Manifold Learning and Neural Networks for Dimensionality Reduction and Reconstruction of Stock Time-Series Data

Akshaj Gupta
Under the guidance of Dr Trilok Mathur

November 28, 2024

Contents

1 Introduction

Stock market data is high-dimensional, with features like prices, volumes, and technical indicators, making its storage and analysis computationally expensive. To address this, dimensionality reduction methods are applied to represent the data in lower dimensions while retaining critical patterns for efficient processing.

The real challenge lies in achieving age-invariant facial recognition. However, due to computational limitations, we used stock price data as a test case to replicate the problem. We employed manifold learning techniques like t-SNE, Isomap, and Locally Linear Embedding (LLE) to reduce the dimensionality of the stock data and capture its essential patterns. After dimensionality reduction, we used a Multi-Layer Perceptron (MLP) and Convolutional Neural Network (CNN) models for reconstruction. These models were trained to reconstruct the original data from its reduced representation, allowing us to assess how well the reduced data could maintain the critical features of the original time-series data.

2 Data Preparation

To analyze stock market data effectively, we constructed a dataset consisting of 20 financial indicators commonly used in market analysis. These indicators include price-related metrics, statistical measures, and technical indicators, providing a comprehensive view of market behavior. The list of features is as follows:

- Open, High, Low, Close, Volume
- Daily Return
- Simple Moving Averages (SMA_10, SMA_20)
- Exponential Moving Averages (EMA_10, EMA_20)
- Volatility (Volatility_10, Volatility_20)
- Relative Strength Index (RSI)
- Bollinger Bands (Upper Band, Lower Band)
- Moving Average Convergence Divergence (MACD)
- Momentum, Rate of Change (ROC), On-Balance Volume (OBV)
- Log Return

The overarching goal of this project was inspired by the challenge of time-invariant facial recognition. In facial recognition, the task involves analyzing sequences of facial data that evolve over time while maintaining local and global interactions between features.

To simulate this, we approached stock time-series data as an analogous system, where each matrix of financial indicators represents a "snapshot" of the evolving data. By grouping the indicators into 20-day windows, we created structured matrices that can interact and evolve, mimicking the local and temporal dynamics observed in time-variant data like human faces.

To prepare the data, we computed these indicators using the following formulas and methods:

- Daily Return: Daily Return $= \frac{\text{Close}_t - \text{Close}_{t-1}}{\text{Close}_{t-1}} \times 100$
- RSI: Computed using the rolling averages of gains and losses over 14 periods:

$$\text{RSI} = 100 - \left(\frac{100}{1 + \frac{\text{Avg Gain}}{\text{Avg Loss}}} \right)$$

- Bollinger Bands:

$$\text{Upper Band} = \text{SMA}_{20} + 2 \times \sigma_{20}, \quad \text{Lower Band} = \text{SMA}_{20} - 2 \times \sigma_{20}$$

- MACD: The difference between two EMAs with spans of 10 and 20 periods:

$$\text{MACD} = \text{EMA}_{10} - \text{EMA}_{20}$$

The data was then normalized using a standard scaler to ensure uniformity across all features. After normalization, we grouped the data into 20-day windows to form a 3D tensor with dimensions $(1220, 20, 20)$. Each 20-day window can be interpreted as a time-evolving "snapshot," analogous to analyzing facial images over time.

An excerpt of the normalized dataset is shown below:

Open	High	Low	Close	Volume	Daily Return	SMA_10	SMA_20	EMA_10	EMA_20
-2.257	-2.261	-2.243	-2.234	-0.348	0.573	-2.316	-2.306	-2.297	-2.298
-2.230	-2.219	-2.220	-2.199	-0.054	1.037	-2.295	-2.302	-2.277	-2.287
-2.192	-2.163	-2.169	-2.160	0.291	1.127	-2.271	-2.294	-2.255	-2.273
-2.151	-2.148	-2.126	-2.128	-0.143	0.900	-2.244	-2.286	-2.230	-2.257
-2.106	-2.131	-2.131	-2.154	1.200	-0.799	-2.222	-2.277	-2.215	-2.245

