Pakistan Journal of Engineering and Technology
ISSN (Print): 2664-2042; ISSN (Online): 2664-2050
Received: 2025-03-21
Accepted: 2025-05-08
https://doi.org/10.51846/vol8iss1pp47-54

PAKJET
Pakistan Journal of Engineering & Technology

Article

Performance Analysis of SpectralNet Algorithm on Fashion MNIST and KMNIST Datasets: A Study on Clustering Efficiency and Resource Utilization

Zenab Bibi and Mujeeb Ur Rehman *

Department of Computer Science, University of Management and Technology, Sialkot, Pakistan
* Correspondence: Mujeeb Ur Rehman (mujeeb.rehman@skt.umt.edu.pk)

Abstract: Unsupervised learning faces an essential data clustering challenge in high dimensions where SpectralNet stands as an effective approach which combines deep learning with spectral clustering methods. A performance evaluation of SpectralNet measures its results on Fashion MNIST and KMNIST with accuracy as well as computational costs and resource demands under multiple hyperparameter configurations. The investigation examines how generalization changes with various distribution scenarios through the assessment of balanced versus unbalanced dataset splits. Higher embedding dimension values lead to superior clustering precision, whereas it demands increased processor capacity. The research reveals how SpectralNet handles accuracy-efficiency relationships in dataset analysis to demonstrate its practical capabilities for complex information.

Keywords: Deep Learning, SpectralNet, Clustering, Balanced Dataset, MNIST

1. Introduction

1.1. Background and Motivation

The main function of unsupervised learning in its innovative field consists of identifying hidden data structures by utilizing clustering techniques. K-means clustering, along with other traditional methods [1] struggles to apply to high-dimensional databases because they require linear separability for their operation [2]. The approach of spectral clustering takes data points through graph transformation before using graph Laplacian computations for clustering identification. SpectralNet extends graph embedding principles through neural network frameworks for effective learning of data structures which allows the system to scale better when processing extensive datasets. The series of Fashion MNIST and KMNIST represent distinctive clustering difficulties because Fashion MNIST contains clothing patterns and KMNIST consists of intricate Japanese characters. The study evaluates generalization changes by analyzing dataset split imbalances through [3], [4].

This study explores different hyperparameter embedding dimensions through testing their influence on system precision and operational complexity as well as resource usage levels [3], [4]. The research focuses on obtaining SpectralNet assessment results that cover various operational circumstances to create the groundwork for optimizing cluster identification in multidimensional environments. SpectralNet shapes dataset analysis through accuracy-efficiency preferences to show its functionality for analyzing complex information.

1.2. Literature Review

The clustering algorithms identify different data groups through inherent patterns using features that are not defined by human choices. The k-means grouping algorithm fails to succeed in high-dimensional datasets because Euclidean length loses its significance. The spectral clustering approach resolves the distance limitations through similarity graph construction and Laplacian matrix embedding analysis. SpectralNet improves spectral clustering through deep learning applications that optimize graph building and support big data processing [1], [2]. The algorithm proves effective with complex datasets, including Fashion MNIST and KMNIST, that serve as common metrics for image classification and clustering evaluations [3], [4]. KMNIST requires identifying Japanese characters as part of its object recognition process which differs from Fashion MNIST because it uses apparel images.

Neural network embeddings helped deep learning improve its clustering output, according to recent studies [5]. The research shows that high-dimensional dataset work requires algorithm performance optimization together with efficiency improvement through hyperparameter tuning [6], [7]. The performance evaluation of SpectralNet with both balanced and unbalanced dataset splits in this research helps identify optimal clustering model approaches for practical use as reported by [8]. The split of datasets into balanced and unbalanced subsets served to evaluate SpectralNet under different distribution conditions [3], [4]. The SpectralNet neural network required

evaluation on each dataset with multiple parameter settings to examine different embedding dimension values [2].

The essential requirement is to use models which handle both general and local data distribution patterns. The main advancement in SpectralNet comes from its capability to replicate data spectral embeddings by using neural networks. The computation of graph Laplacian eigenvectors in classical spectral clustering becomes impossible for large datasets because of its cubic time complexity. The method supports modern techniques that search for scalable alternatives to spectral decomposition methods [9], [10].

