CAB420 Machine Learning

Assignment 1B

Semester 1, 2020

Catherine Xu, 10684069

Amalie Omholt Ellestad, 10678271

Problem 1. Person Re-Identification

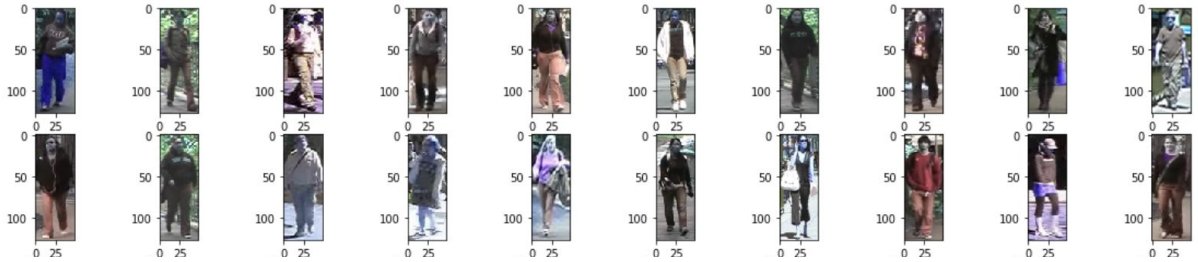
In this problem we are given images taken with two different cameras; cam_a and cam_b. The training set consists of 482 identities, with one photo from camera a and one from camera b. The testing set consists of 150 identities, also visible from both cameras with one photo from each. With this we are to identify identities by performing person re-identification and match a person with a person previously seen. To do this we proceed with two different techniques; PCA/LDA and deep learning with siamese networks.

PCA/LDA

Initial Data Preparation

We are provided with four sets of data; training and testing data from camera a and training and testing data from camera b. These are imported independently such that we have images in one array, image names in another and finally class labels in a third for each of the four sets. The arrays are labeled according to the information and the camera. All the images are imported as RGB and have the dimension 128x48x3, and this is set we use for the PCA. The next step before applying the PCA is to convert all the data sets into vectors, as this is the input PCA requires. Lastly, we concatenate the training set from camera_a and camera_b together such that we only have one training set. In a similar manner we obtain one testing set.

To get an idea of what kind of images we are working with, below are the 20 first photos from camera_a training set.



PCA/LDA

We use *PCA()* from *sklearn's decomposition* to transform our training data. Then we study the variance ratio of the transformed data. PCA transforms our data such that the first principal component contains most of the variance, the second contains the second most variance and so on.
Below are the variance related to the first 12 components. It is clear that the first indeed explains most of the variance, however it's only 16.3 percent, which is little.

```
[1.63107684e-01 9.70410592e-02 7.13471347e-02 3.63820346e-02
 2.71306114e-02 2.62265988e-02 2.25144435e-02 1.83113945e-02
 1.56887531e-02 1.39360936e-02 1.31858780e-02 1.21798338e-02
```

By generating the cumulative sum with the criteria of >0.95, we obtain that 95% of the variance is explained by 99 components. We also tried different thresholds of the variance, however none yielded a perfect training performance and even worse test performance.

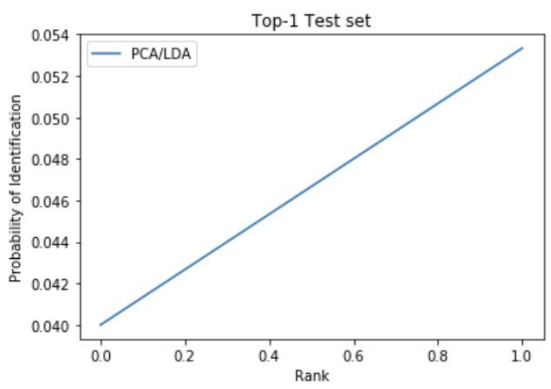
As PCA in itself only is a transformation of data and a way to reduce dimensions by only keeping the useful ones, the data that we obtain is condensed in power. As we essentially wish to classify images, we further perform LDA on the reduced space because it maximizes distances between classes and minimized distances in classes. With this we are able to collect similar data points closely together and the different classes far away in order to differentiate better. However, we don't have many samples per class, which can affect the accuracy.

