Image Compression Research Based On Convolutional Autoencoder

Irina Marchenko[1,†], Elena Balalayeva[1,*,†], Olena Piatykop[1,†] and Volodymyr Kukhar[2,†]

[1] *Pryazovskyi State Technical University, 29 Gogolya St, Dnipro, 49000, Ukraine*
[2] *Technical University Metinvest Polytechnic, LLC, 80 Pivdenne Hwy, Zaporizhzhia, 69106, Ukraine*

Abstract
The article presents the results of the study on image compression algorithms based on neural networks. The study analyses classical compression methods, such as JPEG, PNG, GIF, TIFF and identifies the advantages of neural network methods, in particular the use of an autoencoder, a variational autoencoder, and generative adversarial networks. A comparative analysis of classical compression algorithms, such as JPEG, with new approaches based on neural networks is carried out using the example of an autoencoder. A mathematical model describing the principle of operation for an autoencoder is presented, illustrating how a neural network encodes and restores images using latent space. To achieve the best reconstruction quality, a hybrid loss function comprising three components was employed: perceptual loss based on VGG16, SSIM loss, and MSE loss. A modular software system was developed using the Python programming language to conduct the experiments. The software includes a graphical interface, a compression module for encoding and decoding images using an autoencoder model, and a quality assessment module for calculating the main quality. The study found that traditional image compression methods demonstrate high efficiency, but are more prone to generating artifacts, especially at high compression levels, compared to neural network methods. The research results indicate that the autoencoder model can encode and decode images with minimal loss of quality, on par with JPEG, but is inferior to classical algorithms in speed and compression ratio.

Keywords
autoencoder, image compression, neural networks, JPEG, compression algorithms

1. Introduction

With the rapid advancement of information technology and the growing volume of digital content, there is an increasing need for efficient data compression methods. Images make up for a significant portion of digital data, and effective compression is essential for their storage and transmission. Most traditional compression methods, however, result in information loss and visual artefacts, which degrades image quality. Using neural networks for image compression offers strong potential for maintaining higher quality images compared to classical methods at the same compression ratios. This approach requires further research and development to enable its widespread adoption in industrial and commercial systems.

Over the past decade, numerous studies have focused on optimising image compression using neural networks. In Neural Image Compression and Explanation [1], X. Li and S. Ji demonstrated that neural networks can not only reduce image size but also retain key semantic details. Their approach allows for image compression of up to 60% of the original size without significant loss of important information. Similarly, research by [2] highlighted the advantages of end-to-end optimized compression, which uses nonlinear transformations to improve image restoration quality

ICST-2025: Information Control Systems & Technologies, September 24-26, 2025, Odesa, Ukraine.
* Corresponding author.
† These authors contributed equally.
irsa665@gmail.com (I. Marchenko); balalaevaeu@gmail.com (E. Balalayeva); piatykop_o_ye@pstu.edu (O. Piatykop); kvv.mariupol@gmail.com (V. Kukhar)
0000-0002-4566-3866 (I. Marchenko); 0000-0003-1461-4399 (E. Balalayeva); 0000-0002-7731-3051 (O. Piatykop); 0000-0002-4863-7233 (V. Kukhar)
© 2025 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR-WS.org/Vol-4048/paper23.pdf
CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

compared to traditional methods. While these studies demonstrate the strong potential of neural networks in compressing images, their practical implementation is often hindered by high computational requirements, making them inaccessible for many commercial applications.

The relevance of the research is determined by the need for a deeper understanding of modern image compression approaches that address the limitations of traditional methods. Neural network-based compression represents an innovative approach with the potential to enhance both compression efficiency and quality. Research in this field is essential for the continued development of technologies, especially as the volume of digital data continues to grow.

2. Literature review

The goal of this study is to assess the effectiveness of image compression algorithms based on neural networks and compare them with traditional compression methods.
Analysis of Recent Research and Publications. Classical image compression methods (JPEG, PNG, GIF, and TIFF) have their own strengths and weaknesses. For example, JPEG works well for photographs but can introduce noticeable artefacts when compressed too much, while PNG preserves high image quality but results in larger file sizes. The choice of compression method depends on specific requirements for the image, such as image quality, content size and type. Neural networks can offer a new approach to image compression [3-5], with their main advantage being their ability to learn from large datasets, identify patterns and extract the most important information from an image. They can also be tailored for specific image types, such as medical scans [6, 7] or satellite imagery [8], making them a flexible and versatile tool.

One popular approach is an autoencoder, a type of neural network used for encoding and decoding data [9]. It is commonly applied in dimensionality reduction and noise removal and consist of two main components: 1) encoder, which compresses the input data by identifying and keeping only the most essential features while discarding noise and irrelevant information; 2) decoder, which restores the original image from the compressed data as accurately. The main advantages of autoencoders include the ease of implementation and configuration, as well as adaptability to different data types. However, their compression quality may not be as high as more advanced approaches, while their latent space is often linear and has limitations in handling complex data.

A variational autoencoder builds on the traditional autoencoder by introducing a probabilistic approach to feature representation. Instead of using a fixed feature vector in the latent space, variational autoencoders model the data as a probability distribution [10], allowing for more flexible and generalized data representations. This flexibility in the latent space enables them to encode more complex features, generate new information based on the data not previously seen by the network and effectively handle complex, uneven data distributions. However, variational autoencoders are computationally more demanding than standard autoencoders and require more complex training.

Generative adversarial networks (GANs) are a class of artificial intelligence algorithms used in unsupervised learning. They consist of two competing neural networks in a zero-sum game [11]: one network generates candidate images (generator), while the other (discriminator) evaluates them. The generator network typically learns to build matches from the latent space to a specific data distribution, while the discriminator distinguishes between real data and the candidates produced by the generator. The training goal is to increase the discriminator's error rate. GANs are particularly effective at operating with complex patterns and have a potential in generating new data. However, they require substantial computational resources, long training times, and can sometimes introduce artefacts into the generated images.

Considering the strengths and limitations of the above methods, autoencoders were chosen for further experimentation due to their simplicity and operational features.

