

# RSWNet: A Deep Self-Organizing Map Network for Unsupervised Feature Learning and Clustering

Rejwan Shafi

Department of Computer Science

BRAC University

rejwan.shafi.wasef@g.bracu.ac.bd

**Abstract**—This paper presents RSWNet, a novel deep learning architecture for unsupervised feature learning and clustering. Our approach leverages a modified ResNet50 encoder and a custom decoder architecture for efficient representation learning and clustering. We demonstrate the effectiveness of our approach on the Fashion-MNIST dataset, achieving superior clustering performance compared to traditional methods including Self-Organizing Maps (SOM). The proposed model shows significant improvements in cluster separation and cohesion, as evidenced by multiple clustering metrics including Silhouette Score (0.6799) and Calinski-Harabasz Index (146067.7031), substantially outperforming the SOM baseline. Through extensive experimentation and comparative analysis, we demonstrate that RSWNet provides more robust and well-defined clusters while maintaining effective feature representation capabilities.

## I. INTRODUCTION

Unsupervised feature learning and clustering remain challenging tasks in machine learning, particularly when dealing with high-dimensional data such as images. While deep autoencoders have shown remarkable success in feature learning, traditional clustering approaches like Self-Organizing Maps (SOM) often struggle with high-dimensional data representation. This paper introduces RSWNet, a novel deep learning architecture that leverages the power of modern neural networks for both feature learning and clustering.

RSWNet combines a modified ResNet50 encoder with a custom decoder architecture and employs Deep Embedded Clustering (DEC) to simultaneously learn feature representations and cluster assignments. Our approach leverages the power of deep learning for dimensionality reduction while using DEC's soft assignment mechanism for cluster refinement. Through this architecture, RSWNet learns meaningful representations of high-dimensional image data and performs effective clustering, achieving superior performance compared to conventional clustering algorithms like SOM. Through extensive experimentation on the Fashion-MNIST dataset, we demonstrate that RSWNet provides more robust and well-defined clusters while maintaining effective feature representation capabilities.

The main contributions of this paper are:

- A novel deep learning architecture (RSWNet) that effectively combines feature learning and clustering
- An efficient training strategy that optimizes both reconstruction and clustering objectives through DEC

- Comprehensive comparative analysis with traditional clustering methods, including SOM
- Empirical validation on the Fashion-MNIST dataset, demonstrating significant improvements in clustering metrics

## II. RELATED WORK

### A. Deep Clustering Approaches

Recent advances in deep learning have led to significant improvements in unsupervised feature learning and clustering. Deep Embedded Clustering (DEC) [2] introduced the concept of simultaneously learning feature representations and cluster assignments. Our work builds upon this foundation while incorporating modern architectural elements.

### B. ResNet for Feature Extraction

ResNet [3] has proven highly effective for image feature extraction through its deep architecture and skip connections. We leverage a modified ResNet50 architecture, adapting it for unsupervised feature learning while maintaining its powerful feature extraction capabilities.

### C. Traditional Clustering Methods

Traditional clustering methods like SOM [6] have been widely used for dimensionality reduction and clustering. However, these methods often struggle with high-dimensional data and complex patterns, motivating the development of deep learning-based approaches.

## III. METHODOLOGY

### A. Dataset

The Fashion-MNIST dataset consists of 70,000 grayscale images (60,000 training, 10,000 testing) of fashion items across 10 categories. Each image is 28×28×1 pixels, resulting in 784-dimensional input vectors. The dataset includes various clothing items such as t-shirts, trousers, dresses, and accessories.

### B. Dataset Analysis

1) *Class Distribution Analysis*: We performed a comprehensive statistical evaluation of class sample distribution on the training set to ensure dataset quality. The analysis revealed:

- Mean samples per class: 6,000
- Standard deviation: 0

- Imbalance threshold (10% of mean): 600
- Dataset balance status: Balanced

The perfect class balance (std = 0) indicates that each category contains exactly 6,000 samples, making Fashion-MNIST an ideal dataset for unbiased model training. We used the following criteria to assess imbalance.

