Advances in Artificial Intelligence and Machine Learning; Research 4 (3) 2629-2647 Received 3-07-2024; Accepted 20-09-2024; Published 27-09-2024

Integration of an Autoencoder Model with an Actor-Oriented System

Sithembiso Dyubele ctheradyubele@gmail.com
Department of Information Systems, Durban University of Technology
Durban, South Africa

Noxolo Pretty Cele noxolocele53@gmail.com
Department of Information Systems, Durban University of Technology
Durban, South Africa

Lubabalo Mbangata lubabalo.mbangata@gmail.com
Department of Information Systems, Durban University of Technology
Durban, South Africa

Corresponding Author: Sithembiso Dyubele

Abstract

Traditional machine learning frameworks often struggle with scalability, modularity, and efficient resource management, especially when dealing with vast data. Actor-Oriented Systems offer a robust framework for building such scalable systems, allowing concurrent processing and efficient handling of large datasets. This study investigated the integration of Autoencoders (AE), which are pivotal in unsupervised learning, with Actor-Oriented Systems to enhance the modularity, scalability, and maintainability of the model training process. The study seeks to leverage the capabilities of AE and Actor-Oriented Systems to achieve high-quality image reconstruction and efficient processing. The study also attempted to understand the underlying patterns in the data, assess the performance of the model, and demonstrate the benefits of modular and scalable systems. Key findings from the results showed significant improvements in training efficiency and performance of the model, especially when using Actor-Oriented Systems. The training time was reduced from 16.96 seconds to 14.21 seconds, and the validation loss improved from 0.2768 to 0.2100, indicating better generalisation and learning. Data augmentation techniques further enhanced the robustness of the model, leading to more accurate reconstructions of the test images. Actor-Oriented Systems facilitated concurrent processing, improved modularity, and enabled the system to scale efficiently with increasing data volume. This study also highlighted the practical benefits of integrating AE with Actor-Oriented Systems, providing valuable insights into building more robust, maintainable, and scalable machine learning workflows.

Keywords: Autoencoder, Actor-oriented system, Machine Learning.

Citation: Sithembiso Dyubele et al. Integration of an Autoencoder Model with an Actor-Oriented System. Advances in Artificial Intelligence and Machine Learning. 2024;4(3):153.

1. INTRODUCTION

With the increasing complexity and volume of data in various domains, the need for scalable and modular systems has become more critical [1]. Li et al. [1], further indicated that there is a developing interest among researchers in combining various methods of deep learning or machine learning to mitigate challenges related to the complexity and volume of data in multiple industries. This study integrated an AE with an Actor-Oriented System to address challenges in handling large and complex datasets in machine learning. Zhang et al. [2], described AE as a form of neural network used especially for unsupervised learning tasks, such as dimension reduction, efficient coding, and generative modelling. Zhang et al. [2], further indicated that AE is one of the significant methods utilised to capture the main features of data.

Similarly, Xu et al. [3], indicated that Autoencoders have demonstrated great superiority in learning latent feature representation in various application domains, e.g., image recognition, computer vision, and speech recognition. As for Actors, Kumar et al. [4], define Actor-Oriented System as a computing paradigm that provides concurrent units of computation. Similarly, Galkin and Shkilniak [5], revealed that Actor-Oriented Systems offer a robust framework for building scalable systems by allowing concurrent processing and efficient handling of large datasets. This is corroborated by Srirama et al. [6], for whom the Actor-Oriented System addresses the need to work in a distributed environment with concurrency, resiliency, and scalability requirements. By implementing and evaluating an AE model with an Actor-Oriented System, the study seeks to understand the underlying patterns in the data, assess the performance model, and demonstrate the benefits of modular and scalable systems.

1.1 Motivation

The primary motivation for this study stems from the increasing importance of unsupervised learning techniques and the development of scalable, modular systems in machine learning. Hurtado et al. [7], revealed that unsupervised learning, where the model is trained on data without explicit labels, is crucial for discovering hidden patterns and representations. Similarly, Mansour et al. [8], indicated that Autoencoders are particularly effective in tasks such as data compression, noise reduction, and feature extraction, while Kumar et al. [4], described Actor-Oriented System as an enormous system that enables concurrent processing, which is crucial for managing the extensive computational requirements of modern machine learning tasks. This study seeks to leverage the capabilities of AE and Actor-Oriented Systems to achieve high-quality image reconstruction and efficient processing. By encapsulating different components of the training process within actors, these systems improve the modularity and maintainability of the workflow, making it easier to manage and scale.

The specific motivations include:

- Exploring the effectiveness of Autoencoders in learning compact and meaningful representations of complex image data.
- Evaluating the performance of Autoencoders.

- Investigating the benefits of data augmentation techniques in improving the generalisation ability of the model.
- Implementing an Actor-Oriented System to modularise the training and evaluation process improves scalability and maintainability. Demonstrating the advantages of integrating AE with Actor-Oriented Systems through detailed visualisations and mathematical formulations.

The current study aims to contribute to the growing body of knowledge on AE and its applications in unsupervised learning. It also demonstrates the practical benefits of integrating AE with Actor-Oriented Systems to enhance the scalability and efficiency of machine learning workflows.

