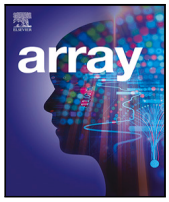
Array 19 (2023) 100314

Contents lists available at ScienceDirect

Array

journal homepage: www.elsevier.com/locate/array



A novel lossy image compression algorithm using multi-models stacked AutoEncoders

Salam Fraihat *, Mohammed Azmi Al-Betar

Artificial Intelligence Research Center (AIRC), College of Engineering and Information Technology, Ajman University, Ajman P.O. Box 346, United Arab Emirates

ARTICLE INFO

Keywords:
Deep learning
Stacked AutoEncoder
Lossy image compression
Image classification
Image processing

ABSTRACT

The extensive use of images in many fields increased the demand for image compression algorithms to overcome the transfer bandwidth and storage limitations. With image compression, disk space, and transmission speed can be efficiently reduced. Some of the traditional techniques used for image compression are the JPEG and ZIP formats. The compression rate (CR) in JPEG can be high but to the detriment of the quality factor of the image. ZIP has a low compression rate, where the quality remains almost unaffected. Machine learning (ML) is considered an essential technique for image compression using different algorithms. The most widely used algorithm is Deep Learning (DL), which represents the features of the image at different scales by using different types of layers. In this research, an AutoEncoder (AE) deep learning-based compression algorithm is proposed for lossy image compression and experimented with using three standard dataset types: MNIST, Grayscale, and Color images datasets. A Stacked AE (SAE) for image compression and a binarized content-based image filter are used with a high compression rate while keeping the quality above 85% using structural similarity index metric (SSIM) compared to traditional techniques. In addition, a convolutional neural network (CNN) classification model has been utilized as SAEs compression model selector for each image class. Experimental results demonstrate that the proposed SAE image compression algorithm outperforms the JPEG-encoded algorithm in terms of compression rate (CR) and image quality. The CR that the proposed model achieved with an acceptable reconstruction accuracy was about 85%, which is almost 20% higher than the standard JPEG's compression rate, with an accuracy of 94.63% SSIM score.

1. Introduction

A camera's capabilities and quality rapidly advance, image size becomes more significant as they contain more pixels. With the eruption of social media use (such as Facebook, Instagram, and Snapchat) over the past few years, images rapidly form substantial data volumes that eventually need to be stored on storage drives. Media, articles, and information must be well kept for customers, as client data is the key to most (if not all) businesses in our century. The need for image compression algorithms becomes vital to reduce the image size to store even more images using fewer resources. Creating thumbnails and content for smaller screens and tablets is also needed to minimize Internet traffic overload and computation time for slower devices. Many algorithms that are widely used for compressing text, files, and images are found in the literature, one of which is the Deflate Compression [1] used for Zip file formats. It is used on various platforms for its ease of use when sharing files online or on a flash drive. The use of these tools can ensure compression speed due to the absence of a training phase before compression. However, the CR for compressing images is modest. Embedded Block Coding with Optimized Truncation (EBCOT) [2] is another technique for image compression, where the algorithm exhibits state-of-the-art compression performance while producing a bit-stream with a rich set of features, including resolution and SNR scalability together with a "random access" property. Some compression techniques could be used for other uses besides the ones mentioned earlier, such as fractional image compression, using an algorithm to transform image bits. For example, printing the same image on the same paper multiple times in iterations, each with a different size and orientation, with a condition of the attractor, removing the spread-out images. Machine Learning (ML) has emerged as a tool of Artificial Intelligence (AI) [3], which allows systems to learn, improve, and predict from experience automatically. Continuously provided with data, the machine learning algorithms can give more accurate results for the application.

Machine learning algorithms are used for classification, clustering, and regression problems through supervised and unsupervised learning [4]. Deep Learning (DL) [5] is a learning technique that mimics the human way of thinking through neural networks (NN). It continuously analyzes data with a logical structure to detect and predict patterns

* Corresponding author.
E-mail address: s.fraihat@ajman.ac.ae (S. Fraihat).

https://doi.org/10.1016/j.array.2023.100314
Received 19 June 2023; Received in revised form 21 July 2023; Accepted 26 July 2023
Available online 28 July 2023

and compare them to known objects. DL can perform all ML tasks from classification to clustering. Also, ML can only perform a few of the DL tasks. This is due to the needless feature extraction in the DL, while the ML needs to perform feature extraction on the data at the preprocessing step before feeding the model. Feature extraction is a step that a domain expert usually does to select or derive an informative and non-redundant set of values from being used for the model. DL is used for image compression using different algorithms. The most widely used algorithm is the convolutional neural network (CNN) which represents the features of the image at different scales and by using pooling layers, which summarize the features of each image region. However, the image compressed and decompressed using a CNN model is found to be blurry due to loss of information during the compression (encoding) process.

This research paper uses a Stacked AutoEncoders (SAE) deep learning model and a content-based image filter to compress the image. SAE is a feed-forward-based type of NN, where the output is the same as the input. It consists of an encoder, a decoder, and a loss function. In the training phase, SAE transforms the bits into an encoded (compressed) format with fewer bits called latent-space representation (code) using the encoder and then reconstructs it into a lossy image using the decoder with the same shape. In the testing phase, the code alone is used in the decoder model to be reconstructed, as the weights and underlying functions (activation functions) are kept in the trained model. The model is trained using Back Propagation (BP) algorithm. BP is a supervised algorithm that tunes and enhances the output by adjusting the hyperparameters. On the another hand, a content-based image filter is used to enhance the quality of the reconstructed image generated using the SAE decoder.

Furthermore, this research adds an image classifier to the proposed compression technique to choose the AE that serves that particular image class. The proposed image compression algorithm reduces data size by removing redundant and excessive information. This process reduces the cost of storing and transmitting images, especially over low-bandwidth networks. The main objective of the proposed approach is to considerably increase the compression ratio beyond the existing compression algorithms, such as Linear Lossy Data Compression (e.g., JPEG), along with an image classifier to identify the class of new images that need compression.

Our contributions to this paper are summarized as follows:

- We present a novel lightweight image compression approach that aims to compress and decompress images without processing their content, in contrast to most state-of-the-art approaches based on complex deep learning models such as convolutional, Recurrent and Transformer models, which capture spatial features such as edges, corners, and textures from the image.
- The proposed approach demonstrates that incorporating a lightweight binary filter substantially improves the image reconstruction process with comparable quality to that of convolutional models, which rely on more complex convolutional layers requiring training for image reconstruction.
- The CNN image classifier used in the proposed method, is employed as a model selector to determine which pre-trained SAE to use for compressing and decompressing the image based on its content. This approach serves as a good alternative when the required hardware resources are not available to train a very large single SAE model that can compress and decompress any image irrespective of its content.

The rest of the paper is structured as follows: Section 2 presents the literature review of image compression. DL algorithms are analyzed, showing the early techniques of image compression till the recent techniques that employed DL. The proposed methodology, including background on the base method used and the complete architecture of the image classifier, compression, and decompression algorithms, are given in Section 3. The experimental results and discussion of the results obtained are provided in Section 4. Finally, Section 5 concludes the research findings and offers possible future directions.

2. Background and related work

2.1. Autoencoder

The Autoencoder (AE) is an unsupervised deep learning algorithm that aims to transform the input into output while minimizing the amount of distortion [6]. Different AE architectures can be used for many purposes, including feature extraction and selection, data denoising, and data compression. Fig. 1 shows a traditional AE consisting of an encoder, latent representation, and a decoder. The number of output nodes is equal to the number of input nodes, and generally, the number of neurons in the hidden layers is less than the number of neurons in the input and output layers. A simple AE consists of several Restricted Boltzmann Machines (RBMs) [7]. RBM is a particular type of Boltzmann machine having two layers (visible and hidden layers), where it connects all nodes from the inputs only (visible layer) to all nodes from the next layer (hidden layer) instead of connecting all nodes, which is restricted by not connecting nodes within the same layer.

An AE takes an input $x \in R^d$, where d is the input dimensions, and first maps it to the latent representation (hidden layer) $h \in R^{d'}$ using a deterministic function, Eq. (1), with parameters $\theta = \{W, b\}$.

$$h = f_\theta = \sigma(Wx + b) \tag{1}$$

where W is $d \times d'$ weight matrix and a bias vector is represented by b. The weights W are multiplied by the inputs X and added to it the bias vector to not be a 0. As a result, in Eq. (2), the latent representation h will be mapped back to the reconstructed vector in the input space.

$$\widetilde{x} = f'_\theta = \sigma(W'h + b') \tag{2}$$

having W', b' as parameters. The reverse mapping's weight matrix W' is constrained by $W' = W^T$, where the T implies that the AE has tied weights [8]. Furthermore, the appropriate cost function will be minimized over the training set $D_n = \{(x_0, t_0), \ldots, (x_n, t_n)\}$ by optimizing the parameters W. The Back-propagation algorithm [8] is used to fine-tune the weights to minimize the loss rate, where the loss rate is calculated as the difference between the actual output of the AE and the predicted output as Mean Squared Error (MSE). The equation for the loss function is shown in Eq. (3) where $\widetilde{Y}$ is the predicted value, and the Y is the actual value. In the training process, Y represents the input X, and $\widetilde{Y}$ represents the output $\widetilde{X}$.

$$L = \sum (Y - \widetilde{Y})^2 \tag{3}$$

After calculating the Total Loss (L_t) shown in Eq. (3), the weights are updated by propagating the loss rate over the weights matrix using the Gradient Descent technique.

