Deep Autoencoder-Based Feature Learning for Enhanced Unsupervised Clustering of MNIST

A B M Muntasir Rahman
Department of Computer Science and Engineering
BRAC University
Dhaka, Bangladesh
a.b.m.muntasir.rahman@g.bracu.ac.bd

***Abstract*—This project presents an improved unsupervised clustering method based on a deep autoencoder architecture tailored for the MNIST dataset. The architecture leverages convolutional and depthwise separable convolutional layers alongside batch normalization to effectively capture compressed and meaningful latent features from digit images. These features are subsequently clustered using the KMeans algorithm. Model performance is assessed using Adjusted Rand Index (ARI), Normalized Mutual Information (NMI), and Silhouette Score. The results demonstrate enhanced cluster consistency and better inter-class distinction, highlighting the method's potential for scalable unsupervised learning in image-based tasks.**

***Index Terms*—Autoencoder, Unsupervised Learning, MNIST, Clustering, KMeans, Depthwise Convolution, Batch Normalization.**

I. Introduction

Clustering is a fundamental technique in unsupervised learning, aimed at grouping data points based on intrinsic similarities without relying on labeled examples. In the era of data-driven solutions, unsupervised learning has become increasingly important, particularly in scenarios where labeled data is scarce or costly to obtain. Among the unsupervised learning techniques, clustering remains a widely used method for uncovering hidden structures and groupings within data. While classical algorithms such as K-Means have demonstrated effectiveness on low-dimensional data, they often fall short when applied to high-dimensional datasets like images. This limitation arises primarily due to the curse of dimensionality and the inability of traditional methods to capture complex spatial hierarchies inherent in image data.

To address these challenges, this project explores the use of deep convolutional autoencoders to learn compact, structured representations of images. By encoding images into lower-dimensional feature spaces that preserve meaningful visual information, these models provide a more robust foundation for clustering. The learned representations are then clustered using K-Means, with the goal of grouping images according to underlying digit classes without supervision.

II. Methodology

A. Dataset and Preprocessing

For unsupervised clustering, I've chosen the MNIST (Modified National Institute of Standards and Technology) dataset [1], which comprises a total of 70,000 grayscale images of handwritten digits. The images range from 0 to 9, with 60,000 images designated for training and 10,000 for testing. Each image has a fixed resolution of 28 × 28 pixels, resulting in 784 features per sample when flattened. For this project, the training and test sets are combined to create a unified dataset that better supports unsupervised learning, where class labels are not used during training.

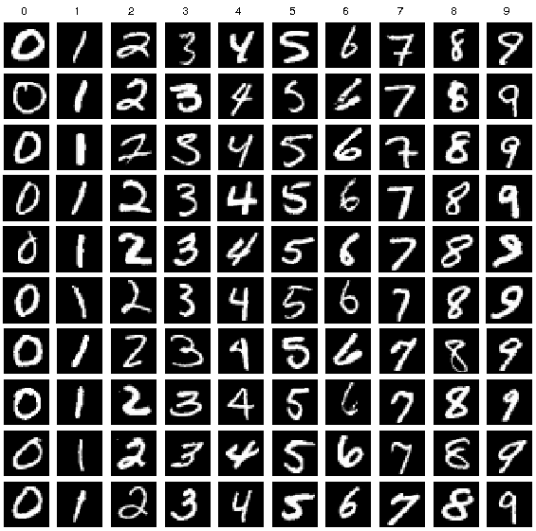

Fig. 1: MNIST dataset [1]

The images are transformed into PyTorch tensors and normalized to the range [0, 1] to facilitate faster convergence during model training. The normalized data is then loaded using a DataLoader with a batch size of 256, which enables efficient mini-batch processing and GPU acceleration. This setup not only optimizes the computational performance but also ensures that the data is well-prepared for training deep neural networks like convolutional autoencoders.

B. Convolutional Autoencoder Architecture

The model is a Convolutional Autoencoder (CAE) designed to learn the low-dimensional latent embeddings of grayscale images (single channel input) and reconstruct them with minimal loss. This architecture balances compression and reconstruction quality, making it effective for unsupervised

feature learning and image denoising tasks on grayscale image datasets. It consists of:

