high-quality decompressed images at the application server level using a deep learning-based super-resolution model. As a result, this paper proposes ignoring the trade-off between image quality and size and increasing the reduction size further by using a lossy compressor with downscaling to conserve energy. The experimental study demonstrates that the proposed technique effectively improves the visual quality of compressed and downscaled images. The proposed solution was evaluated on resource-constrained microcontrollers. The obtained results show that the transmission latency and energy consumption can be decreased by up to 10% compared to conventional lossy compression techniques.**

***Index Terms*—Multimedia data compression, Deep learning image super-resolution, Visual degradation, Communication overhead, Multimedia IoT devices.**

I. Introduction

Nowadays, IoT is witnessing an extensive development with the emergence of new wireless technologies and new types of devices. In this context, the availability of cheaper hardware, as CMOS cameras and microphones, has enabled the development of large-scale Multimedia IoT (MIoT) networks [1], [2]. Therefore, a set of IoT applications have been developed, requiring the transmission of a large amount of multimedia data such as images, audios, or videos.

A. Problem Formulation

However, multimedia data is more scalable than standard scalar data and can be delivered with real-time limitations [3]. The transmission of such bulky data requires high bandwidth at the network level and high energy consumption, memory, and computational power processing at the device level [4]. Imposing real-time constraints with the limited bandwidth networks and limited IoT devices hinder the wide deployment and adoption of some MIoT applications [5].

As a result, there is an increasing need to improve the MIoT transmission task's efficiency and reliability. Data compression is an essential function for reducing the size of transmitted data. At the time, the standard JPEG [6] and BPG [7] are used extensively in academic and industrial image compression projects. For these algorithms, the compression efficiency can be increased by gradually decreasing the compressed multimedia data, hence lowering communication, computation, and time overhead. This can be accomplished by using a high compression ratio. This, however, comes at the expense of content quality. Thus, with existing compression algorithms, preserving high image quality impose a variety of constraints on the operation of MIoT devices, including limitations on processing, memory, and/or energy consumption. Thus, new data processing algorithms are required to maximize the trade-off between the size of transmitted data (or compression ratio) and the content quality.

Deep Learning (DL) was recently applied to numerous scientific domains such as security [8], medical [9], smart city [10], social network analysis [11], network computing [12] in general but also into many other scientific domains. In the domain of computer vision, DL allows to solve difficult tasks that were previously unsolvable using conventional machine learning methods [13], [14]. DL has been used in this context for image denoising and super-resolution [15]-[18], which can help to mitigate the effects of severe visual degeneration. Additionally, DL has been utilized recently for applications such as image recovery in conjunction with multi-view approaches [19], depth image denoising [20], [21], image captioning [22], and age invariant face recognition [23].

B. Motivation

Deep learning has revolutionized computer vision in a matter of a few years. Prior to the hundreds of papers in computational vision, notably in image reconstruction and super-resolution, the typical method to image compression was to consider the trade-off between image quality and compression ratio. Motivated by recent improvements in super-resolution, this article proposes to push the compressor performance of existing compression techniques such as JPEG and BPG beyond this trade-off and attain a maximum compression ratio. This is now achievable because super-resolution at the gateway level can be used to repair the compressor's damage and restore high image quality.

C. Contributions

To the best of our knowledge, this is the first work designing a DL-based solution to handle the MIoT intrinsic limitations. This approach integrates the data reduction at the MIoT devices (by applying data compression with a maximal compression rate) with the visual content enhancement and super-resolution at the application server(s) (by employing a DL model). It should be noted here that any denoising & super-resolution DL model that can achieve good results, can be applied [18].

This paper presents two variants of the data reduction solution. While both variants employ data compression, the second variant applies the down-scaling operation to reduce the image size before compression, and consequently, more data reduction can be achieved compared to the first variant at the cost of higher visual degradation.

Mainly, the proposed approach is designed to satisfy the following objectives:

1) Reduce the quantity of data communicated in order to minimize communication overhead and latency, as well as transmission energy consumption. This is accomplished by increasing the compression ratio beyond the trade-off between data reduction and image quality.
2) Preserve (or enhance) the perceived visual quality of the heavily compressed images. This is accomplished by proposing a flexible denoising and super-resolution solution capable of working with a variety of image quality (or compression ratios) and downscaling sizes, depending on the requirements of MIoT devices and applications.

D. Organization

The rest of this paper is organized as follows: Section II reviews the related work, especially the recent DL denoising and super-resolution approaches. Then, the proposed system model is presented and described in Section III. Then, Section IV details the experimental setup and the obtained results. Section V discusses the obtained results and presents future research directions. Finally, Section VI concludes the paper.

II. Background & Related Work

In this section, we start by briefly describing the well-known image compression algorithms, which are JPEG and BPG. After this, a set of recent DL-based image denoising and super-resolution approaches are discussed. Finally, the related work to applying DL for compressed images restoration is reviewed.

A. Image Compression

The image compression techniques can be divided into two classes: lossless and lossy (as illustrated in Fig. 1). The lossless compression techniques preserve better the visual content compared to the lossy ones since the decompressed image with lossless compression is the same as the original one. However, the lossless compression results in larger compressed data size compared to the lossy one. In general, the lossy compression is used to reduce the bandwidth overhead in existing networks. In fact, the compression ratio is inversely proportional to the image quality. Using a low compression ratio results into a high perceived image quality and vice versa.

Existing lossy compression techniques convert data from the spatial to the frequency domain by applying a spatial-frequency transformation such as the Discrete Cosine Transform (DCT2) used in JPEG [6] and BPG or Discrete Wavelet Transform (DWT2) in JPEG 2000 [24]. On the other hand, DCT2 is also used in the well-used lossy video compression standards such as MPEG 1/2, AVC/H.264, MPEG-4, and HEVC. After the spatial-frequency transformation, the frequency values are quantized (according to the frequency level) and ordered, then they get through the Huffman or arithmetic encoding process. Given that the human eye is good in remarking changes in low frequency regions while disregarding the changes in the busy patterns regions, the lossy compression techniques represents the less important data with the high frequency components, while the most important information are represented by the lower frequencies. Consequently, the lossy compression targets to eliminate more data from the high frequency components than from the low frequency ones. Therefore, the frequency coefficients have different importance levels and consequently they are quantized according to their importance and the desired compression ratio. This can lead to different visual effects. A high compression ratio means that only high and middle-frequency coefficients are quantized more coarsely compared to low-frequency coefficients.

In the following, JPEG and BPG image compression algorithms will be described.

1) JPEG Compression Standard: JPEG [6] is a lossy image compression standard that was developed by the Joint Photographic Experts Group (JPEG) committee. It is based on the concept of frequency transform coding, where DCT is used as frequency transformation on each image block (8×8).

The JPEG compression process can be divided into five main steps, which are:

1) **Pre-processing**: In this step, a color transformation is applied to convert the pixels of the R (Red), G (Green), and B (Blue) components into Y (Luminance), C_b (Blue-difference Chroma), and C_r (Red-difference Chroma) components. Then, the chroma sub-sampling process is applied.
2) **Discrete Cosine Transform (DCT)**: After preprocessing, the output matrices are divided into blocks (sub-matrix) of equal size (e.g $8\times$ 8). Then, a frequency transformation is applied to each image block. This transformation decomposes each block into high, middle, and low-frequency sub-bands (DCT coefficients), resulting into 64 coefficients for each block. Each obtained coefficient carries distinct information of the transformed signal.

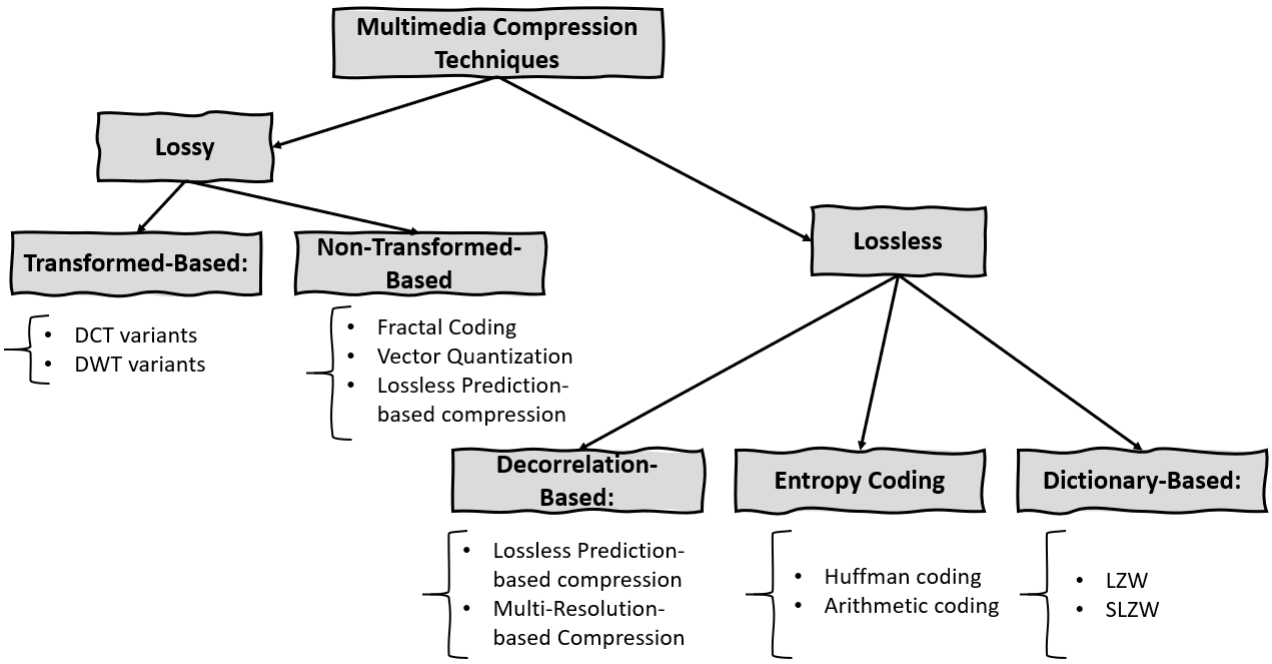


Fig. 1: Taxonomy of multimedia compression

The first DCT coefficient is called the DC coefficient, which carries the average intensity of the transformed block (64 elements). The remaining DCT coefficients are called AC coefficients. Therefore, DC consists of the most important lowest frequency. DC and first AC coefficients represent the low frequencies, while other AC coefficients represent middle and high frequencies.

