Spectrum of Engineering Sciences
ISSN (e) 3007-3138 (p) 3007-312X Volume 3, Issue 4, 2025

PARAMETER OPTIMIZATION OF AUTOENCODER FOR IMAGE CLASSIFICATION USING GENETIC ALGORITHM

Ghulam Gilanie[*1], Humaira Shafiq[1], Syeda Naila Batool[1], Syed Naseem Abbas[2], Hina Shafique[1], Sana Cheema[1], Akkasha Latif[1], **Anum Saher**[1], **Muhammad Ahsan**[2]

[1]*Department of Artificial Intelligence, Faculty of Computing, The Islamia University of Bahawalpur, Pakistan*
[2] *Department of Computer Science, Faculty of Computing, The Islamia University of Bahawalpur*

[*1]ghulam.gilanie@iub.edu.pk, [1]humairashafiq333@gmail.com, [1]nailashah313@gmail.com, [2]nasim.naqvi@iub.edu.pk, [1]hinach1912@gmail.com, [1]sanacheema887@gmail.com, [1]akashacheema70@gmail.com, [1]saheranum1@gmail.com, [2]chahsan146@gmail.com

DOI: https://doi.org/10.5281/zenodo.15180569

Keywords
Autoencoder Optimization, Genetic Algorithm, Hyperparameter Tuning, Image Classification, Deep Learning

Article History
Received on 01 March 2025
Accepted on 01 April 2025
Published on 09 April 2025

Corresponding Author: *

Abstract
This research focuses on the parameter optimization of an autoencoder for image classification using a genetic algorithm (GA). An autoencoder is a neural network architecture commonly used for unsupervised learning, dimensionality reduction, and feature extraction. Its performance heavily depends on hyperparameters, which must be carefully tuned to achieve optimal results. In this study, a GA-based optimization approach is proposed to fine-tune the hyperparameters of an autoencoder, including the number of hidden layers, the number of neurons per layer, the activation function, the learning rate, and the batch size. The proposed approach is applied to two datasets: MNIST and EMNIST (an extended version of MNIST for handwritten letters), as well as Fashion-MNIST. The performance is compared against other state-of-the-art optimization techniques. The results demonstrate that GA-based optimization effectively enhances autoencoder performance, outperforming traditional methods in terms of reconstruction error and classification accuracy. Specifically, when using the Adam optimizer, the average classification accuracy achieved is 97.77% with an average computation time of 34.77s, using a learning rate of 0.001, momentum of 0.87, and a sparsity parameter of 0.01. In contrast, the GA-based approach yields an improved accuracy of 98.85% with a slightly higher computation time of 35.44s, using a learning rate of 0.1, momentum of 0.85, and a sparsity parameter of 0.01. This research contributes to the development of an efficient autoencoder optimization framework, applicable to a wide range of tasks, including image classification, data compression, feature extraction, and anomaly detection.

INTRODUCTION
An autoencoder is a type of neural network [1-5]designed to encode input data into a compressed and meaningful representation and then decode it to reconstruct the input as accurately as possible to its original form [6-9]. Their main goal is to learn an informative representation of the data autonomously, without explicit guidance. This learned representation can be applied to various tasks, such as clustering. Autoencoders, a type of neural network architecture, are widely used in various fields, including but not limited to image compression and reconstruction, anomaly detection in time-series data,

recommender systems, image denoising [10-13], generative models, and natural language processing (NLP) tasks such as summarization and text generation [14-16]. In an autoencoder, input data is first encoded into a lower-dimensional representation and then decoded back to reconstruct the original input using this encoded representation. Autoencoders can learn meaningful feature representations from data even when it is unlabeled, making them well-suited for unsupervised learning. They also serve as a valuable alternative to convolutional neural networks (CNNs) and long short-term memory networks (LSTMs). Before training CNN or LSTM, autoencoders can be employed in preprocessing tasks such as dimensionality reduction and feature learning, enhancing model performance [17-20]. Even when the task is not image classification or sequence prediction but rather learning a compact representation of the data, autoencoders can be used as an alternative to CNNs [21] or LSTMs. However, CNNs and LSTMs often outperform autoencoders in many image and sequence-related tasks. [22]. Autoencoders are primarily used for various tasks, including data compression, feature extraction, anomaly detection, image and video processing, natural language processing (NLP), reinforcement learning, recommendation systems, and generative modeling [23-26].

The process of parameter optimization is crucial as it directly influences model performance and helps prevent overfitting or underfitting. Additionally, it ensures that the model remains efficient and interpretable by avoiding unnecessary complexity, which can lead to increased computation time and reduced interpretability [27]. Researchers select parameter optimization algorithms based on their specific requirements. Some commonly used optimization algorithms include Genetic Algorithm (GA), Simulated Annealing (SA), Artificial Bee Colony Algorithm (ABC), Particle Swarm Optimization (PSO), Harmony Search (HS), and Shuffled Frog Leaping (SFL), among others [28].

Genetic Algorithms (GAs) are commonly used to solve real-parameter optimization problems by encoding each parameter as a string of bits [29]. This can be achieved using either standard binary coding or gray coding. The bit strings representing individual parameters are concatenated to form a single bit string, known as a "chromosome," which represents the entire parameter vector. [30]. An autoencoder is a type of neural network architecture used for dimensionality reduction and feature learning. It consists of two main components, encoder
(maps the input data to a lower-dimensional representation, also known as the bottleneck or latent space) and decoder (reconstructs the original data from the latent representation, restoring it to its original dimensionality) [31]. The goal of autoencoders is to learn a compact representation of input data that preserves the most important features while eliminating noise or irrelevant information. This makes them particularly useful for tasks such as image compression, anomaly detection, and generative modeling [32].