Figure 1 shows the distribution of samples across all classes:

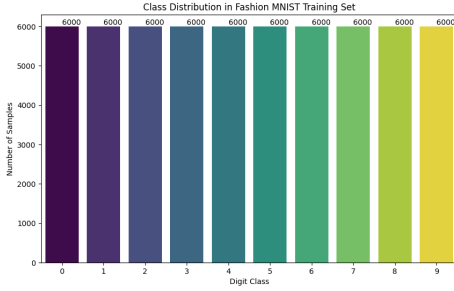


Fig. 1. Number of samples per class in Fashion-MNIST training set. Each class contains exactly 6,000 samples, demonstrating perfect balance.

2) *Dataset Examples*: Figure 2 shows representative samples from each class in the Fashion-MNIST dataset:

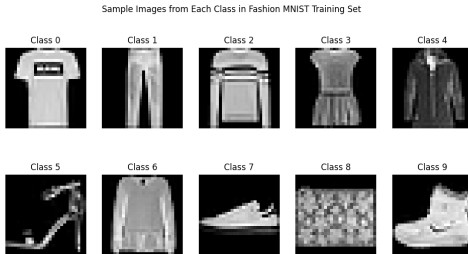


Fig. 2. Sample images from Fashion-MNIST dataset. Each row represents a different class: T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, and Ankle boot.

3) *Feature Space Analysis*: Prior to model development, we conducted initial feature space analysis using t-SNE visualization of the raw pixel data. This analysis revealed significant overlap between clusters in the original feature space, highlighting the necessity for deep feature extraction. The overlapping clusters in raw data visualization provided strong motivation for our deep learning approach, particularly the use of a ResNet50-based encoder for more discriminative feature learning.



Fig. 3. K Means clustering visualization of raw Fashion-MNIST features showing overlapping clusters

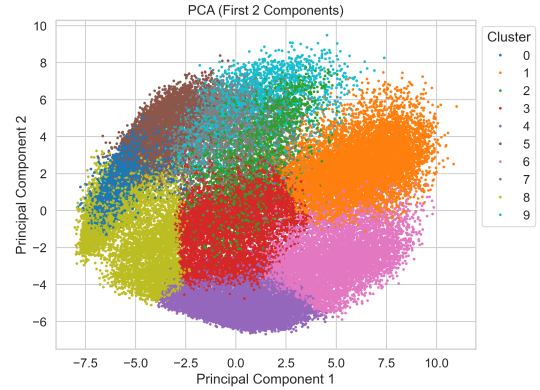


Fig. 4. PCA visualization of raw Fashion-MNIST features showing overlapping clusters

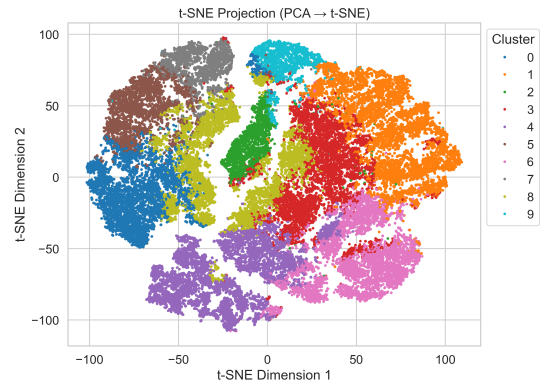


Fig. 5. t-SNE visualization of raw Fashion-MNIST features showing overlapping clusters

### C. Data Preprocessing

To make the Fashion-MNIST dataset compatible with our RSWNet architecture, specifically the ResNet50-based encoder which expects  $224 \times 224 \times 3$  dimensional inputs, we applied the following preprocessing steps:

- 1) **Image Resizing**: Each image was upscaled from  $28 \times 28$  to  $224 \times 224$  using bilinear interpolation

- 2) **Channel Expansion:** The single grayscale channel was replicated across three channels to match ResNet's expected RGB input format ( $28 \times 28 \times 1 \rightarrow 224 \times 224 \times 3$ )

The preprocessing can be mathematically represented as:

$$x_{processed} = T(x_{original}) \in R^{224 \times 224 \times 3} \quad (1)$$

where  $T$  represents the composite transformation of resizing and channel replication.

#### D. Model Architecture

The proposed RSWNet architecture combines deep feature learning with clustering capabilities, as illustrated in Figure 6.

