ML is the New Trend: Dimensionality Reduction and Clustering application on Fashion-MNIST

Myriam Kapon, Eugenia Argyriadi

December 2023

Abstract

Sight is the sense which humans rely on the most. While humans effortlessly navigate the visual world, the realm of image recognition poses a formidable challenge for computers. This complexity stems from the inherently high dimensionality of images, the vast array of object classes, and the nuanced variations within each class. Unlike the innate visual intuition of humans, machines grapple with the intricate task of deciphering and categorizing visual data, emphasizing the intricacies inherent in the field of computer vision.

This report focuses on the performance of several dimensionality reduction and classifying techniques on the Fashion-MNIST dataset. We explore the use of various dimensionality reduction techniques, including Principal Component Analysis, t-SNE, Random Forest, Stacked Autoencoders, and Convolutional Stacked Autoencoders. Subsequently, we employ a diverse set of clustering algorithms, such as K-means, DBSCAN, Gaussian Mixture, Spectral, and Agglomerative clustering, to group similar fashion items. The analysis aims to compare the combinations of dimensionality reduction and clustering techniques to find the one most suited for the Fashion-MNIST dataset. **Keywords:** Fashion

1 Dataset Description

The Fashion MNIST dataset serves as a benchmark for image classification tasks in the realm of computer vision. Comprising a collection of grayscale images, each depicting a fashion item, this dataset provides a diverse and challenging set of examples for analysis. Developed as an alternative to the traditional MNIST dataset of handwritten digits, Fashion MNIST offers a more complex classification task, making it suitable for exploring advanced machine learning techniques.

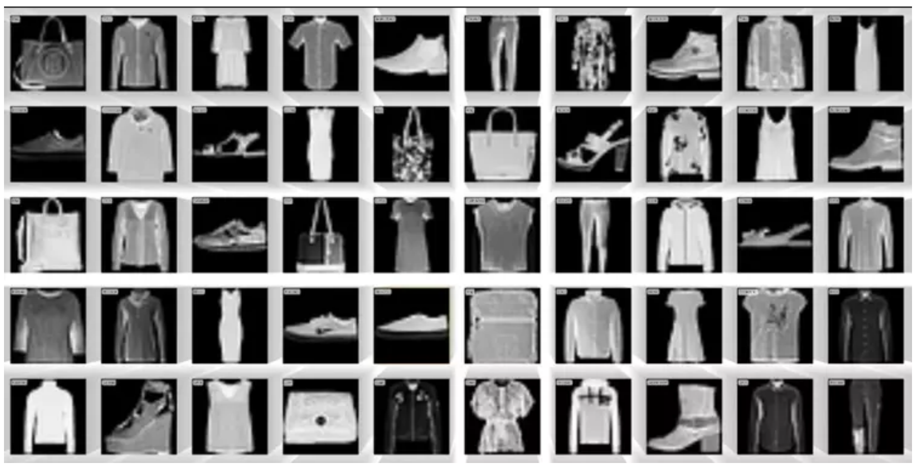


Figure 1: Sample items from the Fashion MNIST dataset

1.1 Dataset Characteristics

- **Number of Samples:** The dataset consists of a total of 70,000 images, split into 48,000 training samples, 12,000 validation samples and 10,000 testing samples.
- **Image Dimensions:** Each image is 28 pixels in height and 28 pixels in width, resulting in a total of 784 pixels per image.
- **Categories:** Fashion MNIST includes 10 distinct classes or categories, representing various fashion items commonly found in everyday life. These classes are as follows:
    1. T-shirt/top
    2. Trouser
    3. Pullover
    4. Dress
    5. Coat
    6. Sandal
    7. Shirt
    8. Sneaker
    9. Bag
    10. Ankle boot

2 Algorithm

The algorithm begins by loading the Fashion MNIST dataset using Keras and unpacks it into training, validation and testing sets. Following the dataset split, the algorithm proceeds to flatten the images in the datasets and scale the pixel values.
The algorithm incorporates various clustering techniques and performance metrics for evaluating their effectiveness. The clustering techniques include MiniBatch KMeans, Gaussian Mixture, DBSCAN, Spectral Clustering, and Agglomerative Clustering, each initialized with specific hyperparameters tailored to the nature of the Fashion MNIST dataset. These techniques are stored in the clustering_techniques dictionary for easy reference.
Additionally, the algorithm defines a set of clustering evaluation metrics, including the Calinski–Harabasz score, Davies–Bouldin index, Silhouette Coefficient, and Adjusted Rand Index. These metrics are essential for assessing the quality of clustering results based on different criteria, such as compactness, separation, and similarity to ground truth labels. The metrics are stored in the metrics dictionary, with associated functions and relevant datasets.
In the main section, the algorithm iterates, for each of the dimensionality reduction technique (and for no DR technique), through these clustering techniques fitting each model to the preprocessed data and recording execution time. Visualizations, including various plots, images (original and reconstructed), and true label percentages, are generated to provide insights into the clustering results. The loop also calculates the metrics for each combination of clustering and dimensionality reduction technique. The recorded information, including algorithm name, execution time, and metric scores, is stored in a dataframe.

3 Metrics

Typically, clustering evaluation metrics are categorized into two main types: internal and external measures. Internal measures evaluate the quality of clusters solely based on the data and the clustering outcomes, without relying on any ground truth. On the other hand, external measures assess the clustering results by comparing them against known or provided ground truth labels.

In this particular analysis, three internal metrics were selected: the Calinski–Harabasz index, Davies–Bouldin index, and the Silhouette Coefficient. These internal measures assess the clustering performance using characteristics inherent to the data and the resulting clusters.

Additionally, the external metric chosen was the Adjusted Rand Index (ARI). This external measure was opted for not only to offer a different perspective but also due to its practical utility in comparing the clustering results against ground truth labels, providing an external benchmark for evaluation. Furthermore, because the Fashion MNIST dataset is balanced, ARI is slightly favored due to its tendency to perform well with balanced clusters.

3.1 Calinski–Harabasz index

The Calinski–Harabasz index, also known as the Variance Ratio Criterion, is defined as the ratio of the between-cluster variance to the within-cluster variance. In other words, it quantifies the separation between clusters relative to the compactness of individual clusters. Higher values of the index indicate better-defined and more separated clusters.

3.2 Davies–Bouldin index

For a given clustering result, the Davies–Bouldin index is calculated as the average similarity ratio of each cluster with its most similar cluster. The index is defined as follows:

$$DBI = \frac{1}{k}\sum_{i=1}^{k} \max_{j \neq i} \left(\frac{\text{avg}_i + \text{avg}_j}{d(c_i, c_j)} \right)$$

In the context of the Davies–Bouldin index, a lower value is considered better. The Davies–Bouldin index measures the compactness and separation of clusters, and a lower index indicates that the clusters are more compact and well-separated.

3.3 Silhouette Coefficient

The silhouette coefficient assesses how well individual data points conform to their assigned clusters. It considers both cohesion (proximity of a data point to others in its cluster) and separation (distance of a data point from points in other clusters). The silhouette coefficient for a single data point i is calculated using the following formula:

$$\text{Silhouette}(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

Where:
- $a(i)$ represents the average distance from the point i to other points within the same cluster.
- $b(i)$ represents the smallest average distance from the point i to points in a different cluster, minimized over clusters (excluding the cluster to which i belongs).

Its scale ranges from -1 to 1: A higher silhouette score signifies well-defined clusters with distinct separation and tight cohesion within each cluster. Conversely, a lower score suggests less accurate clustering, potentially with overlapping clusters or poorly-assigned data points. To gauge the overall clustering quality of the entire dataset, the average silhouette score across all datapoints is calculated.

3.4 Adjusted Rand Index (ARI)

Adjusted Rand Index (ARI) is a metric used to evaluate the similarity between two clusterings, considering the agreement between the predicted clustering and the ground truth labels (if available). It measures the similarity by considering pairs of samples and quantifies how often the pairs are assigned to the same or different clusters in the predicted and true clusterings. It is computed as:

$$\text{ARI} = \frac{\text{ad} - \text{bc}}{\sqrt{(a+b)(a+c)\cdot(c+d)(b+d)}}$$

Where:

- a: The number of pairs that are in the same cluster in both the predicted and true clusterings.
- b: The number of pairs that are in different clusters in both the predicted and true clusterings.
- c: The number of pairs that are in the same cluster in the predicted clustering but in different clusters in the true clustering.
- d: The number of pairs that are in different clusters in the predicted clustering but in the same cluster in the true clustering.

The ARI score ranges from -1 to 1, where a score of 1 indicates perfect similarity between the two clusterings (perfect agreement between predicted and true clusterings) and a score around 0 suggests random clustering. A score less than 0 indicates that the agreement between the clusterings is worse than random.

4 Dimensionality reduction

Why did the dataset break up with its high-dimensional features?
Because it needed some space and wanted a more compressed relationship!

Dimensionality reduction is a the problem of taking a matrix with many observations, and "compressing it" to a matrix with fewer observations which preserves as much of the information in the full matrix as possible.

4.1 PCA

Principal Component Analysis is a statistical method used to reduce the dimensions of high-dimensional data by transforming it into a smaller set of uncorrelated variables called principal components.

Principal components in PCA capture the maximum variance present in the data. Each component contributes a certain proportion of the total variance, termed as explained variance. The cumulative explained variance across components illustrates how much of the original data's variability is retained as more components are considered, guiding the choice of how many components to retain for effective dimensionality reduction.

The relationship between components and cumulative variance in the graph displays a logarithmic pattern: initially, the first components capture the majority of the variance, but as additional components are included, their contribution to explaining variance diminishes. For instance, in this scenario, a mere 25 components suffice to elucidate 80% of the variance in the data, a good threshold for retaining information effectively while reducing dimensionality.