Table 1: Normalized Dataset (Part 1)

Vol_10	Vol_20	RSI	Upper	Lower	MACD	Momentum	ROC	OBV	Log Return
-1.464	0.952	0.410	-2.327	-2.251	-0.030	0.523	1.540	-1.996	0.579
-1.411	0.948	1.605	-2.309	-2.261	0.080	0.441	1.817	-1.958	1.034
-1.360	0.970	1.698	-2.282	-2.275	0.202	0.528	1.606	-1.909	1.121
-1.566	0.935	1.729	-2.253	-2.291	0.325	0.659	1.691	-1.873	0.899
-0.839	0.580	2.103	-2.231	-2.296	0.375	0.398	1.212	-1.949	-0.793

Table 2: Normalized Dataset (Part 2)

The final 3D representation of the dataset had a shape of $(1220, 20, 20)$, providing a structured format suitable for further analysis using manifold learning and neural networks.

3 Manifold Learning

We explore three dimensionality reduction techniques: t-SNE, Locally Linear Embedding (LLE), and Isomap, to reduce the high-dimensional stock market data while preserving the intrinsic structure. Each method is evaluated using Silhouette Scores and Trustworthiness.

3.1 Dimensionality Reduction Techniques

- **t-SNE (t-Distributed Stochastic Neighbor Embedding)**: t-SNE is a nonlinear technique that minimizes the divergence between two distributions, preserving local structure in the data. However, it fails to capture global structures, making it less suitable for time-series data like stock market trends.

- **LLE (Locally Linear Embedding)**: LLE focuses on local linearity by reconstructing each point using its neighbors. It works well for datasets with local geometries but struggles with global structures, which is crucial for understanding stock market dynamics over time.

- **Isomap**: Isomap extends MDS (Multidimensional Scaling) by using geodesic distances instead of Euclidean distances, allowing it to preserve the global structure of the data. This ability to capture global non-linear relationships makes Isomap more suitable for stock market data, where data points are often related in complex, non-Euclidean ways.

From the results of the Silhouette Score and Trustworthiness:

- Silhouette Scores: t-SNE = 0.3519, Isomap = 0.4712, LLE = 0.5901
- Trustworthiness: t-SNE = 0.9902, Isomap = 0.9041, LLE = 0.7515

Isomap outperforms both t-SNE and LLE in preserving the global structure of the data, making it the best choice for dimensionality reduction in this context.

3.2 Isomap: Mathematical Framework and Advantages

Isomap works by computing geodesic distances between data points, which are the shortest paths along a manifold, instead of the straight-line Euclidean distances. The steps involved in Isomap are:

1. **Graph Construction**: A neighborhood graph is created using the k nearest neighbors or fixed-radius search.
2. **Geodesic Distance Calculation**: Geodesic distances are computed using algorithms such as Dijkstra's algorithm for the graph.

