ECOLABEL: AN IMAGE CLASSIFIER FOR RECYCLED CLOTHING

MACHINE LEARNING REPORT
A.Y. 2023/2024

Giacomo Frigerio
Department of Physics
University of Pavia, Italy
giacomo.frigerio01@universitadipavia.it

June 19, 2024

ABSTRACT

Efficiently sorting discarded clothing is crucial for recycling warehouses. This report explores the development and implementation of image classification algorithms to automate the grouping of similar clothes. A dataset of 70,000 grayscale images, representing ten types of clothing, was divided into training and test sets. Various classification models, including Multinomial Logistic Regression, K-Nearest Neighbors (KNN), Multilayer Perceptrons (MLP), and Convolutional Neural Networks (CNN), were evaluated. Pre-processing techniques such as L1 and L2 normalization, PCA for dimensionality reduction, and edge direction histogram for feature extraction were applied. The best model, a CNN, achieved high accuracy, effectively identifying items and distinguishing between categories. Analysis of confusion matrices indicated that shirts were the hardest to classify. By grouping items into broader categories like shoes, accessories, and clothes, the model achieved near-perfect classification, demonstrating its potential to streamline sorting processes in recycling facilities. This automation could significantly reduce the need for human personnel, improving operational efficiency and supporting sustainable clothing management.

1 Introduction

Efficient sorting of discarded clothing is essential for recycling warehouses. Machine learning based techniques could be used to implement the efficiency of the system. This report investigates the use of image classification algorithms to automate the grouping of similar clothes. Our goal is to compare different classification algorithms, including Multinomial Logistic Regression, K-Nearest Neighbors (KNN), Multilayer Perceptrons (MLP), and Convolutional Neural Networks (CNN). We will use various normalization techniques and other methods, such as dimensionality reduction, for the evaluation. By systematically analyzing their performance, we aim to identify the most effective approach for automating the classification of clothing items in recycling warehouses.

2 Data

The dataset used in this report consists of 70,000 images captured with a low-resolution grayscale camera. These images encompass ten different types of clothing items, classified as follows:

0 T-shirt
1 Trouser
2 Pullover
3 Dress
4 Coat
5 Sandal

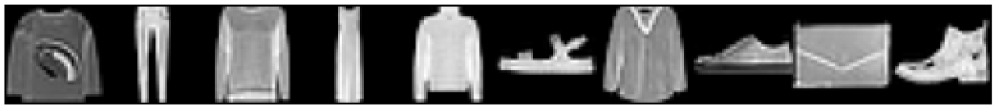


Figure 1: Image samples from all the different classes

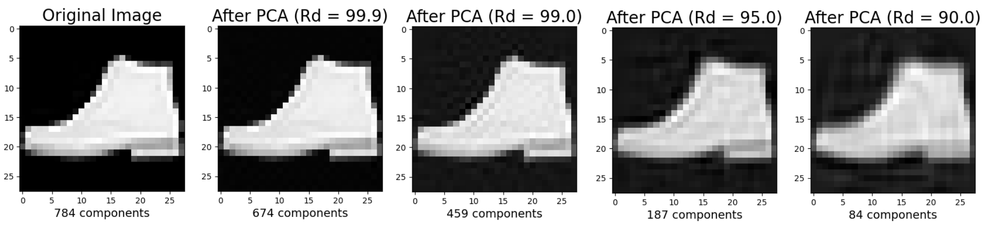


Figure 2: Visualization of one image of the dataset after dimensionality reduction with PCA varying the value of R_d defined in equation 1

6 Shirt

7 Sneaker

8 Bag

9 Ankle boot

The images have been resized to a resolution of 28x28 pixels and divided into training and test sets, with 60,000 images designated for training and 10,000 for testing.

2.1 Data: pre-processing

In preparation for the analysis, several pre-processing techniques were applied to the dataset to enhance the performance of the classification algorithms, such as normalization, dimensionality reduction and low level feature extraction.
Firstly, the dimension of the feature vector was reshaped as a $(60.000, 784)$ vector and normalized with different techniques. The one that gave the better results was L_2, which consists in dividing each component of the feature vector by the norm of the feature vector, i.e.

