Inspiring Excellence

Unsupervised Learning with Neural Networks: Autoencoder-based Clustering

Project Report

Submitted by

Tasfia Tasnim Prioty

Student ID: 23241101

Section: 3

CSE425: Neural Networks

School of Computer Science and Engineering
BRAC University

Submitted to

Moin Mostakim
Senior Lecturer,
Department of Computer Science and Engineering

Tuesday 20th May, 2025

Contents

Project Repository:
https://github.com/AuroraGrace501/Unsupervised-Learning-with-Neural-Networks

1 | Introduction

Unsupervised learning is a crucial area in machine learning where the model identifies patterns in data without using labeled examples. Clustering is one such task, which groups similar data points together based on learned embeddings. This work implements and compares unsupervised clustering techniques using neural network architectures on image data, specifically the CIFAR-10 dataset from Hugging Face. We design a convolutional autoencoder to extract latent embeddings and use clustering algorithms to form groups. The performance is compared against features extracted from ResNet50, a popular pretrained model.

2 | Dataset Analysis

The dataset used in this project is CIFAR-10, obtained via the Hugging Face datasets library. It contains 60,000 color images (32×32), across 10 balanced classes such as airplane, automobile, bird, cat, and others. We selected a subset of 1,000 images from the training split for efficient experimentation. All images were resized to 64×64 for the autoencoder and to 224×224 for ResNet50. Labels were not used at any stage, to maintain an unsupervised pipeline.

3 | Neural Network Architecture

We designed a custom convolutional autoencoder with 4 encoder layers and 3 decoder layers, using ReLU activation and a final sigmoid output.

3.1 Block Diagram

Input (64x64x3)
↓
Conv2D(32) → MaxPool → Dropout
↓
Conv2D(64) → MaxPool → Dropout
↓
Conv2D(128) → MaxPool → Dropout
↓
Conv2D(256) → Bottleneck
↓
Conv2D(128) → UpSampling → Dropout
↓
Conv2D(64) → UpSampling → Dropout
↓
Conv2D(32) → UpSampling
↓
Conv2D(3) → Output

The encoder's final output (shape: (8, 8, 256)) was flattened and used as the embedding for clustering.

4 | Performing Hyperparameter Optimization and Tuning

- I increased the number of epochs from 20 to 30 which helped improve Epoch.
- The Batch Size is recommended to be set to 64.
- I am using Adam and setting the learning rate at 1e-3.
- Latent Dimensionality: Used a last encoder layer with a total of 256 filters.
- Configuration: DBSCAN eps should be 5 and min_samples should be 5
- Neither batch normalization nor weight decay was applied to the present model, but it could be tested in later experiments.
- After every encoder/decoder block, 30% of all images are randomly dropped out to avoid overfitting.

They were set to ensure that each value affects time spent learning, clustering output and memory saving as little as possible.

5 | Model Parameter Counting

- Custom Autoencoder:
  - The total number of parameters comes to ∼335,000.
  - All layers within the network can be modified.
- ResNet50 (Pretrained):
  - The model is trained on a set of parameters totaling 23 million (set as a frozen extractor for features).
  - Since custom Autoencoders have only a few parameters, they are much quicker to process for custom algorithms compared to ResNet50.

6 | Regularization Techniques Used

- Dropout: Each max-pooling and upsampling layer in the autoencoder is followed by a dropout layer (rate = 0.3).
- BatchNormalization: This technique is unnecessary to avoid making the architecture complex.
- Weight Decay / L2 Regularization: No, weight decay or L2 regularization is not being used, yet investigating them could help improve the results.

7 | Clustering Results Comparison

Method	Silhouette Score	Davies-Bouldin Index
Autoencoder + KMeans	0.0294	3.2697
ResNet50 + KMeans	0.0404	3.9020

Table 7.1: Comparison of clustering performance metrics

- K-means performed good , compared to the pre-trained model Resnet, DBSCAN found only one cluster, making it incompatible to evaluate.
- ResNet50, despite being pre trained on ImageNet, did not yield high clustering performance in this context, likely due to the domain gap and lack of fine-tuning.

