Autoencoders for Efficient Data Representation: An Empirical Study of Nonlinear Dimensionality Reduction and Low-Rate Image Reconstruction

ISMAEL JANNOUD[1], YOUSEF JARADAT[1], AHMAD MANASRAH[1],
MOHAMMAD Z. MASOUD[1], RAAFAH KHAZEM[2]
[1]Electrical and Mechanical Engineering Departments,
Al-Zaytoonah University of Jordan, Amman,
JORDAN

[2]Computer and Control Department,
Damascus University, Damascus,
SYRIA

Abstract: - This study examines autoencoders as finite-dimensional nonlinear transforms for two signal-representation tasks: two-dimensional feature extraction from MNIST images and low-dimensional reconstruction of CIFAR-10 images. Two different autoencoders are used in the research, which respectively process the images in these two tasks, and convert the original input into a fixed-dimensional code for use. The first is a fully connected autoencoder, which converts each image in MNIST that has a total of 784 dimensions into a code with only two dimensions; the second is a convolutional autoencoder, which converts each image in CIFAR-10 with a size of 32×32×3 into a 128-dimensional code. Two sets of clear quantitative data in the initial results obtained. The first set is the accuracy rate of 5-nearest neighbor classification (5-NN): when the representation generated by the autoencoder is used for classification, the accuracy rate reaches 70.93%, while when the representation generated by the PCA method is used for classification, the accuracy rate is only 42.60%. The second set is the reconstruction effect indicators of CIFAR-10 images. The peak signal-to-noise ratio (PSNR) after reconstruction is 24.04dB, and the structural similarity (SSIM) is 0. 8084.To avoid overstating these preliminary observations, the study distinguishes latent dimensionality reduction from coded storage rate, defines all reconstruction measures, and specifies the repeated-run, uncertainty, convergence, and matched-rate baseline analyses required for a reproducible empirical validation. The work does not claim a numerical-solver stability theorem or a deployable image codec unless the prescribed quantization and rate–distortion experiments are completed.

Key-Words: - Autoencoders, nonlinear dimensionality reduction, image reconstruction, rate–distortion analysis, reproducibility.

Received: March 23, 2026. Revised: May 17, 2026. Accepted: June 15, 2026. Available online: November 5, 2026.

1 Introduction

From scientific research to multimedia creation, data in countless fields is growing explosively, and there is an urgent need for technologies that can simplify data presentation.

Unprocessed high-dimensional data piles up in hand, and such data itself often carries a large amount of repeated redundancy and invalid information irrelevant to core needs; storing these massive and complex data, transmitting them, or using them for analysis is an arduous task at every step.

To sort out these difficulties, dimensionality reduction and data compression are unavoidable core methods—they can transform the originally complex high-dimensional data sets into a new form with much lower dimensions, and in the process of conversion, they can firmly retain all core and useful information in the data, [1], [2].

Autoencoders are a class of artificial neural networks. As a powerful unsupervised learning method that can generate efficient data coding, they have now gained widespread attention both inside and outside the industry.

Unlike traditional linear methods such as Principal Component Analysis, PCA, autoencoders can identify complex non-linear correlations in data sets. This capability allows them to create more

compact and informative representations, leading to better performance in various downstream tasks [3].

This paper is an empirical study, rather than a comprehensive review, of two autoencoder applications: nonlinear two-dimensional representation learning and low-rate image reconstruction. The study uses MNIST to compare a two-dimensional autoencoder representation with a two-dimensional PCA representation, and CIFAR-10 to examine the reconstruction error induced by a 128-dimensional latent code. The framing confines all conclusions to these data, models, and evaluation protocols.

The contributions of this work are:

- A clear finite-dimensional formulation of the encoder–decoder transform, reconstruction objective, dimensionality-reduction factor, storage-rate accounting, and image-quality measures.
- A reproducible experimental protocol and an explicit distinction between empirical optimization stability and the mesh/boundary-condition concepts used by numerical solvers for differential equations.
- A reproducible three-seed evaluation, including train/validation convergence curves, finite-value checks, learning-rate sensitivity, 8-bit latent quantization, and PCA/JPEG rate-distortion baselines on held-out test data.