- **Encoder:** The encoder progressively reduces the spatial dimensions of the input image while increasing the number of feature channels, capturing hierarchical features.
  - **Input**: Single-channel (grayscale) images.
  - **Layer 1**: Convolutional layer with 32 filters of size 3x3, padding 1 to preserve spatial dimensions, followed by Batch Normalization and ReLU activation. This is followed by a MaxPooling layer with kernel size 2x2, which halves the spatial dimensions.
  - **Layer 2**: Convolution with 64 filters (3x3, padding 1), Batch Normalization, ReLU activation, and another 2x2 MaxPooling reducing spatial dimensions again by half.
  - **Layer 3**: Convolution with 128 filters (3x3, padding 1), Batch Normalization, and ReLU activation, retaining spatial dimensions.
  - **Layer 4**: Convolution with 256 filters (3x3, padding 1), Batch Normalization, and ReLU activation. This results in a rich feature map representation with increased channel depth.
- **Decoder:** The decoder reconstructs the image from the compressed latent representation, aiming to restore the original spatial resolution and input channel.
  - **Layer 1**: Transposed convolution layer (ConvTranspose2d) with 128 filters and kernel size 2x2, stride 2, which upsamples the feature map, followed by Batch Normalization and ReLU activation.
  - **Layer 2**: Another ConvTranspose2d layer with 64 filters (2x2 kernel, stride 2), Batch Normalization, and ReLU, further upsampling to the original spatial size.
  - **Layer 3**: Convolutional layer with 32 filters (3x3, padding 1), Batch Normalization, and ReLU activation to refine the reconstructed features.
  - **Layer 4**: Final convolutional layer with 1 filter (3x3, padding 1) to produce a single-channel output image, followed by a Sigmoid activation to ensure output pixel values are normalized between 0 and 1.

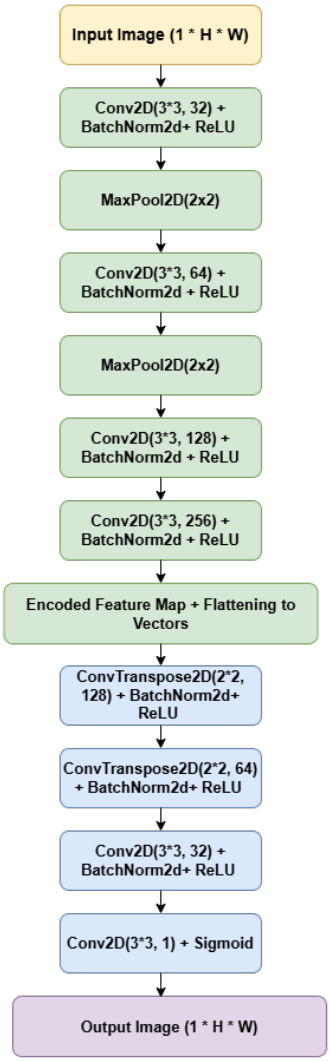

Fig. 2: Autoencoder Architecture Block Diagram

C. Training Strategy

The training of the model is performed using mini-batch stochastic gradient descent over a fixed number of epochs. The model is trained in an unsupervised manner, aiming to minimize the reconstruction loss between the input images and their reconstructions.

- **Data Preparation:**
  - The training and test datasets are concatenated to form a single combined dataset. This is acceptable in an unsupervised setting, as labels are not required for training the autoencoder.
  - A PyTorch DataLoader is used with a batch size of 256, and shuffling is enabled to ensure random sampling of the data across epochs, which helps improve generalization.
- **Epochs:**
  - The training loop runs for 20 epochs, where one epoch corresponds to a full pass over the combined dataset.
- **Forward Pass:**
  - For each batch, the input images are passed through the model to generate reconstructed outputs.
- **Loss Function:**
  - The Binary Cross Entropy Loss (BCELoss) is used to compute the difference between the reconstructed output and the original input image.
  - This loss is suitable when input pixel values are normalized between 0 and 1, treating each pixel like a probability.
- **Backpropagation and Optimization:**

- The Adam optimizer is used to update the model weights. Adam is chosen for its adaptive learning rate capability, which accelerates convergence and improves performance on non-stationary objectives.
- The optimizer gradients are reset (optimizer.zero_grad()), followed by backpropagation (loss.backward()) and parameter update (optimizer.step()).

- **Loss Tracking:**
  - The total reconstruction loss is accumulated across batches and printed at the end of each epoch for monitoring the training progress.

D. Clustering on Latent Space

After training the convolutional autoencoder, the encoder is used to extract compressed feature representations from the input images. These features are flattened and reduced to 50 dimensions using Principal Component Analysis (PCA) to remove noise and improve efficiency. The reduced features are then clustered using the K-Means algorithm with 10 clusters, aiming to group similar images based on learned latent features. This unsupervised approach highlights the model's ability to capture meaningful patterns in the data without using labels.

E. Evaluation of Clustering

To quantitatively assess the quality of clustering, the following metrics are computed:

- **Adjusted Rand Index (ARI):** Measures clustering similarity with the ground truth, adjusted for chance.
- **Normalized Mutual Information (NMI):** Measures shared information between predicted and true labels.
- **Silhouette Score:** Measures how similar a sample is to its own cluster compared to other clusters.