3) **Quantization**: In this step, the DCT coefficients are quantized using a quantization matrix. The quantization matrix depends on the compression ratio (image quality). In this step, many of the AC coefficients (middle and high frequencies) will be ignored. The quantization and rounding step determines the number of DCT coefficients that will be rounded to zeros.
4) **Zigzag and run-length ordering**
5) **Encoding**: Finally, each compressed block is encoded using the Huffman entropy encoding.

2) BPG Compression Standard: The Better Portable Graphics (BGP) is an image format, presenting several advantages compared to JPEG [7], [25]. It achieves a lower visual degradation compared to JPEG, with the same compression ratio. In addition, BPG can be applied for lossless compression, provides the animation, and supports multiple color spaces (gray-scale, YCbCr, RGB, YCqCo) as well as chroma formats.

The BPG video encoder is based on HEVC [26], which is the successor of H.264/AVC. It is a well-known video compression standard [27]. HEVC presents good compression efficiency and it is considered as a major advance in the video compression domain. Like JPEG, BPG uses an 8×8 block as the basic coding unit, and DCT as the frequency transformation mechanism. Furthermore, as BPG is based on HEVC still-picture coding, also larger block sizes (i.e., size of Coding Unit (CU)) should be possible. Besides, BPG can use the Discrete Sine Transform (DST) instead of DCT, reaching better performance compared to JPEG at the cost of higher computational overhead. Besides, many multimedia compression standards use the DCT transform on two dimensions (DCT-2D) with a small coding unit to reduce the overall time complexity.

B. DL-based Image Restoration Models

The image restoration task can be very challenging since the image degradation process is mostly irreversible. Recently, a set of DL-based models were presented to tackle this challenge and they are listed and described briefly in the following.

The current DL-based models have demonstrated a high-efficiency in extracting patterns from high dimensional-data (images). These models can achieve promising results in several image restoration tasks such as image super-resolution, and image denoising. In this context, residual networks, a type of DL architecture, have been widely adopted for image super-resolution. A CNN-based super-resolution architecture has been proposed in [28], [29]. This architecture is based on mapping the low-resolution parts of an image to their corresponding interpolated high-resolution parts. This architecture has been later enhanced by adding more layers and sharing weights. Introducing residual layers and adding substantial layers, Kim et al. present VDSR [30] and then add recursive connections with a deeper network with weights sharing in DRCN [31]. Tai et al. propose a similar architecture based on recursive blocks in DRRN [32] and then they add a memory to these blocks in Memnet [33]. A common characteristic of the residual-based architectures is

adding skip connections between layers. This was applied in SRResNet [34] and EDSR [34] for designing efficient super-resolution networks. However, the main limitation of such networks is the reuse of features in just one layer per block. DenseNet was proposed permitting subsequent reuse of features. Concatenating the layers instead of the summation in ResNet, DensNET achieves better performance. In this context, Tong et al. propose SRDenseNet [35] for image super-resolution by removing the pooling layers from DensNet. Similarly, Zhang et al. use the dense residual blocks in RDN [36]. Haris et al. propose DBPN [37], which is constructed by iterative up and down sampling blocks restricting the block information flow during propagation. To guarantee a better flow of information between blocks, DBDN was proposed in [38] enabling the reuse of the local and high-level information by adding intra-block and inter-block dense connections. More recently, RGSR was proposed as a two-step image super-resolution scheme having as input compressed images [39].

In this paper, the Residual Dense Network (RDN) [36] model is used as a denoising and super-resolution model to be applied for recovering compressed images with low image quality (high compression ratio).

C. DL-based Image Compression

The work described in [40] presents a novel approach for compressing data using JPEG image compression standards. This method is specific for the JPEG codec and sends fewer AC values for each block than 63. At the application server(s), compressed data is analyzed by a reconstruction model to obtain all AC values for each block. The reconstructed image is then applied to a trained model to improve the visual quality of the received image. However, this solution is limited to the JPEG codec. Moreover, this solution requires modifying the JPEG standard, which is not preferable for practical applications. Similarly, another solution is proposed in [41], in which a Convolution Neural Network (CNN)-based auto-encoder is used to compress an image captured from a MIoT device, followed by the Discrete Wavelet Transformation (DWT). However, implementing CNN networks with a large number of input dimensions and multi-layers with multiple operations (convolution, Relu activation function, and batch normalization) in addition to DWT, is infeasible for MIoT devices. this is due to the high memory, computational power, and energy consumption requirements. As a result, this method cannot be used effectively with MIoT devices and is also incompatible with existing image compression algorithms. Recently, the authors in [42] proposed a DL-based compression technique. However, DL-based image compression approaches demand more memory and energy compared to the standard image compression algorithms. Applying an existing compression standard, the authors in [43] presents a solution to compress raw IoT data by using the lossy SZ algorithm. To enhance the reconstructed 1D signal, a trained autoencoder-based model is applied at the edge node. However, this approach cannot be easily applied to multimedia content such as images.

As a summary, existing DL image compression techniques can achieve large gains but cannot be applied on MIoT devices, that present memory, computation, and energy constraints. Additionally, employing a hybrid solution (a specific lossy image compression method combined with a DL model) is inflexible and could necessitate changes to existing standards. To overcome this limitation, our proposed solution is generic and can be applied with any image lossy compression.

III. System Model

As shown in Fig. 2, MIoT consists of a large number of edge MIoT devices that can capture, process, and deliver multimedia data (e.g. images) to the data center or application server(s) through multi-hop or star wireless communications. Each MIoT device, after image acquisition, compresses the image and sends it to the application server. Then, the application server or data center decompresses the received image to recover the visual content. Given the large scale of the MIoT network and the large amount of communicated multimedia data, there is a real burden on the network bandwidth and MIoT devices' resources. Therefore, to reduce the required resources overhead, the compression algorithm realized on the MIoT devices should reduce the size of the transmitted image to the minimum possible to optimize the energy and bandwidth consumption.

Thus, our proposed solution aims to allow MIoT devices compressing the collected images with a high compression ratio to minimize the transmitted data size. Then, the visual contents of the received decompressed images can be enhanced at the application server(s) by employing a DL denoising & super-resolution model such as the RDN model. Two variants of the proposed solution are considered (see Fig. 3): the first consists of having only compressed images sent by the MIoT devices, while the second variant considers sub-sampled and compressed images sent by MIoT devices. In this context, our proposed system model, illustrated in Fig. 4, consists mainly of two steps:

1) Using a lossy image compression with a high compression ratio, the MIoT devices send the compressed images to the application server.
2) Upon receiving the compressed images from MIoT devices, the application server (or data center) recovers their visual contents, which might present visual degradation, depending on the employed compression ratio. Then, the RDN model is applied to further enhance the quality of the decompressed images by reducing the compression noise effect. Let us indicate that, for the first variant of the proposed solution, the RDN model plays the role of a denoiser, while, for the second variant, the RDN model ensures also the super-resolution property in addition to the denoising one since the images are down-scaled before being compressed.

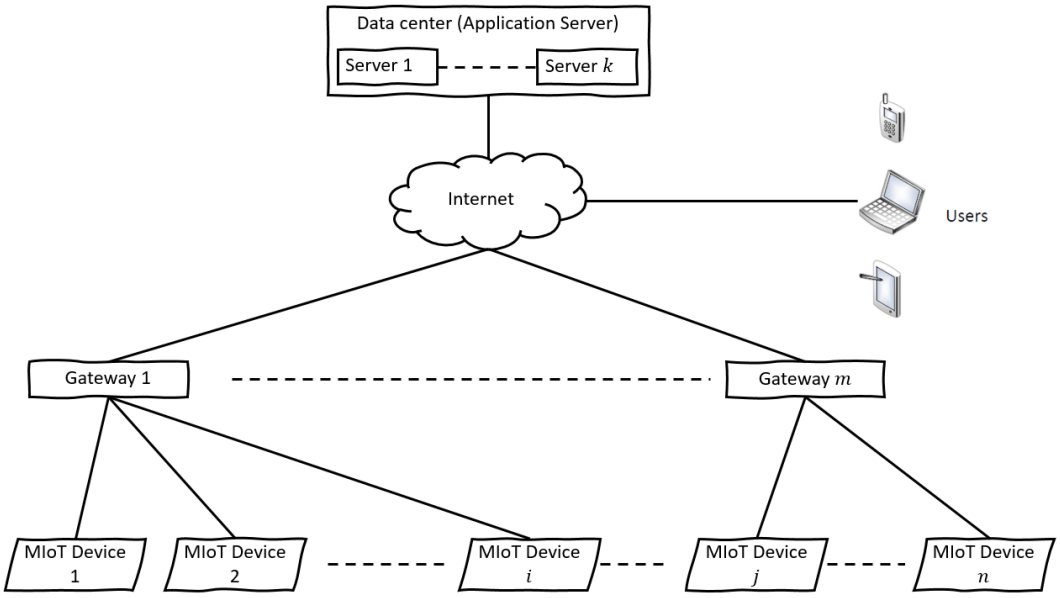
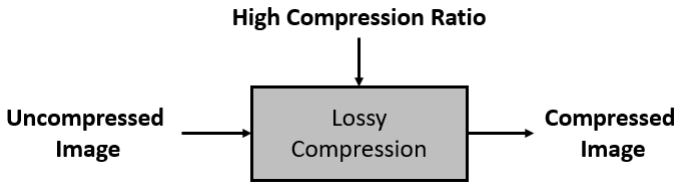
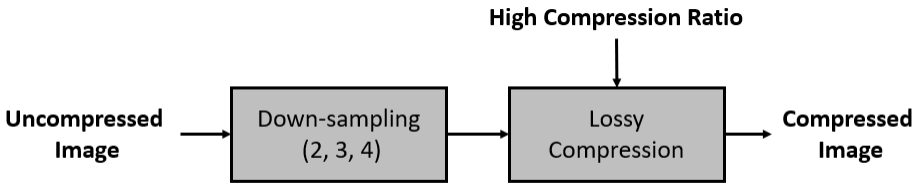


