AUTOENCODER FOR IMAGE CLASSIFICATION WITH GENETICS ALGORITHMS

John Tsiligaridis

Department of Math and Computer Science, Heritage University Toppenish, WA, USA

ABSTRACT

Autoencoders (AEs) are Deep Learning (DL) models that are well known for their ability to compress and reconstruct data. When an AE compresses input data, a latent space is created which yields a compressed representation of the original data with a smaller set of features. Genetic Algorithms (GAs) based on evolutionary principles can be used to optimize various hyperparameters of a DL model.

This work involves two tasks. First, it focuses on the application of an AE on image data along with various configurations of the AE structure and its constituent encoder/decoder structure using Multi-Layer Perceptrons (MLPs). Visualizations of the AE loss functions during training are provided, along with various latent space results obtained using clustering techniques. The second focus of the paper is on the application of the GA on a Convolutional AE where optimization of the Convolutional Neural Networks (CNN) encoder/decoder structures is done by converting the architecture into genes for image classification. We see that the AE is a flexible and robust model that can successfully be applied on a variety of image datasets and the GA model initially surpasses the AE model. After discovering the appropriate hyperparameters values the performance of AE can be improved and predominate the one of the GA model.

KEYWORDS

Machine Learning, Deep Learning, Autoencoders, Genetic Algorithms

1. INTRODUCTION

AEs are encoder/decoder-based DL models that can be used for many purposes such as: dimensionality reduction, feature extraction, etc. The AE has a compressed latent space between the encoder/decoder which is a lower-dimensional space that captures the essential features of the input data. This latent space is a powerful concept in machine learning that provides a way to represent complex data in a manageable form.

The fundamental parameters of the latent space are explored in the experiments to yield improved AE performance. Additionally, the experiments focus the application of the AE on the digit MNIST dataset, along with various configurations of the AE structure and the application of t-SNE and k-Means clustering to visualize the latent space in a low dimensional setting.

In [1], an explanation of the principle of a convolutional AE and its primary development process are shown. In [2], the different types of AEs along with their operations are described. The description of different variants of AEs along with their relationships are discussed in [3]. In [4], an effective and reliable DL method known as stacked denoising AE (SDAE) for process pattern recognition in manufacturing processes is presented.

Regarding the second focus of this work, the manual optimization process for a DL model can require substantial effort and time. Grid search for DL model parameters is not suitable for many hyperparameter combinations. GAs are optimization techniques based on evolutionary principles to find solutions to complex problems. The GA involves the following processes: Encoding the solution, initialization, evaluation, selection, crossover, mutation, and creation of the new generation.

The CNN is described in [5] with introduction, overview, building blocks of CNN, different architecture of CNN, applications in several Domain areas, issues, and challenges.

The study of the application of GA in immunology for the task of solving the OneMax problem, which is to maximize the number of 1s in a bit string of length n, is described in [6]. In [7], the genetic operators and their usages, considering also hybrid algorithms, are discussed. In [8], an algorithm called Genetic Algorithm Based On Natural Selection Theory (GABONST) is proposed, along with an extension using Extreme Learning Machine (ELM). The ELM is considered as one of the most useful learning models for carrying out classification and regression analysis.

In [9] GAs are particularly suitable for optimization problems in which an effective system design or set of parameter values is sought. In nature, genetic regulatory networks (GRNs) form the basic control layer in the regulation of gene expression levels. A new algorithm is presented in [10] named gene-pool optimal mixing evolutionary algorithm (GOMEA) with explicitly design to estimate and exploit linkage information. In [11] a Genetic Algorithm-based Scheduling Tool (GAST) has been developed for the scheduling of complex products with multiple resource constraints and deep product structure. This method produces significantly better delivery performance and resource utilization than the Company plans. A chapter in [12] introduces two popular methods for unsupervised representation learning using Neural Networks namely autoencoders and variational autoencoders. While autoencoders and variational autoencoders share the same general idea, they differ significantly in their theoretical foundations and abilities. The chapter outlines the theoretical foundations of both methods, discusses their advantages and practical challenges, outlines some of their various extensions,

In [13] a novel deep clustering algorithm is presented that utilizes a variational autoencoder (VAE) framework with an entangled multi encoder-decoder neural architecture that can produce for each cluster high quality synthetic examples. Novel autoencoders for unsupervised feature-learning from hyperspectral data is presented in [14].

