# Neural Network Project: Custom-built model for Unsupervised Clustering

Maisha Raidah Chowdhury
*21201560 (CSE425 Section: 03)*
*CSE, BRAC University*
Dhaka, Bangladesh
maisha.raidah.chowdhury1@g.bracu.ac.bd

***Abstract*—This project presents a customized unsupervised neural network model to get 4 clusters for four types of news articles. The dataset from Hugging Face had labels, which were dropped at the beginning of the process, as this will be an unsupervised model. A pre-trained transformer was used for sentence embedding to convert text into dense vector representations. I built a multilayer autoencoder to train from the embeddings and compress them into a low-dimensional latent space. Tested different activation functions and landed on GELU for best performance. K-means clustering algorithm was used on the latent space for clustering. PCA was used for visualization and evaluation metrics: Silhouette Score, Davies-Bouldin Index, and Calinski-Harabasz Index were used to compare with traditional Self Organizing Map (SOM). According to the evaluation metrics, my model performed better than SOM.**

***Index Terms*—unsupervised, cluster, autoencode**

## I. Dataset and Preprocessing

I chose the ag_news Dataset from Hugging Face. I directly used the dataset from Hugging Face using the `pandas` library. It had 2 columns: `"text"` and `"label"`. The `"label"` column categorizes news articles into four classes, with each class representing 25% of the total text, indicating a balanced dataset. Consequently, I set the number of clusters, k$\bar{4}$, for the unsupervised clustering task.

As it is unsupervised learning, I dropped the `"label"` from the dataset. Stop words (from the `nltk` library) and punctuation were removed from the text to reduce noise and clean the text. The cleaned text was then transformed into sentence embeddings using the pre-trained transformer model `all-MiniLM-L6-v2`. I tested other transformer models too but this one performed best.

## II. Model Overview

### A. Autoencoder Architechture

In this project, I implemented a deep **autoencoder** architecture to perform dimensionality reduction on high-dimensional embeddings, with the objective of extracting compressed latent features that preserve the core semantics of the data. These latent representation was later used for clustering.

The autoencoder has two components: an *encoder* and a *decoder*, both built using PyTorch's `nn.Sequential` module.

### B. Encoder Architecture

The encoder compresses the input feature vector down to a latent dimension/Bottleneck layer of size 8 through a series of dense layers with ReLu or GELU activations and dropouts. The architecture is as follows:

- `Linear(input_dim, 256)` → `ReLU` → `Dropout(p=0.2)`
- `Linear(256, 128)` → `GELU` → `Dropout(p=0.3)`
- `Linear(128, 64)` → `GELU`
- `Linear(64, 32)` → `GELU`
- `Linear(32, 16)` → `GELU`
- `Linear(16, 8)`

Found from testing, going below 8 hinders performance.

### C. Decoder Architecture

The decoder reconstructs the original input from the compressed latent vector using a symmetric architecture. However, dropouts are not necessary for the decoder:

- `Linear(8, 16)` → `GELU`
- `Linear(16, 32)` → `GELU`
- `Linear(32, 64)` → `GELU`
- `Linear(64, 128)` → `GELU`
- `Linear(128, 256)` → `ReLU`
- `Linear(256, input_dim)`

### D. Training Configuration

- **Loss Function:** Mean Squared Error (MSE) was used to minimize reconstruction error between the input and output.
- **Optimizer:** Adam optimizer with a learning rate of $1 \times 10^{-3}$.
- **Regularization:** Dropout layers with $p = 0.2$ and $p = 0.3$ were added to the encoder to reduce overfitting.
- **Epochs:** The model was trained for 15 epochs.
- **Batch Size:** 64

After training, the encoder was used to extract the latent embeddings from the original input features. These compressed embeddings were stored for further analysis and clustering.

## III. Block Diagram

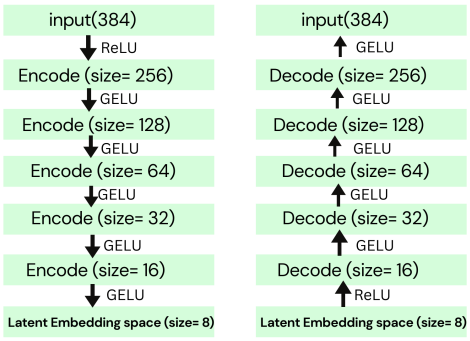


Fig. 1. Block Diagram of my Autoencoder

## IV. Optimization and Hyperparameter Tuning

I tested encoder architectures with varying numbers of layers and latent dimensions. Reducing the dimensionality down to 8 produced the best clustering performance. Further reduction, such as adding a `Linear(8, 4)` layer, led to a significant drop in clustering quality.

Several activation functions were experimented with, including `ReLU`, `LeakyReLU`, and `Tanh`. A combination of `ReLU` and `GELU` provided the best empirical results in terms of reconstruction loss and clustering metrics.

