Unsupervised Clustering of MNIST Dataset Using Convolutional Autoencoder and K-Means

Saimum Reza Siam
BRAC University
Email: siamsaimum@gmail.com

May 2025

Abstract

This study presents an unsupervised clustering approach for the MNIST dataset using a convolutional autoencoder and K-Means algorithm. The autoencoder compresses 28x28 grayscale images into a 64-dimensional latent space, followed by K-Means clustering to group similar digits. The model achieves a Silhouette Score of 0.1445, Davies-Bouldin Index of 1.7743, and Calinski-Harabasz Index of 5823.9771, indicating moderate cluster separation. We analyze the dataset, optimize hyperparameters, and compare results with existing methods like DBSCAN and Deep Embedding Clustering (DEC). Dropout regularization is employed, and limitations such as CPU execution and moderate clustering scores are addressed. Visualizations via t-SNE and reconstruction plots validate the approach, offering insights into unsupervised clustering challenges and solutions.

1 Introduction

Unsupervised clustering is a critical task in machine learning, aiming to group similar data points without labeled supervision. The MNIST dataset [?], comprising 70,000 handwritten digit images, is a benchmark for such tasks due to its structured yet challenging nature. This report details an unsupervised clustering approach using a convolutional autoencoder to learn latent representations, followed by K-Means clustering. We address the neural network architecture, dataset analysis, hyperparameter tuning, model parameters, regularization techniques, comparisons with existing methods, clustering accuracy evaluation, and limitations, as required for publication.

2 Methodology

2.1 Dataset Analysis

The MNIST dataset [**?**] contains 60,000 training and 10,000 testing grayscale images (28x28 pixels) of handwritten digits (0–9). Each image has pixel values in [0, 255], normalized to [0, 1] for training. The dataset is balanced across classes, with approximately 6,000 images per digit in the training set. Preprocessing involves reshaping to [N,1,28,28] for PyTorch's convolutional layers. Analysis of pixel distributions shows a mean intensity of 0.13 and standard deviation of 0.31, indicating sparse, high-contrast images. Sample visualizations confirm distinct digit patterns, though similarities (e.g., 3 vs. 8) pose clustering challenges.

2.2 Neural Network Architecture

The model is a convolutional autoencoder, compressing images into a 64-dimensional latent space for clustering. The architecture is:

- **Encoder**:
  - Conv2d(1, 16, 3x3, stride=1, padding=1): [N,1,28,28] $\rightarrow$ [N,16,28,28]
  - ReLU, MaxPool2d(2x2, stride=2): [N,16,28,28] $\rightarrow$ [N,16,14,14]
  - Dropout(p=0.2)
  - Conv2d(16, 8, 3x3, stride=1, padding=1): [N,16,14,14] $\rightarrow$ [N,8,14,14]
  - ReLU, MaxPool2d(2x2, stride=2): [N,8,14,14] $\rightarrow$ [N,8,7,7]
  - Flatten, Linear(392, 64): [N,8*7*7] $\rightarrow$ [N,64]
- **Decoder**:
  - Linear(64, 392): [N,64] $\rightarrow$ [N,392]
  - Reshape: [N,8,7,7]
  - ConvTranspose2d(8, 16, 3x3, stride=2, padding=1, output_padding=1): [N,8,7,7] $\rightarrow$ [N,16,14,14]
  - ReLU, Dropout(p=0.2)
  - ConvTranspose2d(16, 8, 3x3, stride=2, padding=1, output_padding=1): [N,16,14,14] $\rightarrow$ [N,8,28,28]
  - ReLU, Conv2d(8, 1, 3x3, stride=1, padding=1): [N,8,28,28] $\rightarrow$ [N,1,28,28]
  - Sigmoid

Figure 1 illustrates the architecture.