Let us consider the existing image compression software. A study in [1] examines a new system that combines convolutional neural network (CNN) explanations with semantic image compression in a single, end-to-end process. The authors develop a structure that explains the CNN's predictions, while compressing input images for efficient storage or transmission. The method offers an innovative approach combining neural network transparency with high compression efficiency, making it especially useful when the resources are limited in terms of data storage and transmission. However, one drawback of this software is that the algorithm's generated masks may be less accurate for images or classes not included in the training set, leading to inconsistencies in explanations or reduced compression efficiency. Additionally, the choice of parameters, such as block size, may affect the trade-off between image quality and compression level.

In [2], researchers propose a new approach for image compression using deep learning neural networks, optimized from start to finish with consideration of the trade-off between data transmission rate and distortion. The authors apply non-linear transformations inspired by biological neuron models, significantly improving the quality of compressed images compared to standard methods like JPEG and JPEG 2000. The method demonstrates strong improvements in image compression, especially at low bitrates, making it a promising option for future real-world applications. However, optimising all the parameters of this model requires considerable time and computational power, and the use of GDNs and other non-linear transformations makes implementation more complex compared to traditional algorithms, such as JPEG.

Another study [12] explores a different approach for lossy image compression using GANs, with the goal of preserving high visual quality of the restored images at low bitrates. The primary idea is to combine generative models with compression techniques, allowing the preservation of textures and fine details even when their size is significantly reduced. To improve the quality of the restored images, a discriminator is used to "train" the generator to produce realistic images. Introducing perceptual losses helps to achieve a high level of similarity between the restored and original images. This method combines advanced neural network and data compression methods to ensure high-quality restored images. However, it has some drawbacks: images with tiny details or text may lose quality, especially at extremely low bitrates; and implementing GANs for compression requires significant computational resources during the training, which limits their widespread use on devices with restricted processing power.

Thus, the main advantages of neural networks include the preservation of high texture and detail quality at low bitrates, as well as the ability to work with high-quality images. These methods, however, require substantial computational resources and may struggle with preserving fine details and tiny text. In summary, neural networks enable to achieve effective image compression, improving the balance between file size and visual quality, however, further research is required before they can be widely adopted.

The scientific novelty of this study is the establishment of the dependence of compression efficiency on the architecture of the convolutional autoencoder. This allows us to provide specific recommendations for further optimization of the model architecture and increasing its efficiency.

3. Research methodology

The autoencoder model is a complex system of functions that encodes and decodes the input image using neural networks [9, 13]. The model aims to find a latent representation of the data that would minimize information loss during the image restoration process [13, 14].

An autoencoder can be represented as a pair of functions:

– encoder $E(X) = Z$, which matches the input image $X \in \mathbb{R}^{H\times W\times C}$ with the latent space $Z \in \mathbb{R}^{d}$, where $d \ll H \times W \times C$, with H being the height, W representing the width, and C showing the number of channels;

- decoder $D(Z) = \hat{X}$, which restores the image $\hat{X}$, from the latent representation Z.

The objective is to determine the sets of parameters (weight and offsets) for the encoder θ_E and decoder θ_D that minimize the difference between the input image X and the restored image $\hat{X}$.

The encoder performs a series of convolutional operations:

$$Z = f(X) = ReLU(Conv(X, W_1) + b_1), \tag{1}$$

where $Conv$ denotes convolution, W_1 is the filter weigh matrix, b_1 is the offset, and $ReLU$ is the activation function.

After a sequence of convolutions, a latent vector is obtained:

$$Z \in \mathbb{R}^d, \tag{2}$$

where $d \ll H \times W \times C$.

The decoder restores the image from the latent space using transposed convolutional layers:

$$\hat{X} = g(Z) = Sigmoid(ConvTranspose(Z, W_2) + b_2), \tag{3}$$

where $ConvTranspose$ represents the transposed convolution, and $Sigmoid$ is the activation function that confines pixel values to the range [0, 1].

Model training algorithm:

1. The input image X is passed through the encoder and decoder, resulting in $\hat{X}$;
2. The difference between X and $\hat{X}$ is used to assess the quality of the restoration;
3. The gradients are computed and used to update the network weights;
4. A stochastic gradient descent algorithm or one of its variations is employed to minimize the difference between X and $\hat{X}$.

Characteristics of the latent space:

- the latent space contains fewer parameters than the input data, enabling the compression of information;
- the latent representation contains only the essential features for restoration, discarding irrelevant details;
- the latent space assumes a linear structure, which may limit the model's ability to handle highly complex data.

The model described outlines the functioning of the autoencoder (1) and illustrates how the neural network encodes and restores images through the latent space. During this process, some information is lost, leading to a reduction in the size of the output image. However, if the model is properly trained, this loss of information does not significantly affect the visual perception of the image, which allows the process to be regarded as a form of lossy compression. This approach serves as the foundation for the subsequent research presented in this study.

To train the model to correctly identify image features and achieve optimal reconstruction quality, a hybrid loss function is employed. This function combines three components: perceptual loss based on VGG16, SSIM loss, and MSE loss.

Perceptual loss leverages intermediate features from the VGG16 neural network, pre-trained on the ImageNet dataset. It measures the similarity between the true images y_{true} and the predicted images y_{pred} at the feature map level, rather than at the pixel level.

The formulation is as follows:

$$L_{perc}(y_{true}, y_{pred}) = \frac{1}{n}\sum_{i=1}^{n}\left(\phi(y_{true})_i - \phi(y_{pred})_i\right)^2, \tag{4}$$

where ϕ represents the function used to compute features with VGG16; n is the number of values in the feature maps; $\phi(y_{true})_i$ and $\phi(y_{pred})_i$ are the i-th feature values for the true and predicted images, respectively.

This part of the loss function allows the network to retain high-level semantic details of the images.

SSIM loss (Structural Similarity Index Loss) assesses the similarity between two images by considering structural characteristics such as brightness, contrast, and texture [15].

The formula for calculating SSIM is:

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}, \tag{5}$$

where x and y represent the image areas being compared; μ_x and μ_y are the mean pixel intensity values for blocks x and y; $\sigma_x^2$and σ_y^2 are the intensity dispersions for blocks x and y; σ_{xy} is the covariance between x and y; C_1 and C_2 are small constants preventing division by zero, typically defined as $C_1 = (K_1L)^2$ and $C_2 = (K_2L)^2$, where L is the dynamic range of pixels (255 for 8-bit images), K_1 and K_2 are larger constants typically between 0.01 and 0.03.

