A kernel methods / DL pipeline for the FashionMNIST dataset

Advanced Deep Learning and Kernel Methods

Luis Fernando Palacios Flores *[1]

[1] *University of Trieste*

This report outlines my approach to addressing the challenge's objective, as detailed in the assignment: developing a data analysis pipeline that leverages unsupervised machine learning techniques, such as PCA and kernel-PCA, to cluster a dimensionally reduced dataset, assign labels to its data points, and subsequently train various supervised classification models. These models, including kernel-SVM and Artificial Neural Networks, were trained on the FashionMNIST dataset using both generated and original labels for comparison. The results highlight the inherent difficulties of expert labeling. My complete solution is available on my GitHub page. Finally, I acknowledge using ChatGPT to debug and enhance my code and to better understand some of the functions employed in the challenge.

1 Understanding data geometry

The FashionMNIST dataset is a benchmarking dataset containing 70k grayscale images of 28x28 pixels, representing 10 categories of clothing and accessories. As shown in Figure 1, examples from each class reveal that some categories share similar features or shapes, such as the T-shirt and Dress classes or the Pullover, Coat, and Shirt classes.

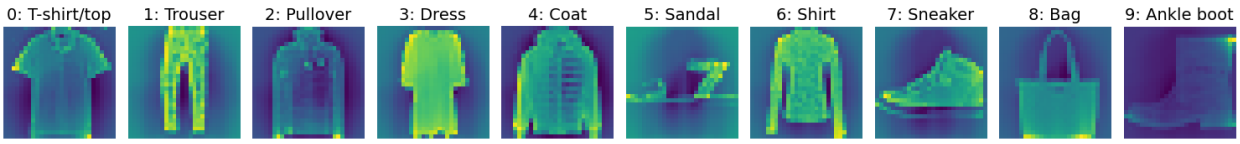

Figure 1: FashionMNIST examples. Each image is titled with its corresponding label and class name.

This dataset is high-dimensional, with each image represented by 784 features. To explore its geometric structure, I applied linear PCA and kernel PCA with Gaussian and Polynomial kernels. Given the similarity between classes (the features aforementioned), non-linear separability is anticipated, making kernel-based PCA potentially more effective for capturing the data's structure.

I used a normalized training dataset to ensure the proper application of PCA and kernel PCA. For linear PCA, I utilized all examples from the training dataset, whereas for kernel PCA, I worked with a subset consisting of every fifth data point due to RAM limitations.

The parameter γ in the Gaussian kernel controls the dispersion or scale of the Gaussian function, determining the kernel's sensitivity to distances between data points. When data points are farther apart than the scale defined by γ, their similarity approaches zero. A small γ can make the kernel nearly linear, as the exponential function approximates linearity in its first-order expansion for sufficiently small γ, reducing the impact of distances. The optimal value of γ depends heavily on the data distribution.

I used several heuristics to compute values for γ, such as the inverse of half the variance (0.5 due to normalization), the inverse of the number of features (a method used by `scikit-learn`), and the inverse of the mean squared difference between all data points. These heuristics yielded γ values in the range of 10^{-1} to 10^{-4}. I opted to test powers of 10 within the range 10^{1} to 10^{-3}, including higher values than those suggested by the heuristics to account for the observed similarity in the class features.