Parameter optimization of an autoencoder involves finding the optimal set of values for the model's parameters to achieve the best performance on a given task. This typically includes adjusting the encoder and decoder parameters, such as the number of neurons, activation functions, and learning rate [33].

In this research, GA was used to optimize the parameters of an autoencoder. A lower gradient descent value generally indicates better-optimized parameters, but in some cases, even with a minimal gradient descent, the training complexity of the autoencoder increases. The goal of this study is to provide an optimal solution for selecting the best autoencoder parameters using GA, ensuring maximum accuracy during training while maintaining efficiency [34-38].

The paper is organized as follows: Section 2 provides a review of related studies. Section 3 discusses GA, autoencoders, and the parameter optimization of autoencoders using GA. Section 4 presents the experimental setup, results, discussions, and a comparison with state-of-the-art methods. Finally, Section 5 concludes the study and outlines future research directions [39-43].

2.0 Literature Review

In this work [44], a unique deep autoencoder-based feature learning approach is proposed for defect diagnostics in rotating equipment. To enhance feature learning from observed vibration signals, a novel deep autoencoder loss function is designed using maximum cross entropy. Additionally, the primary parameters of the deep autoencoder are optimized using an artificial fish swarm algorithm to improve learning from input signal characteristics. The proposed approach can be applied to fault detection in roller bearings of gearboxes or electrical locomotives. Experimental findings validate the superiority of this method over existing techniques in feature learning and fault diagnostics [45-47].

In this paper [48], a novel Human Pose Recovery (HPR) system is introduced, improving upon previous methods by employing a new framework that simultaneously learns joint localization and joint detection. This framework consists of two key components: (1) regularization for multiple manifolds is computed and integrated, and (2) autoencoders are utilized to develop robust representations of images and joints. This results in a multi-task learning framework specifically designed for HPR. Experimental evaluations on the HumanEva-I and Human3.6M datasets demonstrate that the proposed method outperforms existing HPR techniques. The proposed [49], Optimal Deep Autoencoder Network-Based Website Phishing Detection and Classification (ODAE-WPDC) model employs input data pre-processing to eliminate missing values from the dataset. Next, feature selection (FS) is performed based on feature extraction results using the Artificial Algae Algorithm (AAA). The selected features are then fed into a Deep Autoencoder (DAE) model for classification, with its parameters optimized using the Invasive Weed Optimization (IWO) method to enhance performance. The Kaggle dataset was used to validate the ODAE-WPDC model, and testing results confirm its superior performance, achieving a maximum accuracy of 99.28%. This study proposed [50] a novel deep learning-based model, AdacDeep, is introduced for predicting various types of cyberattacks. It integrates an Enhanced Genetic Algorithm (EGA), a Deep Autoencoder, and a Deep Feedforward Neural Network (DFFNN) with backpropagation learning. The CICIDS2017 and UNSW NB15 datasets, widely recognized benchmarks, are used to evaluate the model's performance. Experimental results demonstrate that AdacDeep outperforms existing state-of-the-art models, achieving prediction accuracy improvements of 0.22–35% and F-Score enhancements of 0.1–34.7%. In this study [51], the Genetic Algorithm-Based Autoencoder (GAAE) model is proposed to address data imbalance. Initially, both majority and minority samples, along with genetic operators, are used to train an autoencoder, where the chromosome effectively represents the autoencoder structure. A fitness function is established using an error function and a set of classifiers. To balance the dataset, the optimized autoencoder generates synthetic data for the minority class. After applying GAAE to equalize the data distribution, feature selection is performed using the correlation coefficient. Finally, various classifiers, including Multi-Layer Perceptron (MLP), k-Nearest Neighbors (k-NN), C4.5 Decision Tree (DT), and Random Forest (RF), are employed for classification [52-56].

3.0 Autoencoder

An autoencoder is a type of neural network designed to learn a compressed representation of an input and then reconstruct the original input with minimal loss of information. It consists of two main components: an encoder and a decoder. The encoder compresses the input into a lower-dimensional latent representation, while the decoder reconstructs the input from this compressed form. The objective of an autoencoder is to minimize the difference between the input and its reconstruction, typically measured using a reconstruction loss function. For example, given an image of a handwritten digit or alphabet, an autoencoder encodes it into a latent space representation and then decodes it back into an image. Through this process, the model learns how to efficiently compress data while minimizing reconstruction errors [21, 57-59].

In this study, an autoencoder was implemented using the MNIST dataset [60], the step-by-step procedure for implementing the autoencoder is as follows: First, the necessary libraries, such as TensorFlow and NumPy, were imported at the beginning of the code. Next, the MNIST dataset was loaded for training the autoencoder. The data was then preprocessed by scaling the pixel values to the range [0, 1], which was achieved by dividing all pixel values by 255. After preprocessing, the data was flattened to be used as input for the autoencoder by reshaping it from (num_samples, 28, 28) to (num_samples, 784). Once the data was prepared, the autoencoder architecture was built. An autoencoder consists of two main components: an encoder and a decoder. The encoder maps the input data to a lower-dimensional latent space, while the decoder reconstructs the input from this compressed representation. The general architecture of the autoencoder is illustrated in Figure 1. Table 1 presents the pseudocode for a autoencoder [61-64].