Since SSIM is a similarity measure (values closer to 1 indicate a higher degree of similarity), the loss function is defined as:

$$L_{SSIM}(y_{true}, y_{pred}) = 1 - SSIM(y_{true}, y_{pred}), \tag{6}$$

where $SSIM \in [0.1]$ represents the value of similarity between the images.

This function improves the structural similarity between the input and restored images.

MSE loss (Mean Squared Error Loss) measures the mean-square deviation between the pixels of the true and predicted images:

$$L_{MSE}(y_{true}, y_{pred}) = \frac{1}{m}\sum_{i=1}^{m}(y_{true,i} - y_{pred,i})^2, \tag{7}$$

where m is the number of pixels in the image; $y_{true,i}$ and $y_{pred,i}$ represent the intensity values of the i-th pixel in the corresponding images.

This part of the loss function minimizes the numerical difference between the original and restored images.

Hybrid loss function is a linear combination of the three above losses:

$$\begin{gathered} L_{hybrid}(y_{true}, y_{pred}) = a \cdot L_{SSIM}(y_{true}, y_{pred}) + b \cdot L_{perc}(y_{true}, y_{pred}) + \\ + c \cdot L_{MSE}(y_{true}, y_{pred}), \end{gathered} \tag{8}$$

where a, b, c are the importance factor coefficients, $a + b + c = 1$.

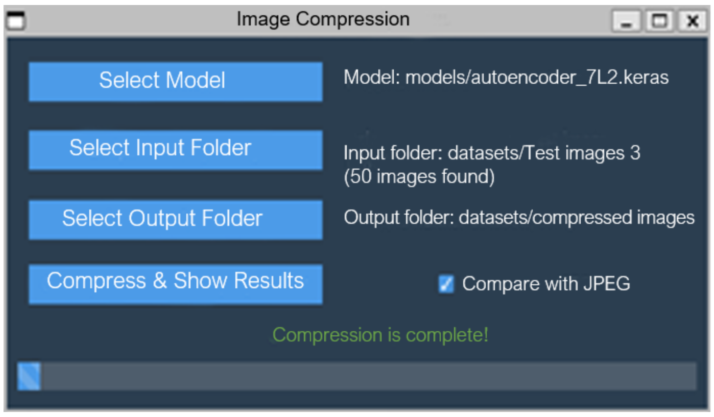
This loss function enables balancing between numerical precision, structural similarity, and high-level features.

4. Results and discussions

The experiment consisted of comparing the performance of the developed autoencoder model with classical algorithms. In particular, the compression efficiency, speed of operation, and quality of reconstructed images, assessed using PSNR and SSIM, were investigated.

To conduct the experiment, a modular software system (Figure 1) was developed in Python, comprising of the following components:

- graphical user interface, which facilitates user interaction, enabling the selection of compression models and working directories, as well as the visualization of the results and diagrams.
- compression module, which handles the encoding and decoding of images using the autoencoder model;
- quality assessment module, which computes the key evaluation metrics, specifically PSNR and SSIM.



Model: models/autoencoder_7L2.keras

Input folder: datasets/Test images 3
(50 images found)

Output folder: datasets/compressed images

☑ Compare with JPEG

Compression is complete!

Figure 1: Graphical user interface.

The development of the software relied on the following third-party libraries: tensorflow (version 2.17.0), keras, numpy, pillow, matplotlib, scikit-image, ttkbootstrap, opencv-python, and scipy.

The convolutional autoencoder model was trained using the following key hyperparameters:

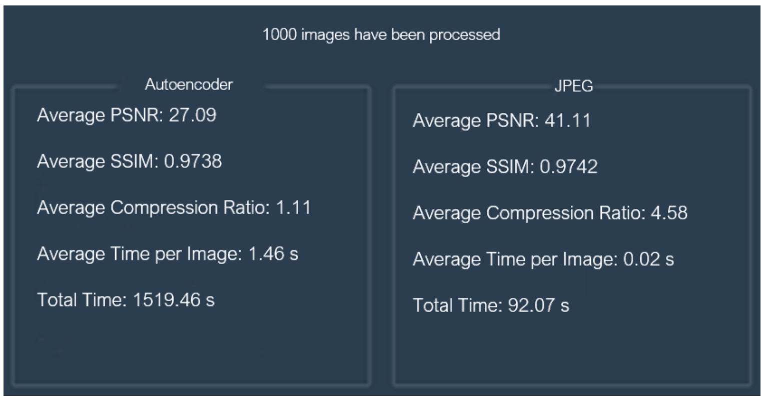
- batch_size=64 - the number of images that are simultaneously fed to the model during one training step. This parameter value allowed for efficient use of computing resources and accelerated the training process;
- epochs=40 - the number of epochs during which the model was trained. To prevent overtraining and optimize the process, the EarlyStopping technique was used with the patience=3 parameter. This allows training to be automatically terminated if the validation loss value does not improve over three consecutive epochs;
- optimizer='adam' - Adam (Adaptive Moment Estimation) optimizer to minimize the loss function with standard parameters (learning_rate=0.001, beta_1=0.9, beta_2=0.999 and epsilon=1e-07). It adaptively adjusts the learning rate for each model parameter separately, which allows for faster convergence and stability of the learning process.

The images used in this paper were taken from CelebA-HQ (CelebFaces Attributes Dataset High-Quality), an extended and improved version of the popular CelebA dataset. This dataset is used to train models such as generative adversarial networks (GANs) and autoencoders. CelebA-HQ is available for non-commercial use.

To perform the experiments, 4 encoder configurations were developed: incorporating 3 layers, 5 layers, 7 layers, and 10 layers of convolution.

Let us consider the performance of each configuration and compare them with the JPEG method.

The results obtained using the 3-layer encoder are presented in Figure 2.



1000 images have been processed

Autoencoder

Average PSNR: 27.09

Average SSIM: 0.9738

Average Compression Ratio: 1.11

Average Time per Image: 1.46 s

Total Time: 1519.46 s

JPEG

Average PSNR: 41.11

Average SSIM: 0.9742

Average Compression Ratio: 4.58

Average Time per Image: 0.02 s

Total Time: 92.07 s

Figure 2: Performance of the 3-layer model.

