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

Myriam Kapon, Eugenia Argyriadi

December 2023

Abstract

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-MNIST, Dimensionality Reduction, Clustering, Comparison

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!"

Image dimensionality reduction refers to the process of reducing the number of features or dimensions in an image dataset while preserving its essential information. It's aimed at condensing high-dimensional image data into a lower-dimensional space, making it more manageable for analysis, visualization, and often improving computational efficiency.

Feature extraction in image dimensionality reduction involves extracting and selecting important features from the images. These features might represent edges, textures, shapes, or other patterns present in the images. The goal is to retain the most informative aspects of the images while reducing their dimensionality.

Fashion-MNIST contains grayscale images of fashion items, with 28x28 pixels in size. Each image is represented as a matrix where each pixel's value ranges from 0 to 255, indicating its intensity. During this run, items indexed at [859, 9543, 4203, 194, 5881, 9255, 8974, 2077, 505, 7550] were randomly chosen for comparison against their respective reconstructed counterparts.

4.1 Principal Component Analysis

Principal Component Analysis (PCA) 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.

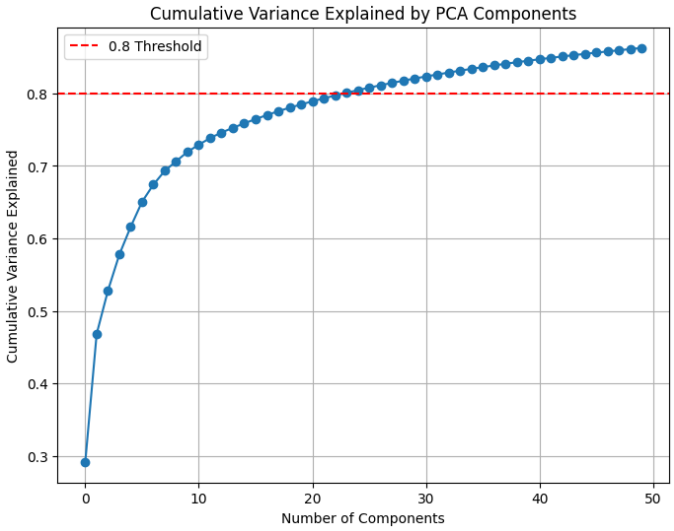


Figure 2: The Cumulative Variance explained by PCA components

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.

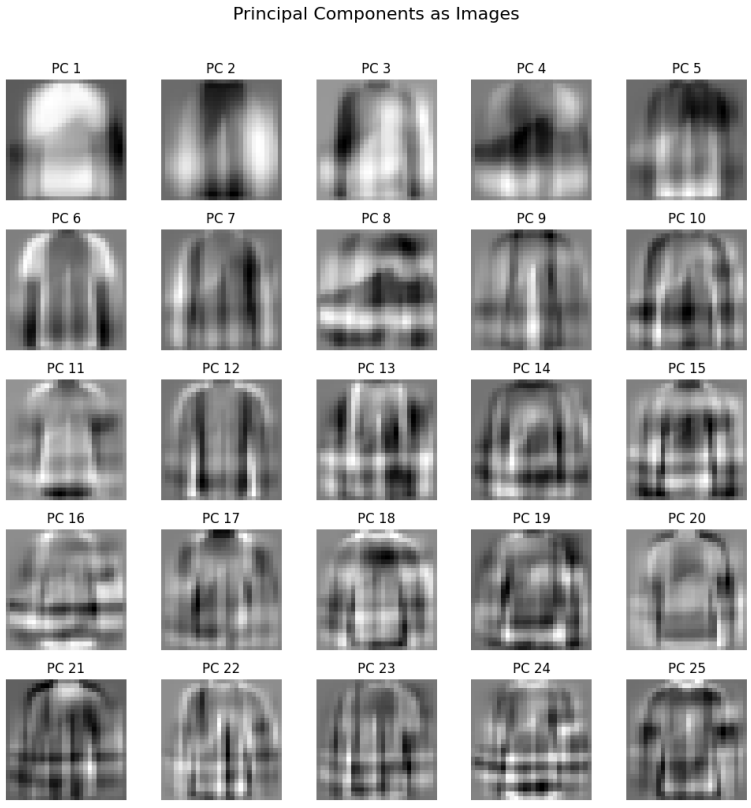


Figure 3: The 25 most important Principal Components

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.

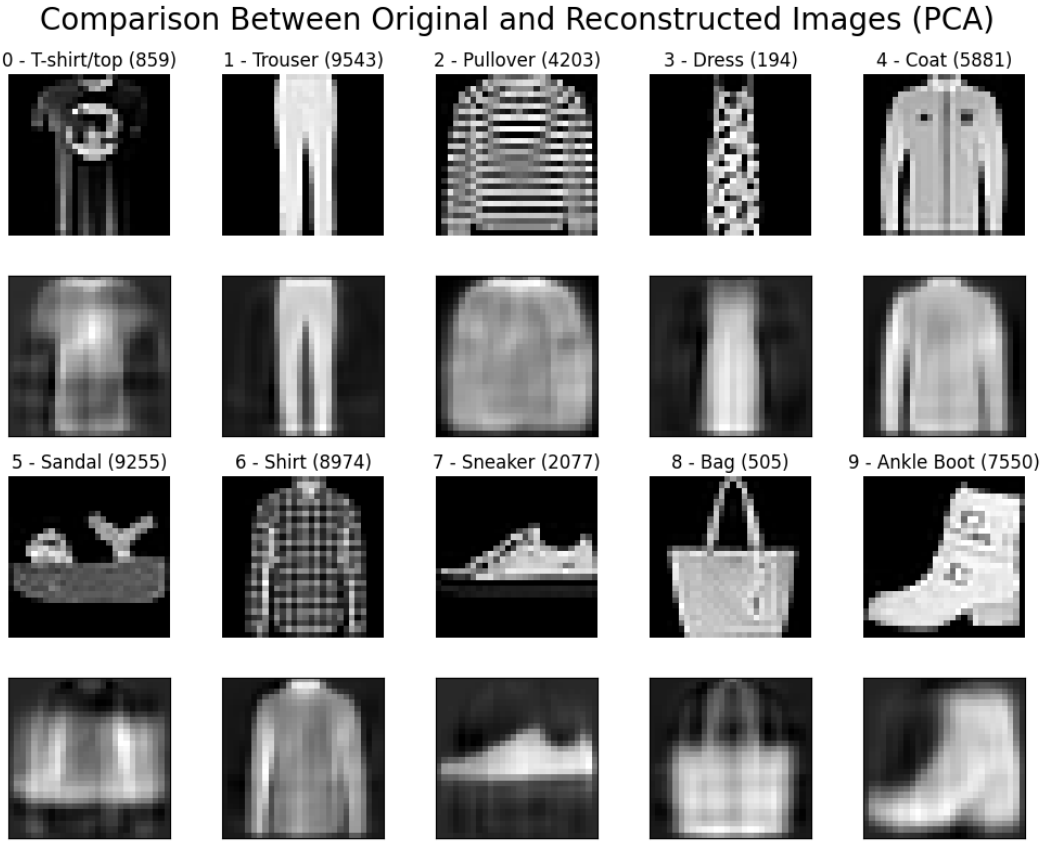
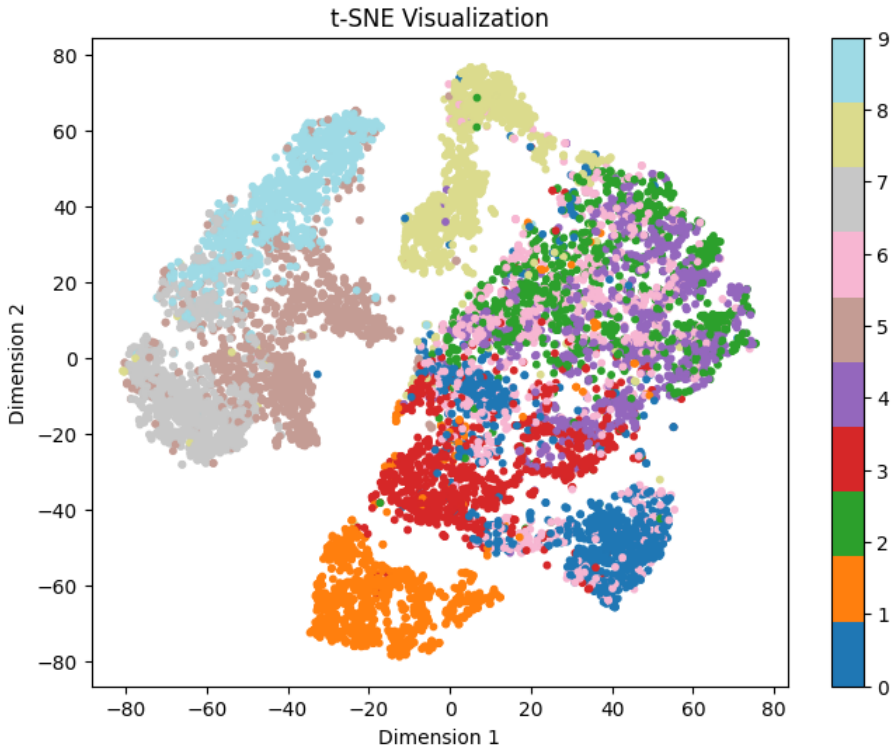


Figure 4: Comparison between Original and Reconstructed images when performing PCA

4.2 t-Distributed Stochastic Neighbor Embedding

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).

