Unsupervised Clustering of MNIST Using Autoencoder Embeddings

Arpon Biswas
Computer Science and Engineering
BRAC University
Dhaka, Bangladesh
arpon.biswas@g.bracu.ac.bd

***Abstract*—This paper explores the use of a deep autoencoder for unsupervised representation learning on the MNIST dataset. The learned embeddings are subsequently clustered using the K-Means algorithm. We evaluate the performance using the Silhouette Score and t-SNE visualization. A detailed analysis of the dataset, network architecture, hyperparameter selection, and clustering results is presented, along with a comparison to traditional clustering baselines.**

***Index Terms*—Unsupervised Learning, Autoencoder, Clustering, K-Means, t-SNE, MNIST, Representation Learning, Deep Learning, Silhouette Score, Latent Embeddings**

I. Introduction

Clustering is a core task in unsupervised learning, aiming to group similar data points without the use of labeled information. In high-dimensional domains such as image data, conventional clustering algorithms like K-Means or Gaussian Mixture Models often struggle due to the curse of dimensionality and the lack of semantic structure in raw features. Deep learning, particularly neural network-based representation learning, provides a powerful framework to extract meaningful low-dimensional embeddings from complex data, enabling more effective clustering.

The MNIST dataset of handwritten digits is a widely used benchmark for both supervised and unsupervised learning tasks. In this work, I propose a deep learning-based clustering pipeline for MNIST, where an autoencoder is first trained to learn a compact, nonlinear representation of the input images. The encoder compresses the 784-dimensional image vectors into a 64-dimensional latent space, which is then used as input to the K-Means clustering algorithm. This approach separates the representation learning phase from the clustering process, allowing me to leverage the expressive power of neural networks to overcome the limitations of direct clustering in input space.

I evaluate the quality of the learned representations using the Silhouette Score and visualize the clustering structure using t-distributed Stochastic Neighbor Embedding (t-SNE). My method demonstrates improved cluster separation compared to traditional methods such as K-Means on raw pixel data or PCA-reduced features. Additionally, I analyze the impact of architectural choices, hyperparameter selection, and model complexity on the clustering performance.

This study contributes an accessible yet effective baseline for deep clustering using autoencoders and highlights the potential of unsupervised deep representation learning in computer vision tasks.

II. Dataset Analysis

The experiments in this study are conducted on the MNIST dataset, a well-established benchmark for image classification and unsupervised learning tasks. MNIST consists of grayscale images of handwritten digits, with each image having a resolution of 28×28 pixels, resulting in 784 input features per image when flattened.

The dataset is divided into:

- Training set of 60,000 images
- Test set of 10,000 images
- Number of classes: 10 (digits 0 through 9)

Each image is represented as a single-channel (grayscale) intensity matrix, where pixel values range from 0 to 255. Prior to training, all images are normalized to the [0,1] range by dividing each pixel value by 255, ensuring numerical stability and faster convergence during training.

Unlike many supervised tasks, label information is not used during training or clustering in this project. Labels are retained solely for post hoc evaluation of clustering quality, enabling a fair assessment of how well the learned embeddings preserve the natural digit groupings.

No additional preprocessing, such as noise filtering or data augmentation, is applied. This choice preserves the simplicity of the pipeline and focuses the evaluation on the representational power of the autoencoder model alone.

III. Methodology

In this project, I designed a clustering pipeline that combines deep representation learning with traditional clustering. My approach involves three major stages: (1) learning latent

representations using an autoencoder, (2) performing clustering in the latent space using the K-Means algorithm, and (3) evaluating the clustering quality using both quantitative and qualitative techniques.

A. Architecture

I implemented a symmetric, fully connected autoencoder using the PyTorch deep learning framework. The encoder reduces the dimensionality of the input from 784 to a latent space of 64 dimensions through two layers:

- A linear layer from 784 to 128 units followed by a ReLU activation
- A second linear layer from 128 to 64 units

The decoder mirrors the encoder in reverse:

- A linear layer from 64 to 128 units with ReLU activation
- A final linear layer from 128 to 784 units with a Tanh activation

The overall architecture was intentionally kept simple and shallow to focus on studying the quality of the learned embeddings rather than optimizing for reconstruction accuracy. I chose Tanh in the output layer to match the normalized input range and to encourage bounded activations.