The model achieved an SSIM score of 0.97, which corresponds to a JPEG quality setting of 95 out of 100 (where 100 represents the highest possible image quality). An SSIM value close to 1 indicates a high degree of structural similarity between the compressed image and its original. The SSIM distribution is presented in Figure 3.

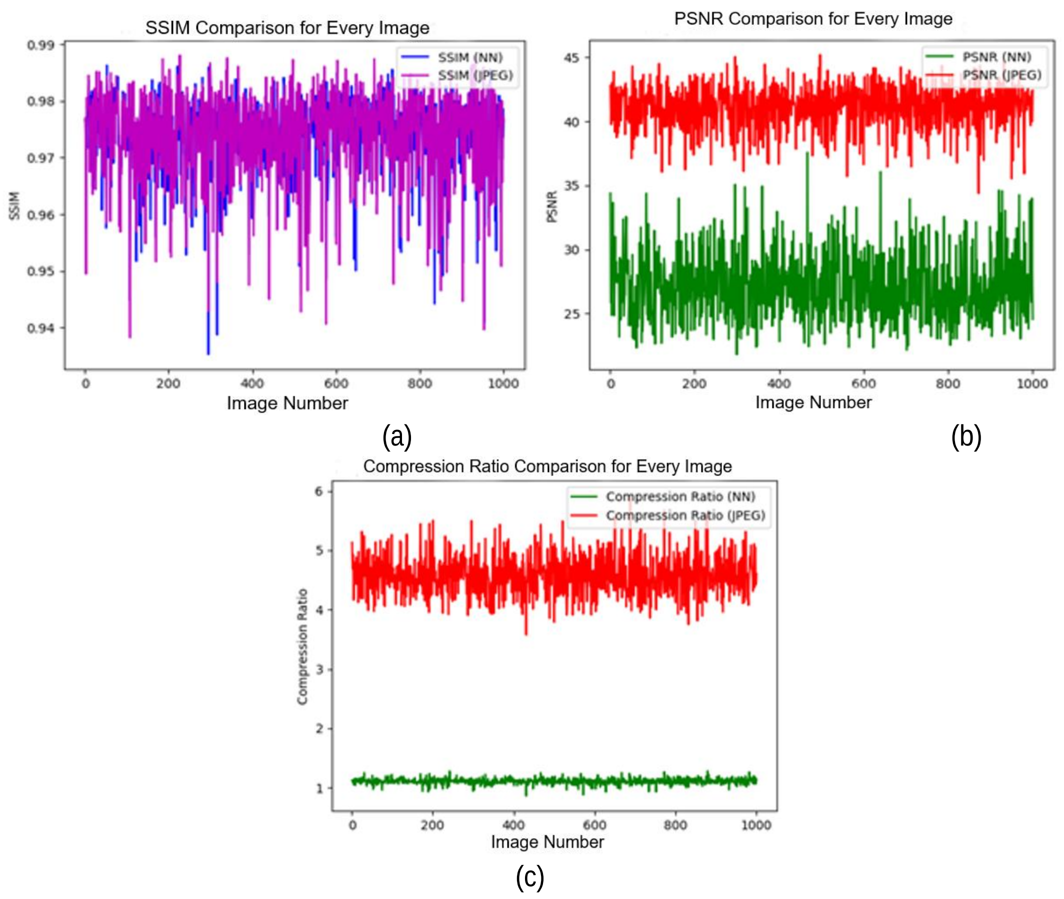


Figure 3: Performance (a), PSNR distribution (b) and compression ratio distribution (c) of the 3-layer model.

The PSNR for JPEG compression is 41.11 dB, while the autoencoder achieves 27.09 dB. The higher PSNR value for JPEG suggests that, from a mathematical standpoint, JPEG-compressed image retains greater fidelity to the original compared to the autoencoder-compressed image. However, while PSNR is an important quality parameter, it does not always correlate with perceptual visual quality. The distribution of PSNR is illustrated in Figure 3 (b).

The JPEG compression ratio (4.58) is nearly 4 times higher than that achieved by the model (1.11). Therefore, the autoencoder currently exhibits limited compression efficiency. This suggests that the model has learned to effectively encode and decode images with no significant information loss, providing a positive impact on the quality of the output images, while reducing the degree of compression. The compression ratio distribution is shown in Figure 3 (c).

JPEG compression significantly outperforms the autoencoder in terms of processing speed, requiring 92.07 seconds compared to 1519.46 seconds. This result is expected, as neural network algorithms generally demand more computational resources. A visual comparison of the output images is shown in Figure 4 and Figure 5.

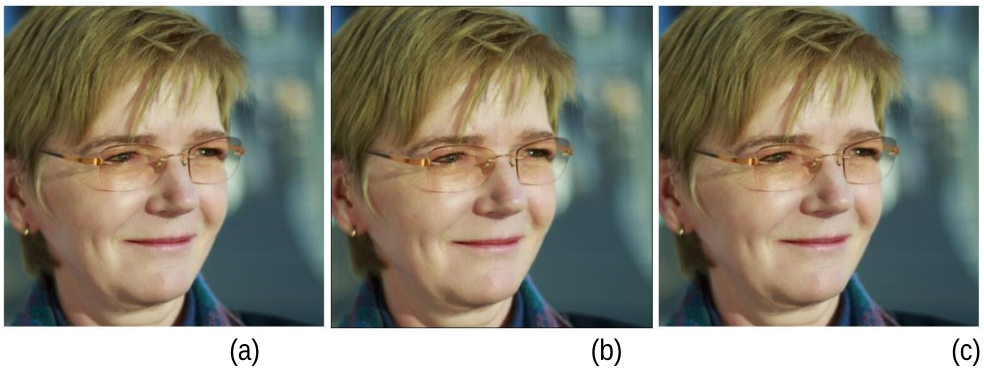


Figure 4: Comparison of image details for the 3-layer model: original image (a); autoencoder (b); JPEG (c).