As shown in Fig. 1, a stacked AutoEncoder (SAE) is a deep AE where multiple encoders are stacked on top of one another, and multiple decoders are stacked on top of one another. SAE must have three or more hidden layers to achieve "stacked" meaning and eliminate the singularity of hidden layers. The number of neurons in each layer must be smaller than the layers preceding it and more significant than the following layer (the inner layer has the least number of neurons). As the SAE is based on one of the types of AE, it uses the AE's activation function with the weights and inputs as parameters. The SAE also uses the backpropagation algorithm, having MSE calculate the loss function.

2.2. Lossy image compression

There are mainly two types of lossy image compression [9] based on the image compression technique: Linear and Non-Linear. The most popular linear lossy image compression algorithm is JPEG [10]. JPEG encodes images and graphics using Discrete Cosine Transform-Based coding (DCT) [10] for coding transformation. The algorithm quantizes the image into low frequency, followed by a Zigzag scan process for

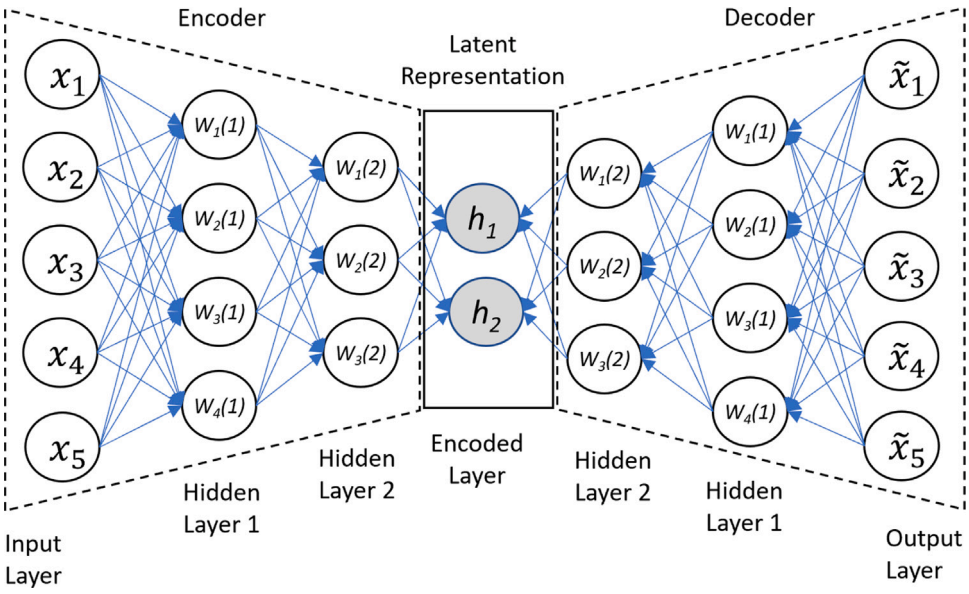


Fig. 1. Stacked Autoencoder architecture.

grouping the low-frequency coefficients. After that, an entropy encoder is used to compress the image, such as Run-Length Encoding and Huffman coding [11].

A non-linear lossy image compression technique is based on DL models. DL has a significant influence on numerous research fields. DL models' capability of feature extractions caused a revolution in the image processing domain. Consequently, many DL architectures were proposed to compress images. The basic form of DL for image compression is to feed the DL model with images for training and then to test the model with new images and check its accuracy. Several deep-learning algorithms are used for image compressions, such as AutoEncoder, Generative Adversarial Networks (GAN), Recurrent Neural Networks (RNN), and Convolutional Neural Networks (CNN).

2.3. Deep learning based image compression

Recent works on various DL architectures are explored in this section, including Auto-encoders (AEs), variational auto-encoders (VAEs), convolutional neural networks (CNNs), long short-term memories (LSTMs), gated recurrent units (GRUs), generative adversarial networks (GANs), and transformers.

AE has become popular in image compression as it aims to reduce the dimensionality of the input image. In [12], Ollivier et al. used AE to compress data by minimizing the code length and utilizing the generative property of AE. In other work, Sento employed AE with techniques like Kalman filter for the compression of MNIST images to enhance the training process and improve the compression performance of MNIST images [13]. However, this model showed a weak performance for red-green-blue (RGB) images. AE has been used to compress images and has been shown to perform better than traditional compression methods such as JPEG, JPEG-2000, and WebP. Additionally, AE has been utilized in the compression of high-resolution images. Correspondingly, in [14], the Kodak dataset was used by authors to train AE and has been shown to perform better than traditional compression methods such as JPEG. In [15,16], and [17], the authors proposed AE for image compression. They used generalized divisive normalization (GDN) and inverse generalized divisive normalization (IGDN) activation functions as an alternative to the rectified linear unit (ReLU) activation function to boost the training process. Cheng et al. [18] and Alexandre et al. [19] built AE for high-resolution image compression.

In [20], Fei Hu et al. used a stacked autoencoder SAE defined with AE five-layer and Chaotic Logistic Map (CLM) encryption model to compress the image. The authors used the CLM to generate the sequence map and then applied XOR on the compressed image to encrypt it.

The variational autoencoder VAE is another type of AE used for image compression. VAE uses the mean and variance as input of latent space distributions and has been shown to perform better than simple Autoencoders. Many researchers employed VAE for image compression using nonlinear transforms and uniform quantization techniques. For example, Zhou et al. used VAE with the Challenge on Learned Image Compression (CLIC) dataset for training the model [21]. The model was enormous and complex due to multiple training parameters. Similarly, Chen et al. proposed a VAE-based architecture for high-resolution image compression, which utilized a non-local attention module to improve the training process, but at the cost of increased model complexity [22]. In [23], the authors proposed a VAE for image compression, and in [24], VAE was used to compress images and achieve a 4.10 bits per pixel rate, but with a complex model architecture.

Convolutional AutoEncoder (CAE) is instrumental in image compression due to its success in image processing. Layers used in CAE [25] are convolution layers rather than fully connected ones in the encoding section and deconvolution layers in the decoding section. In general, CAE generates blurry images because it is designed to reduce the dimensionality of the input image by encoding its most important features in a compact representation and then reconstructing the image from this compact representation. The encoding and reconstruction process may result in a loss of information and the generation of a blurry image. This is particularly the case when the autoencoder is trained on a large dataset, as it will learn to encode and reconstruct the average features of the images in the dataset, which can lead to a loss of detail and the generation of blurry images. The paper introduces a sub-pixel convolutional generative adversarial network to reconstruct compressed sensing images with a compound loss function and multiple building blocks. The algorithm outperforms conventional and deep learning-based reconstruction algorithms on multiple datasets. The authors suggest enhancing the network structure by introducing residual connections and attention mechanisms and extending it to compressed sensing of image sequences. In [26], Tawfik et al. introduces a neural network-based lossy compression architecture using convolutional autoencoders. In [27] Nan Wang et al. propose a new learning method for the compression autoencoder based on transfer learning and incremental learning. D. Sebai et al. [28] present a loss-conditional autoencoder

tailored to the specific task of semantic image understanding to achieve higher visual quality in lossy variable-rate compression. Yubao Sun et al. [29] propose a sub-pixel convolutional generative adversarial network to reconstruct compressed sensing images with a compound loss function and multiple building blocks. The algorithm outperforms conventional and deep learning-based reconstruction algorithms on MNIST, F-MNIST and CelebA datasets. The authors suggest enhancing the network structure by introducing residual connections and attention mechanisms and extending it to compressed sensing of image sequences. Alex Krizhevsky et al. [30] applied CAE on the ImageNet dataset of 1.2 million images of high resolution to have a classifier with 1000 different classes. Their classifier had three fully connected layers and five convolutional layers, and max-pooling layers were followed by fully connected layers. Their CAE achieved 37.5% on error rates while the best was 47.1% using Sparse AE and 45.7% using Fisher Vectors. Zhengxue Cheng et al. [18] introduced an image compression approach based on Deep CAE, having convolutional–deconvolutional filters for the symmetric CAE network. Their model used a rate–distortion (RD) loss function to train the CAE greedily and a uniform noise added to imitate quantization noises at the optimization process.

Recently, Chowdary et al. in [31] used a nonnegative Tucker decomposition NTD based on GRUs (Gated Recurrent Units) for compressing satellite images. Their approach resulted in a notable 56.40% enhancement in computing efficiency and compromised just-0.58 dB of PSNR. In [32] Wang et al. used convolutional LSTMs for compressing multispectral images. The proposed model demonstrated superior performance compared to JPEG2000. However, the framework was not compared to other end-to-end state-of-the-art image compression methods. Li et al. in [33] investigated the utilization of advanced learned transforms, particularly Vision Transformers, for variable rate image compression. The results demonstrated that their model surpassed BPG (4:4:4) in terms of Structural Similarity Index (SIM) across a range of bit rates from 0.021 to 0.21 bpp. In [34] Qian et al. introduced a vision transformer-based entropy model named Entroformer. This model effectively captured long-range dependencies in probability distribution estimation. The performance of Entroformer was demonstrated on the Kodak dataset, where it achieved an average peak signal-to-noise ratio (PSNR) of 27.63 dB and a multiscale structural similarity (MS-SSIM) of 0.90132 when optimized for the mean squared error (MSE) loss function. However, the method encountered an out of memory (OOM) problem during the encoding and decoding of high-resolution images. In [35] Li, B et al. propose an Encoding (ROI) Region Of Interest-based deep image compression framework using transformers. They integrate a binary ROI mask into various layers of the network to provide spatial information guidance. The experimental results was demonstrated a higher PSNR while maintaining a reasonable average PSNR.

3. Methodology

This section presents the architecture of the research methodology, starting with an illustration of the proposed system's architecture. As shown in Fig. 2, the proposed system consists of two main processes: (1) The compression process, and (2) Decompressing process.