Successfully reducing the dimension from 784 to 25, these 25 components amalgamate to construct the dataset's reconstructed images. However, The presence of "ghosts" from other clothing items in these images results in a blurred effect, making item identification challenging and consequently complicating clustering. To mitigate this effect and enhance clarity, selecting more components would be necessary, albeit at the expense of increased dimensionality.

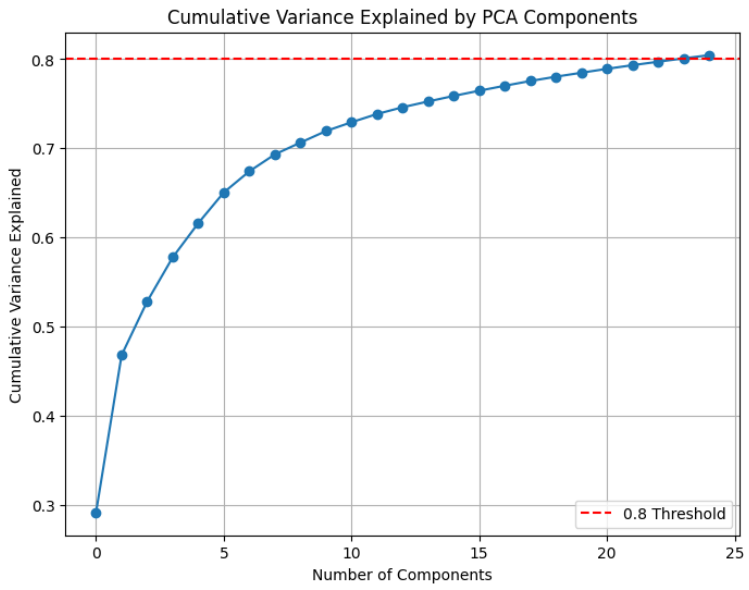


Figure 2: The Cumulative Variance explained by PCA components

4.2 t-SNE

T-distributed stochastic neighbour embedding takes a high dimensional data set and reduces it to a low dimensional graph that retains a lot of the original information. It does so by giving each data point a location in a two or three-dimensional map. This technique finds clusters in data there by making sure that an embedding preserves the meaning in the data. t-SNE reduces dimensionality while trying to keep similar instances close and dissimilar instances apart. It's important to note that t-SNE is particularly effective in preserving local structures, making it suitable for capturing intricate relationships present in the Fashion MNIST images.
The t-SNE algorithm was implemented using the scikit-learn library. The TSNE class was employed with the number of components set to 2 to project the data onto a two-dimensional space. The random state was fixed to ensure reproducibility. The reduction process transformed the original dataset with a shape of (10000, 784) into a two-dimensional representation with a shape of (10000, 2).

4.3 Random Forest

While Random Forests are primarily known as a powerful ensemble learning method for classification and regression tasks, they can also be used for feature selection, which indirectly serves as a form of dimensionality reduction. As a Random Forest model is trained on a dataset, it assigns importance scores to each feature based on their contribution to the overall model performance. These importance scores can then be used to rank the features in descending order. By selecting the top-ranked features, either based on a threshold or a predetermined number, the dimensionality of the dataset is effectively reduced. The selected features can subsequently be used as input for training another model or for further analysis.

In our program, the Random Forest algorithm is utilized with a total of 100 estimators to train a classifier on the Fashion-MNIST dataset. Feature selection is performed using the SelectFromModel method, and features with importance above a threshold value of 0.004 are retained. The Random Forest algorithm identified and preserved 26 pixels considered crucial for capturing the essential characteristics of the images.

When reconstructing the images from the reduced representation obtained through Random Forest-based dimensionality reduction, a threshold of 0.001 was initially chosen to understand the reconstruction process. The reconstructed images were the same as the original ones, but with noticeable gaps on them. These gaps were indicative of the pixels that were not selected during the dimensionality reduction process. A more stringent threshold of 0.004 was subsequently employed and only 26 pixels were selected in this manner

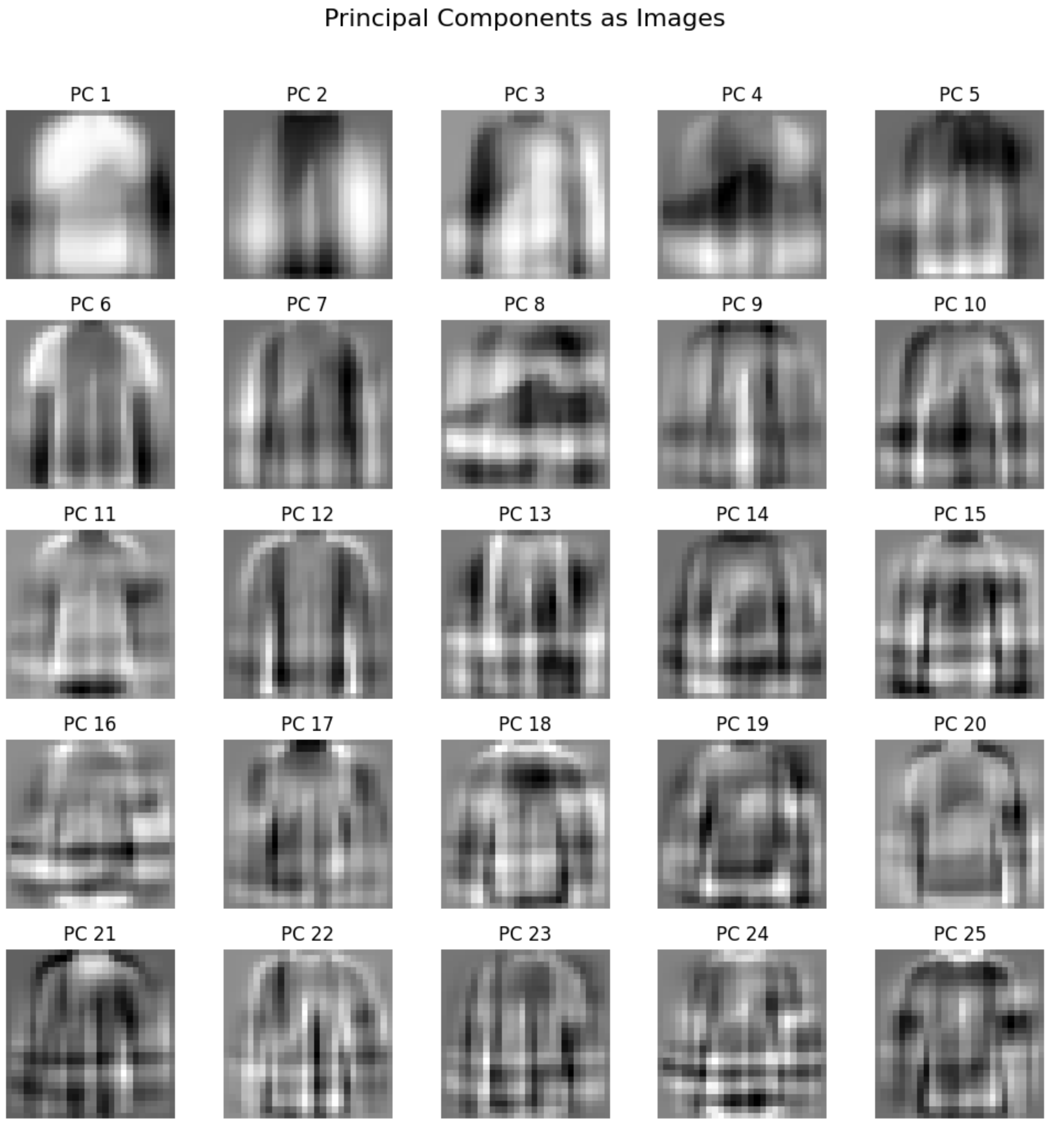


Figure 3: The 25 most important Principal Components

contrasting to the previous selection of over 300 pixels. Interestingly, while the reconstructed images were visually less similar due to the sparsity of selected pixels, the metric scores were comparable to those achieved with a less strict threshold of 0.001. Although the human eye may find it challenging to perceive the subtle similarities in the sparsely represented reconstructed images, the choice to implement the 0.004 threshold was driven by the optimization of computational efficiency and resource utilization.

4.4 Stacked Autoencoders

Stacked Autoencoders are a type of artificial neural network architecture used for unsupervised learning and dimensionality reduction. Comprising multiple layers of encoding and decoding units, these autoencoders aim to learn a compressed representation of the input data by minimizing the reconstruction error. Each layer in the encoder captures increasingly abstract features, while the corresponding decoder layer reconstructs the original input. Stacking multiple such layers forms a hierarchical structure, allowing the model to learn complex hierarchical representations of the data. The training process involves feeding the input data through the encoder-decoder pairs, adjusting the model's weights to minimize the difference between the