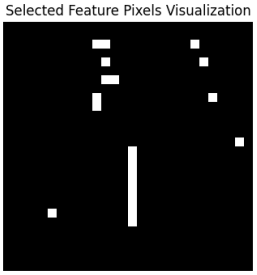


Observing this data map reveals intriguing patterns: Trousers (orange) form an isolated island within the data, indicating easier clustering. Similarly, Sandals (brown), Sneakers (grey), and Ankle Boots (light blue) form a distinct cluster, as do Bags (olive). These categories—trousers, shoes, and bags—show promising potential for forming well-defined clusters.

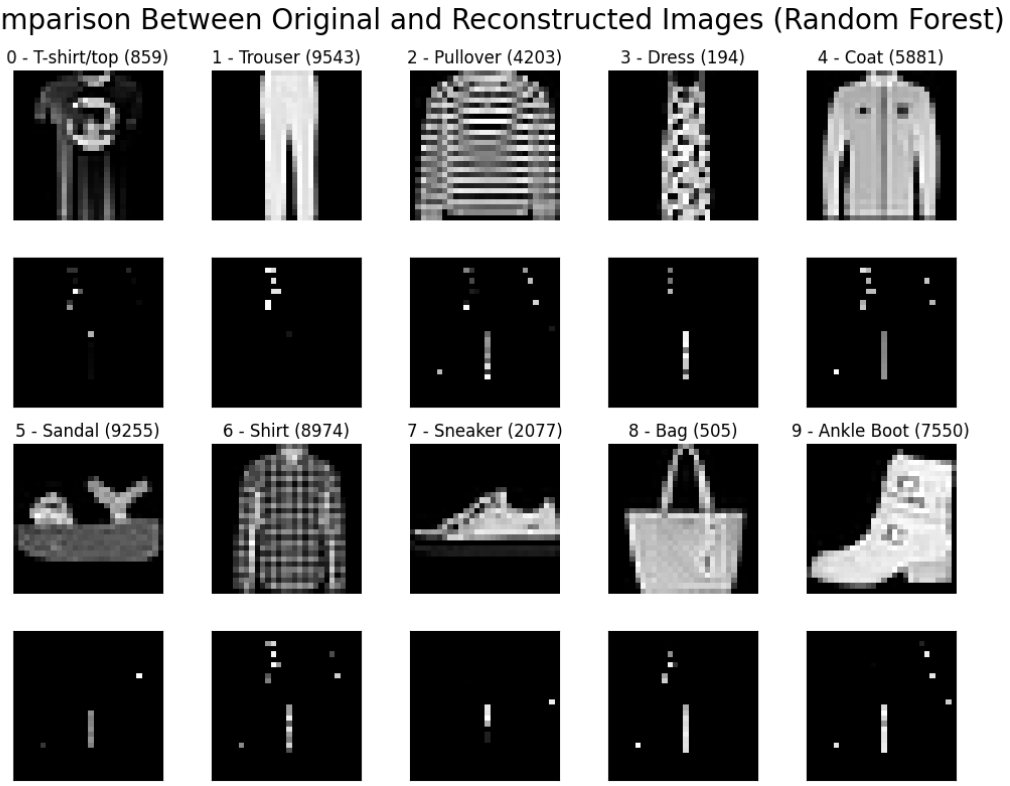
4.3 Random Forest

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 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, encompassing the principle of Occam's Razor.



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 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.

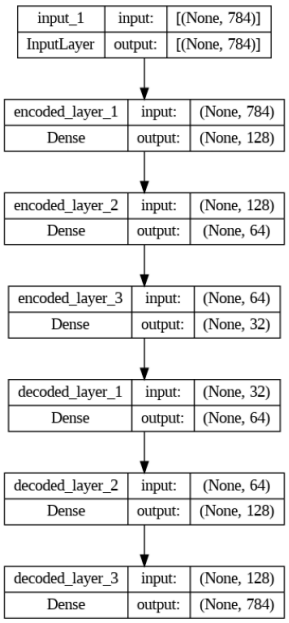


Figure 5: The architecture of the SAE Model

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.

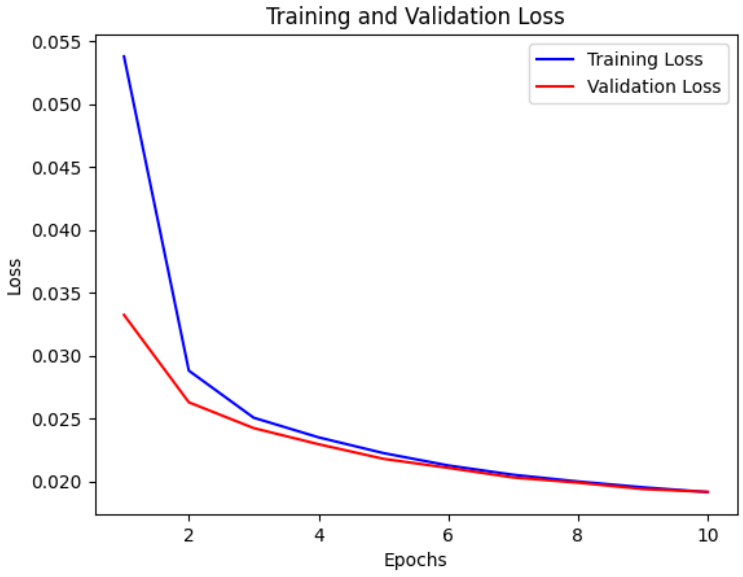


Figure 6: Loss function for the train and validation sets

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 **0.0191**.

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.

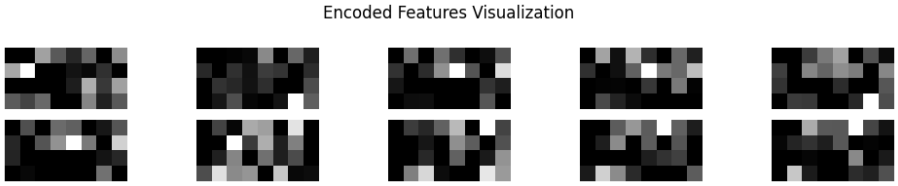


Figure 7: A selection of encoded features using a Stacked Autoencoder

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. Furthermore, there's a blurring effect around the edges, making for a not-so-perfect reconstruction.