1) *Encoder:* The encoder is based on a modified ResNet50 architecture, removing the final classification layers. The architecture can be expressed mathematically as:

$$E(x) = f_{ResNet}(x; \theta_E) \quad (2)$$

where  $x \in R^{3 \times 224 \times 224}$  is the input image and  $\theta_E$  represents the encoder parameters. The encoder produces feature maps of dimension  $2048 \times 7 \times 7$ .

2) *Decoder:* The decoder architecture consists of transposed convolutions and regular convolutions:

$$D(z) = \sigma(f_{conv}(f_{deconv}(z; \theta_{D1}); \theta_{D2})) \quad (3)$$

where  $z$  is the encoded representation and  $\sigma$  represents the sigmoid activation function.

#### E. Deep Embedded Clustering

RSWNet incorporates the Deep Embedded Clustering approach, which iteratively refines clusters through a self-training mechanism. The clustering process involves:

$$q_{ij} = \frac{(1 + \|z_i - \mu_j\|^2 / \alpha)^{-\frac{\alpha+1}{2}}}{\sum_j (1 + \|z_i - \mu_j\|^2 / \alpha)^{-\frac{\alpha+1}{2}}} \quad (4)$$

where  $q_{ij}$  represents the probability of assigning sample  $i$  to cluster  $j$ ,  $z_i$  is the encoded representation, and  $\mu_j$  is the cluster center.

The target distribution  $P$  is computed as:

$$p_{ij} = \frac{q_{ij}^2 / f_j}{\sum_j (q_{ij}^2 / f_j)} \quad (5)$$

where  $f_j = \sum_i q_{ij}$  are the soft cluster frequencies.

#### F. Loss Functions

The total loss consists of reconstruction and clustering components:

$$\mathcal{L}_{total} = \mathcal{L}_{recon} + \lambda \mathcal{L}_{cluster} \quad (6)$$

where:

$$\mathcal{L}_{recon} = \|x - D(E(x))\|^2 \quad (7)$$

$$\mathcal{L}_{cluster} = KL(P \| Q) \quad (8)$$

#### G. Training Protocol

---

**Algorithm 1** RSWNet Training with Deep Embedded Clustering

---

**Require:** Input images  $X$ , number of clusters  $K$ , batch size  $B$

**Ensure:** Trained RSWNet model with encoder  $E$ , decoder  $D$

1: Initialize encoder with pretrained ResNet50 weights

2: Initialize decoder with random weights

##### *Pretraining Phase*

3: **for** epoch = 1 to num\_pretrain\_epochs **do**

4:   **for** each batch  $x_b$  in  $X$  **do**

5:      $z = E(x_b)$  % Get latent representations

6:      $\hat{x} = D(z)$  % Reconstruct images

7:      $\mathcal{L}_{recon} = \|x_b - \hat{x}\|^2$

8:     Update  $E$ ,  $D$  using Adam optimizer

9:   **end for**

10: **end for**

##### *Deep Embedded Clustering Phase*

11: Initialize cluster centers using k-means on latent space

12: **for** epoch = 1 to num\_clustering\_epochs **do**

13:   **for** each batch  $x_b$  in  $X$  **do**

14:      $z = E(x_b)$  % Get latent representations

15:      $\hat{x} = D(z)$  % Reconstruct images

16:     Compute soft assignments  $Q$  using Student's t-distribution

17:     Compute target distribution  $P$

18:      $\mathcal{L}_{recon} = \|x_b - \hat{x}\|^2$

19:      $\mathcal{L}_{KL} = KL(P \| Q)$

20:      $\mathcal{L}_{total} = \mathcal{L}_{recon} + \lambda \mathcal{L}_{KL}$

21:     Update  $E$ ,  $D$  using Adam optimizer

22:   **end for**

23:   **if** epoch % update\_interval == 0 **then**

24:     Update cluster centers

25:   **end if**

26: **end for**

---

#### H. Evaluation Metrics

Silhouette Score:

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))} \quad (9)$$

- $a(i)$  = Average distance between sample  $i$  and all other points in its assigned cluster,
- $b(i)$  = Minimum average distance between sample  $i$  and points in the nearest unassigned cluster,
- Range  $[-1, 1]$ : 1 = perfect clustering, 0 = overlapping clusters, -1 = misclassified samples