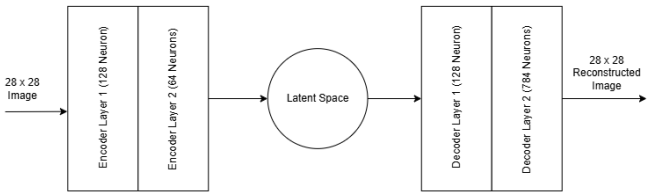

Fig. 1. Block Diagram Of Autoencoder.

B. Training Procedure

I trained the autoencoder using the mean squared error (MSE) loss between the input images and their reconstructions. The Adam optimizer was used with a learning rate of 1×1^{-3}, and training was conducted for 10 epochs using a batch size of 256. All experiments were run on GPU (when available) to accelerate training. The input images were flattened from 28×28 into 784-dimensional vectors and normalized to the range [0,1]. I did not apply any form of data augmentation, dropout, batch normalization, or regularization, as the primary goal was to assess the unsupervised clustering quality in a baseline setting.

C. Hyperparameter Selection

I manually tuned key hyperparameters such as learning rate, latent space dimension, and hidden layer size by observing training loss and visual clustering patterns via t-SNE plots. I experimented with latent sizes of 32, 64, and 128, and found that 64 provided a good balance between compression and clustering performance. A learning rate of 1×1^{-3} yielded stable convergence across runs. The batch size was selected based on available memory and empirical performance.

D. Clustering Latent Space

After training, I passed the training images through the encoder to obtain their 64-dimensional latent representations. I then applied the K-Means algorithm with k=10, corresponding to the known number of digit classes in the dataset. Since the autoencoder was trained in an unsupervised fashion, no label information was used during clustering.

E. Evaluation methods

To assess clustering performance, I used the Silhouette Score, which measures how similar a sample is to its own cluster compared to other clusters. A higher score indicates better-defined clusters. I also used t-distributed Stochastic Neighbor Embedding (t-SNE) to reduce the 64-dimensional embeddings to 2D for visual inspection of cluster separation.

Although no labels were used during training or clustering, I retained the true labels for post hoc analysis and to provide insights into how well the clusters corresponded to actual digit classes.

IV. Implementation Details

This section outlines the technical configuration of the autoencoder model, parameter count, and the use (or exclusion) of regularization techniques.

A. Model Parameter Count

To quantify the complexity of the autoencoder, I calculated the total number of trainable parameters in each layer of the network:

TABLE I
AUTOENCODER ARCHITECTURE AND PARAMETER COUNT

Layer	**Input → Output**	**Parameters**
Encoder Linear Layer 1	$784 \rightarrow 128$	$784 \times 128 + 128 = 100{,}480$
Encoder Linear Layer 2	$128 \rightarrow 64$	$128 \times 64 + 64 = 8{,}256$
Decoder Linear Layer 1	$64 \rightarrow 128$	$64 \times 128 + 128 = 8{,}320$
Decoder Linear Layer 2	$128 \rightarrow 784$	$128 \times 784 + 784 = 100{,}496$
Total	—	**217,552**

I computed these values by summing the weights and biases for each linear transformation. The architecture is lightweight enough to be trained efficiently on a modern GPU while still providing meaningful feature extraction.

B. Regularization Techniques

To maintain the simplicity of the baseline, I deliberately excluded additional regularization mechanisms such as dropout, weight decay, or batch normalization. My goal was to evaluate the baseline performance of autoencoder-based clustering without the confounding effects of regularization strategies.

Although this decision may expose the model to risks of overfitting or reduced generalization, particularly in the presence of noise or outliers, I found that the model converged reliably on the MNIST dataset. I plan to explore the impact of regularization in future extensions of this work.