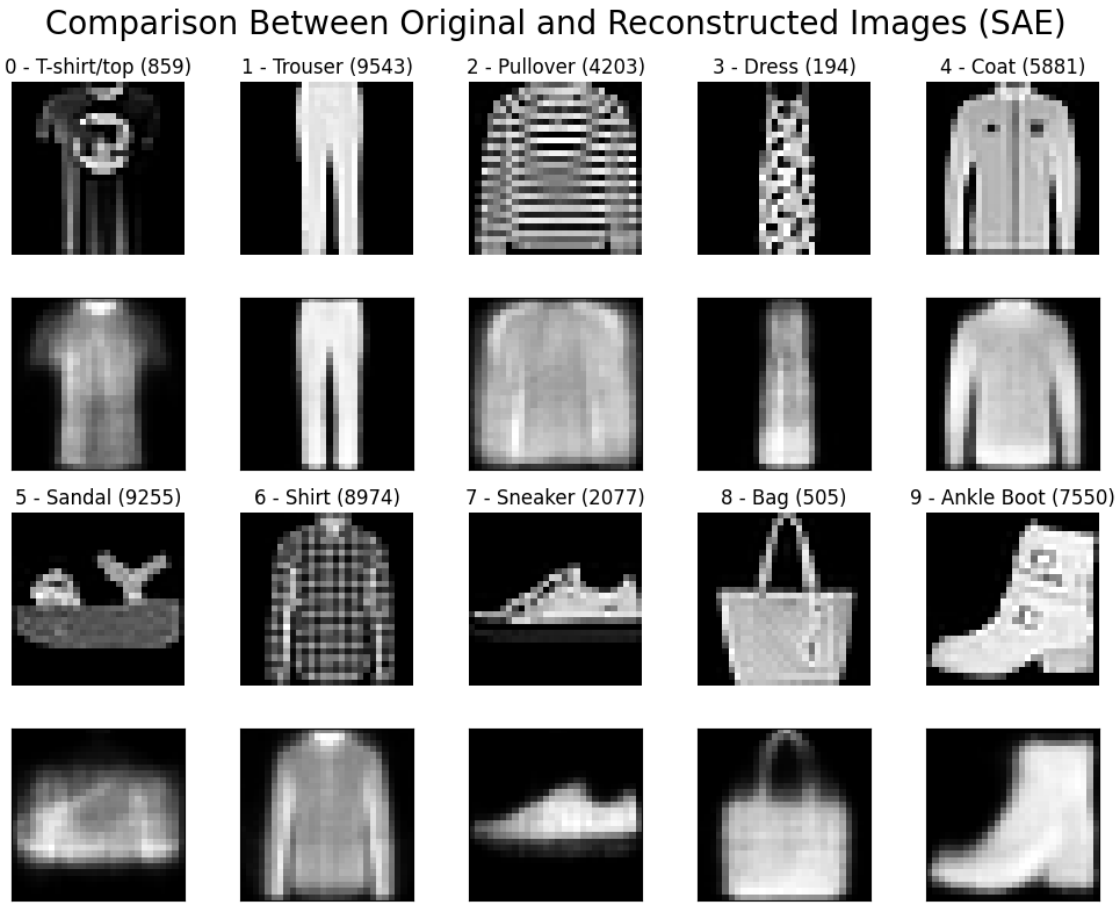


Figure 8: Comparison between the original and the reconstructed images using a Stacked Autoencoder

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 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.

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.

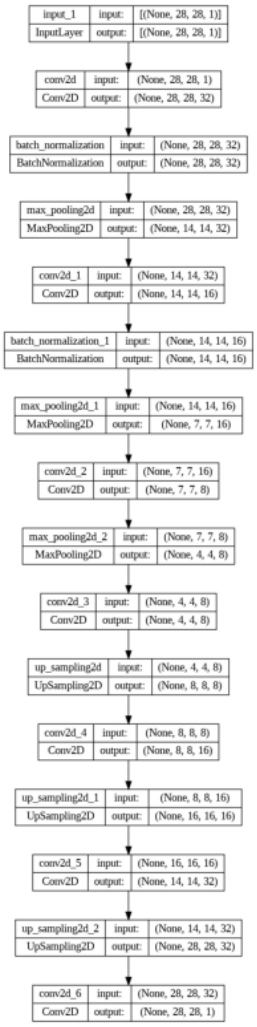


Figure 9: The architecture of the CSAE Model

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.

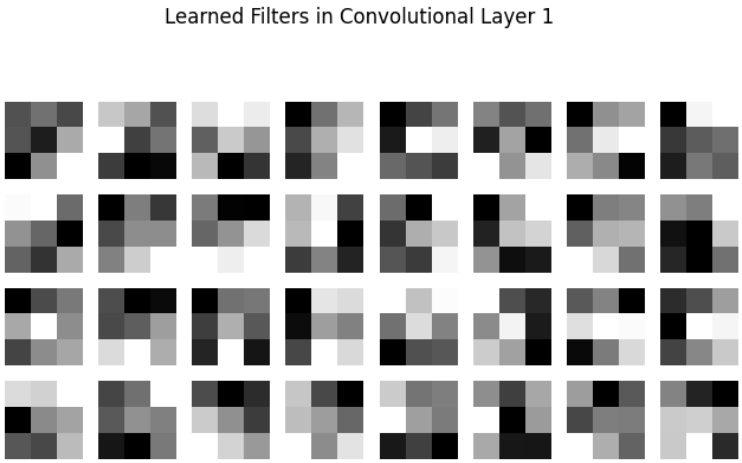


Figure 10: Learned Filters of the first Convolutional layer

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 **0.0134**.

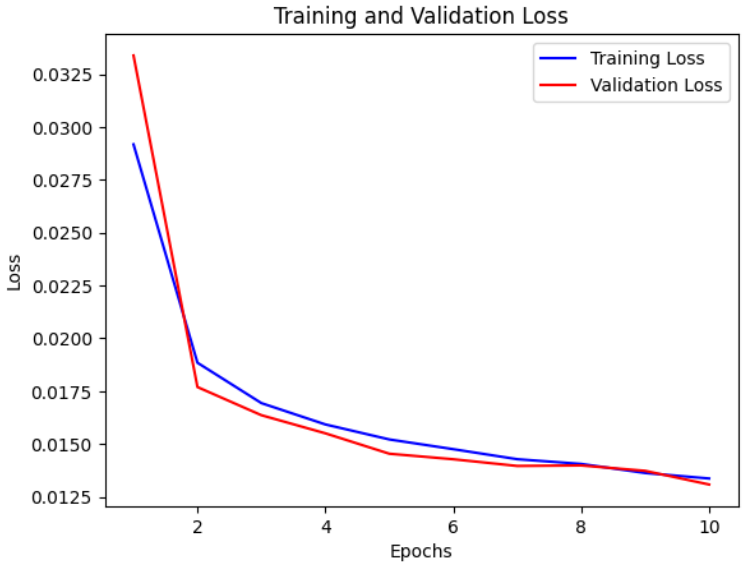


Figure 11: Loss function for the train and validation sets

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.

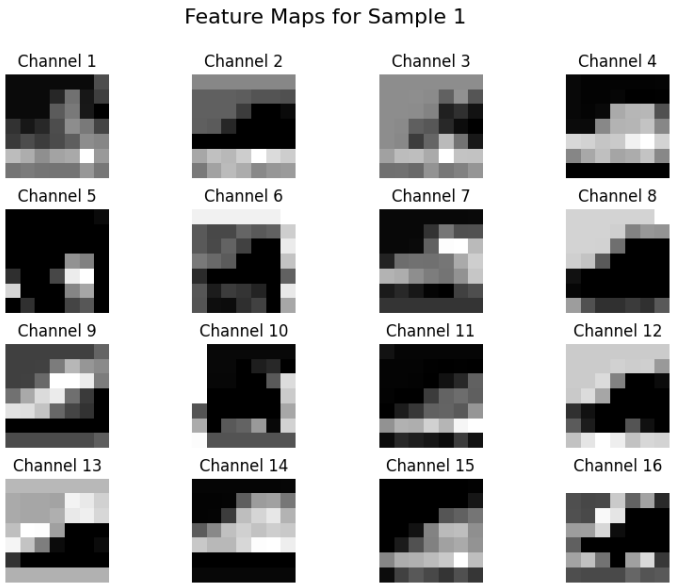


Figure 12: The feature maps of the first item using CSAE

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 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.

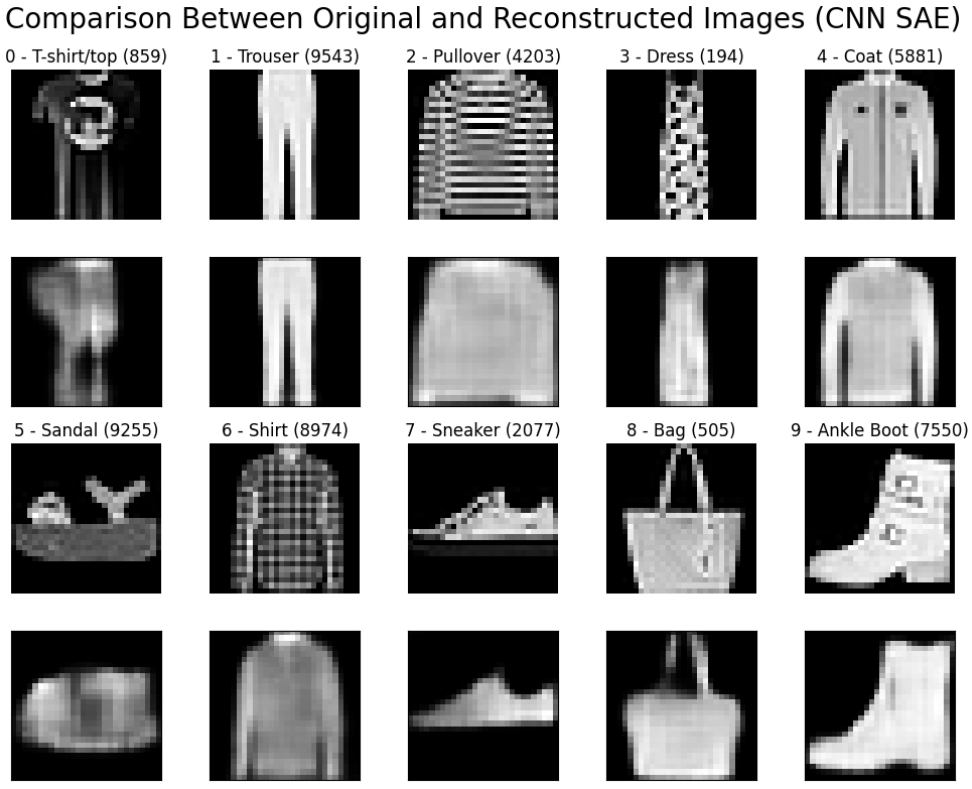


Figure 13: Comparison between Original and Reconstructed images when using a Convolutional Autoencoder

5 Clustering

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

Clustering, in essence, is the process of organizing a dataset into groups, where objects within each group are more similar to each other compared to those in other groups. It's a method used to uncover inherent patterns or structures within data without prior knowledge of group labels or categories, facilitating better understanding and analysis of complex datasets. In the context of Fashion-MNIST, clustering involves organizing the images of fashion items into distinct groups or clusters based on inherent similarities in their pixel intensities, textures, shapes, or patterns.

