RESEARCH ARTICLE

Adaptive Manifold Concept with Regularized Autoencoders (AMRAE) for Effective Dimensionality Reduction

Seyed-Ali Sadegh-Zadeh[1], Nasrin Sadeghzadeh[2], Saeed Shiry Ghidary[1], Sobhan Movahedi[3], Kaveh Kavianpour[4], Mohammad Amin Barati[5] and Alireza Soleimani Mamalo[6,*]

[1] Department of Computing, School of Digital, Technologies and Arts, Staffordshire University, Stoke-on-Trent ST4 2DE, United Kingdom
[2] Department of Mathematics, Faculty of Basic Sciences, University of Qom, Qom 37161-46611, Iran
[3] Islamic Azad University, Science and Research, Tehran, Iran
[4] Department of Mathematics and Computer Science, Amirkabir University of Technology, Tehran 15875-4413, Iran
[5] Department of Mechanical Engineering, School of Mechanical Engineering, College of Engineering, University of Tehran, Tehran 14179-35840, Iran
[6] Student Research Committee, Urmia University of Medical Sciences, Urmia, Iran

Abstract

The Adaptive Manifold Concept with Regularized Autoencoders (AMRAE) algorithm is introduced as a novel dimensionality reduction technique that integrates manifold learning with autoencoders to capture the intrinsic geometry of high-dimensional data effectively. The study evaluates the impact of various adjustments and enhancements within the "Manifold Adjustment Box," including different regularization techniques, activation functions, and architectural choices, across diverse datasets. Key findings demonstrate that configurations such as Leaky ReLU Activation and Batch Norm Layer consistently improve accuracy, results highlight the flexibility and robustness of AMRAE. The results underscore the significance of AMRAE in addressing the limitations of traditional dimensionality reduction methods, with potential applications in various fields requiring high-dimensional data analysis. This paper provides a comprehensive summary of the objectives, methodology, results, and conclusions, showcasing AMRAE's efficacy in improving data representation and machine learning performance.

Keywords: dimensionality reduction, manifold concept regularized autoencoders, machine learning, high-dimensional data.

1 Introduction

Dimensionality reduction is a critical preprocessing step in many data analysis and machine learning tasks. It aims to reduce the number of random variables

Submitted: 31 March 2025
Accepted: 07 July 2025
Published: 02 March 2026

Vol. 1, **No.** 2, 2026.
10.62762/FBSP.2025.800185

***Corresponding author:**
✉ Alireza Soleimani Mamalo
Soleymanialireza688@gmail.com

Citation
Sadegh-Zadeh, S. A., Sadeghzadeh, N., Ghidary, S. S., Movahedi, S., Kavianpour, K., Barati, M. A., & Mamalo, A. S. (2026). Adaptive Manifold Concept with Regularized Autoencoders (AMRAE) for Effective Dimensionality Reduction. *Frontiers in Biomedical Signal Processing*, 1(2), 79–104.

under consideration, making high-dimensional data more manageable while preserving its intrinsic properties [1, 2]. This process is essential for various applications, including visualization, noise reduction, and enhancing the performance of machine learning algorithms. By reducing the dimensionality of data, we can mitigate the curse of dimensionality, which often leads to overfitting and high computational costs [3]. Dimensionality reduction techniques can be broadly categorized into linear and nonlinear methods. Principal Component Analysis (PCA) is a widely used linear method that transforms the data into a new coordinate system where the greatest variances lie on the first few coordinates. While PCA is effective for many problems, it often fails to capture the complex structures inherent in real-world data, especially when the relationships between variables are nonlinear [4–7].

To address these limitations, manifold learning techniques have been developed. These techniques, such as t-SNE, Isomap, and Locally Linear Embedding (LLE), aim to uncover the underlying manifold structure in the data, providing a more faithful representation of its intrinsic geometry [8]. Manifold learning assumes that high-dimensional data lies on a low-dimensional manifold embedded in the higher-dimensional space [9]. By preserving the local or global structure of the data, these methods can reveal meaningful patterns and relationships that linear techniques might miss [10].

Autoencoders, a type of artificial neural network, have emerged as powerful tools for nonlinear dimensionality reduction. An autoencoder consists of an encoder, which maps the input data to a lower-dimensional latent space, and a decoder, which reconstructs the original data from this latent representation [11, 12]. By training the autoencoder to minimize the reconstruction error, it learns an efficient encoding of the data that captures its essential features. Autoencoders can be enhanced with various regularization techniques, such as sparsity constraints and denoising, to improve their performance and robustness [13].

Our novel approach, Adaptive Manifold concept with Regularized Autoencoders (AMRAE), combines the strengths of manifold concept and autoencoders. By integrating geometric regularization derived from manifold methods into the autoencoder framework, AMRAE aims to capture the intrinsic geometry of the data more effectively. This hybrid approach leverages the power of deep learning while maintaining the structural integrity of the data, offering a promising direction for dimensionality reduction in complex, high-dimensional datasets. The primary objective of this research is to develop and validate the AMRAE algorithm, a novel dimensionality reduction technique that addresses the limitations of existing methods. The purpose of AMRAE is to combine the strengths of manifold concept and autoencoders to effectively capture the intrinsic geometry of high-dimensional data, while maintaining scalability, robustness to noise, and interpretability. By integrating geometric regularization derived from manifold concept into the autoencoder framework, AMRAE aims to preserve both local and global structures within the data, providing a more faithful and meaningful representation. This research seeks to demonstrate the efficacy of AMRAE across a variety of datasets, highlighting its potential to improve the performance of machine learning tasks and its applicability to real-world data analysis problems.

This paper makes several key contributions to the field of dimensionality reduction and manifold concept by introducing the AMRAE algorithm, which uniquely integrates manifold concept with regularized autoencoders to enhance the capture of intrinsic data geometry. We conduct extensive experiments on a diverse set of datasets, demonstrating AMRAE's robustness and effectiveness compared to traditional techniques such as PCA, t-SNE, LLE, MDS, and Isomap. Our analysis of the manifold adjustment box, including various regularization techniques and architectural choices, provides insights into their impact on AMRAE's performance. Additionally, we estimate the intrinsic dimensionality of datasets using the Fisher Information Matrix, discuss its implications, evaluate dataset complexity using silhouette scores, and detect anomalies with Isolation Forests, highlighting AMRAE's resilience to noise and outliers. Finally, we offer practical guidelines for selecting and tuning manifold adjustment box settings to achieve optimal performance, providing valuable insights and advice for researchers and practitioners working with high-dimensional data.

2 Related Work

Dimensionality reduction is a fundamental task in data analysis and machine learning, aimed at reducing the number of random variables under consideration while retaining the essential structure and patterns within the data. Over the years, numerous techniques

have been developed, ranging from classical linear methods to more complex nonlinear approaches. This section provides an overview of existing work in the field, starting with linear dimensionality reduction techniques.

2.1 Linear Dimensionality Reduction Techniques

Linear dimensionality reduction techniques aim to transform high-dimensional data into a lower-dimensional space using linear projections. These methods are often preferred for their simplicity, computational efficiency, and interpretability. Two of the most widely used linear techniques are PCA and Linear Discriminant Analysis (LDA) [14, 15].

PCA transforms the data into a new coordinate system such that the greatest variances by any projection of the data come to lie on the first coordinates (called principal components), the second greatest variances on the second coordinates, and so on. PCA is simple to implement and computationally efficient, preserving the directions of highest data variance, which aids in noise reduction by focusing on the largest variances. However, it has limitations, such as an inability to capture nonlinear relationships and sensitivity to scaling and outliers [16, 17].

LDA is another popular linear dimensionality reduction technique, particularly used for classification tasks. Unlike PCA, which is unsupervised and focuses on maximizing variance, LDA is supervised and aims to maximize the separability between different classes. It does so by finding a linear combination of features that best separates the classes. Computing the within-class and between-class scatter matrices measures the dispersion of data points within each class and between different classes. LDA performs an eigenvalue decomposition of these matrices to find directions (linear discriminants) that maximize the ratio of between-class variance to within-class variance, making it useful when classes are well-separated in the original feature space. This method provides a clear criterion for dimensionality reduction based on class separability, though it assumes that data follows a Gaussian distribution with equal covariance matrices for all classes, which might not always be true in practice [18–20].

Both PCA and LDA have been foundational techniques in the field of dimensionality reduction, offering valuable insights and providing a basis for more advanced methods. While they are powerful tools, their linear nature can be a limitation in many real-world scenarios where the data lies on a nonlinear manifold [21]. This limitation has driven the development of nonlinear dimensionality reduction techniques, such as manifold concept and autoencoders, which we explore further in subsequent sections.

2.2 Nonlinear Dimensionality Reduction Techniques

While linear dimensionality reduction techniques like PCA and LDA are effective in many scenarios, they often fall short when dealing with data that lies on a complex, nonlinear manifold. Nonlinear dimensionality reduction techniques aim to address these limitations by preserving the intrinsic structure and geometry of the data. This section provides an overview of several prominent nonlinear dimensionality reduction techniques, including t-SNE, LLE, and other manifold learning methods.

t-Distributed Stochastic Neighbour Embedding (t-SNE) is a widely used nonlinear dimensionality reduction technique, particularly popular for visualizing high-dimensional data. Introduced by Laurens van der Maaten and Geoffrey Hinton in 2008 [22], t-SNE aims to map high-dimensional data points into a lower-dimensional space in a way that preserves the local structure of the data [23, 24]. The key steps in t-SNE involve computing pairwise similarities between data points in the high-dimensional space using a Gaussian kernel, then mapping these similarities to a low-dimensional space with a Student-t distribution to address the "crowding problem" [25]. This is followed by optimizing the Kullback-Leibler divergence between the high- and low-dimensional similarity distributions using gradient descent. While t-SNE excels at visualizing data clusters, it has limitations including high computational complexity, difficulty with large datasets, and sensitivity to hyperparameters like perplexity [26, 27].

LLE is another powerful nonlinear dimensionality reduction technique introduced by Sam Roweis and Lawrence Saul in 2000 [28]. LLE focuses on preserving the local geometry of data by linearizing small neighbourhoods and then mapping these local neighbourhoods into a lower-dimensional space. The main steps in LLE involve identifying the nearest neighbours for each data point and expressing each point as a weighted linear combination of its neighbours to construct a weight matrix that captures these local linear relationships. Then, a

low-dimensional representation is found by solving an eigenvalue problem to best preserve these relationships. While LLE is effective at unfolding nonlinear manifolds and is computationally efficient, it can struggle with noise and requires careful selection of the number of neighbours [29–31].

Isomap, introduced by Joshua Tenenbaum, Vin de Silva, and John Langford in 2000 [32], extends the principles of classical multidimensional scaling (MDS) to nonlinear manifolds. Isomap aims to preserve the global geometric structure of the data by incorporating geodesic distances instead of Euclidean distances [33]. The main steps in Isomap involve constructing a neighbourhood graph where edges represent distances between neighbouring points, computing the shortest paths between all pairs of points to approximate geodesic distances, and applying classical MDS to the geodesic distance matrix to obtain the low-dimensional embedding. Isomap effectively captures global structures and is robust to noise, but it can be computationally intensive for large datasets and is sensitive to the choice of neighbourhood size [34, 35].

UMAP (Uniform Manifold Approximation and Projection), developed by Leland McInnes and John Healy in 2018, is a newer nonlinear dimensionality reduction technique that aims to provide high-quality embeddings with better preservation of global structure compared to t-SNE [36]. UMAP is based on manifold learning assumptions and leverages algebraic topology to construct a fuzzy topological representation of the high-dimensional data. The main steps in UMAP involve constructing a high-dimensional fuzzy simplicial complex to represent the data manifold, and then optimizing the low-dimensional embedding by minimizing the cross-entropy between the high- and low-dimensional fuzzy simplicial complexes. UMAP is known for its computational efficiency, scalability, and ability to effectively preserve both local and global structures in the data [37–39].

Nonlinear dimensionality reduction techniques provide powerful tools for capturing the intricate structures in high-dimensional data. While methods like t-SNE, LLE, Isomap, and UMAP have their strengths and weaknesses, combining these approaches with the flexibility of autoencoders offers a promising direction for advanced dimensionality reduction methods. Our proposed AMRAE builds upon these nonlinear techniques to address their limitations and enhance their capabilities.

2.3 Autoencoders and Manifold Concept

Dimensionality reduction is essential for processing high-dimensional data, uncovering intrinsic structures, and enhancing the performance of machine learning models. Among the various methods developed, autoencoders and manifold concept have emerged as powerful tools for nonlinear dimensionality reduction. This section provides a background on these methods and their significance.

Autoencoders are a type of artificial neural network designed for unsupervised learning. They are particularly effective at learning efficient coding of high-dimensional data by compressing it into a lower-dimensional latent space and then reconstructing it. The primary components of an autoencoder are the encoder, the latent space, and the decoder [40].

- **Encoder:** The encoder is a neural network that transforms the input data into a lower-dimensional representation, or latent space. It learns a function f that maps the input x to the latent space z, where $z = f(x)$.
- **Latent Space:** The latent space is a compressed representation of the input data. It captures the most salient features of the data in a reduced dimensionality.
- **Decoder:** The decoder is another neural network that reconstructs the input data from the latent space. It learns a function g that maps the latent space z back to the original space $\hat{x}$, where $\hat{x} = g(x)$.