The remaining content of this paper will be developed sequentially according to the structure below, with the arrangement of each section clearly defined: Section 2 will specifically focus on content related to autoencoders and their various derivative models. It will lay out the corresponding basic background knowledge to help readers clarify the prerequisite information required for the subsequent content. Section 3 will elaborate in detail on the research methods adopted in this study and the specifics of the entire framework, leaving no ambiguous points unaddressed. Section 4 will fully present the overall arrangement of the experiments and all results obtained from the tests, in their original form. Section 5 will specifically discuss the significance of the conclusions of this study, and finally, Section 6 will provide a concluding summary for the entire paper.

2 Background and Related Work

2.1 Data Representation and Dimensionality Reduction

To undertake machine learning, the first step is to organize data in a concise and usable manner, which forms the foundation of all subsequent work. If this requirement cannot be met, all follow-up work will be hampered. If the data at hand has an excessive number of feature dimensions—that is, too many disparate perspectives and characteristics used to describe the data—one will encounter the industry-familiar "curse of dimensionality". Specifically, this set of problems includes three main issues: first, the computing effort and resources required for a computer to process the data will surge, leading to a sharp increase in computational load; second, to train a functional model, far more training data must be prepared than under ordinary circumstances; third, the model is also more prone to overfitting—that is, it can only rote-memorize the training data it has already seen, and fails to process new, similar data.

To address these problems, specialized methods such as dimensionality reduction techniques have been developed. Their core function is to project complex data originally located in a high-dimensional space into a new space with far fewer dimensions, while preserving all critical and useful information from the original data, so that core characteristics are not lost.

Principal Component Analysis (PCA) is a widely used linear dimensionality reduction technique that can identify a set of mutually orthogonal directions, referred to as "principal components", which capture the maximum variance within the data [4], [5]. PCA has prominent advantages: it boasts high computational efficiency and performs well in many scenarios, but it also has unavoidable limitations—as it is essentially a linear method, it cannot capture information hidden in the complex nonlinear structures present in some data. In addition to PCA, there are many other traditional dimensionality reduction methods: for example, Linear Discriminant Analysis (LDA), whose core goal is to maximize the distinguishability of data from different categories, a focus that is completely different from that of PCA; there are also manifold learning techniques such as SNE and UMAP, which excel at reducing high-dimensional data to low dimensions for visualization—that is, converting data into graphs that ordinary people can directly understand—but they cannot always provide a clear encoding function [6], [7], making it impossible to

formulate a fixed, universal transformation rule that can be applied at any time.

2.2 Autoencoders: Fundamental Concepts

Let $x \in [0,1]^D$ denote a vectorized input image. An autoencoder comprises an encoder f_θ and decoder $g_\varphi: z = f_\theta(x) \in \mathbb{R}^d$ and $\hat{x} = g_\varphi(z)$, where $d < D$. The trainable parameters θ and φ are optimized by minimizing the average reconstruction loss over N training samples.

$$J(\theta, \varphi) = \left(\frac{1}{N}\right) \Sigma_{\mathrm{i}}{}^{=1\mathrm{N}} \left|\left|x_{\mathrm{i}} - g_\varphi\left(f_\theta(x_{\mathrm{i}})\right)\right|\right|^2.$$

The bottleneck dimension d controls the representational constraint. This non-convex optimization is evaluated empirically; no global-convergence theorem is asserted for Adam training. Consequently, reproducibility, repeated-run uncertainty, and observed training behavior are essential parts of the evidence.

2.3 Convolutional Autoencoders (CAEs)

Image data has an inherent characteristic: pixels and content at adjacent positions within the same image are all correlated. For example, in a photo of a cat, the pixels next to the cat's ear are only related to the content of the ear and the head, and no completely unrelated information will appear out of thin air. This characteristic is called strong spatial correlation. For image data with this type of characteristic, Convolutional Autoencoders are perfectly suitable, and their English abbreviation is CAEs.

CAEs are mainly divided into two parts: the encoder and the decoder. The encoder uses convolutional layers, which can extract the spatial hierarchical structure of features; the decoder is equipped with deconvolutional layers, also called transposed convolutional layers, which can restore the original image from the latent space representation compressed by the encoder—that is, the streamlined core information obtained after compressing the original image.