*luisfernando.palaciosflores@studenti.units.it

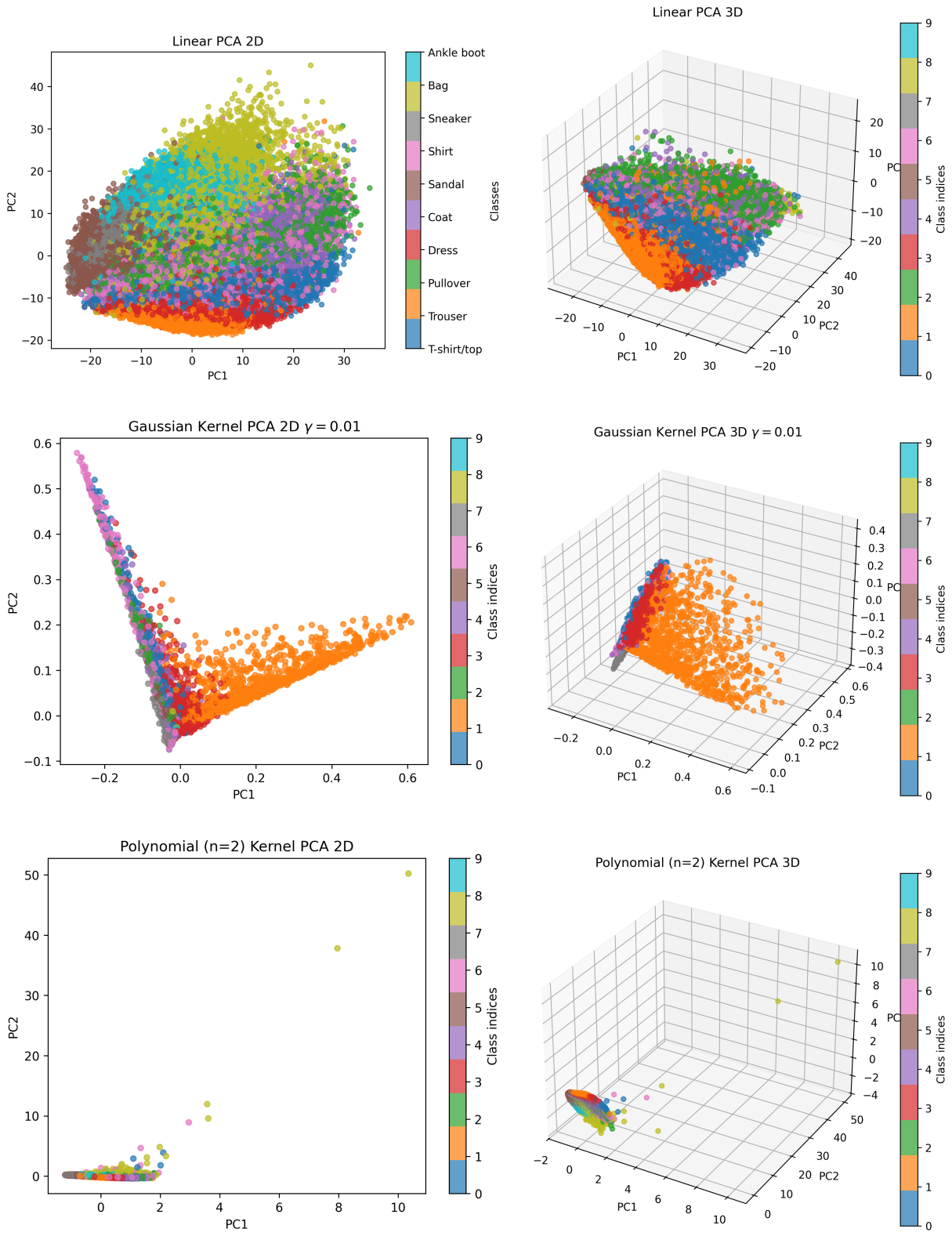

Figure 2: Principal Component Analysis results in two and three dimensions.

For the Polynomial kernel, I experimented with polynomials of degrees 2, 3, and 4 to capture relationships between features beyond the linear case. For the γ parameter, I chose the default value (10^{-3} in this case), as the magnitudes of the maximum and minimum dot products between the feature vectors ranged between 10^3 and 10^5 in absolute value, which might already lead to complex decision boundaries and overfitting. Although I tested other γ values, the results remained largely unchanged.

Figure 2 displays the results of applying PCA and kernel PCA in both two and three dimensions. While non-linear relationships between features are evident, the principal components fail to separate the "numerous" data points into "well-defined" clusters in these low-dimensional projections, resulting in significant class overlap. This aligns with earlier observations of similar features across classes, as illustrated in Figure 1.