8 | Clustering Accuracy Estimation in Unsupervised Learning

Since no labels were used during training, evaluation was done using unsupervised metrics:

8.1 Internal Clustering Metrics

We used the following internal metrics to evaluate the clustering results:

8.1.1 Silhouette Score

The Silhouette Score measures how similar each sample is to its own cluster (cohesion) compared to other clusters (separation). It ranges from -1 to +1:

- +1: Strong clustering (well-separated and cohesive)
- 0: Overlapping clusters or ambiguous boundaries
- -1: Poor clustering (likely misclassified)

8.1.2 Davies-Bouldin Index (DBI)

The DBI measures average similarity between clusters — lower values are better. A lower DBI indicates tighter clusters that are well-separated from one another.

8.2 Qualitative Evaluation via Visualization (t-SNE)

To further evaluate clustering effectiveness, we visualized the learned feature space using t-SNE (t-Distributed Stochastic Neighbor Embedding) — a dimensionality reduction technique that maps high-dimensional embeddings into a 2D space while preserving the local structure of the data.

8.2.1 Autoencoder + KMeans Results

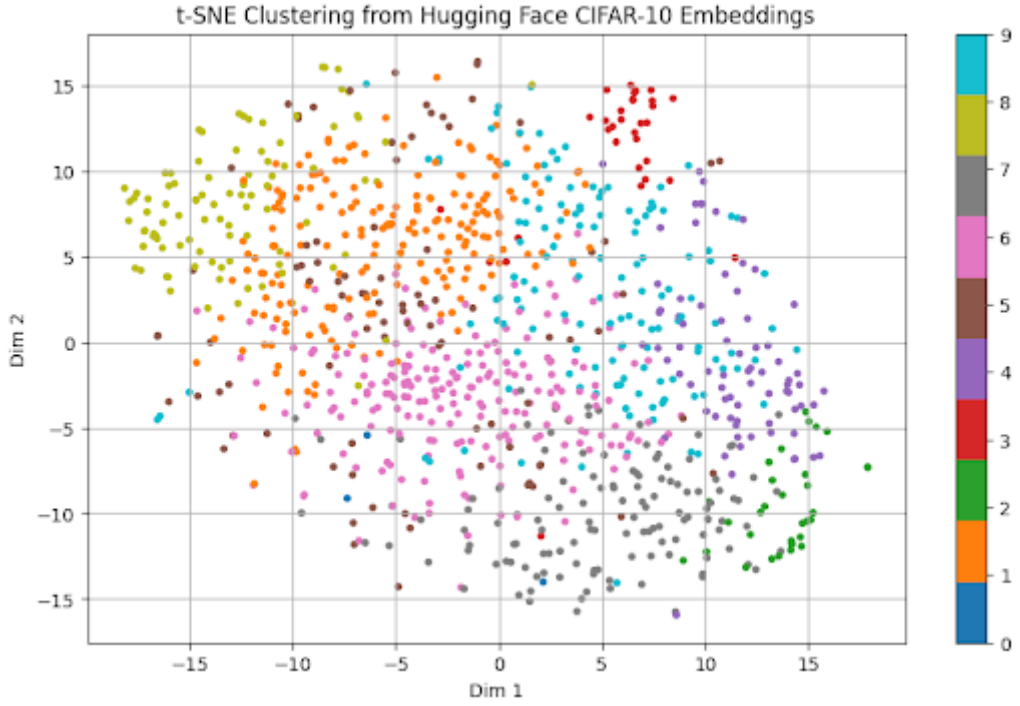

Figure 8.1: t-SNE visualization of Autoencoder + KMeans clustering

A Silhouette Score of 0.0294 indicates that the clusters heavily overlap, and many points may have been assigned to incorrect clusters or are too close to cluster boundaries.

A DB Index of 3.2697 is relatively high, which further suggests that the clusters are not well-separated or compact.