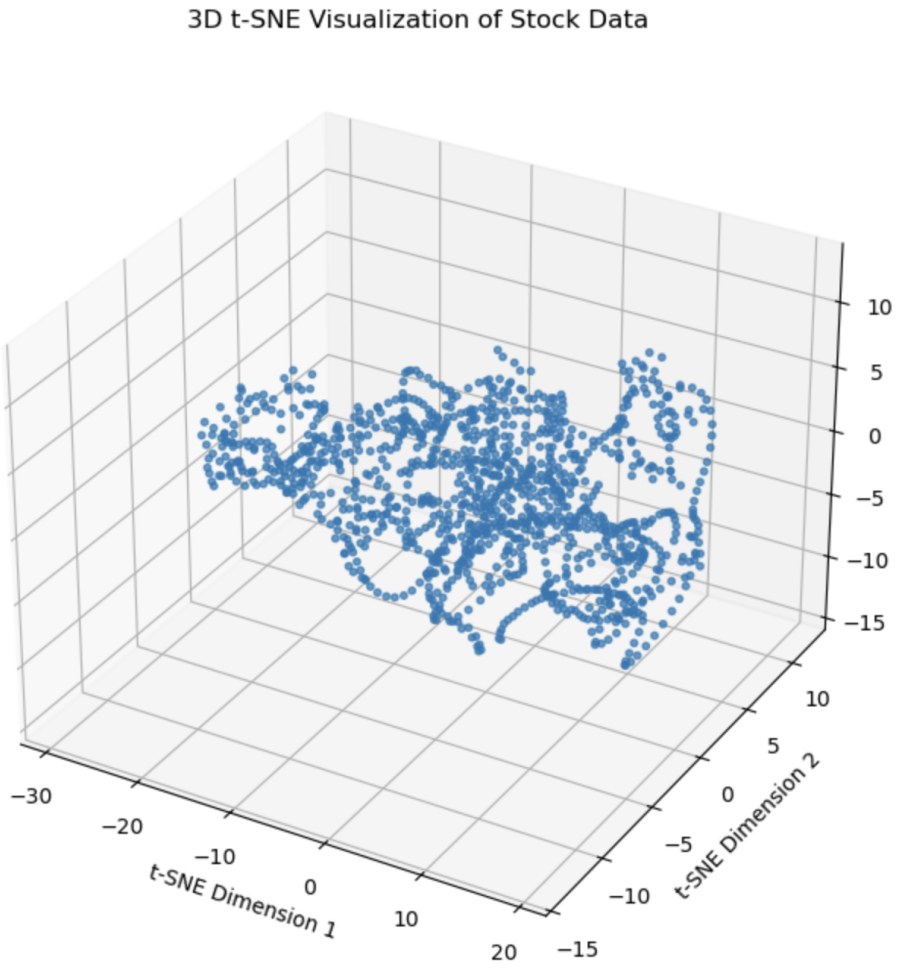

Figure 1: *t-SNE visualization of stock market data.*

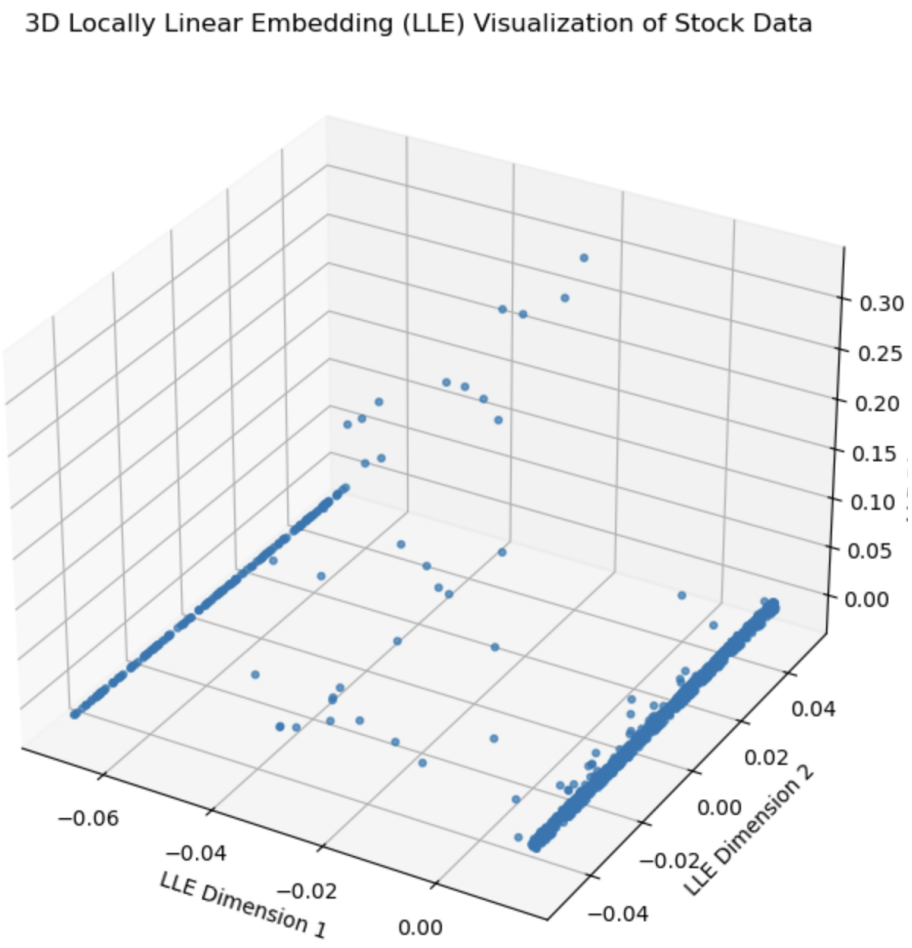

Figure 2: *LLE visualization of stock market data.*

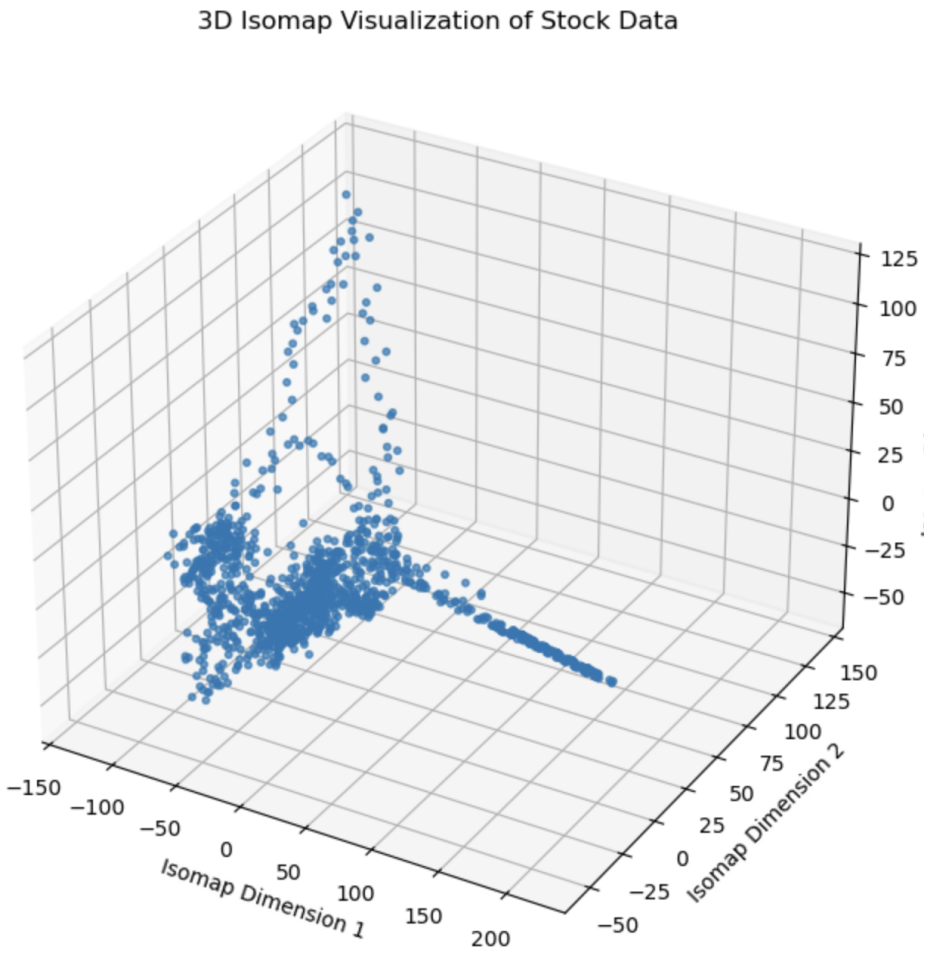

Figure 3: *Isomap visualization of stock market data.*

3. **Embedding**: Classical Multidimensional Scaling (MDS) is applied to the geodesic distance matrix to find a lower-dimensional representation of the data.