We can see that the Gaussian kernel with $\gamma = 0.01$ projected the data points into seemingly orthogonal components, but only the Trouser class was clearly separated from the rest. In contrast, the polynomial kernel performed poorly at the task across all degrees, grouping most data points in a small, overlapping region with a few outliers. For the Gaussian kernel, γ values larger than 0.01 produced results similar to the polynomial kernel, while smaller values yielded results comparable to the linear kernel, although the decision boundary could be slightly different. Overall, these plots alone aren't informative enough to decide which dimensionality reduction technique "explains the data geometry better".

2 Bridging unsupervised and supervised

In this section, the goal is to select the most suitable method for dimensionality reduction and assign labels to the data points as part of a data labeling exercise. I chose to use the clustering techniques KMeans and Agglomerative Clustering because they can handle a "large number" of samples and employ different clustering strategies, allowing me to explore contrasting approaches. KMeans is a general-purpose method that works well with flat geometries and follows an inductive approach to create clusters. In contrast, Agglomerative Clustering supports non-Euclidean distances and adopts a transductive clustering approach [1]. Although systematically testing multiple clustering methods is not the most efficient approach, it provided me valuable insights into the behavior and performance of different techniques.

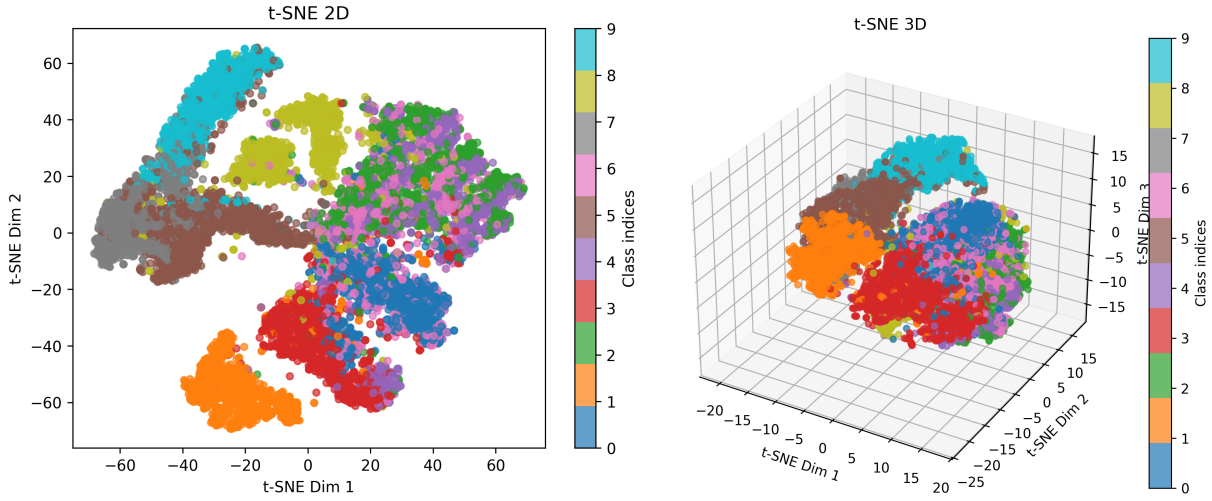

Figure 3: Dimensionality reduction with t-SNE in 2D and 3D.

Here, PCA and kernel PCA are not used solely for dimensionality reduction to feed ML models. Given the goal of assigning labels to clustered data points, I decided to utilize the t-distributed Stochastic Neighbor Embedding (t-SNE) technique. Although this method is primarily designed for visualizing high-dimensional

data in 2D and 3D, I chose it because it effectively captures non-linear relationships and preserves local structures in the data [2], which can help in identifying meaningful clusters for label assignment. As shown in Figure 3, the separation of data instances by their corresponding classes was more distinct with t-SNE than with PCA or kernel PCA.

Since the goal is to generate accurate labels for the original images that closely approximate the true labels, I used the Adjusted Rand Index (ARI) and the Silhouette Score (SC) to evaluate the potential best dimensionality reduction and clustering technique. The ARI measures the similarity between the predicted labels and the ground truth labels, correcting for chance, while the SC evaluates the cohesion and separation of clusters in the dataset based on the provided labels. Both metrics are constrained to the range -1 to 1, where higher values indicate better clustering quality.