The training objective of an autoencoder is to minimize the reconstruction error, which measures the difference between the input and the reconstructed output . This is typically achieved through backpropagation and gradient descent. Common loss functions used in autoencoders include MSE for continuous data and binary cross-entropy for binary data [41, 42]. Autoencoders can be enhanced with various regularization techniques to improve performance and robustness: Sparse Autoencoders introduce sparsity constraints on the latent space to encourage learning more meaningful features; Denoising Autoencoders train the model to reconstruct original data from corrupted versions, enhancing noise robustness; and Variational Autoencoders (VAEs) use a probabilistic approach to model the latent space, incorporating a Kullback-Leibler divergence term for smoothness and disentanglement [43, 44].

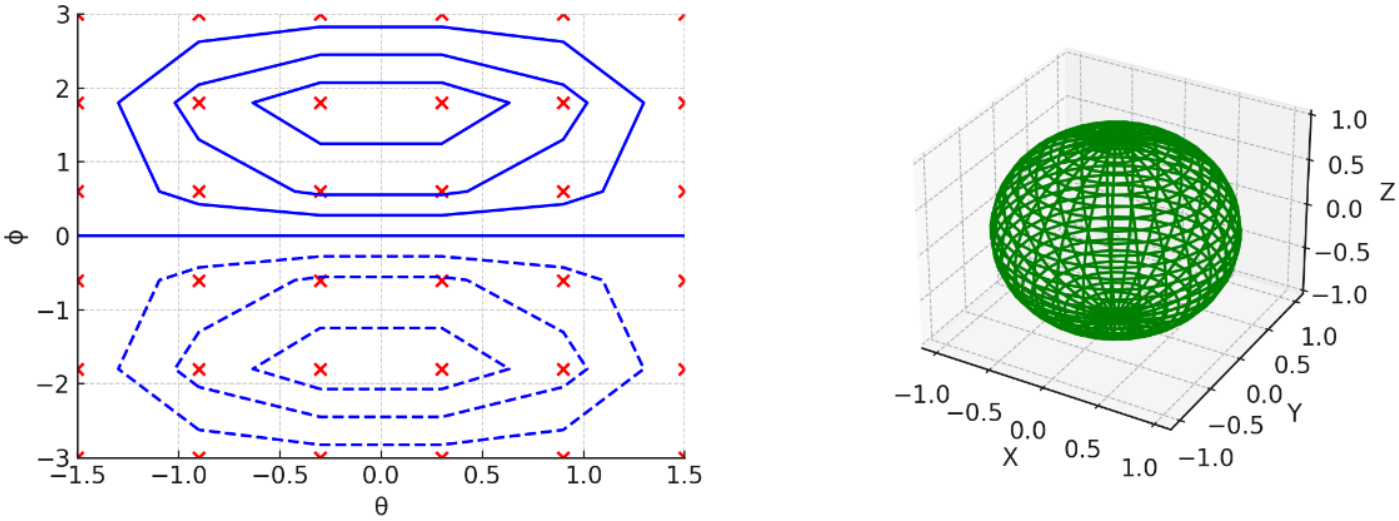

Figure 1. Visualization of Manifolds: The left plot illustrates a 2D grid on a plane, representing a simple 2-dimensional manifold with local Euclidean properties. The right plot depicts a 2-dimensional spherical surface embedded in 3-dimensional space, demonstrating the concept of a curved manifold that locally appears flat.

In the context of data science and machine learning, a manifold is a fundamental concept that refers to a topological space that locally resembles Euclidean space. Essentially, a manifold can be thought of as a mathematical space that, despite possibly having a complex global structure, looks like ordinary Euclidean space at a small enough scale. This property makes manifolds particularly useful for modelling and understanding high-dimensional data [45, 46].

Manifold concept is based on the assumption that high-dimensional data often lie on or near a low-dimensional manifold within the higher-dimensional space. This means that although the data might have many features, the intrinsic dimensionality, or the number of variables needed to describe the underlying structure of the data, is much lower. For example, while images of handwritten digits in the MNIST dataset are represented in a 784-dimensional space (28x28 pixels), the actual variations in the digits can be captured by far fewer dimensions.

In the Figure 1, the left plot displays a 2D grid of ellipses and points on a plane, representing a 2-dimensional manifold. The grid points indicate specific locations on the manifold, each identifiable by a pair of coordinates (θ, ϕ). The contours, shown as ellipses, represent levels of a function defined on this manifold, such as $cos(\theta) \cdot sin(\phi)$. This plot illustrates the local flatness of the manifold, where small regions behave like a flat Euclidean plane, highlighting a fundamental property of manifolds.

In contrast, the right plot depicts a 2D surface embedded in a 3D space, commonly known as a 2-sphere. The spherical grid illustrates how points on the 2D surface of the sphere are mapped, each identifiable by spherical coordinates (μ, ν). This manifold is curved, but small regions appear flat when zoomed in, like the grid in the left plot. This plot demonstrates that manifolds can exist in higher-dimensional spaces while retaining local Euclidean properties, emphasizing that small regions of a manifold will always resemble Euclidean space regardless of the overall shape or dimensionality.

Several key characteristics of manifolds include:

- Local Linearity: At a sufficiently small scale, the manifold is approximately linear and resembles a flat Euclidean space.
- Global Nonlinearity: While the manifold appears linear locally, its global structure can be highly nonlinear.
- Dimensionality: The intrinsic dimensionality of a manifold is the minimum number of coordinates needed to describe the manifold's structure.

Manifold concept techniques aim to discover these low-dimensional structures within high-dimensional data. Techniques such as t-SNE, Isomap, and LLE attempt to map the high-dimensional data to a lower-dimensional space in a way that preserves the geometric and topological properties of the original data. This process helps to uncover meaningful patterns and relationships that are not apparent in the higher-dimensional space [47, 48].

By integrating manifold concept with autoencoders, the AMRAE algorithm leverages the strengths of both approaches. Autoencoders provide a flexible framework for learning nonlinear representations, while manifold concept ensures that the intrinsic geometric properties of the data are preserved, leading to more accurate and interpretable low-dimensional representations. The integration of autoencoders and manifold concept aims to leverage the strengths of both approaches for more effective dimensionality reduction. Autoencoders provide a flexible and powerful framework for learning nonlinear representations, while manifold concept ensure that the intrinsic geometric properties of the data are preserved. This hybrid approach can address the limitations of individual methods, such as the inability of linear techniques to capture complex structures and the scalability issues of traditional manifold concept [49–51].

Our proposed AMRAE algorithm combines these two approaches by incorporating geometric regularization derived from manifold concept into the autoencoder framework. This integration allows AMRAE to capture the intrinsic geometry of the data more effectively, providing a robust and scalable solution for nonlinear dimensionality reduction. Through this research, we aim to demonstrate the efficacy of AMRAE in preserving both local and global structures, enhancing interpretability, and improving the performance of subsequent machine learning tasks.

2.4 Limitations in Existing Techniques

Existing dimensionality reduction techniques have several limitations. Linear methods like PCA and LDA are restricted to capturing linear relationships, missing complex, nonlinear structures often present in real-world datasets. Nonlinear techniques such as t-SNE, LLE, Isomap, and UMAP, while effective at capturing nonlinear relationships, face challenges with scalability, noise sensitivity, and balancing the preservation of local and global structures. Scalability is a major issue as many nonlinear methods have high computational costs and memory requirements, making them impractical for large datasets. Additionally, techniques like LLE and Isomap are sensitive to noise and outliers, which can distort low-dimensional representations. Methods like t-SNE excel at preserving local structures but can distort global structures, whereas those focusing on global structures might neglect important local relationships. The interpretability of latent representations produced by many techniques, especially neural network-based ones, is often challenging, limiting their utility in understanding the underlying data. Moreover, the effectiveness of these methods depends on careful tuning of hyperparameters, requiring extensive experimentation and domain expertise. Finally, some non-parametric methods struggle to generalize to new data points, limiting their applicability in dynamic environments with continuous data influx [52, 53].

2.5 Addressing Limitations with AMRAE

The AMRAE algorithm addresses several limitations through key innovations: it integrates nonlinear structures by leveraging autoencoders to capture complex relationships within the data, ensuring accurate representation of intrinsic data geometry. Combining manifold concept with regularized autoencoders, AMRAE maintains scalability and efficiency, making it suitable for large-scale datasets. Regularization techniques like L1, L2, and elastic net enhance robustness to noise and outliers, providing stable dimensionality reduction. Geometric regularization balances the preservation of both local and global structures, ensuring comprehensive data representation. Autoencoders in AMRAE offer more interpretable latent representations, aiding understanding of underlying data patterns and decision-making. Its design includes manifold adjustment box settings tailored to different datasets, reducing hyperparameter sensitivity and simplifying tuning. The parametric nature of autoencoders enables AMRAE to generalize well to new data points, making it versatile for dynamic data environments. Thus, AMRAE provides a robust, scalable, and interpretable solution for nonlinear dimensionality reduction, enhancing the ability to uncover meaningful patterns and structures in high-dimensional data.

3 The AMRAE Algorithm

Dimensionality reduction is crucial in the preprocessing pipeline of many machine learning and data analysis tasks. The AMRAE algorithm is designed to address the limitations of existing dimensionality reduction techniques by combining the strengths of manifold concept and deep learning. This section elaborates on the concept, objectives, and key features of AMRAE.

3.1 Concept of AMRAE

AMRAE is an innovative approach that integrates manifold concept with regularized autoencoders to

create a robust and effective dimensionality reduction algorithm. The core idea is to leverage the intrinsic advantages of both manifold concept and deep learning:

- **Manifold Concept:** This component focuses on preserving the local and global geometric structure of high-dimensional data. Techniques like LLE, Isomap, and t-SNE are examples of manifold concept that aim to maintain the manifold structure in the reduced dimensional space.
- **Deep Learning:** Autoencoders, a type of artificial neural network, are used to learn efficient representations of data by minimizing the reconstruction error. Autoencoders can capture complex, nonlinear relationships within the data, which linear methods like PCA cannot achieve.

By combining these approaches, AMRAE aims to provide a comprehensive solution that addresses the limitations of traditional methods, such as capturing nonlinear structures, scalability, noise sensitivity, and balancing local and global structures.

3.1.1 Objective and Key Features

The primary objective of AMRAE is to achieve effective and scalable dimensionality reduction while preserving the intrinsic geometric properties of high-dimensional data. The key features of AMRAE include:

1. **Adaptive Integration:** AMRAE integrates manifold concept principles into the autoencoder framework, ensuring that the learned latent space accurately reflects the data's underlying manifold structure. This integration allows the algorithm to adaptively preserve both local and global relationships in the data.
2. **Regularization Techniques:** To enhance robustness and generalization, AMRAE incorporates various regularization techniques. These include L1 regularization, L2 regularization, and elastic net regularization, which help prevent overfitting and improve the stability of the learned representations.
3. **Flexible Manifold Adjustment Box:** AMRAE includes a manifold adjustment box, a flexible component that allows for different configurations and adjustments. This feature enables the algorithm to be fine-tuned for specific datasets and tasks, improving its adaptability and performance. Options within the manifold adjustment box include different activation functions (e.g., ReLU, tanh, leaky ReLU), dropout layers, batch normalization, and multiple hidden layers.
4. **Scalability and Efficiency:** By leveraging the computational efficiency of autoencoders and the intrinsic structure-preserving capabilities of manifold concept, AMRAE is designed to be scalable and efficient. It can handle large datasets and high-dimensional data, making it suitable for a wide range of applications.
5. **Enhanced Interpretability:** The latent representations produced by AMRAE are designed to be interpretable, aiding in understanding the underlying patterns and structures in the data. This interpretability is crucial for applications where insights from the data are as important as the predictive performance.
6. **Robustness to Noise:** Through the use of regularization and robust training techniques, AMRAE is resilient to noise and outliers, ensuring that the reduced representations are reliable and meaningful even in the presence of imperfect data.

AMRAE represents a significant advancement in the field of dimensionality reduction by combining the strengths of manifold concept and deep learning. Its adaptive, robust, and scalable nature makes it a powerful tool for uncovering the intrinsic structures of high-dimensional data, ultimately enhancing the performance of subsequent machine learning tasks and providing valuable insights into complex datasets.

3.2 Components of AMRAE

The AMRAE algorithm comprises several key components that work together to achieve effective dimensionality reduction. This section details the main components of AMRAE, including autoencoders, regularization techniques, and the integration of manifold concept principles.

3.2.1 Autoencoders

Autoencoders are a type of artificial neural network used for unsupervised learning. They are designed to learn a compressed representation of the input data by minimizing the reconstruction error between the input and the output. The structure of an autoencoder consists of three main parts:

- **Encoder:** The encoder maps the input data to a lower-dimensional latent space . It consists of several layers of neurons that progressively reduce the dimensionality of the input data. Mathematically, the encoder can be represented as $z = f(x)$, where f is the function learned by the encoder.
- **Latent Space:** The latent space is the compressed representation of the input data. It captures the essential features and patterns of the data in a reduced dimensionality. The dimensionality of the latent space is a key hyperparameter in the design of the autoencoder.
- **Decoder:** The decoder maps the latent representation z back to the original input space to reconstruct the data. It consists of several layers of neurons that progressively increase the dimensionality of the latent representation. The decoder can be represented as $\hat{x} = g(z)$, where $\hat{x}$ is the reconstructed data and g is the function learned by the decoder.