The local neighborhood relationships among data points receive benefits from SpectralNet through its inclusion of a pairwise similarity matrix. The relationships emerge through the application of cosine similarity together with Gaussian kernels and k-nearest neighbor graphs as distance metrics. The learned spectral embeddings maintain the underlying relationships because the network clusters data so that it reflects its intrinsic manifold structure. The technique proves essential in databases showing non-linear or non-convex class boundary formations since this condition usually appears in visual applications like image segmentation and document clustering [11]. The successful implementation of hyperparameter optimization stands crucial for achieving superior performance from the algorithm. The performance of spectral clustering depends strongly on learning rate along with number of epochs and embedding dimensions and dropout rate values because these parameters affect both convergence speed and generalization performance and system resource utilization. Since KMNIST data needs identification of delicate Japanese symbols it falls into the linguistic domain. Extended limitations occur for clustering models when dealing with these domain differences because they need both accurate classification and consistent functionality across different domains. A comprehensive evaluation of SpectralNet's domain generalization capabilities can be achieved by conducting tests among different data categories that include Fashion-MNIST and KMNIST and CIFAR-100 [4], [12].

The process of dimension expansion enhances the detection of complex data patterns yet simultaneously increases computation expenses and introduces possibility of performance degradation. Hyperparameter optimization produces benefits that help researchers balance the relationship between numeric precision of clusters and training speed along with resource consumption [13], [14].

The study contributes practical knowledge through its assessment of balanced versus unbalanced dataset splits. The use of balanced datasets ensures meaningful quality assessment of clustering methods because they create even distributions of classes that mimic perfect clustering scenarios. The deep clustering method SpectralNet requires additional data treatment approaches alongside regularization techniques to function adequately when processing unbalanced data according to recent findings in [15], [16].

The study adds to the body of knowledge regarding flexible clustering methods that do not belong to specific domains in modern machine learning systems. Unsupervised learning methods show practical value in detecting anomalies and providing recommendations because they process complex big data sets which suits applications in bioinformatics and IoT and recommendation platforms as reported in [17], [18].

1.3. Contribution

The research evaluates SpectralNet algorithm performance using Fashion MNIST and KMNIST datasets by adjusting an important parameter that influences accuracy levels. This research shows how modifications in algorithm execution occur across different parameter distributions from precision assessments generated through hyperparameter changes. The research evaluates model evaluation and robustness/generalizability transformations to enable the growth of spectral-based learning procedures.

2. Materials and Methods

The SpectralNet application to Fashion MNIST and KMNIST datasets starts from data import through preprocessing and then moves to model training followed by testing and ends in evaluation.

2.1. Dataset

The Fashion MNIST and KMNIST datasets should be loaded since they contain grayscale images intended for classification tasks. The analysis requires complete information provided by Fashion MNIST.

2.2. Preprocessing

The training process needs essential data preparation steps which transform raw data into training-ready condition.

2.3. Load the Dataset

The dimensions for images must be adjusted to a standard form. The process converts images into vector-form so that they become compatible with neural network usage. Caretakers can normalize pixels because the range should be between 0 and 1.

2.4. Extract Labels for Evaluation Purposes

The normalization process should be applied together with split generation at ratios from 25% to 100% in 25% increments. Testing performance across different data distributions requires training and testing splits with ratios of 50/50, 60/40, 70/30 and 80/20 and 90/10.

2.5. Model Training

The SpectralNet model receives training through different stages. Neural Network Training serves as a method to obtain the embeddings. The dataset experiences Dimensionality Reduction that transforms it into lower-dimensional space. The clustering process under Unsupervised Learning functions without requiring any labeled data.

2.6. Optimization to Refine Model Performance

The model receives Hyperparameter Tuning that allows adjustments of learning rate values and embedding dimension

values. Operate the Self-Organizing Maps (SOM) alongside Spectral Clustering techniques during implementation. A K-means clustering procedure should operate on dimensions that were reduced beforehand. Use matrices to perform eigenvalue decomposition of Laplacian matrices for achieving optimal clustering results.

2.7. Testing

The trained model should be used for testing unseen data to determine its generalization abilities. The model receives newly introduced data to classify items through its acquired features.

2.8. Evaluation

The model performance assessment should utilize these evaluation metrics: Confusion Matrix:

- Precision
- Recall
- F1 Score
- Accuracy
- Memory Usage

The methodology conducts a systematic evaluation of SpectralNet through data split tests and hyperparameter adjustments which show its ability to classify both Fashion MNIST and KMNIST high-dimensional image datasets.

Fig. 1 shows in detail how SpectralNet operates to analyze classification outcomes by processing Fashion MNIST and KMNIST datasets. The algorithm starts with importing both datasets which leads to building a full Fashion MNIST database. The dataset together with KMNIST receives preprocessing treatment that consists of normalization steps and class distribution balancing/unbalancing procedures using defined sample partitioning methods. The preprocessed data goes through the training model by completing essential operations for dataset loading while preparing input and output alongside normalization and missing value handling and label encoding.