1.2 Objectives

The objectives of this study are multifaceted and aimed at exploring the potential of AE for image reconstruction and the benefits of using Actor-Oriented Systems in machine-learning workflows. The specific objectives include:

- **Implementation of Autoencoder (AE) Models:** The first objective is to implement an AE model using the Fashion MNIST dataset. This involves defining the architecture of the AE, including the encoder and decoder components, and training the model to learn effective representations of the input data.
- **Evaluation of Model Performance:** After implementing the AE, the next objective is to evaluate its performance in terms of quality reconstruction. This includes assessing the training and validation losses over multiple epochs and comparing the original and reconstructed images to visually inspect the performance of the model.
- **Data Augmentation:** To improve the generalisation capability of the Autoencoder, another objective is to apply data augmentation techniques by artificially expanding the training dataset through transformations such as rotation, shifting, and flipping, which aims to enhance the robustness of the model.
- **Actor-Oriented System Implementation**: One of the core objectives is to integrate an Actor-Oriented System for training and evaluating the AE model. This involves encapsulating the training process within an actor, enabling modular and scalable handling of the model training.
- **Concurrent Processing:** Leveraging the Actor-Oriented approach, the study aims to implement concurrent processing for encoding and decoding images. By using Actors for these tasks, the study seeks to demonstrate the efficiency gains in handling large datasets and the potential for parallel execution.
- **Scalability and Modularity**: Finally, the study aims to demonstrate the benefits of scalability and modularity in machine learning workflows using Actor-Oriented Systems. This includes discussing the ease of extending the system for more complex models and larger datasets.

By achieving these objectives, this study aims to contribute valuable insights into using AE for image reconstruction and the advantages of Actor-Oriented Systems in enhancing the scalability and

modularity of machine learning processes. The outcomes are expected to provide a comprehensive understanding of both theoretical and practical aspects of integrating Autoencoders with Actor-Oriented Systems.

1.3 Architecture of the Study

FIGURE 1 illustrates the architecture of the current study. The Summary of the Process is explained below:

- **Load and Preprocess Data**: Loaded the Fashion MNIST dataset, visualised samples, and normalized the images.
- **Data Augmentation:** Applied various augmentation techniques to enhance the training dataset.
- **Train the Autoencoder Model**: Defined and trained an autoencoder model on the augmented dataset.
- **Reconstruct Images:** Used the trained autoencoder to reconstruct test images.
- **Implement Actor-Oriented System:** Data Preprocessing Actor: Handled the normalisation of images.
- **Model Training Actor:** Managed the training of the autoencoder model.
- **Encoding and Decoding Actor**: Performed encoding and decoding operations using the trained model.
- **Use Actors for Asynchronous Tasks:** Leveraged threading to simulate actor behavior, ensuring modular and asynchronous processing of tasks.

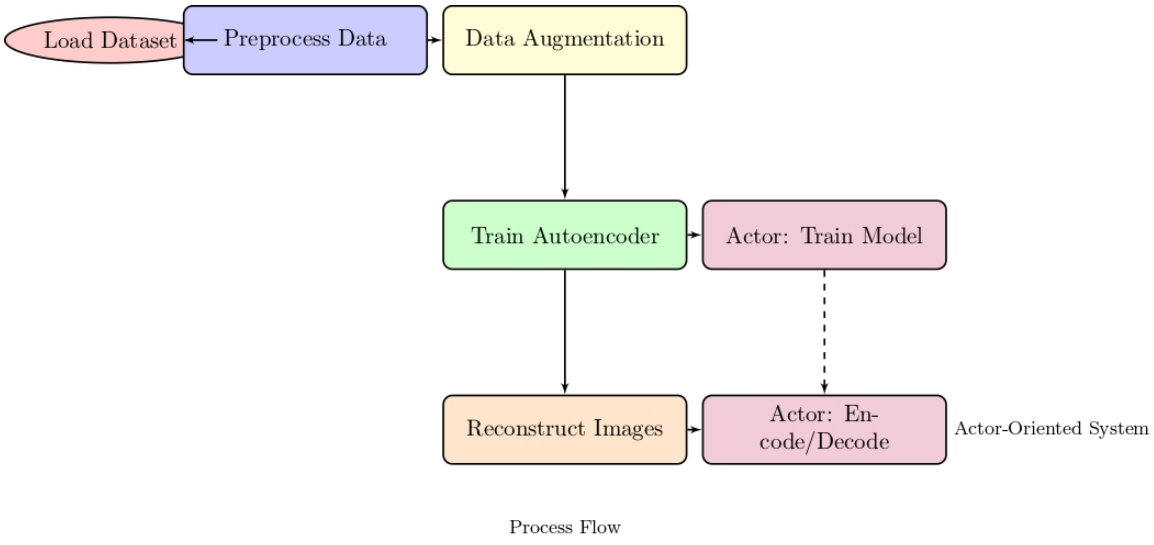

Figure 1: Architecture of the Study

2. REVIEWED LITERATURE

2.1 The Architecture of Autoencoders

According to Andresini et al. [9], AE involves three parts in its construction. This includes Encoder, Bottleneck, and Decoder.