The encoder model comprises the CNN with MaxPooling layers, where the input size is reduced per layer. The automatic optimization of a CNN model can be achieved using GAs. Encoding a CNN into a variable length gene size sequence is achieved with the rule that all will start with a convolutional layer and finish with a dense layer (architecture encoding). The construction process deals with the addition of a new convolutional layer, using BN/pooling layer.

Among the usual hyperparameter tuning, the main purpose is concentrated at the network architecture itself. The automating hyperparameter optimization of a DL model using Evolutionary Methods (EMs) can be advantageous. There are two approaches that are presented for Convolutional AEs in this work, both of which are applied on the Fashion-MNIST dataset.

In the first configuration, a convolutional AE is applied using CNNs as the encoder/decoder mechanism, where the CNN contains dropout and Batch Normalization (BN) layers that can reduce memorization and avoid the vanishing/exploding gradient issue, respectively. With this approach, the loss function is unsatisfactory.

In the second configuration, the Convolutional AE structure is optimized using the GA, where the encoding of the architecture into a gene sequence is carried out. A construction process for individuals includes the creation of the gene sequence, the crossover and mutation, and the check for additional layers. The best individual is selected, and with the evolution, better loss performance is achieved for network optimization.

The paper organization is as follows. In Section 2 the Autoencoders ML models are included. Section 3 deals with Genetic Algorithms and Convolutional Autoencoder.

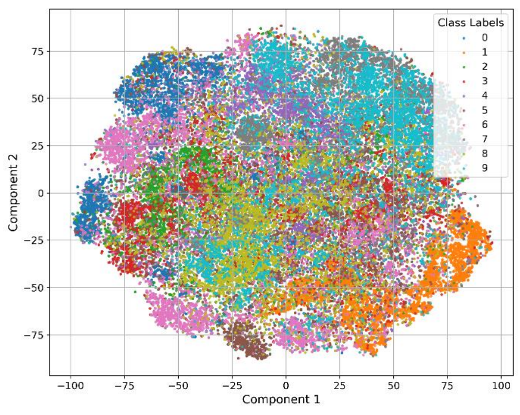
2. AUTOENCODERS

AEs are deep learning architectures that are used for a variety of tasks such as dimensionality reduction, data generation, anomaly detection, and feature extraction. An AE is an encoder/decoder structure that uses an encoder to compress high dimensionality input data, x, to a smaller dimension, z, that captures the most meaningful features of the input data. A decoder also exists that is used to reconstruct the original input data from the compressed representation; an estimate $\hat{x}$ is created that aims to replicate x. Both the encoder and decoder are neural networks, and the choices of what neural networks to use depends on the input data at hand.

2.1. Autoencoders for Clustering

For our case, we aim to apply the AE to image data, specifically MNIST image data of numerical digits, and study its components in detail through performance results and visuals. We begin by taking image data and transforming it into a one-dimensional vector which will be our input vector.

The digits MNIST data has standard train/test data that we use to train and test our AE. We train the AE using a cross-entropy loss function over 100 epochs. To visualize the training procedure, we look at the compressed representation of the data, z, which we select to be 20. We apply t-SNE to the compressed data to get a 2D representation of the compression that takes place after the encoder in the AE. Figure 1a shows at epoch 0, the color-coded (based on the digit of the data point) embedding space color-coded visualizations, where each point represents a data sample from the training set. All the points seem scattered as the AE has not yet learned the structure of the data. Figure 1b shows that after 100 epochs of training, the AE understands the patterns in the data and the embedding space representations are in clusters based on the 10 different digits in the MNIST data.



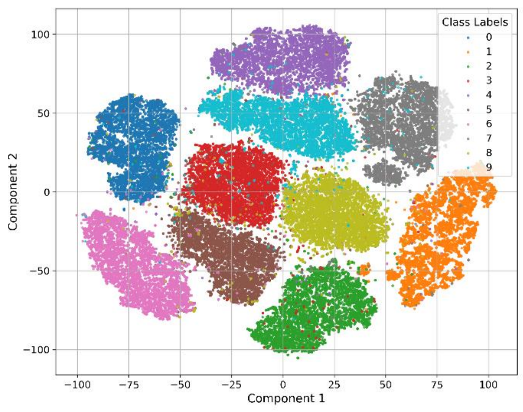


Figure 1a. visualization at Epoch 0 Figure 1b. visualization at Epoch 99.