3.1. SAE training

The proposed image compression algorithm employs a trained SAE model using an image dataset. SAE consists of stacked neural network layers fully connected, where each layer's output is the input for the next layer. The reason behind using SAE instead of CAE is that the performance of CAE decreases as the number of filters (feature maps) is more significant, resulting in a drawback that needs to be avoided, as AE already has its time overhead with the training time. Also, the sparse AE will not be suitable for image compression with only one hidden layer. Even if used, the CR will be at its minimum, and quality during the reconstruction phase will be lost. VAE is used mainly for generating new images [36], such as a derivative of multiple images to form a new, for example, design of an item or a new scene in video games, so it is not an excellent choice to use it for image compression.

As shown in Fig. 2, the SAE consist of (a) Encoder layers (Input layer, Layer1 and layer2) used to compress the image (b) Encoded layer (latent layer) represent the image compression vector ICV and (c) Decoder layers (layer 1, layer2 and Decoder output layer) used to decompress the image.

The SAE training is done by passing all images dataset by all layers of the model (a, b, and c), and by applying BP to adjust the weights for each neuron to have a more accurate result for the reconstructed image, which means the original input image is close to the reconstructed output image.

As activation functions, the ReLU and Sigmoid have been shown to outperform the other activation functions experimentally. ReLU is used in all the hidden layers for its inexpensive performance and simple mathematics. It returns the value of 0 if the value is negative; otherwise, it will return the same value as in Eq. (4), where a is the input of the activation function.

$$f(a) = Max(0, a) \tag{4}$$

The sigmoid activation function is used in only the last hidden layer for the values to be within a gradient range instead of having either 0 or a fixed value in the ReLU.

3.2. Compression process

Once the SAE model is trained, it will be used to compress the image as shown in Fig. 2, where the original input image is fed to the SAE model to generate the reconstructed image. The compressed image is represented through two structures:

1. Image compression vector (MCV): The output of the encoded layer called the Image Compression vector of the original image. The MCV length is the same as the number of neurons on the encoded layer (Latent Representation).
2. Residual Error Vector (REV): as shown in Fig. 2, the reconstructed image is slightly blurred due to the loss of quality throughout the original image. In order to minimize this loss during the image decompression, the Residual Error Vector is extracted from the difference between the reconstructed image and the original image. The REV was saved to be merged with the reconstructed image during the decompression process.

3.3. Decompression process

For the decompression process, the trained SAE decoder layers are utilized as a module to reconstruct the compressed image. As depicted in Fig. 2, the input to the decoder is the compressed image representation (MCV), and the output is the reconstructed image. The purpose of this stage is to generate an image that closely resembles the original image. The SAE model's hidden layers, which play a crucial role in the decoding process, contain nodes with fixed sizes. In our implementation, the number of hidden layers and the number of nodes in each layer were determined experimentally and are typically floating-point values.

To enhance the quality of the reconstructed image, an additional step called the Residual Enhancement Vector (REV) is introduced. By adding the REV, the decompression phase aims to achieve a reconstructed image that is as close as possible to the original image in terms of visual quality and details. This approach helps to enhance the overall performance of the compression/decompression algorithm.

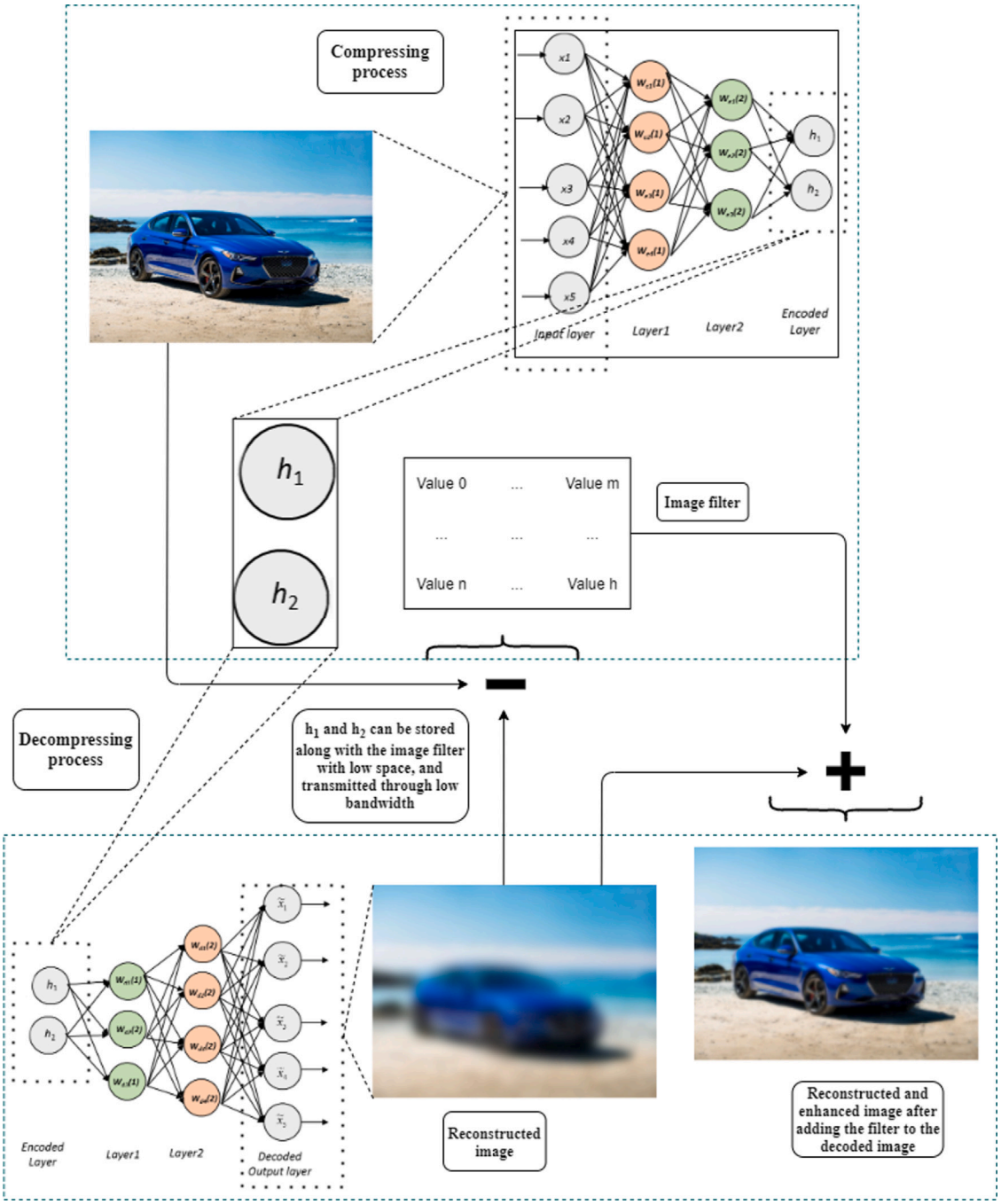


Fig. 2. Proposed image compression architecture.

3.4. Compressed image quality enhancement

The resulting REV from the compression process is the same size as the original image. In order to reduce the REV size in memory, it was transformed and binarized to have a smaller range of numbers by eliminating values with the least density. The following steps are to binarize REV to have a minimum size and then enhance the compression rate.

3.4.1. Values quantization

In order to reduce the REV values range which decreases the number of bits to save REV, the following steps are applied:

1. The values were rounded to have one decimal point, then multiplied by 128 to deal with them as integers.
2. The values were quantized; values that fall between 0 and 10 will become 5, values between 10 and 20 will become 15, and so on. The values that are greater than 35 are set to 35, and values smaller than −35 are set to −35. The resulting range of values is around 7 to 9 values -35, -25, -15, -5, 0, 5, 15, 25, 35 as shown in Fig. 3.

Values that were eliminated do not impact the REV since the frequency of these values is very low compared to other values with higher frequencies. As shown in Fig. 3, the distribution of the values is

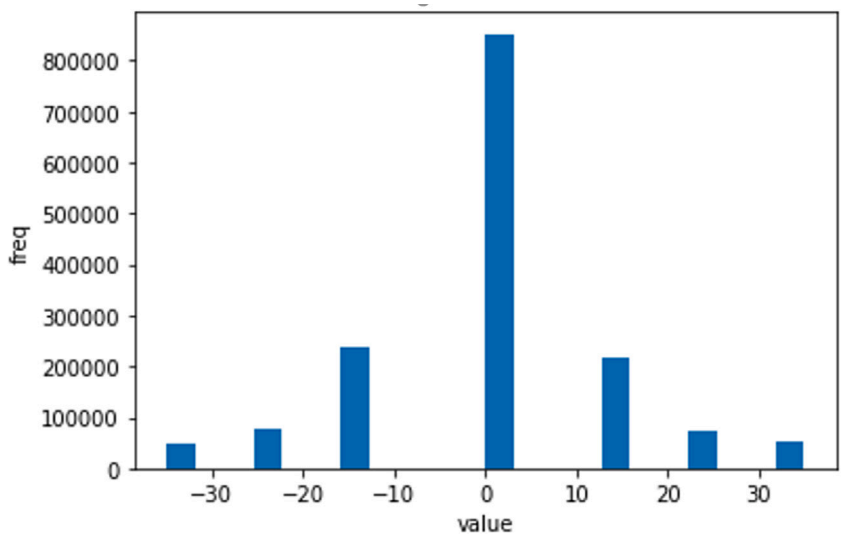


Fig. 3. Histogram for images value's density after range reduction.