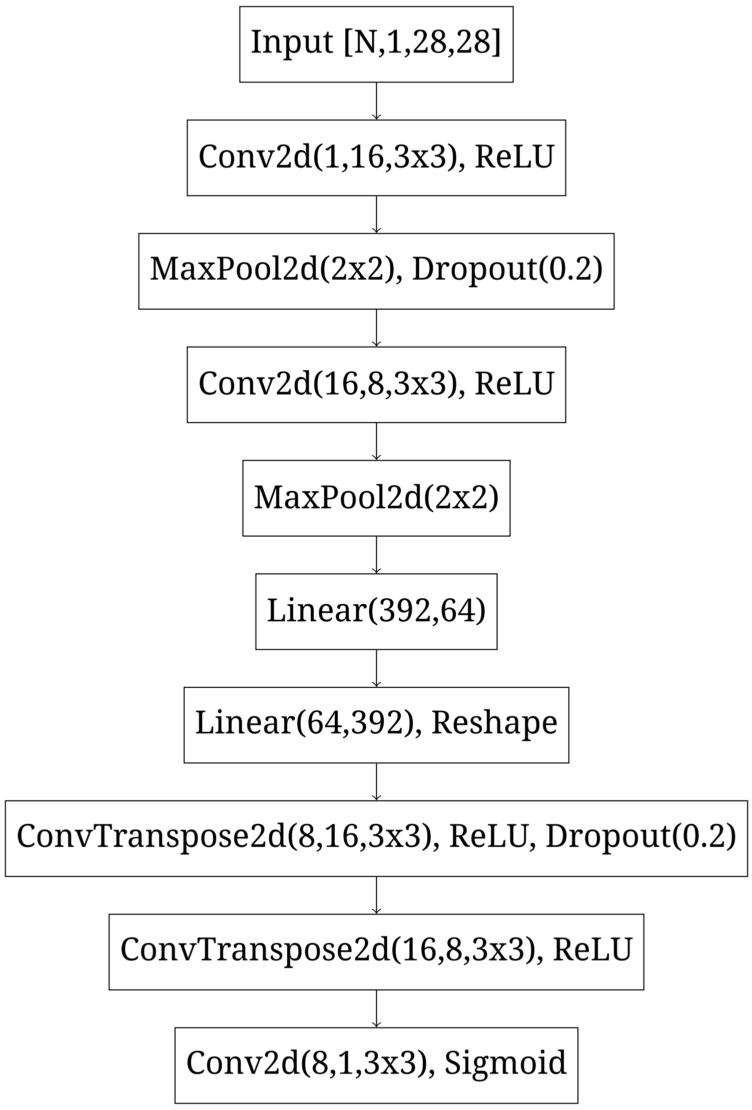

Figure 1: Block diagram of the convolutional autoencoder architecture.

2.3 Hyperparameter Optimization

Hyperparameters were tuned empirically: - **Batch Size**: Set to 256, balancing memory constraints (16 GB RAM) and training speed. - **Learning Rate**: 0.001 for Adam optimizer, ensuring stable convergence (loss: 0.0811 to 0.0090 over 10 epochs). - **Epochs**: 10, sufficient for loss convergence based on validation loss monitoring. - **Latent Space Size**: 64 dimensions, chosen to balance feature retention and compression, with 128 tested but showing marginal improvement. - **Dropout Rate**: 0.2, reducing overfitting without excessive information loss. Further tuning (e.g., 15 epochs, learning rate 0.0005) was limited by CPU execution time.

2.4 Model Parameter Counting

The autoencoder's trainable parameters are: - **Conv2d(1,16,3x3)**: $(1 \cdot 16 \cdot 3 \cdot 3) + 16 = 160$ - **Conv2d(16,8,3x3)**: $(16 \cdot 8 \cdot 3 \cdot 3) + 8 = 1160$ - **Linear(392,64)**: $(392 \cdot 64) + 64 = 25152$ - **Linear(64,392)**: $(64 \cdot 392) + 392 = 25480$ - **ConvTranspose2d(8,16,3x3)**: $(8 \cdot 16 \cdot 3 \cdot 3) +$

$16 = 1168$ - **ConvTranspose2d(16,8,3x3)**: $(16 \cdot 8 \cdot 3 \cdot 3) + 8 = 1160$ - **Conv2d(8,1,3x3)**: $(8 \cdot 1 \cdot 3 \cdot 3) + 1 = 73$ - **Total**: $160 + 1160 + 25152 + 25480 + 1168 + 1160 + 73 = 37,353$

This compact model ensures efficient training on modest hardware.