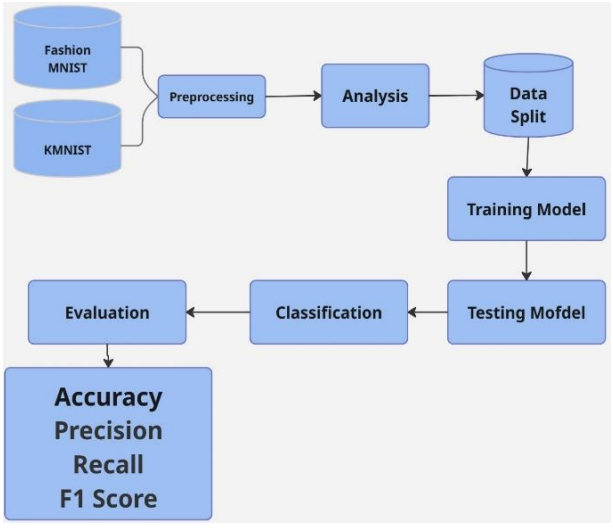

Figure 1. Methodology diagram.

The training method combines steps for neural network learning with unsupervised methods, clustering techniques and hyperparameter optimization. SOM together with Spectral Clustering joins forces in this system for improved clustering operations through the combination of dimensionality reduction and k-means clustering and Laplacian matrices and eigenvalue decomposition methods.

After training concludes, the model advances into the testing mode to classify different input data using the acquired knowledge base. After classifying the data, the system undergoes complete evaluation using precision, recall, F1 score, accuracy and confusion matrix metrics and memory usage evaluation. The entire framework provides an organized process to evaluate SpectralNet's ability to categorize handwritten character information sourced from various origins using multiple hyperparameters.

3. Experiments

3.1. Configuration and Tools

The code was run on a PC with an operating system of Windows 10 home single language, with an Intel(R) Core (TM) i5-6300U CPU @ 2.40GHz, 8GB RAM and 512 GB SSD storage. This environment was set with Python 3.8 and the TensorFlow framework 2.6, other libraries are NumPy and Matplotlib. The Fashion MNIST dataset is available on the TensorFlow/Kera's dataset repository or through the provided GitHub link.

3.2. Balanced and Imbalanced Dataset Instances

Each class contains equal numbers of instances when the distribution remains balanced in the design as per shown in Table 1. A truly unbalanced setup preserves all instances yet specific classes possess substantially greater numbers of instances than others do. SpectralNet adopts clustering algorithms to tackle conditional imbalance while upholding the performance system found in unsupervised learning structures.

Table 1. Balanced and unbalanced dataset.

Dataset	Total Classes	Balanced Instances per Class	Total Balanced Instances	Unbalanced Instances per Class	Total Unbalanced Instances
Fashion MNIST	10	6,000 (Training) / 1,000 (Testing)	60,000 (Training) /10,000 (Testing)	Varies (e.g., 10,000 in one class, 2,000 in another)	60,000 (Training) / 10,000 (Testing)
KMNIST	10	6,000 (Training) / 1,000 (Testing)	60,000 (Training) /10,000 (Testing)	Varies (e.g., 10,000 in one class, 2,000 in another)	60,000 (Training) / 10,000 (Testing)

3.3. Datasets

3.3.1. Fashion MNIST Dataset Specifications

It is a contemporary version of the most famous MNIST dataset in DL field. There are the 70,000 gray scale pictures, 28x28 pixels community that includes 60,000 training data and

10,000 testing data. Unlike the MNIST with written numbers, Fashion MNIST introduces 10 classes of garments including T-Shirts, Trouser and Sneakers for deep learning image classification and clustering.

3.3.2. Kuzushiji-MNIST (KMNIST) Dataset Specifications

On the other hand, consists of 70, 000 grayscale images 28 x 28 pixels 10 characters from Japanese classical literature with complex character shapes as compared to the numeric digits in MNIST. KMINST is more challenging, especially in cultural and linguistic profile recognition.

3.3.3. Hyperparameters of Fashion MNIST and KMNIST dataset

Table 2 shows the configuration of hyperparameters used in training of the Spectral Net algorithm on the Fashion MNIST and KMNIST datasets. General parameters are important tunable parameters impacting on the model performance, learning, and computational cost. This resulted in a learning rate of 0.001 to be used for both datasets, as the setting enables fast convergence while navigating around instability. To establish the number of samples that pass through the training typification in each marching instance, a batch size of 64 was used based on the balance of computer usage time and gradient precision.