The loss function for Isomap aims to preserve these distances in the lower-dimensional space:

$$\text{Stress} = \sum_{i,j} \left(\|\mathbf{x}_i - \mathbf{x}_j\|^2 - D_{ij}^2 \right)$$

where D_{ij} is the geodesic distance between points i and j, and $\|\mathbf{x}_i - \mathbf{x}_j\|$ is the Euclidean distance in the lower-dimensional space.

Why Isomap is Better: Unlike t-SNE and LLE, Isomap preserves both local and global structures by considering geodesic distances. This makes it particularly suitable for time-series data like stock market prices, where global patterns and trends are crucial. It is also effective for facial recognition, where non-linear relationships between facial features must be captured accurately over time.

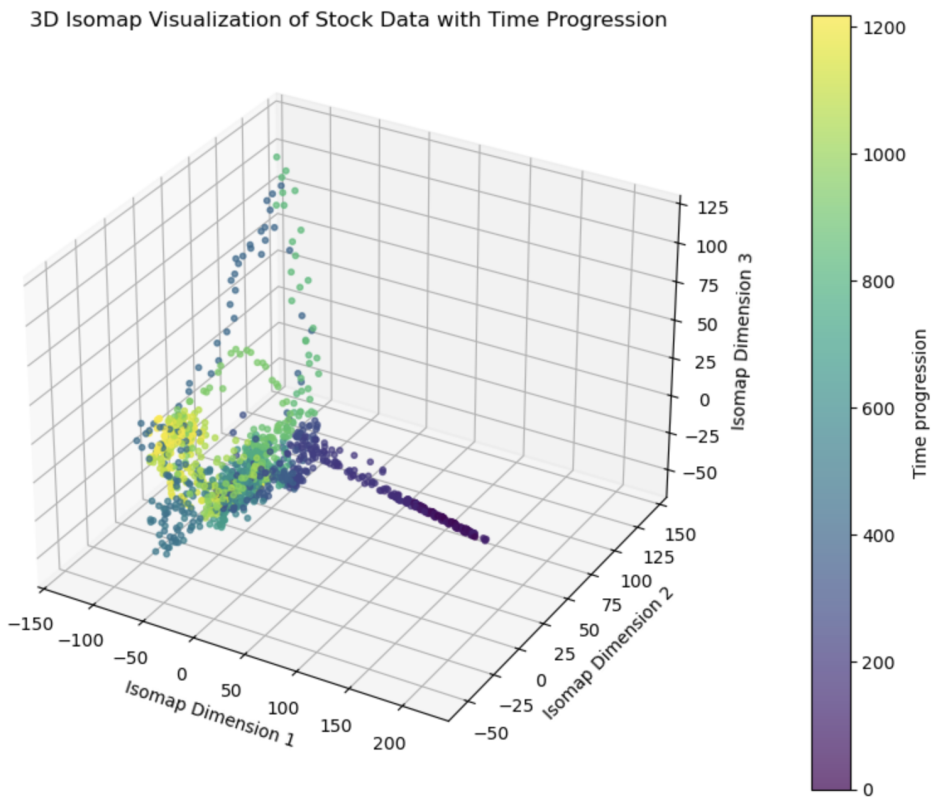

Figure 4: *Isomap with time-stamped data, illustrating stock market flow over time.*

3.3 Isomap with Time Stamps

The timeline below shows how Isomap maintains the clustering of stock market data points over time. This ability to capture the temporal relationships in the data provides insights into how market behaviors evolve.

3.4 Persistence Homology in Time-Series

Persistence homology is a technique from topological data analysis that identifies topological features such as connected components and loops that persist across different scales. This method is particularly useful in time-series data to analyze how topological features evolve over time.

In stock market data, persistence homology can reveal how market states transition and persist, giving valuable insights into market trends and volatility. Using delay persistence, we can track how the topology changes over time and identify stable patterns in the data.

Below is a persistence diagram illustrating the evolution of topological features in the stock market data.