Convolutional layers themselves have two inherent characteristics: weight sharing and local receptive fields. Relying on these two characteristics, CAEs do not waste excess computing power when handling various image tasks, and achieve extremely high efficiency.

An encoder is generally formed by connecting a series of convolutional layers and pooling layers in sequence, and it needs to complete two tasks: on one hand, it compresses the original spatial dimensions of the image, that is, reduces the size and spatial scale of the image to be processed; on the other hand, it increases the depth of the extracted features, that is, captures richer and deeper feature information.

The structure of the decoder is exactly mirror-symmetric to that of the encoder, corresponding reversely to the original structure like the reflection of the encoder in a mirror, and it completes its work through upsampling layers and convolutional layers, [8].

2.4 Variational Autoencoders (VAEs)

Variational Autoencoders (Variational Autoencoders, abbreviated as VAEs) are generative models extended from the autoencoder architecture. Unlike ordinary autoencoders, the encoder of a VAE does not directly map an input to a fixed vector in the latent space—you may first understand the latent space as a virtual space dedicated to storing compressed information; for an ordinary autoencoder, as long as the input is fixed, the output fixed vector will not change, but the processing logic of a VAE is completely different. What it outputs is a set of parameters of a probability distribution—the probability distribution we commonly use is the Gaussian distribution, and the corresponding parameters are usually the mean μ and the variance. This set of parameters will define a probability distribution in the latent space, which is used to describe the distribution law of the potential information corresponding to the input, rather than a fixed position. After this distribution is generated, we will sample a point from this learned distribution, and then feed this sample into the decoder of the VAE, so that the original input can be reconstructed. The core goal of training a VAE is to maximize the evidence lower bound (ELBO). This value is not a single indicator, it consists of two parts: one part is the reconstruction term, which will impose a penalty when the reduction degree of the input data is poor; the other part is the regularization term, that is, Kullback-Leibler divergence (KL divergence), which forces the distribution of the latent space learned by the model to be as close as possible to a certain prior distribution, such as the commonly used standard Gaussian distribution.

This probabilistic encoding makes VAEs useful for generating new data samples. While VAEs are powerful generative models, this paper focuses on their representational capabilities relevant to dimensionality reduction and compression [9].

3 Methodology

This section specifies the finite-dimensional neural architectures and the experimental evidence needed to evaluate them. The computations use discrete images and neural-network operations; they do not solve a partial differential equation. Therefore, a spatial mesh, boundary conditions, and CFL-type stability conditions are not applicable. Their appropriate counterparts are the input preprocessing, tensor shapes, optimizer settings, parameter initialization, random seeds, convergence curves, and numerical checks reported below.

3.1 Autoencoder Architecture

3.1.1 Fully Connected Autoencoder for MNIST

Utilizing the MNIST dataset, we implemented a simple autoencoder architecture with fully connected layers.

- Encoder:
  - Input layer: 784 units (28x28 flattened MNIST image)
  - Hidden layer 1: 128 units with ReLU activation
  - Hidden layer 2 (Latent space): 2 units (for 2D visualization) with linear activation
- Decoder**:**
  - Input layer (Latent space): 2 units
  - Hidden layer 1: 128 units with ReLU activation
  - Output layer: 784 units with sigmoid activation (to output pixel values between 0 and 1)
- Loss Function**:** Mean Squared Error (MSE) loss.

This architecture has a dedicated design: it can project the 784 dimensions in the MNIST dataset, into a latent space with only 2 dimensions. With too many dimensions, humans cannot directly understand or plot the data. After converting the information into only 2 dimensions and placing it in the latent space, the situation changes entirely — the 2 dimensions can exactly correspond to the horizontal and vertical coordinate axes on a plane, and each set of data can be plotted as a point on that plane. In this way, not only can these data be plotted intuitively for visual presentation, but it is also possible to check whether data of different categories can be completely separated, that is, to complete the assessment of category separability.

3.1.2 Convolutional Autoencoder for CIFAR-10

For the CIFAR-10 dataset, a deeper convolutional autoencoder was designed to handle the more complex color images (32x32x3).