These values imply that the embeddings learned by the autoencoder at this stage were not optimal for clustering using K-Means — likely due to high dimensionality, insufficient structure in the embedding space, or lack of cluster-friendly separation in the latent features.

8.2.2 Autoencoder + DBSCAN Results

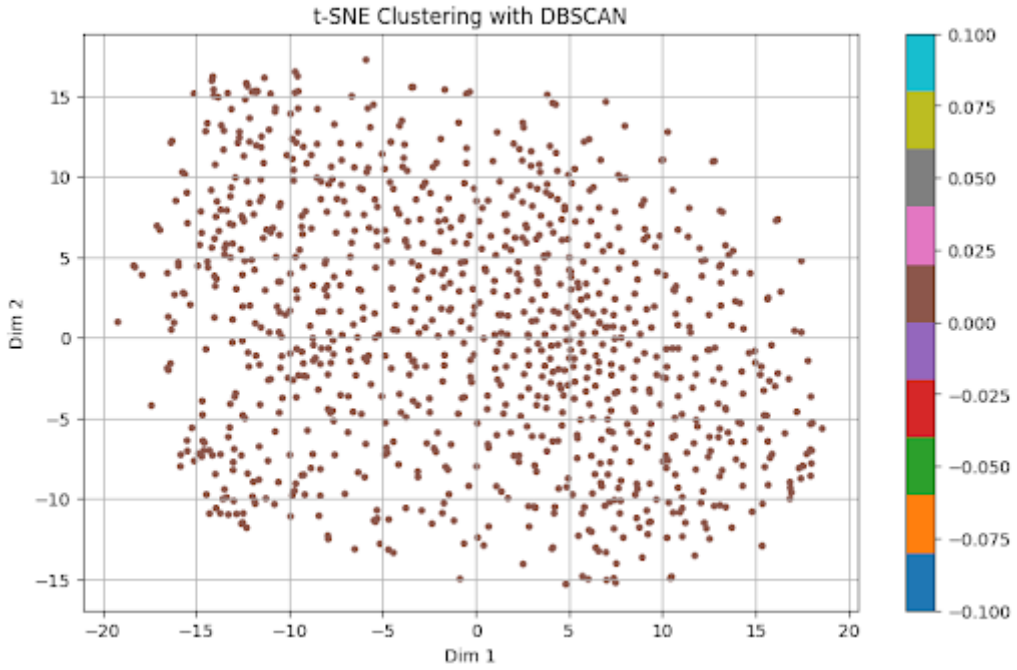

Figure 8.2: t-SNE visualization of Autoencoder + DBSCAN clustering

This visualization shows how the DBSCAN algorithm grouped the embeddings into a single large cluster, indicating that the density-based clustering approach was not effective in separating the data points into meaningful groups.