- **Encoder:** This layer compresses and changes the input information into a code layer [10].
- **Bottleneck:** Chen et al. [11], revealed that the bottleneck is one of the essential aspects of the neural network. Chen et al. [11], further indicated that the bottleneck controls the movement of information from the encoder to the decoder. This helps so that only significant information can be processed and reach the destination.
- **Decoder:** This is the last and final stage after the bottleneck. Sun et al. [12], claim that this is a layer where the bottleneck is reconstructed. Sun et al. [12], also revealed that the decoder part recreates the original input from the compressed vector representation (the latent space vector).

2.2 Actor-Oriented Systems

Srirama et al. [6], Actor-Oriented Systems offers a robust framework for building scalable and modular applications by enabling concurrent processing. Actors have been integrated into this study to enhance the modularity, scalability, and maintainability of the AE training process. Kumar et al. [4], indicated that every Actor has a mailbox to store received messages. After a message has been received, the Actor executes three basic actions. This includes:

- Sending messages among actors.
- Creating new actors.
- Modifying its state.

3. METHODOLOGY

This section outlines the steps taken to achieve the objectives of this study, including the implementation of the Autoencoder model, data preprocessing, Actor-Oriented System integration, and performance evaluation. Each step is detailed below to provide a comprehensive understanding of the process.

3.1 Dataset Preparation

The first step in the methodology involves the preparation of the dataset for training the AE model. The Fashion MNIST dataset, consisting of 70,000 grayscale images of 28x28 pixels each, is used.

The dataset is divided into 60,000 training images and 10,000 test images, each associated with a label from 10 different classes of clothing items. The following has been performed:

- **Loading the Dataset:** The dataset is loaded using the tensorflow.keras.datasets module, splitting it into training shapes of 60,000 images and test datasets of 10,000 images, each with 28x28 pixel images, and their dimensions are verified by printing the shapes to provide an overview of the data dimensions.
- **Visualisation:** A subset of the training images is visualised to understand the types of images and their corresponding labels, helping to gain initial insights into the dataset. This visual examination, as shown in FIGURE 2, confirms the dataset is correctly loadedand familiarizes the user with the variety of clothing items represented, highlighting the importance of data visualisation [13].

Figure 2: Visualisation of Training Images with Labels

- **Normalisation:** The pixel values of the images in the Fashion MNIST dataset are normalised to the range of [0, 1] to stabilize and speed up the training process of the Autoencoder model. This normalisation is achieved by dividing each pixel value by 255, to ensure that the neural network receives input data on a consistent scale, enhancing training efficiency [14]. The following formula was applied to compute normalisation. Given an image x with pixel values in the range [0, 255], the normalised image x' is computed as:

$$x' = \frac{x}{255} \tag{1}$$

Histograms demonstrate the spreading of pixel values previously and later normalisation, signifying its efficiency, as illustrated in FIGURE 3.

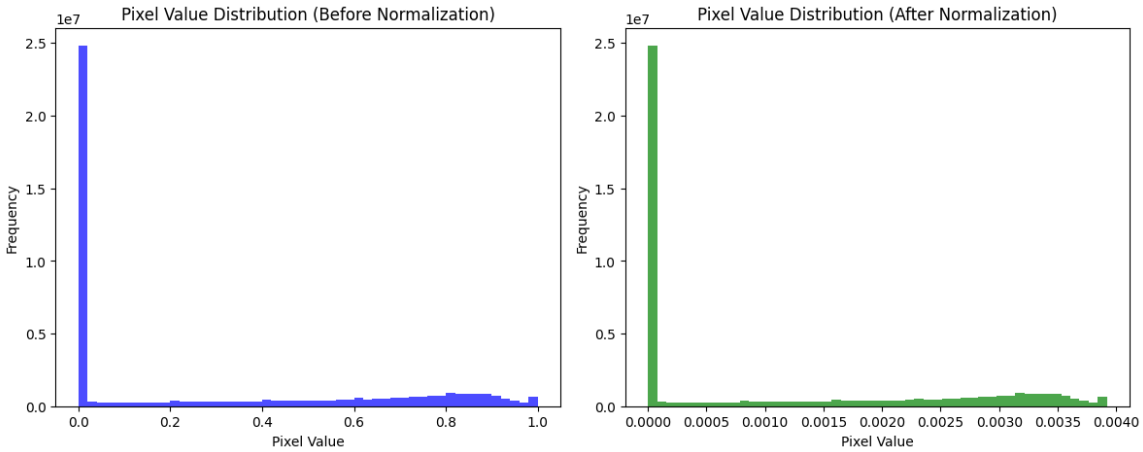

Figure 3: Pixel Value Distribution Before and After Normalisation

- **Data Augmentation:** Data expansion procedures are functional to theatrically increase the size of the drill dataset. Alterations such as rotation, shifting, shear, zoom, and horizontal flip are utilised to improve the enhance of the model. This method produces an improved form of the original images assisting to avoid overfitting and refining the Autoencoder's generalisation ability [15]. FIGURE 4 shows the examples of the amplified images, approving the submission of these alterations.