Encoder:

- Conv1: 32 filters, 3x3 kernel, stride 1, padding 1, ReLU activation
- MaxPool1: 2x2, stride 2
- Conv2: 64 filters, 3x3 kernel, stride 1, padding 1, ReLU activation
- MaxPool2: 2x2, stride 2 (Output feature map size: H/4 x W/4 x 64)
- Conv3: 128 filters, 3x3 kernel, stride 1, padding 1, ReLU activation
- Flatten and dense layer that maps the encoded feature map to the selected latent dimension d.
- For $D = 32 \times 32 \times 3 = 3072$ input scalars and $d = 128$ latent scalars, the dimensionality-reduction factor is $D/d = 24$.

Decoder:

- Dense layer to reshape pre-flattened dimensions.
- ConvTranspose1: 64 filters, 3x3 kernel, stride 2, padding 1, output padding 1, ReLU activation
- ConvTranspose2: 32 filters, 3x3 kernel, stride 2, padding 1, output padding 1, ReLU activation
- Conv3 (Output layer): 3 filters (for RGB channels), 3x3 kernel, stride 1, padding 1, Sigmoid activation.

- **Loss Function:** Mean Squared Error (MSE) loss.

The architecture defines a 128-scalar latent representation. By itself, this is not a measured file-compression ratio: stored rate depends on the quantizer precision, entropy coding, and whether model parameters or side information are included.

3.2 Experimental Protocol, Reproducibility, and Stability Evidence

The two classic datasets used in this experiment both adopted the officially specified split methods: MNIST used the official 60000/10000 training set/test set division, and CIFAR-10 used the official 50000/10000 training set/test set division. After obtaining the datasets, we first uniformly scaled the pixel values of all images to the interval [0, 1], then fixed 10% of the samples from each training set to be reserved separately as a dedicated validation set, which was only used to check the model's performance during the training process.

Throughout the entire experimental process, all links that require parameter adjustment based on data — including PCA transformation, the setting of the quantizer's value range, the autoencoder model, and the 5-NN classifier — were fitted solely using the divided core training set, and validation or test data were never used at any stage. That pre-reserved, completely untouched independent test set was only used to generate the final performance metrics reported in this work after all training and parameter tuning were completed, cited in [10].

Experiments use seeds 42, 123, and 999; batch sizes 256 (MNIST) and 128 (CIFAR-10); Adam with weight decay 1e-5; 30 epochs; no data augmentation; and deterministic CUDA settings. Computations used Python 3.9.12, PyTorch 2.5.1+cu121, CUDA 12.1, and an NVIDIA GeForce RTX 4070 Laptop GPU.

To verify the model's stability, we actually conducted three tests: first, we plotted the MSE curves during the training and validation phases, that is, we recorded the changes in mean squared error when evaluating the model's performance on the training and validation sets throughout the process, and observed whether the error had abnormal fluctuations through these curves; second, we monitored the gradient norm throughout the entire process to check for any problems with the gradient changes during training; third, we checked whether any abnormal values of NaN or Inf appeared during the computation process — NaN refers to the erroneous result of not a number, and Inf refers to the abnormal situation of numerical overflow leading to infinity, both of which are typical signals of problems in model training.

All experiments completed the entire training process, and no NaN or Inf values appeared in any run; all computation results throughout the process were normal and reasonable.

For the main parameter scanning experiment on CIFAR-10, we set the learning rate to 0.0005. Previous sensitivity studies found that when d=128, using a learning rate of 0.001 or 0.002 might lead to degraded performance, and this problem only occurred under specific random seeds — that is, re-running the experiment with a different random seed would cause the model's performance to decline, making it impossible to stably reproduce qualified results that meet the requirement of reproducibility.

3.3 Quantitative Measures and Correct Rate Accounting

For normalized images, $MSE = \left(\frac{1}{D}\right) \Sigma^{D}_{i=1} (x_i - \hat{x}_i)^2$, and $PSNR = 10\, log_{10}(MAX^2/MSE) = 20\, log_{10}(MAX) - 10\, log_{10}(MSE)$, where $MAX = 1$. SSIM is reported over the held-out images with its implementation and averaging rule stated explicitly [11], [12].