The training objective of an autoencoder is to minimize the reconstruction error, typically measured using mean squared error (MSE) or binary cross-entropy loss, depending on the nature of the data. This training process encourages the autoencoder to learn efficient and meaningful representations of the input data.

3.2.2 *Regularization*

Regularization is a critical component of the AMRAE algorithm, ensuring that the autoencoder learns robust and generalizable representations. Regularization techniques help prevent overfitting, improve stability, and enhance the interpretability of the learned representations. AMRAE incorporates several regularization methods:

- L1 Regularization: Also known as Lasso regularization, L1 regularization adds a penalty equal to the absolute value of the magnitude of coefficients to the loss function. It encourages sparsity in the learned representations by driving less important feature weights to zero [54].
- L2 Regularization: Also known as Ridge regularization, L2 regularization adds a penalty equal to the square of the magnitude of coefficients to the loss function. It helps prevent overfitting by penalizing large weights and encouraging smoother, more generalizable representations [55].
- Elastic Net Regularization: Elastic net regularization combines L1 and L2 regularization, balancing the benefits of both techniques. It is particularly useful when dealing with correlated features, as it encourages group selection and sparsity [56].
- Dropout: Dropout is a regularization technique that randomly drops neurons during training, preventing the network from becoming too reliant on any single feature. This technique helps improve the generalization of the autoencoder by reducing overfitting [57].
- Batch Normalization: Batch normalization normalizes the inputs to each layer, stabilizing the learning process and improving convergence. It also acts as a regularizer by introducing a slight noise in the learning process [58].

3.2.3 *Manifold Concept*

Manifold concept is a key aspect of AMRAE, ensuring that the learned representations capture the intrinsic geometry of the data. The integration of manifold concept into the autoencoder framework involves the following steps:

1. **Geometric Regularization:** AMRAE incorporates geometric regularization derived from manifold concept. This regularization ensures that the latent space learned by the autoencoder preserves the local and global structures of the data. Techniques such as Laplacian Eigenmaps or methods inspired by t-SNE, LLE, and Isomap can be used to compute geometric loss terms that guide the training of the autoencoder.
2. **Manifold Adjustment Box:** The manifold adjustment box is a flexible component within AMRAE that allows for various configurations and adjustments based on manifold concept principles. This box can include different activation functions (e.g., ReLU, tanh, leaky ReLU), multiple hidden layers, and other architectural choices that enhance the autoencoder's ability to capture manifold structures.
3. **Preservation of Local and Global Structures:** By integrating manifold concept, AMRAE ensures that both local neighbourhood relationships and global data structures are preserved in the latent space. This balanced preservation is crucial

for accurately representing the data's intrinsic geometry.

The combination of these components in AMRAE provides a robust and scalable solution for dimensionality reduction, addressing the limitations of traditional techniques and enhancing the ability to uncover meaningful patterns in high-dimensional data.

3.3 Detailed Explanation

The AMRAE algorithm consists of several critical steps, each contributing to its overall effectiveness in dimensionality reduction. This section provides a detailed explanation of these steps, including data preprocessing, the architecture of the autoencoder, the incorporation of manifold regularization, the training process, and the extraction of lower-dimensional representations.

3.3.1 Data Preprocessing

Data preprocessing is an essential step to ensure that the input data is suitable for training the autoencoder. The main task in this step is normalization. To ensure that the data is on a common scale, normalization is performed. This process involves scaling the features of the dataset such that they have zero mean and unit variance. Normalization helps in accelerating the training process and improving the convergence of the autoencoder.

3.3.2 Autoencoder Architecture

The autoencoder used in AMRAE is designed to efficiently capture the underlying structure of the data. The architecture consists of two main components: the encoder and the decoder [59–61].

- **Encoder:** The encoder compresses the input data into a lower-dimensional latent space. It consists of multiple layers of neurons, each applying a nonlinear transformation to the input. The encoder's layers progressively reduce the dimensionality of the data. For example:
  - Input Layer: Takes the preprocessed data as input.
  - Hidden Layers: Several dense (fully connected) layers with activation functions (e.g., ReLU) that reduce the dimensionality step-by-step.
  - Latent Space: The final layer of the encoder represents the lower-dimensional latent space.
- **Decoder:** The decoder reconstructs the input data from the latent space. It consists of multiple layers of neurons that progressively increase the dimensionality of the data back to its original form. The decoder's layers mirror the encoder's structure. For example:
  - Latent Space: Takes the lower-dimensional representation as input.
  - Hidden Layers: Several dense layers with activation functions that increase the dimensionality step-by-step.
  - Output Layer: The final layer of the decoder outputs the reconstructed data (See Figure 2).

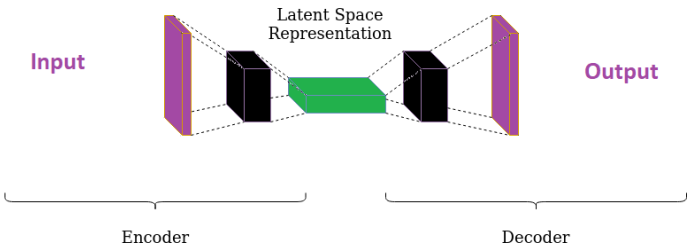

Figure 2. Structure of an autoencoder: The encoder maps input data to a lower-dimensional latent space, which the decoder then reconstructs back to the original input, minimizing reconstruction error.

3.3.3 Manifold Regularization

Manifold regularization is integrated into the autoencoder's loss function to ensure that the learned latent space preserves the intrinsic geometric properties of the data. The key idea is to add a regularization term that enforces the preservation of local and global structures.

Geometric Loss: This loss term is derived from manifold concept. For example, Laplacian Eigenmaps can be used to compute a Laplacian matrix that captures the local structure of the data. The geometric loss ensures that the latent space respects these local relationships.

Regularization Term: The overall loss function of the autoencoder is modified to include the geometric loss. The modified loss function can be expressed as:

$$Loss = Loss_{Reconstruction} + \lambda \times Loss_{Geometric} \quad (1)$$

where λ is a regularization parameter that controls the trade-off between the reconstruction loss and the geometric loss [62].

3.3.4 Training Process

The training process of the autoencoder involves optimizing the loss function to learn efficient representations of the data. The main components of the training process are:

Loss Function: The loss function includes both the reconstruction loss and the manifold regularization term. The reconstruction loss measures the difference between the input data and the reconstructed data. Commonly used reconstruction losses include MSE and binary cross-entropy, depending on the nature of the data [63, 64].

Optimization: The autoencoder is trained using gradient-based optimization methods, such as stochastic gradient descent (SGD) or Adam. The optimization process involves updating the weights of the encoder and decoder to minimize the loss function. The training is performed iteratively over multiple epochs, with mini-batch gradient descent often used to improve convergence [65].

3.3.5 Dimensionality Reduction

Once the autoencoder is trained, the lower-dimensional representations of the input data can be extracted from the latent space. The steps involved in this process are:

- Encoding: The encoder part of the autoencoder is used to transform the input data into the latent space. This step involves feeding the input data through the encoder's layers to obtain the lower-dimensional representation [66].
- Latent Space: The output of the encoder, which is the latent space, represents the reduced-dimensional data. These representations capture the most significant features and structures of the original data while reducing its dimensionality [67].

The lower-dimensional representations obtained from the latent space can be used for various downstream tasks, such as visualization, clustering, classification, and anomaly detection. The integration of manifold concept principles ensures that these representations preserve the intrinsic geometry of the data, making them more meaningful and interpretable. By combining data preprocessing, a carefully designed autoencoder architecture, manifold regularization, and an effective training process, AMRAE achieves robust and scalable dimensionality reduction. This detailed explanation highlights the key components and steps involved in the algorithm, showcasing its ability to address the limitations of traditional dimensionality reduction techniques.

3.4 The Manifold Adjustment Box

The Manifold Adjustment Box is a crucial component of the AMRAE algorithm. It allows for flexible configurations and adjustments to capture the intrinsic geometric properties of high-dimensional data effectively. This section provides a detailed explanation of the Manifold Adjustment Box, its role, basic implementation, and potential enhancements.

3.4.1 Definition and Role

The Manifold Adjustment Box refers to a set of techniques and configurations integrated within the autoencoder framework to enhance its ability to capture and preserve the geometric structure of the data. This box includes various regularization methods, activation functions, architectural choices, and other enhancements that guide the autoencoder to learn representations that respect the underlying geometry of the data.

The primary role of the Manifold Adjustment Box is to ensure that the autoencoder not only reduces the dimensionality of the data but also maintains its intrinsic structure. By incorporating manifold concept principles and regularization techniques, the Manifold Adjustment Box helps in:

- Preserving local neighbourhood relationships.
- Maintaining global data structures.
- Enhancing the robustness of the learned representations to noise and outliers.
- Improving the interpretability and stability of the lower-dimensional representations.

3.4.2 Basic Implementation

A baseline implementation of the Manifold Adjustment Box can be achieved using L1 regularization. L1 regularization, also known as Lasso regularization, adds a penalty equal to the absolute value of the magnitude of coefficients to the loss function. This encourages sparsity in the learned representations, helping the autoencoder to focus on the most critical features and reducing the risk of overfitting. Algorithm 1 is an example code snippet demonstrating the baseline implementation of the Manifold Adjustment Box using L1 regularization in the AMRAE framework:

Algorithm 1: Baseline Implementation of Manifold Adjustment Box with L1 Regularization

Input: Isolet dataset, train-test split, scaling factor
Output: Classification accuracy after dimensionality reduction
Load Isolet dataset:
Load dataset using `fetch_openml`;
Split into features `X` and labels `y`;
Convert `y` to integers;
Split data into training and test sets;
Preprocess data:
Scale `X_train` and `X_test` with `StandardScaler`;
Define Autoencoder with L1 regularization:
Define function `basic_l1(input_img, encoding_dim);`
  `encoded` = Apply `Dense` with `encoding_dim` and L1 regularization;
  `decoded` = Apply `Dense` for decoding;
Autoencoder model:
Define input layer with shape `input_dim`;
Use `basic_l1` to create encoded and decoded layers;
Define `autoencoder` and `encoder` models;
`autoencoder.compile(optimizer='adam', loss='binary_crossentropy');`
Train the autoencoder:
Fit `autoencoder` with `X_train`;
Extract representations:
Use `encoder.predict` to obtain `X_train_amrae` and `X_test_amrae`;
Handle NaN values:
Replace NaNs in `X_train_amrae` and `X_test_amrae` with 0;
Evaluate Classifier:
Define function `evaluate_classifier(X_train, y_train, X_test, y_test);`
  Train `LogisticRegression` on `X_train` and `y_train`;
  Predict `X_test` and compute accuracy;
  Return accuracy;
Evaluate the model:
Call `evaluate_classifier` with `X_train_amrae`, `y_train`, `X_test_amrae`, `y_test`;
Output:
Print the accuracy score;

Figure 3 is the visual representation of the basic_l1 function architecture illustrates a simple autoencoder model. It consists of three layers: an input layer, a dense layer with L1 regularization, and an output layer. The input layer receives data with a shape of input_dim, which represents the dimensionality of the input features. The dense layer, with an encoding dimension specified by encoding_dim, uses the ReLU activation function and applies L1 regularization to encourage sparsity in the encoded representation. Finally, the output layer reconstructs the input data using the sigmoid activation function. This visualization effectively captures the flow of data through the network and highlights the application of regularization in the dense layer to improve the model's robustness and generalization.

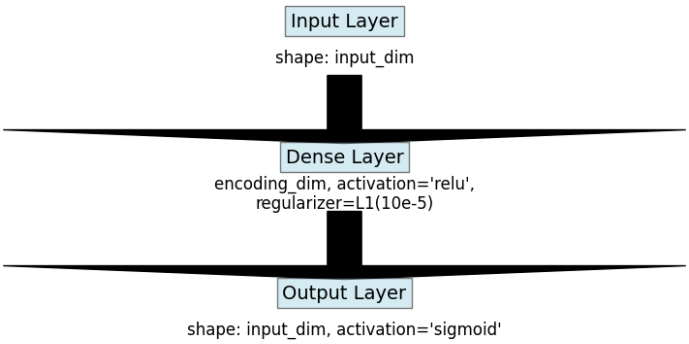

Figure 3. Visualization of the basic_l1 Function Architecture: Input Layer to Dense Layer with L1 Regularization to Output Layer

3.4.3 Potential Enhancements and Adjustments in Manifold

The baseline implementation of the Manifold Adjustment Box using L1 regularization can be enhanced with various techniques to further improve the performance and robustness of AMRAE. Some potential enhancements and variations include:

1. L2 Regularization: Also known as Ridge regularization, L2 regularization adds a penalty equal to the square of the magnitude of coefficients to the loss function. This helps in reducing overfitting and encouraging smoother representations (See Algorithm 2). Figure 4 is the visual representation of the l2_regularization function architecture illustrates a simple autoencoder model. It consists of three layers: an input layer, a dense layer with L2 regularization, and an output layer. The input layer receives data with a shape of input_dim, representing the dimensionality of the input features. The dense layer, which reduces the data to the encoding dimension specified by encoding_dim, uses the ReLU activation function and applies L2 regularization. This regularization technique penalizes the squared magnitude of the coefficients, helping to prevent overfitting and

Algorithm 2: Applying L2 Regularization in Autoencoder

Input: Input data `input_img`, encoding dimension `encoding_dim`

Output: Encoded and Decoded representations

Define L2 regularization autoencoder architecture:

Define function `l2_regularization(input_img, encoding_dim);`

`encoded` = Apply `Dense` layer with `encoding_dim` units, ReLU activation, and L2 regularization with penalty 10^{-5};

`decoded` = Apply `Dense` layer with `input_dim` units and sigmoid activation to reconstruct the input;

Output:

Return `encoded` and `decoded` layers;

encourage smoother representations. Finally, the output layer reconstructs the input data using the sigmoid activation function. This visualization effectively demonstrates the flow of data through the network, highlighting the application of L2 regularization in the dense layer to improve the model's generalization.