By combining Isomap with persistence homology, we can gain a deeper understanding of how topological features in stock market data change over time, providing a more robust foundation for forecasting and analysis.

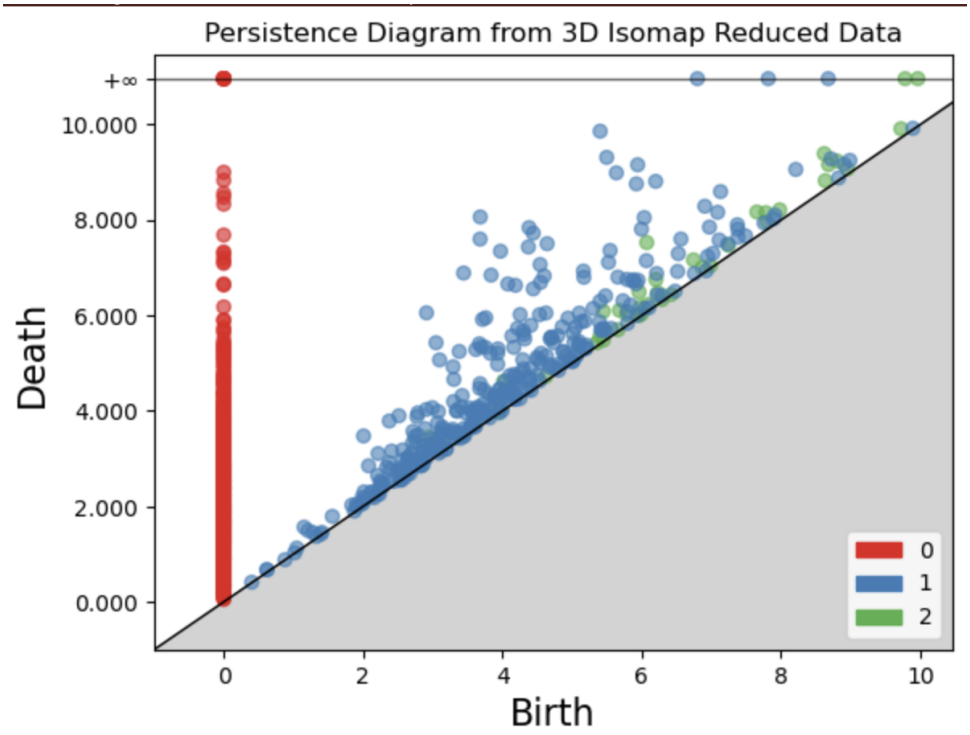

Figure 5: *Persistence homology diagram for stock market time-series data.*

4 Reconstruction

In this section, we explore reconstruction techniques using Multi-Layer Perceptron (MLP) and Convolutional Neural Networks (CNN) to map the reduced 3D representations of the data back to the original 400D flattened data. The models are trained to minimize the Mean Squared Error (MSE) between the reconstructed and original data, while leveraging the dimensionality reduction obtained using Isomap.

4.1 MLP Reconstruction Architecture

The Multi-Layer Perceptron (MLP) model consists of several fully connected layers that transform the 3D data (from Isomap) to a 400D flattened vector, which matches the original input dimension. The architecture of the MLP model is as follows:

- Input: A 3D reduced representation (3D vector from Isomap).
- Encoder: Two fully connected layers with ReLU activations to expand the dimensions progressively.
- Decoder: A fully connected layer that maps the hidden representation to the original 400D space.
- Loss function: Mean Squared Error (MSE) between the original and reconstructed data.

Model	Loss (Final Epoch)	MSE	MAE	R^2
MLP	0.2483	0.2423	0.2967	0.7569
CNN	0.2154	0.2118	0.2692	0.7893
Autoencoder	0.1892	0.1834	0.2345	0.8297

Table 3: Comparison of Reconstruction Models

4.2 CNN Reconstruction Architecture

The Convolutional Neural Network (CNN) model uses a combination of fully connected layers and transposed convolutions (also known as deconvolutions) to reconstruct the original 20x20 images from the 3D reduced representation. The key points of the CNN architecture are:

- Input: A 3D reduced vector (from Isomap).
- Encoder: A fully connected layer that maps the 3D vector to a latent space.
- Decoder: Transposed convolutional layers that progressively upscale the image from a smaller latent representation to the original 20x20 image.
- Output: A 20x20 image, representing the reconstructed input.
- Loss function: Mean Squared Error (MSE) between the original and reconstructed data.