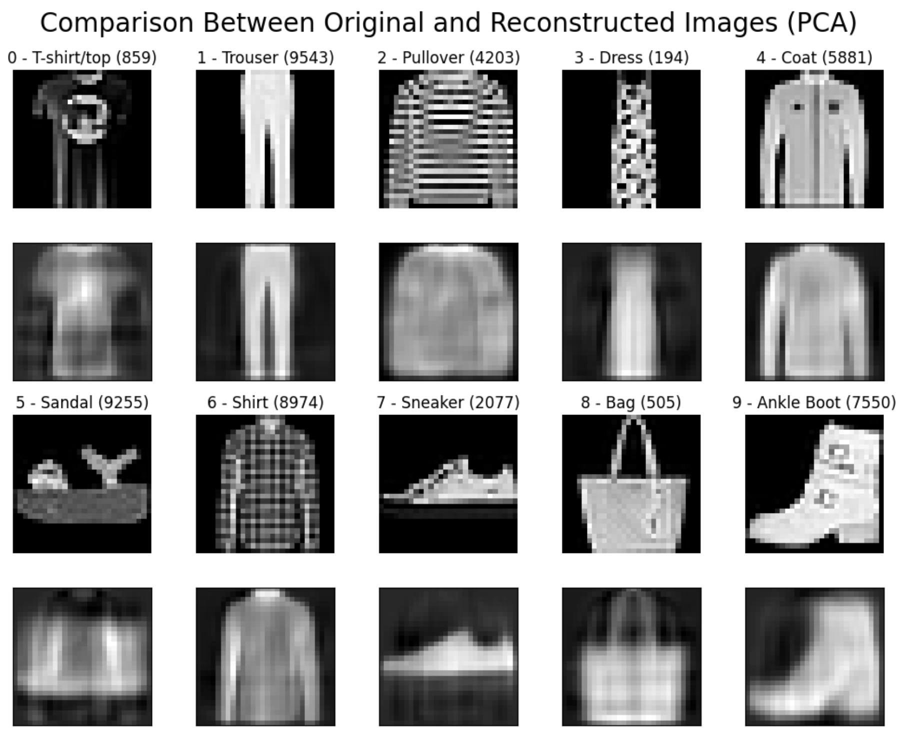
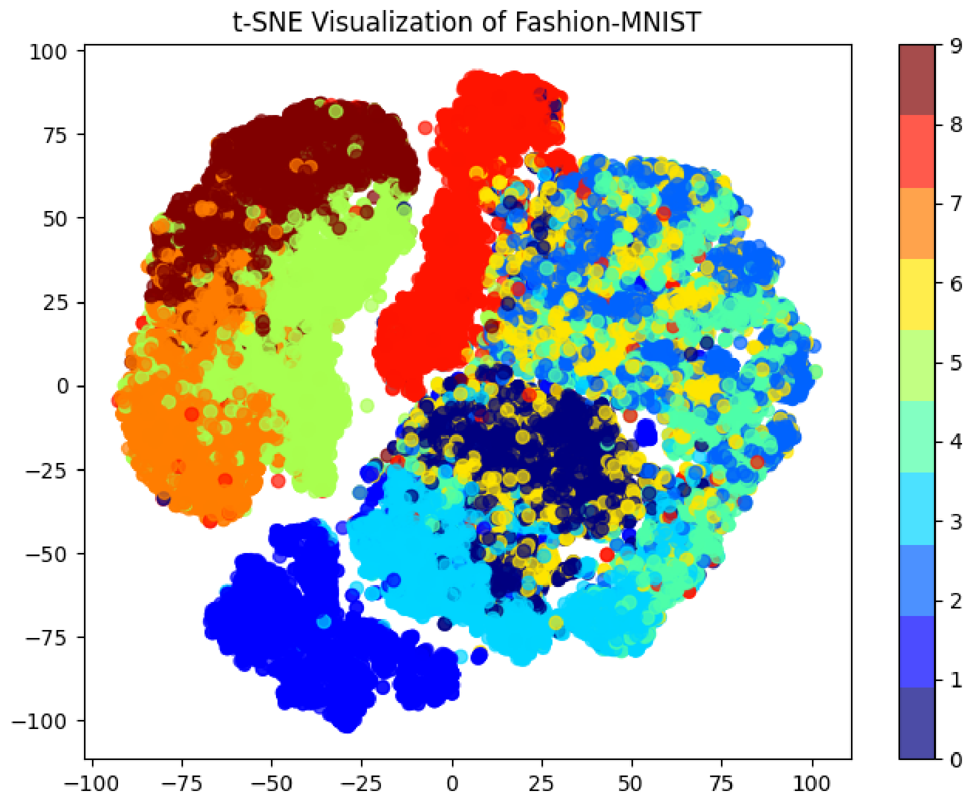
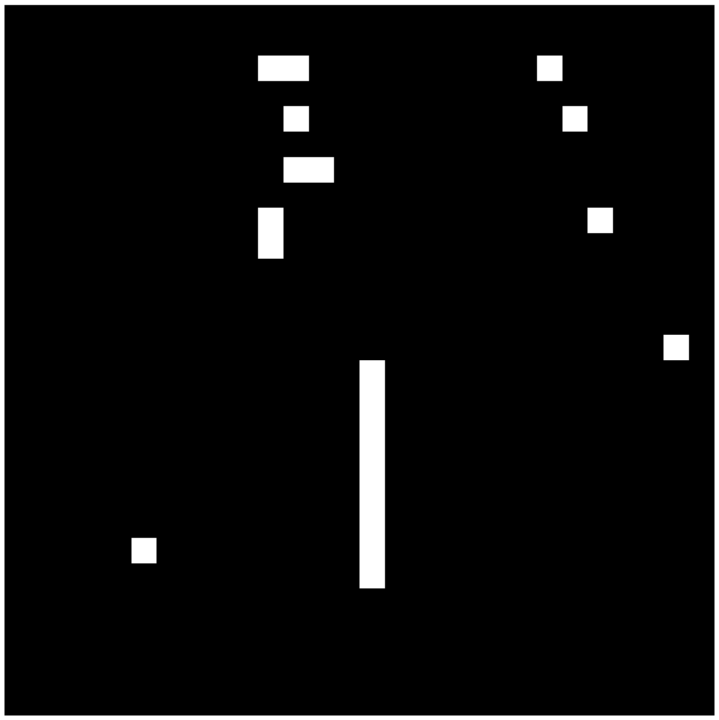


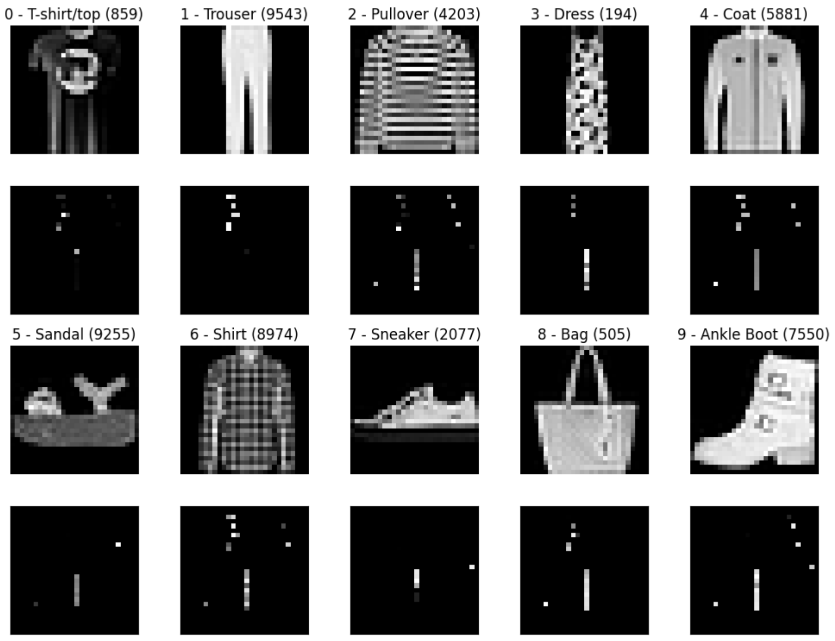
Figure 4: Comparison between Original and Reconstructed images when performing PCA



Selected Feature Pixels Visualization



Comparison Between Original and Reconstructed Images (Random Forest)



input and the reconstructed output. The result is a powerful feature extraction mechanism that can be employed for tasks like data denoising, anomaly detection, or as a preprocessing step for supervised learning tasks. Stacked Autoencoders have found applications in various domains, including computer vision, natural language processing, and signal processing, providing an effective means of capturing intricate patterns and reducing the dimensionality of high-dimensional datasets.

The architecture in this analysis employs a relatively simple Stacked Autoencoder, comprised of an encoder and a decoder, with a total of 6 layers.

- Input Layer: Receives a flattened image of 784 features.
- Encoder Layers: Compress information across 3 layers (128, 64, and 32 neurons) using ReLU activation, enforcing sparsity.
- Decoder Layers: Reconstruct the original input across 3 layers (64, 128, and 784 neurons) using ReLU and sigmoid activation on the final layer.
- Optimizer: Adam. It's a strong default choice.
- Loss function: Mean Squared Error. For dimensionality reduction tasks with autoencoders, the Mean Squared Error (MSE) loss function is commonly used. It measures the average squared difference between the input and the reconstructed output, effectively guiding the network to minimize reconstruction errors, which is crucial in dimensionality reduction tasks.

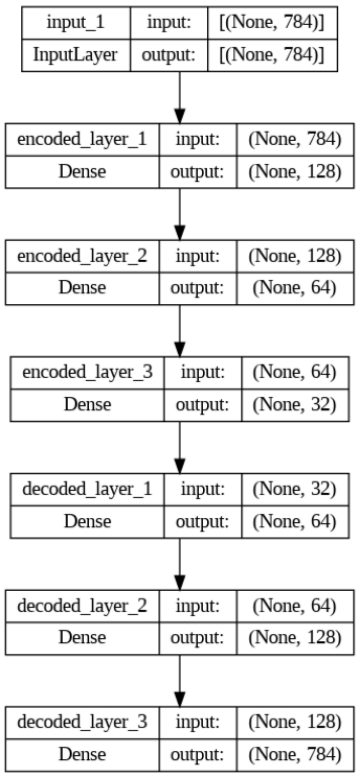


Figure 5: The architecture of the SAE Model

The sparsity was added because in this architecture, the representations were only constrained by the size of the hidden layer (32). In such a situation, what typically happens is that the hidden layer is learning an approximation of PCA (principal component analysis). But another way to constrain the representations to be compact is to add a sparsity contraint on the activity of the hidden representations, so fewer units would "fire" at a given time.