8.2.3 ResNet50 + KMeans Results

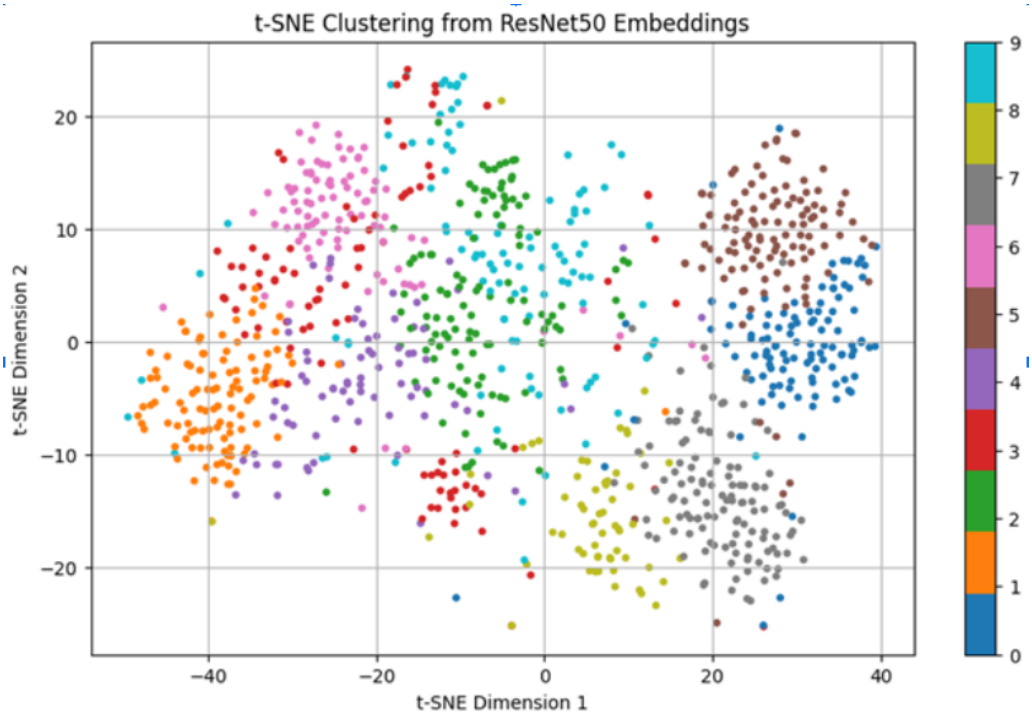

Figure 8.3: t-SNE visualization of ResNet50 + KMeans clustering

In contrast, the ResNet50 features — while extracted from a powerful pretrained model — led to overlapping and unclear clusters, as shown in the t-SNE visualization. The visualization showed distinct color-coded clusters (numbered 0-9), but with significant overlap between them, particularly in the central region of the plot. This reflects the difficulty of directly applying pretrained features to unsupervised tasks without domain-specific fine-tuning.

8.3 Conclusion on Evaluation

In the absence of labeled data, a combination of internal metrics and visual analysis was used to validate the clustering effectiveness. The DBSCAN clustering on autoencoder embeddings failed to produce multiple clusters. This may be attributed to a suboptimal value or high dimensionality of the embeddings. Despite not yielding quantitative scores, this outcome highlights the importance of parameter sensitivity in density-based clustering methods. Moreover, the custom autoencoder + K-Means combination produced the best overall balance between compactness and separability, reflected in its lower DBI (3.2697) compared to the ResNet50-based approach.

While ResNet50 + K-Means achieved a marginally higher Silhouette Score (0.0404), its higher DBI (3.9020) indicates that the clusters were poorly separated and less cohesive. This suggests that although ResNet50 is powerful for supervised classification, its learned features may not be ideal for clustering tasks without domain-specific fine-tuning.

From this comparative analysis, it is evident that the custom convolutional autoencoder, trained from scratch and optimized on the target data, is more effective at producing clusterable embeddings than the generic features extracted from ResNet50. Although the silhouette scores for all models are relatively low, the lower Davies-Bouldin Index in the autoencoder model suggests more compact and distinct clusters.

This supports the conclusion that task-specific unsupervised embedding learning via autoencoders offers more suitable representations for clustering than relying solely on off-the-shelf pre-trained CNNs.

9 | Weakness and Obstacles

- The need for extra RAM to handle ResNet and the 224x224 resize was a major issue.
  Solution: Reduce the dataset to 1000 samples.
- Autoencoder training took a lot of time.
  Solution: Dropout was used to aid in convergence at a fast pace.
- Typically, only 1 cluster is found by DBSCAN.
  Solution: I changed the eps value and compared it with K-Means.
- Without a reference set, there is no clear way to measure accuracy.
  Solution: Metrics such as clustering and t-SNE were used to explain the model findings.

10 | Conclusion

The project successfully demonstrated unsupervised learning using both a custom neural network and a pretrained CNN. Clustering was evaluated using appropriate metrics, and results showed that even lightweight autoencoders can compete with heavyweight models in certain setups. Further improvements could include label-assisted validation, data augmentation, and architecture enhancements.

Bibliography

[1] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," arXiv preprint arXiv:1312.6114, 2013.

[2] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," JMLR, 2011.

[3] Hugging Face Datasets Library – https://huggingface.co/datasets/uoft-cs/cifar10

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," CVPR 2016.