$$\bar{x}_j = \frac{x_j}{||\vec{x}||_2}$$

which also helped in the visualization of the weights of the MLP.
Feature selection was also addressed, with **low level features extraction** (such as Color Histogram, RGB co-occurrence matrix, Edge Direction) for images: low-level features are essential aspects of an image or video that may be retrieved such as colors, edges etc . The results, once low-level feature extraction was applied, were not particularly effective. In particular, features such as color histogram or RGB Co-occurrence matric yielded less effective results (the images, in fact, were gray scaled) while edge direction, which dwells more on the shape of the images, returned slightly better results.
For **dimensionality reduction**, Principal Component Analysis (**PCA**) was implemented. This method consists in looking at the d-dimensional linear projections retaining the most of the variability of the original set (maximizing the variance). This technique was applied for several numbers of principal components, changing the value of

$$R_d = \frac{\sum_{j=0}^{d-1} \lambda_j}{\sum_{j=0}^{n-1} \lambda_j} \tag{1}$$

where λ_j are the eigenvalues of the Covariance matrix estimated on the dataset, d is the number of the principal components used and n is the number of the initial components. The effect of PCA for an image can be visualized in figure 2.
The discussion about the effects of these techniques on the accuracy of the classifiers can be found in section 4.

3 Classification models and training

We employed and evaluated four different classification models: Multinomial Logistic Regression, Multilayer Perceptron, K-Nearest Neighbors, and Convolutional Neural Network. Each model mechanism is briefly described below.

1. **Multinomial Logistic Regression**: Multinomial Logistic Regression is a generalized form of logistic regression used for multi-class classification problems. Mathematically, it uses the softmax function to convert the linear combination of input features $\vec{x}$ and weights $\vec{W}$ into probabilities for the different k classes:

$$\hat{p}_c = \frac{e^{z_c}}{\sum_{j=0}^{k-1} e^{z_j}}\ , \quad c = 0, 1, \ldots, k-1.$$

   where

$$z_i = \sum_{j=0}^{n-1} W_{ij} x_j + b_i$$

2. **Multilayer Perceptron (MLP)**: a Multilayer Perceptron is a type of feedforward artificial neural network consisting of multiple layers of nodes, including an input layer, one or more hidden layers, and an output layer. Each node in a layer is connected to every node in the subsequent layer with associated weights. The activation of each element of the layer, called neuron, is performed by an activation function $(a(\cdot))$

$$h = a(\vec{w} \cdot \vec{x} + b)$$

   The implementation of MLP was made via the PVML library, in which the activation function for the output layer is the softmax operator, while hidden neurons use ReLU. The loss function used during training is the cross entropy.

3. **K-Nearest Neighbours (KNN)**: K-Nearest Neighbors is a learning algorithm used for classification. It classifies a data point based on the majority class among its k-nearest neighbors in the feature space, after memorizing all the training samples together with their class labels. The value of k was chosen via K-cross validation, using the "leave one out" strategy.

4. **Convolutional Neural Networks (CNN)**: convolutional neural networks are deep neural networks where linear operators (in MLP) are replaced with **convolutions**. A convolution is an operator that transforms an input signal $\vec{x}$ into another signal $\vec{z}$. This was implemented in two different ways: the first one using the PVML library and the other using KERAS.

4 Training and accuracies

Accuracies		
Model	Training Set	Test set
Multinomial Logistic regression	77.78 %	76.38 %
MLP (0 hidden layer)	87.05 %	84.53 %
MLP (1 hidden layer)	93.88 %	88.34 %
MLP (1 hidden layer) + PCA	95.07 %	88.96 %
MLP (2 hidden layer)	93.5 %	88.7 %
MLP (3 hidden layer)	94.65 %	89.08 %
KNN (20 %)	87.68 %	82.05 %
KNN (30 %)	89.16 %	83.23 %
KNN (40 %)	89.52 %	83.77 %
KNN (50 %)	89.76 %	84.32 %
KNN (60 %)	90.10 %	84.60 %
CNN (pvml)	90.26 %	88.36 %
CNN (keras)	94.83 %	93.00 %

Table 1: Table of accuracies for different models.

Performance measures Firstly, we wanted a measure to evaluate the efficiency of our classifier. Accuracy was the ideal one, which is defined as the fraction of cases for which the model gives the right answer. Other performance

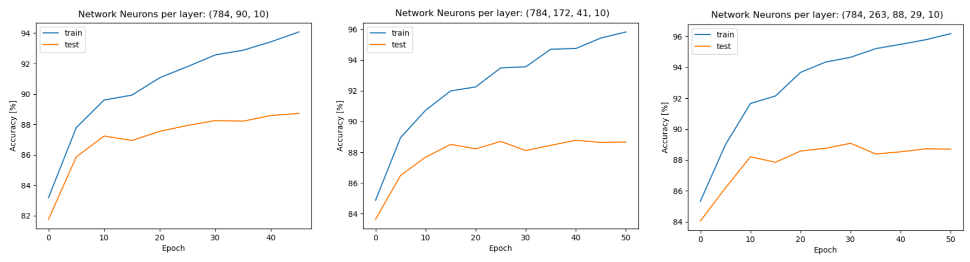


Figure 3: Accuracies for Multilayer perceptron with different hidden layers (in order, 1, 2 and 3).

measures could be taken into account, such as precision or recall, but in our case we wanted that each article was labeled in the correct way (each error had the same importance). Another measure was implemented for this kind of problem considering that some items belong to the same categories, or example, sandals and sneakers belong to footwear. Therefore, detecting a sandal as a sneaker is a minor impact error than classifying a T-shirt as a bag.