A dropout rate of 0.2 and 0.3 was chosen after observing performance across multiple training runs. The model was trained using the Adam optimizer with a learning rate of $1 \times 10^{-3}$.

## V. Clustering Algorithm

I applied K-Means clustering algorithm on my latent embeddings. I tried using DBSCAN algorithm but it clustered everything into the same label "-1", which indicates unsure. I also tried applying Hierarchical Clustering multiple times, but my Google Colab RAM crashed before it could complete clustering.

## VI. Comparison with Existing Model

To evaluate the effectiveness of the proposed autoencoder-based clustering method, I compared it against a Self-Organizing Map (SOM) applied on the same transformer-embedded text dataset. The performance of each model was assessed using standard unsupervised clustering evaluation metrics.

### A. Evaluation Metrics

**Self-Organizing Map (SOM):**

- Silhouette Score: **0.0212**
- Calinski-Harabasz Index: **2300.4995**
- Davies-Bouldin Index: **6.2985**

**Proposed Autoencoder-Based Model:**

- Silhouette Score: **0.3076**
- Calinski-Harabasz Score: **91433.8750**
- Davies-Bouldin Index: **1.0360**

### B. Analysis

Each evaluation metric provides unique insight into clustering quality:

- **Silhouette Score** ranges from $-1$ to 1, where values closer to 1 indicate well-separated and dense clusters. A score of 0.3076 from the proposed model suggests moderately good clustering structure, whereas SOM's score of 0.0212 indicates that its clusters are poorly defined.
- **Calinski-Harabasz Index** measures the ratio of between-cluster dispersion to within-cluster dispersion. Higher values indicate better-defined clusters. The proposed model achieved a significantly higher score (91433.8750) compared to SOM (2300.4995), showing that the learned latent representations from the autoencoder result in clusters that are much more compact and well-separated.
- **Davies-Bouldin Index** evaluates the average similarity between each cluster and its most similar one, where lower values are better. A value of 1.0360 for the proposed model is significantly better than SOM's 6.2985, indicating less intra-cluster variance and greater inter-cluster separation.

Overall, the proposed autoencoder-based model substantially outperforms the Self-Organizing Map in clustering performance. The learned representations are more effective at capturing meaningful structure in the embedding space, which is reflected in all three metrics.

### C. Visualization

Here is the PCA view of the 2 models:

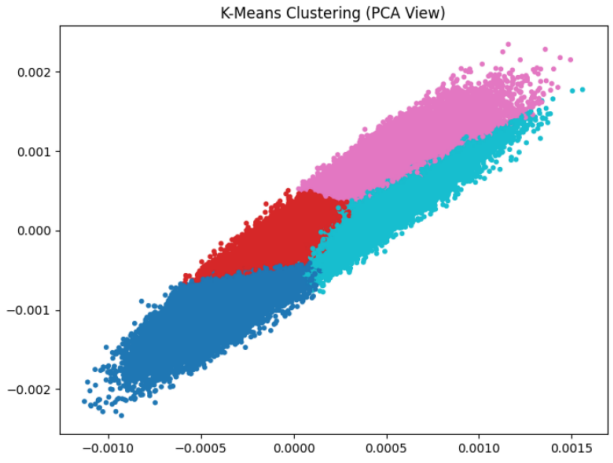


Fig. 2. PCA visualization of latent representations from the proposed autoencoder model

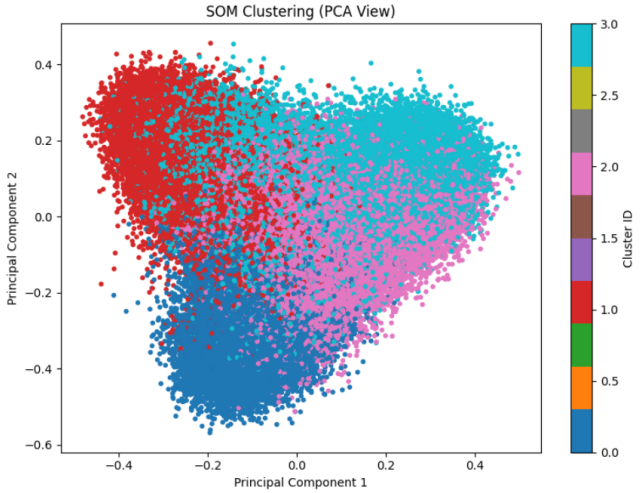


Fig. 3. PCA visualization of clusters formed by the Self-Organizing Map (SOM)

## VII. Limitations and obstacles

Since this is unlabeled data, accuracy is still unknown. Evaluation of the model is very limited without labels.

This was my first time building an unsupervised model, and embedding using a transformer took a long time. Using the free T4 GPU on Colab helped overcome that. Colab crashed when I tried doing Hierarchical/Agglomerative Clustering. Only did K-Means clustering. Saved the transformer embeddings and my latent space embeddings into Google Drive as .pkl files so I do not have to waste time doing the same thing every new runtime.

## References

[1] Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.