Fig. 2: An example of an MIoT network model: a set of n MIoT devices, m gateways, and the application server (or data center)



(a) First variant



(b) Second variant

Fig. 3: The proposed first (a) and second (b) variants

In fact, the MIoT devices have different constraints, and using the same image quality for all applications is not practical. The image quality (and down-scaling factor for the second variant) depends on the target application requirements. Thus, the denoising and super-resolution model should be able to function with these different configurations (ensuring flexibility) towards effectively reconstructing the decompressed images. To this aim, the considered RDN model is trained (scaling or without scaling) with different image quality values to respond better to real-world scenarios. The proposed model aims to reduce the effect of lossy compression and down-scaling operations in contrast to existing applications of the RDN model, which aim to reduce the effect of other types of noise (e.g. Gaussian white noise). There is a range of targeted applications that can profit from the suggested approach, from military monitoring applications to automated assistance for older individuals, including advanced healthcare systems, home automation applications, etc.

A. RDN Architecture

RDN consists mainly of four main components as illustrated in Fig. 5:

1) **The Shallow Feature Extraction Net (SFENet)**: which consists of two convolutional layers, introduced to extract shallow features.
2) **The Residual Dense Blocks (RDB)**: which take as input the extracted shallow features.
3) **The Dense Feature Fusion (DFF) layer**: that fuses features from all the preceding layers, which are:

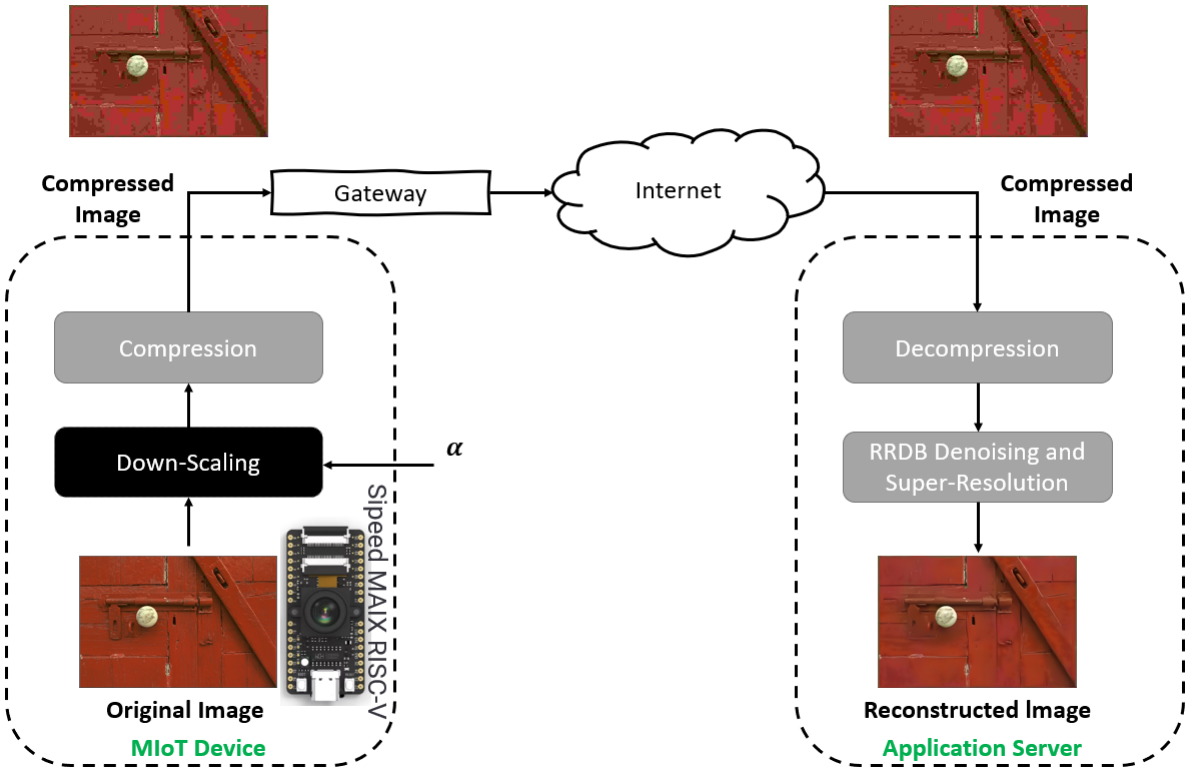


Fig. 4: Functional diagram of the proposed solution.

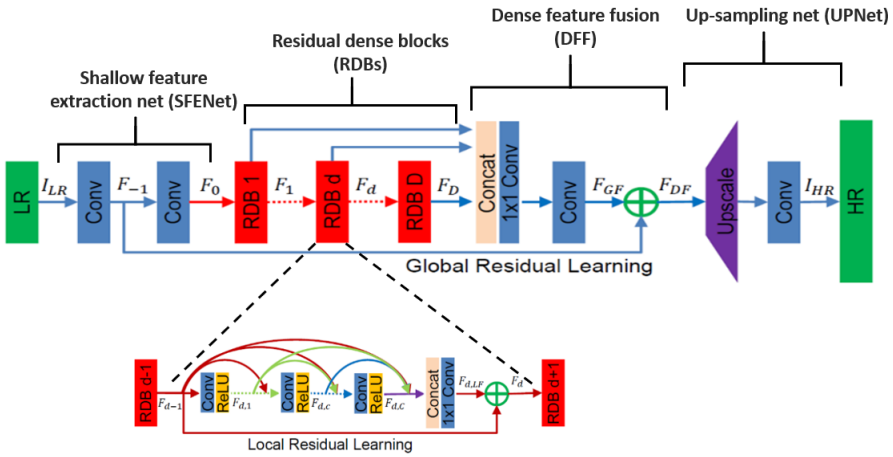


Fig. 5: Residual Dense Network (RDN) architecture [36].

- The Global Residual Learning (GRL) layer: that processes the shallow feature-maps that represent the output of the first convolution layer of SFENet.
- The Global Feature Fusion (GFF) layer: that fuses features from all the RDB.

4) **The Up-sampling Net (UPNet)**

As illustrated in Fig. 5, the RDN architecture consists of D identical Residual Dense Blocks (RDB). Each residual block contains $C = 8$ convolution layers (with 3×3 filters) and 1 convolution layer (with 1×1 filters). In each residual dense block, there are Ns shortcut connections, which represent an identity mapping that can help to solve the visual degradation problem appearing in stacked non-linear layers. Moreover, all layers use the Rectified Linear Unit (ReLU) as activation function to fit the residual dense mappings. The final convoluional layer has 3 output channels, given that the output high-resolution images can be colored, but also it can process gray images. For more information about the RDN architecture, readers can refer to [36].

The benefit of the RDN model is that the hierarchy of all convolutional layers can be completely used. RDN consists of a D Residual Dense Block (RDB), which can extract several local features (characteristics) through densely connected

convolution layers. RDB creates direct links from previous RDB states to all current RDB layers, resulting into a Contiguous Memory (CM) mechanism. RDB's local feature fusion is used to acquire more effective features from previous and current local features in addition to stabilizing the learning of a wider network. The global feature fusion is employed to learn global hierarchical features jointly and adaptively in a comprehensive way after acquiring dense local features. Compared to DenseNet, RDN eliminates batch normalization and pooling layers. Pooling layers are removed from RDB since it could discard some pixel-level information, which is not preferable for image denoising and super-resolution.

The proposed solution applies the RDN model at the application server(s) to improve the image quality of the recovered decompressed images. Moreover, this model can also ensure the super-resolution property, which is beneficial for the second variant (that uses down-sampling). The purpose of RDN in the proposed approach is to learn mapping functions between the original uncompressed image I and the compressed image J. For achieving this purpose, a set of denoising and super-resolution models were tested, and we select RDN since it can ensure acceptable performance. Any new image restoration model that can ensure better results can be used instead of RDN. The model's input image is $I = J + N$, where J represents the compressed image and N represents the reflecting block artifacts image introduced by lossy compression. The aim is to learn a residual dense mapping function between the original and the decompressed one.

In addition, the trained model for each variant, that will be applied at the application server (or cloud/ data center), requires a standard desktop computer without GPU or with a GPU to speed up the computation process. This work proposes the use of denoising and super-resolution DL model to enhance MIoT communication that their devices suffer from limitations in terms of computation and resources in addition to latency requirements for these applications.

Besides, this type of DL model can help these limited devices by reducing computation and communication overhead, which can help to reach a good balance between communication size and image quality. We have tested other models in addition to the listed RDN model and similar results were obtained. After receiving all packets of compressed images, they are decoded and form the compressed image at the application server. Then, the decompression image algorithm is applied to provide a decompressed image that will be enhanced by using the corresponding trained model. Trained model can be selected in fixed or dynamic manner (first or second variant) according to the desired configuration.

IV. Experimental Analysis

In this section, we include the implementation details of the proposed solution. Then, we present and discuss the evaluation results including the compression visual effect, the communication size efficiency, the visual quality enhancement after applying the proposed model and the power consumption. Experiments were done on a Tesla V100 GPU. Moreover, we present the results of applying the proposed method on limited microcontrollers with different communication technologies, including transmission time and energy consumption results.