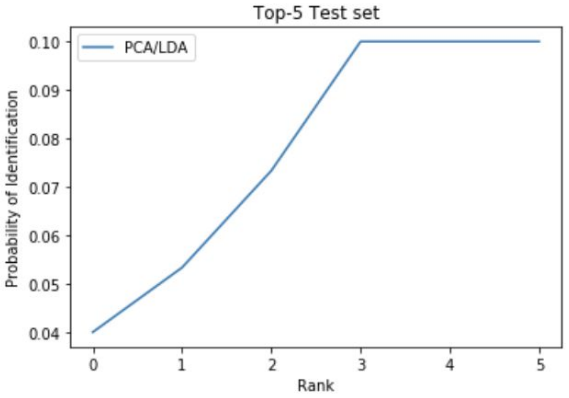
To fit the LDA model we first create a model using *LinearDiscriminantAnalysis()* from sklearn. The inputs are the transformed training data from PCA with 99 components and the class labels. We then generate the transformed values from the LDA for our training set, that serves as our embedding values for the camera a and camera b training set. We simply split the joint array to reobtain the embedding arrays separately. The same is performed on the testing set.

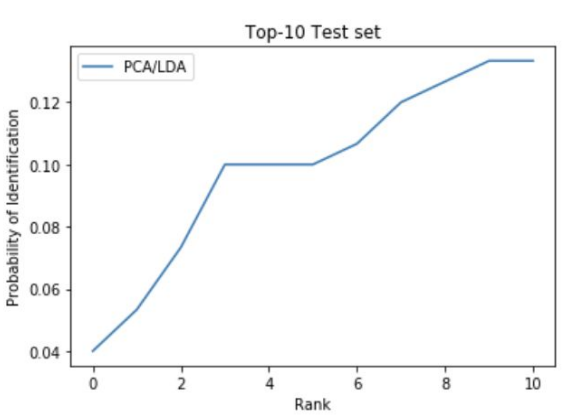
Results

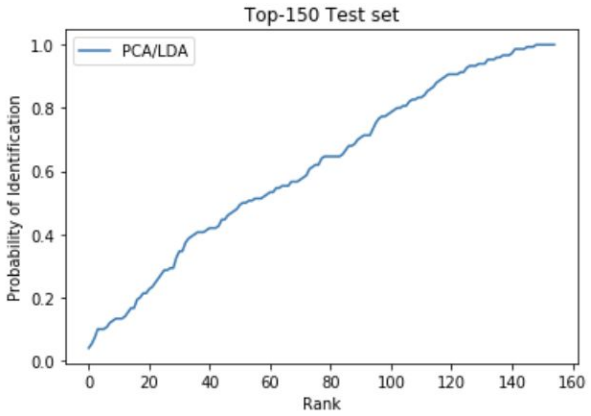
The Cumulative Matching Characteristics is a popular evaluation metric for person re-identification. The similarity between images is calculated based on the euclidean distance, where the smallest distance indicates the most similar picture. When running this we need to decide a number N, which is the number of top N ranked entries we consider. Evaluating the training set yields that all our 482 samples were correctly ranked as the most similar, which is perfect, although insignificant for the test set.

Let's study the figures below of the CMC evaluated on the test set. If we have top-N with rank 1 (top-1), CMC tells us the probability that the top 1 matches the identity we are searching for. The output of CMC tells us that probability is 4%. Looking at top-10, it tells us the probability that at least one of the top ten matches the identity we search for is 13%.









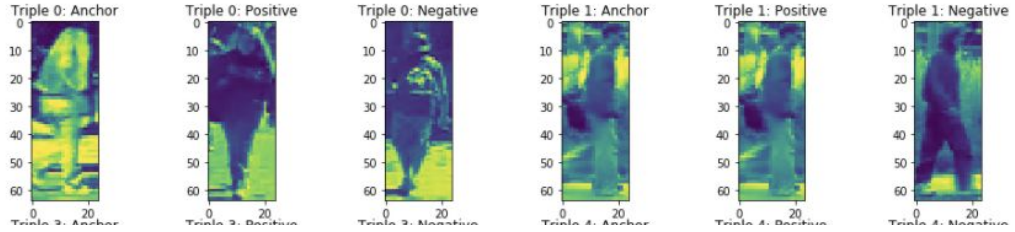
Deep Learning - Siamese Network

We will replicate the PCA/LDA using deep networks. Siamese networks are good for comparing one element to another. Once again we think of it as a classification task and use a distance approach. The basic idea is having two streams that extract features. We have in our solution decided to train a triplet network, as the embeddings will get better.