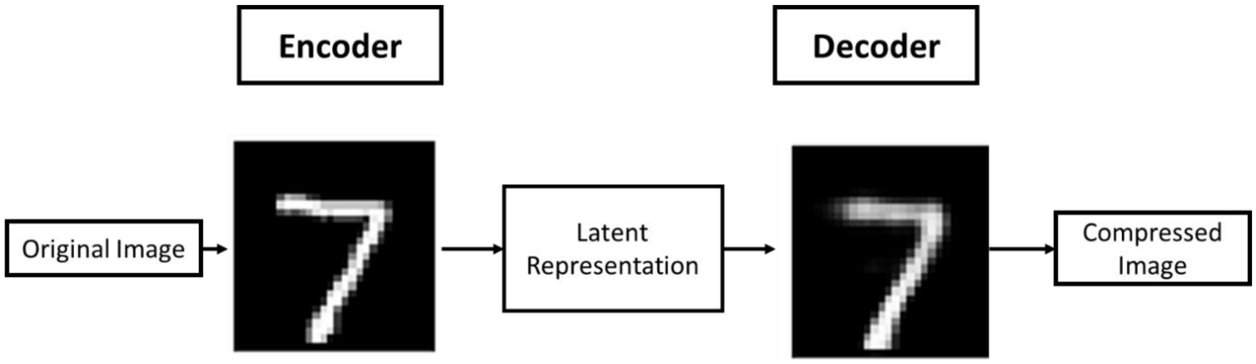

Figure 1: General Architecture of Autoencoder

Table 1: Pseudocode for autoencoder

Pseudocode for autoencoder

```
# Define the architecture of the autoencoder
1. input_dim = ... # dimensionality of the input data
2. encoding_dim = ... # dimensionality of the encoded representation
3. decoder_layers = ... # list of layer sizes for the decoder, in reverse order
4. encoder_layers = ... # list of layer sizes for the encoder, in forward order
# Define the input and output tensors
5. input_data = ... # placeholder for input data
6. encoded_data = ... # output of the encoder
7. decoded_data = ... # output of the decoder
# Define the encoder layers
8. for layer_size in encoder_layers:
   encoded_data = Dense(layer_size, activation='relu')(encoded_data)
# Define the decoder layers
9. for layer_size in decoder_layers:
   decoded_data = Dense(layer_size, activation='relu')(decoded_data)
# Define the final layer to reconstruct the input
10. decoded_data = Dense(input_dim, activation='sigmoid')(decoded_data)
# Define the autoencoder as a keras model
11. autoencoder = Model(input_data, decoded_data)
# Compile the autoencoder with an appropriate loss function and optimizer
12. autoencoder.compile(optimizer='adam', loss='binary_crossentropy')
# Train the autoencoder on input data
autoencoder.fit(x_train, x_train, epochs=num_epochs, batch_size=batch_size)
```

4.0 Genetic Algorithm (GA)

GA is a metaheuristic optimization algorithm inspired by the process of natural selection. It follows a population-based approach to search for an optimal solution to a given problem. In GA, potential solutions (referred to as individuals) are represented as chromosomes, typically encoded as bit strings. A fitness function is then defined to evaluate the quality of everyone. The algorithm operates by first generating a random population of individuals. Then, it evaluates their fitness scores and selects the fittest individuals for reproduction through crossover and mutation. This process of selection, crossover, and mutation is repeated over multiple generations, enabling the algorithm to gradually converge toward the optimal solution. GA is widely applicable to various optimization problems, particularly those involving non-differentiable, nonlinear, or discontinuous objective functions. Table 2 presents the algorithm for a genetic algorithm used for parameter optimization.

Table 2: Algorithm of GA

Algorithm of GA

```
1. START
2. Generate an initial population of candidate solutions
3. Compute the fitness of each candidate solution
4. REPEAT
   Select the best-fit candidate solutions from the population
   Create new offspring candidate solutions through crossover and mutation
   Compute the fitness of each new candidate solution
   Replace the least-fit candidate solutions with the new offspring candidate solutions
5. UNTIL the population has converged, or a stopping criterion is met
6. OUTPUT the best candidate solution found during the optimization process
7. STOP
```

5.0 Parameter optimization of autoencoder using GA

The parameters of an autoencoder that are typically optimized include the weights and biases of both the encoder and decoder neural networks. These parameters are trained using an optimization algorithm to minimize a reconstruction loss function, which quantifies the difference between the input and the reconstructed output of the autoencoder. During training, the autoencoder processes an input, encodes it into a lower-dimensional representation, and then decodes it back to its original dimensionality. The reconstruction loss function compares the original input with the output of the decoder and computes the error. The optimization algorithm then updates the weights and biases of the encoder and decoder to minimize this reconstruction loss, ensuring better feature learning and representation.

Autoencoder with random parameters, where '*X*' is random variable, and '*W*' is weights for these parameters.

1st Iteration

$$X^1 = (W_1, W_2, W_3, \ldots\ldots\ldots\ldots, W_n \tag{1}$$

$$X^2 = (W_1, W_2, W_3, \ldots\ldots\ldots\ldots, W_n \tag{2}$$

$$X^3 = (W_1, W_2, W_3, \ldots\ldots\ldots\ldots, W_n \tag{3}$$

$$X^n = (W_1, W_2, W_3, \ldots\ldots\ldots\ldots, W_n \tag{4}$$

After the 1st iteration, we got n/2 optimized parameters

2nd Iteration

In this iteration '*n/2*' parameters used from previous iteration and rest of the parameters are selected randomly.