Figure 2 shows this result from test data where we look at the embedding space representations, apply t-SNE to get the data in 2D, and then apply k-Means clustering with $k = 10$ to obtain clusters of the low dimensional data. The same cluster pattern is obtained, as in Figure 1b and the obtained clusters correctly match with the test labels.

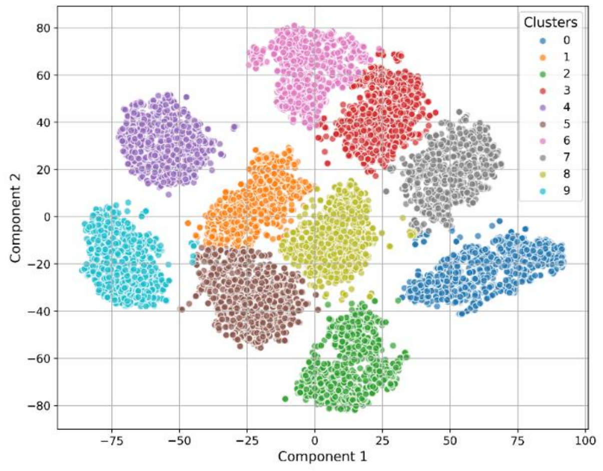


Figure 2. K-means clustered embedding space visualization

The latent space in the AE allows for a smaller representation of the input. For visualization, we apply a two-dimensional t-SNE to the latent space representation so we can observe the compressed data in the embedding space. As the AE is trained, the data points transition from being scattered at the start of training process to being more focused and belonging to specific groups of data at the end of training. The t-SNE is an invaluable tool for visualizing and understanding the complex relationships within the latent space of an AE. By effectively preserving local structure and revealing underlying patterns, t-SNE allows us to gain insights into the AE's feature learning capabilities.

By applying clustering algorithms (i.e. K-means) to the latent space representations generated by the AE, we can group similar data points together and create clusters. These clusters represent data points that are of the same image label.

If the AE has effectively learned meaningful features, the clusters will correspond to meaningful categories or semantic groupings within the image dataset.

2.2. Latent Space Size

The choice of the latent space size (z) can affect the performance of the AE. As the AE trains, we can look at the embedding space outputs, but we can further look at the cross-entropy loss function. In Figure 3, we look at how the choice of z affects the loss function. A small z results in higher loss values and a slower convergence. Using a large z can yield small losses, but we want to avoid using a large z as the data compression will only be minimal. Thus, we aim to use a medium z value, such as 20, which yields a small loss, quick convergence, and a good amount of data compression for representing the input data efficiently in low dimensions.

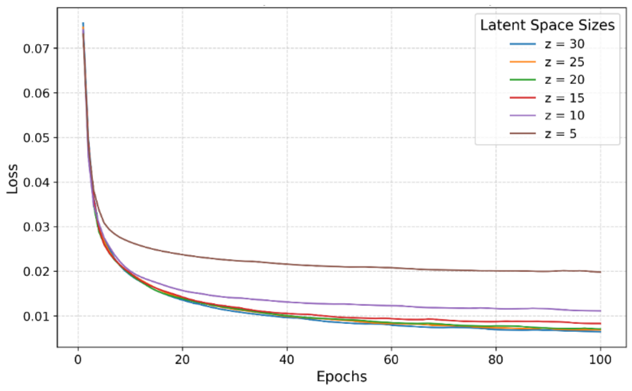


Figure 3. Loss function comparisons of various embedding space sizes.

2.3. Hidden Layers and Size

The user choice of the encoder/decoder with different sizes of a single layer and a number of hidden layers can affect the performance of the AE. A single hidden layer MLP of length l. Is used. Figure 4 displays loss functions for encoder/decoder MLP sizes of 1, 2, and 3 hidden layers. All choices yield similar loss functions, but the single layer choice yields a quicker convergence. Figure 5 depicts, for varied length of the single layer MLP, that smaller length hidden layer configurations of the MLP yield larger losses; this can be explained as the AE not being able to learn a model of the input data due to such harsh compressions.