Initial Preparation

We start off with using the image array with imported image data from the previous data preparation step. As we are training a neural network, we resize the images such that training time is shorter and normalize the data such that large pixels won't disturb. Then we collect the training data from both cameras in one array and the same for testing data. These are reshaped into vectors. We did not normalize as we are dealing with images, and they are defined in the same range. In addition, we actually observed worse performance when normalizing the training and testing set. However we normalize the distance measure.

Further, we define a few functions that will help us train our siamese network. First we need a function that generates triplets with a label indicating if they are the same or not. Then we extract out triplets and label the anchor, which is the identity we are trying to find, a positive sample and a negative sample. Below is an example of the two first triplets.



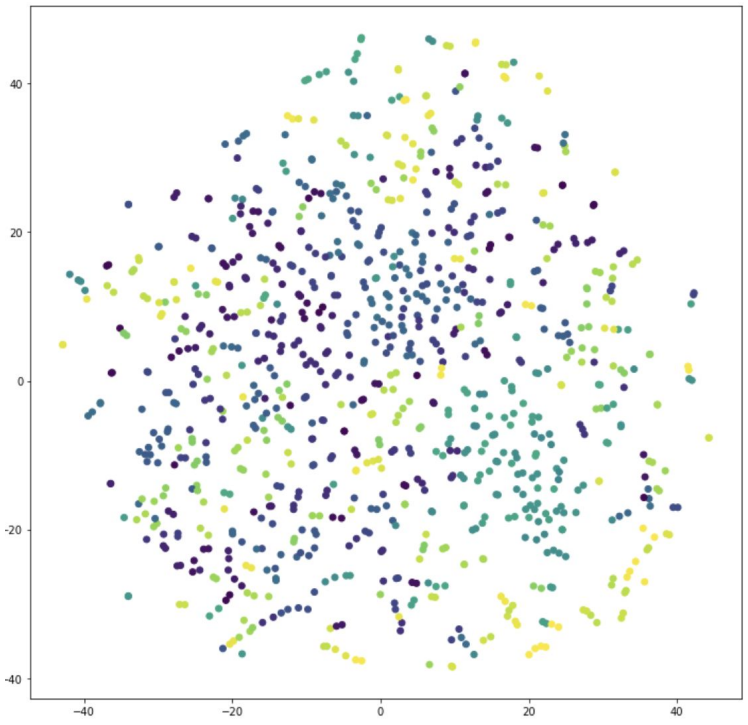
Our loss layer is calculated using the euclidean distance, with the goal of minimizing the distance between the anchor and the positive sample while maximizing distance between the anchor and the negative sample.

Siamese Network - Triplet Network

The goal with our network is that the branches extract the same features, hence the output embeddings will tell us if two images are the same or not. The output embedding of two identical images should be the exact same. For our siamese branch we use a VGG network. We use an input of the size (64, 28, 1), three filters and a fully connected layer. A 20% spatial dropout is added to prevent overfitting. We also add an embedding layer with a size of 16, with the goal of being big enough to get enough information but not catch noise. From our network we generate embeddings for the anchor, the positive and the negative sample. Then we use this base network, add a loss layer for the triplet loss, and train it on our training set with the image data and the class label. Our triple network is trained with a batch size of 64, 50 epochs and steps per epochs 600//50.

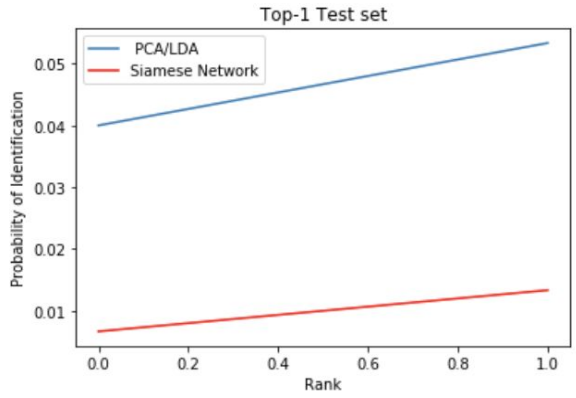
The trained network outputs embeddings, which we plot in a scatter diagram to see how the classification is. As we can see, it separates one class better than the remainings, but the separation in general isn't good.