Results The results on the training and test set of the different models is pictured in table 1

In the next paragraph, a brief overview of the setup of the different models is provided.

4.1 Training setup and issues

- Multinomial logistic regression:
  While this performance demonstrates a reasonable baseline for multi-class classification in this context, it suggests that more complex models may be needed to achieve higher accuracy levels. Moreover, the computational time required to obtain this accuracy was really high compared to other models.
  Setup: (Learning rate = 0.001 , Steps = 10.000) .
- MLP:
  For the multilayer perceptron, all the models were trained over 50 epochs (higher epochs were evaluated but they gave almost the same results requiring a much higher computational time). The number of the hidden neurons per layer was found with the following formula

  $$N_{l-1} : N_l = N_l : N_{l+1}$$

  where N_l is the number of neurons in the l layer, N_{l-1} in the previous one and N_{l+1} in the subsequent one. Having 784 neurons in the input vector and 10 neurons in the output, the following values were obtained:
  - 1 Hidden Layer : (784, 88, 10)
  - 2 Hidden layers : (784, 181, 42 , 10)
  - 3 Hidden Layers : (784, 263, 88, 29, 10)

  Moreover, the learning rate was equal to 0.0001 and the steps were equal to 6000. The batch size was fixed to 10, after using a validation set (20% of the training set) to determine the best value. For a batch size equal to 5, the accuracy was slightly better but required way more computational time, so the other value was deemed to be the best.
  Training curves are displayed in figure 3.
  Analyzing the training curve for the 1 hidden layer MLP with a big number of epochs (figure 4), it is clear that the model overfits the data. Further implementations could introduce regularization or by enhancing the gradient descent (including, for example, a momentum term). As introduced in section 2.1, the PCA technique was applied. This resulted in an accuracy very similar to that without PCA but with less computation time. Training curves with PCA (for 1 hidden layer) are displayed in figure 5.

- CNN:
  Two different convolutional neural networks were introduced. The former, implemented with the PVML library, consisted in a 7×7 convolution followed by three 3×3 convolutions, with a stride of 2 for the first two and a stride of 1 for the others. Padding was not implemented. The best result, which is shown in the table, is comparable to that of MLPs although with a larger number of epochs (which would have required much more computational time) the accuracy would probably have increased further. Setup: (Learning rate: 0.0001, batch size = 100, steps = 6000).

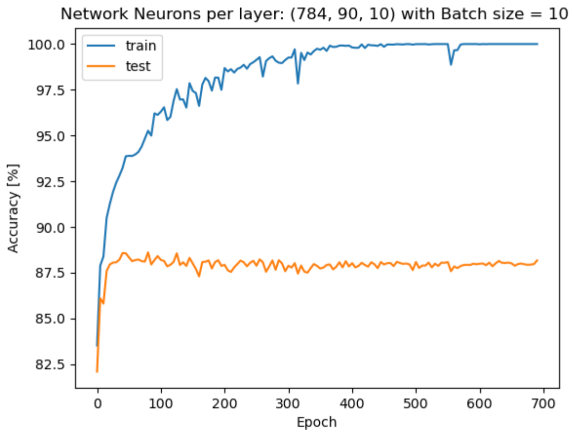


Figure 4: The training curve for the MLP with 1 hidden layer.

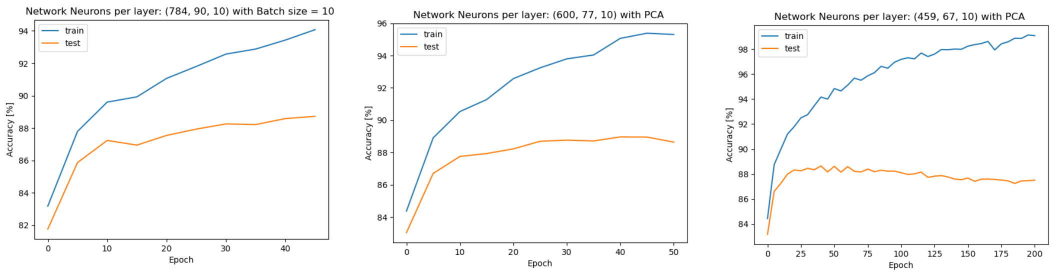


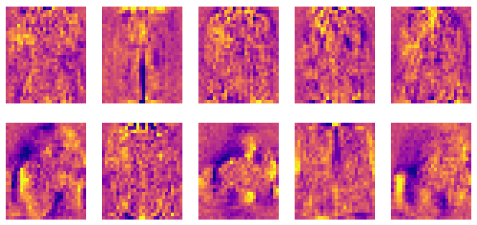
Figure 5: Training curves for different PCA reduction.