$$X^1 = (W_1, W_2, W_3, \ldots\ldots\ldots\ldots, W_n \tag{5}$$

$$X^2 = (W_1, W_2, W_3, \ldots\ldots\ldots\ldots, W_n \tag{6}$$

$$X^3 = (W_1, W_2, W_3, \ldots\ldots\ldots\ldots, W_n \tag{7}$$

$$X^n = (W_1, W_2, W_3, \ldots\ldots\ldots\ldots, W_n \quad (8)$$

nth Iteration

$$X^1 = (W_1, W_2, W_3, \ldots\ldots\ldots\ldots, W_n \quad (9)$$

$$X^2 = (W_1, W_2, W_3, \ldots\ldots\ldots\ldots, W_n \quad (10)$$

$$X^3 = (W_1, W_2, W_3, \ldots\ldots\ldots\ldots, W_n \quad (11)$$

$$X^n = (W_1, W_2, W_3, \ldots\ldots\ldots\ldots, W_n \quad (12)$$

After iteration obtained objective function as mentioned in equation no 13.

$$\boldsymbol{X^{min}} = \mathrm{f(x)} = \mathrm{V} \quad (13)$$

Where 'V' is minimum for the loss function.
Such that

$$X \leq \mathrm{g} \geq \mathrm{h} \quad (14)$$

Where g=1 and h=0, due to the images, which are normalized before processing.
'X' is parameters that need to optimize, W' is weights of these parameters. In the 1st iteration got 'n/2' parameters that use used next iteration rest of weights selected randomly. *The nth* iteration done as result got best optimized parameters. After each iteration parameters are given to models to check whether results improved or not. 'V' is a loss function. 'g' is maximum range that is 1 and 'h' is minimum range and that is 0.
In addition to optimizing the weights and biases, other hyperparameters of an autoencoder—such as the learning rate, batch size, number of hidden layers, and activation functions—can also be fine-tuned using techniques like grid search or randomized search. These hyperparameters significantly influence the performance and generalization of the autoencoder, and careful tuning is required to achieve optimal results. A GA is a metaheuristic search algorithm that simulates the process of natural selection to optimize solutions. It starts with a population of candidate solutions, each evaluated using a fitness function. The fittest individuals are selected for reproduction, where crossover and mutation operations introduce variations in the population. Over successive generations, GA refines the solutions, converging toward an optimal or near-optimal configuration for the given problem.

6.0 The Proposed Methodology

Experiments were conducted using the MNIST dataset to evaluate the effectiveness of GA-based optimization for autoencoders. Initially, classification was performed using a default set of parameters. Subsequently, GA was employed to optimize these parameters iteratively. After each iteration, the algorithm adjusted the hyperparameters and network weights, refining the settings to achieve improved performance. This process continued until an optimal configuration was found, ensuring that the autoencoder performed efficiently with enhanced accuracy and minimal reconstruction loss.

6.1 Dataset

In this study, three 3 datasets used MNIST [65], EMNIST: Extending MNIST to handwritten letters [60], Fashion-mnist [66]. These datasets are used to evaluate performance of autoencoder using default optimizer and with genetic algorithm as optimizer.

6.2 Experimental Setup

The experiments were conducted using the MNIST dataset, where classification was initially performed with default parameters. Subsequently, a GA was applied to iteratively optimize these parameters. After each iteration, GA adjusted the hyperparameters and network settings, refining them to enhance the autoencoder's performance. This optimization process continued until the optimal parameters were obtained, ensuring improved accuracy and efficient reconstruction.

6.2.1 Preprocessing

The MNIST dataset was preprocessed by scaling pixel values to the range [0,1]. This was achieved by dividing each pixel value by 255.0, a standard normalization technique in image classification tasks. Normalizing the input data helps neural networks perform more efficiently, improving gradient flow, convergence speed, and overall model accuracy. By transforming the pixel values into a more suitable format, the preprocessing step enhances the autoencoder's ability to learn meaningful features during training.

6.2.2 Data Reshaping and Splitting

The MNIST dataset was flattened so it could be used as input to the autoencoder. Each 28×28 image was reshaped into a 1D vector of 784 features. The dataset was then divided into training and testing samples, ensuring that both sets contained the same number of features. The reshape method was used to maintain equal feature representation in both the training and testing arrays. This step ensures that the data is in a suitable format for feeding into the neural network, allowing the autoencoder to learn effectively.

6.2.3 Autoencoder Architecture

An autoencoder is an unsupervised learning neural network architecture designed to learn efficient representations of input data. It consists of two main components: an encoder that maps the input data to a lower-dimensional latent space and a decoder that reconstructs the original input from this compressed representation. The primary objective of an autoencoder is to minimize the reconstruction error, which measures the difference between the original input and the regenerated output. By doing so, the autoencoder captures the most essential features of the data while discarding noise and redundant information.

- **Encoder**

The tf.keras.models.Sequential function is used to create a sequential model, where each layer is connected in a linear sequence to the next layer in the network. The first dense layer is added using the tf.keras.layers.Dense function with 32 units and a ReLU activation function. The input_shape argument defines the shape of the input data, which is a 1D array of length 784 (representing a flattened image). This layer serves as the input layer of the encoder, receiving the flattened image. The second dense layer, also created using tf.keras.layers.Dense, consists of 16 units with a ReLU activation function. It acts as a hidden layer within the encoder, further processing the input data by reducing the number of units from 32 to 16. This reduction helps in compressing the input into a lower-dimensional space, a common approach in auto-encoder architecture.

- **Decoder**

The decoder is implemented using a sequential model from the Keras API in TensorFlow, consisting of two dense layers. The first dense layer has 32 units with a Rectified Linear Unit (ReLU) activation function and an input shape of (16,), meaning it takes in input vectors of size 16. The second dense layer has 784 units with a sigmoid activation function, producing output values between 0 and 1 for each of the 784 units. The sigmoid activation function is commonly used in binary classification tasks to generate probability values indicating class membership. In image generation tasks, it helps represent pixel intensities in the range [0, 1].