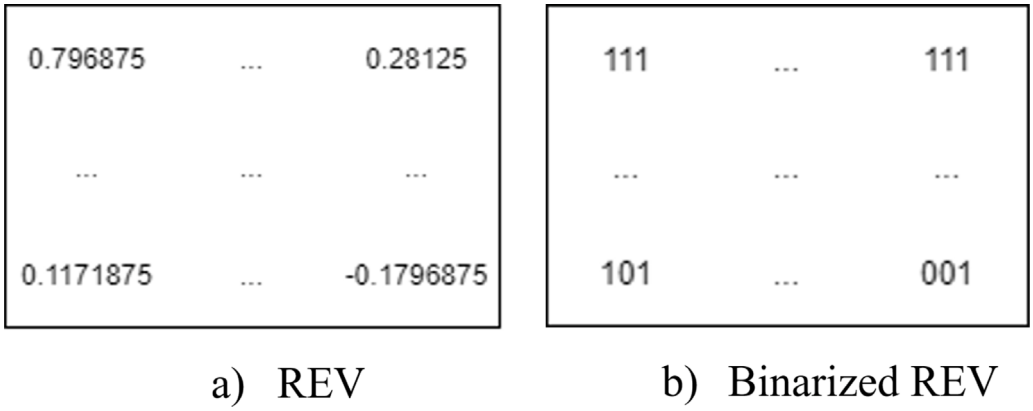


Fig. 4. Binarized REV vectors for enhancement (BREV).

a normal distribution where the most right and most left values have the least frequencies.

3.4.2. REV binary coding (BREV)

The REV is then scaled into values between 0–7 and converted to a binary representation of 3 bits, having 8 values (2^3), where 0 is represented as 000, 1 as 001, 2 as 010, and so on until 7 is represented as 111. Fig. 4 shows the initial and final REV vectors used in the proposed model.

The BREV size matches the image size, but the values it contains use fewer bits, and that is where the size is reduced.

In the decompression process, the REV was added to the reconstructed image after rescaling it to values between −35 and 35 and dividing them by 128 to retrieve the original values of the REV before binarizing it (BREV). The result is an enhanced image with much better quality than the decoded one alone. The BREV will be stored/sent having a minimum size as it can be saved as a binary file of 3 bits for each pixel instead of 8 bits initially (0–255) alongside the latent representation.

3.5. New images compression

The SAE models can compress and decompress the image class trained for. It performs well with these images used in the training process. Furthermore, it is preferable to train a single SAE model for each image class which helps to reconstruct an image close to the original input image. There are two scenarios that the proposed approach will go through:

1. For new test images, an image classifier is added to select the trained SAE model for compressing and decompressing the image (such as cars, landscape, and people), as shown in Fig. 5.
2. When there is no pre-trained SAE model available for a new image class, a new SAE will be trained from scratch using a set of images from the same class. This adaptive training process allows the model to create a specific SAE model tailored for the new image class, enabling effective compression and decompression.

4. Experiments and results

The SAE image compressing model has been trained on Google's Collaboratory (Colab), where the code run on GPUs with 12.72 GB of ram, 2.2 MHz CPU speed, and 15 GB of GPU memory. The SAE model has been implemented using Python programming language and TensorFlow and Keras libraries and OpenCV for image processing.

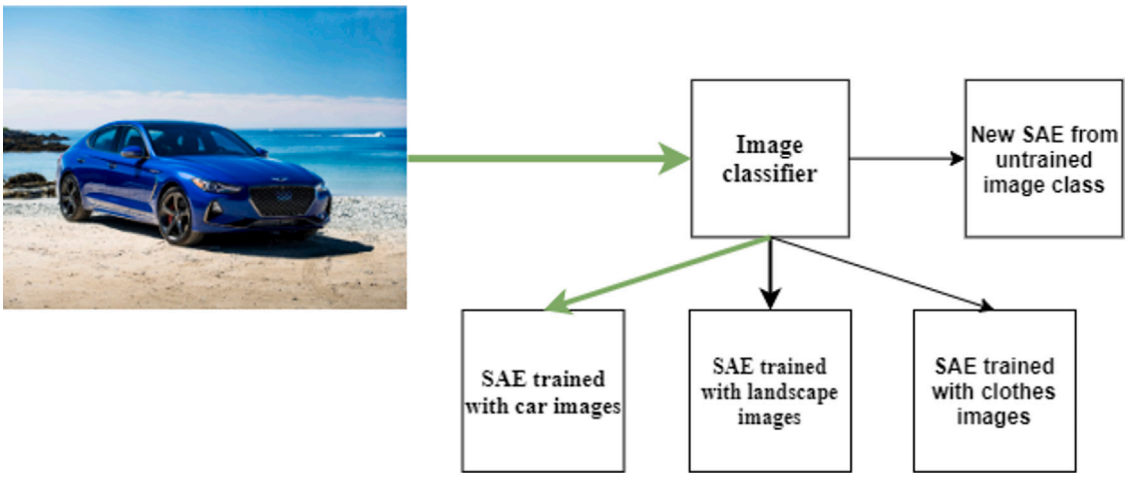


Fig. 5. SAEs selection based on image class.

Table 1
Datasets characteristics..

Dataset name	# of images	Image size	# of images training set	# of images testing set
MNIST	70,000	28 by 28	60,000	10,000
Grayscale car images	156,370	512 by 512	59,500	10,500
Color car images	64,467	320 by 220	54,796	9670

4.1. Dataset description

The datasets used for the experiments are:

- MNIST dataset [37] from Keras as the baseline dataset. The MNIST dataset is a widely used benchmark dataset in computer vision. It consists of a set of 70,000 grayscale images of handwritten digits from 0 to 9, with 10,000 images for testing and 60,000 images for training. Each image is of size 28 × 28 pixels.
- Grayscale Images experimented with the dataset "all_car_images_dhcl" [38] dataset from Kaggle.com owned by "DmitryPak". This dataset consists of approximately 70,000 grayscale images of cars, each with a size of 512 × 512 pixels. The TCC dataset is specifically utilized for conducting experiments involving grayscale images with 10,000 images for testing and 60,000 images for training.
- The "Car Connection TCC dataset" was obtained through web scraping of thecarconnection.com, which is a car review and comparison website [39]. This dataset consists of approximately 65,000 images of cars, each with a size of 320 × 220 pixels. The TCC dataset is specifically utilized for conducting experiments involving colored images with 10,000 images for testing and 55,000 images for training.

Table 1 shows the details for each dataset. The size of the datasets was reduced to match the size of the baseline MNIST dataset. Due to the lack of high resources, images were resized into different shapes in each experiment to have a good performance during the training of the SAE models. Then the datasets were split into an 85% training set and a 15% testing set.

4.2. Evaluation measures

As a measurement for evaluating the image compression/decompression quality, the similarity between images can be calculated using the widely used metrics: (1) Peak signal-to-noise ratio (PSNR) (2) structural index similarity (SSIM) [40]. PSNR uses MSE to be calculated as in Eq. (5) [40] where max is the highest scale value of a grayscale.

$$PSNR = 10log10\left(\frac{max^2}{MSE}\right) \tag{5}$$

SSIM is calculated on various windows of an image. The measure between two windows x and y of common size $N \times N$ is calculated based on three factors, such as s structure, l luminance, and c contrast, that are better suited to work with the human visual system. The SSIM has a maximum value of 1, and it is calculated by Eqs. (6), (7), (8), (9) [40].

$$SSIM(x, y) = l(x, y).c(x, y).s(x, y) \tag{6}$$

$$l(x, y) = \frac{2\mu_x\mu_y + c1}{{\mu_x}^2 + {\mu_y}^2 + c1} \tag{7}$$

$$c(x, y) = \frac{2\sigma_x\sigma_y + c2}{\sigma_x^2 + \sigma_y^2 + c2} \tag{8}$$

$$s(x, y) = \frac{\sigma_{xy} + c3}{\sigma_x\sigma_y + c3} \tag{9}$$

with:

- μ_x the pixel sample mean of x;
- μ_y the pixel sample mean of y;
- σ_x^2 the variance of x;
- σ_y^2 the variance of y;
- σ_{xy} the covariance of x and y;

where $c1$, $c2$, and $c3$ are constant values that are used to avoid having a zero in the denominators, usually values of $c1 = (0.01 \times 255)^2, c2 = (0.03 \times 255)^2$, and $c3 = c2/2$ are recommended to be used [40]. Eq. (7) calculates the luminance by comparing the original and reconstructed images. It returns a value from 0–1; if both images are equal, it returns a maximum of 1. Eq. (8) calculates the contrast by comparing the original and reconstructed images based on the standard deviation. It returns a value from 0–1 and returns the maximum value of 1 when the images are identical. Lastly, Eq. (9) is used to calculate the structure of the images by comparing the original and reconstructed images based on the covariance.

The compression rate (CR) measures the relative reduction in the image size in bytes without degrading its quality. CR is calculated by

Table 2
Summary of the best experiments applied on MNIST, grayscale, and colored car images datasets.

Exp Nbr	Dataset/image size	Nbr hidden layer	Nbr of neurons per layer		
			Encoding layers	Latent layer	Decoding abbr layers
1	MNIST 28 × 28	7	784-392-196-49	32	49-196-392-784
2			784-392-112-56	28	56-112-392-784
3		5	784-196-98	28	98-196-784
4			784-392-112	32	112-392-784
5	Grayscale car images 64 × 64	7	4096-1024-256-64	16	64-256-1024-4096
6			4096-324-64-16	8	16-64-324-4096
7		5	4096-512-64	16	64-512-4096
8			4096-324-32	8	32-324-4096
9	Color car images 64 × 64 × 3	7	12288-1944-486-192	48	192-486-1944-12288
10			12288-3072-768-192	24	192-768-3072-12288
11		5	12288-972-96	48	96-972-12288
12			12288-1536-192	24	192-1536-12288

adding the BREV image size to the MCV size in bytes and dividing the sum by the original image size in bytes, as in Eq. (10).

$$Compression\ Rate = (1 - \frac{Binary\ Residual\ Error\ Vector\ size\ +\ image\ compression\ vector\ size}{Original\ image\ size}) \times 100 \quad (10)$$

4.3. Experiments design

Experiments were applied using MNIST, Grayscale, and Colored images datasets. Many attempts were made using different numbers of layers, and neurons per layer to find the best model. Table 2 presents the configuration summary of the best experiments applied on MNIST, Grayscale, and Colored car images datasets.