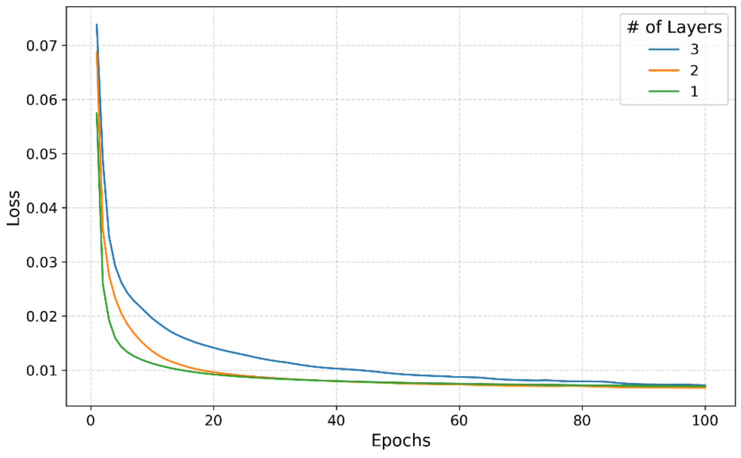


Figure 4. Loss function comparisons of multi-layer encoder/decoder setups.

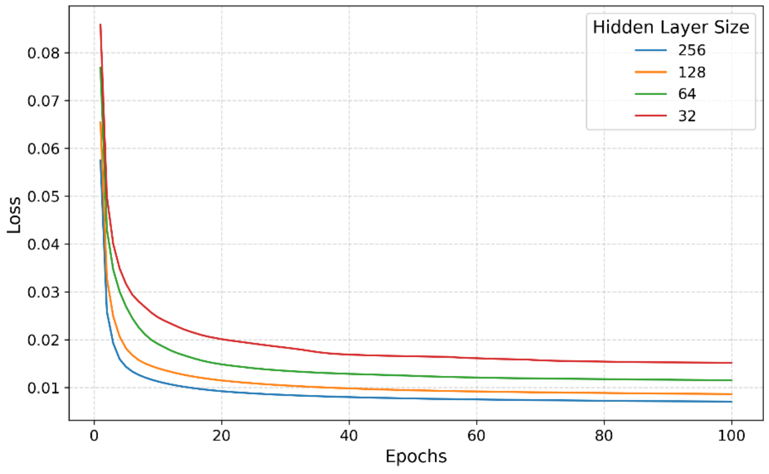


Figure 5. Loss function comparisons of single-layer encoders/decoders of varying lengths.

3. Genetic Algorithms and Convolutional Autoencoder

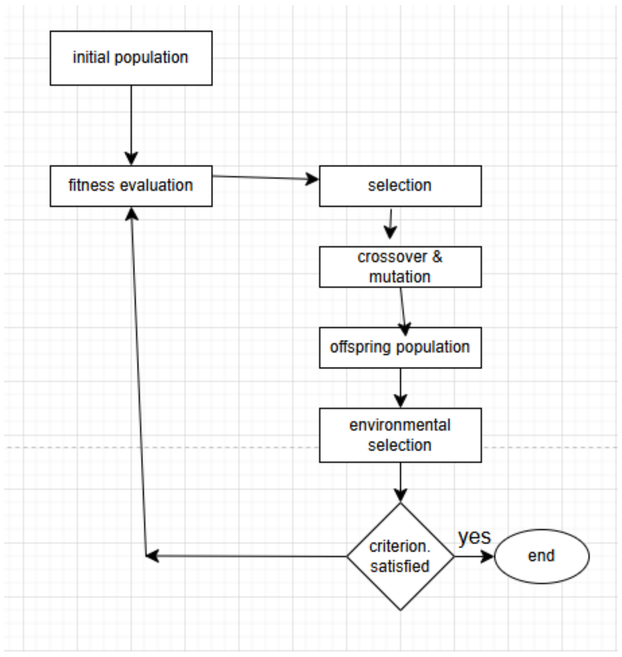
The focus here is the application of the GA on the Convolutional AE to optimize the parameters of the encoder/decoder CNNs in the AE architecture. When applying the GA, the following take place; encoding the solution, initialization, evaluation, selection, crossover, mutation, create the new generation. During the selection phase, individuals from the current population are evaluated based on a fitness function, quantifying how well each solution solves the problem. The individuals with higher fitness values are more likely to be selected for further processing, simulating the survival of the fittest. The GA continues the selection, crossover, and mutation process for a fixed number of generations or until a termination criterion is met.