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.

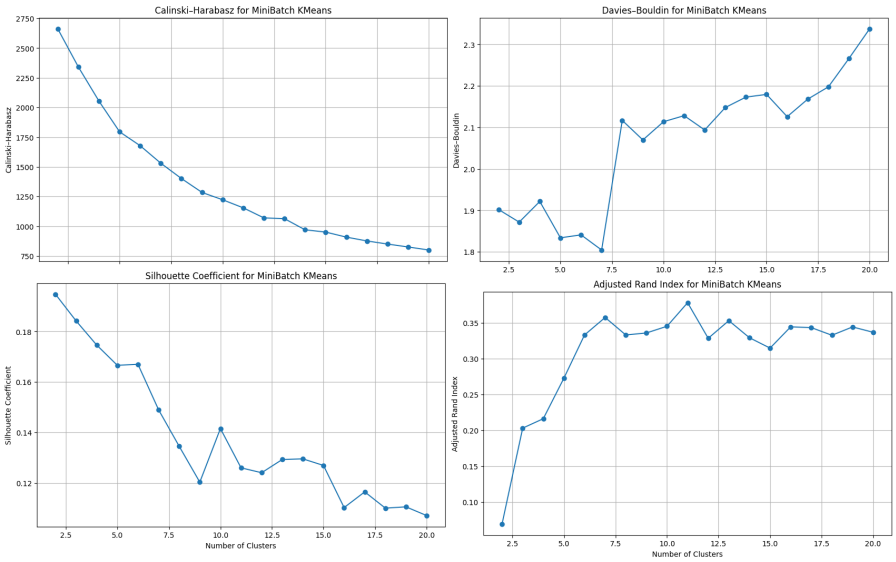


Figure 14: The score of different cluster numbers for Calinski-Harabasz, Davies-Bouldin, Silhouette and ARI metrics

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

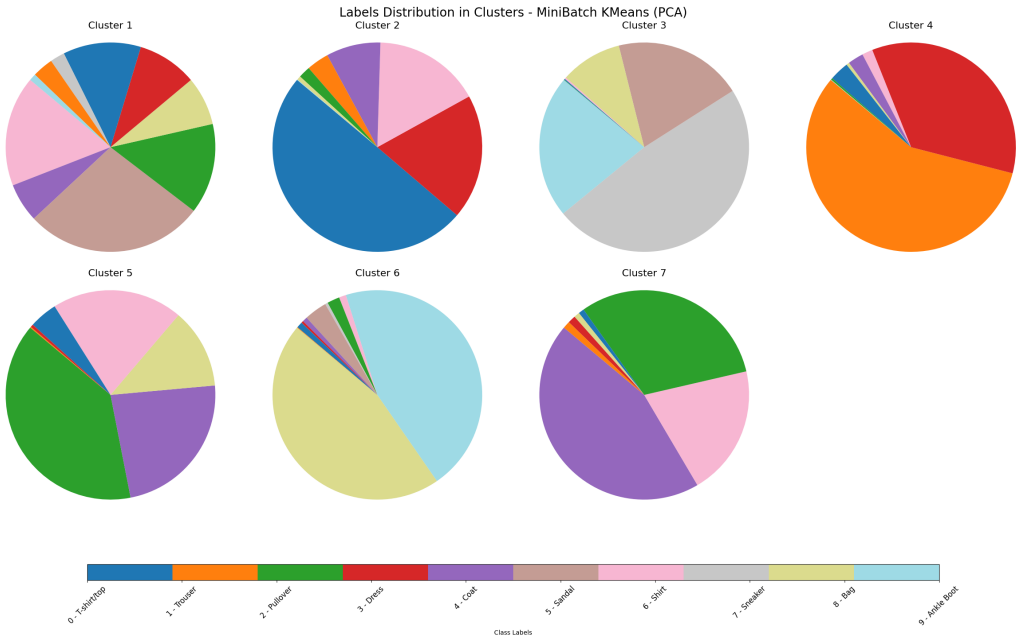


Figure 15: The distribution of labels within MiniBatch KMeans clusters using PCA for dimensionality reduction.

Pie charts were generated to visualize the contents of the 7 clusters formed by MiniBatch KMeans, providing an insightful depiction of the captured items within each cluster. Each chart illustrates the percentage distribution of clothing categories within the clusters. Notably, clusters 2, 3, 4, and 6 exhibit higher uniformity with fewer colors, indicating more homogeneity within these clusters. Interestingly, the algorithm shows better proficiency in identifying shoe categories—Sandals (brown), 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.

Dimensionality Reduction	Calinski-Harabasz	Davies-Bouldin	Silhouette Score	ARI
None	1458.14	2.12	0.14	0.33
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

According to the metric scores, the best dimensionality reduction technique for MiniBatch KMeans is a Convolutional Autoencoder, scoring the highest for Calinski-Harabasz, Davies-Bouldin and Silhouette Score. If the accuracy based on some ground truth is more important however, t-SNE should be preferred.

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.

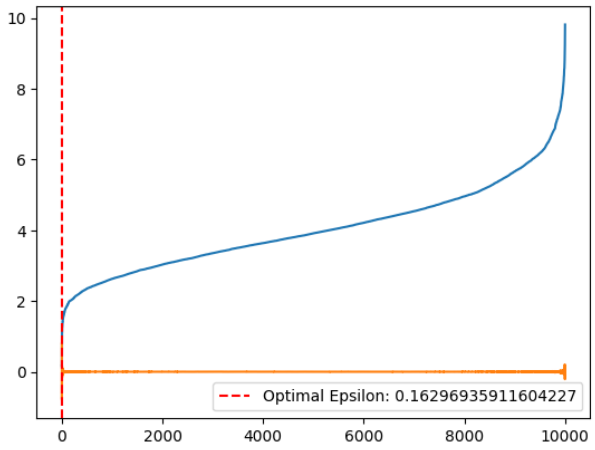


Figure 16: A k-distance graph showing optimal e values

However, DBSCAN is still perforimng poorly and is not recommended for the Fashion MNIST dataset, at least not with these parameters. The only result was given using PCA.

Dimensionality Reduction	Calinski-Harabasz	Davies-Bouldin	Silhouette Score	ARI
None	-	-	-	-
PCA	525.42	1.57	-0.035	0.024
t-SNE	-	-	-	-
Random Forest	-	-	-	-
SAE	-	-	-	-
CNN SAE	-	-	-	-

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

5.3 Gaussian Mixture Model

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.

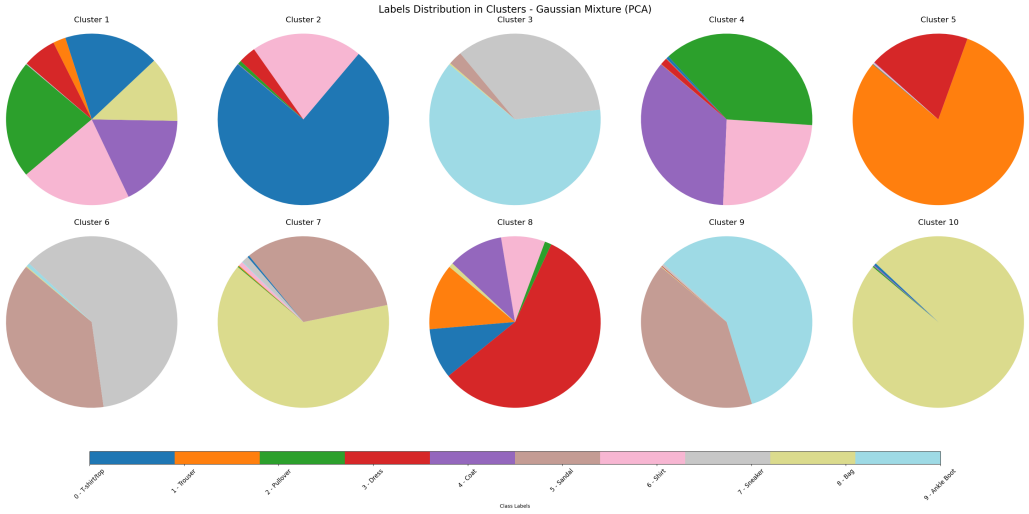


Figure 17: The distribution of labels within GMM clusters using PCA for dimensionality reduction.