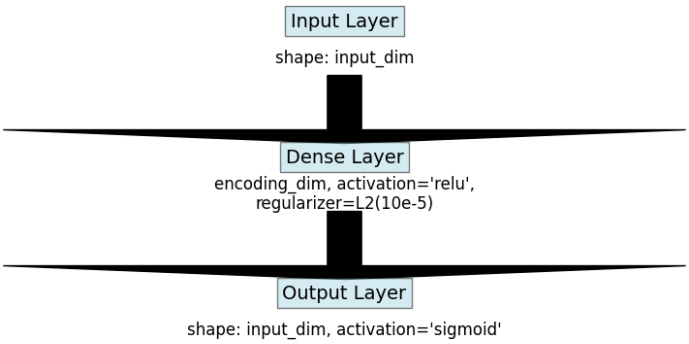

Figure 4. Visualization of the l2_regularization Function Architecture: Input Layer to Dense Layer with L2 Regularization to Output Layer.

2. **Elastic Net Regularization:** Combines L1 and L2 regularization, balancing the benefits of both techniques. It is particularly useful when dealing with correlated features (See Algorithm 3). Figure 5 is the visual representation of the elastic_net function architecture shows a simple autoencoder model consisting of three layers: an input layer, a dense layer with Elastic Net regularization, and an output layer. The input layer receives data with a shape of input_dim, which represents the dimensionality of the input features. The dense layer reduces

Algorithm 3: Elastic Net Autoencoder Layer

Input: Input vector $x \in \mathbb{R}^n$, encoding dimension d

Output: Encoded representation z, reconstructed vector $\hat{x}$

Set L1 regularization coefficient $\lambda_1 \leftarrow 10^{-4}$;

Set L2 regularization coefficient $\lambda_2 \leftarrow 10^{-4}$;

`// Encoding layer with ReLU activation`

$z \leftarrow \text{ReLU}(W_e x + b_e)$;

`// Apply Elastic Net regularization during training`

$\mathcal{L}_{reg} \leftarrow \lambda_1 \|W_e\|_1 + \lambda_2 \|W_e\|_2^2$;

`// Decoding layer with Sigmoid activation`

$\hat{x} \leftarrow \text{Sigmoid}(W_d z + b_d)$;

return $z, \hat{x}$;

the data to the encoding dimension specified by encoding_dim, uses the ReLU activation function, and applies Elastic Net regularization, which combines both L1 and L2 penalties (L1=10e-5, L2=10e-5). This hybrid regularization technique helps in managing both sparsity and smoothness, making it robust to different types of data. Finally, the output layer reconstructs the input data using the sigmoid activation function. This visualization effectively demonstrates the flow of data through the network and highlights the application of Elastic Net regularization in the dense layer to improve model generalization and robustness.

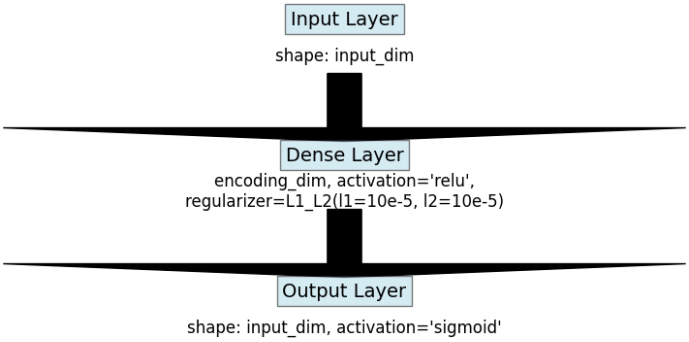

Figure 5. Visualization of the elastic_net Function Architecture: Input Layer to Dense Layer with Elastic Net Regularization to Output Layer.

3. **Advanced Activation Functions:** Different activation functions can be used to improve the learning capabilities of the autoencoder.
   - **Tanh Activation:** Provides a smooth, nonlinear transformation and helps in capturing complex patterns (See Algorithm 4). Figure 6 is the visual representation of the tanh_activation function architecture shows

Algorithm 4: Tanh Activation Autoencoder Layer

Input: Input vector $x \in \mathbb{R}^n$, encoding dimension d
Output: Encoded representation z, reconstructed vector $\hat{x}$

```
// Encoding layer with Tanh activation
```
$z \leftarrow \text{Tanh}(W_e x + b_e)$;
```
// Decoding layer with Sigmoid activation
```
$\hat{x} \leftarrow \text{Sigmoid}(W_d z + b_d)$;
return $z, \hat{x}$;

a simple autoencoder model consisting of three layers: an input layer, a dense layer with Tanh activation, and an output layer. The input layer receives data with a shape of input_dim, representing the dimensionality of the input features. The dense layer reduces the data to the encoding dimension specified by encoding_dim and uses the Tanh activation function, which provides a smooth, nonlinear transformation that helps capture complex patterns in the data. Finally, the output layer reconstructs the input data using the sigmoid activation function. This visualization effectively demonstrates the flow of data through the network and highlights the application of the Tanh activation function in the dense layer to improve the model's ability to learn from the data.

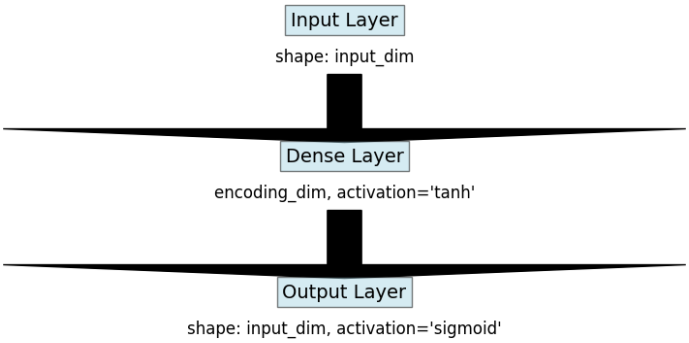

Figure 6. Visualization of the tanh_activation Function Architecture: Input Layer to Dense Layer with Tanh Activation to Output Layer.

- **Leaky ReLU Activation:** Allows a small gradient when the unit is not active, improving the model's ability to learn (See Algorithm 5). Figure 7 is the visual representation of the leaky_relu_activation function architecture shows a simple autoencoder model consisting of four

Algorithm 5: Leaky ReLU Activation Autoencoder Layer

Input: Input vector $x \in \mathbb{R}^n$, encoding dimension d
Output: Encoded representation z, reconstructed vector $\hat{x}$

```
// Encoding layer with Linear transformation
   followed by Leaky ReLU activation
```
$z \leftarrow W_e x + b_e$;
$z \leftarrow \text{LeakyReLU}(z, \alpha = 0.1)$;
```
// Decoding layer with Sigmoid activation
```
$\hat{x} \leftarrow \text{Sigmoid}(W_d z + b_d)$;
return $z, \hat{x}$;

key components: an input layer, a dense layer with linear activation, a Leaky ReLU activation layer, and an output layer. The input layer receives data with a shape of input_dim, representing the dimensionality of the input features. The dense layer reduces the data to the encoding dimension specified by encoding_dim and initially uses a linear activation function. This is followed by a Leaky ReLU activation with an alpha value of 0.1, which allows a small gradient when the unit is not active, thereby improving the model's ability to learn. Finally, the output layer reconstructs the input data using the sigmoid activation function. This visualization effectively demonstrates the flow of data through the network and highlights the application of Leaky ReLU activation to enhance the model's learning capability.

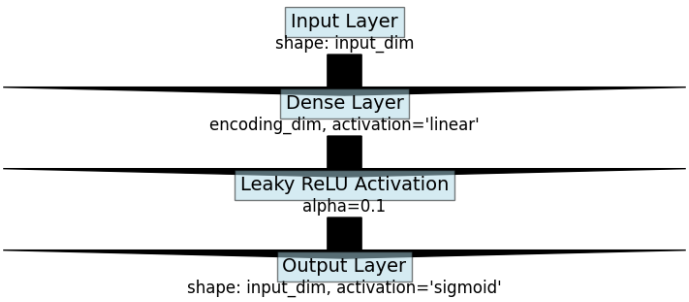

Figure 7. Visualization of the leaky_relu_activation Function Architecture: Input Layer to Dense Layer with Linear Activation, followed by Leaky ReLU Activation to Output Layer.

4. **Dropout Layers:** Dropout is a regularization technique that randomly drops neurons during training, preventing the network from becoming too reliant on any single feature. This enhances

the generalization of the autoencoder. Figure 8

Algorithm 6: Dropout Layer Autoencoder

Input: Input vector $x \in \mathbb{R}^n$, encoding dimension d
Output: Encoded representation z, reconstructed vector $\hat{x}$

```
// Encoding layer with ReLU activation
```
$z \leftarrow \text{ReLU}(W_e x + b_e)$;
```
// Apply Dropout to encoded layer
```
$z \leftarrow \text{Dropout}(z, \text{rate} = 0.5)$;
```
// Decoding layer with Sigmoid activation
```
$\hat{x} \leftarrow \text{Sigmoid}(W_d z + b_d)$;
return $z, \hat{x}$;

is the visual representation of the dropout_layer function architecture shows a simple autoencoder model consisting of three layers: an input layer, a dense layer with ReLU activation, a dropout layer, and an output layer. The input layer receives data with a shape of input_dim, representing the dimensionality of the input features. The dense layer reduces the data to the encoding dimension specified by encoding_dim and uses the ReLU activation function to introduce non-linearity. This is followed by a dropout layer with a dropout rate of 0.5, which randomly drops 50% of the neurons during training to prevent overfitting and enhance generalization. Finally, the output layer reconstructs the input data using the sigmoid activation function. This visualization effectively demonstrates the flow of data through the network and highlights the application of dropout regularization to improve the model's robustness.

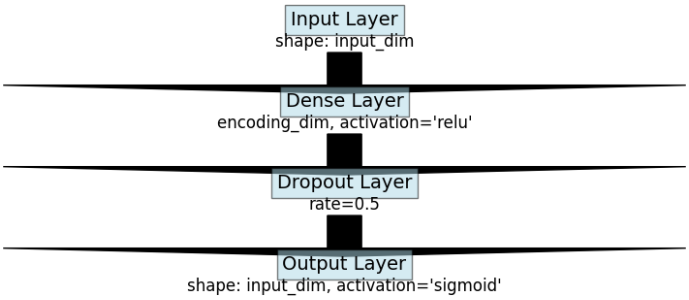

Figure 8. Visualization of the dropout_layer Function Architecture: Input Layer to Dense Layer with ReLU Activation, followed by Dropout Layer to Output Layer.

5. *Batch Normalization:* Normalizes the inputs to each layer, stabilizing the learning process and improving convergence. It also acts as a regularizer by introducing slight noise in the learning process.

Algorithm 7: Batch Normalization Layer

Input: Input vector $x \in \mathbb{R}^n$, encoding dimension d
Output: Encoded representation z, reconstructed vector $\hat{x}$

```
// Encoding layer with ReLU activation
```
$z \leftarrow \text{ReLU}(W_e x + b_e)$;
```
// Apply Batch Normalization to stabilize
   training
```
$z \leftarrow \text{BatchNorm}(z)$;
```
// Decoding layer with Sigmoid activation
```
$\hat{x} \leftarrow \text{Sigmoid}(W_d z + b_d)$;
return $z, \hat{x}$;

Figure 9 is the visual representation of the batch_norm_layer function architecture shows a simple autoencoder model consisting of four key components: an input layer, a dense layer with ReLU activation, a batch normalization layer, and an output layer. The input layer receives data with a shape of input_dim, representing the dimensionality of the input features. The dense layer reduces the data to the encoding dimension specified by encoding_dim and uses the ReLU activation function to introduce non-linearity. This is followed by a batch normalization layer, which normalizes the inputs to the next layer, stabilizing the learning process and improving convergence. Finally, the output layer reconstructs the input data using the sigmoid activation function. This visualization effectively demonstrates the flow of data through the network and highlights the application of batch normalization to enhance the model's stability and learning efficiency.

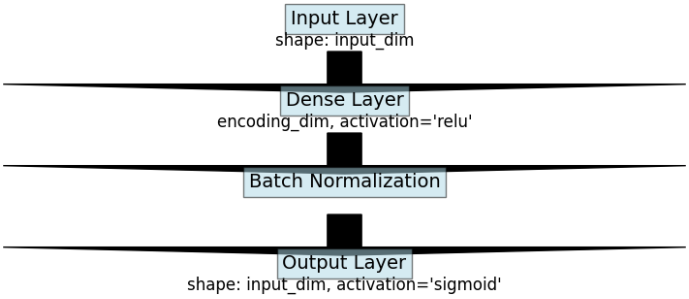

Figure 9. Visualization of the batch_norm_layer Function Architecture: Input Layer to Dense Layer with ReLU Activation, followed by Batch Normalization to Output Layer.