Throughout training, both the training and validation sets were continuously monitored using a loss function, which gauges the disparity between predicted and actual values. Initially, the loss was notably high, but from the second epoch onward, it rapidly decreased, stabilizing around the tenth epoch. This convergence indicated that the model had likely reached its learning limit, prompting the decision to halt further training. To further minimize the loss function, employing a more intricate model would likely be necessary. During the final epoch, the Training and Validation Loss was **0.0266**.

So, how well did this Stacked Autoencoder perform dimensionality reduction? By isolating the encoder in the model, it significantly reduced dimensions from 784 to a mere 32.

Using the decoder part, these 4 x 8 features can be decoded back into their original images with surprising accuracy. While it tends to lose some finer details, this simplification makes it easier to create distinct clothing categories, effectively removing "noise". However, there are shortcomings in areas where the clothing has gaps, like in the case of sandals, where the reconstruction fills in these gaps inaccurately.

Overall, Stacked Autoencoders offer a versatile reduction of dimensions.

4.5 Convolutional Stacked Autoencoders

Convolutional Stacked Autoencoders are a variant of stacked autoencoders designed for processing spatially correlated data, such as images. By incorporating convolutional layers, CSAEs efficiently capture local patterns and hierarchical features. The encoder extracts abstract representations using convolution and pooling layers, while the decoder reconstructs the input through deconvolution and upsampling. This stacked architecture enables the model to learn intricate spatial hierarchies. CSAEs find applications in image-related

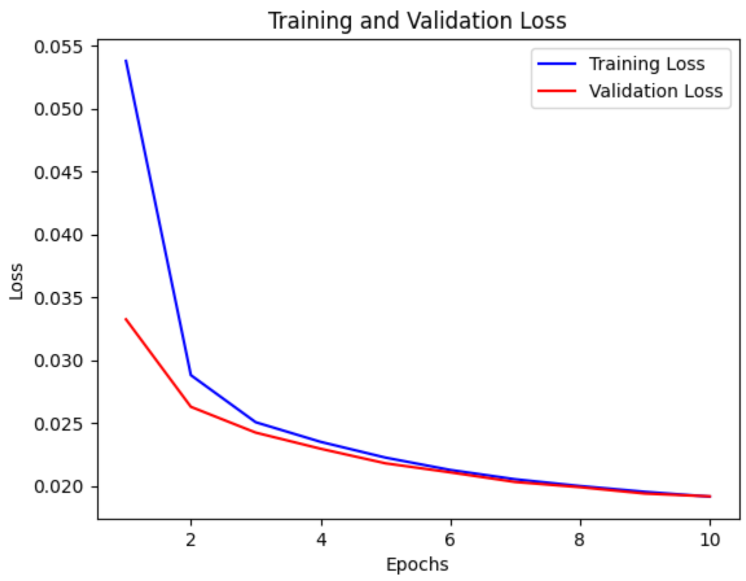


Figure 6: Loss function for the train and validation sets

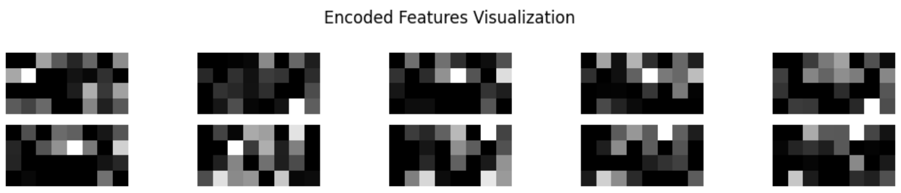


Figure 7: A selection of encoded features using a Stacked Autoencoder

tasks, offering powerful features for tasks like denoising, feature learning, and image generation. Their ability to automatically extract relevant features from structured data makes them particularly valuable in computer vision applications.

In this Convolutional Autoencoder, the 3x3 filters hold a crucial role in extracting features from the input image. Applied across multiple layers, these filters progressively capture and abstract hierarchical representations. Starting from the early layers (32 filters), they discern low-level features such as edges, gradients, and simple textures. As the network deepens, moving towards the later layers (8 filters), these filters transform into representations of high-level features. Here, they potentially encode complex combinations of visual elements, distinct object parts, and nuanced textures.

The loss function graph revealed that the validation loss value surpassed the training set value in the early epochs. This discrepancy, where the validation loss exceeds the training loss, often signals overfitting. It implies that relying solely on Batch Normalization might not have been sufficient to counter overfitting tendencies in this instance. To address this in upcoming models, it's advisable to incorporate additional techniques like Dropout and Regularization. These methods can provide further safeguards against overfitting, ensuring a more robust and generalized model performance. During the final epoch, the Training and Validation Loss was **???**.

As for the dimensionality reduction, the final dimensions were (7, 7, 16). Each sample is represented as a 7x7 grid and the number 16 refers to the number of channels or filters in each of these 7x7 feature maps. In the context of an autoencoder, these channels hold encoded representations of different features extracted from the input images. Each channel might capture different patterns, textures, or characteristics learned by the model during the encoding phase.

The reconstructed images from the decoder exhibit a unique characteristic—they appear "sculpted." They lack detailed edges, emphasizing the primary shape of the original image. This sculpting effect can be

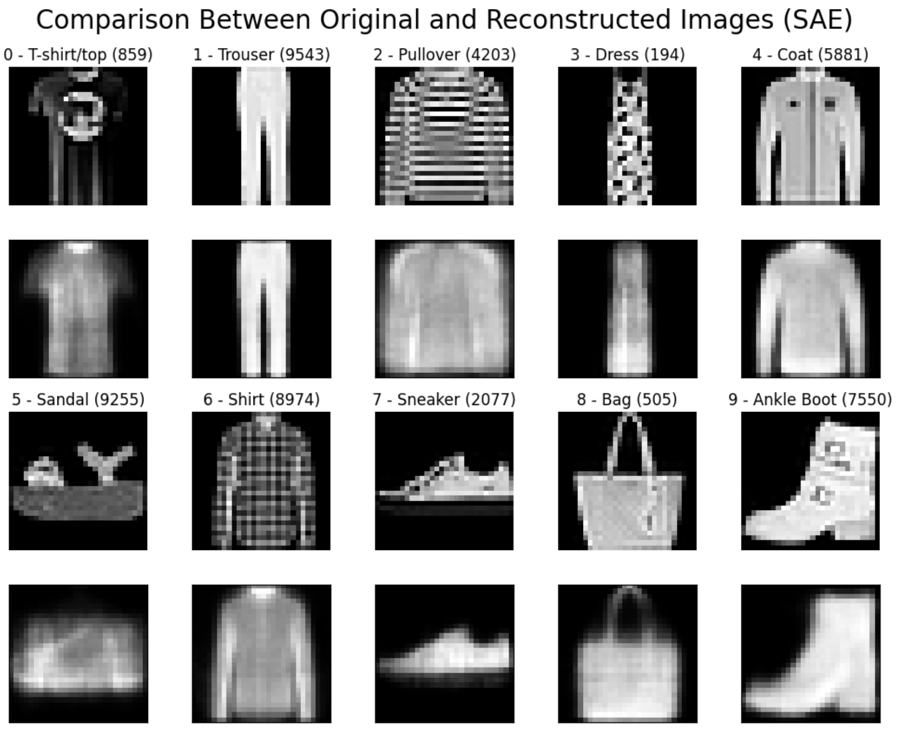


Figure 8: Comparison between the original and the reconstructed images using a Stacked Autoencoder

advantageous, especially for items like sandals with holes, as unlike other dimensionality reduction methods, these gaps remain unfilled, preserving the distinct features of the clothing items.

5 Clustering

Why did the clustering algorithm go to a meditation retreat?
It needed to find its inner centroid peace.

Clustering is the assignment of objects to homogeneous groups (called clusters) while making sure that objects in different groups are not similar. Clustering is considered an unsupervised task as it aims to describe the hidden structure of the objects. Each object is described by a set of characters called features.

5.1 MiniBatch KMeans

MiniBatch K-Means is a variation of the traditional K-Means clustering algorithm designed for more efficient processing of large datasets. It operates by randomly selecting small batches of data in each iteration to update cluster centroids, making it computationally faster and more memory-efficient than standard K-Means, especially for extensive datasets that might not fit into memory at once, like Fashion Mnist. K-Means partitions a dataset into K clusters by iteratively updating centroids, assigning data points to the nearest centroid, and optimizing cluster centers to minimize the sum of squared distances within each cluster, requiring the predefined number of clusters (K) as an input.

To determine the ideal number of clusters, an assessment ranging from 2 to 21 clusters was conducted, evaluating various performance metrics. While some metrics favored fewer clusters and others leaned toward a higher count, a compromise was struck at 7 clusters. Although the dataset comprises 10 distinct classes, opting for 7 clusters finds a balance, especially for similar-looking fashion items like pullovers and shirts, where clustering them together seems more intuitive. Another alternative could have been exploring a larger

The Convolutional AutoEncoder architecture:

1. **Input Layer**: The network takes grayscale images of size 28x28x1 as input.
2. **Encoder**:
   - **Convolutional Layers**: Three sets of convolutional layers followed by max-pooling are used to progressively reduce the spatial dimensions and extract features. The number of filters increases from 32 to 16 to 8 in each convolutional layer, with a 'relu' activation function and 'same' padding.
   - **Max Pooling**: Reduces the spatial dimensions (by a factor of 2) after each set of convolutional layers, aiding in feature extraction and dimensionality reduction.
3. **Decoder**:
   - **Convolutional Layers**: Mirrors the encoder structure in reverse, using upsampling layers (UpSampling2D) to gradually reconstruct the original spatial dimensions.
   - **Convolutional Layers with 'relu' Activation**: Upsampled layers reconstruct the image features.
   - **Output Layer**: The final convolutional layer with a 'sigmoid' activation function outputs a reconstructed image of the same dimensions as the input.