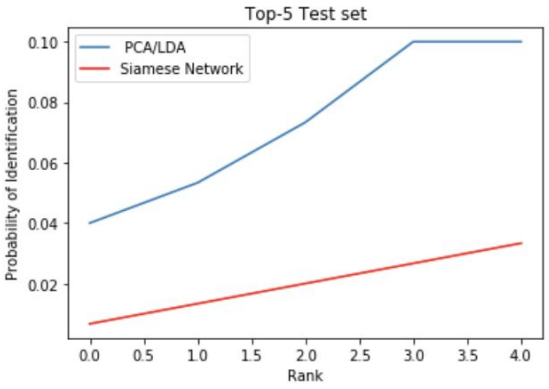
Scatter diagram of embeddings

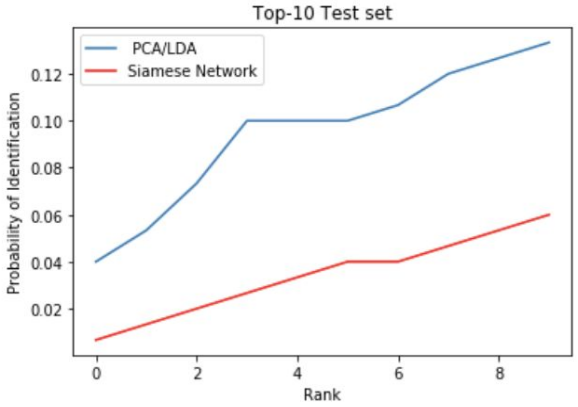


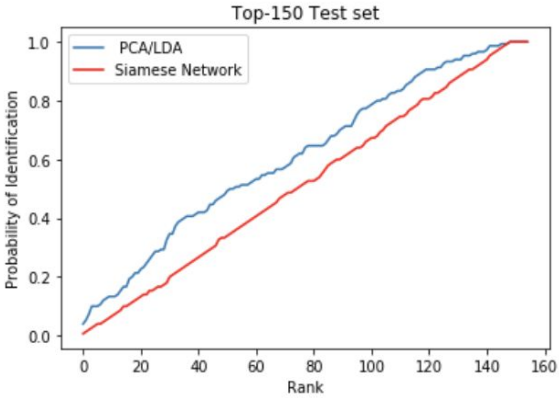
Results and comparison

We use the same evaluation metrics as with PCA/LDA in order to easily compare our results. Below we have plotted the PCA/LDA and Siamese Network CMC-evaluation on the test set considering Top-1, Top-5 and Top-10 performance. We have also added a graph that shows the performance for the entire test set. It is obvious that our Siamese network is consequently performing worse than our PCA/LDA, which is not the expected result.

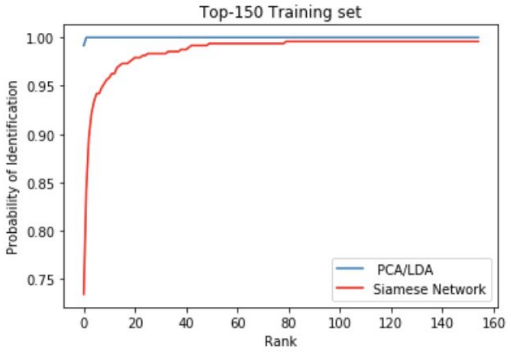








The architecture of siamese networks makes it good for limited datasets, which our definitely is. We therefore expect an alright performance. PCA/LDA however expects more samples than dimensions, and performs the best if each pixel in the images contains more or less the same details across the different images. This is not the case we have, so the performance is expected to be bad, and most definitely worse than our network. When our result were first observed, we checked our distance measures to be sure everything was calculated with Euclidean distance and the loss function looked alright. It could be that there is some minor computation mistake in the code when it comes to distance or our network is overfitting. We tried to alter the embedding size, batch size and steps, but these values seemed to give the best results regarding the scatter matrix and probability of prediction, although all our results were pretty bad.



We suspect that our siamese network might be overtraining, as it seems like the performance of the siamese network is considerably worse than the PCA/LDA on the test set and considerably worse than its training set performance. Looking at the CMC evaluation of the training set, there could be a problem with both the training speed and overfitting.

Problem 2. Clustering and Recommendations

For this problem, we were given a dataset containing movie review data for 600 subjects. The data was split over four different files, however we chose to include only two of them, namely ratings.csv and movies.csv. Each entry in ratings.csv consists of a user ID, a movie ID, a rating and a timestamp. The entries in movies.csv consist of a movie ID, the movie title and a list of genres.