A. Data Description

The RDN model was trained using the dataset described in [44]. This dataset consists of a large set of colored images collected from the Internet. These high-resolution images are first down-scaled to images having random size between 500 and 1000 pixels. Then, the obtained images of variable size are cropped randomly to get 256x256 images. These images are compressed using JPEG and BPG, respectively, and the obtained pixel values are scaled between [0;1]. The testing is done with images chosen from the Kodak dataset (24 images) [45].

B. Model Implementation

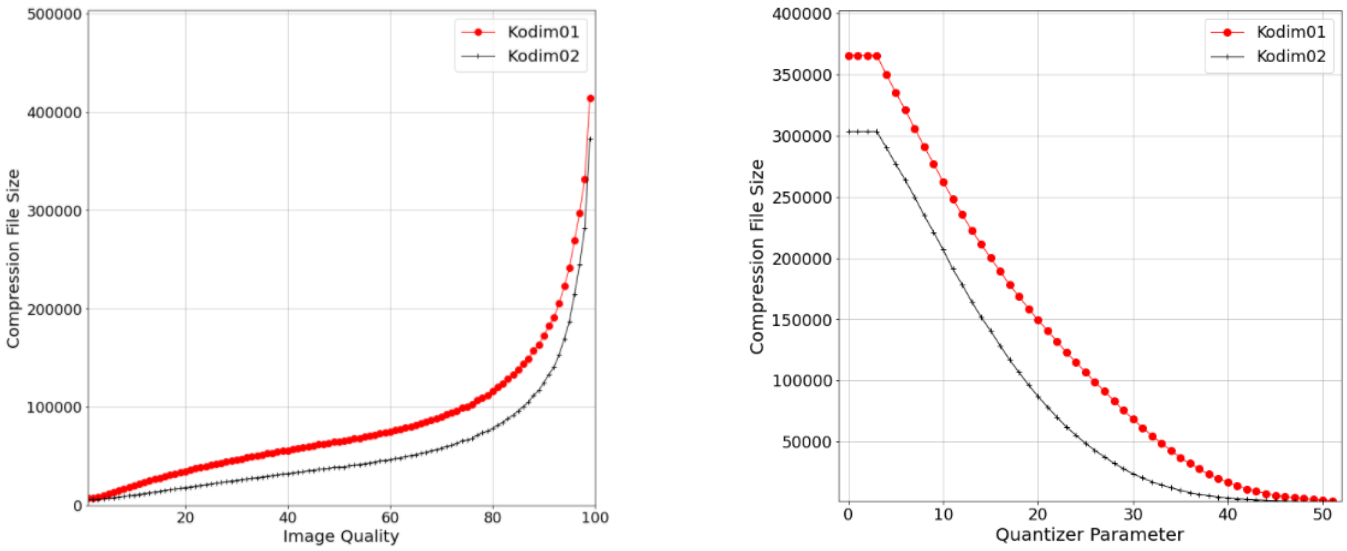
The employed RDN model was implemented in Pytorch [46]. The mini-batch technique is used to train the RDN model. The learning rate is initialized to 10^{-4}. At each iteration, the normalization is applied to each mini-batch. The final output (desired output) is the reference uncompressed image (enhanced) I, which will be compared with the compressed image through the loss function. The Adam optimizer with an adaptive learning factor is used to optimize the loss function.

Before being passed to the RDN model, the images are compressed with JPEG and BPG, respectively, using variable compression ratio. Increasing the compression ratio will lead to an increase in the block artifacts. The size of the image patch should be then selected to include relevant useful patterns. According to the empirical evaluation, we find that the size of the image patch varies between 96 and 192 to contain enough information to remove noise and compression artifacts.

Two models are trained, where the first model is for the first variant (compression image without down-scaling operation), and the second one is for the second variant (compression with down-scaling operation). The compressed (and scaled) and uncompressed images are then fed into the proposed model. The output of the first RDN model (first variant) has the same size as the input, which can be considered as the input image plus the related compressed noised image. While for the second model of the second variant, the input size is down-scaled by 4 compared to the output high-resolution image, which has the same size as the original uncompressed one. These RDN models can be then employed to better remove and/or reduce the compressed noise.

C. Compression Effect on Image Quality

In this part, we analyze the effect of the compression on the visual degradation, considering different compression



(a) JPEG (b) BPG

Fig. 6: Variation of the compression size versus compression image quality for two Kodak images with JPEG compressor.

ratios. The lower is the compression ratio, the better is the compressed image quality, and vice-versa. The image quality parameter depends on the target IoT multimedia application and should be set accordingly (see Fig. 6). We should note that this parameter has not to be static and could be adapted dynamically regarding the instantaneous requirement of the IoT multimedia application. For instance, if an event of major importance occurs (e.g. robber detection in a surveillance system) and the application requires a higher image quality, it could then configure the MIoT devices to send the image with low compression ratio. Otherwise, the application server (or data center) could enhance the quality of highly compressed images. In this sense, the proposed approach can offer the flexibility on the quality of the received multimedia contents that can be controlled by the application server (or the data center).

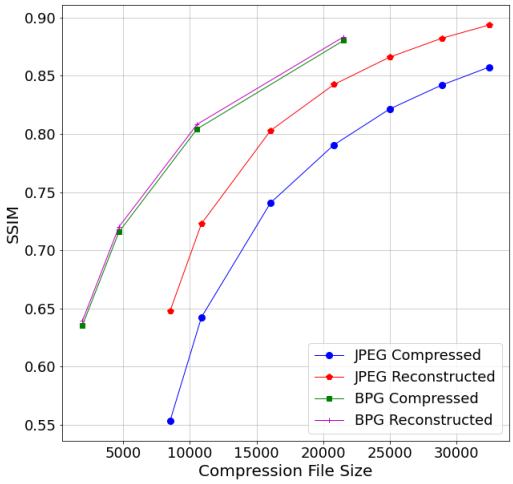
The variation of the Structural Similarity (SSIM) and Peak Signal-to-Noise Ratio (PSNR) versus the compressed data size are presented for both variants in Fig. 7 for the images shown in Fig. 8. The obtained results indicate that by increasing the image quality (or decreasing the compression ratio), the PSNR, SSIM, and the size of the transmitted (stored) data increase. This clearly indicates that the visual degradation increases when the compressed data size decreases (or when the compression ratio increases), as shown in Fig. 7. Thus, a trade-off between visual quality and compressed data size exists.

D. Denoising and Super-resolution Effect on Image Quality

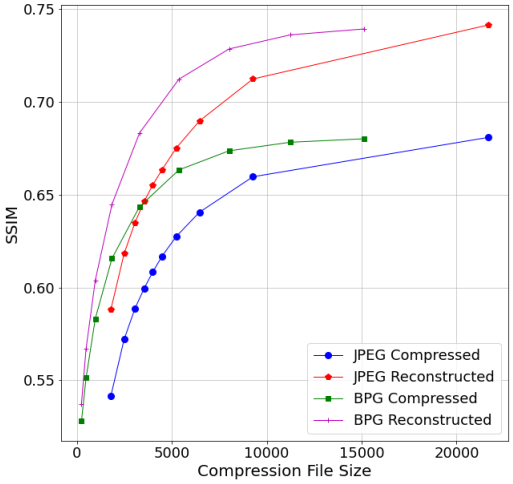
Given that some visual information are lost after lossy compression and cannot be recovered, the proposed approach is introduced to provide a better image quality by reducing the compression noise effect. In this context, the results shown in Fig. 7 show that the SSIM and PSNR values of the enhanced reconstructed images are higher compared to the decompressed ones with or without down-scaling operations. Moreover, the obtained visual content quality with the first variant is better compared to the second variant. Therefore, if the down-scaling operation is applied (second variant), a lower compression ratio is required compared to the first variant towards preserving the main visual content quality. However, the down-scaling results in higher data reduction compared to the first variant (without down-scaling) even with low compression ratio. Thus, the choice of which variant should be applied (with or without down-scaling operation) depends on the target IoT multimedia application and MIoT device limitations. For example, 33% of compressed data reduction compared to a JPEG image with default image quality (85%) is required to reach the same image quality by using the proposed approach (with the first variant).

To evaluate the image quality enhancement, the proposed solution (two different models for each image compression algorithm) was tested with 24 images selected from the Kodak dataset. For two images randomly chosen from the testing dataset (see Fig. 8), we show the visual degradation versus image compression ratio for the decompressed and enhanced images with the first variant (without down-scaling operation) in Fig. 9 and 10, respectively. Also, same results are provided with the second variant (with down-scaling operation) in Fig. 11 and 12. It should be noted here that these results are for the JPEG standard, and that similar results were obtained with BPG, but they were omitted to conserve space.

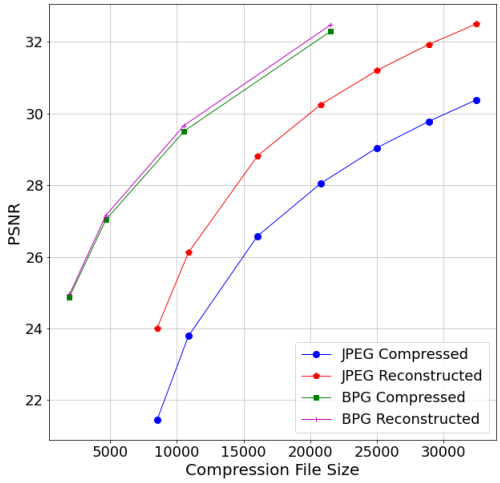
To evaluate the visual degradation before and after applying the proposed model, the PSNR and SSIM metrics were used. According to the obtained results, illustrated in Fig. 7, the proposed solution ensures better image quality compared to the decompressed one. This indicates clearly that the effect of lossy compression with high compression ratio can be reduced



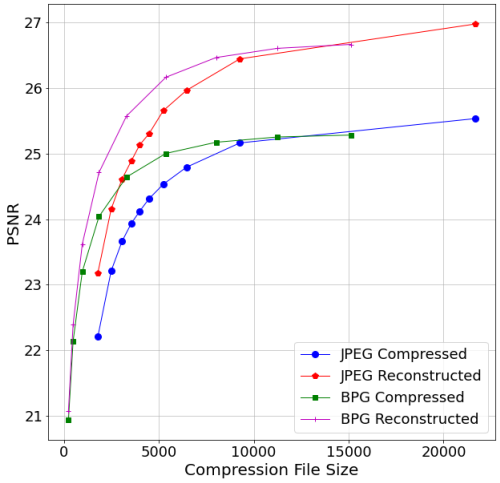
(a) SSIM without down-scaling operation



(b) SSIM with down-scaling operation



(c) PSNR without down-scaling operation



(d) PSNR with down-scaling operation

Fig. 7: Variation of the average of SSIM (a), PSNR (b) versus image quality for Kodak test images without or with down-scaling.