3.2 Building the Autoencoder

The next step involves building the AE model. Gu et al. [16], claim that an AE is a type of artificial neural network used to learn efficient coding of unlabeled data. The architecture of the AE built in the current study consists of an encoder and a decoder. It is elaborated below:

- **Encoder**: The encoder compresses the input into a lower-dimensional representation. It consists of an input, flattened, and dense layer with *Rectified Linear Unit (ReLU)* or *rectifier* activation function. The encoder maps the input x to a latent space representation z:

$$z = f(x) = \sigma(W_e x + b_e) \tag{2}$$

Figure 4: Examples of Augmented Images

Where W_e and b_e are the weight matrix and bias vector of the encoder, and σ is the activation function (e.g., *ReLU*).

- **Decoder:** The decoder reconstructs the original input from the lower-dimensional representation. It consists of a dense layer with *sigmoid* activation and a reshaped layer to restore the original image dimensions. The decoder reconstructs the input $\hat{x}$ from the latent representation z:

$$\hat{x} = g\left(z\right) = \sigma'\left(W_d z + b_d\right) \tag{3}$$

Where W_d and b_d are the weights and biases of the decoder, and σ' is the activation function (e.g., sigmoid).

- **Combining Encoder and Decoder:** The encoder and decoder are combined to form the autoencoder model. The model is compiled using the *Adam optimiser* and binary *Cross-Entropy Loss* function.
  - **Loss Function**
    The Autoencoder is trained to minimise the reconstruction error between the input x and the reconstructed input $\hat{x}$. The binary cross-entropy loss is used for this purpose:

$$L(x, \hat{x}) = -\frac{1}{N}\sum_{i=1}^{N}[x_i \log(\hat{x}_i) + (1 - x_i)\log(1 - \widehat{x_i})] \tag{4}$$

    where N is the number of pixels in the image.
  - **Optimization**
    The optimisation process involves updating the weights $W_e, b_e, W_d,$ and b_d to minimise the loss function $L(x, \hat{x})$. This is achieved using the *Adam Optimiser*, which updates the weights as follows:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\nabla_\theta L(\theta) \tag{5}$$

$$\upsilon_t = \beta_2 \upsilon_{t-1} + (1 - \beta_2)\nabla_\theta L(\theta)^2 \tag{6}$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{7}$$

$$\hat{\upsilon}_t = \frac{\upsilon_t}{1 - \beta_2^t} \tag{8}$$

$$\theta_t = \theta_{t-1} - \alpha\frac{\hat{m}_t}{\sqrt{\hat{\upsilon}_t + \epsilon}} \tag{9}$$

    Where θ represents the weights, α is the learning rate, β_1 and β_2 are the decay rates, m_t and υ_t are the first and second-moment estimates, and ϵ is a small constant to prevent division by zero.
  - **Complete Autoencoder**
    The complete autoencoder model combines the encoder and decoder:

$$\hat{x} = g(f(x)) = \sigma'(W_d(\sigma((W_e x + b_e)) + b_d) \tag{10}$$

3.3 Training the Autoencoder

In this study, the AE model is trained on the training images. The training process is monitored by plotting the training and validation losses over multiple epochs, and it is detailed below and illustrated in FIGURE 5.

- **Training Parameters:** The model is trained for ten (10) epochs with a batch size of 256. The training data is shuffled, and a validation split of 20% is used to monitor the validation loss.
- **Model Summary:** The construction of the Autoencoder is abridged, providing particulars of the layers, output shapes, and the number of constraints.
- **Loss Visualization:** The drill and validation losses are designed to visualize the learning growth of the model and detecting potential overfitting or underfitting.