6. **Multiple Hidden Layers:** Adding multiple hidden layers allows the autoencoder to capture more complex patterns and hierarchical

representations in the data.

Algorithm 8: Multi-Layer Autoencoder with Hidden Layers

Input: Input vector $x \in \mathbb{R}^n$, encoding dimension d
Output: Encoded representation z, reconstructed vector $\hat{x}$

```
// First hidden encoding layer with ReLU activation
```
$z \leftarrow \text{ReLU}(W_{e1}x + b_{e1})$;
```
// Second hidden encoding layer with ReLU activation
```
$z \leftarrow \text{ReLU}(W_{e2}z + b_{e2})$;
```
// Decoding layer with Sigmoid activation
```
$\hat{x} \leftarrow \text{Sigmoid}(W_d z + b_d)$;
return $z, \hat{x}$;

Figure 10 is the visual representation of the multiple_hidden_layers function architecture shows a simple autoencoder model consisting of five key components: an input layer, two dense layers with ReLU activation, and an output layer. The input layer receives data with a shape of input_dim, representing the dimensionality of the input features. The first dense layer reduces the data to the encoding dimension specified by encoding_dim and uses the ReLU activation function to introduce non-linearity. This is followed by a second dense layer, which also uses ReLU activation to further process the encoded data. Finally, the output layer reconstructs the input data using the sigmoid activation function. This visualization effectively demonstrates the flow of data through the network, highlighting the application of multiple hidden layers to capture more complex patterns and hierarchical representations in the data.

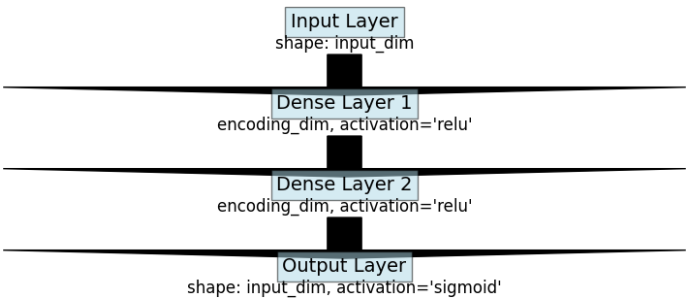

Figure 10. Visualization of the multiple_hidden_layers Function Architecture: Input Layer to Two Dense Layers with ReLU Activation, followed by Output Layer.

7. **Residual Connections:** Residual connections add shortcuts between layers, helping to mitigate the vanishing gradient problem and allowing for deeper networks.

Algorithm 9: Autoencoder with Residual Connections

Input: Input vector $x \in \mathbb{R}^n$, encoding dimension d
Output: Encoded representation z, reconstructed vector $\hat{x}$

```
// Initial encoding layer with ReLU activation
```
$z \leftarrow \text{ReLU}(W_e x + b_e)$;
```
// Compute residual branch
```
$r \leftarrow \text{ReLU}(W_r z + b_r)$;
```
// Add residual connection to support gradient flow
```
$z \leftarrow z + r$;
```
// Decoding layer with Sigmoid activation
```
$\hat{x} \leftarrow \text{Sigmoid}(W_d z + b_d)$;
return $z, \hat{x}$;

Figure 11 is the visual representation of the residual_connections function architecture shows a neural network model consisting of several key components: an input layer, two dense layers with ReLU activation, a residual connection layer, and an output layer. The input layer receives data with a shape of input_dim, representing the dimensionality of the input features. The first dense layer reduces the data to the encoding dimension specified by encoding_dim and uses the ReLU activation function to introduce non-linearity. The second dense layer, also using ReLU activation, serves as the residual component. The output of this layer is added to the output of the first dense layer through the add layer, creating a residual connection that helps mitigate the vanishing gradient problem and allows for deeper networks. Finally, the output layer reconstructs the input data using the sigmoid activation function. This visualization effectively demonstrates the flow of data through the network, highlighting the use of residual connections to enhance the model's learning capabilities and stability.

8. **Variational Autoencoders (VAEs):** Use a probabilistic approach to model the latent space, incorporating regularization through a Kullback-Leibler divergence term to ensure smoothness and disentanglement.

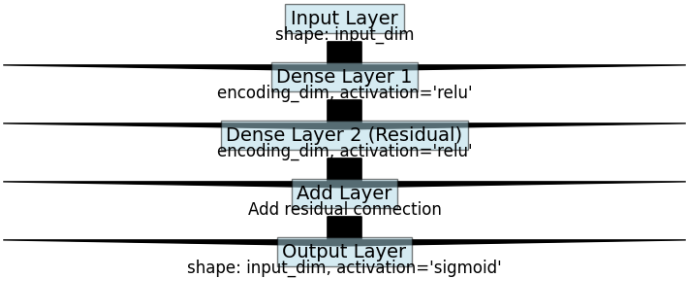

Figure 11. Visualization of the residual_connections Function Architecture: Input Layer to Two Dense Layers with ReLU Activation and Residual Connection, followed by Output Layer.

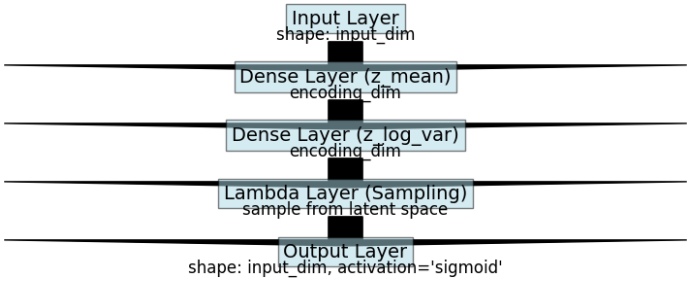

Figure 12. Visualization of the variational_autoencoder Function Architecture: Input Layer to Dense Layers with Mean and Log Variance, Sampling Lambda Layer, and Output Layer.

Algorithm 10: Variational Autoencoder (VAE)

Input: Input vector $x \in \mathbb{R}^n$, encoding dimension d
Output: Latent sample z, reconstructed vector $\hat{x}$

```
// Compute parameters of the latent
   distribution
```

$\mu \leftarrow W_\mu x + b_\mu$;
$\log \sigma^2 \leftarrow W_{\log \sigma} x + b_{\log \sigma}$;

```
// Sample latent variable using
   reparameterization trick
```

Sample $\epsilon \sim \mathcal{N}(0, I)$;
$z \leftarrow \mu + \exp(0.5 \cdot \log \sigma^2) \odot \epsilon$;

```
// Decode latent sample to reconstruct input
```

$\hat{x} \leftarrow \text{Sigmoid}(W_d z + b_d)$;
return z, $\hat{x}$;

Figure 12 is the visual representation of the variational_autoencoder function architecture shows a sophisticated neural network model consisting of several key components: an input layer, two dense layers (z_mean and z_log_var), a lambda layer for sampling, and an output layer. The input layer receives data with a shape of input_dim, representing the dimensionality of the input features. The dense layers calculate the mean and log variance of the latent space distribution, essential for the variational autoencoder's probabilistic approach. The lambda layer samples from the latent space using these parameters, introducing randomness that allows for the generation of new, similar data points. Finally, the output layer reconstructs the input data using the sigmoid activation function. This architecture highlights the unique aspects of variational autoencoders, particularly their ability to model the underlying data distribution and generate new samples, making them powerful tools for generative modeling tasks.

9. **Laplacian Regularization:** Incorporate Laplacian Eigenmaps to add a geometric regularization term that preserves the local structure of the data. Figure 13 is the visual representation of

Algorithm 11: Autoencoder with Laplacian Regularization

Input: Input vector $x \in \mathbb{R}^n$, encoding dimension d, adjacency matrix $A \in \mathbb{R}^{d \times d}$
Output: Encoded representation z, reconstructed vector $\hat{x}$

```
// Encode input using ReLU activation
```

$z \leftarrow \text{ReLU}(W_e x + b_e)$;

```
// Compute Laplacian regularization loss
```

$\mathcal{L}_{\text{lap}} \leftarrow \|z - Az\|^2$;

```
// Add Laplacian loss to total training
   objective
```

Add $\mathcal{L}_{\text{lap}}$ to model loss;

```
// Decode with sigmoid activation
```

$\hat{x} \leftarrow \text{Sigmoid}(W_d z + b_d)$;
return z, $\hat{x}$;

the laplacian_regularization function architecture shows a neural network model consisting of an input layer, a dense layer, an adjacency matrix input, a Laplacian loss layer, and an output layer. The input layer receives data with a shape of input_dim, representing the dimensionality of the input features. The dense layer reduces the data to the encoding dimension specified by encoding_dim and uses the ReLU activation function to introduce non-linearity. The adjacency matrix input provides a matrix that captures the relationships between the data points. The Laplacian loss layer computes a custom loss function based on the local structure of the data, ensuring that the learned representations respect these local relationships. Finally, the output layer reconstructs the input

Table 1. Detailed descriptions of the datasets used in the study, including the number of instances, classes, and dimensionality for each dataset.

Dataset Name	Description	Number of Instances	Number of Classes	Dimensionality
Isolet	Spoken letter data with 617 features extracted from raw audio recordings.	7,797	26 (letters A-Z)	617
Digits	Images of handwritten digits, each an 8x8 pixel grayscale image, flattened into a vector.	1,797	10 (digits 0-9)	64
Iris	Measurements of iris flowers from three different species.	150	3 (Setosa, Versicolour, Virginica)	4
Wine	Chemical analysis of wines grown in the same region in Italy from three different cultivars.	178	3 (different cultivars)	13
Breast Cancer	Features computed from digitized images of breast mass tissue samples.	569	2 (benign, malignant)	30
Musk 1	Features of molecules classified based on whether they smell musky or not.	476	2 (musk, non-musk)	166
MNIST	Images of handwritten digits, each 28x28 pixels, flattened into a vector.	70,000 (60,000 for training, 10,000 for testing)	10 (digits 0-9)	784
Yeast	Protein localization sites within a yeast cell, characterized by 8 features.	1,484	10 (localization sites)	8
Abalone	Physical measurements of abalone, used to predict their age.	4,177	28 (age groups)	8
Olivetti Faces	Grayscale images of faces, each 64x64 pixels, flattened into a vector.	400	40 (individuals)	4,096

data using the sigmoid activation function. This architecture effectively demonstrates the integration of geometric regularization through the Laplacian loss, highlighting its role in preserving the local structure of the data within the latent space.

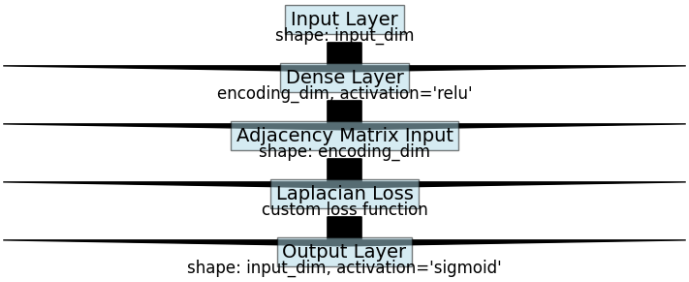

Figure 13. Visualization of the laplacian_regularization Function Architecture: Input Layer to Dense Layer with ReLU Activation, Adjacency Matrix Input, Laplacian Loss Layer, and Output Layer.

These enhancements can be tailored to specific datasets and applications, making AMRAE a flexible and powerful tool for dimensionality reduction. By exploring different configurations within the Manifold Adjustment Box, researchers can optimize the algorithm's performance and achieve more accurate and meaningful representations of high-dimensional data.

4 Experiments and Results

4.1 Dataset Description

In this study, we evaluate the performance of the AMRAE algorithm across a diverse set of datasets. These datasets are chosen to cover a wide range of data types, dimensionalities, and complexities, providing a comprehensive assessment of the algorithm's capabilities. Below, we provide detailed descriptions of each dataset used in our experiments.

Table 1 summarizes the diverse datasets used to evaluate the performance of the AMRAE algorithm, encompassing a wide array of data types, dimensionalities, and complexities. These datasets include spoken letter data (Isolet), handwritten digit images (Digits, MNIST), flower measurements (Iris), chemical analysis of wines (Wine), breast cancer tissue samples (Breast Cancer), molecular features (Musk 1), protein localization in yeast (Yeast), physical measurements of abalone (Abalone), and grayscale facial images (Olivetti Faces). Each dataset's unique characteristics provide a comprehensive assessment of AMRAE's capabilities. These datasets were selected to provide a broad spectrum of challenges, including different levels of intrinsic dimensionality, complexity, noise, and structure. This variety ensures a thorough evaluation of AMRAE's ability to perform dimensionality reduction and maintain the meaningful structure of data across different contexts.

4.2 Experimental Setup

All experiments were conducted using Google Colab, a cloud-based platform that provides free access to GPUs and TPUs, significantly accelerating deep learning tasks. The tools and libraries employed included Python for implementation, TensorFlow and Keras for building and training autoencoders, Scikit-learn for data preprocessing and evaluation, NumPy for numerical computations, and Matplotlib for visualizing results, such as the distribution of anomaly scores and cumulative explained variance.