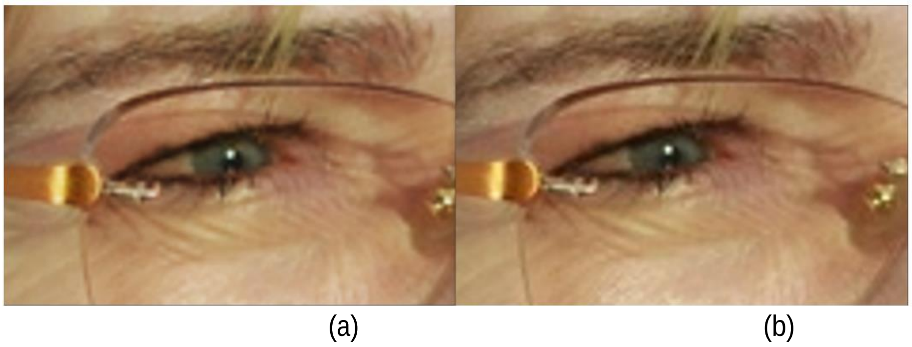
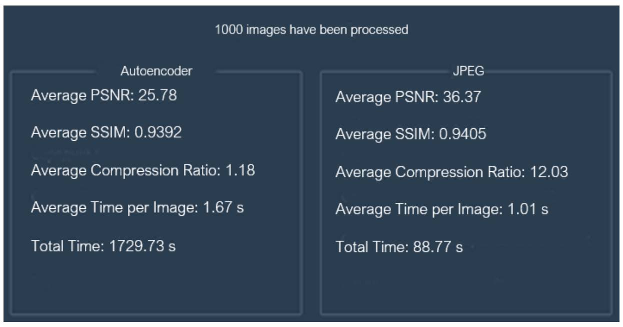


Figure 5: Comparison of image details for the 3-layer model: model (a); JPEG (b).

Visually, both images are similar to the original, with no noticeable artefacts. These results indicate that while the autoencoder achieves high-quality image restoration as indicated by SSIM, it still lags behind JPEG in terms of compression efficiency and processing speed.

Next, we evaluate the performance of the 5-layer autoencoder and compare it to JPEG (Figure 6).


```
1000 images have been processed

Autoencoder                              JPEG
Average PSNR: 25.78                      Average PSNR: 36.37
Average SSIM: 0.9392                     Average SSIM: 0.9405
Average Compression Ratio: 1.18          Average Compression Ratio: 12.03
Average Time per Image: 1.67 s           Average Time per Image: 1.01 s
Total Time: 1729.73 s                    Total Time: 88.77 s
```

Figure 6: Performance of the 5-layer model.

This model achieves an SSIM of 0.93, which approximates to a JPEG quality setting of 75/100. This represents a slight decrease in image quality compared to the 3-layer model (Figure 7).

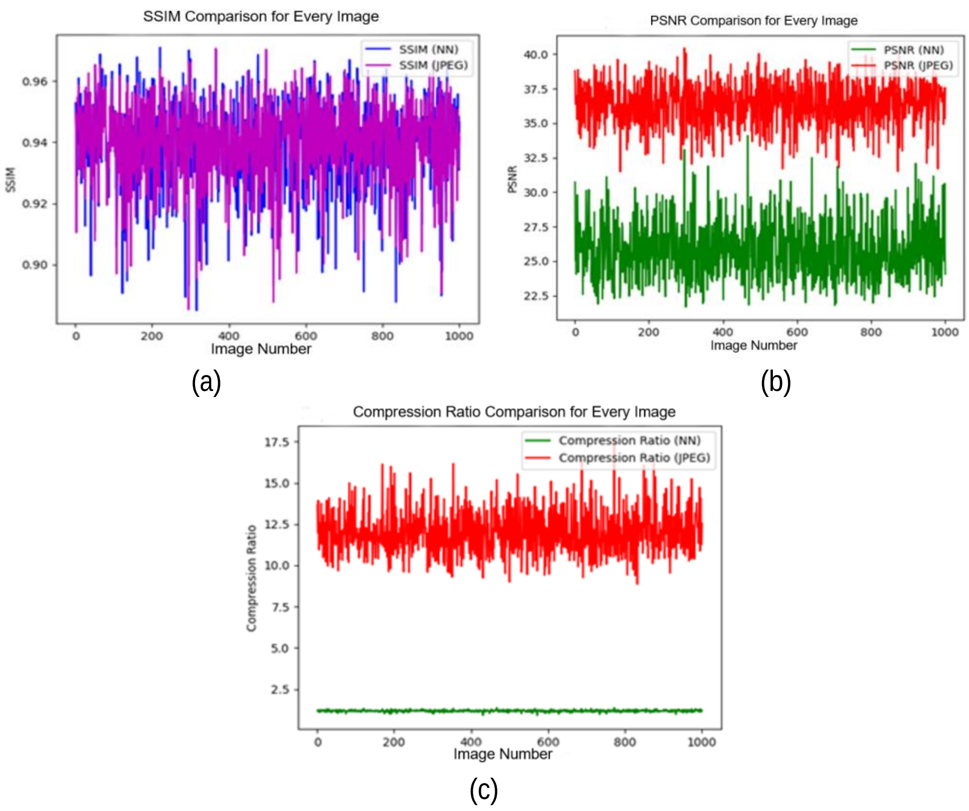


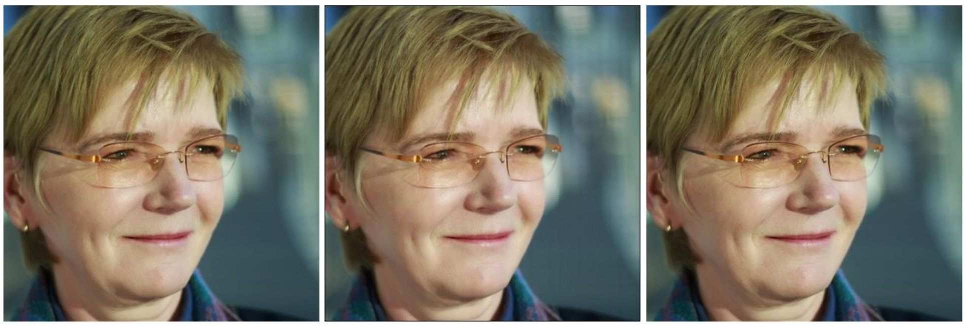
Figure 7: Performance (a), PSNR distribution (b) and compression ratio distribution (c) of the 5-layer model.

The 5-layer model also performs worse than JPEG in terms of PSNR (25.78 vs. 36.37); however, the gap between the two is narrower (Figure 7).