The pseudocode and the architecture diagram are as below:

```
Pseudocode for GA  model
Input: a set of values for GA parameters
            (population size, the maximum generation number, etc.,
             the image dataset for classification)
Output: the discovered best architecture of AE.
$P_0$: initialize a population with the given population
      size using the proposed encoding strategy
t=0;
while t< maximum generation number do
      evaluate the fitness of each individual in $P_t$
      $Q_t$: generate offspring from the selected parent
      Individuals using the proposed mutation and the
      crossover operators
      $P_{t+1}$ = selection from $P_t \cup Q_t$;
      t= t+1;
end
return the individual having the best fitness in $P_t$.
```

Specifically, the current population is composed of the parent population and the generated offspring population. Finally, the counter is increased by one, and the evolution continues until the counter exceeds the predefined maximal generation. Each individual means the particular CNN architecture). The population is randomly initialized with the predefined population size, using the proposed encoding strategy. Population initialization are the layers of encoder and decoder (i.e. convolutional layers, pooling layers etc.)

The architecture diagram of the GA model will include a series of evolutionary processes until the proposed algorithm can discover the best architecture of the AE to classify the fashion MNIST dataset.



In fitness evaluation each individual of the Pt is evaluated and following the process of selection, crossover and mutation create the offspring generating. In the environmental selection process the P_t individuals are selected from the current population P_t U Q_t and then these selected individuals are placed into the next population P $_{t+1}$. Also, the best individual is selected and placed into P $_{t+1}$. For the computational overhead of GAs, GA can struggle with complex interactions between genes and are computationally expensive. These issues make them less efficient for large-scale problems and require significant time and resources to find the best solutions. The factors for computational overhead of GA are: Fitness Function evaluation, large population size, many generations, complex genetic operators, and parameter tuning (it needs significant experimentation and computational effort).

The AE uses CNNs to better extract features in images; the GA is applied to the CNNs to optimize parameters as an alternative approach to the standard backpropagation used in conventional DL problems. The construction process, which works with both the encoder and decoder, deals with the addition of a new convolutional layer, with batch normalization (BN)/pooling layer. The encoding of the architecture into a gene sequence is done using GA. Encoding a CNN into a variable length gene size sequence is achieved with the rule that all will start with a convolutional layer and finish with a dense layer (architecture encoding).

The encoder and decoder are based on CNN. The CNN layers are created (convolutional and max pooling, and Batch Normalization -BN-) starting by encoding the network architecture of a CNN model into gene sequence (chromosome) of an individual. After that, additional hyperparameters are defined for each layer type.

GA can be used as DL optimizers, instead of using back propagation (i.e. gradient descent). GA is used with a function that sets the parameters of the GA model (probabilities of crossover and mutation, number of genes in the individual, population size, and number of generations) and it allows search for values with the use of GA. The number of model parameters is the number of genes. For the architecture of the model, a function can create the needed layers for the encoder and decoder and creates the gene sequence. Finally, the build function generates models from the gene sequences using the mating and mutation operators.

The convolutional AE model can be created by using different number of parameters (i.e. convolutional layers, etc.). The selection of the appropriate parameter values can provide better performance. This combination of parameters cannot provide successful results. Figure 6a displays the loss functions for the standard Convolutional AE model. Figure 6b shows the GA model (optimized Convolutional AE). The loss functions in 6a can be improved, as shown in the result in 6b, where the loss functions drop quicker, and a smaller validation loss is observed over training epochs. The GA optimized model outperforms the standard one due to the advantage of selecting the best hyperparameters for the network architecture using the GA process.