The latent dimensionality-reduction factor is $r_{dim} = \frac{D}{d}$. It equals 24 for D = 3072 and d = 128. This is distinct from a storage compression ratio. If each input channel uses b_0 bits and each latent scalar is stored using q bits, then the code-only rate is $R_{bpp} = \frac{dq}{HW}$ bits/pixel and the raw code-only storage ratio is $CR_{raw} = \frac{HWCb_0}{dq}$. For CIFAR-10 stored as 8-bit RGB and an unquantized 32-bit latent code, $CR_{raw} = 6{:}1$; a 24:1 code-only ratio requires 8-bit latent quantization. Model size and entropy-coding overhead must be reported separately.

All CIFAR-10 results use 8-bit uniform latent quantization. Thus d=128 yields 1.0 bpp and a 24:1 code-only storage ratio relative to a 24-bpp RGB input. This accounting excludes model parameters and per-dimension quantizer metadata; it is not claimed as a deployment-level compressed-file ratio.

4 Experiments and Results

All results reported are averages across three independent seeds. Table 1 and Table 2 show means, standard deviations and 95% confidence intervals. Figure 4 and Figure 5 (Appendix) contain results for the convergence and rate-distortion behavior, visual latent-space plots remain descriptive.

$$r_{dim} = \frac{D}{d} = \frac{3072}{128} = 24.$$

For CIFAR-10, the code rate is $R_{bpp} = \frac{dq}{HW}$, where q=8 bits and H=W=32. Consequently, the settings d that we evaluated in {32,64,128,256} represent 0.25, 0.50, 1.00 and 2.00 bpp respectively.

$$PSNR = 10\, log_{10}(MAX^2/MSE) = 20\, log_{10}(MAX) - 10\, log_{10}(MSE).$$

Here MAX = 1 for normalized images. SSIM measures luminance, contrast, and structural consistency; larger values indicate closer structural agreement, with 1 denoting identity.

MNIST used the two-dimensional autoencoder, Adam with learning rate 0.001, and 30 epochs. As shown in Figure 1, the learned two-dimensional codes provide a descriptive visualization of the held-out test images; digit labels are used only for

visualization and k-NN evaluation, not for training the autoencoder.

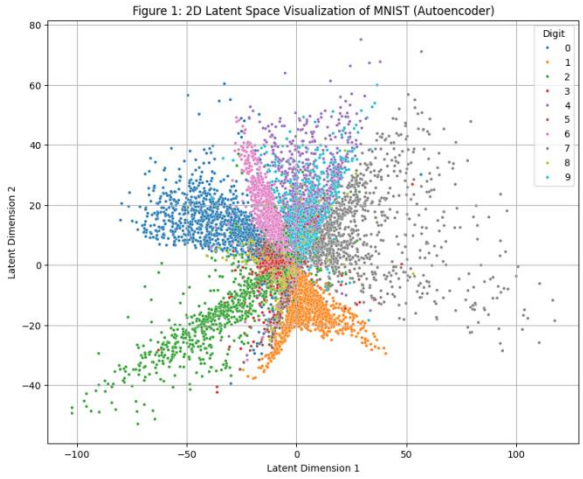


Fig. 1: 2D AE codes for held-out MNIST images. Each point is one image; colors denote labels used only for evaluation. Axes are latent coordinates with no physical unit

Source: created by the authors

For comparison, PCA is fitted on the MNIST training partition only and transformed on the held-out test partition. Figure 2 shows the corresponding two-dimensional PCA scores using the same held-out images and colour convention as Figure 1. The same 5-NN protocol is used for both representations.