To evaluate clustering quality, I used a composite criterion: the sum of the ARI and the SC with clustered labels. This approach balanced two perspectives: the similarity between the clustered labels and the original ones, and the alignment between the new labels and the clustered data, without relying too much on the true labels to simulate a scenario where access to real labels is limited. At this stage, the ARI and SC metrics were calculated directly using the cluster labels, without applying a one-to-one mapping between classes and labels.

Table 1 shows that, based on the defined metric, the dimensionally reduced dataset using Gaussian kernel PCA with $\gamma = 0.1$, clustered via Agglomerative Clustering, produced the best labels, achieving a clustering quality of 0.994. This result is consistent with the observations in Figure 2. However, this method generated highly imbalanced labels compared to a uniform distribution, as previously noted, whereas the original labels were uniformly distributed among the classes.

Clustering Method	**Dimensionality Reduction**	**Clustering Quality**	**Imbalance Ratio**
Agglomerative Clustering	Gaussian_PCA_gamma-0.1	0.994	0.945
KMeans	Gaussian_PCA_gamma-0.1	0.993	0.946
KMeans	tsne_3D	0.865	0.045
Agglomerative Clustering	tsne_3D	0.770	0.090
Agglomerative Clustering	Gaussian_PCA_gamma-0.001	0.699	0.068
KMeans	Gaussian_PCA_gamma-0.001	0.659	0.063
KMeans	Linear PCA	0.653	0.122
KMeans	Polynomial_PCA_degree-2	0.645	0.263
Agglomerative Clustering	Linear PCA	0.591	0.201
Agglomerative Clustering	Polynomial_PCA_degree-2	0.576	0.277
KMeans	Gaussian_PCA_gamma-0.01	0.569	0.627
Agglomerative Clustering	Gaussian_PCA_gamma-0.01	0.513	0.561

Table 1: Clustering results for various dimensionality reduction techniques. PCA and kernel PCA datasets had 10 features and t-SNE 3.

Since the primary goal of the challenge was to use PCA or kernel PCA to generate labels, I selected the results obtained from Gaussian kernel PCA with $\gamma = 0.001$ clustered using the Agglomerative Clustering method. Figure 4 highlights the class imbalance in the Gaussian kernel PCA dataset with $\gamma = 0.1$, whereas the datasets generated using Gaussian kernel PCA with $\gamma = 0.001$ and t-SNE (3D) show a more balanced label distribution, as summarized in Table 1.

To determine which dimensionality reduction technique best captured the data geometry for clustering and label generation, I examined the elbow point on the cumulative explained variance ratio of PCA and kernel PCA techniques and its difference from the value at 10 components, which I used for clustering. However, this approach proved uninformative, prompting me to explore the alternative procedure described earlier. For comparison, Linear PCA achieved a cumulative explained variance ratio of 0.62 with 10 components, while Gaussian kernel PCA with $\gamma = 0.001$ reached 0.41. Despite this difference, their clustering quality was approximately the same. Overall, the use of kernel PCA for this task provided only marginal benefits, yielding diminishing returns.

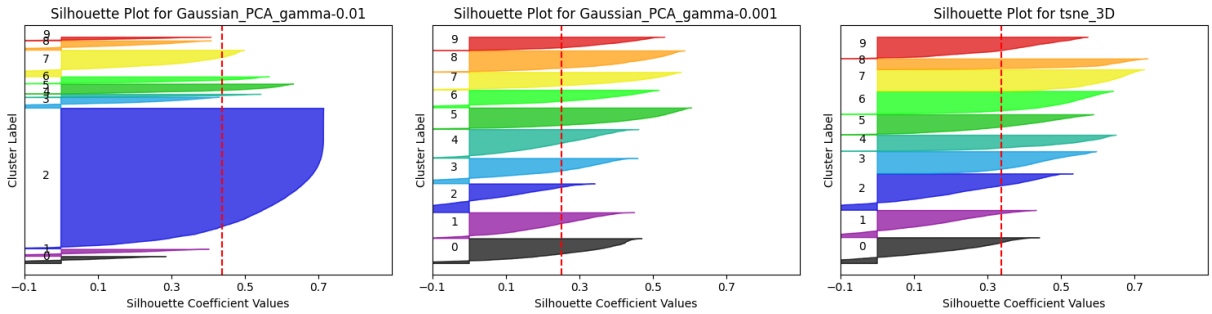

Figure 4: Silhouette Plot of the Agglomerative Clustering method for several dimensionally reduced datasets with 10 components.