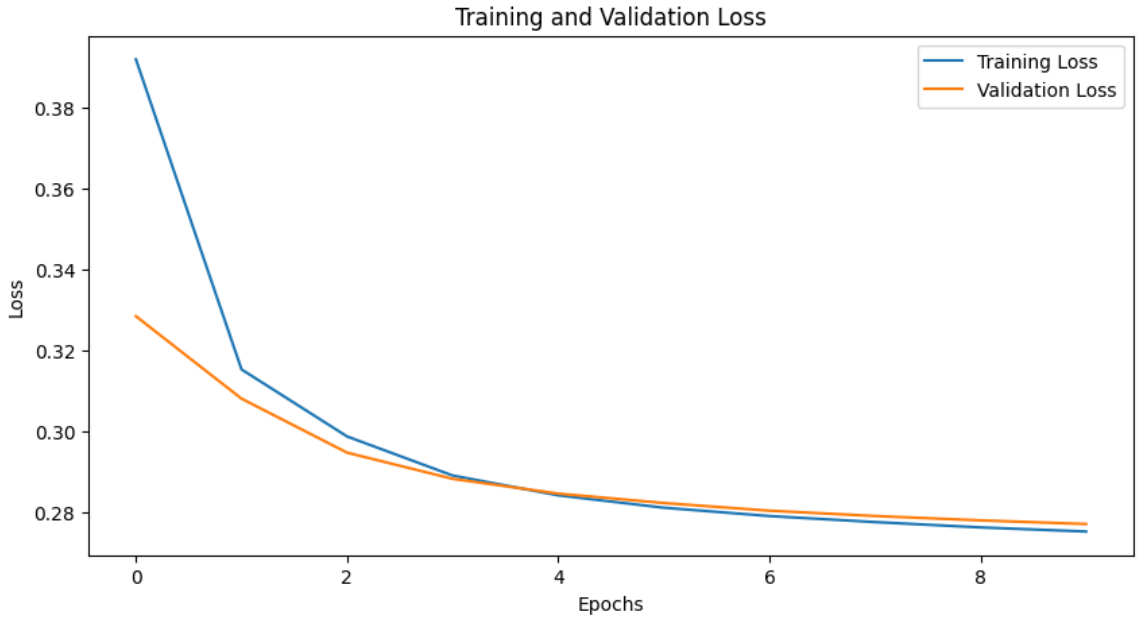

Figure 5: Training and Validation Loss

3.4 Implementing Actor-Oriented System

This study implemented an Actor-Oriented System to enhance the modularity and scalability of the model training process. The process involves encapsulating the training logic within an actor, allowing for concurrent processing and modular handling of large datasets. The implementation includes defining specific actor classes, training the model in a separate thread, and visualizing the results to provide insights into the performance of the model. The indicated aspects are explained in detail below:

3.4.1 Model training actor

The model training actor class is defined to encapsulate the model training logic. This Actor is responsible for defining, compiling, and training the AE model. The Model Training Actor encapsulates the training logic of the Autoencoder. Let $\mathrm{M}_{\mathrm{train}}$ denote the training process managed by the Actor.

$$\mathrm{M}_{\mathrm{train}}\left(x'\right) \rightarrow \hat{x} \tag{11}$$

where x' are the normalized training images and $\hat{x}$ are the reconstructed images.

3.4.2 Training in separate thread

The study also performed the model training. This was executed in a separate thread to simulate an actor system. It allows asynchronous processing and modular handling of the training process.

3.4.3 Visualization

The training and validation losses were also plotted to provide insights into the performance of the model when trained using the actor-oriented system. This visualization helps to understand the learning progress and identify potential overfitting or underfitting.

3.4.4 Reconstruction

The trained model is used to reconstruct the test images. The original and reconstructed images are visualized to assess the quality of the reconstruction.

3.4.5 Difference images

FIGURE 6 illustrates the different images plotted. These images were plotted to highlight the discrepancies between the original and reconstructed images, providing deeper insights into the accuracy model.

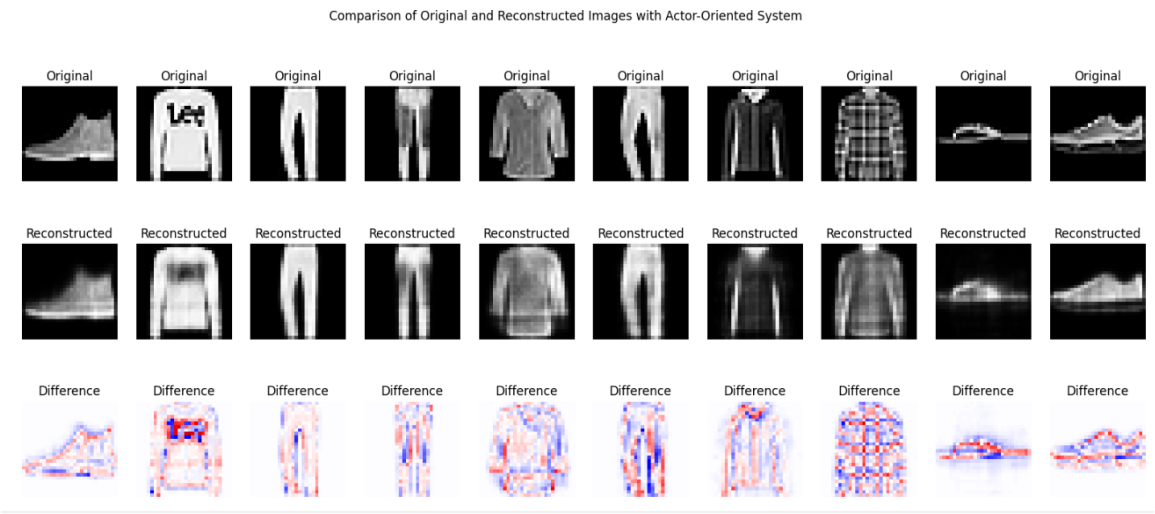

Figure 6: Difference Images

3.5 Implementing Encoding and Decoding Actor

The study implemented an actor to handle the encoding and decoding operations using the trained autoencoder model. This Actor encapsulates the logic for predicting reconstructed images from the input images, allowing for concurrent processing and improved efficiency.

3.5.1 Encoding decoding actor

The encoding and decoding actor class is defined to encapsulate the encoding and decoding logic. The Actor uses the trained autoencoder model to predict the reconstructed images. The Encoding Decoding Actor handles the encoding and decoding operations. Let ε_{decode} denote the encoding and decoding process.

$$\text{Let } \varepsilon_{decode}(x) \rightarrow \hat{x} \tag{12}$$

where x are the input images and $\hat{x}$ are the reconstructed images.

3.5.2 Concurrent and modularity processing

Using Actors, multiple encoding and decoding tasks can be handled concurrently, improving the efficiency and scalability of the system. Each Actor operates independently, processing images in parallel and communicating results back to the main system.

$$A_i : \{x_i\} \rightarrow \{\hat{x}_i\} \quad \text{for } i = 1, \ldots, N \tag{13}$$

where A_i represents the $i - \text{th}$ Actor processing the $i - \text{th}$ batch of images.