Another approach was made using Keras Library, with a different (and more complex) architecture. More advanced techniques, such as max pooling, dense and droput, have been implemented. In particular, the library is more performant than the PVML library, so computation times were much smaller. The architecture is summarized in appendix A.

- KNN:
The main issue related to KNN was that, due to computational power, it was not able to process the whole training set. For this reason, only a part of the training set was considered and the accuracy depending on the percentage of the training set used is displayed in table 1. Overall, the accuracy was not comparable to that of MLP or CNN but the computational time is really low (practically instantaneous!). Further implementations could check its value using the whole training set. The best value of k (the number of nearest neighbours taken into account) was implemented via the "leave one out" strategy. The "leave one out" strategy is a specific type of cross-validation used to evaluate the performance of a machine learning model. The model is trained on all data points except one, and then tested on the single data point that was left out. This process is repeated for each data point in the dataset. The best value obtained was $k = 4$.

5 Best Model Analysis

To evaluate our best model, we first visualized the weights for the Multilayer Perceptron (MLP) without normalization, and with L2 normalization (figure 6). The visualizations revealed that the model identified the shapes of the items, with L2 normalization providing the clearest representation.
In deeper networks, however, the patterns became harder to interpret due to the extraction of high-level features (figure 7).
Examining the confusion matrix (as shown in figure 8a) for CNN network (the one with the highest accuracy), we found that the most challenging category to classify was clothes, with "shirt" being the hardest item to distinguish. The worst predictions, i.e. the wrong predictions made with higher confidence, made by the model


(a) Visualization of Weights for the MLP with no hidden layers.

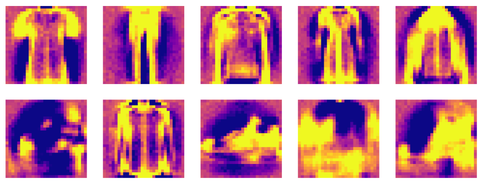

(b) Visualization of weights for the MLP with no hidden layers and L_2 normalization.

Figure 6: Weights for MLP with no hidden layers.

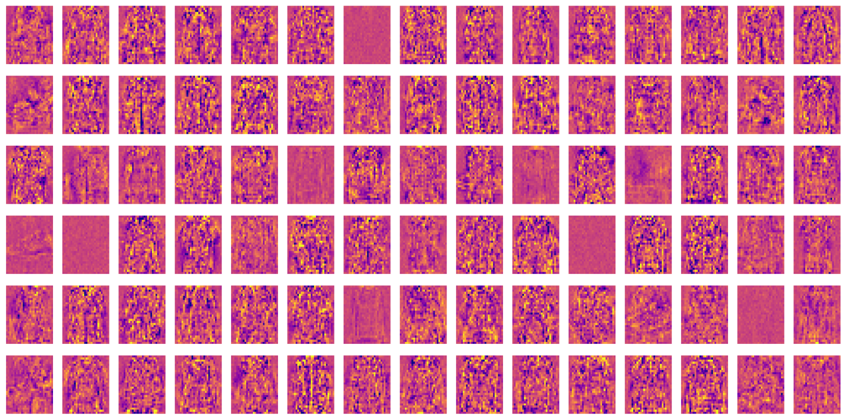


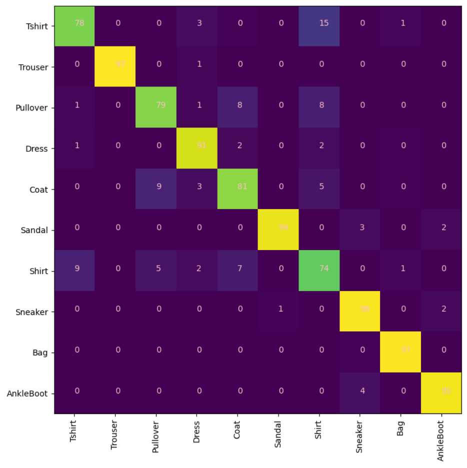
Figure 7: Weights for the MLP with 1 hidden layer.