A notable characteristic of the JPEG method is its ability to adjust the trade-off between the quality and compression. This means that as the SSIM decreases, the compression ratio increases, as evident from the experiment results.

The 5-layer autoencoder demonstrates a slightly better compression ratio compared to the 3-layer model (1.11 vs. 1.18); however, the difference is minimal when compared to JPEG. The compression ratio distribution is shown in Figure 7.

The visual comparison is presented in (a) (b) (c)
Figure 8 and (a) (b)
Figure 9.



(a) (b) (c)

Figure 8: Comparison of image details for the 5-layer model: original image (a); autoencoder (b); JPEG (c).

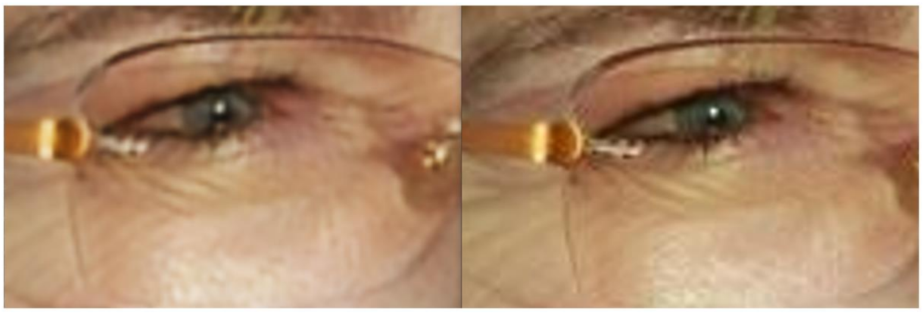
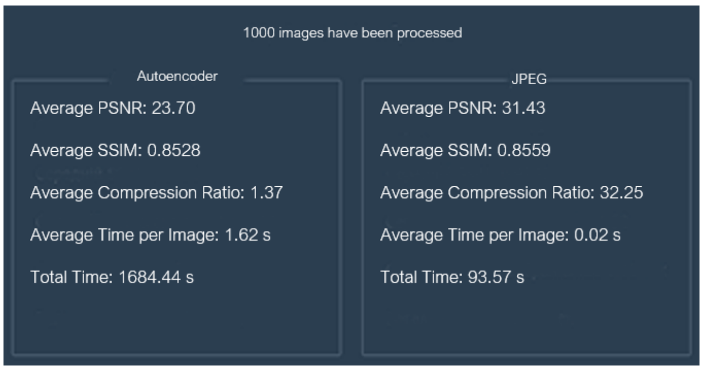


Figure 9: Comparison of image details for the 5-layer model: model (a); JPEG (b).

The images still closely resemble their original, though a slight difference in colour becomes apparent.

In comparison to the 3-layer model, there is a modest reduction in both SSIM and PSNR values. However, a small increase in the compression ratio is observed. This indicates that the compression process performed by the autoencoder is lossy, with the compression ratio dependent on the quality of the original.

We now turn to the performance of the 7-layer autoencoder model (Figure 10).



1000 images have been processed

Autoencoder	JPEG
Average PSNR: 23.70	Average PSNR: 31.43
Average SSIM: 0.8528	Average SSIM: 0.8559
Average Compression Ratio: 1.37	Average Compression Ratio: 32.25
Average Time per Image: 1.62 s	Average Time per Image: 0.02 s
Total Time: 1684.44 s	Total Time: 93.57 s

Figure 10: Performance of the 7-layer model.

An SSIM of 0.85 corresponds to a JPEG setting of 20/100.

A noticeable decline in SSIM indicates potential overfitting in the model. With an excessive number of parameters, the model may start to “memorise” the training data rather than generalising from them. Despite this, the restored images maintain clear structure, especially when compared to JPEG, where “image blocking” becomes evident. However, the restored images show a near-complete absence of green colour (Figure 11 (a, b)), and some finer details begin to blur (Figure 11 (c)).

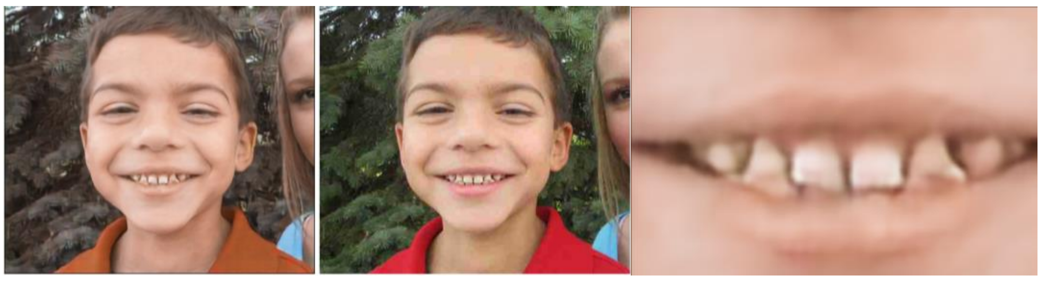
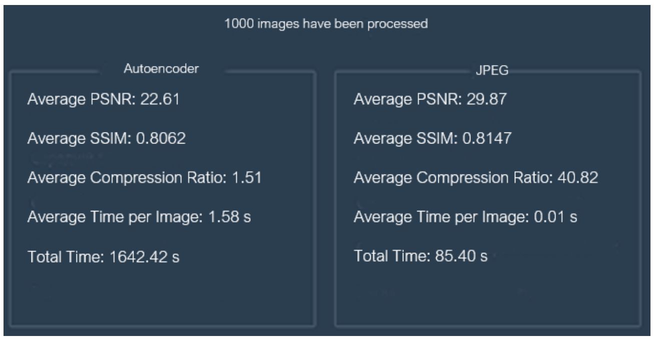


Figure 11: Performance of the 7-layer model: model (a); JPEG (b); blurring of fine details (c).

Based on these observations, we can conclude that for images of size 128x128 pixels, using a model with more than 5 convolutional layers is not advisable.

Next, we examine a case of significant overfitting with the 10-layer model (Figure 12). The SSIM of 0.80 approximates to the JPEG setting of 13/100.