To ensure a fair and comprehensive evaluation of the AMRAE algorithm and its various configurations, we followed standardized training and evaluation protocols. Multiple datasets, including Isolet, Digits, Iris, Wine, Breast Cancer, Musk 1, MNIST, Yeast, Abalone, and Olivetti Faces, were used to test the effectiveness of AMRAE. Each dataset was pre-processed by normalizing the features to have zero mean and unit variance, and for certain datasets, initial dimensionality reduction using Principal Component Analysis (PCA) was performed to reduce computational complexity.

Various configurations of the autoencoder were implemented, incorporating different regularization techniques and activation functions within the Manifold Adjustment Box. The autoencoders were trained using the Adam optimizer with a learning rate of 0.001 for 50 epochs, and a batch size of 256 was used to balance training time and stability. During training, a validation set was utilized to monitor performance and prevent overfitting, employing early stopping if the validation loss did not improve for 10 consecutive epochs.

After training, the encoder part of the autoencoder was used to transform the high-dimensional data into lower-dimensional representations. A logistic regression classifier was then trained on these lower-dimensional representations, with accuracy as the primary evaluation metric. The performance of AMRAE was compared against traditional dimensionality reduction techniques, including PCA, t-SNE, LLE, MDS, and Isomap, to demonstrate its effectiveness.

By following these standardized protocols, we ensured that the experiments were reproducible, and the results were reliable. The comprehensive evaluation across multiple datasets and configurations demonstrated the robustness and effectiveness of the AMRAE algorithm, highlighting its potential as a powerful tool for dimensionality reduction in various applications.

4.3 Evaluation Metrics

To comprehensively assess the performance of the AMRAE algorithm and its various configurations, we employed a range of evaluation metrics. These metrics were selected to provide insights into different aspects of the dimensionality reduction process, including the quality of the reduced representations, the preservation of intrinsic data structures, and the robustness of the learned features. The following metrics were used for evaluation:

1. **Accuracy:**
   Accuracy is a fundamental metric used to evaluate the performance of a classification model. It is defined as the ratio of correctly predicted instances to the total instances in the dataset.

$$\text{Accuracy} = \frac{Number\ of\ Correct\ Predictions}{Total\ \ number\ \ of\ \ samples} \tag{2}$$

   After training the autoencoder, a logistic regression classifier was trained on the lower-dimensional representations. The accuracy of this classifier was used to measure how well the dimensionality reduction technique preserved the discriminative information in the data [68].

2. **Silhouette Score:**
   The silhouette score is a measure of how similar an object is to its own cluster compared to other

Table 2. Accuracy results of the AMRAE algorithm across different configurations of the Manifold Adjustment Box on various datasets. Each configuration represents a different enhancement or regularization technique applied within the autoencoder framework. The results demonstrate the effectiveness and robustness of AMRAE in capturing intrinsic data structures and improving classification performance.

Configuration	Isolet	Digits	Iris	Wine	Breast Cancer	Musk 1	MNIST	Yeast	Abalone	Olivetti Faces
Basic L1 Regularization	61.54	94.17	96.67	97.22	95.61	84.17	91.49	62.29	30.26	27.5
L2 Regularization	62.69	94.72	96.67	97.22	97.37	84.17	91.50	60.94	28.83	30
Elastic Net Regularization	62.31	95.83	96.67	99.9	98.25	84.17	91.64	60.94	29.31	27.5
Tanh Activation	64.74	95.00	99.09	99.9	98.25	84.17	91.59	60.61	27.75	56.25
Leaky ReLU Activation	72.24	96.39	96.67	99.9	96.49	84.17	91.62	60.61	29.78	68.75
Dropout Layer	60.9	94.17	96.67	97.22	97.37	84.17	92.2	61.28	29.07	32.5
Batch Norm Layer	81.86	95.00	99.09	99.9	95.61	84.17	90.76	59.93	26.56	61.25
Multiple Hidden Layers	14.42	87.22	96.67	97.22	97.37	84.17	91.50	60.27	26.08	22.5
Residual Connections	12.31	92.50	93.33	97.22	98.25	84.17	91.87	58.59	25.84	13.75
Variational Autoencoder	3.21	83.06	90	88.89	93.86	84.17	91.63	48.82	19.74	10

clusters. It provides an indication of the clustering tendency of the data.

$$\text{Silhouette Score} = \frac{b - a}{max(a, b)} \tag{3}$$

where a is the mean distance between a sample and all other points in the same cluster, and b is the mean distance between a sample and all other points in the next nearest cluster. The silhouette score was used as a proxy for data complexity, helping to evaluate the effectiveness of the dimensionality reduction in maintaining cluster structures [69].

3. **Intrinsic Dimensionality:**
   Intrinsic dimensionality refers to the minimum number of dimensions needed to represent the data without significant loss of information. It provides insights into the true complexity of the data. The intrinsic dimensionality was estimated using the Fisher Information Matrix. This method involves calculating the eigenvalues of the covariance matrix of the data and determining the number of dimensions required to capture a specified amount of variance (e.g., 95%). This metric helped us understand the complexity of the datasets and assess how well the dimensionality reduction techniques captured the intrinsic structure of the data [70].

4. **Mean Anomaly Score:**
   The mean anomaly score is used to evaluate the presence of outliers in the data. It is computed using the Isolation Forest algorithm, which detects anomalies by isolating observations through random partitioning. The anomaly score for each data point is calculated based on the number of splits required to isolate it. The mean anomaly score is then computed as the average of these scores. This metric provided insights into the robustness of the learned representations to noise and outliers, indicating the stability of the dimensionality reduction process [71].

By employing these diverse evaluation metrics, we were able to thoroughly assess the performance of AMRAE and its various configurations across multiple dimensions. This comprehensive evaluation provided a deeper understanding of the algorithm's strengths and limitations, highlighting its potential as a robust and effective tool for dimensionality reduction in high-dimensional data.

5 Results

This section presents the results of the AMRAE algorithm across different adjustments and enhancements. We provide detailed results for each configuration and perform a comparative analysis against traditional dimensionality reduction techniques. Table 2 summarizes the accuracy results for each configuration of the Manifold Adjustment Box on various datasets:

The Isolet dataset achieved the best accuracy with batch normalization at 81.86%, followed by leaky ReLU activation at 72.24%. However, the variational autoencoder underperformed on this dataset. For the Digits dataset, leaky ReLU activation (96.39%) and elastic net regularization (95.83%) demonstrated high accuracy, effectively capturing the digit patterns.

For the Iris dataset, tanh activation and batch normalization both achieved the highest accuracy at 99.09%, showcasing their ability to maintain

class separability. The Wine dataset saw multiple configurations achieving nearly perfect accuracy (99.9%), indicating its relative ease for dimensionality reduction and classification tasks. In the Breast Cancer dataset, elastic net regularization and residual connections provided the highest accuracy (98.25%), proving effective in distinguishing cancerous from non-cancerous samples.

The Musk 1 dataset had most configurations achieving the same high accuracy of 84.17%, suggesting that its intrinsic patterns are well captured by the autoencoder. For the MNIST dataset, dropout layers reached the highest accuracy of 92.2%, enhancing generalization in digit recognition. The Yeast dataset, challenging for dimensionality reduction, showed the highest accuracy with basic L1 regularization at 62.29%. The Abalone dataset had relatively low performance, with L2 regularization and leaky ReLU activation showing higher accuracies of 29.78% and 29.31%, respectively, due to its complexity. Lastly, the Olivetti Faces dataset achieved the highest accuracy with leaky ReLU activation at 68.75%, emphasizing the importance of activation functions in facial feature recognition.

Figure 14 presents a heatmap summarizing the accuracy performance of all Manifold Adjustment Box configurations across the evaluated datasets. The color intensity clearly highlights which combinations (e.g., batch normalization on Isolet, leaky ReLU on Olivetti Faces, elastic net on Wine and Breast Cancer) deliver consistently strong results and where performance drops noticeably (e.g., multiple hidden layers and residual connections on several datasets).

Regularization techniques such as elastic net regularization generally provided balanced performance across different datasets, highlighting its effectiveness in managing correlated features and ensuring robustness. Advanced activation functions like leaky ReLU and tanh significantly improved the model's ability to learn complex patterns, especially in image datasets like Digits and Olivetti Faces. Additionally, the inclusion of dropout layers enhanced generalization capabilities in most datasets, while batch normalization contributed to stabilizing the learning process and convergence, thereby improving overall performance.

The use of multiple hidden layers and residual connections showed varied results, improving performance in some cases but also leading to overfitting or underfitting in certain datasets, indicating the necessity for careful tuning. Variational autoencoders (VAEs), which provide probabilistic modelling, exhibited inconsistent performance across different datasets, suggesting that further tuning and enhancements are required for more reliable results. These findings underscore the importance of tailoring model configurations to the specific characteristics of each dataset to achieve optimal performance.

Table 3 presents a comparative analysis of five traditional dimensionality reduction techniques—PCA, t-SNE, LLE, MDS, and Isomap—applied to various datasets, including Isolet, Digits, Iris, Wine, Breast Cancer, Musk 1, MNIST, Yeast, Abalone, and Olivetti Faces. PCA consistently performs well across most datasets, particularly excelling in the Isolet and Olivetti Faces datasets, demonstrating its effectiveness in capturing the principal components of high-dimensional data. t-SNE, while performing exceptionally on datasets like Digits and MNIST due to its ability to preserve local structure, struggles with more

Table 3. Accuracy results of traditional dimensionality reduction techniques (PCA, t-SNE, LLE, MDS, and Isomap) across various datasets. The results highlight the strengths and limitations of each method in capturing the intrinsic structure of high-dimensional data and preserving it for classification tasks.

Dataset Name	PCA	t-SNE	LLE	MDS	Isomap
Isolet - scikit-learn	97.8	72.13	83.52	3.01	79.68
Digits - scikit-learn	96.22	15.83	95.83	8.06	96.11
Iris - scikit-learn	90	30	93.33	23.33	93.33
Wine - scikit-learn	72.22	33.33	72.22	38.89	72.22
Breast Cancer - scikit-learn	95.61	85.96	64.04	83.33	95.61
Musk 1 - scikit-learn	77.89	48.42	72.63	66.03	75.25
MNIST - tensorflow.keras.datasets	82.85	58.98	79.87	79.34	63.76
Yeast - scikit-learn	59.26	32.32	33.58	30.64	23.39
Abalone - scikit-learn	27.99	21.41	11.83	9.69	18.69
Olivetti Faces - scikit-learn	96.25	1.25	18.52	10.00	67.5

complex datasets like Isolet and Olivetti Faces, indicating sensitivity to intrinsic data patterns. LLE shows high accuracy in simpler datasets like Iris but falls short in more complex ones, such as Abalone, highlighting its limitations in preserving global data structures. MDS performs poorly overall, especially on high-dimensional datasets like Isolet, suggesting its inadequacy in capturing meaningful low-dimensional representations. Isomap, while performing moderately well, demonstrates variability across datasets, with notable success in simpler datasets like Iris but less so in complex ones like MNIST. This comparative analysis underscores the varying effectiveness of these techniques, revealing PCA's robustness and the limitations of other methods in handling diverse data complexities, thereby setting the stage for the need for more adaptive and integrated approaches like AMRAE.

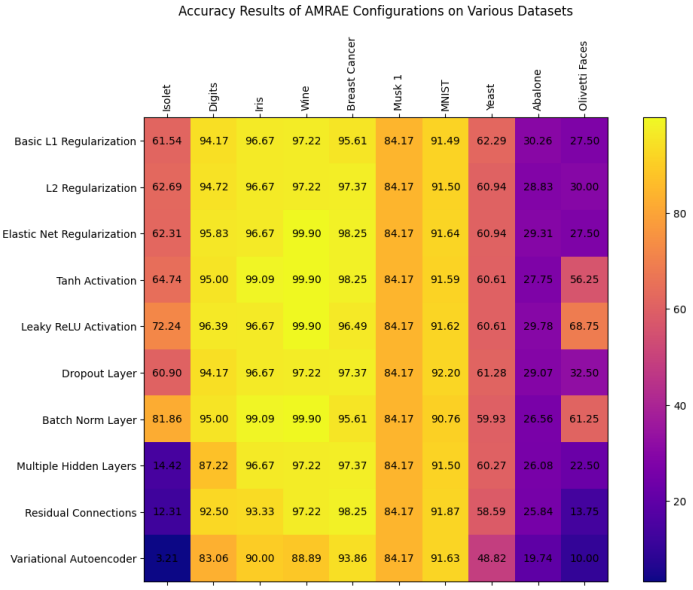

Figure 14. Heatmap of accuracy results for various configurations of the Manifold Adjustment Box within the AMRAE algorithm across multiple datasets. The color intensity represents the accuracy, with brighter colors indicating higher accuracy. The analysis highlights the strengths and limitations of each configuration, demonstrating the effectiveness of AMRAE in adapting to diverse data complexities and structures.