Data Preparation

We chose to cluster the users based on what genres they prefer. After importing the two csv-files into pandas dataframes, we calculated the average ratings of each genre for every user. The IMAX-genre and movies without a genre was excluded, as these are not giving any information about the type of movie. The data was then saved in a new dataframe containing the user id and the average ratings for each genre. If the user had not seen any movies of a specific genre (i.e. average genre rating=NaN), we made the assumption that the user did not like that genre, and the average rating was set to 0. The data was then split into two subsets: one containing user 42, 314 and 444 and one containing all other users.

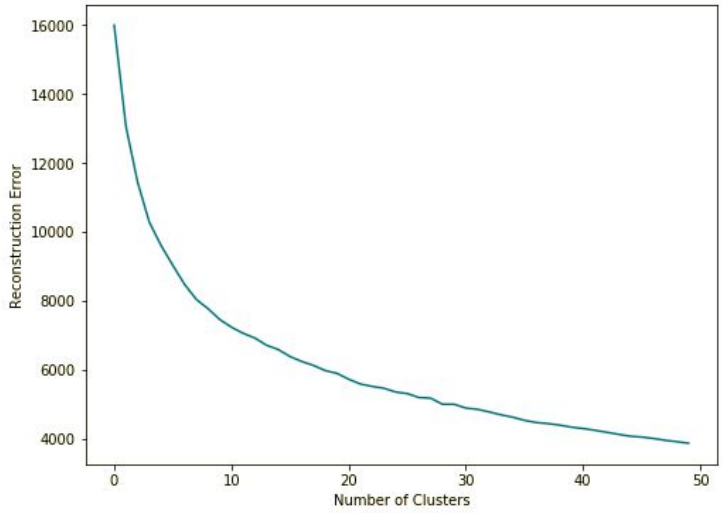
Clustering the Data

We chose to use K-means to cluster our data. This was mostly because our computers have limited computational power, and one of the advantages of K-means is that it has a fast running time. Consequently, this allowed for more time to improve our model by testing different parameters.

To determine the number of clusters to use, we first measured the reconstruction error for different values of k. The results are shown in figure 1. As we are measuring the distance between the cluster centres and their assigned points, the more clusters the better results. However, since increased number of clusters also leads to increased complexity of the model, a common heuristic is to choose the number of clusters based on the “elbow” in the graph. For our case, this occurs around 8 clusters. After this point, we see that the gain of each added cluster is becoming smaller.

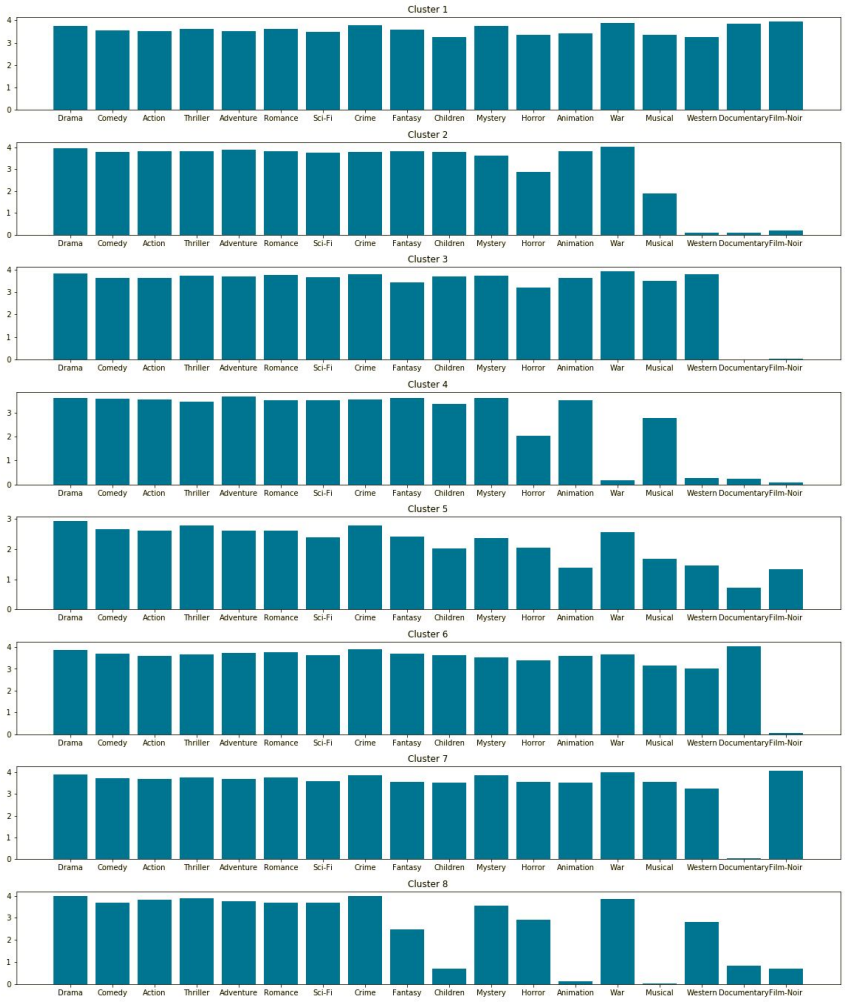
We also tried to approximate the number of clusters by looking at the bayesian information criterion (BIC). In contrast to the method above, BIC adds a penalty for model complexity, however in our case, BIC bottomed out at k=1.