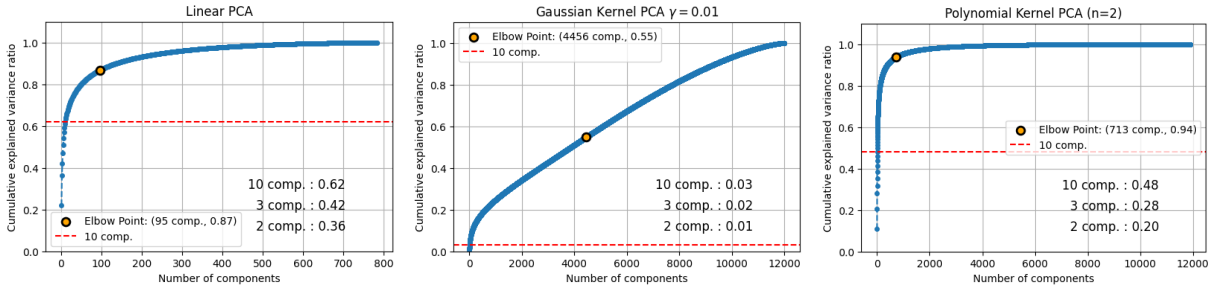

Figure 5: Cummulative explained variance ratio for some PCA and kernel PCA configurations.

Figure 6 illustrates the assignment of clustered labels to the original training dataset. It highlights the imperfections in clustering, where some labels are assigned to multiple classes. For instance, label 4 is assigned to both a trouser (at position (1,3)) and a T-shirt (at position (2,1)), among other cases. To address this, I applied a one-to-one mapping between the numerical labels and the dataset classes. This mapping was subsequently used to consistently represent the numerical labels in both the train and test datasets. Moreover, this new one-to-one mapping does not necessarily align with the true label-to-class mapping.

3 (Supervised) classification

In this section, the goal is to perform supervised classification of the images in the original dataset using: i) the clustered labels, and ii) the true labels. I employed three models: kernel SVM, a fully connected neural network (FCN), and a convolutional neural network (CNN). The evaluation was conducted on the test dataset using the true labels.

For the kernel SVM, I selected a Gaussian kernel with $\gamma = 10^{-3}$, as it demonstrated good performance for the task of the previous section. The FCN consisted of three simple linear layers with 64 and 32 hidden units. A dropout operation of 0.1 was performed before the last layer. The CNN architecture included two convolutional layers with the second one followed by a max-pooling operation. The output of the convolutional layers was flattened and passed through two fully connected layers. To mitigate overfitting, a dropout operation of 0.5 was added after the first fully connected layer. ReLU activations were used after all intermediate layers in both networks.

For both neural networks, I used the Adam optimizer, the cross-entropy loss function, a batch size of 32, a learning rate of 0.001, and trained the models for 50 epochs.

In addition to the selected clustered labels, I also used the labels generated from the t-SNE technique combined with KMeans clustering. This method was selected for comparison due to its good clustering quality and low imbalance ratio illustrated in the previous section.

Before passing the clustered labels to the supervised classification pipeline, I applied the one-to-one mapping obtained in the previous section. This mapping transformed the clustered labels to align with the true labels, ensuring an accurate evaluation of the trained models on the test set using the true labels.