Since the SAE model is nonlinear, the error rate and accuracy of each reconstructed image compared to the original image were slightly different. The SSIM, PSNR, and CR values were averaged for each dataset's proposed, reconstructed, and JPEG-encoded images.

The image classifier shown in Fig. 5, used as an SAE selector based on the image class, was implemented using CNN. It consists of three convolutional layers. The convolutional layers are configured to have 16, 32, and 64 filters, respectively, a kernel size of 5 × 5, and a stride of 1. A max-pooling layer follows each convolutional layer. The output of the last max pooling layer is fed into a fully connected layer with ReLU activation. A dropout layer with a rate of 0.2 was used to avoid overfitting.

4.4. Comparative evaluation

It is worth clarifying here that the reconstructed image is the image constructed by feeding the MVC to the SAE decoder (decompressor) without enhancement using BREV. The enhanced image is constructed by adding BREV to the reconstructed image.

In Table 3, the original images of the MNIST dataset were compared against the reconstructed images and JPEG-encoded images. As shown in Table 3, despite having a smaller MCV size than the JPEG encoded images, the proposed model had a better SSIM score of 94.63%, while the JPEG encoding had 69.45% SSIM score, which is around 25% better. The size of the original MNIST images in memory is 586.68 bytes, and the JPEG image size is 200 bytes resulting in a CR of 65.91% with a (2.93:1) image quality factor. The size of the reconstructed image is the number of bytes used to store the MVC vector in memory.

Experiment 1 showed that the proposed model has a 24.61% better SSIM score than the JPEG-encoded images and a 4.09 dB better PSNR score. Experiment 2 showed that the proposed model has a 24.66% better SSIM score than the JPEG-encoded images and a 4.09 dB better PSNR score that matches experiment 1. Experiment 3 showed that the proposed model has a 23.62% better SSIM score than the JPEG-encoded images and a 3.38 dB better PSNR score. Experiment 4 showed that the proposed model has a 25.18% better SSIM score than the JPEG-encoded images and a 4.48 dB better PSNR score which is the best result.

Another observation from Table 3 is that the number of layers did not significantly impact the performance compression of the MNIST images type.

Fig. 6 shows samples from MNIST experiments. To show the differences visually, the original image is compared with the reconstructed and the JPEG encoded image with a quality factor of 1 in all experiments since it is the closest image size to the compressed size of the proposed model. Images decoded using the proposed model are visually better than the JPEG encoded in all four experiments. No enhancement was needed since it was already good enough with high SSIM and PSNR scores.

Table 4, shows a comparison of experiments for the grayscale car image sizes and CR. The original size is 711.48 bytes on average for 64 by 64 pixels for each image.

As shown in Table 5 the grayscale images with more details, it is noticeable the enhancement in quality that the use of BREV add to the reconstruction of the decompressed image in term of SSIM and PSNR scores for the grayscale images dataset. Experiment 5 showed that the enhanced image has almost the same SSIM as the JPEG encoded images with 90.50%, and 1.09 dB better PSNR score than the JPEG encoded images, which is the best PSNR score for grayscale images. Experiment 6 showed that the enhanced images have almost the same SSIM score as the JPEG-encoded, with 90.50% and 90.26%, respectively, and have almost the same CR with 85.51%. The enhanced image has a 1.05 dB better PSNR score than the JPEG-encoded images. Experiment 7 showed that the enhanced image has almost the same SSIM score and almost the same PSNR score as the JPEG-encoded images. Experiment 8 showed that the JPEG encoded image is only 0.74% better SSIM score than the enhanced image with less than 0.8% of CR, which is the best CR score for grayscale images. The enhanced image has a 0.16 dB better PSNR score than the JPEG-encoded images.

The same observation applies to the MNIST image type, form Table 4, it is clear that the number of layers did not significantly impact the performance compression of the grayscale images type.

Next, samples from the grayscale images with more details are shown in Fig. 7. In each experiment, an original image from the grayscale dataset is compared with the reconstructed, enhanced, and JPEG-encoded image with a quality factor of 45 to show the differences visually.

As can be observed, the reconstructed images were slightly different, having almost the same quality but with a loss in different parts of the images.

As shown in Tables 6 and 7, the color image with more details, experiment 9 showed that the JPEG encoded images have only 9%

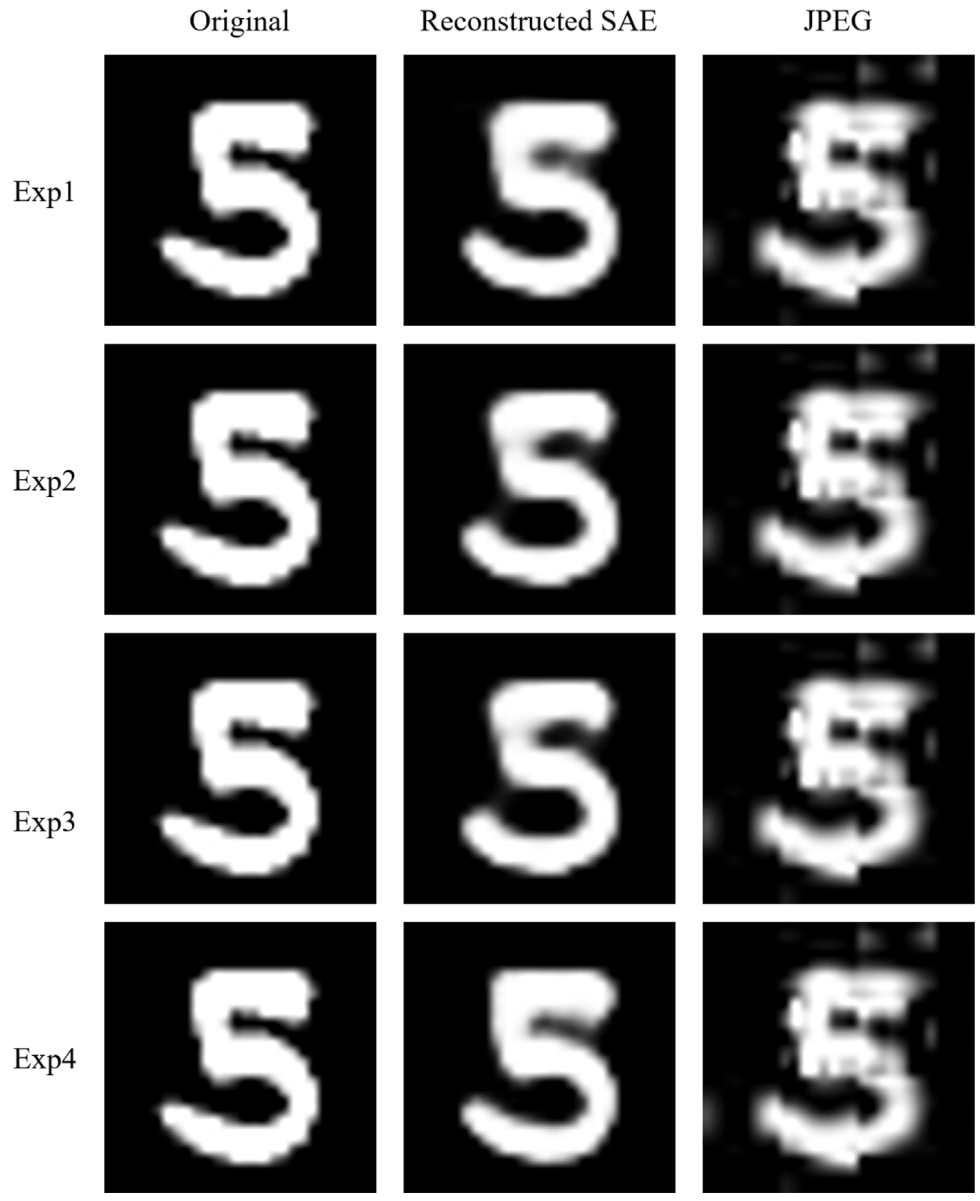


Fig. 6. Original, Reconstructed, and JPEG encoded image of MNIST images dataset.

Table 3
MNIST data SSIM, PSNR and CR comparison.

	Size (Bytes)		CR		SSIM		PSNR	
	Reconstructed	JPEG	Reconstructed	JPEG	Reconstructed	JPEG	Reconstructed	JPEG
Exp 1	85.94	200	85.35%	65.91%	94.06%	69.45%	71.24 dB	67.15 dB
Exp 2	85.80		**85.37%**		94.11%		71.24 dB	
Exp 3	88.37		84.94%		93.07%		70.53 dB	
Exp 4	94.86		83.83%		**94.63%**		**71.63 dB**	

Table 4
Grayscale car images average size and CR comparison.