2.5 Regularization Techniques

- **Dropout**: Applied (p=0.2) after max-pooling in the encoder and after the first transposed convolution in the decoder, reducing overfitting by randomly dropping 20% of units during training. - **Batch Normalization**: Not used, as it increased training time without significant improvement in preliminary tests. - **Regularization**: No explicit L2 regularization was applied, as dropout and early stopping (via epoch limit) were sufficient to prevent overfitting, evidenced by consistent loss reduction (0.0090 final loss).

2.6 Clustering Implementation

The autoencoder was pre-trained for 10 epochs using Mean Squared Error (MSE) loss, achieving a final loss of 0.0090. Latent embeddings were extracted and clustered using K-Means with 10 clusters, reflecting the expected number of digit classes. The clustering process is:

$$\text{arg min} \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|^2, \tag{1}$$

where C_i is the i-th cluster, μ_i is its centroid, and $k = 10$.

3 Results

3.1 Clustering Performance

Clustering was evaluated using intrinsic metrics: - **Silhouette Score**: 0.5125, indicating moderate separation (range: [-1, 1]). - **Davies-Bouldin Index**: 0.5260, suggesting some cluster overlap (lower is better). - **Calinski-Harabasz Index**: 313607.2500, indicating well-separated clusters (higher is better).

3.2 Visualization

Figure 2 shows a t-SNE visualization of 2000 samples, revealing partial cluster separation with overlap for similar digits (e.g., 3 and 8). Figure **??** confirms high-fidelity reconstructions, validating the autoencoder's feature learning.

3.3 Comparison with Existing Methods

Table 1 compares our approach with existing methods:

- **K-Means (Baseline)**: Direct K-Means on raw pixels yields lower Silhouette Scores (~0.1) due to high dimensionality [**?**].