GMM was chosen for its ability to perform good clusters on homogeneous data. A model 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 cluster memberships. This approach is particularly valuable when the dataset exhibits complex structures or overlaps between clusters.

DR \ Metric	Calinski-Harabasz	Davies-Bouldin	Silhouette Score	ARI
None	1053.56	2.09	0.12	0.40
PCA	820.58	3.08	0.09	0.39
t-SNE	1072.65	2.27	0.11	0.43
Random Forest	570.13	4.08	0.02	0.24
SAE	803.30	2.71	0.06	0.32
CNN SAE	1110.98	2.18	0.13	0.40

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

According to the metric scores, the best dimensionality reduction technique for Gaussian Mixture is a Convolutional Autoencoder, scoring the highest for Calinski-Harabasz, Davies-Bouldin and Silhouette Score. If the accuracy based on some ground truth is more important however, t-SNE should be preferred.

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.

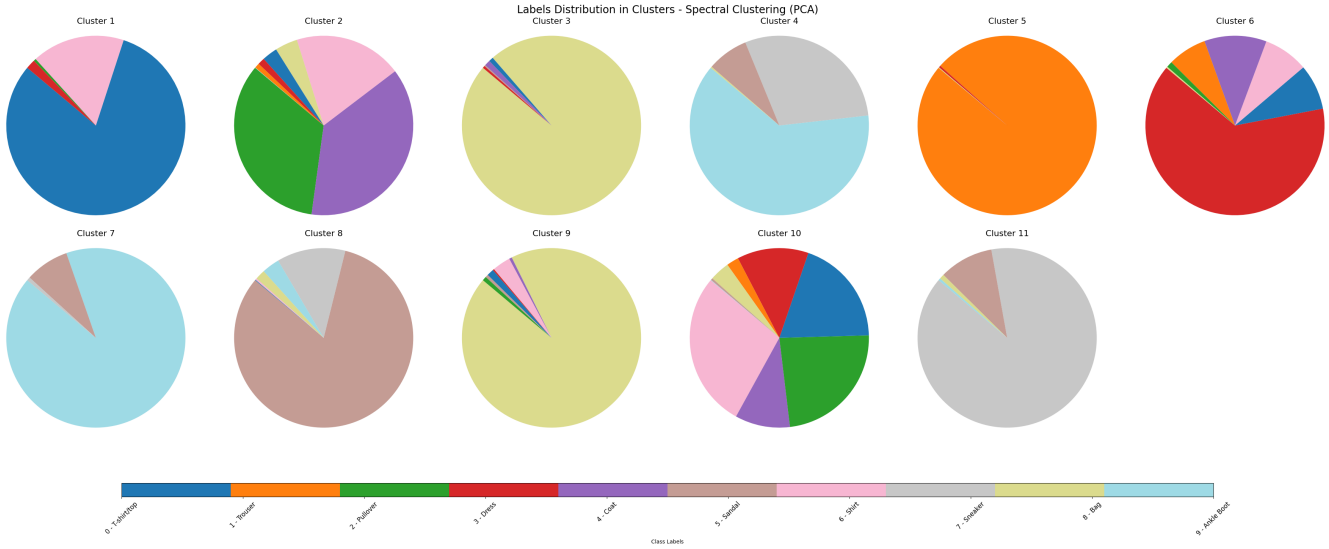


Figure 18: The distribution of labels within Spectral clusters using PCA for dimensionality reduction.

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.

DR \ Metric	Calinski-Harabasz	Davies-Bouldin	Silhouette Score	ARI
None	1117.81	1.98	0.15	0.38
PCA	1130.63	1.92	0.14	0.43
t-SNE	872.29	2.14	0.08	0.38
Random Forest	803.16	2.41	0.05	0.28
SAE	806.97	2.17	0.06	0.31
CNN SAE	1054.33	2.13	0.13	0.39

Table 4: Performance metrics for different dimensionality reduction techniques for Spectral 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.

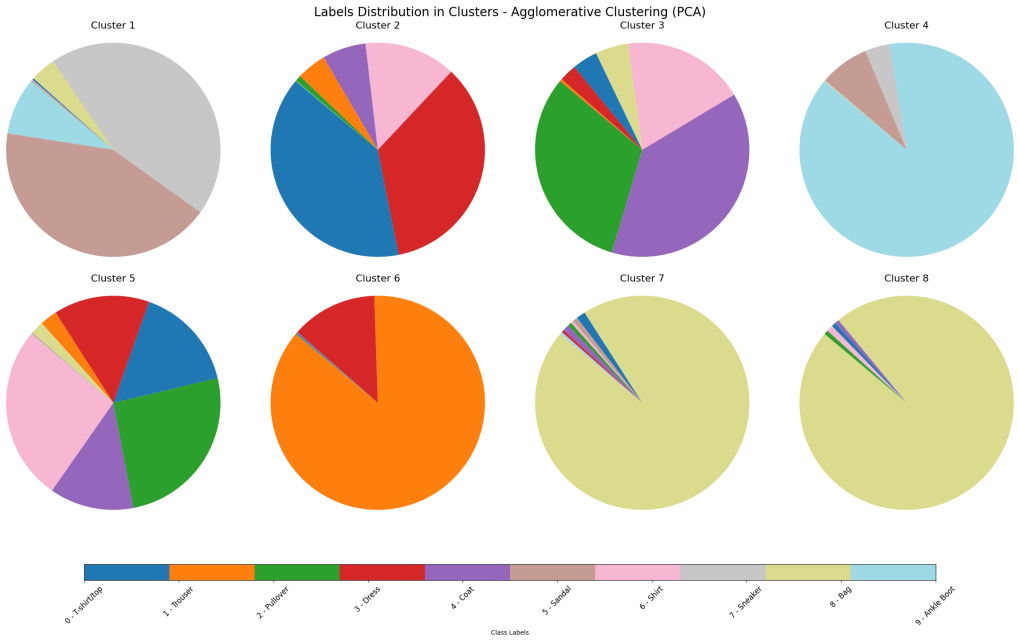


Figure 19: The distribution of labels within Agglomerative clusters using PCA for dimensionality reduction.

According to the metric scores, the best dimensionality reduction technique for Agglomerative Clustering is PCA, scoring the highest for Calinski-Harabasz, Davies-Bouldin and Silhouette Score. If the accuracy based on some ground truth is more important however, t-SNE should be preferred.

Metric	**Calinski-Harabasz**	**Davies-Bouldin**	**Silhouette Score**	**ARI**
None	1270.35	1.94	0.12	0.32
PCA	1301.24	1.78	0.15	0.38
t-SNE	1141.04	2.27	0.10	0.45
Random Forest	981.37	2.49	0.08	0.36
SAE	1000.29	2.48	0.08	0.29
CNN SAE	1202.28	2.20	0.11	0.34

Table 5: Performance metrics for different dimensionality reduction techniques with Agglomerative Clustering

6 Comparative Analysis

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 is why selecting the appropriate approach for each dataset is pivotal. Here are the outcomes obtained by combining various dimensionality reduction techniques (PCA, t-SNE, Random Forest, SAE, and CNN SAE) with several clustering algorithms (MiniBatch KMeans, DBSCAN, GMM, Spectral Clustering, and Agglomerative Clustering) to determine the most effective pair for yielding optimal results on the Fashion-MNIST dataset.