Figure 15 provides a visual representation of the accuracy results for traditional dimensionality reduction techniques—PCA, t-SNE, LLE, MDS, and Isomap—across various datasets, including Isolet, Digits, Iris, Wine, Breast Cancer, Musk 1, MNIST, Yeast, Abalone, and Olivetti Faces. Each cell's colour intensity represents the accuracy achieved by a specific technique on a particular dataset, with brighter colours indicating higher accuracy and darker

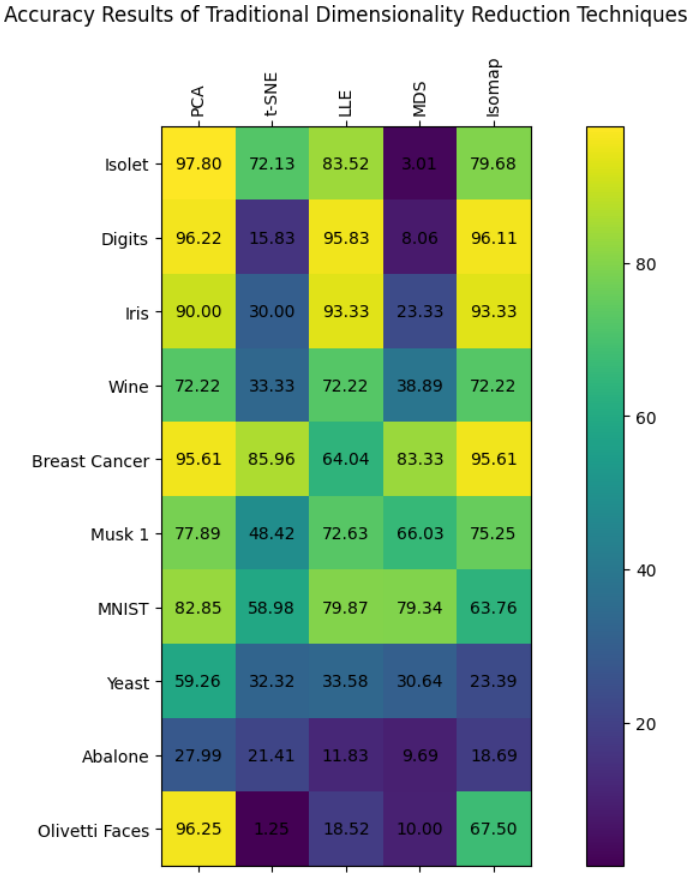

Figure 15. Heatmap of accuracy results for traditional dimensionality reduction techniques (PCA, t-SNE, LLE, MDS, and Isomap) across multiple datasets. The color intensity represents the accuracy, with brighter colors indicating higher accuracy. The analysis highlights the variability in performance across different techniques and datasets, showcasing the strengths and limitations of each method.

colours indicating lower accuracy.

Table 4 summarizes the properties of various datasets used in evaluating the AMRAE algorithm and traditional dimensionality reduction techniques, showcasing significant variations in dimensionality, intrinsic dimensionality, complexity, and mean anomaly scores, thus providing a comprehensive testing ground for the algorithms. Key observations include datasets with higher intrinsic dimensionality, like MNIST (331), which indicate a more complex structure requiring more features for accurate representation, and simpler datasets like Iris (2) requiring fewer features. Complexity, measured using the silhouette score, also varies, with the Iris dataset showing high complexity (0.5811) and distinct clusters, whereas the Digits dataset exhibits lower complexity (0.1087) and less distinct clustering, highlighting that higher complexity often correlates with more challenging dimensionality reduction tasks. Mean

Table 4. Properties of Various Datasets Used for Evaluating AMRAE and Traditional Dimensionality Reduction Techniques, Including Dimensionality, Intrinsic Dimensionality, Complexity, and Mean Anomaly Scores.

Dataset Name	Dimension	Intrinsic Dimensionality using Fisher Information Matrix	Complexity (Using silhouette scoreas a proxy for complexity)	Mean Anomaly Score using Isolation Forest
Isolet - scikit-learn	617	202	0.1438	0.0445
Digits - scikit-learn	64	40	0.1087	0.0223
Iris - scikit-learn	4	2	0.5811	0.0202
Wine - scikit-learn	13	10	0.2649	0.0543
Breast cancer - scikit-learn	30	10	0.3431	0.0792
Musk 1 - scikit-learn	166	55	0.2536	0.0528
MNIST - tensorflow.keras.datasets	784	331	0.1276	0.0439
Yeast - scikit-learn	8	7	0.1996	0.0956
Abalone - scikit-learn	8	4	0.4031	0.0209

anomaly scores, derived from the Isolation Forest, provide insights into the prevalence of outliers, with lower scores suggesting fewer anomalies as seen in the Digits dataset (0.0223) and higher scores indicating more anomalies, such as in the Yeast dataset (0.0956), which is crucial for understanding the robustness of DR techniques in identifying and handling outliers. The implications for AMRAE and traditional DR algorithms are significant: the adaptability of the AMRAE algorithm, which leverages manifold learning and autoencoders to capture complex structures, allows it to handle a variety of datasets effectively, with manifold regularization ensuring the preservation of intrinsic data structures, making AMRAE particularly effective on datasets with high intrinsic dimensionality and complexity. In contrast, traditional DR techniques like PCA and Isomap perform well on datasets with lower complexity and intrinsic dimensionality, whereas nonlinear techniques such as t-SNE and LLE may struggle with high-dimensional data or data with numerous outliers.

The AMRAE algorithm demonstrated robust performance across a variety of datasets, with specific configurations of the Manifold Adjustment Box yielding significant improvements. By integrating manifold learning principles and regularization techniques within the autoencoder framework, the approach provided a powerful method for dimensionality reduction, effectively addressing the limitations of traditional techniques. This combination of advanced methods and thoughtful configuration underscores the potential of the AMRAE algorithm in various applications, offering valuable insights into model optimization and performance enhancement.

6 Discussion

The AMRAE algorithm's performance was evaluated across various datasets using different configurations within the Manifold Adjustment Box. Each adjustment was analysed in terms of its impact on the accuracy and robustness of the dimensionality reduction process. Basic L1 regularization showed consistent performance improvement by introducing sparsity in the feature space, particularly beneficial for high-dimensional datasets like Isolet and MNIST. L2 regularization provided smoother representations by penalizing large weights, reducing overfitting, and maintaining stable performance across datasets. Elastic net regularization balanced the benefits of L1 and L2, proving effective for datasets with correlated features, such as Breast Cancer and Musk 1, showing improved accuracy and robustness.

In terms of activation functions, tanh activation helped capture complex patterns in datasets like Iris and Wine, leading to near-perfect accuracy due to its smooth nonlinear transformation capabilities. Leaky ReLU activation allowed small gradients when units were inactive, significantly improving learning capabilities in image datasets like Digits and Olivetti Faces. Architectural enhancements such as dropout layers prevented over-reliance on specific features, enhancing generalization, especially in MNIST and Yeast datasets. Batch normalization stabilized learning and improved convergence, contributing to higher accuracy in datasets with varied complexities, such as Isolet and Olivetti Faces.

Multiple hidden layers captured complex hierarchical patterns but required careful tuning to avoid overfitting, particularly in datasets like Digits and MNIST. Residual connections mitigated the vanishing gradient problem, allowing for deeper networks, which was beneficial for complex datasets like Breast Cancer and Musk 1. Variational autoencoders (VAEs) provided probabilistic modelling but showed inconsistent performance, indicating a need for further tuning for reliable results across datasets. Laplacian regularization preserved local data structures effectively, enhancing the interpretability

and robustness of the learned representations.

Comparatively, traditional dimensionality reduction techniques like PCA, t-SNE, LLE, MDS, and Isomap were also evaluated. PCA showed robust performance across most datasets, excelling in simpler and well-structured datasets like Isolet and Olivetti Faces. t-SNE and LLE, while effective for certain datasets like Digits and Iris, struggled with scalability and noise sensitivity in more complex datasets. MDS performed poorly overall, particularly with high-dimensional data, indicating its limitations in preserving meaningful low-dimensional representations. Isomap demonstrated moderate performance but varied significantly across datasets, highlighting its sensitivity to dataset complexity and structure. Overall, AMRAE's integration of manifold concept with autoencoders, coupled with various regularization techniques and architectural enhancements, provided superior performance and robustness compared to traditional methods. The flexibility and adaptability of the Manifold Adjustment Box allowed for tailored configurations that significantly improved accuracy and stability across a wide range of datasets, demonstrating the algorithm's potential for effective dimensionality reduction in diverse applications.

Throughout our experiments with the AMRAE algorithm, certain adjustments within the Manifold Adjustment Box demonstrated particularly significant improvements in performance. For instance, the Leaky ReLU activation function consistently enhanced accuracy across various datasets, with notable performance in the Digits and Olivetti Faces datasets. Batch normalization also proved beneficial, especially for the Isolet dataset, where it achieved the highest accuracy. These findings underscore the importance of choosing appropriate activation functions and normalization techniques to capture complex data structures effectively.

Despite the promising results, the current study has several limitations. Firstly, the performance of the AMRAE algorithm varies significantly across different datasets, indicating a need for more adaptive or dataset-specific tuning mechanisms. Additionally, some configurations, such as the Variational Autoencoder, did not perform consistently well, suggesting that further refinement and parameter tuning are necessary. The computational complexity and training time of the algorithm also pose challenges, particularly for larger datasets. Moreover, the study primarily focuses on classification tasks, and the applicability of AMRAE to other types of data analysis tasks, such as regression or clustering, remains to be explored.

Future research could address these limitations by exploring several directions. One area for further investigation is the development of more adaptive techniques within the Manifold Adjustment Box that can dynamically adjust parameters based on the dataset's characteristics. Enhancements to the Variational Autoencoder component, possibly through advanced optimization techniques or better regularization methods, could improve its performance. Additionally, extending the evaluation to include regression and clustering tasks would provide a more comprehensive assessment of AMRAE's capabilities. Exploring the application of AMRAE to real-world datasets, such as those in genomics, finance, or social network analysis, could also uncover new insights and potential improvements. Lastly, incorporating more sophisticated anomaly detection mechanisms could enhance the robustness of the algorithm in identifying and handling outliers.

7 Conclusion

AMRAE algorithm represents a significant advancement in dimensionality reduction by integrating the strengths of manifold concept and autoencoders to effectively capture the intrinsic geometry of high-dimensional data, addressing the limitations of traditional techniques. Extensive experiments across diverse datasets demonstrate AMRAE's robustness and flexibility, particularly in handling complex and nonlinear structures, with configurations like Leaky ReLU activation and batch normalization consistently improving accuracy. Comparative analysis underscores AMRAE's advantages over conventional methods like PCA, t-SNE, and LLE, showcasing its superior performance in maintaining data structure and enhancing classification outcomes. Tailoring model configurations to specific dataset characteristics ensures optimal performance, while the algorithm's scalability, robustness to noise, and enhanced interpretability make it valuable for real-world high-dimensional data analysis applications. These findings open avenues for further research, suggesting potential enhancements and the exploration of additional datasets, with future work involving refining regularization techniques and expanding manifold learning components. AMRAE offers a

powerful and versatile solution for dimensionality reduction with broad potential across various fields, highlighting the need for continued exploration and development in this promising area of research.

Data Availability Statement

Data will be made available on request.

Funding

This work was supported without any funding.

Conflicts of Interest

The authors declare no conflicts of interest.

AI Use Statement

The authors declare that no generative AI was used in the preparation of this manuscript.

Ethical Approval and Consent to Participate

Not applicable.

References

[1] Wang, Z., Zhang, G., Xing, X., Xu, X., & Sun, T. (2024). Comparison of dimensionality reduction techniques for multi-variable spatiotemporal flow fields. *Ocean Engineering, 291*, 116421. [CrossRef]

[2] Ray, P., Reddy, S. S., & Banerjee, T. (2021). Various dimension reduction techniques for high dimensional data analysis: a review. Artificial Intelligence Review, 54(5), 3473-3515. [CrossRef]

[3] Jiang, X., Kong, X., & Ge, Z. (2023). Augmented industrial data-driven modeling under the curse of dimensionality. *IEEE/CAA Journal of Automatica Sinica, 10*(6), 1445-1461. [CrossRef]

[4] Gunsilius, F., & Schennach, S. (2023). Independent nonlinear component analysis. *Journal of the American Statistical Association, 118*(542), 1305-1318. [CrossRef]

[5] Saccenti, E. (2024). A gentle introduction to principal component analysis using tea-pots, dinosaurs, and pizza. *Teaching Statistics, 46*(1), 38-52. [CrossRef]

[6] Abdi, H., & Williams, L. J. (2010). Principal component analysis. *Wiley interdisciplinary reviews: computational statistics, 2*(4), 433-459. [CrossRef]

[7] Gewers, F. L., Ferreira, G. R., Arruda, H. F. D., Silva, F. N., Comin, C. H., Amancio, D. R., & Costa, L. D. F. (2021). Principal component analysis: A natural approach to data exploration. *ACM Computing Surveys (CSUR), 54*(4), 1-34. [CrossRef]

[8] Turaga, P., Anirudh, R., & Chellappa, R. (2020). Manifold learning. In *Computer Vision: A Reference Guide* (pp. 1-6). Cham: Springer International Publishing. [CrossRef]

[9] Lunga, D., Prasad, S., Crawford, M. M., & Ersoy, O. (2013). Manifold-learning-based feature extraction for classification of hyperspectral data: A review of advances in manifold learning. *IEEE Signal Processing Magazine, 31*(1), 55-66. [CrossRef]

[10] Lin, T., & Zha, H. (2008). Riemannian manifold learning. *IEEE transactions on pattern analysis and machine intelligence, 30*(5), 796-809. [CrossRef]

[11] Pinaya, W. H. L., Vieira, S., Garcia-Dias, R., & Mechelli, A. (2020). Autoencoders. In *Machine learning* (pp. 193-208). Academic Press. [CrossRef]