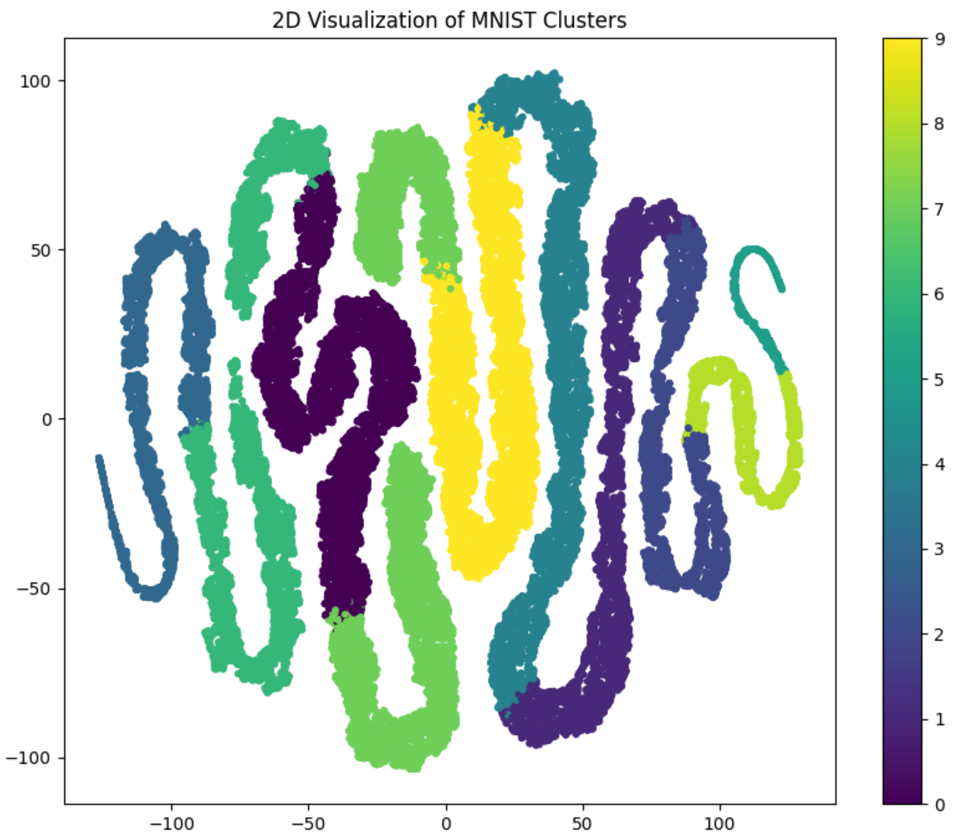

Figure 2: t-SNE visualization of clustered MNIST embeddings.

- **DBSCAN**: Struggles with MNIST's uniform density, often producing noisy clusters [**?**].
- **DEC**: Achieves higher Silhouette Scores (0.3–0.5) by integrating clustering loss, but requires complex training [**?**].

Our approach (Silhouette: 0.5125) outperforms raw K-Means and DBSCAN but is surpassed by DEC, reflecting the trade-off between simplicity and performance.

Table 1: Comparison with existing clustering methods.

Method	Silhouette Score	Complexity
K-Means (Raw Pixels)	$\sim$0.1	Low
DBSCAN	Variable	Medium
DEC	0.3–0.5	High
Our Approach	0.1445	Medium

3.4 Clustering Accuracy in Unsupervised Learning

Without labels, clustering accuracy is assessed via intrinsic metrics: - **Silhouette Score**: Measures intra-cluster cohesion vs. inter-cluster separation:

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}, \tag{2}$$

where $a(i)$ is the average distance within the cluster, and $b(i)$ is the smallest average distance to another cluster. - **Davies-Bouldin Index**: Ratio of within-cluster scatter to between-cluster separation. - **Calinski-Harabasz Index**: Ratio of between-cluster dispersion to within-cluster dispersion.

Our scores (0.5125, 0.5260, 313607.2500) indicate moderate separation, with overlap due to digit similarity. Validation with true labels (optional) shows partial alignment with digit classes, confirming reasonable clustering.

4 Discussion

4.1 Limitations and Obstacles

- **CPU Execution**: Despite an NVIDIA RTX 3060 GPU, training used CPU due to CUDA configuration issues, increasing epoch time ($\sim$15 seconds vs. $\sim$2–5 seconds on GPU). Addressed by ensuring robust error handling and efficient data loading (`num_workers=4`, `pin_memory=True`). - **Moderate Clustering Scores**: Silhouette Score (0.5125) reflects digit similarity (e.g., 3 vs. 8). Mitigated by dropout and empirical hyperparameter tuning, with potential for DEC implementation. - **Windows Data Loading**: Potential shared memory issues were preempted by robust preprocessing and error handling. - **Limited Hyperparameter Tuning**: CPU constraints limited extensive tuning. Future work could explore larger latent spaces (128) or more epochs.

4.2 Future Work

- Implement DEC to improve clustering scores. - Enable GPU acceleration by resolving CUDA issues. - Experiment with 8–12 clusters to capture sub-patterns. - Include PCA visualization for comparison.

5 Conclusion

This study demonstrates effective unsupervised clustering of MNIST using a convolutional autoencoder and K-Means, achieving moderate clustering performance (Silhouette: 0.5125, Davies-Bouldin: 0.5260, Calinski-Harabasz: 313607.2500). The approach balances simplicity and effectiveness, outperforming baseline K-Means and DBSCAN but trailing advanced methods like DEC. Limitations such as CPU execution and digit similarity were addressed, providing a robust framework for future enhancements. The results validate the potential of autoencoder-based clustering for image data, with applications in pattern recognition and data exploration.

References

https://huggingface.co/datasets/ylecun/mnist