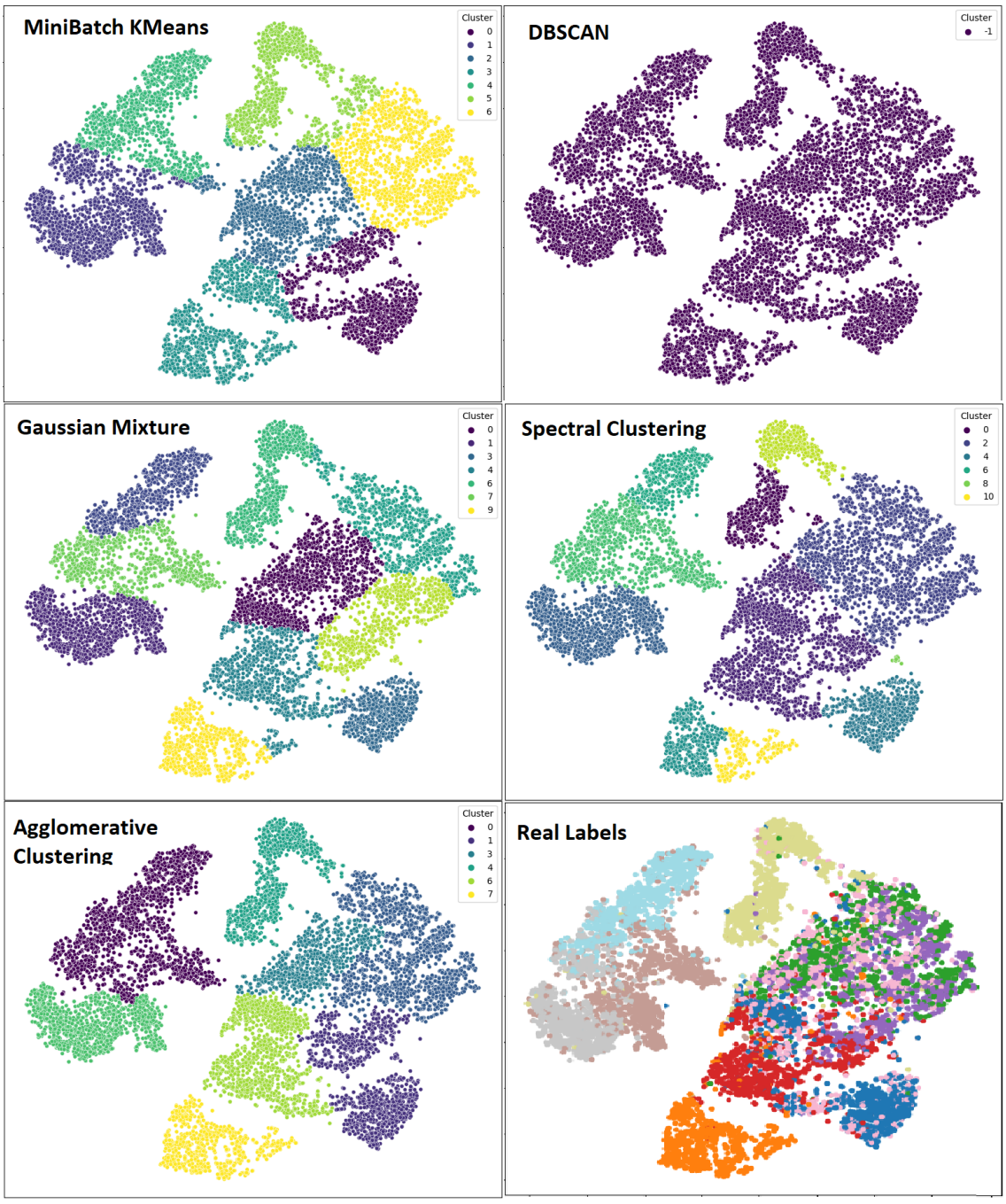


Figure 20: Clustering results of MiniBatch KMeans, DBSCAN, Gaussian Mixture, Spectral Clustering and Agglomerative Clustering on the Fashion-MNIST dataset

6.1 Overview

Starting with the averaged metric results, providing a quick overview by normalizing the scores and converting Davies-Bouldin values into their reciprocals. In this context, superior performance is indicated by higher values across all metrics.

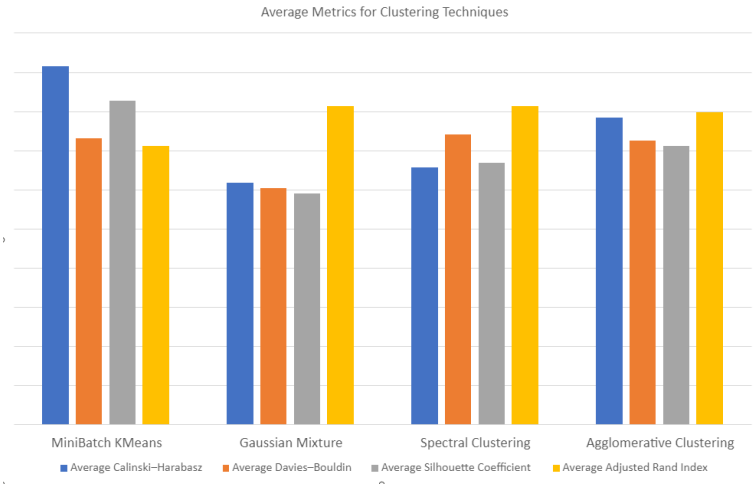


Figure 21: Average Metrics for Clustering Algorithms

In summary, MiniBatch KMeans demonstrated superior performance, securing the highest scores across three key metrics: Calinski-Harabasz, Davies-Bouldin, and Silhouette Coefficient. Its notably higher Calinski-Harabasz score suggests the creation of more distinct and well-defined clusters. However, for a more balanced approach, Agglomerative Clustering also excelled, particularly with a high Adjusted Rand Index (ARI), signifying clustering closer to the true labels. In essence, choosing KMeans would prioritize better-defined clusters, whereas Agglomerative Clustering leans toward more accurately aligned clusters with the true labels.

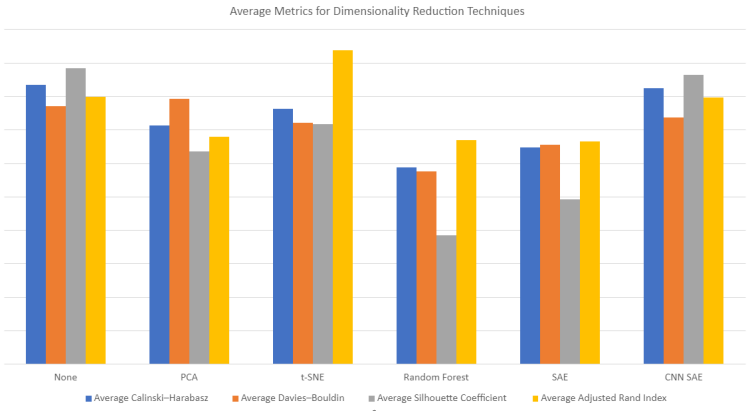


Figure 22: Average metrics for dimensionality reduction

Though not imperative, exploring the impact of various dimensionality reduction techniques on metric scores was intriguing. Surprisingly, clusters formed using the raw data exhibited higher quality. Nevertheless, CNN-SAE showcased nearly identical results with reduced dimensions, albeit accompanied by substantial training time. Notably, t-SNE generated clusters closest to the truth by representing the data in a 2-dimensional space, enabling better class distance visualization.

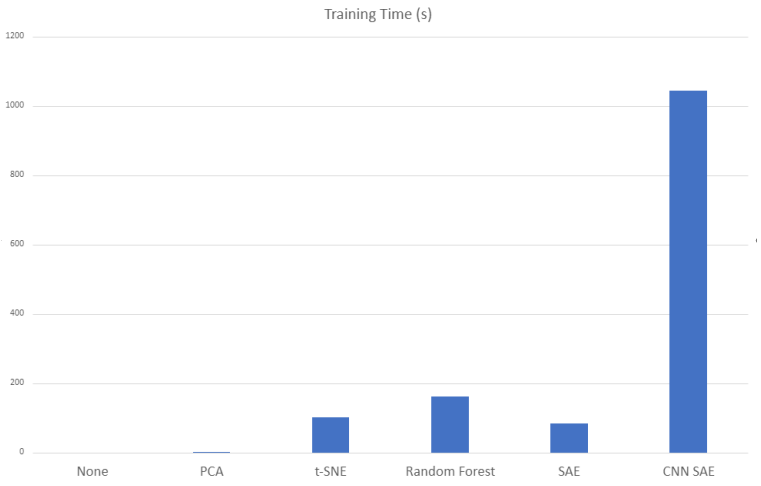


Figure 23: Training Time for Dimensionality Reduction Techniques

Regarding implementation time, CNN-SAE required the longest training duration, surpassing 1000 seconds, consistent with its model complexity. Remarkably, Gaussian Mixture Model exhibited notable inefficiency in combination with CNN-SAE, requiring an astonishing almost 300 seconds to execute—significantly higher compared to other execution times. Conversely, the quickest combination was observed between PCA and MiniBatch KMeans.