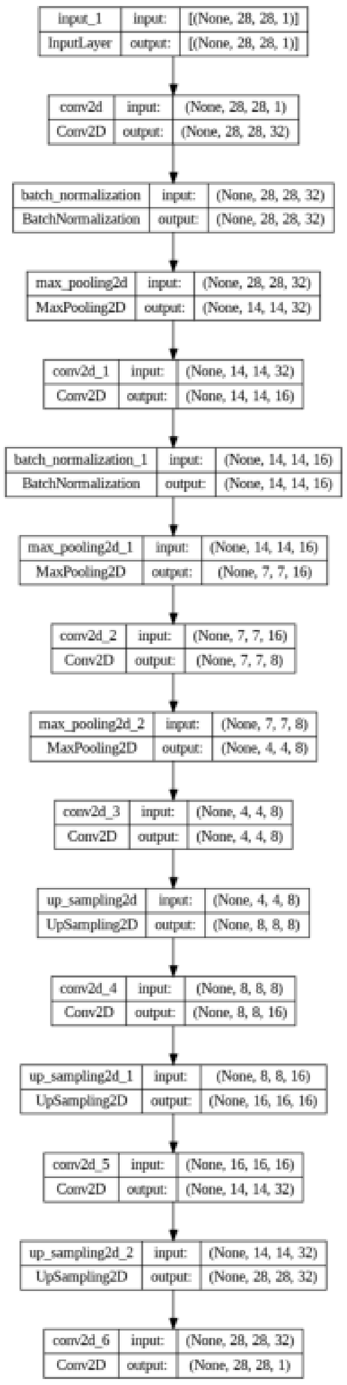

input_1 | input: | [(None, 28, 28, 1)]
InputLayer | output: | [(None, 28, 28, 1)]
conv2d | input: | (None, 28, 28, 1)
Conv2D | output: | (None, 28, 28, 32)
batch_normalization | input: | (None, 28, 28, 32)
BatchNormalization | output: | (None, 28, 28, 32)
max_pooling2d | input: | (None, 28, 28, 32)
MaxPooling2D | output: | (None, 14, 14, 32)
conv2d_1 | input: | (None, 14, 14, 32)
Conv2D | output: | (None, 14, 14, 16)
batch_normalization_1 | input: | (None, 14, 14, 16)
BatchNormalization | output: | (None, 14, 14, 16)
max_pooling2d_1 | input: | (None, 14, 14, 16)
MaxPooling2D | output: | (None, 7, 7, 16)
conv2d_2 | input: | (None, 7, 7, 16)
Conv2D | output: | (None, 7, 7, 8)
max_pooling2d_2 | input: | (None, 7, 7, 8)
MaxPooling2D | output: | (None, 4, 4, 8)
conv2d_3 | input: | (None, 4, 4, 8)
Conv2D | output: | (None, 4, 4, 8)
up_sampling2d | input: | (None, 4, 4, 8)
UpSampling2D | output: | (None, 8, 8, 8)
conv2d_4 | input: | (None, 8, 8, 8)
Conv2D | output: | (None, 8, 8, 16)
up_sampling2d_1 | input: | (None, 8, 8, 16)
UpSampling2D | output: | (None, 16, 16, 16)
conv2d_5 | input: | (None, 16, 16, 16)
Conv2D | output: | (None, 14, 14, 32)
up_sampling2d_2 | input: | (None, 14, 14, 32)
UpSampling2D | output: | (None, 28, 28, 32)
conv2d_6 | input: | (None, 28, 28, 32)
Conv2D | output: | (None, 28, 28, 1)

Figure 9: The architecture of the CSAE Model

number of clusters, say 256, to capture the intricate nuances between various clothing types, such as different styles of sandals.

These pie charts offer an insightful glimpse into the content of clusters formed by MiniBatch KMeans. Each chart illustrates the percentage distribution of clothing categories within the clusters. Notably, clusters 1, 3, 4, and 7 exhibit higher uniformity with fewer colors, indicating more homogeneity within these clusters. Interestingly, the algorithm shows better proficiency in identifying shoe categories—Sandals (brown),

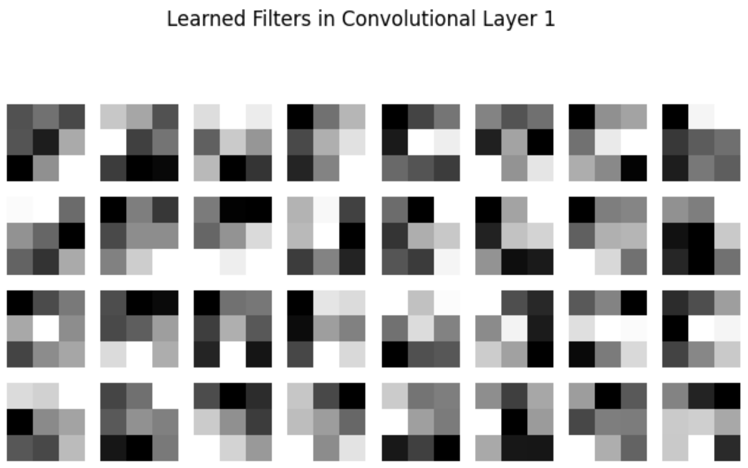


Figure 10: Learned Filters of the first Convolutional layer

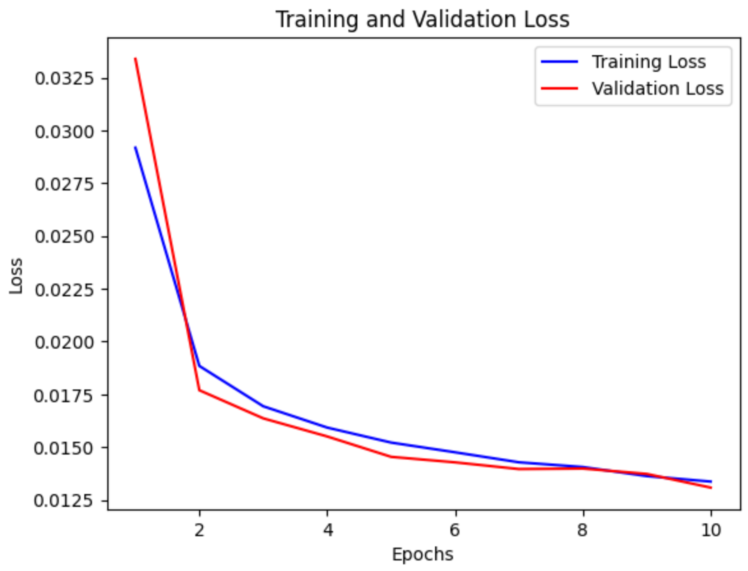


Figure 11: Loss function for the train and validation sets

Sneakers (grey), and Ankle Boots (light blue)—yet struggles to differentiate between them, making shoes the most distinguishable category but with minimal distinctions among shoe types. Moreover, trousers (orange) emerge as another category where the algorithm performs relatively well in cluster identification.

Metric	Calinski-Harabasz	Davies-Bouldin	Silhouette Score	ARI
PCA	1456.60	2.08	0.14	0.29
t-SNE	1391.95	2.02	0.14	0.42
Random Forest	1088.81	2.45	0.09	0.31
SAE	1188.14	2.27	0.10	0.27
CNN SAE	1467.91	1.99	0.15	0.30

Table 1: Performance metrics for different clustering algorithms with kMeans using various dimensionality reduction techniques

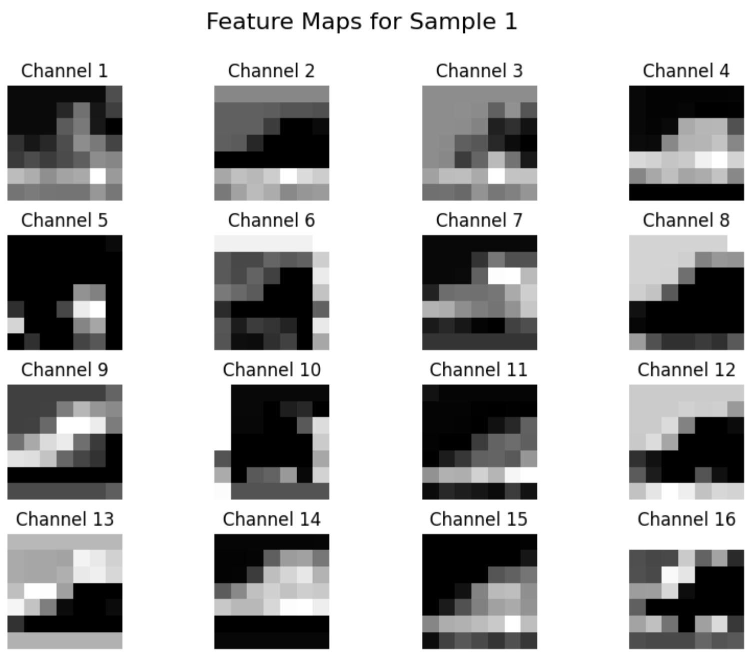


Figure 12: The feature maps of the first sample using CSAE

According to the metric scores, the best reduction technique for MiniBatch KMeans is a Convolutional Autoencoder, scoring the highest for Calinski-Harabasz, Davies-Bouldin and Silhouette Score.

5.2 DBSCAN

DBSCAN stands for Density-Based Spatial Clustering of Applications with Noise. It's a popular clustering algorithm used in machine learning and data mining for grouping points in a dataset based on their spatial density.

Upon initially applying DBSCAN to the Fashion MNIST dataset, a glaring issue emerges: the data points exhibit high density. Consequently, DBSCAN struggles, leading to the formation of either a single massive cluster or labeling nearly all points as outliers due to the overwhelming density.