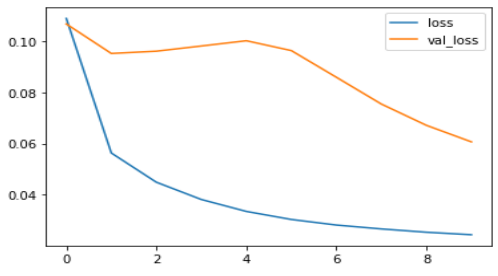


Figure 6a. AE model

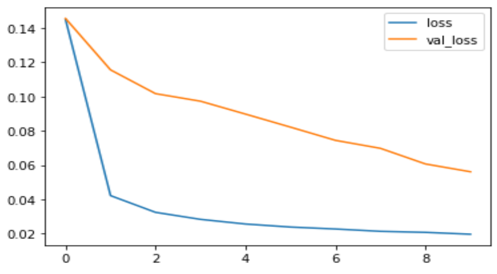


Figure 6b. GA model

These two experiments show the superiority of the GA model since the AE model's different parameter values are selected manually. After subtracting the BN and dropout layers of the AE architecture, the AE can provide better performance than the CA model as in Figure 6c. From the above, the problem is based on discovering the values of the optimized hyperparameters for the AE model to play a predominate role. The AE model can be competitive compared to the GA model after appropriate selection of the architectural structure.

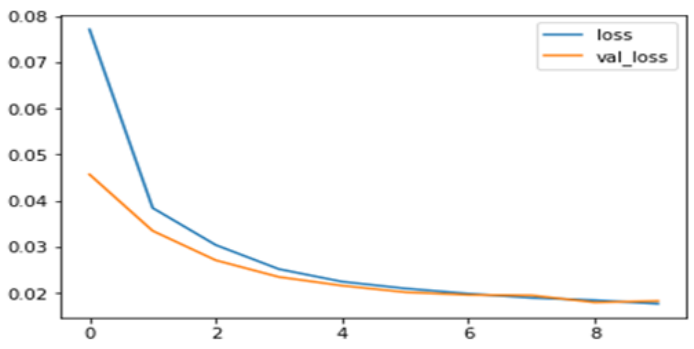


Figure 6c AE revised model

The GA model, which includes the Genetic Algorithm and the Convolutional AE, can be used to achieve a degree of generalization in image datasets. This can be realized by feature selection, parameter optimization and model selection. With the feature selection the GA model can identify the most informative feature and generalize well to unseen data. The feature selection for image classification includes: the representation of each chromosome, and the fitness function. The genetic operators, selecting the fittest chromosome based on the classification accuracy/loss on the validation set. GA can optimize the hyperparameters of the model which can lead to generalization of unseen data. Also, GA can select the best model architecture for a given dataset. This can be achieved by evaluating the performance of different model structures on a validation set. In this way GA can identify models that can be generalized well. As an example of medical image dataset could be the Breast Cancer Histopathology Images and for natural image dataset, the Caltech 101.

The AE can be generalized well and there are some factors that can improve it. Regularization that prevents overfitting, data augmentation (i.e. rotating images, adding noises), and careful architecture design (i.e. selecting the correct number of layers). As an example of image datasets are: CIFAR-10, CelebA. Medical datasets: Chest X-rays, MRI/CT scans, MIMIC-III, natural datasets: BCI Competition Datasets, Xero-Canto.

When the gene sequences are relatively small, we should generally expect a quick convergence. Although Genetic Algorithms has proved to be a fast and powerful problem-solving approach, some limitations are found embedded in it. Some of these limitations are discussed below.

One of the main issues with GAs is their slow convergence rate. This means that they can take a long time to find the optimal solution, especially for complex problems. This is because GAs rely on random search techniques, which can be inefficient when the search space is large.

The weak point of a genetic algorithm is that it often suffers from so-called premature convergence to suboptimal solutions, getting stuck in local optima. which is caused by an early homogenization of genetic material in the population. This means that no valuable exploration can be performed anymore. Techniques to mitigate the issue are increasing the population size, adjusting the mutation rate, and using crossover techniques (i.e. inversion crossover).

The computational cost can be computationally expensive because of evaluating the fitness of each problem especially for a complex one. This can make the GA model inappropriate for large-scale problems. GAs do not scale well with complexity. That is when the number of elements exposed to mutation is large there is often an exponential increase in search space size.

Strategies for improving scalability include parallel processing (i.e. embarrassingly parallel fitness) using efficient Data Structures (i.e. trees, hash tables0) to store and retrieve information about individuals and their fitness values. Also, distributed GAs (i.e.by dividing the population into subpopulations (islands) and the Fine-Grained Parallelism (parallelize operations within a single generation).