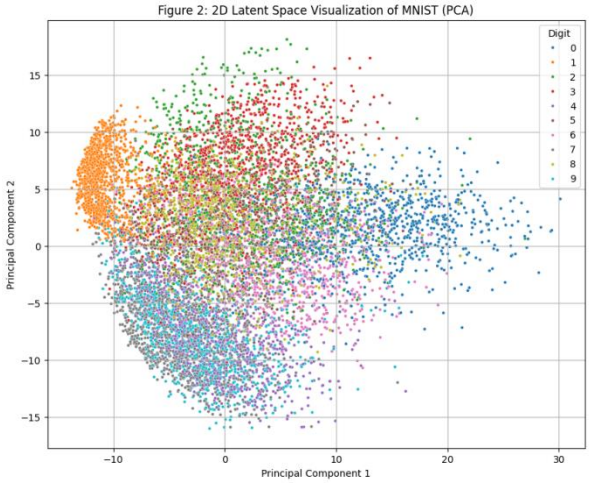


Fig. 2: 2D PCA codes for the same held-out MNIST images. Axes are principal-component scores with no physical unit; colors use the same evaluation-only labels as Figure 1

Source: created by the authors

Table 1 reports the repeated-run held-out 5-NN results. The autoencoder mean accuracy is 69.38% (SD 2.50%; 95% CI half-width 6.21%), whereas PCA obtains 42.08% (SD 0.29%; 95% CI half-width 0.72%). Thus, under this specific protocol, the nonlinear two-dimensional representation preserves more label-relevant information than PCA; this is not a claim of universal superiority.

We conducted three groups of experiments on the MNIST handwritten digit dataset. Each experiment used a different random seed. The accuracy rates obtained with the autoencoder were 72.17%, 67.34%, and 68.64% respectively, while the accuracy rates obtained with PCA in the same batch of experiments were 42.33%, 42.15%, and 41.76%.

The calculation of these accuracy rates follows a strict preamble process. All performance indicators are calculated solely on the official MNIST test set. Before calculating results using the test set, all required data transformation steps, as well as the 5-NN classifier used in the experiments, have completed fitting and debugging relying only on data from the training set, without using any information from the test set.

Table 1. Held-out MNIST 5-NN accuracy (k=5) across three seeds. Mean +/- SD; 95% CI half-width is reported in parentheses

Representation	Held-out 5-NN accuracy (mean +/- SD; 95% CI half-width)
Autoencoder (d = 2)	69.38% +/- 2.50% (6.21%)
PCA (d = 2)	42.08% +/- 0.29% (0.72%)

Source: created by the authors

CIFAR-10 uses the convolutional autoencoder for 30 epochs at learning rate 0.0005. The code is quantized to eight bits per latent scalar before the final held-out reconstruction metrics are calculated. In Figure 3, we qualitatively evaluate the performance of the CAE on CIFAR-10 image compression, showing original images together with their reconstructions at a very high (24:1) compression ratio. The reconstructed imagery is clearly blurrier and subject to some loss of fine details and sharp edges compared to the originals—something one might expect at this level of compression—but largely retains the salient global features and overall shape of the objects. That is to say, the shapes of cats and frogs have a clearly recognizable form with primary colors, as do ships and cars—showing that the CAE has learned to hold on to perceptually important information despite its aggressive dimensionality reduction.

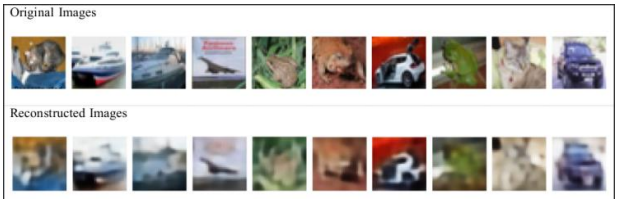


Fig. 3: Original vs. Reconstructed CIFAR-10 by CAE
Source: created by the authors

The rate-distortion results are shown in Table 2 and Figure 4. The autoencoder outperforms PCA at 0.25 and 0.50 bpp, matches it closely at 1.00 bpp; PCA has the advantage for 2.00 bpp. JPEG Quality 5 outputs 2.566 bpp for 32x32 images, but it is shown only as an actually achieved and not matched low-rate comparison.