To combat this, hyperpamater tuning was attempted. DBSCAN has two parameters:

- Minimum samples ("MinPts"): the fewest number of points required to form a cluster.
- ϵ (epsilon or "eps"): the maximum distance two points can be from one another while still belonging to the same cluster.

To derive the minimum value for MinPts, a good rule of thumb is to set it to be double to the number of dimensions D in the data set. However, since the dimensions change with each dimensionality reduction technique, the biggest dimension was chosen (**MinPts = 80**).

To determine the value of e in DBSCAN, a k-distance graph is constructed by arranging the distances to the k = MinPts-1 nearest neighbors in ascending order. Optimal e values often exhibit an "elbow" in this graph. A small e might leave a significant portion of data unclustered, while a high e could merge clusters, leading to most objects belonging to a single cluster. Generally, smaller e values are preferred, ensuring only a limited fraction of points lie within this distance from each other. The graph shows that e = 0.16 is the optimal choice, however the "elbow" seems to be at **e = 2**. Using this value allowed DBSCAN to create some different clusters.

However, DBSCAN is still performimng poorly and is not recommended for the Fashion MNIST dataset. The only result was given using PCA.

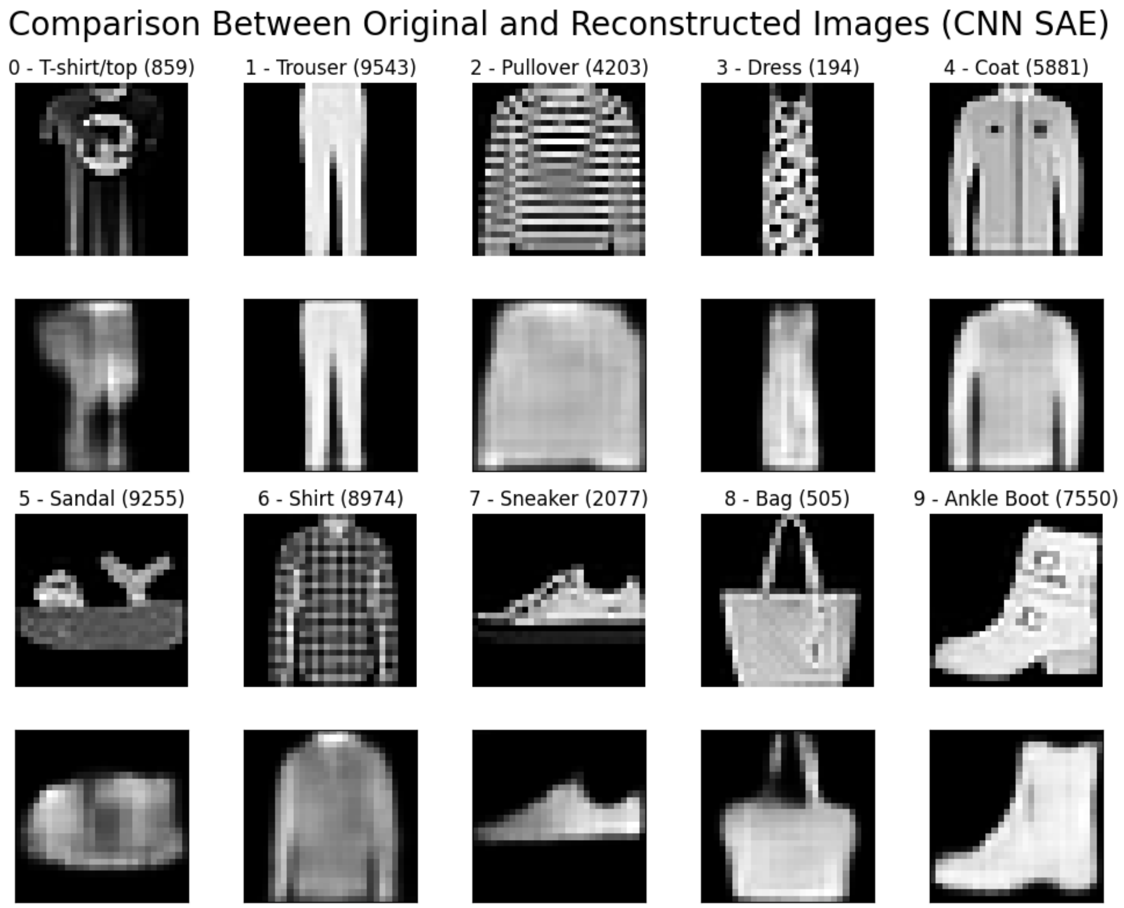


Figure 13: Comparison between Original and Reconstructed images when using a Convolutional Autoencoder

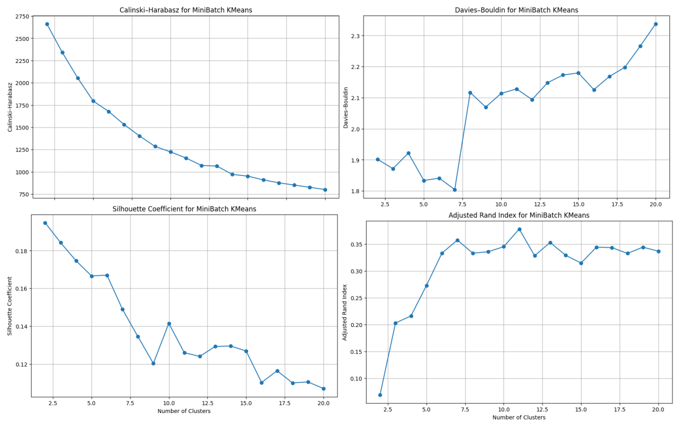


Figure 14: The score of different cluster numbers for Calinski-Harabasz, Davies-Bouldin, Silhouette and ARI metrics

Metric DR	PCA	t-SNE	Random Forest	SAE	CNN SAE
Calinski-Harabasz	525.42	-	-	-	-
Davies-Bouldin	1.57	-	-	-	-
Silhouette Score	-0.035	-	-	-	-
ARI	0.024	-	-	-	-

Table 2: Performance metrics for different dimensionality reduction techniques for DBSCAN

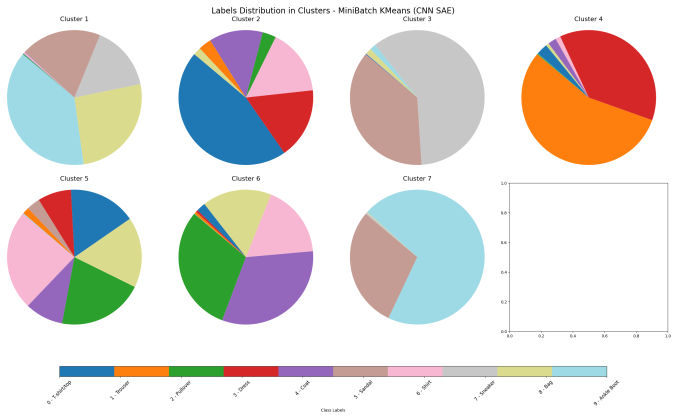


Figure 15: The distribution of labels within MiniBatch KMeans clusters using CNN-SAE for dimensionality reduction.

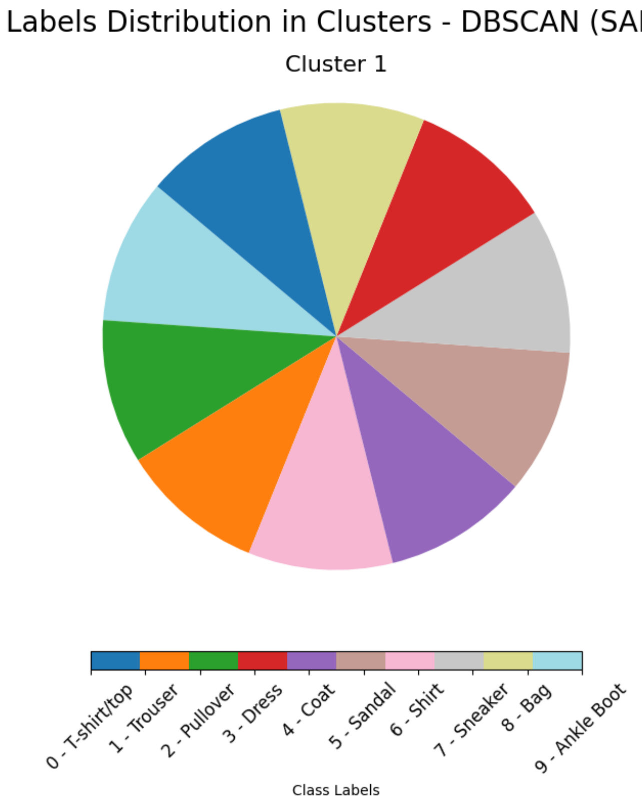


Figure 16: The label distribution in the one cluster of DBSCAN

5.3 Gaussian Mixture

A Gaussian Mixture Model (GMM) is a probabilistic model that represents a mixture of multiple Gaussian (normal) distributions. Each Gaussian distribution in the mixture is associated with a certain weight, and the overall probability density function is a weighted sum of these individual Gaussian distributions.