4. Conclusions

This work focused on the application of AEs on various image datasets. Using the AE structure, the tasks of data compression, low dimensional feature representation, and image classification using clustering were completed. The presented experiments studied many aspects of the AE framework, such as their latent embedding space structure and loss function performance under various configurations. The digit MNIST dataset was used.

For the optimization of large DL networks, apart from standard DL hyperparameter optimization methods, the GA can be used as an alternative approach to optimizing DL models. The

secondary focus of this work, which was the application of the GA on the Convolutional AE, showed that the encoder/decoder components could successfully be optimized using the GA and this would yield improved model performance. The fashion MNIST data set was used.
Future work could benefit from testing on more complex datasets, incorporating additional performance metrics, and addressing scalability challenges.

REFERENCES

[1] [1] Berahmand, K., Daneshfarm, F., Salehi, E., Li, Y., Xu, Y. (2024) "Autoencoders and their applications in machine learning: a survey", Springer Nature, Vol. 57, No. 28 https://link.springer.com/article/10.1007/s10462-023-10662-6

[2] Bank, D., Koenigstein, N., Giryes, R. (2021), "Autoencoders", book chapter, http:/arxiv.org/pdf/2003.05991

[3] Chen, S., Guo, W. (2023), "Auto-Encoders in Deep Learning—A Review with New Perspectives", Mathematics Journal, Vol.11, No: 8.

[4] Yu, J., Zheng, X., Wang, S. (2019)."A deep autoencoder feature learning method for process pattern recognition", Journal of Process Control, Elsevier. Vol.79, pp. 1-15.

[5] Pandey, K., Patel, S. (2023). Deep Learning with Convolutional Neural Networks: from Theory to Practice. In 7th International Conference on Trends in Electronics and Informatics (ICOEI), Tirunelveli, India.

[6] McCall, J. (2005), "Genetic algorithms for modelling and optimization", Journal of Computation and Applied Mathematics, Vol.184, pp. 205-222.

[7] Katoch, S., Chauhan, S., Kumar, V. (2021)," A review on genetic algorithm: past, present, and future", Springer Nature, Vol. 57, No. 28, https://link.springer.com/article/10.1007/s11042-020-10139-6

[8] Albadr, M., Tiun, S., Ayob, M., Al-Dhief, F. (2020), " Genetic Algorithm Based on Natural Selection Theory for Optimization Problems", Journal Symmetry, Vol. 12, Issue 11.

[9] Knabe, J., Wegner, K., Nehaniv, C., Schilstra, M., (2010), "Genetic algorithms and their application to in silico evolution of genetic regulatory networks" PubMed, NIH, PMID: 20835807, DOI: 10.1007/978-1-60761-842-3_19 https://pubmed.ncbi.nlm.nih.gov/20835807/

[10] Dushatsky, A., Virgolin, M., Bouter, A., Thierens, D., Bosman, P., (2024), "Parameterless Gene-Pool Optimal Mixing Evolutionary Algorithms", Journal Evolutionary Computation (ECJ), MIT Press Direct, Vol. 32, Issue 4 pp. 371-397

[11] Pongcharoen, P., Hicks, C., Braiden, P., Stewardson, D., (2002),"Determining optimum Genetic Algorithm parameters for scheduling the manufacturing and assembly of complex products", International Journal of Production Economics, ,Elsevier, Vol. 78, Issue 3, pp. 311-322.

[12] Ehrhardt, J., Wilms, M.,(2022), "Chapter 8- Autoencoders and variational autoencoders in medical image analysis", Journal Biomedical Image Synthesis and Simulation, Methods and Applications, The MICCAI Society book Series, pp.129- 162

[13] Caciularu, A., Goldebergerm J., (2023), "An entangled mixture of variational autoencoders approach to deep clustering", Neurocomputing, Elsevier, Vol. 529, Issue 7, pp.182-189

[14] Windrim, L., Ramakrishnan, R., Melkumyan, A., Murphy, R., Chlingaryan, A.,(2019), "Unsupervised Feature-Learning for Hyperspectral Data with Autoencoders", Journal Remote Sensing, MDPI, Vol.11, Issue 7.