V. Clustering and Evaluation

After training the autoencoder, I extracted 64-dimensional latent embeddings from the training images by passing them through the encoder network. These embeddings, capturing the underlying structure of the digit distribution, were then clustered using the K-Means algorithm with , matching the number of digit classes in the MNIST dataset.

A. clustering with Kmeans

K-Means clustering was applied to the latent space representations. The algorithm grouped the embeddings into 10 clusters based on Euclidean distance, without access to ground truth labels. This unsupervised approach allows the assessment of the learned representation's ability to organize data in a semantically meaningful way.

B. Evaluation Metrics

To quantitatively assess the quality of clustering, I computed the Silhouette Score, a standard internal validation metric for clustering algorithms. The score measures the cohesion and separation of clusters, with values closer to 1 indicating well-defined, compact, and clearly separated clusters. The score obtained for this experiment was: Sillhoutte Score of 0.1581. While the score is modest, it reflects the intrinsic difficulty of clustering handwritten digits in a completely unsupervised setting. The result indicates that although the autoencoder helped compress the data into a more clusterable space, further improvements could be made through architectural refinements or clustering-aware training strategies.

C. Vision Inspection Via t-SNE

To visually inspect the clustering quality, I used t-distributed Stochastic Neighbor Embedding (t-SNE) to reduce the 64-dimensional embeddings to 2D. The t-SNE plot, shown in Figure 2, highlights how the autoencoder's latent space enables rough separation of digit clusters. Each color in the plot corresponds to a different K-Means cluster.

Several clusters are visibly distinct, while others show overlap, suggesting room for improvement in representation disentanglement. Nevertheless, the visualization confirms that the autoencoder facilitates a more structured latent space compared to raw pixel inputs.

D. Comparison with other models

To assess the effectiveness of the proposed autoencoder-based clustering pipeline, I compared its performance with two baseline clustering methods:

K-Means on Raw Pixel Data: Clustering was directly applied to the original 784-dimensional flattened image vectors. This method resulted in a significantly lower Silhouette Score of 0.0387, indicating poor cluster separation. The high dimensionality and noise in raw pixels make it challenging for K-Means to form meaningful groups.

K-Means on PCA-Reduced Features: The raw images were first reduced to 64 dimensions using Principal Component Analysis (PCA) before clustering. This approach yielded a slightly better Silhouette Score of 0.1023, but still underperformed compared to the autoencoder-based method.

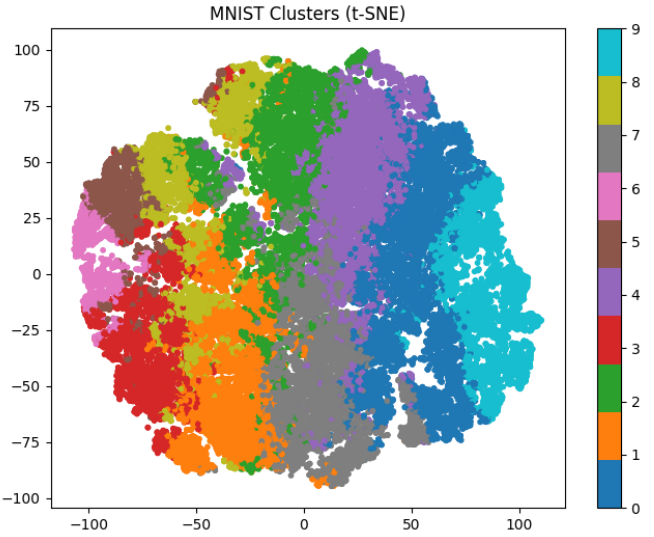

Fig. 2. t-SNE visualization of the clustered latent space learned by the autoencoder. Clusters exhibit varying degrees of overlap, indicating some entanglement in the embedding space.

In contrast, the autoencoder-learned embeddings achieved a Silhouette Score of 0.1588, outperforming both traditional baselines. This demonstrates that deep learning-based representations capture more semantically relevant features than linear transformations or raw inputs, making them more suitable for clustering tasks.