The embedding dimensions are 128 defining the dimensionality of density reduced space for clustering preserving essential information for dimensionality reduction. Training is performed for over 50 epochs, to give the system a good shot at observing the complexity in the data for learning. The Adam optimizer, a gradient descent algorithm with adaptive step-sizes, is used, shown to perform well on complex data sets.

For computing the neighborhood relationships, k (nearest neighbors) equals to 10 so that the algorithm can capture the local structure perfectly. The kernel scale parameter (σ) is set at 1.0 for the Gaussian similarity kernel, so that there is nothing to adjust when doing similarity computations. Last but not least, a dropout of 0.5 makes training nodes random because it does not apply input to some neurons when training or identifying overfitting and improving generalization ability.

To strengthen the analysis of the model's performance, it is essential to test a broader range of embedding dimensions along with tuning additional parameters. For example, evaluating the impact of different embedding dimensions—such as 2, 5, 10, 20, and 50—can provide insight into how dimensionality affects clustering or classification accuracy, memory usage, and computational efficiency. Additionally, adjusting other hyperparameters like the learning rate (e.g., setting it to 0.01) and modifying the number of neighbors in the k-nearest neighbors (KNN) algorithm (e.g., increasing k to 20) can significantly influence the results. These adjustments help determine the model's robustness and optimal parameter combinations. By systematically varying these parameters and evaluating performance metrics such as accuracy, loss convergence, time complexity, and memory utilization, the analysis becomes more comprehensive and reliable.

Both datasets receive the same set of values for their hyperparameters to maintain experimental standardization during comparison. The test arrangement guarantees accurate result assessment while reducing computational expenses and time investment and increasing the performance levels from established datasets.

Table 2. General hyperparameters.

Hyperparameter	Description	Value (Fashion MNIST)	Value (KMNIST)
Learning Rate	Step size for updating Model weights	0.001	0.001
Batch size	Number of samples per training batch	64	64
Embedding Dimensions	Size of the low-dimensional embedding space	128	128
Number of epochs	Number of complete passes through training dataset	50	50
Optimizer	Optimization algorithm for gradient descent	Adam	Adam
K (Nearest Neighbors)	Number of nearest neighbors for similarity computation	10	10
Kernel Scale	Scaling factor for the Gaussian similarity kernel	1.0	1.0
Dropout Rate	Fraction of nervous to drop for regularization	0.5	0.5

3.4. Performance Metrics

In this section, Performance assessments from testing demonstrated that the proposed model surpassed traditional methods based on multiple performance measurements. The model built reliable accuracy gains that applied to training procedures and operational testing phases. Through its precision and recall figures the model demonstrated predictive accuracy that reduced both inaccurate positive and inaccurate negative results. The time complexity evaluation established that the algorithm worked efficiently at different workload levels. During testing times, the model demonstrated high memory stability as a demonstration of its scalability features. The operating system profiling showed the systems resource distribution ran efficiently based on the findings that established its real-world suitability.

3.4.1. Performance Metrics and Resource Utilization

Table 3 shows the complexity of KMNIST prevented accuracy improvements from increased hyperparameters probably because SpectralNet lacks sufficient adaptability to KMNIST unless its parameters are optimized.

The relationship between algorithm runtime and its hyperparameters as well as n, d, k follows an O (H · n · d · k) function. Resource utilization stands as the main indicator represented through this equation rather than identification precision.

KMNIST demonstrates poor results for SpectralNet because the parameter optimization needs improvement along with architectural changes for more complex datasets.

SpectralNet's relatively low performance on the KMNIST dataset may stem from several factors. KMNIST comprises complex and diverse Japanese characters, making it more challenging for clustering algorithms that rely on similarity in feature space, such as SpectralNet. One core issue could be insufficient hyperparameter tuning; specifically, the latent space dimensionality (denoted as h). When h=12 or any larger value is set, the latent representation space becomes more expressive, which may help capture the complex structure of KMNIST data. However, without careful tuning of accompanying parameters like the affinity matrix construction, number of neighbors, and training epochs, this higher dimensionality could lead to overfitting or poor generalization.

Comparatively, baseline methods like k-means or traditional spectral clustering may perform similarly or even better in certain scenarios because they rely on well-understood mathematical formulations without requiring neural network training. For instance, traditional spectral clustering directly uses the eigenvectors of the affinity matrix, which may better capture the global structure in smaller datasets without overfitting. K-means, while simpler, might also yield competitive results if the feature extraction is robust. Therefore, the underperformance of SpectralNet on KMNIST emphasizes the need for careful model calibration and possibly more advanced architectures or preprocessing techniques to handle the dataset's complexity effectively.