1000 images have been processed

Autoencoder

Average PSNR: 22.61
Average SSIM: 0.8062
Average Compression Ratio: 1.51
Average Time per Image: 1.58 s
Total Time: 1642.42 s

JPEG

Average PSNR: 29.87
Average SSIM: 0.8147
Average Compression Ratio: 40.82
Average Time per Image: 0.01 s
Total Time: 85.40 s

Figure 12: Performance of the 10-layer model.

Due to significant overfitting, the model generates artefacts, as illustrated in Figure 13. Overall, the restored images frequently display facial features or even entire faces in areas where they should not be present.

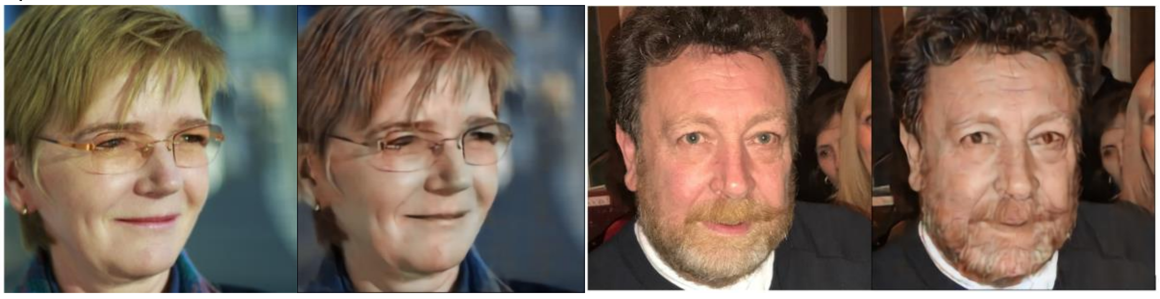


Figure 13: Comparison of image details for the 10-layer model: original images (a, c), autoencoder (b, d).

Comparative characteristics of 3 layers, 5 layers, 7 layers, 10 layers models and JPEG method with settings of 95/100, 75/100, 20/100, 13/100 is shown in Table 1.

Table 1
Comparative characteristics of the encoder with different configurations and the JPEG method

Model / JPEG	PSNR	SSIM	Compression ratio	Time per image, s	Total time, s
3-layer	27.09	0.9738	1.11	1.46	1519.46
5-layer	25.78	0.9392	1.18	1.67	1729.73
7-layer	23.70	0.8528	1.37	1.62	1684.44
10-layer	22.61	0.8062	1.51	1.58	1642. 42
JPEG (95/100)	41.11	0.9742	4.58	0.02	92.07
JPEG (75/100)	36.37	0.9405	12.03	0.01	88.77
JPEG (20/100)	31.43	0.8559	32.25	0.02	93.57
JPEG (13/100)	29.87	0.8147	40.82	0.01	85.40

Figure 14: shows graphs that demonstrate the dynamics of changes in parameters such as PSNR (a), SSIM (b), Compression ratio and Time per image with increasing number of layers (autoencoder modules with

3, 5, 7, 10 layers of convolution). The graphs also show similar results of the JPEG method with the corresponding settings (quality values 95/100, 75/100, 20/100, 13/100).

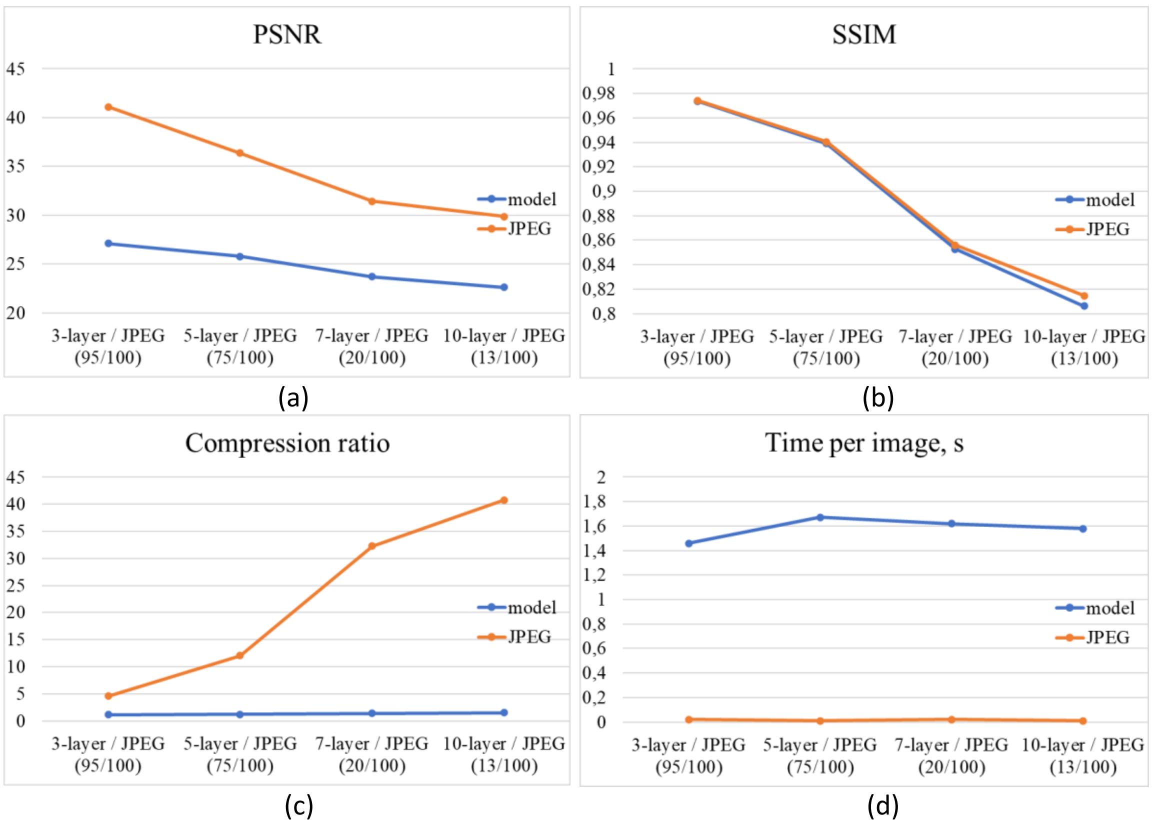


Figure 14: Graphs of the dependence of changes in image recognition results for encoder models with different numbers of layers and the JPEG method with different quality settings: PSNR (a), SSIM (b), Compression ratio (c),Time per image (d).