	Size (Bytes)				CR		
	Original	Reconstructed	Enhanced	JPEG	Reconstructed	Enhanced	JPEG
Exp 5	6778	52.74	1194.38	981.52	99.22%	82.37%	85.51%
Exp 6		23.31	986.86		99.66%	85.44%	
Exp 7		43.16	1133.41		99.36%	83.27%	
Exp 8		19.98	927.49		**99.71%**	**86.31%**	

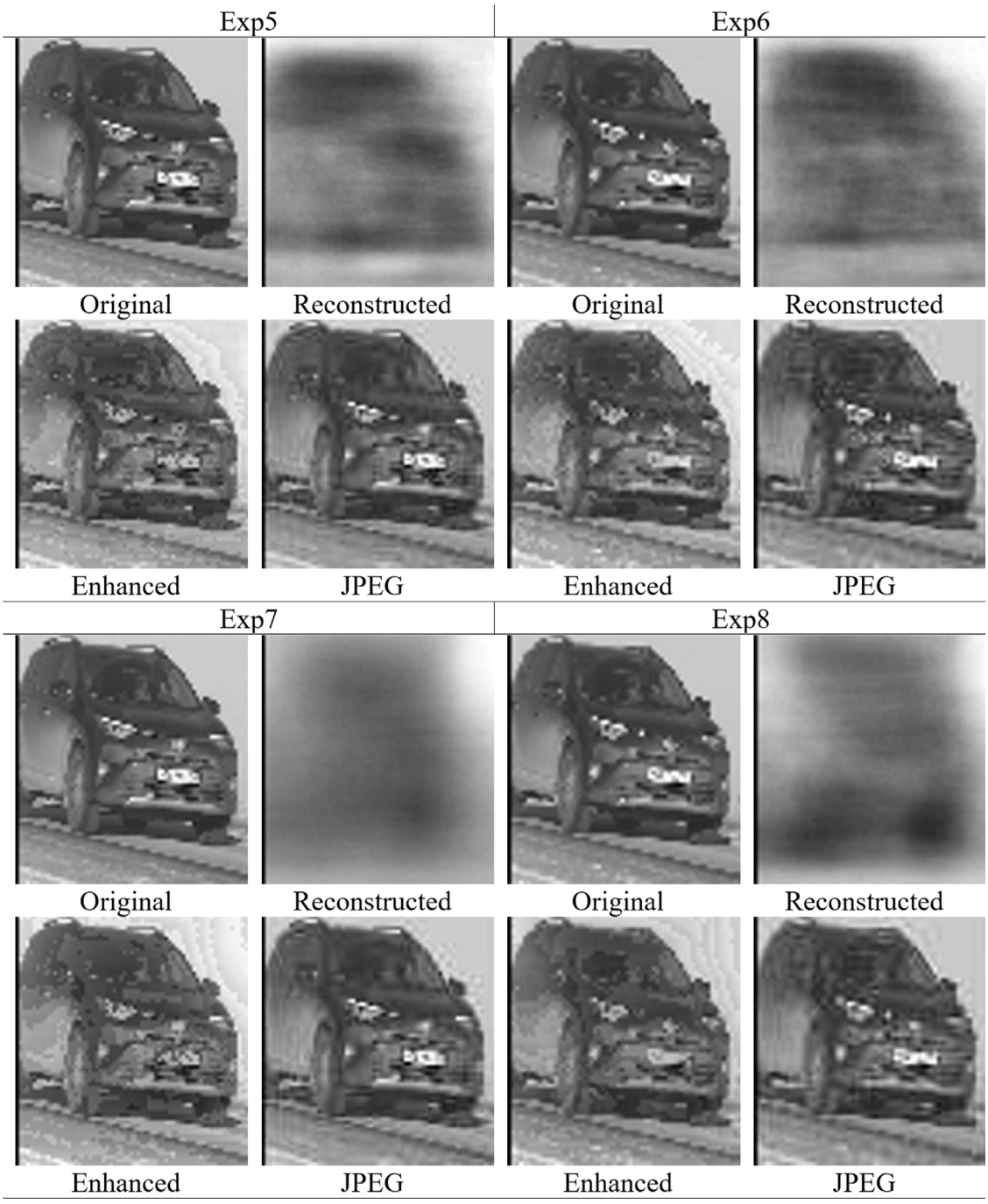


Fig. 7. Original, Reconstructed, Enhanced and JPEG encoded image of grayscale images dataset.

Table 5
Grayscale car images SSIM and PSNR scores comparison.

	SSIM			PSNR		
	Reconstructed	Enhanced	JPEG	Reconstructed	Enhanced	JPEG
Exp 5	37.72%	89.77%	90.50%	**65.98 dB**	**75.44 dB**	74.35 dB
Exp 6	**38.52%**	**90.26%**		**65.89 dB**	75.40 dB	
Exp 7	32.19%	89.14%		64.73 dB	74.39 dB	
Exp 8	32.31%	89.76%		64.75 dB	74.51 dB	

better SSIM score and better CR with 66.39% and almost the same PSNR score than the enhanced image. Experiment 10 shows that the enhanced image has only a 9.11% less SSIM score and has almost the same CR and a 0.21 dB less PSNR score than the JPEG-encoded image. Experiment 11 showed that the enhanced image has only a 9.95% less SSIM score and a 2.67 dB less PSNR score than the JPEG-encoded images. Experiment 12 showed that enhanced images have a 10.02% less SSIM score and 2.15% better CR, and a 2.14 dB less PSNR score than the JPEG-encoded image.

It is noticeable from Table 4 that the number of layers slightly impacts the performance compression of color images type where the SSIM and PSNR of an SAE with seven layers are better than those with five layers.

It is clearly noticeable the enhancement in quality that the use of BREV add to the reconstruction of the decompressed image in term of SSIM and PSNR scores for the Color images dataset.

Table 6
Color car images average size and CR comparison.

	Size (Bytes)				CR		
	Original	Reconstructed	Enhanced	JPEG	Reconstructed	Enhanced	JPEG
Exp 9	6911.10	114.47	2722.47	2323.08	98.34%	60.61%	66.39%
Exp 10		52.03	2332.92		**99.25**%	66.25%	
Exp 11		115.82	2515.16		98.32%	63.61%	
Exp 12		64.10	2174.31		99.07%	**68.54**%	

Table 7
Color car images SSIM and PSNR scores comparison.

	SSIM			PSNR		
	Reconstructed	Enhanced	JPEG	Reconstructed	Enhanced	JPEG
Exp 9	53.79%	**89.37%**	98.37%	65.79 dB	74.16 dB	74.52 dB
Exp 10	**55.80%**	89.26%		**66.05 dB**	**74.31 dB**	
Exp 11	47.70%	88.42%		64.21 dB	71.85 dB	
Exp 12	50.03%	88.35%		64.72 dB	72.38 dB	

Next, samples from the color images with more details are shown in Fig. 8. Each experiment compares an original image from the color cars dataset with the reconstructed, enhanced, and JPEG-encoded image with a quality factor of 94 to show the differences visually.

The reconstructed Color images are slightly better visually because the same image is considered to be trained three times with different channels, so more details are learned in the model and, thus, better image quality even without the enhancement step. Visually, the enhanced images keep quite similar characteristics as the JPEG images. It should be noted that in certain experiments, such as experiments 9 and 11, JPEG produces better visual quality than the proposed method. This may be due to the variable effectiveness of autoencoder-based methods, which depends on the dataset and hardware resources required to train a sufficiently large SAE model capable of compressing and decompressing images while maintaining high visual quality.

As shown in Fig. 9, observing the error pixel distribution of the decompressed image is very interesting. The quality loss in the decompressed image is not clustered in specific areas within the image for the three used datasets. Instead, the image quality will be lost throughout the image, having a nonlinear compression model with multiple signals. In Fig. 9, for MNIST and Grayscale datasets, the error pixels are shown in red, and for the color car images dataset, the areas of the error pixels are shown in green from some experiments.

Table 8 shows a comparison between the Zip format (deflate algorithm), the best of enhanced images and the best JPEG encoded images over the wholes datasets images. Table 8 summarizes the results from each experiment in terms of SSIM and PSNR scores and also CR.

As can be observed from Table 8, some values are negative because of the increase in size due to the Zip files' header containing all of the info necessary to identify the file formats and a list of all the files contained in the archive.

As can be observed from Tables 3 to 7, the proposed model significantly exceeded JPEG encoded images in the MNIST dataset experiments in SSIM, PSNR scores, and CR. In Grayscale car images, the proposed model almost matched the JPEG-encoded images in SSIM, and in some experiments, it exceeded the PSNR score and CR. In Color car images, the proposed model almost matched the PSNR score and was very close to the SSIM score by around 10% only, while the CR was better in exp12. The SSIM scores of images could have been better using large high-quality images to train with, but it was not possible due to limited resources.

In the experiment, we compared the encoding and decoding times for a 5-layer autoencoder and JPEG compression on 10,000 images from the MNIST dataset. The results showed that the encoding time using the autoencoder was 2.6049 s, and the decoding time was 2.8013 s. On the other hand, the encoding time using JPEG was 1.7122 s, and the decoding time was 1.0762 s. As observed, the autoencoder had slightly longer processing times for both encoding and decoding compared to JPEG compression.

4.5. Comparison with other image compression algorithms

The proposed SAE algorithm's performance was compared with the best performance of several state-of-the-art deep learning-based algorithms presented in [29], including ReconNet [41], DCGAN [42], CSGAN [43], SCGAN [44], TwIST and LASSO [29]. The MNIST dataset was used for this comparison, and the evaluation of the reconstruction quality was conducted using the PSNR and SSIM measures. To the best of our knowledge, there are currently no studies that utilize the grayscale (dhcl) and color (TCC) datasets.

As can be observed from Table 9, our algorithm has a better PSNR than all other algorithms and comparable SSIM than SCGAN and ReconNet algorithms. This means, the addition of the binary filter BREC resulted in a significant improvement in the quality of the image reconstruction, which is also demonstrated in Fig. 7.