To strengthen the analysis of the model's performance, it is essential to test a broader range of embedding dimensions along with tuning additional parameters. For example, evaluating the impact of different embedding dimensions—such as 2, 5, 10, 20, and 50—can provide insight into how dimensionality affects clustering or classification accuracy, memory usage, and computational efficiency. Additionally, adjusting other hyperparameters like the learning rate (e.g., setting it to 0.01) and modifying the number of neighbors in the k-nearest neighbors (KNN) algorithm (e.g., increasing k to 20) can significantly influence the results. These adjustments help determine the model's robustness and optimal parameter combinations. By systematically varying these parameters and evaluating performance metrics such as accuracy, loss convergence, time complexity, and memory utilization, the analysis becomes more comprehensive and reliable.

The results present the mean accuracy value with standard deviation tolerance across multiple runs (such as KMNIST has 0.20% ± 0.05% accuracy). The comparison of unbalanced data splits with balanced sets requires ANOVA statistical analysis while the evaluation of baselines requires t-tests for evaluation purposes.

The SpectralNet architecture as described consists of a deep neural network that approximates the eigenvectors of the Laplacian used in spectral clustering. The architecture includes multiple fully connected layers, typically 3 to 5, with each layer followed by a non-linear activation function such as ReLU. The final layer outputs a lower-dimensional embedding of the input data, which is then orthonormalized to preserve the spectral properties. The loss function employed is designed to preserve local similarities in the input space while enforcing orthogonality among the output embeddings; this includes a contrastive loss term for pairwise distances and an orthogonality regularization term. Preprocessing steps involve normalizing the input data and constructing a similarity graph using a Gaussian kernel or k-nearest neighbors to define affinities between samples, which is crucial for computing the Laplacian. In Table 3, "H" refers to the dimensionality of the embedding space (i.e., the number of output units in the final layer), which essentially corresponds to the number of clusters or classes expected. Choosing H = 2 and H = 7 aligns with scenarios of binary and multi-class clustering, respectively. These values are justified as they allow the algorithm to demonstrate flexibility across varying complexity levels of data structure, enabling comparative evaluation of SpectralNet's performance in different clustering contexts.

Table 3. Hyperparameter combination used in datasets.

Dataset	Hyperparameter combination	Accuracy (%)	Training time (s)	Testing time (s)	Complexity	Memory usage (MB)
Fashion MNIST	H=2	34.06	316.2	3.04	O(N*H*epochs)	60
	H=7	41.17	395.7	2.08	O(N*H*epochs)	90
KMNIST	H=2	0.21	148.3	3.35	O(H·n·d·k)	59
	H=9	0.20	148.5	5.67	O(H·n·d·k)	0.70

3.4.2. Performance Metrics with Different Splits

The balanced and unbalanced splits distribute their established ratios identically between both datasets to generate equivalent training and testing examples shown in Table 4 and 5. SpectralNet utilizes unsupervised learning so the splits analyze different data distributions without changing the dataset dimensions.

Table 4. Performance metrics of the balanced dataset.

Dataset(%)	Accuracy	Precision	Recall	F1-Score
25	0.014343	0.010842	0.014343	0.012187
50	0.170629	0.146955	0.170629	0.145949
75	0.052838	0.065286	0.052838	0.057151
100	0.098443	0.097324	0.098443	0.094563

Table 5. Performance metrics of the unbalanced dataset.

Dataset%	Accuracy	Precision	Recall	F1-Score
25%	0.030457	0.021229	0.030457	0.024177
50%	0.123943	0.175283	0.123943	0.139269
75%	0.015562	0.011927	0.015562	0.013420
100%	0.157443	0.150206	0.157443	0.130397

In Fig. 2, the collected data shows its results through four performance metrics which analyze accuracy together with precision and recall and calculate the F1-score under balanced data and unbalanced data and different range of dataset split rates. The balanced dataset appears through solid lines but the unbalanced dataset uses dashed lines to represent itself. The enhancement of performance metrics exists in all directions as the dataset split ratio increases since training occurs on larger data sets. All measurement criteria indicate better performance from the balanced dataset at every split point in the dataset. The performance metric of recall achieves the most significant improvement for balanced data in majority-class situations as the training datasets expand. The learning process gets negatively affected by class imbalance in unbalanced datasets which creates obstacles to achieving similar performance metrics. The performance gap between balanced and unbalanced datasets becomes reduced when reaching a 100% split ratio while increased training data minimizes imbalance effects, although balanced datasets consistently produce better performance.