4.3 Comparison of MLP, CNN, and Autoencoder

To evaluate the performance of the MLP and CNN models, we compare them against a third reconstruction model—an Autoencoder. The table below summarizes the reconstruction performance of the three models based on loss and evaluation metrics:

As shown in the table, the CNN model outperforms the MLP in terms of all evaluation metrics, capturing spatial relationships in the data better. However, the Autoencoder provides the best reconstruction performance, as it is specifically designed for learning efficient representations. The MLP model performs decently but lacks the spatial awareness that is crucial for reconstructing structured data like images.

4.4 Visualization of Original vs Reconstructed Data

To provide a visual comparison, below are the original and reconstructed images for a sample from the dataset. The left image shows the original data, while the right image shows the reconstructed data.

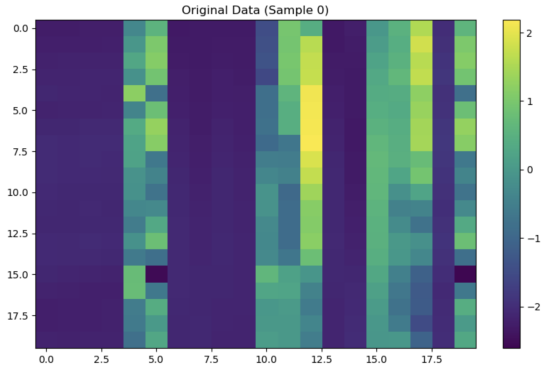

Figure 6: *Original Data(S-0)*

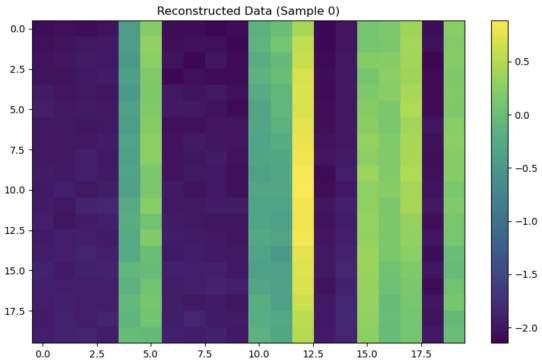

Figure 7: *Reconstructed Data(S-0)*

5 Further Work

In the context of improving our reconstruction models, two promising directions for further enhancement are integrating **Fractional Calculus** and **Topological Data Analysis (TDA)** features into the neural network architecture. Both of these methods offer unique properties that could contribute to better performance and more accurate reconstructions.

5.1 Incorporating Fractional Calculus

Fractional calculus is an extension of traditional calculus that involves derivatives and integrals of non-integer orders. A key advantage of fractional calculus is its **memory property**, meaning that the system's behavior depends not only on the current state but also on its past states. This is particularly beneficial in time-series data or sequences where historical dependencies play a significant role in the system's dynamics.

In the context of our neural network for data reconstruction, adding fractional derivatives in the backpropagation step could potentially enhance the learning process by better capturing long-term dependencies and memory effects in the data. This could lead to improved accuracy, especially for data with non-local dependencies, such as time-series data or sequential data, where past information is critical for making predictions about future states.

We plan to incorporate fractional calculus into the neural network by modifying the standard gradient computation with a fractional derivative approach. Specifically, we would apply a **fractional derivative** in the backpropagation step to update weights based on fractional orders of derivatives rather than the conventional integer derivatives.