The findings of this research indicate that, at present, neural networks are not yet able to outperform traditional compression methods, suggesting that this area requires further exploration. The example of the basic autoencoder illustrates that while neural networks can reconstruct images with adequate quality, they still fall short of the compression efficiency and computational performance offered by classical methods. Furthermore, the study demonstrated that overly deep models tend to underperform in this context. Consequently, for compressing homogeneous images of size 128×128 pixels, shallow neural networks with 2–3 layers are more suitable.

The primary challenges at this stage include suboptimal data compression and a significant demand for computational resources. Addressing the first issue will require further model optimisation and the development of a loss function that can train the model to perform the trade-off between compression and quality. The second issue points to the need for the adoption of advanced model compression techniques, such as quantisation and neuron and weight pruning.

5. Conclusions

In summary, neural networks can reconstruct images with adequate quality, they still fall short of the compression efficiency and computational performance offered by classical methods.

This study developed an autoencoder model for image compression. The experiments demonstrated that the proposed model successfully preserves key structural features of the images, while achieving significantly lower compression ratios when compared to JPEG.

The experiments determined the maximum acceptable number of convolutional layers for the model. For images of size 128×128 pixels, the maximum acceptable number of convolutional layers was found to be 5, with 2–3 layers being the most effective configuration.

The analysis of the experimental data highlighted two main challenges for the neural network-based approach: a low compression ratio and high resource requirements. Optimising the model architecture could enhance compression efficiency. Further improvements can be made through model optimisation and compression. The results indicate that, without addressing the computational cost, neural network-based compression methods cannot replace traditional approaches.

To optimize the architecture, it is planned to explore alternative architectures, such as variational autoencoders (VAEs), to create a more efficient and stable latent space. It will be advisable to study hybrid models that combine different types of neural networks, and use dilated convolutions to increase the receptive field without additional computational costs.

To reduce the size of the model and accelerate the encoding and decoding process, a promising direction will be the study of quantization, pruning, and knowledge distillation methods.

Further research in this direction will allow creating more efficient and compact solutions for image compression based on neural networks.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] X. Li, J. Shihao, Neural Image Compression and Explanation, IEEE Access 8 (2020) 214605-214615. doi: 10.1109/ACCESS.2020.3041416.

[2] S. Singh, S. Abu-El-Haija, N. Johnston, J. Ballé, A. Shrivastava, G. Toderici, End-to-end Optimized Image Compression, ICIP (2020). https://arxiv.org/abs/2007.11797.

[3] H. T. Sadeeq, T. H. Hameed, A. Abdi, A. Nashwan, Image Compression Using Neural Networks, A Review. International Journal of Online and Biomedical Engineering 17 (2021) 135-153. doi: 10.3991/ijoe.v17i14.26059.

[4] M. Bhattacharjee, A. Ghosh, An Efficient Image Compression Algorithm Using Neural Networks, in: T. Choudhury, B. Koley, A. Nath, JS. Um, A.K. Patidar (Eds.), Geo-Environmental Hazards using AI-enabled Geospatial Techniques and Earth Observation Systems. Advances in Geographic Information Science, Springer, Cham, 2024. https://doi.org/10.1007/978-3-031-53763-9_15.

[5] D. Mishra, S. K. Singh, Rajat Kumar Singh, Deep Architectures for Image Compression: A Critical Review, Signal Processing 191 (2022) 108346. https://doi.org/10.1016/j.sigpro.2021.108346.

[6] X. Liu, L. Zhang, Z. Guo, T. Han, M. Ju, B. Xu, H. Liu, Medical Image Compression Based on Variational Autoencoder, Mathematical Problems in Engineering (2022) 7088137. https://doi.org/10.1155/2022/7088137

[7] K. A. Fettah, R. Menassel, A. Gattal, A. Gattal, Convolutional Autoencoder-Based medical image compression using a novel annotated medical X-ray imaging dataset, Biomedical Signal Processing and Control 94 (2024) 106238. https://doi.org/10.1016/j.bspc.2024.106238.

[8] G. Guerrisi, G. Schiavon, F. Del Frate, On-Board Image Compression using Convolutional Autoencoder: Performance Analysis and Application Scenarios, in: Proceedings of 2023 IEEE International Geoscience and Remote Sensing Symposium 2023 IEEE International Geoscience and Remote Sensing Symposium, 2023, pp. 1783-1786. doi: 10.1109/IGARSS52108.2023.10281562.

[9] F. Berahmand, E. S. Daneshfar, Yu.Li Salehi, Xu Yue, Autoencoders and their applications in machine learning: a survey, Artificial Intelligence Review 57(2) (2024). https://doi.org/10.1007/s10462-023-10662-6.

[10] D. P. Kingma, M. Welling, An introduction to variational autoencoders Foundations and Trends® in Machine Learning 12(4) (2019) 307-392. doi: 10.1561/2200000056.

[11] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Yo. Bengio, Generative Adversarial Nets, Advances in Neural Information Processing Systems 3(11) (2014). doi: 2672–2680 10.1145/3422622.

[12] F. Mentzer, G. D. Toderici, M. Tschannen, E. Agustsson, High-fidelity generative image compression, Advances in Neural Information Processing Systems 33 (2020). doi: 10.48550/arXiv.2006.09965.

[13] A. K. Naveen, S. Thunga, A. Murki, M. A. Kalale, S. Anil, Autoencoded Image Compression for Secure and Fast Transmission, in: Proceedings of 2024 IEEE International Conference on Computer Vision and Machine Intelligence (CVMI), 2024. pp. 1-6. doi: 10.48550/arXiv.2407.03990.

[14] D. Bank, N. Koenigstein, R. Giryes, Autoencoders, in: Rokach, L., Maimon, O., Shmueli, E. (Eds), Machine Learning for Data Science Handbook, Springer, Cham, 2023, pp. 353–374. doi: 10.1007/978-3-031-24628-9_16.

[15] V. Mudeng, M. Kim, S. Choe, Prospects of Structural Similarity Index for Medical Image Analysis, Applied Sciences 12(8) (2022) 3754. doi: 10.3390/app12083754.