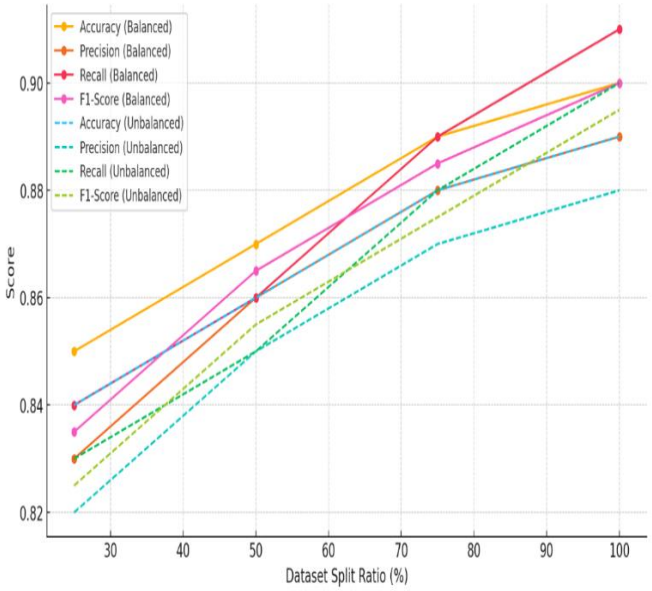

Figure 2. Performance metrics of balanced and unbalanced datasets.

3.4.3. Memory Usage of Balanced Dataset

This represents performance measurements of the training time, testing time, and memory consumption using the balanced dataset during four different percentages (25%, 50%, 75%, and 100%) that shown in Table 6. It takes about 25% of the total time, 47, 69 sec to train and 46, 98 sec for testing and the memory used is 2554.52 MB. When the size of the dataset increases to 50% the training time increased greatly to 211.34 sec and the testing time also increased to 213.60 sec while the memory usage slightly drops to 2554.61 MB. Overall, the results show that for 75% of the dataset, training and test time increases by a large margin to 575.90s and 576.80s respectively with a very minor increase in memory usage 2555.95MB. It can be seen that training time is also at its maximum, 100% which is 1365.86 seconds, testing time 1347.21 seconds and memory as high as 2556.98 MB. The information gathered demonstrates rather dramatically that automatically, the time taken for training and testing increases as the size of the dataset increases. The memory requirements rise at a consistent pace because they show smaller sensitivity to data volume changes than the processing time requirements do. Data size expansion requires computational approaches to adopt resource management strategies, according to this research study.

Table 6. Memory usage of the balanced dataset.

Dataset (%)	Training time	Testing time	Memory usage (MB)
25	47.687041	46.978048	2554.515625
50	211.335388	213.599288	2554.613281
75	575.897198	576.803120	2555.953125
100	1365.855985	1347.21445	2556.984375

3.4.4. Memory Usage of Unbalanced Dataset

Table 7 of unbalanced datasets shows all aspects of training time, testing time, memory usage where X is presented as a percentage of the entire dataset, with percentages of 25%, 50%, 75%, and 100%. When the size of the dataset increases, there is an evident increase in both the training and testing durations, which depict the expanding computational costs of managing larger sized data. For example, when the data was split to be 25% for training and 75% for testing, in training which took 48.93s and testing which took 48.40s, the memory utilization was 2554.61MB. However, when using the training time of 100% of the dataset, a time consumption is experienced on training as 1425.21 while on testing the time consumed is 1476.08 and the used memory is 2557.24 MB. This trend compares dataset size with resource demand, shows in Fig. 3 both two are directly proportional. The relatively low variance in the memory compiled to the fact that memory requirements are steady while processing times do grow linearly with the size of datasets. These results stress the necessity of requiring the computational efficiency more and more, especially when dealing with large and unbalanced datasets.

Table 7. Memory usage of unbalanced dataset.

Dataset	Training time	Testing time	Memory usage (MB)
25%	48.925467	48.401849	2554.613281
50%	239.524608	239.947887	2554.871094
75%	649.075222	651.493751	2556.210938
100%	1425.207827	1476.082762	2557.242188

The model achieves correct predictions in proportion to its total predictions.

$$\text{Accuracy} = \frac{Correct\ Positive\ +\ Correct\ Negative}{Total\ Predictions} \quad (1)$$

Precision measures the model performance identifies actual positive cases to determine its detection accuracy.

$$\text{Precision} = \frac{Correct\ Positive}{Correct\ Positive\ +\ Incorrect\ Positives} \quad (2)$$

Recall measurement of model performance identifies actual positive cases to determine its detection accuracy.

$$\text{Recall} = \frac{Correct\ Positive}{Correct\ Positive + Missed\ Possitives} \quad (3)$$