Labels	**SVM**	**FCN**	**CNN**
Gaussian kernel $\gamma = 0.001$ (Agglomerative Clustering)	60.34	59.54	61.40
Real	88.19	85.29	90.85
t-SNE_3D (KMeans)	61.96	61.37	62.85

Table 2: Accuracy results for classification of the MNIST Fashion Dataset

Table 1 shows that the models trained on the original labels performed better than the ones trained on the clustered labels with about 30% difference in accuracy. Moreover, the t-SNE labels resulted in slightly better models than those trained on labels obtained from the Gaussian kernel dimensionality reduction. While CNNs outperformed FCNs and SVMs, Figure 7 indicates potential overfitting, as the test loss increases over training epochs. Notably, this effect is observed for FCNs and CNNs trained on clustered labels. However, since the training loss and test accuracy remain relatively constant at the presented scale, this evidence alone is insufficient to confirm overfitting. Overall, ANNs achieved good performance, but the SVM with a Gaussian kernel also performed reasonably well for this task, suggesting that Linear PCA could have performed just as well.

References

[1] scikit-learn developers, *Clustering — scikit-learn 1.5 documentation*, [Accessed: 2024-11-20], 2023. [Online]. Available: https://scikit-learn.org/1.5/modules/clustering.html#clustering.

[2] C. Olah, *Visualizing mnist: An exploration of dimensionality reduction*, [Accessed: 2024-11-20], 2014. [Online]. Available: https://colah.github.io/posts/2014-10-Visualizing-MNIST/.

A Clustered labels

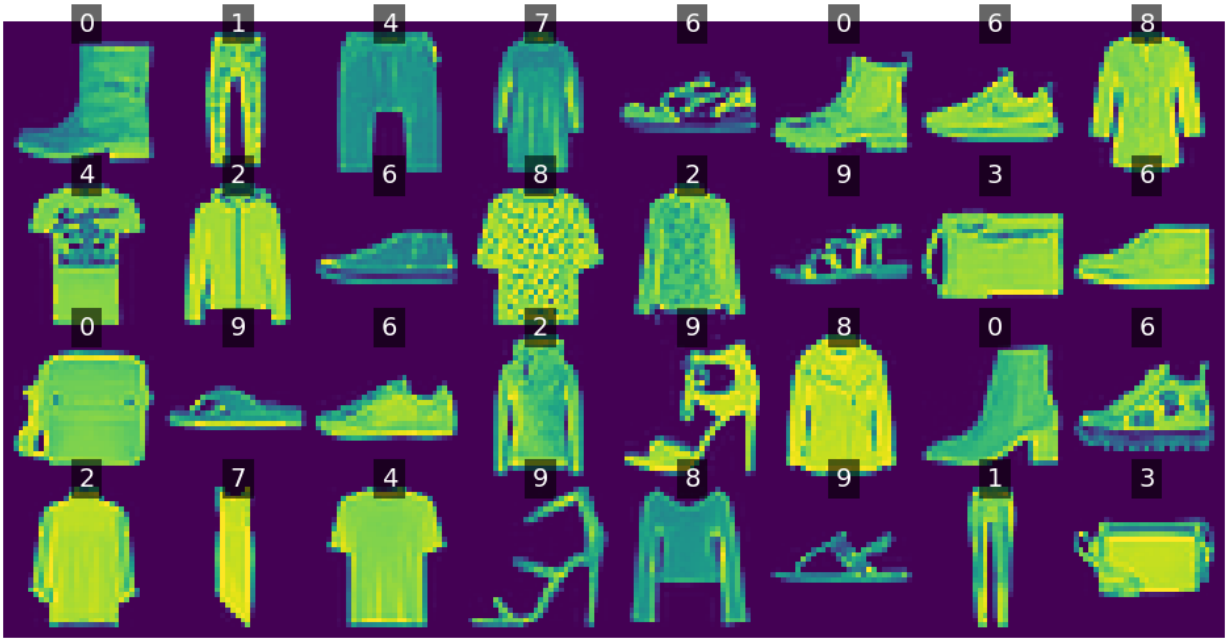

Figure 6: Example of clustered labels on the original training data set.