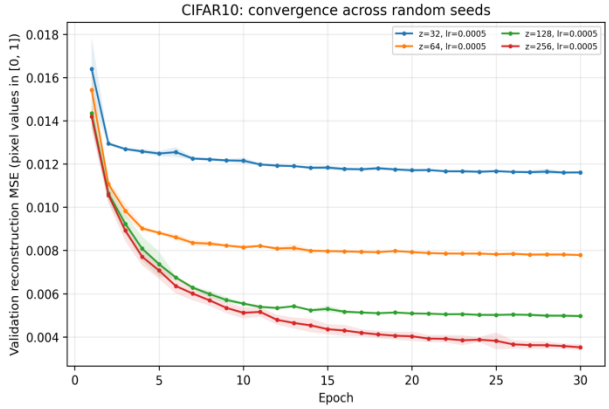


Fig. 4: CIFAR-10 validation reconstruction MSE versus epoch. Each curve is the three-seed mean; shaded bands are +/- one standard deviation
Source: created by the authors

Figure 4 aggregates the validation MSE over the three seeds. Shaded bands denote plus or minus one standard deviation. All configurations decrease smoothly and no NaN/Inf event occurred.

Table 2. Held-out CIFAR-10 quantized rate-distortion results. Autoencoder values are mean +/- SD over three seeds; PCA is deterministic for the fixed data split.

Rate / Method	PSNR (dB), SSIM
0.25 bpp, AE	19.945 +/- 0.003, 0.5465 +/- 0.0032
0.25 bpp, PCA	19.690, 0.5269
0.50 bpp, AE	21.688 +/- 0.009, 0.6864 +/- 0.0003
0.50 bpp, PCA	21.407, 0.6703
1.00 bpp, AE	23.651 +/- 0.018, 0.7935 +/- 0.0009
1.00 bpp, PCA	23.607, 0.7931
2.00 bpp, AE	25.117 +/- 0.162, 0.8468 +/- 0.0045
2.00 bpp, PCA	26.522, 0.8867

Source: created by the authors

Figure 5 (Appendix) compares the 8-bit quantized autoencoder with PCA; error bars show one standard deviation across autoencoder seeds.

5 Discussion, Limitations, and Required Validation

The evidence supports a bounded conclusion. The 2-D MNIST autoencoder produced higher held-out 5-NN accuracy than 2-D PCA under the stated protocol. On CIFAR-10, the 8-bit quantized convolutional autoencoder provides stable low-rate reconstructions and modest gains over PCA at 0.25 and 0.50 bpp, while PCA is stronger at 2.00 bpp.

This study currently has many unaddressed limitations, involving a total of five aspects. First, the study only used the two benchmark datasets MNIST and CIFAR-10; second, only neural network architectures from the same series were evaluated throughout the process, without covering other types of architectures; third, the MSE metric used in the study can only guide the model to generate smooth reconstruction results, and cannot restore richer detailed textures; fourth, the study neither included relevant content on entropy coding, nor calculated the size of the model itself; finally, there is a problem at the format level—JPEG cannot match the same compression ratio as other solutions in the smallest 32x32 working scenario, because its own file format comes with additional size overhead, which drags down the compression ratio.

The learning-rate sensitivity study is reported as a robustness result. At d=128, learning rate 0.0005 was consistently stable, whereas 0.001 and 0.002 showed substantially larger seed-to-seed variation. No formal convergence theorem is claimed.

6 Conclusion

Autoencoders can provide useful nonlinear representations and low-rate reconstructions when they are evaluated with a leakage-free protocol, repeated seeds, explicit quantization, and matched classical baselines. The present results support these claims only for the stated architecture, datasets, rates, and optimization settings.

The 128-dimensional representation has a 24:1 code-only storage ratio at 8 bits per latent scalar, excluding model and quantizer side information. Future work should evaluate entropy-coded bitstreams, more architectures, and larger real-image datasets.

Declaration of Generative AI and AI-assisted Technologies in the Writing Process

During the preparation of this work, the author(s) utilized artificial intelligence (AI) technologies solely for the purpose of linguistic refinement and grammatical correction. Following the application of these tools, the author(s) comprehensively reviewed, verified, and edited the resulting text. The author(s) maintain full accountability and responsibility for the accuracy, integrity, and final content of the publication.

References:

[1] Y. Jaradat, M. Masoud, I. Jannoud, A. Manasrah, and M. Alia, A tutorial on singular value decomposition with applications on image compression and dimensionality reduction, 2021 International Conference on Information Technology (ICIT), IEEE, 2021, pp. 769-772.