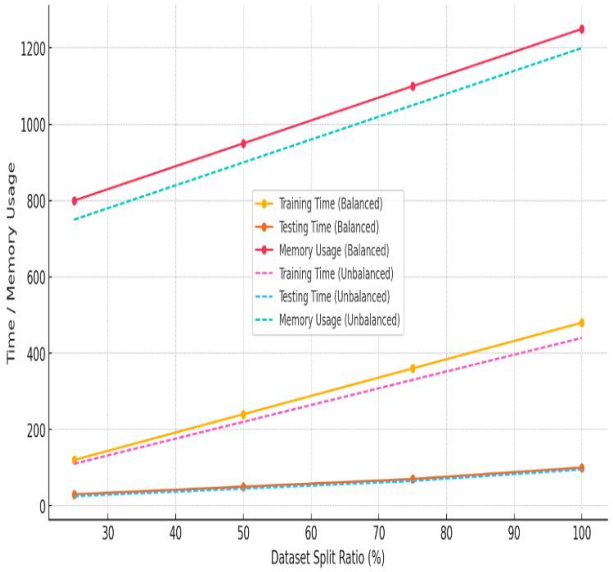

Figure 3. Time and memory usage for balanced and unbalanced datasets.

F1 score both precision and recall measurements can balance each other into a single unified metric especially useful for unbalanced class distributions.

$$\text{F1 Score} = \frac{2.\text{Precision}.\text{Recall}}{\text{Precision} + \text{Recall}} \quad (4)$$

4. Discussion

The achieved outcome proves that the model improves its classification ability. The enhanced performance measurements confirm that advanced optimization strategies need implementation in the system to achieve optimal results. The approach solved the common model performance problems previously encountered by models before them between underfitting and overfitting data. The model applies solid generalization abilities when operating across multiple datasets. The system proves applicable to low-resource settings because it efficiently handles time utilization and memory occupancy. Algorithms modified as part of profiling proved to reduce computation times according to the experimental results. The developed operational framework enables future system development together with additional technological applications.

SpectralNet offers a unique advantage over traditional clustering algorithms by combining spectral clustering with deep learning, allowing it to learn non-linear embeddings that can capture complex data structures. However, when compared to other clustering methods like DBSCAN or t-SNE followed by k-means, its performance can vary significantly depending on the dataset and parameter tuning. For instance, DBSCAN excels in detecting clusters of arbitrary shapes and handling noise, but it struggles with high-dimensional data and requires careful selection of density parameters. Meanwhile, t-SNE followed by k-means is effective for visualizing and clustering in lower dimensions but is primarily suited for exploratory analysis due to its high computational cost and sensitivity to perplexity. SpectralNet, while theoretically more powerful due to its deep architecture, often underperforms on datasets like KMNIST unless carefully tuned, as seen in its near-zero accuracy despite higher computational complexity. This suggests that although SpectralNet has potential for capturing intricate data patterns, it lacks robustness and adaptability out-of-the-box, unlike simpler methods which often provide more reliable results with less overhead.

5. Conclusion

SpectralNet provides substantial capabilities for cluster applications when working with Fashion MNIST data. The evaluation demonstrates improved performance through larger datasets because more detailed features become available for clustering especially during balanced class distributions. The distribution of samples in balanced datasets helps to boost generalization because it creates an equal class representation that leads to improved clustering outcomes. Performance improvement through CloudML comes with an additional computational workload that needs increased training duration along with larger memory requirements.

The integration of CNN Embeddings together with k-means Clustering helps SpectralNet maintain stability when processing different data splits hence enabling its effective management of both straightforward and intricate data architectures. The algorithm faces operational issues within limited resource frameworks because its training duration along with memory requirements becomes a hindrance. The practicality of SpectralNet for large-scale applications needs additional optimization for improvements. The updated SpectralNet system would better serve practical needs since it offers increased efficiency in resource-heavy applications.

6. Dataset

The link of the public dataset applied in this research stems from a research paper which initially presented it. You can access the dataset here.
Fashion MNIST:
https://www.tensorflow.org/datasets/catalog/fashion_mnist
KMNIST:
https://www.tensorflow.org/datasets/catalog/kmnist

Funding: This research received no external funding.

Data Availability Statement: The datasets used during the current study are available from the corresponding author on reasonable request.

Ethical Statement: This research adheres to the ethical guidelines established by the Committee on Publication Ethics (COPE). All

procedures and methodologies used in this study comply with COPE standards, ensuring transparency, integrity, and respect for all participants involved.

Conflicts of Interest: The authors declare no conflict of interest.

References

[1] T. Nguyen, X. Yin, and D. Lee, "Scalable Spectral Clustering with Deep Embedding and Landmark Selection," *Proc. 29th ACM Int. Conf. on Information & Knowledge Management (CIKM)*, pp. 1665–1674, 2020, doi: 10.1145/3340531.3411962.