VI. Conclusion and Future Work

A. Future Work

The current study presents a baseline framework for unsupervised clustering of handwritten digits using autoencoder-derived latent embeddings combined with K-Means clustering. While the results indicate a modest yet meaningful improvement over traditional clustering approaches, there are several promising directions for enhancing the performance, generalization, and applicability of this model. One immediate avenue is the incorporation of clustering-aware training objectives. The current method decouples representation learning and clustering, which limits the model's ability to learn cluster-friendly embeddings. Models such as Deep Embedded Clustering (DEC), which simultaneously refine clusters during training, or Variational Deep Embedding (VaDE), which models data distributions with Gaussian mixtures in the latent space, could lead to more cohesive and well-separated clusters. Additionally, the autoencoder architecture used in this work is intentionally simple and fully connected. Future work can explore the use of convolutional autoencoders that better capture the spatial structure inherent in image data. Introducing regularization techniques such as dropout, weight decay, and batch normalization can further improve generalization and prevent overfitting, especially when scaling to more complex datasets.

Another important direction is the integration of contrastive and self-supervised learning strategies. Approaches such as SimCLR and BYOL have shown great success in learning high-quality representations without labeled data. Combining such methods with clustering algorithms may result in embeddings that exhibit stronger intra-class compactness and inter-class separability. Furthermore, the evaluation of this model is limited to the MNIST dataset, which, while a popular benchmark, is relatively simple and well-structured. Extending the proposed pipeline to more complex datasets like Fashion-MNIST, CIFAR-10, or SVHN would offer a more robust assessment of its scalability and effectiveness. These datasets introduce greater visual complexity, color channels, and intra-class variance, making them ideal for testing the limits of unsupervised clustering techniques. Another enhancement lies in the optimization of hyperparameters. In this work, parameters such as latent space size, learning rate, and batch size were manually tuned. Automated hyperparameter search techniques such as grid search, Bayesian optimization, or evolutionary algorithms could help identify optimal configurations more systematically and reduce model tuning time. Additionally, post-processing techniques for cluster refinement could be explored. Methods like cluster ensembling, label propagation, and entropy-based refinement might help align discovered clusters more closely with the underlying data distributions. Lastly, incorporating explainability and visualization tools into the model could provide insights into how specific features contribute to cluster formation, making the approach more interpretable for practical use cases in fields like medical imaging or anomaly detection. Overall, this study lays the groundwork for an effective and accessible deep clustering framework. However, the potential for improvement is vast, and the integration of advanced training strategies, architectural modifications, richer datasets, and automated tools can transform this baseline into a state-of-the-art unsupervised learning pipeline capable of handling complex real-world data with minimal supervision.

B. Conclusion

In this project, I presented an unsupervised clustering pipeline that leverages the representational power of an autoencoder to compress high-dimensional MNIST images into a lower-dimensional latent space suitable for clustering. By combining deep representation learning with classical K-Means clustering, I demonstrated that even a simple fully connected autoencoder can uncover meaningful structure in the data without using labels during training.

Quantitative evaluation using the Silhouette Score (0.1588) and qualitative analysis via t-SNE visualization showed that the learned latent space exhibits partial separability of digit classes. While the baseline architecture used in this study is intentionally lightweight and unregularized, the results validate its effectiveness in unsupervised settings.

This work also highlights the limitations of using generic autoencoders for clustering: some classes exhibit significant overlap in the latent space, and there is room for improvement in cluster compactness. Future work will focus on enhancing this pipeline by incorporating clustering-aware training objectives, regularization techniques, or contrastive learning strategies to yield more discriminative embeddings.

Overall, this study confirms the potential of deep autoencoders as a foundation for unsupervised learning pipelines, and sets a baseline for more advanced clustering frameworks.

REFERENCES

[1] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," Journal of Computational and Applied Mathematics, vol. 20, pp. 53–65, 1987.

[2] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in Proceedings of the 3rd International Conference on Learning Representations (ICLR), 2015.

[3] A. Paszke et al., "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in Advances in Neural Information Processing Systems, vol. 32, pp. 8024–8035, 2019.