(a) kodim01

(b) kodim02

Fig. 8: Original uncompressed images "kodim01",(a) and "kodim02" (b)

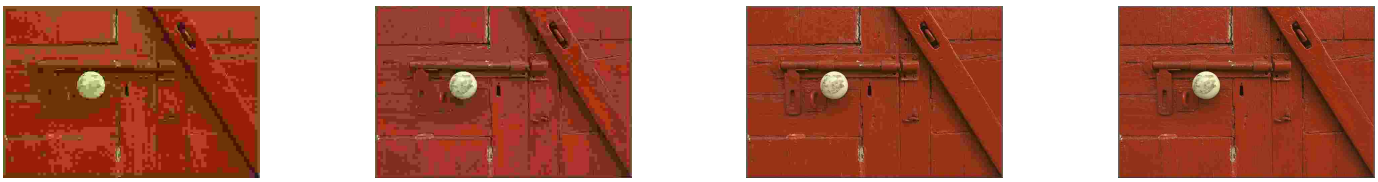
by applying the proposed RDN model (without or with down-scaling). Moreover, the average of SSIM and PSNR, shown in Fig. 7, indicates that the recovered images with the first variant presents higher SSIM and PSNR values compared to the second variant that uses a down-scaling operation.

In the JPEG case (see Fig. 7-a)), with the first variant, the SSIM varies between 0.5532 and 0.8574, while for the reconstructed images, the SSIM is between 0.6477 and 0.8936. Similarly, the PSNR varies between 21.44 dB and 30.38 dB for the decompressed images, and it varies between 24 dB and 32.5 dB for the reconstructed images. For the first variant, the enhancement in the visual content is increased by 0.0585 in terms of SSIM and by 2.25 in terms of PSNR. Additionally, we found that on average, if the compressed file size is increased by 3989 bytes, the SSIM and PSNR values will be increased by 0.041 and 1.417, respectively. Besides,

(a) Quality=1 (b) Quality=5 (c) Quality=10 (d) Quality=15

(e) Quality=1 (f) Quality=5 (g) Quality=10 (h) Quality=15

Fig. 9: Corresponding compressed (JPEG) "kodim01" images (a)-(d) and reconstructed ones (e)-(h) by using the proposed model after decompression, respectively versus different image quality, and for the proposed solution without down-scaling operation.



(a) Quality=1 (b) Quality=5 (c) Quality=10 (d) Quality=15

(e) Quality=1 (f) Quality=5 (g) Quality=10 (h) Quality=15

Fig. 10: Corresponding compressed (JPEG) "kodim02" images (a)-(d) and reconstructed ones (e)-(h) by using the proposed model after decompression, respectively versus different image quality, and for the proposed solution without down-scaling operation.

(a) Quality=10 (b) Quality=20 (c) Quality=30 (d) Quality=40

(e) Quality=10 (f) Quality=20 (g) Quality=30 (h) Quality=40

Fig. 11: Corresponding compressed scaled (JPEG) "kodim01" images (a)-(d) and the reconstructed ones (e)-(h) by using the proposed model after decompression, respectively versus different image quality, and for the proposed solution with down-scaling operation.

with the second variant, the SSIM varies between 0.5415 and 0.6809 for the decompressed images, while the SSIM of the enhanced recovered images varies between 0.5881 and 0.7415. The enhancement of the visual content quality is increased by 0.0491 in terms of SSIM and by 1.09 in terms of PSNR. Additionally, we found that on average, if the compressed file

(a) Quality=10 (b) Quality=20 (c) Quality=30 (d) Quality=40

(e) Quality=10 (f) Quality=20 (g) Quality=30 (h) Quality=40

Fig. 12: Corresponding compressed scaled (JPEG) "kodim02" images (a)-(d) and the reconstructed ones (e)-(h) by using the proposed model after decompression, respectively versus different image quality, and for the proposed solution with down-scaling operation.

size is increased by 574 bytes, the SSIM and PSNR values will be increased by 0.015 and 0.387, respectively. Moreover, for the first variant, the maximum SSIM and PSNR values are 0.89 and 32 dB, respectively, while for the second variant, the maximum values of SSIM and PSNR are 0.741 and 26, respectively. Furthermore, the second variant with an average of 18% compression size of the first variant can ensure less than 0.168 for the SSIM and 4.56 for the PSNR. The first variant with JPEG ensures an SSIM and PSNR greater by an average of 0.15 and 3.3 in the first variant compared to the second one, respectively but with 4 times of additional communication overhead.

In the case of BPG (see Fig. 7-b)), a lower visual content enhancement is achieved for the first variant compared to JPEG. This is due to the fact that decompressed BPG images are less degraded compared to the JPEG case. Furthermore, BPG introduces more computing overhead (more resources for the compression process compared to JPEG) but it ensures less communication overhead. In contrast, for the second variant, the visual content is enhanced with BPG, where a down-scaling operation is applied. The SSIM varies between 0.6355 and 0.8804 for the reconstructed enhanced images with the first variant, while it varies between 0.5374 and 0.7394 for the second variant. Similarly, the PSNR varies between 24.95 dB and 32.48 dB for the first variant, and it varies between 21.08 dB and 26.66 dB for the second variant. The enhancement in the visual content is increased on average by 0.00375 and 0.0373 in terms of SSIM and by 0.1325 and 0.8433 in terms of PSNR for the first and second variants, respectively.

E. Computation Complexity

In fact, the proposed solution is designed to reduce the required computation complexity for IoT end-devices, which depends on the variant, as follows:

1) Only one operation is required, which is the image compression for the first variant. In addition, the computation complexity of this variant depends on the employed image compression algorithm and the input image size. This was the main motivation of proposing the second variant.
2) For the second variant, an additional simple operation is required compared to the first variant, which is the down-sampling operation that decreases the image size and consequently this will lead to reduce the input image size and consequently the computation complexity of the image compression process.

Besides, the proposed solution was tested with two image compression algorithms: JPEG and BPG. The idea of testing with JPEG is that it is more simpler compared to BPG and can be implemented with limited MIoT devices.

On the other hand, BPG compression reduce the communication size (compressed image) compared to the JPEG but it requires more compression computation overhead and consequently increases the computation power consumption. Unfortunately, BPG can be implemented with powerful devices such as raspberry pi Zero but not with limited ones such as micro-controllers.

On the cloud side/ data center/ application server(s), a trained denoising super-resolution DL model will be applied to reduce the introduced visual degradation during the compression with a high compression ratio in addition to the down-sampling operation. The trained model plays the role to reduce/eliminate the strong compression distortion. RDN is used in this solution as proof of concept and similar results are obtained with other models.

At the application server(s)/ data center/ Cloud, no limitation in terms of computation and resources. Therefore, the trained model can be run without any hardware issues in contrast of MIoT end devices. A standard desktop computer can be used in this context. In addition, using GPU devices will accelerate the computation of the enhancement process.

F. WiFi Multimedia Applications

The communication efficiency is proportional to the communicated data size. In our proposed approach, we

TABLE I: Comparative analysis results between proposed variants

Image Quality without downscaling	Image Quality with downscaling	SSIM without downscaling − SSIM with downscaling)	PSNR without downscaling − PSNR with downscaling	Compression Size Ratio (without down-scaling/with downscaling)
1	10	0.012	-0.77	0.209
5	20	0.07	0.58	0.228
10	30	0.152	2.92	0.19
15	40	0.191	4.12	0.17
20	50	0.213	4.92	0.16
25	60	0.225	5.47	0.155
30	70	0.23	5.85	0.161

aim at adapting the existing compression algorithms to be applied with high compression ratio to achieve communication efficiency. The first test involves storing a 1179 Kbyte image[1] on a Raspberry Pi 3 Model B and calculating the time required to send the compressed image to a local PC using various compression approaches. WiFi was employed as the wireless communication technique in this test, and the data packet size was set to 1400 Bytes.

The compression ratio is calculated by dividing the compressed image size by the original image size. As a result, smaller compressed data sizes lead to higher compression ratios, but the decompressed image has important visual degradation. Using the proposed method, a larger compression ratio can be used while maintaining the same image quality.

For instance, in the case of JPEG, a lower-quality image can be used (between 1 and 25 percent without scaling, and between 20 percent and 60 percent with scaling). For various image quality (compression ratio) values, the ratio of compressed transmitted data with and without scaling is shown in TABLE I. This clearly demonstrates that the proposed approach with downscaling reduces the compressed image's size by an average of 18% when compared to utilizing JPEG without scaling.

Table II shows the average compression and transmission times and the average compression data size for both lossless (BMP) and lossy (JPEG codec) techniques. The results show that the compression time is roughly constant for a given image size regardless of the image quality. The second lossy variant (scaling) requires less latency than the lossy variant without scaling and the lossless variant. Moreover, the transmission time increases with image quality as the compressed data size increases.

In summary, the image quality influences both transmission delay and size. The lossy compression with scaling, with the same image quality, can reduce latency by 50-60% and data size by 63-65% compared to the lossy compression without scaling. This clearly shows that the proposed alternative can reduce communication sizes and latency than the first.