- **Combine Encoder and Decoder**

The autoencoder is implemented as a sequential model comprising an encoder and a decoder. The encoder and decoder are separate models defined earlier in the code. The encoder processes the input data and generates a compressed representation. The decoder takes this compressed representation and reconstructs the original input. By combining both models into a single sequential structure, an end-to-end autoencoder is created, enabling training to learn efficient data compression and reconstruction. The training objective is to minimize the difference between the original input and the reconstructed output, typically using a loss function such as mean squared error (MSE).

6.2.4 Compile the autoencoder model

To compile the autoencoder model, we specify the optimizer, loss function, and evaluation metrics.

6.2.4.1 Using Adam Optimizer

An autoencoder model is designed using the Adam optimizer and the binary cross-entropy loss function, with accuracy as the evaluation metric. Autoencoders are a type of neural network used for unsupervised learning, where the objective is to reconstruct the input data at the output layer. The Adam optimizer is an adaptive optimization algorithm that dynamically adjusts the learning rate during training, improving convergence speed and overall performance. The binary cross-entropy loss function measures the difference between the predicted and actual outputs, making it particularly effective for

Spectrum of Engineering Sciences
ISSN (e) 3007-3138 (p) 3007-312X Volume 3, Issue 4, 2025

binary classification tasks where the outputs are represented as either 0 or 1.

6.2.4.2 Using GA

It is possible to implement a Genetic Algorithm (GA) in an autoencoder model using a binary cross-entropy loss function and an accuracy metric. However, this approach requires significant modifications to both the model architecture and the training process. Unlike traditional optimization methods, GA operates as a population-based search algorithm, iteratively refining solutions through evolutionary principles. GA mimics the process of natural selection, where a population of candidate solutions, each representing a unique set of model parameters, evolves over multiple generations. The algorithm applies selection, mutation, and crossover operations to improve these solutions, ensuring that better-performing candidates have a higher chance of propagating their characteristics. To integrate GA with an autoencoder, the first step is to define a population of potential solutions, each corresponding to different sets of model parameters. The autoencoder is then trained using these parameters, and its performance is evaluated based on binary cross-entropy loss and accuracy using cross-validation. After assessing the fitness of each solution, genetic operations are applied to evolve the population toward optimal parameter configurations. This iterative process continues until the autoencoder achieves improved reconstruction performance and classification accuracy.

6.2.5 Train the autoencoder on the training data

The autoencoder model is trained using the fit() method, where each input serves as both the input and the target output. Training is conducted over 100 epochs with a batch size of 256, and the data is shuffled before each epoch to enhance learning stability. During training, the model aims to minimize the binary cross-entropy loss between the original input and its reconstructed output. The Adam optimizer is employed to adjust the model's weights in a direction that reduces this loss, ensuring better convergence. The validation_data parameter is set to the test dataset, allowing the model to be evaluated on unseen data during training. This validation step helps monitor the model's generalization ability and detect potential overfitting.

7.0 Results and Discussions

The MNIST dataset is used for experiments, where classification is initially performed using default parameters. Afterward, a GA is applied to optimize these parameters iteratively. GA continues refining the parameters until an optimal set is found, enhancing the model's performance.

Table 3: Comparison of the results with the state-of-the-art methods

References	Evolutionary algorithms used to optimize parameters	Evaluation measure
[44]	Artificial Fish Swarm	Average accuracy= 94.05% Average computation time= 37.67 s learning rate, momentum and sparsity parameter are set to 0.1, 0.85 and 0.02
[49]	Invasive weed optimization (IWO)	Accuracy of 99.28%.

[50]	Enhances GA	Accuracy=0.22–35%,
		F-Score=0.1–34.7%.

The Proposed	Adam optimizer	Average accuracy=97.77% Average computation time=34.77 s learning rate, momentum and sparsity parameter are set to 0.001, 0.87 and 0.01
	GA	Average accuracy=98.85 % Average computation time= 35.44 s

		learning rate, momentum and sparsity parameter are set to 0.1, 0.85 and 0.01

In this study [44], an Artificial Fish Swarm optimizer was utilized, achieving an average accuracy of 94.05% and an average computation time of 37.67 seconds. The learning rate, momentum, and sparsity parameter were set to 0.1, 0.85, and 0.02, respectively. This study suggests that future deep learning techniques could play a significant role in advancing rotating machinery fault diagnosis. In this study [49], the Invasive Weed Optimization (IWO) algorithm was employed to optimize the parameters, achieving an accuracy of 99.28%. Figure 3 illustrates the comparison between the results obtained using default parameters and those optimized through the GA, in terms of accuracy.