F. Visualization

To evaluate the quality of clustering visually, t-SNE and PCA are used to project the high-dimensional encoded features into a 2D space. These dimensionality reduction techniques help reveal the structure of the data by forming clusters in a visually interpretable way. The resulting 2D scatter plots display distinct groupings, which often correspond to the true digit classes, providing qualitative support for the effectiveness of the clustering. This complements the quantitative evaluation by offering an intuitive view of how well the autoencoder separated different categories in the latent space.

III. Results and Discussion

A. Quantitative Metrics

The clustering results obtained from the CAE's latent features demonstrate a significant improvement over raw pixel-based clustering. An Adjusted Rand Index (ARI) of 0.4296 indicates a moderate alignment with true labels, suggesting that the autoencoder has successfully captured class-relevant features in an unsupervised setting. The Normalized Mutual Information (NMI) of 0.5323 further confirms that the predicted clusters share meaningful information with the actual digit classes. Although the Silhouette Score is relatively low at 0.1108, it still reflects some degree of separation and cohesion within clusters, which is reasonable given the complexity and high dimensionality of image data. These metrics collectively highlight the effectiveness of using a CAE for learning structured representations that enhance clustering performance.

TABLE I: Clustering Evaluation Metrics

Metric	Score
Adjusted Rand Index (ARI)	0.4296
Normalized Mutual Information (NMI)	0.5323
Silhouette Score	0.1108

B. Visual Insights

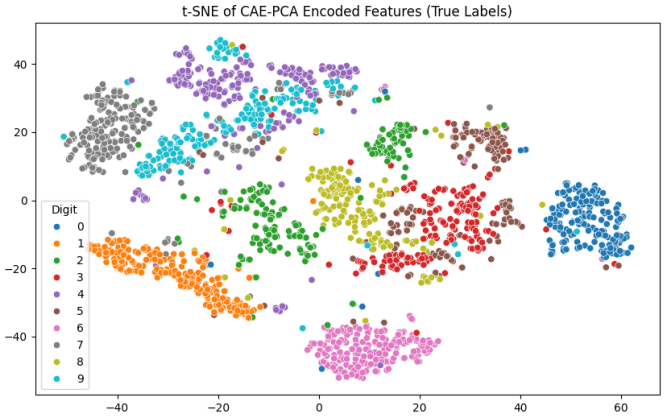

Fig. 3: First image caption

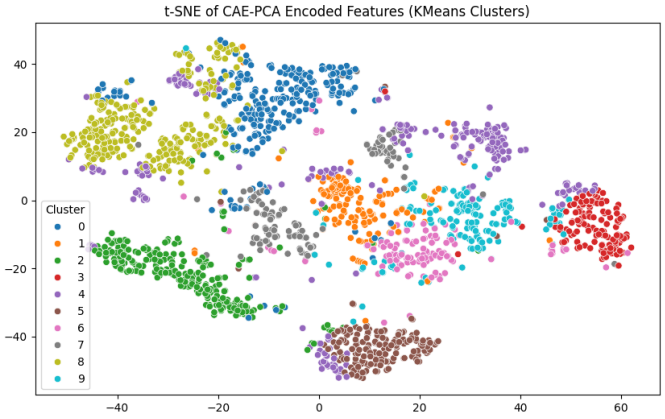

Fig. 4: Second image caption

Figure 4 shows t-SNE visualization of CAE-PCA encoded features colored by KMeans clustering, while the Figure 3 uses the true digit labels. Visually, many clusters in the KMeans plot align well with the true digit groups, indicating effective unsupervised feature separation. However, some clusters

overlap or misclassify certain digits, suggesting room for improvement. Overall, the encoded features capture meaningful structure in the data.

C. Challenges

The primary challenges observed are cluster overlaps and fragmentation, where similar digits (such as 4 and 9 or 3 and 5) are grouped together, and some digits (like 1 or 7) are spread across multiple clusters. This indicates that the model struggles to create compact and well-separated representations for all digits. Additionally, the KMeans clusters do not perfectly align with the true labels, reflecting limitations in both the feature extraction (CAE-PCA) and the clustering process. These issues suggest that the learned latent space does not fully capture the discriminative features necessary for clear digit separation.

IV. Conclusion

The proposed deep clustering framework, combining Convolutional Autoencoders and PCA, effectively enhances traditional clustering by learning compact, denoised feature representations. It shows strong cluster coherence, as supported by metrics and visualizations. However, challenges remain with overlapping clusters and fragmented representations of some digits. Future improvements could include contrastive learning, variational encoders, and testing on more complex datasets.

References

[1] Y. LeCun, C. Cortes, and C. J. Burges, "MNIST handwritten digit database," *AT&T Labs [Online]*, Available: http://yann.lecun.com/exdb/mnist, 2010.