5. Conclusion and future work

In this research work, a Stacked AutoEncoder model was introduced as an image compression model using three publicly available image datasets: MNIST, grayscale, and colored. The tackled problem was a lossy image compression problem. It refers to reducing the size of an image while retaining as much visual information as possible. This type of compression is typically used for digital images because it allows for more efficient storage and transmission of images, but the trade-off is a loss of image quality. The goal of lossy image compression algorithms is to strike a balance between image size reduction and image quality preservation by retaining helpful information, reducing redundant information, and removing useless information contained in an image. To assess the performance of the developed method, several experiments have been performed. The proposed model was compared with the well-known conventional arithmetic image compression standards such as JPEG and the lossless data compression such as ZIP. Experimental results show that the proposed SAE model outperforms JPEG significantly in terms of the speed, compression rate, and image decompression quality in MNIST and grayscale images and is very close to JPEG in color images. The best SSIM for MNIST data was 94.63%, which is 24.66% better than JPEG-encoded images, while the best CR was 85.37%. Grayscale decompressed images' SSIM (90.26%) almost matched the JPEG encoded images value (90.5%), with less than 0.24% difference. Finally, for the colored images, the proposed model was very close to the JPEG-encoded images with an SSIM score of 89.37% compared to 98.37% by JPEG-encoded. Results show also that the proposed SAE model highly outperform Zip lossless compression algorithm in term of compression rate but to the detriment o the quality factor of the image. Moreover, it has been observed that the add of the Residual Error Vector to the image compression vector enhance considerably the image reconstruction by minimizing the loss of image quality.

On the other hand, in dealing with new images of different classes not seen by any trained SAEs, an image classifier was introduced to detect the class of the target image with a score of 99.2% confidence on average so that it chooses the correct SAE for the target image.

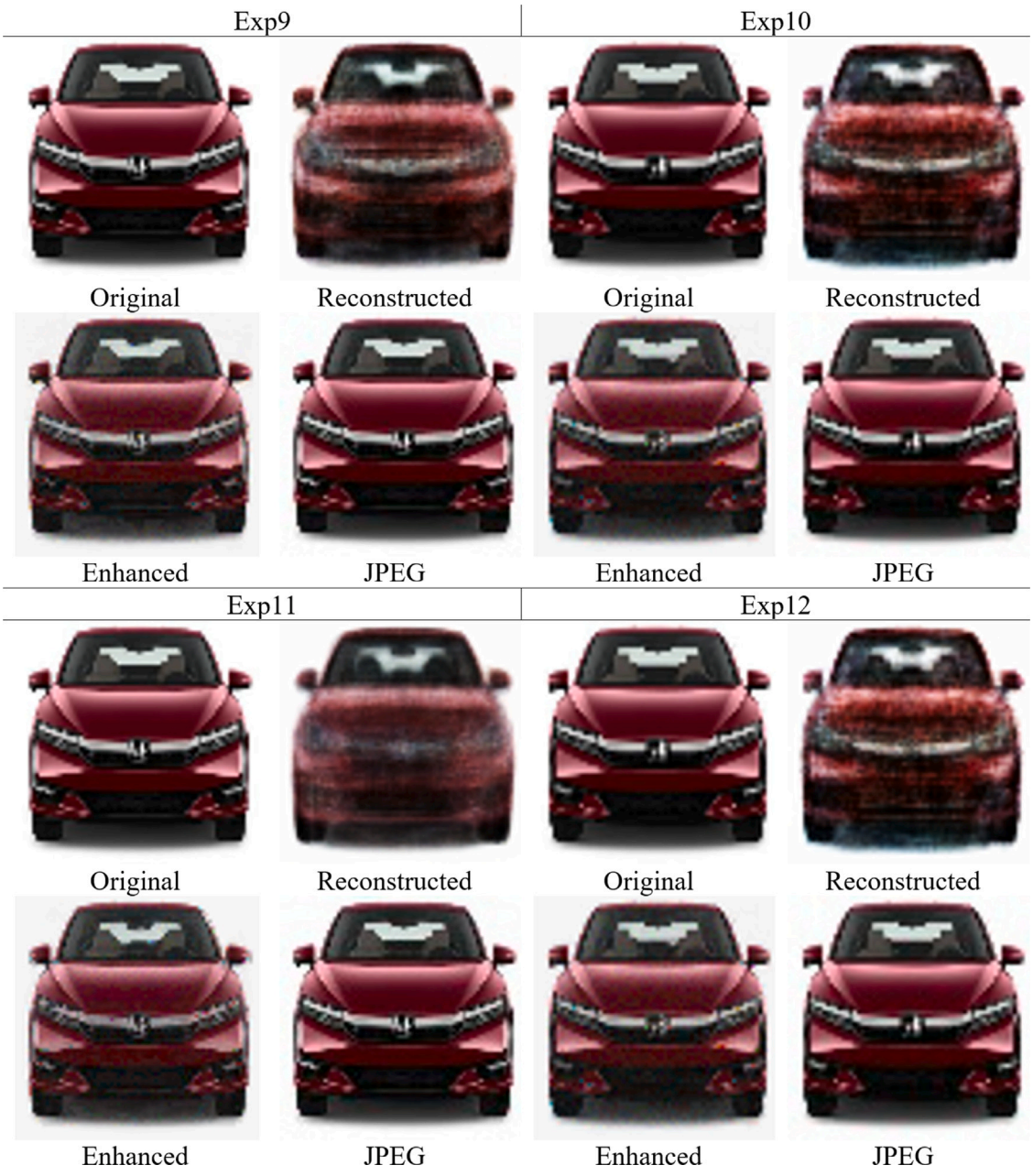


Fig. 8. Original, Reconstructed, Enhanced and JPEG encoded image of color car images dataset.

Table 8
Zip, proposed (enhanced) image, and JPEG encoded CR comparison.

Dataset	Original size (Bytes)	Zip size (Bytes)	Zip CR	CR enhanced	CR JPEG
MNIST	5,866,822	7,424,526	−26.55%	85.37%	65.91%
Grayscale car images	71,347,847	71,065,028	0.40%	86.31%	85.51%
Color car images	88,748,154	90,026,670	−1.44%	68.54%	66.39%

Table 9
Comparison with other image compression algorithms.

Number of measurements [29]	LASSO		TwIST		DCGAN		CSGAN		ReconNet		SCGAN		Proposed SAE+BREC (32 measurements + 784 bits)	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
10	10.10	54.14	5.52	0.12	8.72	20.55	11.16	41.79	14.79	52.33	14.86	54.21	**71.63**	94.6
100	10.56	47.83	8.73	24.38	9.76	31.88	18.19	81.65	23.73	92.63	25.31	**95.31**		

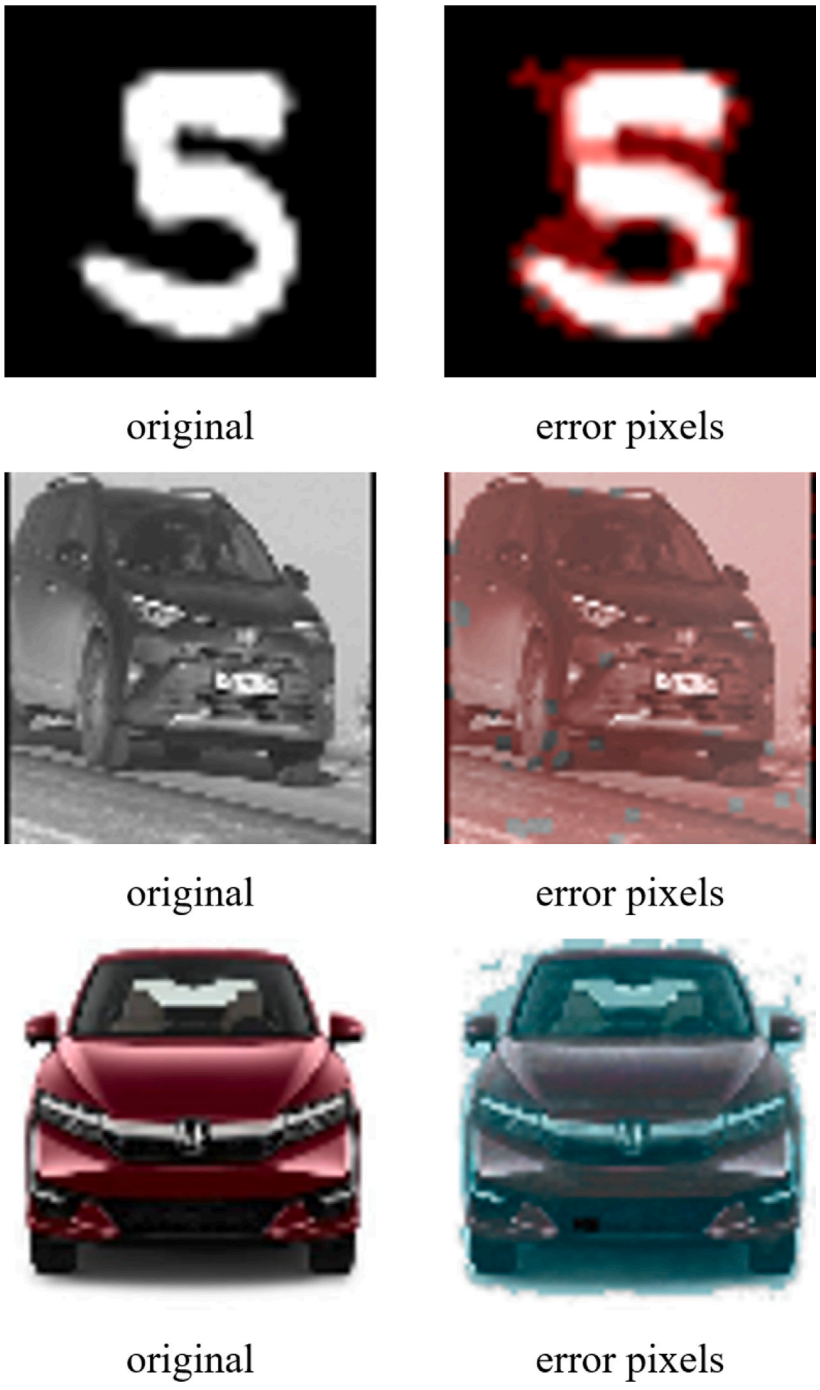


Fig. 9. Error pixels distribution of the decompressed image. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

In conclusion, the proposed method offers several advantages. It demonstrates an improved compression rate, outperforming traditional algorithms like JPEG and ZIP, as shown in Table 8. In addition, it maintains high decompressed image quality with high PSNR and SSIM scores in the experimental results. Furthermore, the use of the Binary Residual Error Vector (BREV) technique enhances the decompressed images, reducing artifacts and improving visual quality.