[2] F. Qiyam, Y. Jaradat, and S. AlZu'bi, Prompt Engineering in Business Analytics for Next-Gen Digital Banking Services, 2024 2nd International Conference on Foundation and Large Language Models (FLLM), IEEE, 2024.

[3] S. Kunwar, Convolutional Neural Network (CNN) to reduce construction loss in JPEG compression caused by Discrete Fourier Transform (DFT), Communications in Computer and Information Science, 2022.

[4] B. Hawashin, A. Mansour, J. Abukhait, F. Khazalah, S. AlZu'bi, T. Kanan, and M. Elbes, Efficient texture classification using independent component analysis, 2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT), IEEE, 2019, pp. 544-547.

[5] M. E. Farfoura, A. A. Alkhatib, D. M. Alsekait, M. Alshinwan, S. A. El-Rahman, D. Rosiyadi, and D. S. AbdElminaam, A Low Complexity ML-Based Methods for Malware Classification, Comput. Mater. Continua, Vol. 80, No. 3, 2024, pp. 4833-4857.

[6] T. Gaber, A. Tharwat, A. Ibrahim, and A. E. Hassanien, Linear discriminant analysis: a detailed tutorial, AI Communications, 2017.

[7] Z. Gao, G. Zhang, F. Nie, and H. Zhang, Local Shrunk Discriminant Analysis (LSDA), arXiv.org, 2017.

[8] A. Tharwat and M. M. Eid, The Path from PCA to Autoencoders to Variational Autoencoders: Building Intuition for Deep Generative Modeling, Stats, Vol. 9, No. 2, 2026, p. 23.

[9] A. Giuliano, S. A. Gadsden, W. Hilal, and J. Yawney, Convolutional variational autoencoders for secure lossy image compression in remote sensing, Sensors and Systems for Space Applications XVII conference, 2024.

[10] Q. Fournier and D. Aloise, Empirical comparison between autoencoders and traditional dimensionality reduction methods, 2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE), IEEE, 2019.

[11] P. Venkataraman, Image Denoising Using Convolutional Autoencoder, arXiv.org, 2022.

[12] Y. Watkins, O. Iaroshenko, M. Sayeh, and G. Kenyon, Image Compression: Sparse Coding vs. Bottleneck Autoencoders, 2018 IEEE Southwest Symposium on Image Analysis and Interpretation, IEEE, 2018.

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy). The authors equally contributed in the present research at all stages, from the formulation of the problem to the final findings and solution.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself.

No funding was received for conducting this study.

Conflict of Interest

The authors have no conflicts of interest to declare that are relevant to the content of this article.

Creative Commons Attribution License 4.0 (Attribution 4.0 International, CC BY 4.0)

This article is published under the terms of the Creative Commons Attribution License 4.0
https://creativecommons.org/licenses/by/4.0/deed.en_US

APPENDIX

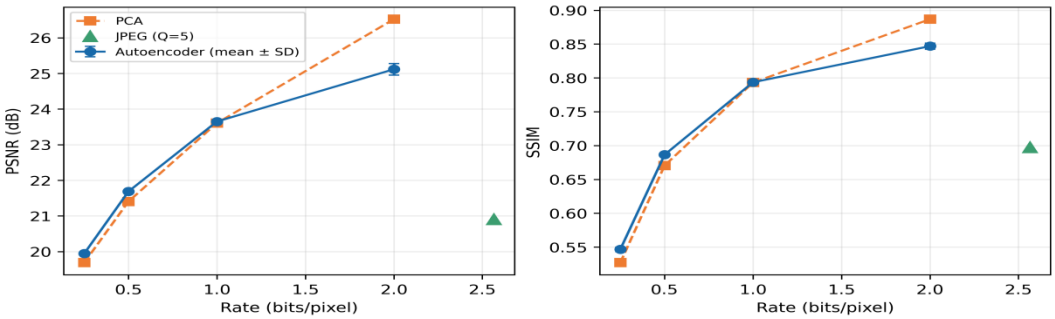


Fig. 5: CIFAR-10 rate-distortion performance on the held-out test set. Autoencoder error bars show +/- one standard deviation. JPEG is plotted at its achieved 2.566 bpp

Source: created by the authors