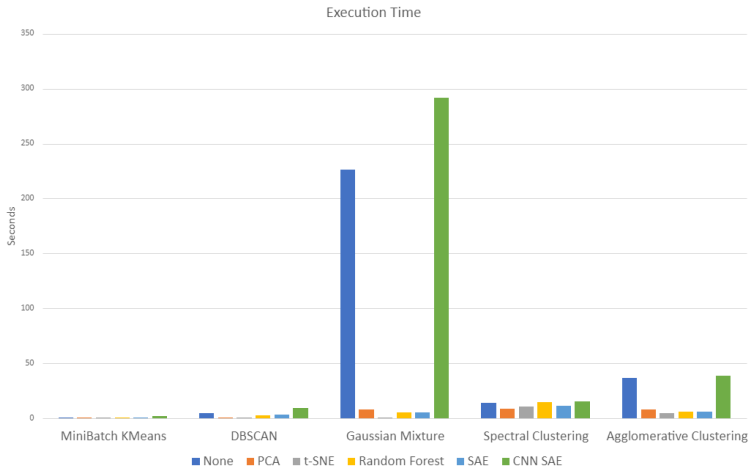


Figure 24: Average execution time for Clustering Techniques

6.2 Metric Comparison

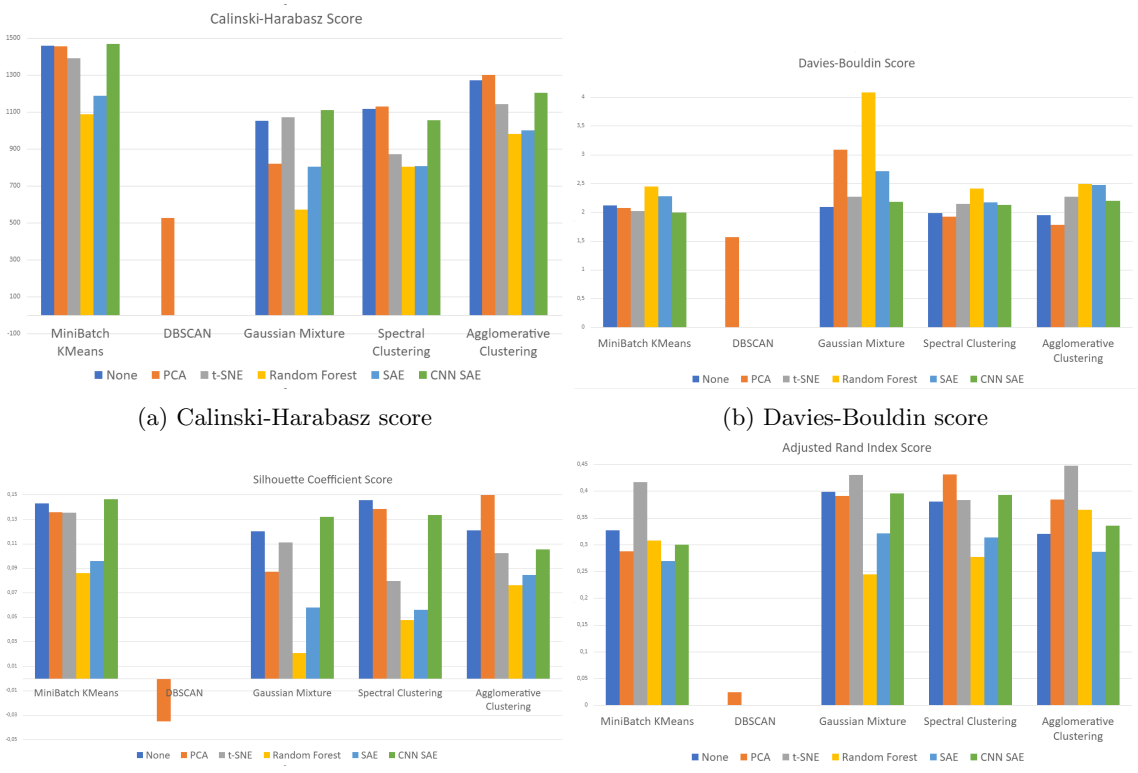
Examining the Calinski-Harabasz scores reveals distinctive performances among various clustering algorithms. Notably, K-Means clustering exhibited robust and consistent performance, achieving the highest Calinski-Harabasz score across all employed dimensionality reduction techniques. In contrast, DBSCAN, renowned for its proficiency in identifying dense regions and noise handling, displayed a suboptimal performance. Specifically, when applied with PCA, DBSCAN yielded the lowest Calinski-Harabasz score, and for other dimensionality reduction techniques, it formed a single cluster, rendering score computation unfeasible. Gaussian Mixture Model (GMM) and Spectral Clustering demonstrated comparable performance, with the latter slightly outperforming the former. Intriguingly, Agglomerative Clustering produced results ranking second after K-Means, yet the proximity of its scores to those of Gaussian and Spectral methods suggests

a nuanced similarity between these clustering approaches, diverging from the distinctiveness observed with K-Means.
The three combinations with the best performance for Calinski-Harabasz were:

1. CNN-SAE - MiniBatch KMeans with a score of 1467.91, indicating well-defined and distinct clusters.
2. No Dimensionality Reduction - MiniBatch KMeans with 1458.14.
3. PCA - MiniBatch KMeans with 1456.60.

DBSCAN stands out with the most favorable Davies-Bouldin score among all five clustering techniques. However, a notable pattern emerges as it consistently yields undefined scores for alternative dimensionality reduction techniques, consolidating the data into a single cluster. Gaussian Mixture, in contrast, exhibits a notably high score when subjected to both Random Forest and PCA reduction methods. Meanwhile, K-Means, Spectral Clustering, and Agglomerative Clustering display minimal distinctions in their Davies-Bouldin scores, suggesting comparable performance across these three methods.
The three combinations with the best performance for Davies-Bouldin were:

1. PCA - DBSCAN with a score of 1.57, showing relatively well-separated clusters and lower intra-cluster similarity.
2. PCA - Agglomerative Clustering with 1.78.
3. PCA - Spectral Clustering with 1.92.



(a) Calinski-Harabasz score

(b) Davies-Bouldin score

(c) Silhouette Coefficient score

(d) Adjusted Rand Index score

Figure 25: Evaluation scores for different combinations of Dimensionality Reduction and Clustering Techniques

Delving into the Silhouette Coefficient scores reveals distinct patterns across clustering techniques. K-Means demonstrates consistent performance across the dimensionality reduction techniques, maintaining relatively equal scores. Gaussian Mixture exhibits a notable divergence, achieving a high score when subjected to CNN SAE reduction but plummeting significantly when reduced by Random Forest. Spectral Clustering attains the highest score without any dimensionality reduction, sharing this distinction with K-Means. However, its performance diminishes notably when subjected to Random Forest and SAE reduction. Agglomerative Clustering ranks second overall in average results, trailing behind K-Means, and excels particularly when subjected to PCA reduction. Meanwhile, DBSCAN continues to exhibit suboptimal performance, forming a single cluster for all techniques except PCA, where it garners a negative score, indicating performance worse than random.
The three combinations with the best performance for the Silhouette Coefficient were:

1. PCA - Agglomerative Clustering with a score of 0.15, implying reasonable separation between clusters and a good balance between cohesion and separation.
2. CNN SAE - MiniBatch KMeans with 0.15.
3. None - Spectral Clustering 0.145.

In the last subplot, Gaussian, Spectral, and Agglomerative Clustering exhibit closely aligned Adjusted Rand Index scores, with Gaussian potentially edging slightly ahead. K-Means, while still performing well, lags slightly behind this trio. Conversely, DBSCAN consistently forms a single cluster for all techniques except PCA, where it attains a noticeably low Adjusted Rand Index score, signifying a suboptimal performance compared to random clustering.
The three combinations with the best performance for Adjusted Rand Index were:

1. t-SNE - Agglomerative Clustering with a score of 0.45, indicating a relatively strong agreement between predicted clusters and true labels.
2. PCA - Spectral Clustering with 0.43.
3. CNN SAE - Spectral Clustering with 0.39.

Since there is no clear optimal combination, our strategy involved employing a composite score derived from the harmonic mean of individual metric scores, with the exception of the Davies-Bouldin Index. For this specific metric, we utilized its reciprocal since a lower value is more favorable.

Technique	**PCA**	**t-SNE**	**RF**	**SAE**	**CNN SAE**
MiniBatch KMeans	0.31	0.339	0.231	0.244	0.329
Gaussian Mixture	0.234	0.294	0.071	0.174	0.325
DBSCAN	0.276	nan	nan	nan	nan
Spectral Clustering	0.349	0.231	0.148	0.172	0.329
Agglomerative Clustering	0.362	0.28	0.218	0.225	0.273

Table 6: Performance scores for different clustering methods using various feature extraction techniques.