B Artificial Neural Networks Metrics

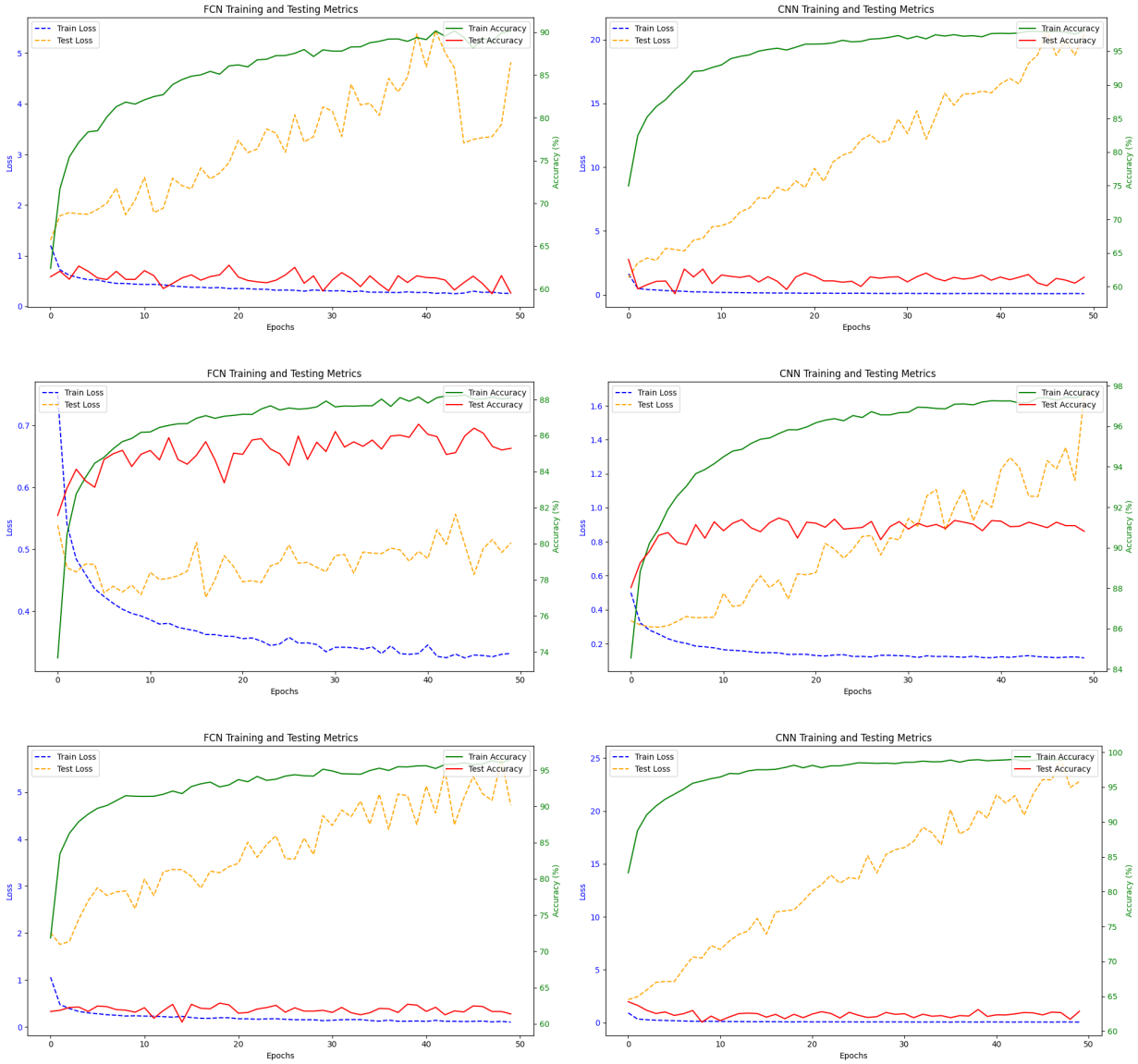

Figure 7: Loss and accuracy during training for the FCN (left) and CNN (right). From top to bottom, labels from: Gaussian PCA ($\gamma = 0.001$) and Agglomerative Clustering; original data; and t-SNE 3D with KMeans.