A GMM with 10 components was utilized, providing a flexible and probabilistic approach to cluster assignment. The choice of 10 components was made to align with the assumed structure of the dataset, where distinct categories or clusters were hypothesized. The GMM leverages a probabilistic framework to assign data points to different clusters, allowing for soft assignments that capture the uncertainty in

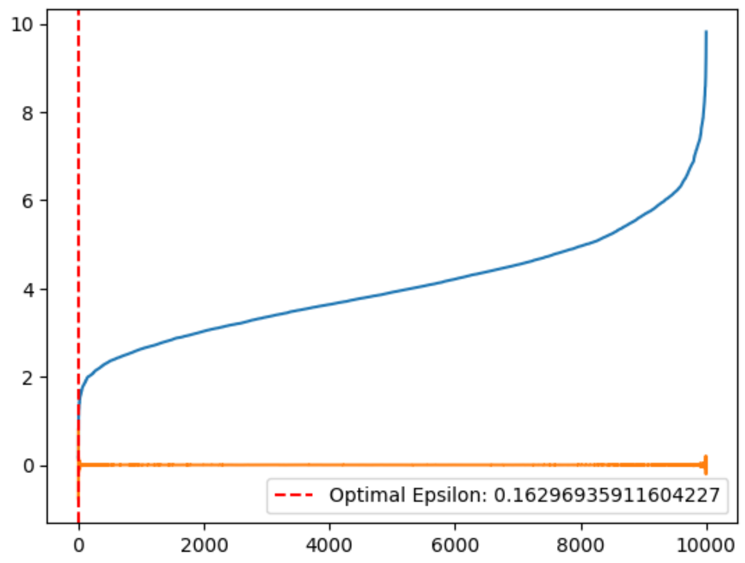


Figure 17: A k-distance graph showing optimal e values

cluster memberships. This approach is particularly valuable when the dataset exhibits complex structures or overlaps between clusters.

Metric DR	PCA	t-SNE	Random Forest	SAE	CNN SAE
Calinski-Harabasz	820.58	1072.65	570.13	803.3	1110.99
Davies-Bouldin	3.08	2.27	4.08	2.71	2.18
Silhouette Score	0.087	0.11	0.021	0.058	0.13
ARI	0.4	0.43	0.25	0.32	0.4

Table 3: Performance metrics for different dimensionality reduction techniques for Gaussian Mixture

5.4 Spectral Clustering

Spectral clustering is effective in capturing complex relationships and can handle non-linear structures. If Fashion-MNIST exhibits non-linear patterns, spectral clustering might be a good choice.

To determine the optimal number of clusters, the dataset was downsized to 1000 samples to expedite computations. Then a systematic analysis was conducted by plotting the scores of each metric for PCA across various cluster numbers. Through this examination, it was discerned that 11 clusters provided a favorable balance, capturing significant variance in the data while avoiding over-segmentation.

Metric DR	PCA	t-SNE	Random Forest	SAE	CNN SAE
Calinski-Harabasz	1301.24	1141.04	981.37	1000.29	1202.28
Davies-Bouldin	1.78	2.27	2.49	2.48	2.20
Silhouette Score	0.15	0.10	0.08	0.08	0.11
ARI	0.38	0.45	0.36	0.29	0.34

Table 4: Performance metrics for different dimensionality reduction techniques for Agglomerative Clustering

5.5 Agglomerative Clustering

Agglomerative clustering is hierarchical and can capture different levels of granularity in the clusters. This can be useful in fashion datasets where items may belong to subcategories within broader categories.

A procedure akin to that in Spectral Clustering was applied, leading to the conclusion that selecting 8 clusters reflects a balance between granularity and cohesion within the dataset's structure. Agglomerative Clustering, with its capacity to capture both local and global structures, provides valuable insights into the inherent patterns present in the reduced feature space.

The 'ward' linkage method, known for its emphasis on minimizing the variance within clusters during the merging process, was selected to encourage the formation of compact and internally cohesive clusters. This method aligns with the objective of revealing clusters with similar internal structures, providing a meaningful representation of the inherent patterns within the reduced feature space. In simpler terms, using the "ward" linkage method tends to create compact and spherical clusters. This method is often preferred when the goal is to form clusters of roughly equal size, and it works well when the clusters have a somewhat globular shape.

6 Combined Analysis

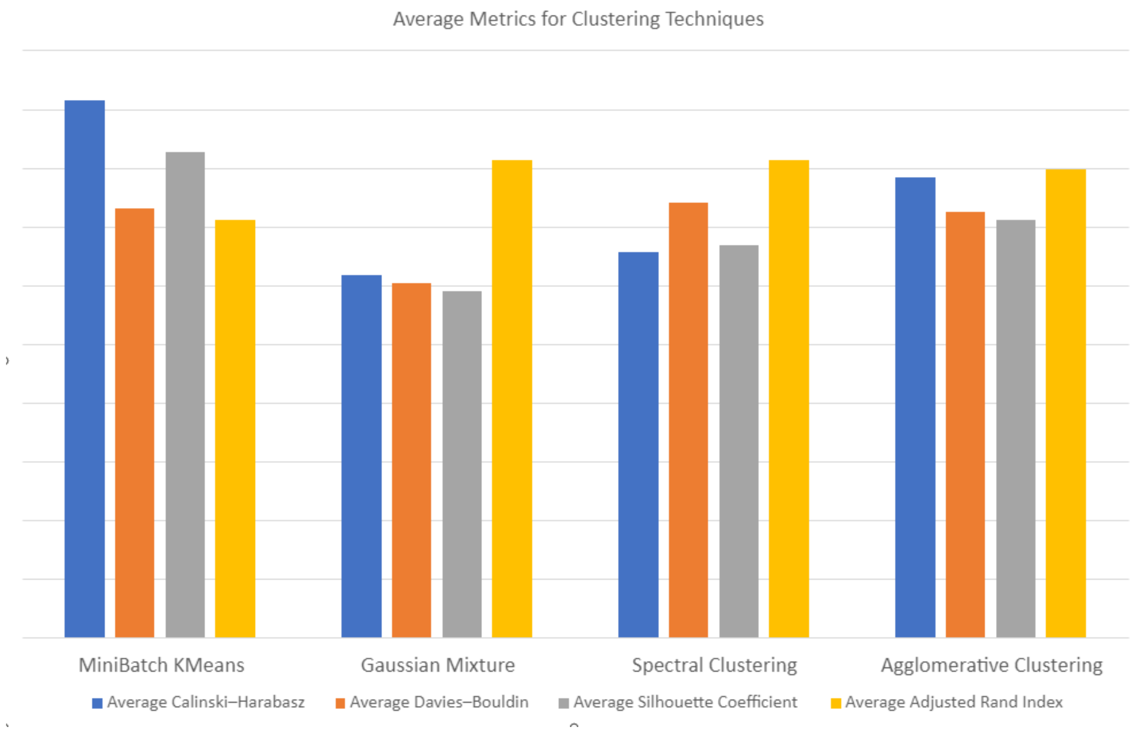


Figure 18: Average metrics for clustering

7 Conclusion and Discussion

In this project, we applied dimensionality reduction and clustering techniques to the Fashion MNIST dataset to gain insights into its inherent structure. The following key findings and conclusions emerge from our analysis:

7.1 Key Findings

In conclusion, the application of dimensionality reduction and clustering techniques has provided valuable insights into the structure of the Fashion MNIST dataset. The combination of PCA and K-means clustering offers a powerful approach for understanding patterns and relationships within complex image datasets.