[1]http://r0k.us/graphics/kodak/kodim01.html

G. Bluetooth Low Energy Multimedia Applications

To demonstrate the proposed approach's efficacy on devices smaller than a Raspberry Pi, various experiments were done on resource-constrained microcontrollers. Bluetooth Low Energy (BLE) was also used for testing due to its widespread use in commercial products in addition to its low energy consumption and short packet size, emphasizing the crucial significance of dramatically lowering the size of communicated data.

1) Experimental setup: The ESP WROOM 32 with 520 KBytes on-chip SRAM and the more resource-constrained RedBear BLE Nano with 64 KBytes RAM were utilized to implement the compressor. The amount of data per packet was set to 1500 Bytes for WiFi communication. The BLE communication was configured with an advertising interval of 1800 milliseconds, and a transmit power of +4 dBm, and the amount of data per packet was set to 15 Bytes. An Arduino UNO with an INA219 breakout board was utilized to measure the energy consumption. Figure 13 depicts the instruments used in this section's experiments.

The JPEGENC library was used to implement the JPEG encoder on the microcontrollers [2]. It was slightly modified to allow an option with a high compression ratio and bad quality. In the following, four image reduction ratios were considered for the proposed approach:

- **High quality**: the compressed data consists of 18% of the original data
- **Medium quality**: the compressed data consists of 12% of the original data
- **Low quality**: the compressed data consists of 8% of the original data
- **Bad quality**: the compressed data consists of 6% of the original data

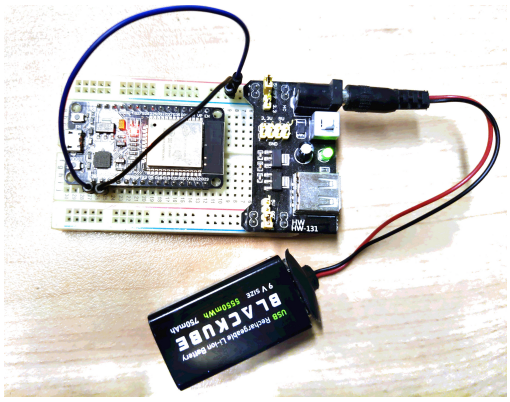
Take note that a compressor of "Bad quality" implies that the compression ratio was highly prioritized over data quality.

The processing time required to compress an image varies between 7 and 150 milliseconds, depending on the image's size. The time required to compress various grayscale photos of varying sizes on an ARM Cortex-M4 is illustrated in Figure

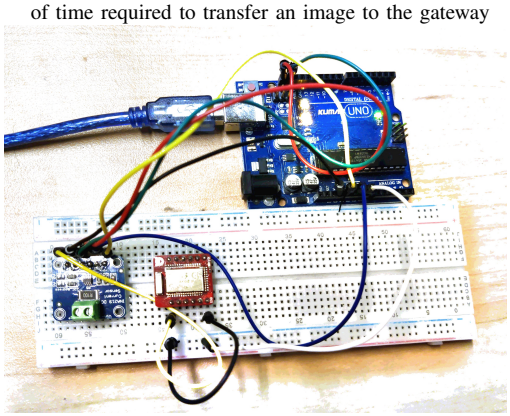
[2]https://www.arduino.cc/reference/en/libraries/jpegenc/

TABLE II: Statistical results about communication latency and data size for the proposed variants

Variant	BMP Lossless	JPEG Without Scaling			JPEG With Scaling (4 times)		
		Quality= 20	Quality=40	Quality=60	Quality= 20	Quality=40	Quality=60
Mean Compression Time (seconds)	0.0223	0.0486	0.0489	0.049	0.0173	0.0175	0.0179
Mean Transmission Time (seconds)	23.01	0.164	0.291	0.4041	0.0594	0.095	0.1929
Mean Compressed Data Size (bytes)	786486	13848	21051	28100	5161	7773	10138



(a) The ESP WROOM 32 was used to calculate the amount of time required to transfer an image to the gateway



(b) Using an Arduino and an INA219 to measure the current of a REDBEAR NANO

Fig. 13: The devices used to evaluate the transmission time and current consumption

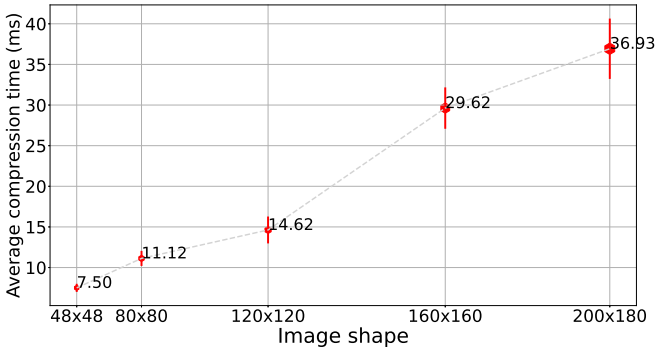


Fig. 14: Average compression time in milliseconds on an ARM Cortex-M4F for multiple images with varying shapes

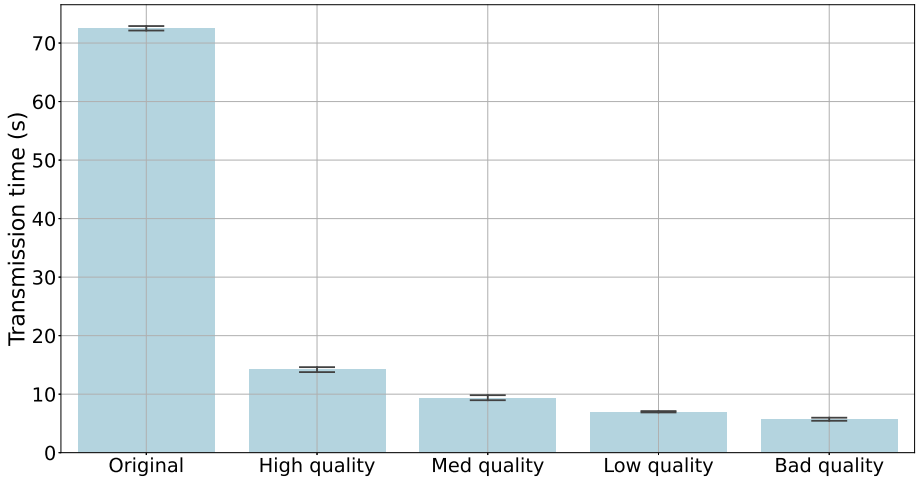
14. Large photos need more time to be compressed. For example, compressing a 120x120 image takes approximately 14.5 milliseconds, whereas compressing a 200x180 image takes approximately 37 milliseconds.

2) Transmission time: The BLE packet data field is dependent on the Bluetooth specification. In this experiment, we considered a gateway that uses Bluetooth v4.0 where the maximum size of the data field is 27 bytes. The BLE throughput is dependent on the duration of the connection interval and the number of packets transmitted per connection event. The latter depends on the BLE stack and chipsets. The transmission time was calculated by storing a 36 Kbytes image in the memory of the ESP WROOM 32 and setting the connection interval to 100 ms. As a gateway, a Raspberry Pi 3 Model B was utilized, and the "pygatt"[3] Python library was used to receive the BLE data. It is worth mentioning that the distance between the sender and receiver varies between one and two meters.

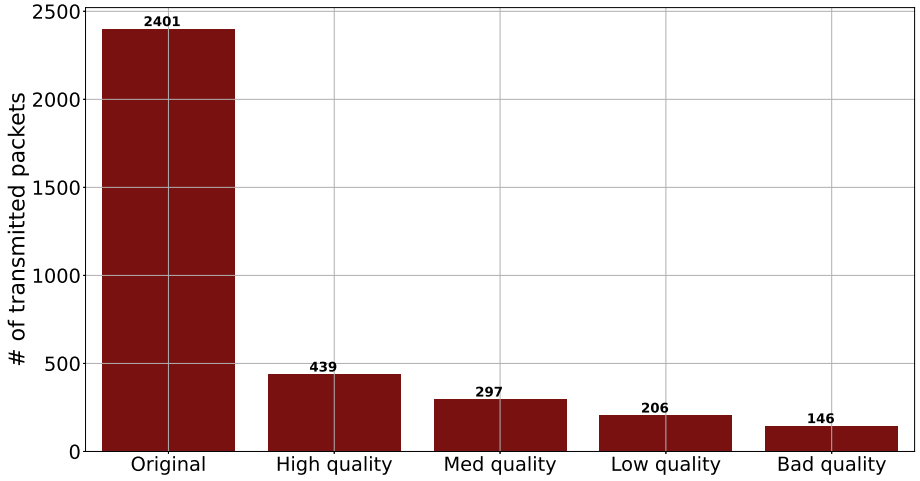
According to the obtained results displayed in Figure 15, the transmission of the uncompressed 36 Kbytes image took around 71 seconds and used approximately 2400 BLE packets. When a high-quality compressor that takes into account the trade-off between compression ratio and data distortion is used, the transmission time is reduced to around 13 seconds. The number of packets necessary is reduced to 18% of the number required to send the original image. When a bad-quality compressor is used to boost the compression ratio at the cost of image quality, an additional 10% in time is gained (equivalent to 8 seconds approximately). The required number of packets is reduced to 6% of the number required to transmit the original image.

3) Energy consumption: In this experiment, four scenarios were examined. Each one was monitored using the INA219 for 90 seconds to determine the REDBEAR NANO's current consumption. Notably, because a fixed voltage was used in this test, we just present the current consumption results given that the energy consumption is proportional to the current one. In the first scenario, the device is in advertising mode and is not connected to the gateway. In the second scenario, the device transmits a 14 Kbytes image stored in its memory to the gateway. The third and fourth scenarios involve compressing and then sending the image using a high-quality compression algorithm and a low-quality compression algorithm, respectively. Thus, there is no transmission during the 90 seconds of recording in the first scenario, whereas

[3]https://github.com/peplin/pygatt



(a) The time in seconds required for transmission using various image reduction ratios



(b) The number of packets required for transmission using a 15-byte packet size

Fig. 15: The transmission time and number of packets required to transfer a 36 Kbytes image from the ESP WROOM 32 to the gateway

there is only one image transmission in the others. Given that our objective is to determine the effect of increasing the compression ratio on energy consumption, we did not employ a camera to take images and hence did not incorporate the sensing operation in this experiment. Instead of that, a picture with a 120x120 shape was loaded from an array and compressed. Note that the image size was reduced from 36 to 14 Kbytes due to the REDBEAR NANO's lower memory capacity compared to the ESP WROOM 32.