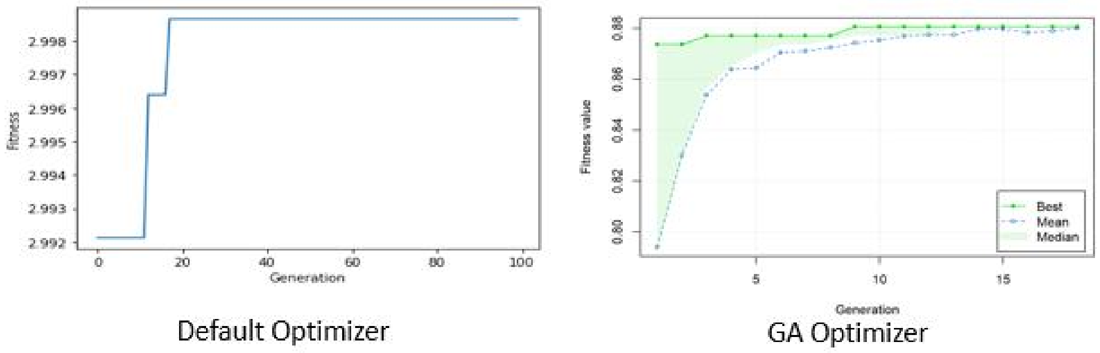

Figure 3: Accuracy with default optimizer and with ga optimizer

8.0 Conclusion

Optimizing an autoencoder's parameters using a Genetic Algorithm (GA) can significantly enhance its performance. GAs are optimization techniques inspired by natural selection, making them useful for finding optimal parameter sets for complex problems. When applying GA to autoencoder optimization, it is crucial to define appropriate selection criteria, such as reconstruction error or other performance metrics. Additionally, fine-tuning GA parameters is essential to maintain a balance between exploration (searching new solutions) and exploitation (refining existing solutions) while ensuring sufficient population diversity.

Overall, GA-based parameter optimization can outperform traditional methods like grid search or random search by more efficiently navigating the search space. However, its effectiveness depends on the complexity and the quality of implementation.

REFERENCES

1.Gilanie G, Batool SN, Khursheed A, Shafique H, Mahmood N, Cheema S, et al. Bit Pattern Selection Based Novel Method of Steganography in RGB Encoding Scheme Based Digital Images.

2.Gilanie G, Batool SN, Shafique H, Khursheed A, Mahmood N, Cheema S, et al. An Overview on X-Rays ImagesProcessing: Methods, Challenges& Issues, and Future Work.

3.Gilanie G, Javedb M, Rauf B, Cheemaa S, Latif A, Perveena S, et al. RiceAgeNet: Age Estimation of Pakistani Grown Rice Seeds using Convolutional Neural Networks.

4.Khera EA, Ullah H, Hussain F, Abubakar S, Majeed A, Tabssum I, et al. Characterization of Nickel Oxide Thin Films for Smart Window Energy Conversion Applications: Comprehensive Experimental and Computational Study. Available at SSRN 4235112.

5.Ullah H, Jahangir A, Gilanie G. Classification of Chronic Kidney Diseases with Statistical Analysis of Textural Parameters: A Data Mining Technique.

6.Attique M, Gilanie G, Mehmood MS, Naweed MS, Ikram M, Kamran JA, et al. Colorization and

automated segmentation of human T2 MR brain images for characterization of soft tissues. PloS one. 2012;7(3):e33616.
7.Gilanie G, Attique M, Naweed S, Ahmed E, Ikram M. Object extraction from T2 weighted brain MR image using histogram based gradient calculation. Pattern Recognition Letters. 2013;34(12):1356-63.
8. Asghar K, Gilanie G, Saddique M, Habib Z. Automatic Enhancement Of Digital Images Using Cubic BÃ© zier Curve And Fourier Transformation. Malaysian Journal of Computer Science. 2017;30(4):300-10.
9.Janjua HU, Andleeb F, Aftab S, Hussain F, Gilanie G. Classification of liver cirrhosis with statistical analysis of texture parameters. International Journal of Optical Sciences. 2017;3(2):18-25.
10.Ullah H, Andleeb F, Aftab S, Hussain F, Gilanie G. Classification of Liver Cirrhosis with Statistical Analysis of Texture Parameters. IJOS. 2017;3(2):1-8.
11.Bajwa UI, Shah AA, Anwar MW, Gilanie G, Ejaz Bajwa A. Computer-aided detection (CADe) system for detection of malignant lung nodules in CT slices-a key for early lung cancer detection. Current Medical Imaging. 2018;14(3):422-9.
12.Gilanie G, Bajwa UI, Waraich MM, Habib Z, Ullah H, Nasir M. Classification of normal and abnormal brain MRI slices using Gabor texture and support vector machines. Signal, Image and Video Processing. 2018;12:479-87.
13.Gilanie G, Ullah H, Mahmood M, Bajwa UI, Habib Z. Colored Representation of Brain Gray Scale MRI Images to potentially underscore the variability and sensitivity of images. Current Medical Imaging Reviews. 2018;14(4):555-60.
14.Janjua HU, Jahangir A, Gilanie G. Classification of chronic kidney diseases with statistical analysis of textural parameters: a data mining technique. International Journal of Optical Sciences. 2018;4(1):1-7.
15.Ullah H, Batool A, Gilanie G. Classification of Brain Tumor with Statistical Analysis of Texture Parameter Using a Data Mining Technique. International Journal of Industrial Biotechnology and Biomaterials. 2018;4(2):22-36.
16.Gilanie G. Automated Detection and Classification of Brain Tumor from MRI Images using Machine Learning Methods: Department of Computer Science, COMSATS University Islamabad, Lahore campus; 2019.
17.Gilanie G, Bajwa UI, Waraich MM, Habib Z. Automated and reliable brain radiology with texture analysis of magnetic resonance imaging and cross datasets validation. International Journal of Imaging Systems and Technology. 2019;29(4):531-8.
18.Gilanie G, Bajwa UI, Waraich MM, Habib Z. Computer aided diagnosis of brain abnormalities using texture analysis of MRI images. International Journal of Imaging Systems and Technology. 2019;29(3):260-71.
19.Gilanie G, Bajwa UI, Waraich MM, Anwar MW. Risk-free WHO grading of astrocytoma using convolutional neural networks from MRI images. Multimedia Tools and Applications. 2021;80(3):4295-306.
20.Gilanie G, Bajwa UI, Waraich MM, Asghar M, Kousar R, Kashif A, et al. Coronavirus (COVID-19) detection from chest radiology images using convolutional neural networks. Biomedical Signal Processing and Control. 2021;66:102490.
21.Batool SN, Yang J, Gilanie G, Latif A, Yasin S, Ikram A, et al. Forensic Radiology: A robust approach to biological profile estimation from bone image analysis using deep learning. Biomedical Signal Processing and Control. 2025;105:107661.
22.Kovenko V, Bogach I, editors. A Comprehensive Study of AutoencodersApplications Related to Images. International Conference" Information Technology and Interactions"(IT&I-2020) Workshops Proceedings, Kyiv, Ukraine, December 02-03, 2020: 43-54; 2020: **Київський національний університет імені Тараса Шевченка**.
23.Gilanie G, Nasir N, Bajwa UI, Ullah H. RiceNet: convolutional neural networks-based model