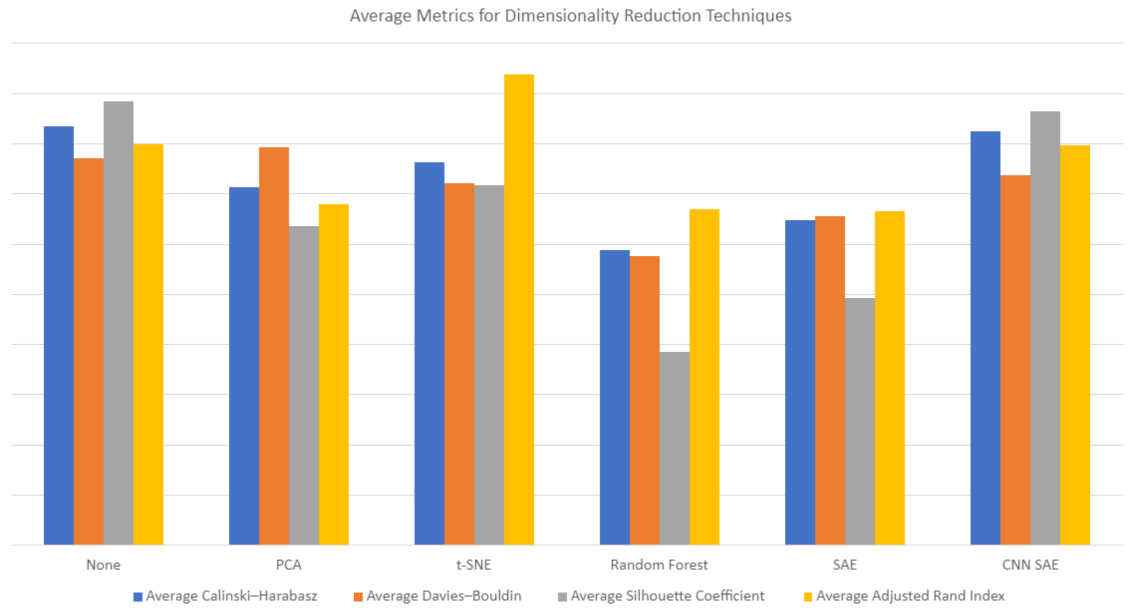


Figure 19: Average metrics for dimensionality reduction

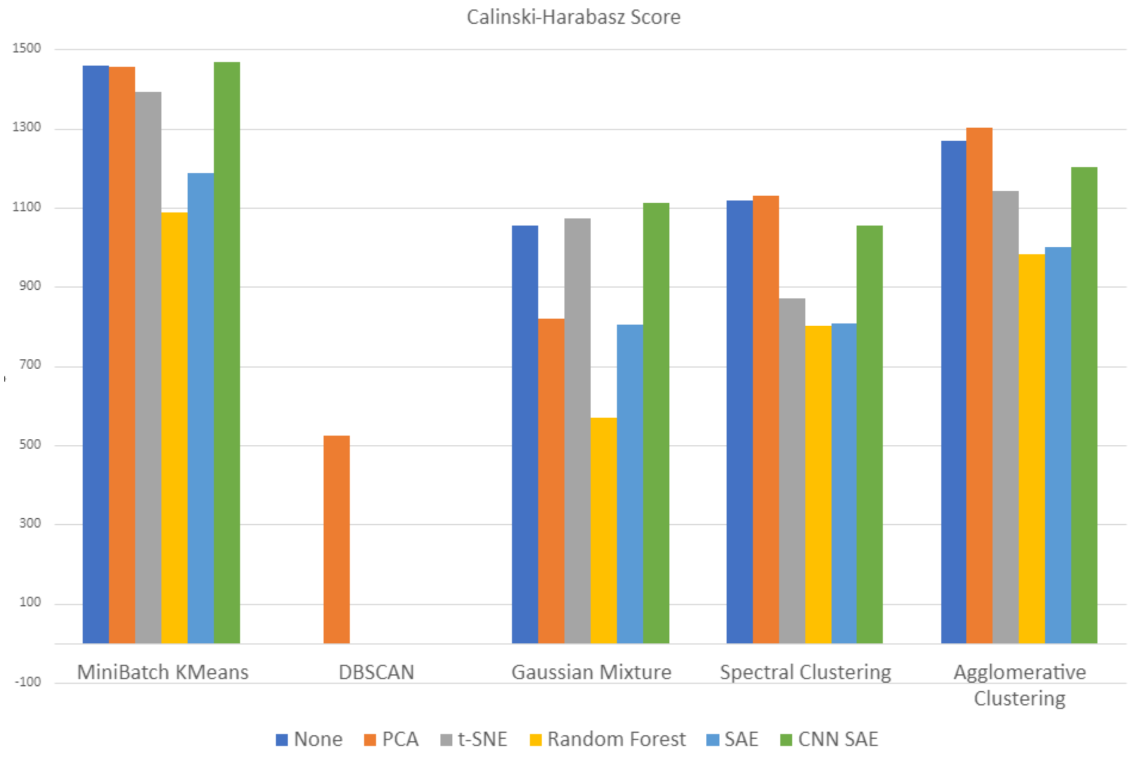


Figure 20: Calinksi-Harabasz score for different combinations of Dimensionality and Clustering Techniques

As we move forward, the findings from this project lay the foundation for more sophisticated applications, including improved classification models and anomaly detection systems in the realm of fashion item recognition.

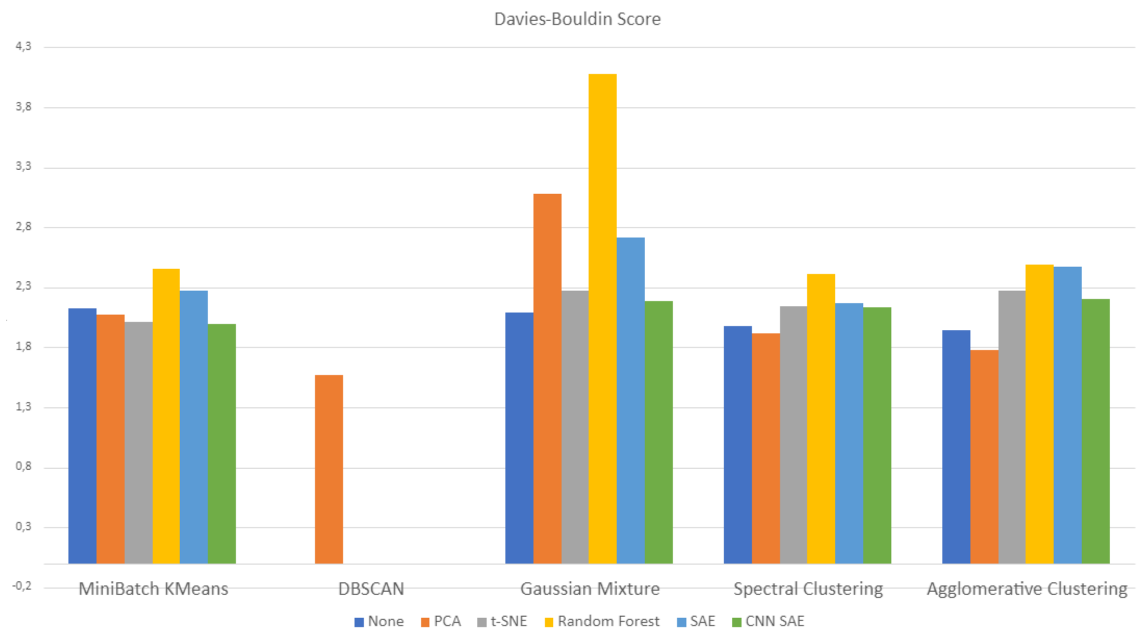


Figure 21: Davies-Bouldin score for different combinations of Dimensionality and Clustering Techniques

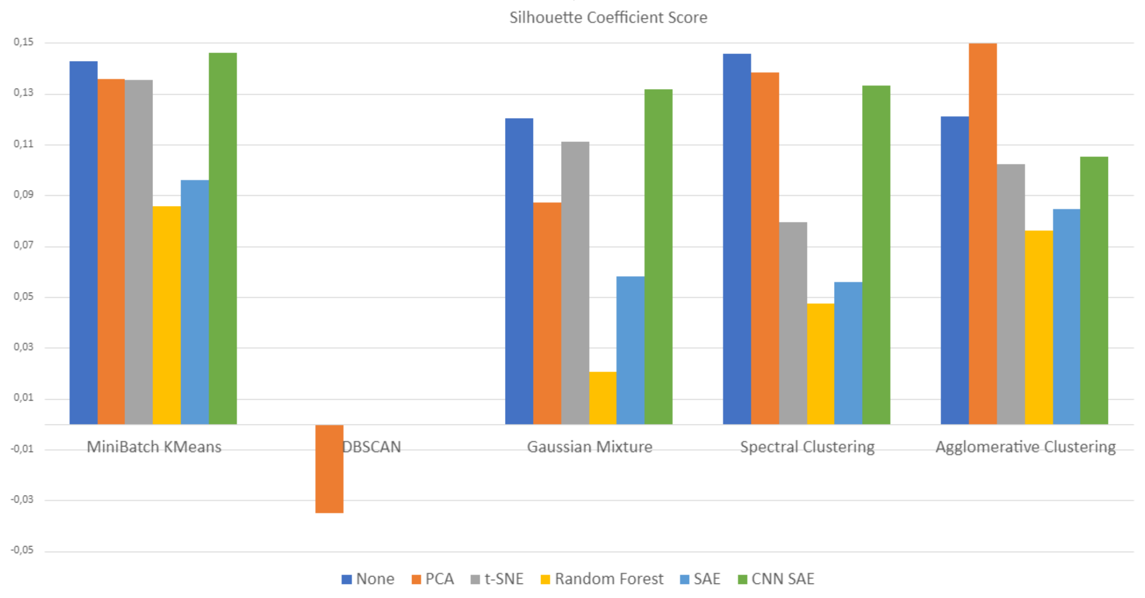


Figure 22: Silhouette Coefficient score for different combinations of Dimensionality and Clustering Techniques

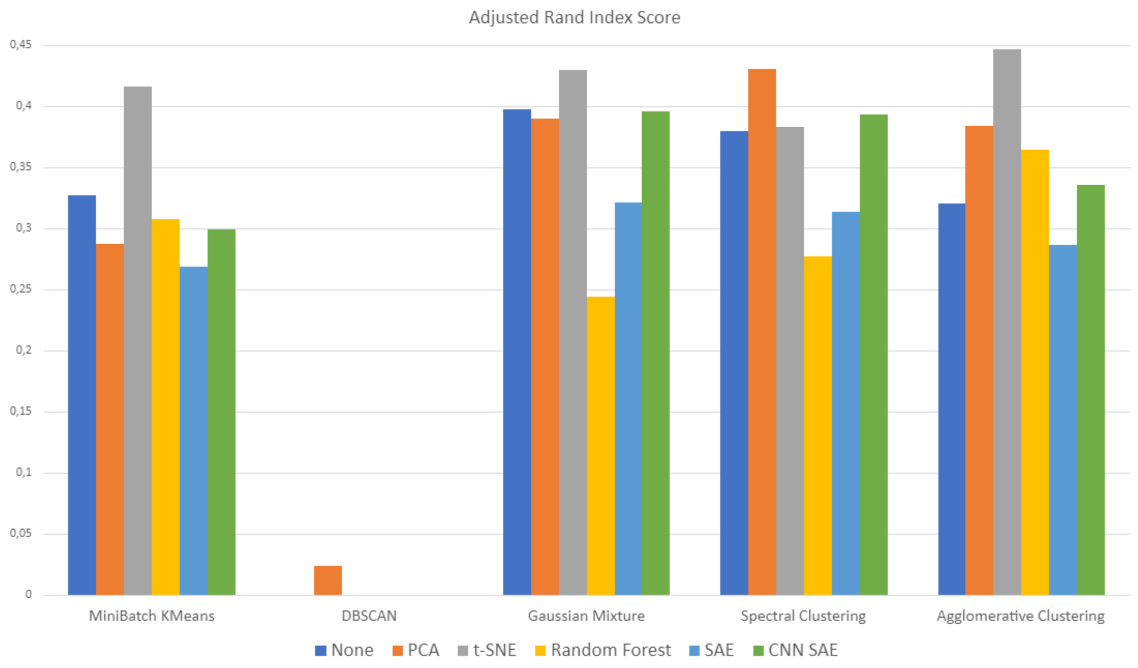


Figure 23: Adjusted Rand Index score for different combinations of Dimensionality and Clustering Techniques