To evaluate the energy consumption (approximated by the current consumption per time), we measure the current over a period of time, 90 seconds in our case. Then, we can calculate the integral over that time period to find the area under the curve denoted in the following equation:

$$E_{t_1-t_n}[mA.s] = \int_{t_1}^{t_n} I(t)dt \tag{1}$$

The trapezoidal function that returns an approximation of the integral was used to approximate the area under a curve. Figure 16 illustrates the current consumption of REDBEAR NANO over a 90-seconds period for each of the four scenarios. When the device is operating solely in the advertising mode (Figure 16-a), it is possible to observe a low base current consumption corresponding to the device's sleep mode consumption and a peak that appears approximately every 2 seconds (defined advertising interval), corresponding to the advertising packet sent for other devices to receive and process. A zone where the base consumption increased to around 6 mA may be seen in Figure 16-b. This zone corresponds to the transmission phase, during which the device exits the sleep mode for around 30 seconds. Prior to and following this phase, the device is connected to the gateway, and the device and the gateway exchange more data, as the connection interval is shorter than the advertisement interval. Figure 16-b demonstrates the importance of drastically lowering the quantity of delivered data, not only to reduce the number of packets but also to avoid waking up the device for an extended period.

Figures 16-c and 16-d show how a compressor can reduce

the current consumption by shortening the time spent outside the sleep mode. As illustrated in Table III, when a high-quality compressor is used, the current consumption per time (mA.s) is decreased from 189 to 47 and the average current consumption (mA) is lowered from 2 to 0.52. By implementing the proposed approach, which involves increasing the compression ratio at the expense of image quality and then recovering the image at the gateway, we gain around 20 mA.s over a 90-seconds period and we reduce the average current consumption from 0.52 to 0.3. Keep in mind that the image was loaded from an array in this experiment. In reality, the microcontroller will exit sleep mode to capture an image, resulting in slightly higher energy consumption. However, the central premise remains valid: to maximize the device's lifetime, it should return to sleep mode as quickly as feasible after capturing and sending an image. This was done in this experiment by modifying the JPEG encoder and allowing a higher compression ratio so the image transmission could be done faster.

V. Discussions

In this paper, the aim was to minimize the communication cost by highly compressing the data without loosing the image quality. The work done in this paper can improve the existing limitations of multimedia compression approaches used in MIoT. The advantage of the proposed solution is that it is independent of the used multimedia compression algorithms and it does not require any modification in the existing multimedia compression standards (practical implementation). The main challenge was to preserve image quality while using high compression ratio. This was addressed by employing the RDN learning model. The proposed solution was tested with the JPEG and BPG image compression algorithms (with or without down-scaling operation).

The employed RDN learning model succeeded in enhancing the compressed image quality. This was validated by evaluating the PSNR and SSIM values that were increased compared to the decompressed images for both variants. This solution was designed to not require any additional operation on the MIoT devices by delegating the role of image quality enhancement to the application server. In addition, the proposed solution optimizes the transmitted data size, that can be reduced to 33% for image quality equal to 5 with JPEG compression compared to default image quality. Thus, the proposed solution could highly improve the transmission efficiency for any MIoT application using any compressor like JPEG or BPG. The experimentation results showed that the average PSNR and SSIM values are increased and the transmission data ratio is decreased by an average of 18% using the second variant compared to the first variant.

A further optimization of the communicated data size would be in delivering gray images while designing (using) a DL-based model for the colorization. As a future work, we will further explore other learning models that can colorize gray images towards reducing more and more the communication overhead.

Besides, applying the proposed method with video compression, key-frame extraction techniques for event summarization such as in [47]-[52] can enhance the proposed approach by generating concisely and intelligently video abstraction. This will reduce the number of transmitted frames that can be compressed by using the proposed solution. Indeed, this requires to employ (or adapt) these summarization techniques at the MIoT devices. Then, each summarized frame can be considered as input to the proposed solution (one of both variants) to benefit from the reduction of the transmitted frames to reduce more and more the communication overhead. This is useful to ensure real time MIoT applications with minimum possible overhead in terms of communication, latency and resources. Furthermore, the generalization performance of this model can be validated since enhancement in image restoration has been confirmed with two different image compression algorithms for variable compression ratio. Besides, similar results are obtained by using another compression format. On the other hand, we have tested the proposed solution with RDN and other efficient denoising/super-resolution models and similar results were obtained. The obtained results confirm that any other efficient model can be used instead of RDN for both proposed variants.

VI. Conclusion

In this paper, we aimed to respond better to the hard challenge of communication and resources overhead in the MIoT domain. One solution to cope with the high communicated data size is to use hard lossy multimedia data compression. In this context, a high compression ratio is required to achieve high communication efficiency. However, the current filtering theory cannot fix the hard effect of multimedia compression with a high compression ratio. Our proposed solution consists of reducing the hard visual degradation at the application server by employing a denoising-super resolution DL-based model. Two variants of the proposed scheme were presented: the first one applies the compression with a high compression ratio but without down-scaling the image size, while the second one down-scales the image before applying the lossy compression. The second variant ensures minimum communication overhead at the cost of additional visual degradation compared to the first variant. The experimentation results showed that enhanced decompressed images were obtained (with or without down-scaling). Equally important, the performance analysis confirms the effectiveness of the proposed solution since it reaches a good balance between visual degradation and communication size. Thus, the proposed solution can be considered as an adequate candidate to enhance compressed MIoT transmitted data. Finally, one of the main features of the proposed approach is being flexible, and functional with any multimedia compressor at different compression ratios and down-scaling factors.

Acknowledgment

This work was partially supported by the EIPHI Graduate School (contract "ANR-17-EURE-0002"). The Mesocentre of Franche-Comté provided the computing facilities.

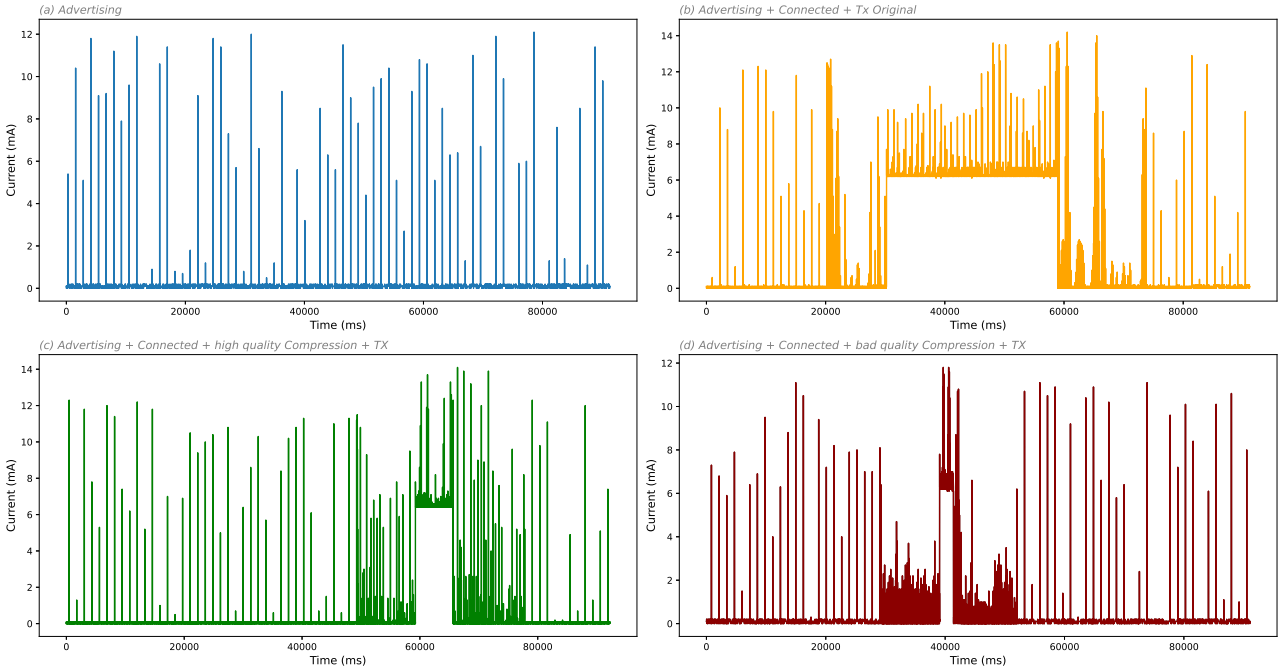


Fig. 16: Current consumption of REDBEAR NANO in milliamps over a 90-seconds period

TABLE III: Total current consumption per time [mA.s] and average current consumption [mA] per second of REDBEAR NANO over a 90-seconds period

	Advertising	Adv + Connected + Tx Original	Adv + Con + Comp HIGH + Tx	Adv + Con + Comp BAD + Tx
Total current consumption (mA.s)	10.76	189.35	47.53	27.56
Average current consumption (mA)	0.12	2.08	0.52	0.30

REFERENCES

[1] Qin Wang, Yanxiao Zhao, Wei Wang, Daniel Minoli, Kazem Sohraby, Hongbo Zhu, and Ben Occhiogrosso. Multimedia iot systems and applications. In *2017 Global Internet of Things Summit (GIoTS)*, pages 1–6. IEEE, 2017.

[2] Mohammed Ali Al-Garadi, Amr Mohamed, Abdulla Al-Ali, Xiaojiang Du, Ihsan Ali, and Mohsen Guizani. A survey of machine and deep learning methods for internet of things (iot) security. *IEEE Communications Surveys & Tutorials*, 2020.