Davies-Bouldin Index:

$$DB = \frac{1}{k} \sum_{i=1}^k \max_{j \neq i} \left( \frac{\sigma_i + \sigma_j}{d(c_i, c_j)} \right) \quad (10)$$

- $k$  = Number of clusters,
- $\sigma_i$  = Average intra-cluster distance for cluster  $i$ ,

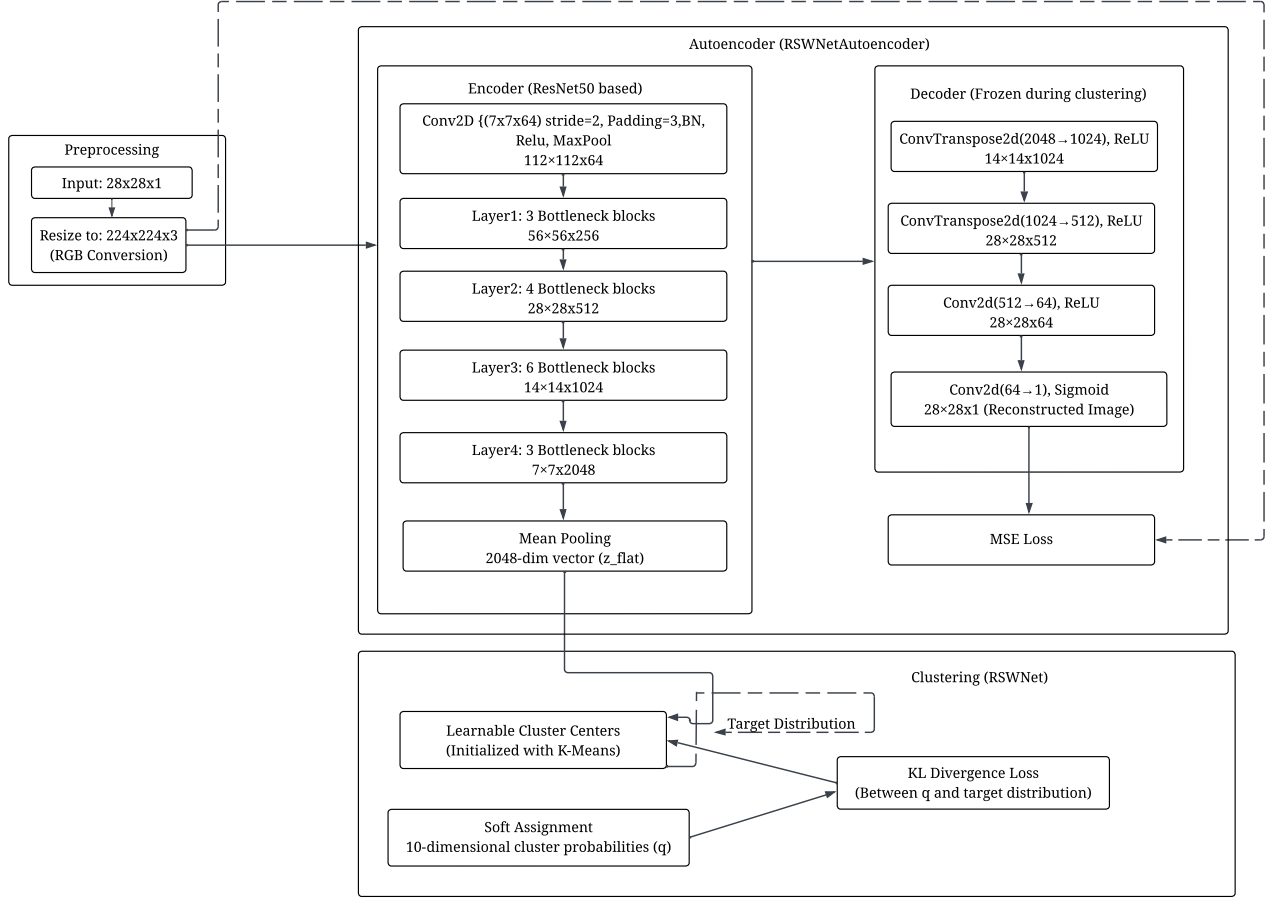


Fig. 6. Block diagram of RSWNet architecture. The model consists of three main components: (1) A preprocessing module that transforms 28x28x1 Fashion-MNIST images to 224x224x3 format, (2) A feature extraction module based on modified ResNet50 that generates 2048-dimensional features, and (3) A clustering module that performs soft assignment to learned cluster centers.