to classify Pakistani grown rice seed types. Multimedia Systems. 2021:1-9.
24.Gilanie G, Saher A, Batool SN, Khursheed A, Shafique H, Perveen S, et al. Digital Image Processing for Ultrasound Images: A Comprehensive. Digital Image Processing. 2021;15(3).
25.Rafiq M, Bajwa UI, Gilanie G, Anwar W. Reconstruction of scene using corneal reflection. Multimedia Tools and Applications. 2021;80(14):21363-79.
26.Ghaffar AA, Mushtaq MF, Amna, Akram U, Samad A, Gilanie G, et al., editors. Refined Sentiment Analysis by Ensembling Technique of Stacking Classifier. International Conference on Soft Computing and Data Mining; 2022: Springer.
27.Hartman TW, Radichev E, Ali HM, Alaba MO, Hoffman M, Kassa G, et al. BASIN: A Semi-automatic Workflow, with Machine Learning Segmentation, for Objective Statistical Analysis of Biomedical and Biofilm Image Datasets. Journal of Molecular Biology. 2023;435(2):167895.
28.Wang G, Shao M, Lv S, Kong X, He Z, Vining G. Process parameter optimization for lifetime improvement experiments considering warranty and customer satisfaction. Reliability Engineering & System Safety. 2022;221:108369.
29.Chalmers G. Introducing ligand GA, a genetic algorithm molecular tool for automated protein inhibitor design. Scientific Reports. 2022;12(1):20877.
30.Wright AH. Genetic algorithms for real parameter optimization. Foundations of genetic algorithms. 1: Elsevier; 1991. p. 205-18.
31.Ali HM, Liu J, Bukhari SAC, Rauf HT. Planning a secure and reliable IoT-enabled FOG-assisted computing infrastructure for healthcare. Cluster Computing. 2022;25(3):2143-61.
32.Patel N, Patel S, Mankad SH. Impact of autoencoder based compact representation on emotion detection from audio. Journal of Ambient Intelligence and Humanized Computing. 2022:1-19.
33.Zhang C, Geng Y, Han Z, Liu Y, Fu H, Hu Q. Autoencoder in Autoencoder Networks. IEEE transactions on neural networks and learning systems. 2022.
34.Gilanie G, Asghar M, Qamar AM, Ullah H, Khan RU, Aslam N, et al. An Automated and Real-time Approach of Depression Detection from Facial Micro-expressions. Computers, Materials & Continua. 2022;73(2).
35.Gilanie G, Rehman N, Bajwa UI, Sharif S, Ullah H, Mushtaq MF, editors. FERNet: A Convolutional Neural Networks Based Robust Model to Recognize Human Facial Expressions. International Conference on Soft Computing and Data Mining; 2022: Springer.
36.Iqbal MJ, Bajwa UI, Gilanie G, Iftikhar MA, Anwar MW. Automatic brain tumor segmentation from magnetic resonance images using superpixel-based approach. Multimedia Tools And Applications. 2022;81(27):38409-27.
37.Rubab SF, Mushtaq MF, Tahir MH, Amna, Samad A, Gilanie G, et al., editors. The Comparative Performance of Machine Learning Models for COVID-19 Sentiment Analysis. International Conference on Soft Computing and Data Mining; 2022: Springer.
38.Wazir E, Gilanie G, Rehman N, Ullah H, Mushtaq MF, editors. Early Stage Detection of Cardiac Related Diseases by Using Artificial Neural Network. International Conference on Soft Computing and Data Mining; 2022: Springer.
39.Yaseen M, Khurshed A, Ullah H, Batool Z, Nazir A, Gilanie G, et al. In-vitro Evaluation of Anticancer Activity of Rhodamine-640 perchlorate on Rhabdomyosarcoma cell line. 2022.
40.Afzal F, Ullah H, Amjad M, Akhtar M, Shah MI, Batool Z, et al. Detection of Uric Acid in UV-VIS wavelength Regime. JOURNAL OF NANOSCOPE (JN). 2023;4(1):75-81.