[3] Asra Aslam and Edward Curry. A survey on object detection for the internet of multimedia things (iomt) using deep learning and event-based middleware: approaches, challenges, and future directions. *Image and Vision Computing*, 106:104095, 2021.

[4] Ali Nauman, Yazdan Ahmad Qadri, Muhammad Amjad, Yousaf Bin Zikria, Muhammad Khalil Afzal, and Sung Won Kim. Multimedia internet of things: A comprehensive survey. *IEEE Access*, 8:8202–8250, 2020.

[5] Kaneez Fizza, Abhik Banerjee, Karan Mitra, Prem Prakash Jayaraman, Rajiv Ranjan, Pankesh Patel, and Dimitrios Georgakopoulos. Qoe in iot: a vision, survey and future directions. *Discover Internet of Things*, 1(1):1–14, 2021.

[6] Gregory K Wallace. The jpeg still picture compression standard. *IEEE transactions on consumer electronics*, 38(1):xviii–xxxiv, 1992.

[7] F. Bellard. Bpg image format. https://bellard.org/bpg/, 12 2014.

[8] Chaochao Luo, Zhiyuan Tan, Geyong Min, Jie Gan, Wei Shi, and Zhihong Tian. A novel web attack detection system for internet of things via ensemble classification. *IEEE Transactions on Industrial Informatics*, 17(8):5810–5818, 2021.

[9] Worku J Sori, Jiang Feng, Arero W Godana, Shaohui Liu, and Demissie J Gelmecha. Dfd-net: lung cancer detection from denoised ct scan image using deep learning. *Frontiers of Computer Science*, 15(2):1–13, 2021.

[10] Jing Qiu, Lei Du, Dongwen Zhang, Shen Su, and Zhihong Tian. Nei-tte: intelligent traffic time estimation based on fine-grained time derivation of road segments for smart city. *IEEE Transactions on Industrial Informatics*, 16(4):2659–2666, 2019.

[11] Dongliang Xu, Zhihong Tian, Rufeng Lai, Xiangtao Kong, Zhiyuan Tan, and Wei Shi. Deep learning based emotion analysis of microblog texts. *Information Fusion*, 64:1–11, 2020.

[12] Ning Hu, Zhihong Tian, Xiaojiang Du, Nadra Guizani, and Zhihan Zhu. Deep-green: A dispersed energy-efficiency computing paradigm for green industrial iot. *IEEE Transactions on Green Communications and Networking*, 5(2):750–764, 2021.

[13] He Li, Kaoru Ota, and Mianxiong Dong. Learning iot in edge: Deep learning for the internet of things with edge computing. *IEEE network*, 32(1):96–101, 2018.

[14] Jie Tang, Dawei Sun, Shaoshan Liu, and Jean-Luc Gaudiot. Enabling deep learning on iot devices. *Computer*, 50(10):92–96, 2017.

[15] Chunwei Tian, Yong Xu, Lunke Fei, and Ke Yan. Deep learning for image denoising: a survey. In *International Conference on Genetic and Evolutionary Computing*, pages 563–572. Springer, 2018.

[16] Chunwei Tian, Lunke Fei, Wenxian Zheng, Yong Xu, Wangmeng Zuo, and Chia-Wen Lin. Deep learning on image denoising: An overview. *Neural Networks*, 2020.

[17] K Bai, X Liao, Q Zhang, X Jia, and S Liu. Survey of learning based single image super-resolution reconstruction technology. *Pattern Recognition and Image Analysis*, 30(4):567–577, 2020.

[18] Zhihao Wang, Jian Chen, and Steven CH Hoi. Deep learning for image super-resolution: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[19] Chenggang Yan, Biao Gong, Yuxuan Wei, and Yue Gao. Deep multi-view enhancement hashing for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(4):1445–1451, 2020.

[20] Chenggang Yan, Zhisheng Li, Yongbing Zhang, Yutao Liu, Xiangyang Ji, and Yongdong Zhang. Depth image denoising using nuclear norm and learning graph model. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 16(4):1–17, 2020.

[21] Chenggang Yan, Tong Teng, Yutao Liu, Yongbing Zhang, Haoqian Wang, and Xiangyang Ji. Precise no-reference image quality evaluation based on distortion identification. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 2021.

[22] Chenggang Yan, Yiming Hao, Liang Li, Jian Yin, Anan Liu, Zhendong

Mao, Zhenyu Chen, and Xingyu Gao. Task-adaptive attention for image captioning. *IEEE Transactions on Circuits and Systems for Video Technology*, 2021.

[23] Chenggang Yan, Lixuan Meng, Liang Li, Jiehua Zhang, Jian Yin, Jiyong Zhang, Zhan Wang, Yaoqi Sun, and Bolun Zheng. Age-invariant face recognition by multi-feature fusion and decomposition with self-attention. *ACM Trans Multimed Comput Commun Appl (TOMM)*, 2021.

[24] Athanassios Skodras, Charilaos Christopoulos, and Touradj Ebrahimi. The jpeg 2000 still image compression standard. *IEEE Signal processing magazine*, 18(5):36–58, 2001.

[25] Umar Albalawi, Saraju P Mohanty, and Elias Kougianos. A hardware architecture for better portable graphics (bpg) compression encoder. In *2015 IEEE International Symposium on Nanoelectronic and Information Systems*, pages 291–296. IEEE, 2015.

[26] Vivienne Sze, Madhukar Budagavi, and Gary J Sullivan. High efficiency video coding (hevc). In *Integrated circuit and systems, algorithms and architectures*, volume 39, page 40. Springer, 2014.

[27] Thomas Wiegand, Gary J Sullivan, Gisle Bjontegaard, and Ajay Luthra. Overview of the h. 264/avc video coding standard. *IEEE Transactions on circuits and systems for video technology*, 13(7):560–576, 2003.

[28] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Learning a deep convolutional network for image super-resolution. In *European conference on computer vision*, pages 184–199. Springer, 2014.

[29] Chao Dong, Chen Change Loy, and Xiaoou Tang. Accelerating the super-resolution convolutional neural network. In *European conference on computer vision*, pages 391–407. Springer, 2016.

[30] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1646–1654, 2016.

[31] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Deeply-recursive convolutional network for image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1637–1645, 2016.

[32] Ying Tai, Jian Yang, and Xiaoming Liu. Image super-resolution via deep recursive residual network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3147–3155, 2017.

[33] Ying Tai, Jian Yang, Xiaoming Liu, and Chunyan Xu. Memnet: A persistent memory network for image restoration. In *Proceedings of the IEEE international conference on computer vision*, pages 4539–4547, 2017.

[34] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017.

[35] Tong Tong, Gen Li, Xiejie Liu, and Qinquan Gao. Image super-resolution using dense skip connections. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4799–4807, 2017.

[36] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2472–2481, 2018.

[37] Muhammad Haris, Gregory Shakhnarovich, and Norimichi Ukita. Deep back-projection networks for super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1664–1673, 2018.

[38] Yucheng Wang, Jialiang Shen, and Jian Zhang. Deep bi-dense networks for image super-resolution. In *2018 Digital Image Computing: Techniques and Applications (DICTA)*, pages 1–8. IEEE, 2018.

[39] Biao Li, Yong Shi, Bo Wang, Zhiquan Qi, and Jiabin Liu. Rgsr: A two-step lossy jpg image super-resolution based on noise reduction. *Neurocomputing*, 419:322–334, 2021.

[40] Han Qiu, Qinkai Zheng, Gerard Memmi, Jialiang Lu, Meikang Qiu, and Bhavani Thuraisingham. Deep residual learning-based enhanced jpeg compression in the internet of things. *IEEE Transactions on Industrial Informatics*, 17(3):2124–2133, 2020.

[41] N Krishnaraj, Mohamed Elhoseny, M Thenmozhi, Mahmoud M Selim, and K Shankar. Deep learning model for real-time image compression in internet of underwater things (iout). *Journal of Real-Time Image Processing*, 17(6):2097–2111, 2020.

[42] Vito Walter Anelli, Yashar Deldjoo, Tommaso Di Noia, and Daniele Malitesta. Deep learning-based adaptive image compression system for a real-world scenario. In *2020 IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS)*, pages 1–8. IEEE, 2020.

[43] Joseph Azar, Abdallah Makhoul, Mahmoud Barhamgi, and Raphaël Couturier. An energy efficient iot data compression approach for edge machine learning. *Future Generation Computer Systems*, 96:168–175, 2019.

[44] Fabian Mentzer, George D Toderici, Michael Tschannen, and Eirikur Agustsson. High-fidelity generative image compression. *Advances in Neural Information Processing Systems*, 33, 2020.

[45] True color kodak images. http://r0k.us/graphics/kodak/, 06 2004.

[46] cszn/kair: Image restoration toolbox pytorch training and testing codes for usrnet dncnn ffdnet srmd dpsr esrgan. https://github.com/cszn/KAIR, 10 2020.

[47] Krishan Kumar and Deepti D. Shrimankar. F-des: Fast and deep event summarization. *IEEE Transactions on Multimedia*, 20(2):323–334, 2018.

[48] Krishan Kumar, Deepti D Shrimankar, and Navjot Singh. Eratosthenes sieve based key-frame extraction technique for event summarization in videos. *Multimedia Tools and Applications*, 77(6):7383–7404, 2018.

[49] Krishan Kumar and Deepti D Shrimankar. Deep event learning boost-up approach: Delta. *Multimedia Tools and Applications*, 77(20):26635–26655, 2018.

[50] Krishan Kumar and Deepti D Shrimankar. Esumm: Event summarization on scale-free networks. *IETE Technical Review*, 2018.

[51] Krishan Kumar. Evs-dk: Event video skimming using deep keyframe. *Journal of Visual Communication and Image Representation*, 58:345–352, 2019.

[52] Krishan Kumar. Text query based summarized event searching interface system using deep learning over cloud. *Multimedia Tools and Applications*, 80(7):11079–11094, 2021.