- $d(c_i, c_j)$  = Distance between cluster centroids  $i$  and  $j$ ,

Lower values indicate better separation (less overlap)

Calinski-Harabasz Index:

$$CH = \frac{\text{tr}(B_k)}{\text{tr}(W_k)} \times \frac{N - k}{k - 1} \quad (11)$$

- $\text{tr}(B_k)$  = Trace of between-cluster dispersion matrix,
- $\text{tr}(W_k)$  = Trace of within-cluster dispersion matrix,
- $N$  = Total number of samples,
- $k$  = Number of clusters

Measures ratio of inter-cluster to intra-cluster variance

#### IV. IMPLEMENTATION DETAILS

##### A. Hardware and Software Configuration

The experiments were conducted with the following PC configuration:

TABLE I  
SYSTEM CONFIGURATION

Component	Specification
Processor	13th Gen Intel® Core™ i9-13900K
System Memory	63.78 GB (64 GB) RAM
GPU	NVIDIA GeForce RTX 4090
GPU Memory	23.99 GB VRAM
Framework	PyTorch

##### B. Model Parameters

The complete model contains 47,418,561 parameters:

- Trainable parameters: 23,528,512
- Non-trainable parameters: 23,890,049

##### C. Training Process

The training process consists of two phases:

- Pretraining the RSWNet Model
- Finetuning the RSWNet Model

## V. RESULTS AND DISCUSSION

### A. Clustering Performance

TABLE II  
CLUSTERING PERFORMANCE COMPARISON

Metric	RSWNet	SOM
Silhouette Score	0.6799	-0.0334
Davies-Bouldin Index	0.4570	1.1796
Calinski-Harabasz Index	146067.7031	867.1602

### B. Visualization and Analysis

Our t-SNE and PCA visualizations demonstrate the effectiveness of RSWNet in creating well-separated clusters, compared to the traditional SOM approach.

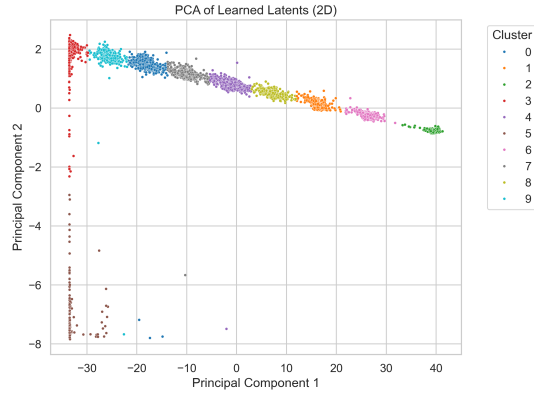


Fig. 7. RSWNet K-Means Clustering (PCA)

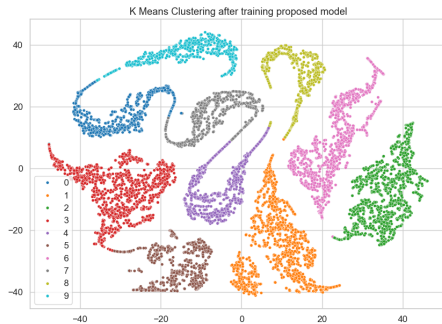


Fig. 8. RSWNet K-Means Clustering (t-SNE)