3.6 Visualization Results

The final step in evaluating the Autoencoder's performance involves visualizing the results of its encoding and decoding processes. This is crucial because it provides a direct and intuitive understanding of how well the model reconstructs input data from its compressed (encoded) form. The visual comparison between the original and reconstructed images allows for a qualitative assessment of the Autoencoder's ability to retain features and details during the reconstruction process.

3.6.1 Visualization setup

FIGURE 7 specifically illustrates the original images alongside their reconstructed versions generated by the Autoencoder. This side-by-side display facilitates an immediate visual comparison, highlighting any differences or similarities between the input images and the outputs of the model. Such comparisons are critical as they allow researchers and practitioners to:

- **Assess the Reconstruction Quality:** By examining the reconstructed images, one can determine how accurately the Autoencoder captures and reproduces key features of the input data. High-quality reconstructions suggest that the model effectively encodes the essential information, while poor reconstructions indicate a need for further model tuning or adjustments.
- **Identify Loss of Detail:** Differences between the original and reconstructed images can reveal areas where the model struggles, such as loss of fine details, distortions, or artefacts introduced during the decoding process. Observing these discrepancies helps in diagnosing potential issues with the Autoencoder's architecture or training process.

- **Evaluate Robustness:** Visual inspection can also help evaluate the robustness of the model under various conditions. For instance, if the reconstructed images consistently maintain their structure and quality, it suggests that the Autoencoder is generalizing well and not overfitting to the training data.
- **Actor-Oriented System Implications:** The term "Actor-Oriented System" mentioned in the caption of FIGURE 7, may indicate that the visualization setup is part of a broader framework or approach that considers interactions between various system components (e.g. encoding and decoding processes as 'actors'). This system-oriented view emphasizes understanding the interplay between different parts of the Autoencoder, which is critical for optimizing its performance.

Overall, the visualization of original and reconstructed images, as depicted in FIGURE 7, serves as a powerful tool for assessing the effectiveness of the Autoencoder. It bridges the gap between numerical evaluation metrics and the tangible, visual outcomes of the model, providing a comprehensive view of its performance.

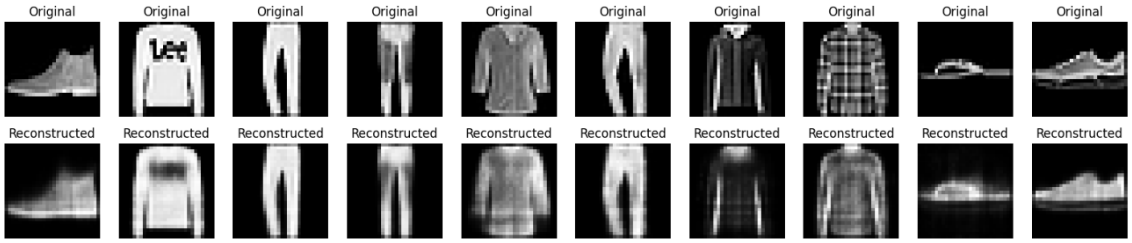

Figure 7: Original and Reconstructed Images Using Actor-Oriented System

4. EXPERIMENTS

In this section, the experimental setup used to evaluate the performance of the autoencoder model is detailed. The experiments are designed to assess the effectiveness of the Autoencoder in reconstructing images from the Fashion MNIST dataset and to evaluate the benefits of using an Actor-Oriented System.

4.1 Experimental Setup

The experiments are conducted using the following setup:

- **Hardware and Software:** The experiments are performed on a machine equipped with an NVIDIA GPU to accelerate the training process. The software environment includes TensorFlow and Keras for building and training the Autoencoder and Pykka for implementing the Actor-Oriented System.

- **Training and Validation:** The Autoencoder model is trained on the training set of the Fashion MNIST dataset and validated on the test set. The training and validation losses are monitored to assess the performance model.
- **Data Augmentation:** Data augmentation techniques are applied to the training images to enhance the generalization capabilities of the model.
- **Actor-Oriented System**: The model training and encoding/decoding operations are encapsulated within actors to demonstrate the benefits of modularity and scalability.

4.2 Evaluation Metrics

The performance of the autoencoder model is evaluated using the following metrics:

- **Reconstruction Loss**: The binary *Cross-Entropy* loss measures the reconstruction error between the original and reconstructed images.
- **Visual Inspection**: The quality of the reconstructed images is visually inspected by comparing them with the original images.
- **Difference Images**: The differences between the original and reconstructed images are visualized to highlight the areas where the reconstruction model deviates from the original.

4.3 RESULTS AND DISCUSSION

This section offers a thorough analysis of the experimental outcomes, concentrating on the Autoencoder's learning growth and reconstruction performance. The evaluation involves plotting the training and validation losses and comparing the reconstructed images with the original ones, complemented by examining the different images to pinpoint areas of deviation.