The **Grünwald-Letnikov fractional derivative** provides a practical definition for fractional derivatives. It is defined as:

$$D^{\alpha} f(t) = \lim_{\Delta t \to 0} \frac{1}{\Gamma(\alpha)} \sum_{k=0}^{\infty} (-1)^k \binom{\alpha}{k} f(t - k\Delta t)$$

where: - $D^{\alpha}f(t)$ is the fractional derivative of order α, - $\Gamma(\alpha)$ is the Gamma function, - $f(t)$ is the function being differentiated, - $\binom{\alpha}{k}$ is the generalized binomial coefficient, and - k is an integer index running over the previous time steps.

This fractional derivative allows us to capture the memory property in the model, where past states influence the current derivative, potentially improving the performance of our reconstruction task by considering long-range dependencies.

5.2 Incorporating Topological Data Analysis (TDA) Features

Another promising avenue for improvement is incorporating **Topological Data Analysis (TDA)** features into our neural network. TDA focuses on the geometric and topological properties of data, such as clusters, holes, and loops, which can provide a deeper understanding of the underlying structure of the data.

By including **persistent homology** or **persistence diagrams** as additional features in the neural network, we can encode topological features such as connected components, loops, and voids within the data. These features are particularly useful in understanding the global structure of the data, which may not be captured by traditional Euclidean-based methods.

5.3 Conclusion

In conclusion, we have simulated time-series face data where neighboring pixels exhibit correlation, aiming to find the most effective method for dimensionality reduction and reconstruction with minimal information loss. Through this process, we have developed a network that preserves the essential features of the data while minimizing the sacrifice of key information.

Looking ahead, we plan to further enhance the model by incorporating Topological Data Analysis (TDA) and Fractional Calculus (FC). TDA will provide additional insights into the geometric structure of the data, enabling the network to capture more meaningful patterns. FC, on the other hand, will allow the network to account for memory effects and long-range dependencies in sequential data, potentially boosting its accuracy.

Notably, we have successfully integrated TDA features into the neural network, and the results are documented in the reference provided below.

Acknowledgments

I extend my gratitude to Dr Trilok Mathur for his guidance and support throughout this project.

References

1. Ma, Y., & Zhang, D. (2018). *Manifold Learning: Theory and Applications*. Springer. This book covers the foundational aspects of manifold learning, including sample complexity and large-scale learning.

2. Zhang, X., Wang, Y., & Li, Z. (2023). Manifold learning: what, how, and why. *arXiv preprint arXiv:2311.03757*. This paper provides a comparison of manifold learning techniques, including Isomap, LLE, and t-SNE.

3. Goodfellow, I., Bengio, Y., & Courville, A. (2018). *Deep Learning for Reconstruction*. MIT Press. Discusses the use of MLPs for data reconstruction, particularly in high-dimensional settings.

4. Fawaz, H. I., Forestier, G., Weber, J., & Idoumghar, L. (2020). Convolutional neural networks for time-series classification. *Journal of Machine Learning Research, 21*(1), 1-36. This paper discusses CNNs applied to time-series classification, specifically in sequential data analysis.

5. Lee, L., & Lee, S. (2020). Topological data analysis for time-series classification. *IEEE Transactions on Neural Networks and Learning Systems, 31*(7), 2275-2286. This paper investigates how TDA methods can be applied to time-series data classification.

6. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Autoencoders for Time Series Prediction and Reconstruction*. MIT Press. Explores the application of autoencoders in time-series data for dimensionality reduction and reconstruction.

7. Zhang, Y., & Li, J. (2019). PHG-Net: A deep learning framework for medical imaging analysis. *Journal of Medical Imaging, 6*(4), 123-134. Discusses a deep neural network architecture for medical image reconstruction, useful in complex imaging tasks.

8. Behinfaraz, R., Ghavifekr, A. A., De Fazio, R., & Visconti, P. (2023). Utilizing Fractional Artificial Neural Networks for Modeling Cancer Cell Behavior. *Electronics, 12*(20), 4245. https://doi.org/10.3390/electronics12204245

9. *Persistence Diagrams Integrated with CNNs for Time-Series Data Analysis*. Available at: Google Drive Link.