[12] Pawar, K., & Attar, V. Z. (2019). Assessment of autoencoder architectures for data representation. In *Deep learning: concepts and architectures* (pp. 101-132). Cham: Springer International Publishing. [CrossRef]

[13] Le, L., Patterson, A., & White, M. (2018, December). Supervised autoencoders: improving generalization performance with unsupervised regularizers. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems* (pp. 107-117). [CrossRef]

[14] Cunningham, J. P., & Ghahramani, Z. (2015). Linear dimensionality reduction: Survey, insights, and generalizations. *The Journal of Machine Learning Research, 16*(1), 2859-2900. [CrossRef]

[15] Tharwat, A., Gaber, T., Ibrahim, A., & Hassanien, A. E. (2017). Linear discriminant analysis: A detailed tutorial. *AI communications, 30*(2), 169-190. [CrossRef]

[16] Arulananth, T. S., Balaji, L., Baskar, M., Anbarasu, V., & Rao, K. S. (2023). PCA based dimensional data reduction and segmentation for DICOM images. *Neural Processing Letters, 55*(1), 3-17. [CrossRef]

[17] Pani, A. K. (2022). Non-linear process monitoring using kernel principal component analysis: A review of the basic and modified techniques with industrial applications. *Brazilian Journal of Chemical Engineering, 39*(2), 327-344. [CrossRef]

[18] Reddy, G. T., Reddy, M. P. K., Lakshmanna, K., Kaluri, R., Rajput, D. S., Srivastava, G., & Baker, T. (2020). Analysis of dimensionality reduction techniques on big data. *IEEE Access, 8*, 54776-54788. [CrossRef]

[19] Tang, E. K., Suganthan, P. N., Yao, X., & Qin, A. K. (2005). Linear dimensionality reduction using relevance weighted LDA. *Pattern recognition, 38*(4), 485-493. [CrossRef]

[20] Hayati, R., Munawar, A. A., Lukitaningsih, E., Earlia, N., Karma, T., & Idroes, R. (2024). Combination of PCA with LDA and SVM classifiers: A model for determining the geographical origin of coconut in the coastal plantation, Aceh Province, Indonesia. *Case Studies in Chemical and Environmental Engineering, 9*, 100552. [CrossRef]

[21] Anowar, F., Sadaoui, S., & Selim, B. (2021). Conceptual

and empirical comparison of dimensionality reduction algorithms (PCA, KPCA, LDA, MDS, SVD, LLE, ISOMAP, LE, ICA, T-SNE). *Computer Science Review, 40*, 100378. [CrossRef]

[22] Maaten, L. V. D., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of machine learning research, 9*(Nov), 2579-2605.

[23] Shi, S., Xu, Y., Xu, X., Mo, X., & Ding, J. (2023). A preprocessing manifold learning strategy based on T-distributed stochastic neighbor embedding. *Entropy, 25*(7), 1065. [CrossRef]

[24] Liu, X., Bao, Y., Zhao, L., & Gu, C. (2024). Establishment and application of steel composition prediction model based on t-distributed stochastic neighbor embedding (t-SNE) dimensionality reduction algorithm. *Journal of Sustainable Metallurgy, 10*(2), 509–524. [CrossRef]

[25] Ghojogh, B., Ghodsi, A., Karray, F., & Crowley, M. (2020). Stochastic neighbor embedding with Gaussian and student-t distributions: Tutorial and survey. *arXiv preprint arXiv:2009.10301*.

[26] Ma, X., Li, D., Zhang, J., Sheng, T., Yang, Z., Nai, W., & Sun, Y. (2020, December). t-SNE with high order truncation fractional gradient descent method. In *2020 IEEE 9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)* (Vol. 9, pp. 1929-1933). IEEE. [CrossRef]

[27] Sun, Y., Li, D., Hua, Z., Zhu, S., Yang, Z., Nai, W., & Xing, Y. (2021, March). t-SNE based on fixed memory step gradient descent method. In *2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)* (pp. 2486-2489). IEEE. [CrossRef]

[28] Roweis, S. T., & Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science, 290*(5500), 2323–2326. [CrossRef]

[29] Chen, J., & Liu, Y. (2011). Locally linear embedding: A survey. *Artificial Intelligence Review, 36*, 29–48. [CrossRef]

[30] Liu, X. F., Zheng, X. D., Xu, G. C., Wang, L., & Yang, H. (2010). Locally linear embedding-based seismic attribute extraction and applications. *Applied Geophysics, 7*(4), 365-375. [CrossRef]

[31] Hou, Y., Zhang, P., Xu, X., Zhang, X., & Li, W. (2009). Nonlinear dimensionality reduction by locally linear inlaying. *IEEE transactions on neural networks, 20*(2), 300-315. [CrossRef]

[32] Tenenbaum, J. B., de Silva, V., & Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science, 290*(5500), 2319–2323. [CrossRef]

[33] Lee, J. A., Lendasse, A., & Verleysen, M. (2002). Curvilinear distance analysis versus Isomap. In *ESANN* (pp. 185–192).

[34] Lee, J. A., Lendasse, A., & Verleysen, M. (2004). Nonlinear projection with curvilinear distances: Isomap versus curvilinear distance analysis. *Neurocomputing, 57*, 49–76. [CrossRef]

[35] Donoho, D. L., & Grimes, C. (2005). Image manifolds which are isometric to Euclidean space. *Journal of Mathematical Imaging and Vision, 23*(1), 5–24. [CrossRef]

[36] McInnes, L., Healy, J., & Melville, J. (2018). Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*.

[37] Ghojogh, B., Crowley, M., Karray, F., & Ghodsi, A. (2023). Uniform manifold approximation and projection (UMAP). In *Elements of dimensionality reduction and manifold learning* (pp. 479-497). Cham: Springer International Publishing. [CrossRef]

[38] Yu, T. T., Chen, C. Y., Wu, T. H., & Chang, Y. C. (2023). Application of high-dimensional uniform manifold approximation and projection (UMAP) to cluster existing landfills on the basis of geographical and environmental features. *Science of The Total Environment, 904*, 167013. [CrossRef]

[39] Lim, H. S., & Qiu, P. (2023). Quantifying cell-type-specific differences of single-cell datasets using uniform manifold approximation and projection for dimension reduction and shapley additive exPlanations. *Journal of Computational Biology, 30*(7), 738–750. [CrossRef]

[40] Yu, S., & Principe, J. C. (2019). Understanding autoencoders with information theoretic concepts. *Neural Networks, 117*, 104–123. [CrossRef]

[41] Asperti, A., & Trentin, M. (2020). Balancing reconstruction error and Kullback-Leibler divergence in variational autoencoders. *IEEE Access, 8*, 199440–199448. [CrossRef]

[42] Wang, W., Huang, Y., Wang, Y., & Wang, L. (2014). Generalized autoencoder: A neural network framework for dimensionality reduction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (pp. 490–497). [CrossRef]

[43] Nusrat, I., & Jang, S.-B. (2018). A comparison of regularization techniques in deep neural networks. *Symmetry, 10*(11), 648. [CrossRef]

[44] Yu, W., Zeng, G., Luo, P., Zhuang, F., He, Q., & Shi, Z. (2013). Embedding with autoencoder regularization. In *Machine Learning and Knowledge Discovery in Databases* (pp. 208–223). Springer. [CrossRef]

[45] Meilă, M., & Zhang, H. (2024). Manifold learning: What, how, and why. *Annual Review of Statistics and Its Application, 11*(1), 393-417. [CrossRef]

[46] Muscoloni, A., Thomas, J. M., Ciucci, S., Bianconi, G., & Cannistraci, C. V. (2017). Machine learning meets complex networks via coalescent embedding in the hyperbolic space. *Nature Communications, 8*(1), 1615. [CrossRef]

[47] Izenman, A. J. (2012). Introduction to manifold learning. W*iley Interdisciplinary Reviews: Computational Statistics*, *4*(5), 439-446. [CrossRef]

[48] Talwalkar, A., Kumar, S., & Rowley, H. (2008). Large-scale manifold learning. In *2008 IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1–8). IEEE. [CrossRef]

[49] Zhu, Y., Wu, X., Li, P., Zhang, Y., & Hu, X. (2019). Transfer learning with deep manifold regularized auto-encoders. *Neurocomputing, 369*, 145–154. [CrossRef]

[50] Duque, A. F., Morin, S., Wolf, G., & Moon, K. (2020). Extendable and invertible manifold learning with geometry regularized autoencoders. In *2020 IEEE International Conference on Big Data* (pp. 5027–5036). IEEE. [CrossRef]

[51] Li, J., Wei, S., & Dai, W. (2021). Combination of manifold learning and deep learning algorithms for mid-term electrical load forecasting. *IEEE Transactions on Neural Networks and Learning Systems, 34*(5), 2584–2593. [CrossRef]

[52] Van Der Maaten, L., Postma, E. O., & Van Den Herik, H. J. (2009). Dimensionality reduction: A comparative review. *Journal of Machine Learning Research, 10*(66–71), 13.

[53] Wang, F., & Sun, J. (2015). Survey on distance metric learning and dimensionality reduction in data mining. *Data Mining and Knowledge Discovery, 29*(2), 534–564. [CrossRef]

[54] Vogt, J. E., & Roth, V. (2010). The group-lasso: $\ell 1,\infty$ regularization versus $\ell 1$, 2 regularization. In *Joint Pattern Recognition Symposium* (pp. 252–261). Springer. [CrossRef]

[55] Cortes, C., Mohri, M., & Rostamizadeh, A. (2009, June). L 2 regularization for learning kernels. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence* (pp. 109-116). [CrossRef]

[56] Li, C. N., Shang, M. Q., Shao, Y. H., Xu, Y., Liu, L. M., & Wang, Z. (2019). Sparse L1-norm two dimensional linear discriminant analysis via the generalized elastic net regularization. *Neurocomputing, 337*, 80-96. [CrossRef]

[57] Khan, S. H., Hayat, M., & Porikli, F. (2019). Regularization of deep neural networks with spectral dropout. *Neural Networks, 110*, 82-90. [CrossRef]

[58] Santurkar, S., Tsipras, D., Ilyas, A., & Mądry, A. (2018, December). How does batch normalization help optimization?. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems* (pp. 2488-2498). [CrossRef]

[59] Baldi, P. (2012, June). Autoencoders, unsupervised learning, and deep architectures. In *Proceedings of ICML workshop on unsupervised and transfer learning* (pp. 37-49). JMLR Workshop and Conference Proceedings. [CrossRef]

[60] Zhang, Y. (2018, March). A better autoencoder for image: Convolutional autoencoder. In *ICONIP17-DCEC* (accessed on 23 March 2017) (p. 34).

[61] Sewak, M., Sahay, S. K., & Rathore, H. (2020). An overview of deep learning architecture of deep neural networks and autoencoders. *Journal of Computational and Theoretical Nanoscience, 17*(1), 182–188. [CrossRef]

[62] Khare, N., Thakur, P. S., Khanna, P., & Ojha, A. (2021, December). Analysis of loss functions for image reconstruction using convolutional autoencoder. In *International Conference on Computer Vision and Image Processing* (pp. 338-349). Cham: Springer International Publishing. [CrossRef]

[63] Zhu, B., Liu, J. Z., Cauley, S. F., Rosen, B. R., & Rosen, M. S. (2018). Image reconstruction by domain-transform manifold learning. *Nature, 555*(7697), 487-492. [CrossRef]

[64] Braunsmann, J., Rajković, M., Rumpf, M., & Wirth, B. (2024). Convergent autoencoder approximation of low bending and low distortion manifold embeddings. *ESAIM: Mathematical Modelling and Numerical Analysis, 58*(1), 335–361. [CrossRef]

[65] Dogo, E. M., Afolabi, O. J., Nwulu, N. I., Twala, B., & Aigbavboa, C. O. (2018, December). A comparative analysis of gradient descent-based optimization algorithms on convolutional neural networks. In *2018 international conference on computational techniques, electronics and mechanical systems* (*CTEMS*) (pp. 92-99). IEEE. [CrossRef]

[66] Bjerrum, E. J., & Sattarov, B. (2018). Improving chemical autoencoder latent space and molecular de novo generation diversity with heteroencoders. *Biomolecules, 8*(4), 131. [CrossRef]

[67] Tov, O., Alaluf, Y., Nitzan, Y., Patashnik, O., & Cohen-Or, D. (2021). Designing an encoder for stylegan image manipulation. *ACM Transactions on Graphics, 40*(4), 1–14. [CrossRef]

[68] Brodersen, K. H., Ong, C. S., Stephan, K. E., & Buhmann, J. M. (2010, August). The balanced accuracy and its posterior distribution. In *2010 20th international conference on pattern recognition* (pp. 3121-3124). IEEE. [CrossRef]

[69] Shahapure, K. R., & Nicholas, C. (2020, October). Cluster quality analysis using silhouette score. In 2020 IEEE 7th international conference on data science and advanced analytics (DSAA) (pp. 747-748). IEEE. [CrossRef]

[70] Fukunaga, K., & Olsen, D. R. (1971). An algorithm for finding intrinsic dimensionality of data. *IEEE Transactions on Computers, 100*(2), 176–183. [CrossRef]

[71] Pang, G., Shen, C., & Van Den Hengel, A. (2019). Deep anomaly detection with deviation networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (pp. 353–362). ACM. [CrossRef]