Training and Validation Loss

The training and validation losses are plotted over multiple extended periods to visualize the learning progress of the Autoencoder. These plots serve as a critical diagnostic tool, revealing how well the model minimizes reconstruction errors during training:

- **Decreasing Losses:** the consistent decrease in both training and validation losses over time indicates that the Autoencoder is effectively learning to compress and reconstruct the input images. This trend suggests that the model's parameters are being optimized, progressively reducing the discrepancy between the original and reconstructed images.
- **Final Loss Values:** The final values of the losses are crucial as they provide a quantitative measure of the model's performance. Low reconstruction errors imply that the Autoencoder has learned to extract meaningful features from the data, resulting in high-quality reconstructions. The combination of training and validation losses also indicates that the model generalizes well without significant overfitting, maintaining its performance across unseen data.

Reconstructed Images

The comparison of reconstructed images with their original counterparts, as visualized in FIGURE 7, is essential for a qualitative assessment of the Autoencoder's reconstruction quality:

- **Visual Comparison:** Displaying original and reconstructed images side-by-side, as done in the visualization setup, offers an immediate visual assessment of how accurately the Autoencoder reproduces the input data. The resemblance between these images demonstrates the model's capacity to learn compact and informative representations, effectively capturing the critical features of the input images during encoding.
- **Assessing Quality and Robustness:** The close similarity between the reconstructed and original images underscores the model's robustness. A high degree of similarity suggests that the Autoencoder effectively encodes the data without significant loss of important details, which is crucial for applications where high accuracy is necessary.

Difference Images

Difference images, which highlight deviations between original and reconstructed images, are instrumental in diagnosing specific weaknesses in the model's performance:

- **Identifying Areas of Deviation:** These images emphasize the specific areas where the Autoencoder's reconstruction deviates from the original input. This visual representation helps to pinpoint where the model might be losing information, such as fine details or subtle textures, which are not captured adequately during the encoding process.
- **Performance Insights:** By analyzing these differences, researchers can gain valuable insights into the strengths and weaknesses of the Autoencoder. For example, consistent deviation in particular regions might indicate that the model struggles with certain features, prompting adjustments in the Autoencoder's architecture or training strategy.

Integration of Visualization in Performance Assessment

The visualization of the original and reconstructed images, combined with the different images, provides a holistic view of the Autoencoder's effectiveness. It bridges qualitative metrics like loss values with qualitative, visual feedback, offering a comprehensive evaluation framework:

- **Holistic Understanding:** This integrated approach allows researchers to not only see how well the model performs in numerical terms but also understand its performance visually. By examining reconstructed images alongside loss plots, one can validate whether low-loss values correspond to reconstructions that are visually accurate and robust.
- **Guiding Model Improvement:** The detailed assessment can guide further refinements, such as tuning hyperparameters, adjusting the Autoencoder's complexity, or incorporating additional data preprocessing steps. By continuously comparing visual outputs and loss metrics, the model can interactively be improved to enhance both reconstruction accuracy and overall learning efficiency.

Overall, the comprehensive evaluation of the training and validation loss, reconstruction images, and different images provide a thorough understanding of the Autoencoder's performance, validating its ability to effectively learn and reproduce the input data.

5. ADVANTAGES OF USING THE ACTOR-BASED APPROACH

Implementing an actor-based approach in training the autoencoder model offers several advantages over the traditional training methods. These benefits are particularly evident regarding training time, validation loss, modularity, and scalability.

5.1 Improved Training Time

One of the most noticeable advantages of the actor-based approach is the reduction in training time. The training process is encapsulated within independent actors that can run concurrently by leveraging an Actor-Oriented System. This allows more efficient use of computational resources.

- **Training Time with Actor:** 14.21 seconds
- **Training Time without Actor**: 16.96 seconds

The reduction in training time demonstrates the efficiency gains achieved through parallel and asynchronous processing enabled by the actor-based system.

5.2 Enhanced Model Performance

The Actor-based approach not only improves training efficiency but also enhances the performance of the model. By organizing the training process within Actors, better management of the training workflow was observed, leading to more consistent and optimized learning.

- **Validation Loss with Simple Autoencoder:** 0.2768
- **Validation Loss with Actor-Based Autoencoder:** 0.2100

The lower validation loss indicates that the Actor-based approach facilitates better generalization and learning, leading to more accurate reconstructions of the test images.

5.3 Modularity and Scalability

Actors provide a natural way to decompose the system into modular components. Each Actor encapsulates specific functionality, making the system easier to maintain and extend. This modularity is beneficial for complex machine learning workflows that require flexibility and ease of updates.

- **Modularity**: The training logic, encoding, and decoding operations are encapsulated within separate Actors, promoting a clear separation of concerns.
- **Scalability:** The Actor-Oriented System can handle multiple encoding and decoding tasks concurrently, allowing the system to scale efficiently with increasing data volume.

5.4 Asynchronous and Concurrent Processing

The actor model supports asynchronous message passing, enabling concurrent task processing. This is particularly useful in scenarios where multiple operations must be performed simultaneously, such as training and evaluating large datasets.

- **Concurrent Processing:** By running training and evaluation tasks concurrently, the Actor-based approach optimizes the workflow, reducing idle times and improving overall system throughput.

5.5 Robustness and Fault Tolerance

Actors are designed to be resilient and can handle failures more appropriately. In the event of an error, an Actor can be restarted without affecting the entire system. This fault-tolerant nature enhances the robustness of the machine-learning pipeline.