41.Ahmed M, Gilanie G, Ahsan M, Ullah H, Sheikh FA. Review of Artificial Intelligence-based COVID-19 Detection and A CNN-based Model to Detect Covid-19 from X-Rays and CT images. VFAST Transactions on Software Engineering. 2023;11(2):100-12.
42.Asghar S, Gilanie G, Saddique M, Ullah H, Mohamed HG, Abbasi IA, et al. Water classification using convolutional neural network. IEEE Access. 2023;11:78601-12.
43.Batool SN, Gilanie G. CVIP-Net: A Convolutional Neural Network-Based Model for Forensic Radiology Image Classification. Computers, Materials & Continua. 2023;74(1).
44.Shao H, Jiang H, Zhao H, Wang F. A novel deep autoencoder feature learning method for rotating machinery fault diagnosis. Mechanical Systems and Signal Processing. 2017;95:187-204.
45.Ghani M, Gilanie G. The IOMT-Based Risk-Free Approach to Lung Disorders Detection from Exhaled Breath Examination. INTELLIGENT AUTOMATION AND SOFT COMPUTING. 2023;36(3):2835-47.
46.Gilanie G, Bajwa UI, Waraich MM, Anwar MW, Ullah H. An automated and risk free WHO grading of glioma from MRI images using CNN. Multimedia tools and applications. 2023;82(2):2857-69.
47.Hafeez HA, Elmagzoub MA, Abdullah NAB, Al Reshan MS, Gilanie G, Alyami S, et al. A CNN-model to classify low-grade and high-grade glioma from mri images. IEEE Access. 2023;11:46283-96.
48.Yu J, Hong C, Rui Y, Tao D. Multitask autoencoder model for recovering human poses. IEEE Transactions on Industrial Electronics. 2017;65(6):5060-8.
49.Alqahtani H, Alotaibi SS, Alrayes FS, Al-Turaiki I, Alissa KA, Aziz ASA, et al. Evolutionary Algorithm with Deep Auto Encoder Network Based Website Phishing Detection and Classification. Applied Sciences. 2022;12(15):7441.
50.Ibor AE, Oladeji FA, Okunoye OB, Uwadia CO. Novel adaptive cyberattack prediction model using an enhanced genetic algorithm and deep learning (AdacDeep). Information Security Journal: A Global Perspective. 2022;31(1):105-24.
51.Ram PK, Kuila P. GAAE: a novel genetic algorithm based on autoencoder with ensemble classifiers for imbalanced healthcare data. The Journal of Supercomputing. 2022:1-32.
52.Khera EA, Ullah H, Hussain F, Abubakar S, Majeed A, Tabssum I, et al. Characterizing nickel oxide thin films for smart window energy conversion applications: Combined experimental and theoretical analyses. ChemistrySelect. 2023;8(37):e202302320.
53.Nazir A, Ullah H, Gilanie G, Ahmad S, Batool Z, Gadhi A. Exploring Breast Cancer Texture Analysis through Multilayer Neural Networks. Scientific Inquiry and Review. 2023;7(3):32-47.
54.Shafiq H, Gilanie G, Sajid M, Ahsan M. Dental radiology: a convolutional neural network-based approach to detect dental disorders from dental images in a real-time environment. Multimedia Systems. 2023;29(6):3179-91.
55.Gilanie G, Cheema S, Latif A, Saher A, Ahsan M, Ullah H, et al. A Robust Method of Bipolar Mental Illness Detection from Facial Micro Expressions Using Machine Learning Methods. Intelligent Automation & Soft Computing. 2024;39(1).
56.Naveed S, Husnain M, Alsubaie N, Samad A, Ikram A, Afreen H, et al. Drug efficacy recommendation system of glioblastoma (GBM) using deep learning. IEEE Access. 2024.
57.Rashid MS, Gilanie G, Naveed S, Cheema S, Sajid M. Automated detection and classification of psoriasis types using deep neural networks from dermatology images. Signal, Image and Video Processing. 2024;18(1):163-72.
58.Saher A, Gilanie G, Cheema S, Latif A, Batool SN, Ullah H. A Deep Learning-Based Automated Approach of Schizophrenia

Detection from Facial Micro-Expressions. Intelligent Automation & Soft Computing. 2024;39(6).
59.Gilanie G, Batool SN, Abbas SN, Cheema S, Latif A, Shafique H, et al. DEEP LEARNING-BASED APPROACH FOR ESTIMATING THE AGE OF PAKISTANI-GROWN RICE SEEDS. Spectrum of Engineering Sciences. 2025;3(1):557-72.
60.Cohen G, Afshar S, Tapson J, Van Schaik A, editors. EMNIST: Extending MNIST to handwritten letters. 2017 international joint conference on neural networks (IJCNN); 2017: IEEE.
61.Gilanie G, Batool SN, Abbas SN, Latif A, Iqbal M, Shafique H, et al. STEGANOGRAPHIC SECRET COMMUNICATION USING RGB PIXEL ENCODING AND CRYPTOGRAPHIC SECURITY. Spectrum of Engineering Sciences. 2025;3(3):323-36.
62.Gilanie G, Batool SN, Abbas SN, Shafique H, Iqbal M, Cheema S, et al. READABLE TEXT RETRIEVAL FROM NOISE-INFLUENCED DOCUMENTS USING IMAGE RESTORATION METHODS. Spectrum of Engineering Sciences. 2025;3(3):337-60.
63.Sajid M, Sharif W, Gilanie G, Mazher M, Iqbal K, Akhtar MA, et al. IoMT-Enabled Noninvasive Lungs Disease Detection and Classification Using Deep Learning-Based Analysis of Lungs Sounds. International Journal of Advanced Computer Science & Applications. 2025;16(2).
64.Siddique MA, Akhtar M, Majid MA, Khera EA, Ahmad M, Gilanie G, et al. A Multi-Modal Approach for Exploring Sarcoma and Carcinoma Using FTIR and Polarimetric Analysis. Microscopy Research and Technique. 2025.
65.Jansson Y, Lindeberg T. MNIST Large Scale data set. 2020.
66.Xiao H, Rasul K, Vollgraf R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:170807747. 2017.