However, there are some limitations to consider. The computational cost of training deep neural networks can be substantial, especially for large datasets. Additionally, fine-tuning the division of layers in the encoding and decoding process may be required to achieve optimal results with different datasets. Moreover, scalability becomes a challenge when dealing with large images or datasets, as the memory required to store the compressed representation of such images can become prohibitively high.

To address these limitations, future investigations will explore dynamic architectures that adapt to specific image types, potentially improving the scalability and flexibility of autoencoders for image compression. Overall, the proposed method shows promise in achieving efficient image compression with preserved image quality and opens opportunities for further optimization and advancements, especially through the use of BREV in the decompression process.

CRediT authorship contribution statement

Salam Fraihat: Conceptualization, Methodology, Software, Data curation, Writing – original draft, Visualization, Investigation, Supervision, Validation, Writing – review & editing. **Mohammed Azmi Al-Betar:** Methodology, Data curation, Investigation, Validation, Writing – Reviewing and Editing.

Declaration of competing interest

We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

Data availability

No data was used for the research described in the article

Acknowledgments

This work was supported by grant from the Deanship of Graduate Studies and Research (DGSR) at Ajman University, Ajman, United Arab Emirates (Grant No. 2022-IRG-ENIT-5).

Ethical approval

We further confirm that any aspect of the work covered in this manuscript that has involved human patients has been conducted with the ethical approval of all relevant bodies and that such approvals are acknowledged within the manuscript.

References

[1] Yao K, Sayagh M, Shang W, Hassan AE. Improving state-of-the-art compression techniques for log management tools. IEEE Trans Softw Eng 2021;48(8):2748–60.
[2] Padmashree S, Nagapadma R. Images using embedded block coding optimization truncation (EBCOT). In: 2015 IEEE international advance computing conference (IACC). IEEE; 2015, p. 1087–92.
[3] Jordan MI, Mitchell TM. Machine learning: Trends, perspectives, and prospects. Science 2015;349(6245):255–60.
[4] Zhou Z-H. Machine learning. Springer Nature; 2021.
[5] Kelleher JD. Deep learning. MIT Press; 2019.
[6] Bank D, Koenigstein N, Giryes R. Autoencoders. 2020, arXiv preprint arXiv:2003.05991.
[7] Ahmed ST, Basha SM, Arumugam SR, Kodabagi MM. Pattern recognition: An introduction. MileStone Research Publications; 2021.
[8] Vincent P, Larochelle H, Bengio Y, Manzagol P-A. Extracting and composing robust features with denoising autoencoders. In: Proceedings of the 25th international conference on machine learning. 2008, p. 1096–103.
[9] Jayasankar U, Thirumal V, Ponnurangam D. A survey on data compression techniques: From the perspective of data quality, coding schemes, data type and applications. J King Saud Univ-Comput Inf Sci 2021;33(2):119–40.
[10] Rao KR, Dominguez HO. JPEG series. CRC Press; 2022.
[11] Sayood K. Introduction to data compression. Morgan Kaufmann; 2017.
[12] Ollivier Y. Auto-encoders: reconstruction versus compression. 2014, arXiv preprint arXiv:1403.7752.
[13] Sento A. Image compression with auto-encoder algorithm using deep neural network (DNN). In: 2016 management and innovation technology international conference (MITicon). IEEE; 2016, p. MIT–99.
[14] Theis L, Shi W, Cunningham A, Huszár F. Lossy image compression with compressive autoencoders. 2017, arXiv preprint arXiv:1703.00395.
[15] Agustsson E, Mentzer F, Tschannen M, Cavigelli L, Timofte R, Benini L, Gool LV. Soft-to-hard vector quantization for end-to-end learning compressible representations. Adv Neural Inf Process Syst 2017;30.
[16] Dumas T, Roumy A, Guillemot C. Image compression with stochastic winner-take-all auto-encoder. In: 2017 IEEE international conference on acoustics, speech and signal processing (ICASSP). IEEE; 2017, p. 1512–6.
[17] Dumas T, Roumy A, Guillemot C. Autoencoder based image compression: can the learning be quantization independent? In: 2018 IEEE international conference on acoustics, speech and signal processing (ICASSP). IEEE; 2018, p. 1188–92.
[18] Cheng Z, Sun H, Takeuchi M, Katto J. Deep convolutional autoencoder-based lossy image compression. In: 2018 picture coding symposium (PCS). IEEE; 2018, p. 253–7.
[19] Alexandre D, Chang C-P, Peng W-H, Hang H-M. An autoencoder-based learned image compressor: Description of challenge proposal by NCTU. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops. 2018, p. 2539–42.
[20] Hu F, Pu C, Gao H, Tang M, Li L. An image compression and encryption scheme based on deep learning. 2016, arXiv preprint arXiv:1608.05001.
[21] Zhou L, Cai C, Gao Y, Su S, Wu J. Variational autoencoder for low bit-rate image compression. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops. 2018, p. 2617–20.
[22] Chen T, Liu H, Ma Z, Shen Q, Cao X, Wang Y. End-to-end learnt image compression via non-local attention optimization and improved context modeling. IEEE Trans Image Process 2021;30:3179–91.
[23] Duan Z, Lu M, Ma Z, Zhu F. Lossy image compression with quantized hierarchical VAEs. In: Proceedings of the IEEE/CVF winter conference on applications of computer vision. 2023, p. 198–207.
[24] Gregor K, Besse F, Jimenez Rezende D, Danihelka I, Wierstra D. Towards conceptual compression. Adv Neural Inf Process Syst 2016;29.
[25] Yildirim O, San Tan R, Acharya UR. An efficient compression of ECG signals using deep convolutional autoencoders. Cogn Syst Res 2018;52:198–211.
[26] Tawfik A, Hosny S, Hisham S, Farouk AA, Mustafa D, Moaty SA, Gamal A, Salah K. A generic real time autoencoder-based lossy image compression. In: 2022 5th international conference on communications, signal processing, and their applications (ICCSPA). 2022, p. 1–6.
[27] Wang N, Liu T, Wang J, Liu Q, Alibhai S, He X. Locality-based transfer learning on compression autoencoder for efficient scientific data lossy compression. J Netw Comput Appl 2022;205:103452.
[28] Sebai D, Shah AU. Semantic-oriented learning-based image compression by Only-Train-Once quantized autoencoders. Signal Image Video Process 2023;17(1):285–93.
[29] Sun Y, Chen J, Liu Q, Liu G. Learning image compressed sensing with sub-pixel convolutional generative adversarial network. Pattern Recognit 2020;98:107051.
[30] Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. Commun ACM 2017;60(6):84–90.
[31] Chowdary KSH, Kalaiyarasi M, Saravanan S. Gated recurrent unit RNN based non-negative tucker decomposition for satellite image compression. In: 2022 trends in electrical, electronics, computer engineering conference (TEECCON). IEEE; 2022, p. 93–6.
[32] Wang K, Zhang N, Hu K, Cao T. Multispectral image compression algorithm based on sliced convolutional LSTM. In: International conference in communications, signal processing, and systems. Springer; 2021, p. 887–91.
[33] Li B, Liang J, Han J. Variable-rate deep image compression with vision transformers. IEEE Access 2022;10:50323–34.
[34] Qian Y, Lin M, Sun X, Tan Z, Jin R. Entroformer: A transformer-based entropy model for learned image compression. 2022, arXiv preprint arXiv:2202.05492.
[35] Li B, Liang J, Fu H, Han J. ROI-based deep image compression with swin transformers. In: ICASSP 2023-2023 IEEE international conference on acoustics, speech and signal processing (ICASSP). IEEE; 2023, p. 1–5.
[36] Imran A-A-Z, Terzopoulos D. Multi-adversarial variational autoencoder nets for simultaneous image generation and classification. In: Deep learning applications, Vol. 2. Springer; 2021, p. 249–71.
[37] LeCun Y, Cortes C, Burges C, et al. MNIST handwritten digit database. 2010.
[38] Kaggle. PAK, D. grayscale images - vehicle number plate dataset. 2020.
[39] Dhondea. Classifying car images in the TCC dataset. 2020.
[40] Setiadi DRIM. PSNR vs SSIM: imperceptibility quality assessment for image steganography. Multimedia Tools Appl 2021;80(6):8423–44.
[41] Ayna CÖ, Gürbüz AC. Robustness analysis for deep learning-based image reconstruction models. In: 2022 56th Asilomar conference on signals, systems, and computers. IEEE; 2022, p. 1428–32.
[42] Wu H-Y, Li Y-Y, Wei G-Z. A DCGAN image generation algorithm based on AE feature extraction. In: 2022 China automation congress (CAC). 2022, p. 4959–64.
[43] Babu KK, Dubey SR. CSGAN: Cyclic-synthesized generative adversarial networks for image-to-image transformation. Expert Syst Appl 2021;169:114431.
[44] Van Gansbeke W, Vandenhende S, Georgoulis S, Proesmans M, Van Gool L. SCAN: Learning to classify images without labels. In: Vedaldi A, Bischof H, Brox T, Frahm J-M, editors. Computer vision – ECCV 2020. Cham: Springer International Publishing; 2020, p. 268–85.