The combination that performed the best for Fashion-Mnist is PCA with Agglomerative Clustering The metrics for Agglomerative Clustering using PCA suggest generally good cluster separation (Calinski-Harabasz), moderate but not optimal cluster quality (Davies-Bouldin, Silhouette Score), and a moderate level of agreement between true labels and clusters (ARI). Fine-tuning or additional analysis might enhance clustering performance.

6.3 Image Reconstruction

To assess the quality of image reconstruction, four case studies were chosen to evaluate PCA, SAE, and CNN SAE in sequence. These are clothing items with some peculiarity, specifically: An item with noise, an unusual item, and item with an intricate shape and an item with a simple shape.

Starting with the first item, dimensionality reduction plays a crucial role in noise reduction within datasets, thereby simplifying both supervised and unsupervised learning tasks. By reducing the number of features or variables, redundant or irrelevant information is minimized, enhancing the model's ability to focus on significant patterns or signals present in the data. This process helps in noise removal as it identifies and retains essential information, filtering out irrelevant or redundant aspects that might cause interference or confusion in learning algorithms.

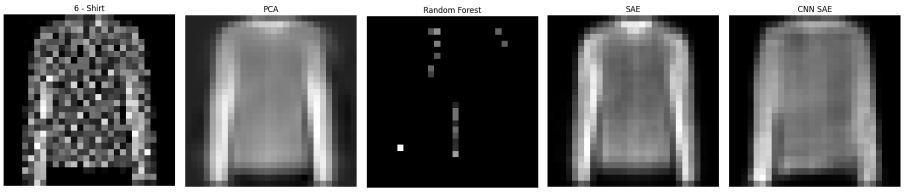


Figure 26: Reconstructed item 562: A shirt containg a lot of noise.

Regarding the comparison among dimensionality reduction techniques (excluding Random Forest), all performed admirably in removing patterns from the Shirt, transforming it into a more generic shape. However, the most effective image reconstruction was achieved by PCA. Despite removing the pattern, PCA retained the gradient color of the sleeves. Conversely, CNN SAE exhibited the least favorable performance, resulting in blurry edges during reconstruction.

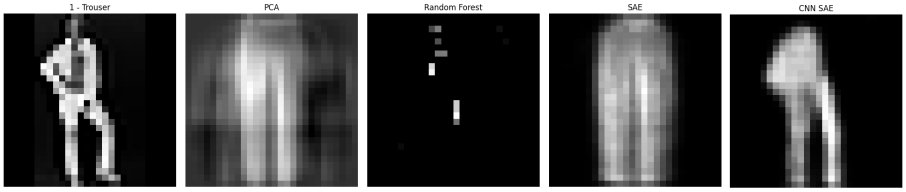


Figure 27: Reconstructed item 1592: A man wearing trousers.

This particular item was selected to assess how various dimensionality reduction techniques handle outliers. A notable observation emerged: both PCA and SAE reconstructed the image into a generic trouser representation, a beneficial transformation for learning algorithms, as it helps mitigate the impact of outliers by creating a more standardized form.

Interestingly, CNN SAE took a different approach by generating an image that closely resembles the original. This adaptation highlights its agility and astuteness, quickly adapting to the outlier while retaining more features akin to the original image.

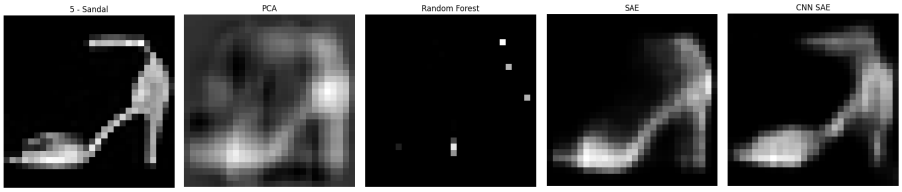


Figure 28: Reconstructed item 5152: A sandal with a lot of empty space.

Image reconstruction techniques faced challenges when dealing with sandals due to their intricate shape and similarity to other shoe types. The resulting reconstructed images consistently exhibited blurriness, filled holes, or even resembled entirely different shoes, primarily because of the sandals' intricate design. This sandal, characterized by numerous thin edges such as the heel, highlighted CNN SAE's proficiency once more. Although slightly blurry, CNN SAE produced an image remarkably close to the original, particularly in capturing intricate details, showcasing its strength in handling complex shapes with finer edges.

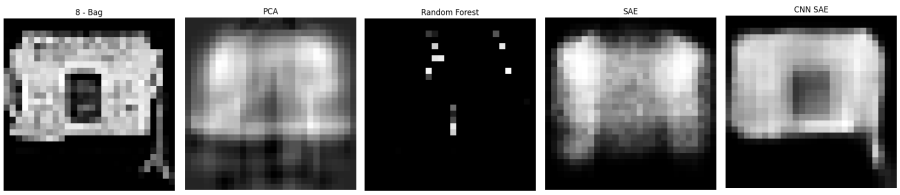


Figure 29: Reconstructed item 9856: A bag shaped like a solid rectangle

Last but not least, a simple rectangular shape of a bag was tested. Here the stark difference in image reconstruction capabilities becomes evident, highlighting the superiority of CNN SAE. Despite SAE also being a neural network, it generated a shapeless representation where the four straight lines, seemingly the simplest to capture, appeared distorted. This discrepancy might be attributed to confusion caused by the bag's tassels, a feature that CNN SAE elegantly captured in its reconstruction.

7 Conclusions

The evaluation of dimensionality reduction and clustering techniques on Fashion-MNIST highlighted the trade-offs between computational efficiency, clustering accuracy, and image reconstruction quality. MiniBatch KMeans consistently formed distinct clusters, prioritizing clear separations, while Agglomerative Clustering aligned closely with true labels, emphasizing accuracy. The choice between these methods depends on the desired outcome—clear cluster definition or label alignment.

Among dimensionality reduction techniques, PCA excelled in computational efficiency and noise reduction, producing generic representations suitable for learning algorithms. Conversely, CNN SAE showcased detailed reconstructions but demanded higher computational resources. Tailoring technique selection based on data characteristics and task requirements is crucial.

In summary, the optimal choice for Fashion-MNIST relies on balancing computational efficiency, clustering accuracy, and image reconstruction fidelity. Agglomerative Clusterings with PCA emerges as the most promising combination, offering a robust approach—PCA simplifies the data, aiding clustering efficiency, while Agglomerative Clustering leverages this reduced representation to create clusters that closely align with the true labels or underlying patterns within the data.

7.1 Further Analysis

Although the report offered an extensive analysis, there remain avenues for further exploration. Replicating the experiment with each clustering algorithm configured to ten clusters, mirroring the number of classes, could provide insight into how these clusters relate to reality, despite potentially not optimizing internal metrics. This comparison might reveal the proximity of clusters to real-world classifications.

Another facet worth exploring is the potential of Random Forest as a dimensionality reduction technique. This concept holds promise, particularly with the identification of an optimal threshold. Images often contain extraneous information for classification or clustering. The ability to distill this wealth of data down to a select few strategic and valuable pixels carries significant power and potential for enhancing efficiency in image analysis.

Building on this, although DBSCAN showcased suboptimal performance on the dataset, there exists potential for improvement. Diluting the data, possibly through leveraging Random Forest or mapping it onto a two-dimensional plane, could offer avenues to enhance DBSCAN's effectiveness and derive more meaningful insights from the dataset.

Above all, we encourage people to stay curious, experiment, and have fun in this ever expanding field, learning together with machine learning algorithms.

8 References

- Sefifian, A.M. (2023) How to determine Epsilon and MinPts parameters of DBSCAN clustering, Amir Masoud Sefidian - Sefidian Academy. Available at: http://www.sefidian.com/2022/12/18/how-to-determine-epsilon-and-minpts-parameters-of-dbscan-clustering/ (Accessed: 23 December 2023).
- 2.3. clustering (no date) scikit. Available at: https://scikit-learn.org/stable/modules/clustering.html (Accessed: 23 December 2023).
- Chollet, F. (no date) The keras blog, The Keras Blog ATOM. Available at: https://blog.keras.io/building-autoencoders-in-keras.html (Accessed: 23 December 2023).
- Simone (1963) Adjusted rand index vs adjusted mutual information, Cross Validated. Available at: https://stats.stackexchange.com/questions/260487/adjusted-rand-index-vs-adjusted-mutual-informatio (Accessed: 23 December 2023).
- Surya T. (2022) convolutional vs Stacked Autoencoder for images, Kaggle. Available at: https://www.kaggle.com/code/tejasurya/convolutional-vs-stacked-autoencoder-for-images (Accessed: 23 December 2023).