(as shown in figure 8b) were often items that were difficult to identify even for humans (is the 5th item in figure really a bag? The author would have said it was a horse). As discussed in the second section, we introduced a measure for classification accuracy by grouping "Sneakers, Sandals, and Ankle Boots" as Shoes, "Bag" as an accessory, and "Trousers, Dress, T-shirt, Shirt, Coat, and Pullover" as clothes. Using this measure, nearly 100% of the items were correctly classified. This suggests that machine learning could effectively pre-divide items, reducing the need for human work and allowing workers to focus on categories where the model is less accurate, saving money and training more specialised personnel in particular categories.

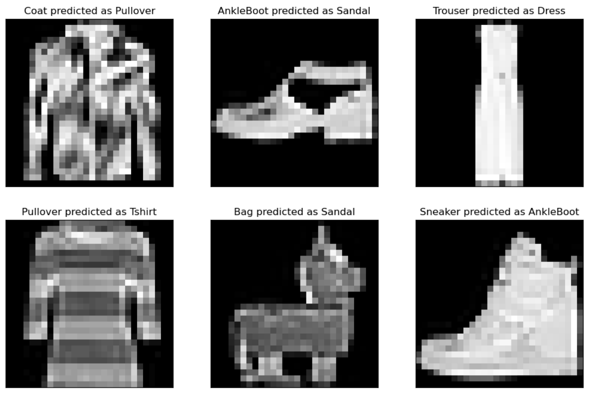
6 Conclusion

In conclusion, this report demonstrates the effectiveness of using image classification algorithms to automate the sorting of discarded clothing in recycling warehouses. The Convolutional Neural Network (CNN) emerged as the best-performing model, accurately identifying and classifying different types of clothing with high precision. Pre-processing techniques such as PCA dimensionality reduction enhanced the model's performance (principally for computing time). Despite challenges in distinguishing certain items, particularly shirts, the overall classification accuracy was significantly improved by grouping items into broader categories. For further implementation, several steps can be taken to enhance the system's efficiency and accuracy:

- Testing the KNN with all the training set.
- Implementing different CNN architectures.
- Advanced Feature Extraction: Exploring more sophisticated feature extraction methods, could provide additional insights and improve classification performance especially for clothes category.
- Transfer Learning: Utilizing pre-trained models on large-scale datasets (e.g., ImageNet) and fine-tuning them on the specific clothing dataset could leverage existing knowledge and enhance the model's capabilities.
- Scalability: Testing the model on a larger dataset and in different warehouse environments will ensure its robustness and scalability.



(a) Confusion Matrix for the CNN model.


Coat predicted as Pullover
AnkleBoot predicted as Sandal
Trouser predicted as Dress
Pullover predicted as Tshirt
Bag predicted as Sandal
Sneaker predicted as AnkleBoot

(b) Worst predictions made by the model.

By implementing these enhancements, the automated classification system can become an invaluable tool for recycling warehouses, promoting sustainable practices and significantly reducing the need for specialized manual sorting. This advancement will contribute to more efficient recycling processes and support the broader goal of sustainable fashion management.

Statement : I affirm that this report is the result of my own work and that I did not share any part of it with anyone else except the teacher.

A CNN architecture with KERAS

Keras Library provides numerous implementations of commonly used neural-network building blocks. The following CNN architecture was implemented:

```
model = tf.keras.Sequential()

model.add(tf.keras.layers.Conv2D(filters=64, kernel_size=2,
          padding='same', activation='relu', input_shape=(28,28,1)))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.3))
model.add(tf.keras.layers.Conv2D(filters=32, kernel_size=2, padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.3))
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(256, activation='relu'))
model.add(tf.keras.layers.Dropout(0.5))
model.add(tf.keras.layers.Dense(10, activation='softmax'))
```

which can be summarized as:

- 64 2D-convolutional layers (kernel size = 2) and ReLU as the activation function.
- A MaxPooling layer (2D), which summarizes as a single value a group of adjacent activations, with size 2.
- A dropout, which helps overfitting, setting to zero 30% (in this case) of input units.
- 32 1D-convolutional layers (kernel size = 2) and ReLU as the activation function.
- Another MaxPooling of size 2.
- Another dropout, equal to 30% of input units.
- A flattening layer, summing the $(7x7x32)$ input units in a single shape vector.
- A dense layer, performing classification with a ReLU activation function, returning 256 output neurons.
- Another dropout, equal to 50% of input units.
- Another dense layer, returning the 10 different classes, using SoftMax as the activation function.