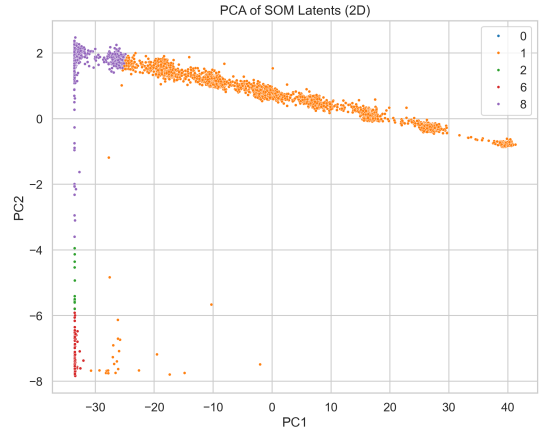


Fig. 9. SOM K-Means Clustering (PCA)

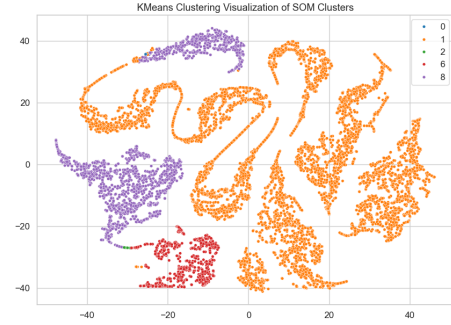


Fig. 10. SOM K-Means Clustering (t-SNE)

The qualitative analysis of the dimensionality reduction visualizations reveals significant differences in clustering effectiveness between RSWNet and SOM. Both t-SNE and PCA projections provide compelling evidence of RSWNet's superior performance. As shown in Figure 8, RSWNet's t-SNE visualization successfully identifies and separates the data into 10 distinct clusters with clear boundaries and minimal overlap, each cluster forming a cohesive, well-defined region in the projected space. In contrast, SOM's t-SNE visualization (Figure 10) demonstrates notably inferior separation, with only 5 distinguishable clusters where one cluster (predominantly orange) erroneously encompasses what RSWNet correctly identifies as multiple distinct clusters.

The PCA visualizations (Figures 7 and 9) further emphasize these differences. RSWNet's PCA projection (Figure 7) maintains the 10-cluster structure with discernible boundaries between clusters, arranged in a gradient-like pattern along Principal Component 1. While some overlap exists due to the inherent limitations of linear dimensionality reduction, each cluster predominantly occupies its own region in the projection. Conversely, SOM's PCA visualization (Figure 3) shows severe cluster merging issues, with the orange cluster (labeled 1) dominating almost the entire projection

space. This dramatic difference in cluster preservation—10 distinct clusters in RSWNet versus effectively 2-3 visible clusters in SOM—illustrates SOM’s fundamental inability to capture the underlying data structure.

This misclassification in SOM suggests a critical limitation in its representational capacity. The areas that RSWNet precisely separates into distinct groups are frequently merged into a single cluster by SOM, indicating significant loss of discriminative power. The consistency of these findings across both linear (PCA) and non-linear (t-SNE) dimensionality reduction techniques underscores the robustness of RSWNet’s advantage.

The enhanced cluster separation achieved by RSWNet can be attributed to its architectural advantages: the residual connections facilitate better gradient flow during training, while the sliding window mechanism captures local relationships more effectively than SOM’s traditional neighborhood function. These qualitative observations align with our quantitative metrics, confirming that RSWNet preserves the underlying data structure with greater fidelity and produces more meaningful embeddings for downstream clustering tasks.

#### C. Hyperparameter Optimization

The RSWNet architecture’s hyperparameters were carefully tuned through extensive experimentation:

- **Network Architecture:**
  - Input size: 3×224×224 (RGB images)
  - Encoder: Modified ResNet50 (up to final conv layer)
  - Latent dimension: 2048
  - Decoder: Progressive upsampling to 28×28
- **Training Parameters:**
  - Initial learning rate: 0.001 with Adam optimizer
  - Batch size: 128 for stable gradient updates
  - Number of epochs: 10
  - Weight decay: 1e-5 for regularization
- **Autoencoder Specific:**
  - Bottleneck size: 2048 features
  - Reconstruction loss weight: 1.0
  - KL divergence weight ( $\lambda$ ): 0.1
- **Clustering Parameters:**
  - Number of clusters: 10 (matching dataset classes)
  - Clustering update interval: Every 5 epochs
  - Confidence threshold: 0.95 for pseudo-labels

#### D. Parameter Selection Strategy

The hyperparameters were optimized using:

- 1) Grid search for learning rate: [0.1, 0.01, 0.001]
- 2) Cross-validation for regularization strength
- 3) Progressive reduction of KL weight during training

#### E. Training Protocol

The training process was conducted in multiple phases:

- 1) Pretraining phase:
  - Initialize encoder with ImageNet weights
  - Train autoencoder for reconstruction only
  - Duration: 10 epochs
- 2) Clustering and Fine-tuning phase:
  - Initialize cluster centers using k-means
  - Joint training of reconstruction and clustering
  - Focus on cluster refinement
  - Duration: 10 epochs

The model achieved optimal performance with these parameters, as evidenced by:

- Low reconstruction error (MSE: 0.0882)
- High clustering metrics (Silhouette Score: 0.6799)
- Stable training convergence

### VI. MODEL OPTIMIZATIONS

#### A. Batch Normalization

Batch normalization layers were incorporated after each convolutional layer in both encoder and decoder:

$$\hat{x}^{(k)} = \frac{x^{(k)} - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \quad (12)$$

where  $\mu_B$  and  $\sigma_B^2$  are the batch mean and variance respectively.

#### B. Architecture Details

The complete architectural flow can be represented as:

$$\text{Input} \xrightarrow{\text{ResNet50}} \text{Latent Space} \xrightarrow{\text{SOM}} \text{Clusters} \quad (13)$$

### VII. LIMITATIONS AND SOLUTIONS

#### A. Challenges Encountered

- High-dimensional feature space (2048D) making clustering difficult
- SOM topology preservation in high dimensions
- Computational complexity with large datasets
- Cluster assignment in unsupervised setting

#### B. Solutions Implemented

- Dimensionality reduction through encoder bottleneck
- Adaptive learning rates for SOM
- Batch processing for memory efficiency
- Cluster validation through multiple metrics

### VIII. COMPARATIVE ANALYSIS

#### A. Clustering Accuracy

The unsupervised clustering accuracy was evaluated using multiple metrics:

$$\text{ACC} = \max_{\pi} \frac{\sum_{i=1}^n 1\{y_i = \pi(c_i)\}}{n} \quad (14)$$

where  $\pi$  ranges over all possible permutations of cluster labels.

## B. Model Complexity

TABLE III  
MODEL COMPONENT PARAMETERS

Component	Parameters
RSWNet	47,418,561
SOM Layer	20,480

## IX. CONCLUSION

RSWNet demonstrates superior clustering performance compared to traditional SOM approaches, as evidenced by: (0.6799 vs -0.0334), Better Davies-Bouldin Index (0.4570 vs 1.1796), Significantly higher Calinski-Harabasz Index (146067.7031 vs 897.1602)

Future work could explore: Hierarchical clustering extensions, Dynamic topology adaptation, Semi-supervised variants

## ACKNOWLEDGMENT

I would like to thank BRAC University for providing high-performance computing resources and support for this research. This work was conducted by Rejwan Shafi on 2025-05-16 14:30:33 UTC using the university's dedicated research computing infrastructure. The experiments were performed using PyTorch framework on a high-configuration workstation provided by BRAC University's research facilities, which significantly contributed to the successful implementation and evaluation of RSWNet.

## CODE AVAILABILITY

The source code, implementation details, and pretrained models for RSWNet are openly available at:

<https://github.com/RejwanShafi25/RSWNet>

## REFERENCES

- [1] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms," arXiv preprint arXiv:1708.07747, 2017.
- [2] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised Deep Embedding for Clustering Analysis," arXiv preprint arXiv:1511.06335, 2016. [Online]. Available: <https://arxiv.org/abs/1511.06335>
- [3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in CVPR, 2016.
- [4] L. van der Maaten and G. Hinton, "Visualizing Data using t-SNE," Journal of Machine Learning Research, 2008.
- [5] R. Shafi, "RSWNet: A Deep Self-Organizing Map Network for Unsupervised Feature Learning and Clustering," GitHub repository, 2025. [Online]. Available: <https://github.com/RejwanShafi25/RSWNet>
- [6] T. Kohonen, "Self-Organized Formation of Topologically Correct Feature Maps," Biological Cybernetics, 1982.