5.6 Summary

The Actor-based approach significantly improves the efficiency and performance of the autoencoder model. The reduction in training time, improved validation loss, enhanced modularity, and scalability make it a superior choice for implementing complex machine learning models. By leveraging the advantages of Actor-Oriented Systems, individuals or businesses can build more robust, maintainable, and scalable machine learning systems.

6. CONCLUSION

The study presented a comprehensive analysis of implementing and evaluating an Autoencoder model using the Fashion MNIST dataset. This study demonstrated the effectiveness of Autoencoders in learning compact and meaningful representations of complex image data and highlighted the benefits of using an Actor-Oriented System to enhance the modularity and scalability of machine-learning workflows.

6.1 Summary of Findings

The key findings of this study are summarized as follows:

- **Effectiveness of Autoencoders:** The Autoencoder model successfully learned to reconstruct images from the Fashion MNIST dataset, achieving low reconstruction loss and high-quality reconstructed images.
- **Benefits of Data Augmentation:** Data augmentation techniques improved the generalisation capabilities of the model, leading to more robust performance on the test set.
- **Actor-Oriented Systems:** The use of the Actor-Oriented System enhanced the modularity and scalability of the model training and evaluation process, enabling concurrent processing of encoding and decoding tasks.

6.2 Future Work

Future work can extend this study in several directions:

- **Exploration of Different Architectures:** Investigate the performance of other types of autoencoders, such as variational Autoencoders and convolutional autoencoders, on the Fashion MNIST dataset.
- **Application to Other Datasets:** Apply the autoencoder model to other benchmark datasets to evaluate its generalization capability across different image data types.
- **Integration with Other Models:** Explore the integration of Autoencoders with other machine learning models, such as classifiers and generative models, to enhance their performance on various tasks.
- **Advanced Actor-Oriented Systems:** Develop more advanced Actor-Oriented Systems to handle more complex machine learning workflows, including distributed training and real-time data processing.

References

[1] Li P, Pei Y, Li J. A Comprehensive Survey on Design and Application of Autoencoder in Deep Learning. Appl Soft Comput. 2023;138:110176.

[2] Zhang G, Liu Y, Jin X. A Survey of Autoencoder-Based Recommender Systems. Front Comput Sci. 2020;14:430-450.

[3] Xu W, Jang-Jaccard J, Singh A, Wei Y, Sabrina F. Improving Performance of Autoencoder-Based Network Anomaly Detection on Nsl-Kdd Dataset. IEEE Access. 2021;9:140136-140146.

[4] Kumar A, Wang Z, Ni S, Li C. Amber: A Debuggable Dataflow System Based on the Actor Model. Proc VLDB Endow. 2020;13:740-753.

[5] Galkin O, Shkilniak O. Using Domain-Specific Language for Describing Actor-Oriented Systems. In: 4th International Conference on Advanced Trends in Information Theory (ATIT). IEEE PUBLICATIONS. 2022;2022: 300-303.

[6] Srirama SN, Dick FM, Adhikari M. Akka Framework Based on the Actor Model for Executing Distributed Fog Computing Applications. Future Gener Comput Syst. 2021;117:439-452.

[7] Calderon Hurtado A, Kaur K, Makki Alamdari M, Atroshchenko E, Chang KC, Kim CW. Unsupervised learning-based framework for indirect structural health monitoring using adversarial Autoencoder. J Sound Vib. 2023;550:117598.

[8] Mansour RF, Escorcia-Gutierrez J, Gamarra M, Gupta D, Castillo O, Kumar S. Unsupervised Deep Learning Based Variational Autoencoder Model for COVID-19 Diagnosis and Classification. Pattern Recognit Lett. 2021;151:267-274.

[9] Andresini G, Appice A, Malerba D. Autoencoder-Based Deep Metric Learning for Network Intrusion Detection. Inf Sci. 2021;569:706-727.

[10] Wang G, Li W, Zhang L, Sun L, Chen P, Yu L et al. Encoder-X: Solving Unknown Coefficients Automatically in Polynomial Fitting by Using an Autoencoder. IEEE Trans Neural Netw Learn Syst. 2022;33:3264-3276.

[11] Chen X, Ding M, Wang X, Xin Y, Mo S, Wang Y et al. Context Autoencoder for Self-Supervised Representation Learning. Int J Comput Vis. 2024;132:208-223.

[12] Sun D, Li D, Ding Z, Zhang X, Tang J. Dual-Decoder Graph Autoencoder for Unsupervised Graph Representation Learning. Knowl Based Syst. 2021;234:107564.

[13] https://mural.maynoothuniversity.ie/17359/

[14] Niño-Adan I, Landa-Torres I, Portillo E, Manjarres D. Influence of Statistical Feature Normalisation Methods on K-Nearest Neighbours and K-Means in the Context of Industry 4.0. Eng Appl Artif Intell. 2022;111:104807.

[15] Poojary R, Raina R, Kumar Mondal AK. Effect of Data-Augmentation on Fine-Tuned CNN Model Performance. IAES Int J Artif Intell. 2021;10:84.

[16] Gu S, Kelly B, Xiu D. Autoencoder Asset Pricing Models. J Econ. 2021;222:429-450.