Figure 1: Reconstruction error for different number of clusters



Taking the characteristics of our problem into account, we concluded that the k-value suggested by “the elbow method” might be a reasonable number. Setting k=8, resulted in the clustering shown in figure 2. Although we see some differences in genre-preferences, the eight first genres are similarly rated by all clusters.

Figure 2: Clustering results with k=8



To get a finer clustering, we decided to cluster the dataset into 18 groups, as 18 is the number of different genres in our dataset. Eight of the clusters are shown in figure 3. As we can see, we now have some more variance among our clusters. For example the users in cluster 12 rated action and thriller movies lower than most of the others. However, considering that we have more than doubled our number of clusters, we would expect a larger improvement. In addition, using 18 clusters leads to some of the clusters containing very few users. It is also worth noting that we are clustering the users based on genre ratings, not movie ratings, and genre preferences are probably less specific to each individual. Consequently, for this application, we decided to stick with 8 clusters.

Figure 3: Clustering results with k=18

Recommendations

To provide recommendations for the users with IDs 42, 314 and 444, we first predicted the cluster of each of the users. For each of those clusters, we made a list containing all the movies that more than 20 people from that cluster had rated. Next, we removed those movies with a rating lower than 4.2, and sorted the lists based on descending movie ratings.

Lastly, we removed those movies that the users had already seen from the list. This resulted in the following recommendations:

Figure 4: Movie Recommendations

```
Top five recommendations for user 42:
Lawrence of Arabia (1962)
Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb (1964)
Sunset Blvd. (a.k.a. Sunset Boulevard) (1950)
Graduate, The (1967)
Monty Python and the Holy Grail (1975)

Top five recommendations for user 314:
Star Wars: Episode VI - Return of the Jedi (1983)
Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark) (1981)
Star Wars: Episode V - The Empire Strikes Back (1980)
Back to the Future (1985)
Matrix, The (1999)

Top five recommendations for user 444:
Star Wars: Episode IV - A New Hope (1977)
Star Wars: Episode VI - Return of the Jedi (1983)
Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark) (1981)
Star Wars: Episode V - The Empire Strikes Back (1980)
Back to the Future (1985)
```

It is worth noting that user 314 and 444 ended up in the same cluster. Consequently, their recommendations are pretty similar.

As mentioned before, we are simplifying our clustering by only focusing on the average genre rating of each user. Consequently, we are also recommending movies based on what genres the user prefer, not the movie itself. Clustering the users based on the rating of each specific movie might lead to more precise movie recommendations, however the movies recommended by our model should be sufficient.

Problem 3. Semantic Person Search

For this problem, we were given a dataset consisting of images of people and a corresponding set of semantic annotations. Giving an input image, we were to implement one or several classifiers to classify gender, torso clothing type, torso clothing colour, torso clothing texture, leg clothing type, leg clothing texture, leg clothing colour and luggage.

Data Pre-Processing

To prepare our data, we first dropped the columns from train.csv and test.csv that were not among the traits to be classified, namely pose, secondary and tertiary torso clothing colour and secondary and tertiary leg clothing colour. We chose to regard he unknown-class (-1) as unknown data, and set it to NaN. Then we removed any rows containing NaNs and their corresponding images from the dataset. The shape of the dataset before and after the pre-processing are shown in figure 5.