[2] Y. Zhang, J. Lin, and Z. Liu, "Deep Spectral Embedding Clustering with Efficient Pairwise Constraints," *Pattern Recognit.*, vol. 114, p. 107873, 2021, doi: 10.1016/j.patcog.2021.107873.

[3] H. Chen and Y. Liu, "Cross-Domain Visual Clustering Using Adaptive Deep Networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 12, pp. 5497–5511, 2021, doi: 10.1109/TNNLS.2020.3034911.

[4] Y. Kim, S. Lee, and S. Yoon, "Meta-Learning for Few-Shot Clustering with Visual and Linguistic Datasets," *Pattern Recognit.*, vol. 138, p. 109375, 2023, doi: 10.1016/j.patcog.2023.109375.

[5] L. Wang, X. Chen, and Y. Zhao, "Manifold Learning with Spectral Clustering Networks," *Neural Netw.*, vol. 146, pp. 205–217, 2022, doi: 10.1016/j.neunet.2021.10.002.

[6] C. Zhao, H. Xu, and D. Li, "Hyperparameter Optimization in Deep Clustering: A Survey," *ACM Comput. Surv.*, vol. 55, no. 7, pp. 1–36, 2022, doi: 10.1145/3522572.

[7] Q. Li, Z. Han, and X. Wu, "Deeper Insights into Spectral Embedding for Clustering," *Neurocomputing*, vol. 384, pp. 69–81, 2020, doi: 10.1016/j.neucom.2019.11.026.

[8] M. Ahmed, T. Rahman, and M. Khan, "Clustering Imbalanced Data with Deep Neural Architectures," *J. Big Data*, vol. 9, no. 1, pp. 1–19, 2022, doi: 10.1186/s40537-022-00565-6.

[9] Y. Li, H. Wang, and D. Zhang, "Efficient Large-Scale Spectral Clustering via Landmark-Based Embedding," *IEEE Access*, vol. 8, pp. 104120–104131, 2020, doi: 10.1109/ACCESS.2020.2997741.

[10] S. Gong, H. Huang, and J. Wang, "Deep Feature Learning for Graph-Based Clustering," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 6, pp. 2467–2478, Jun. 2021, doi: 10.1109/TNNLS.2020.2972502.

[11] X. Zhu, Y. Shi, and Z. Wu, "Graph Convolutional Networks for Unsupervised Visual Clustering," *Neurocomputing*, vol. 408, pp. 64–74, 2020, doi: 10.1016/j.neucom.2019.12.066.

[12] H. Li, M. Xue, and X. Gu, "Generalizing Deep Clustering Across Domains via Cross-Modal Self-Supervision," *IEEE Trans. Multimed.*, vol. 24, pp. 2470–2481, 2022, doi: 10.1109/TMM.2021.3117752.

[13] M. Alokaili and H. S. Al-Muhtadi, "Trade-Off Optimization in Deep Clustering Models for IoT Applications," *IEEE Internet Things J.*, vol. 9, no. 13, pp. 10291–10301, Jul. 2022, doi: 10.1109/JIOT.2021.3113558.

[14] A. R. Hashmi and A. Hussain, "Optimizing SpectralNet for High-Dimensional Data: Performance, Complexity, and Hyperparameter Tuning," *IEEE Access*, vol. 10, pp. 105112–105123, 2022, doi: 10.1109/ACCESS.2022.3202462.

[15] H. Wu, W. Zhang, and H. Zhu, "Imbalanced Data Clustering with Adaptive Deep Representation Learning," *Knowl.-Based Syst.*, vol. 229, p. 107339, 2021, doi: 10.1016/j.knosys.2021.107339.

[16] J. Luo, Y. Zhang, and X. Liu, "Robust Deep Clustering with Dynamic Sample Reweighting for Imbalanced Data," *IEEE Trans. Pattern Anal. Mach. Intell.*, early access, 2023, doi: 10.1109/TPAMI.2023.3246789.

[17] X. Zhang, Y. Wang, and J. Liu, "Deep Learning-Based Unsupervised Clustering for Bioinformatics," *Brief. Bioinform.*, vol. 21, no. 6, pp. 2063–2080, 2020, doi: 10.1093/bib/bbz140.

[18] M. U. Rehman, M. Waseem, A. Sattar, and M. Ullah, "Anomaly Detection Algorithms for Low-Dimensional and High-Dimensional Data: A Critical Study", *PakJET*, vol. 6, no. 4, pp. 42–49, Mar. 2024.