Large scale machine learning : Fusion of algorithms for face recognition

Maël Fabien

Télécom ParisTech, 2018

Abstract. This data challenge focuses on comparing two images and deciding whether these two images represent the same person or not. First, a set of features is computed for each image. Then, a simple function produces a vector of scores for a pair of images. These information are grouped in a single data set. We then build a prediction algorithm able to classify images based on this set of extracted features. Light Gradient Boosting Model produces high a accuracy, and a fast training. Deep Learning models also show encouraging results.

1 The training set

The aim of this challenge is to provide a machine learning approach for face recognition based on pre-extracted face features. The performance criterion is the prediction accuracy on the test set, which is a value between 0 and 1.

The training set provided has 3'196'465 training observations. The features are the following :

- Columns 1-14 : 14 quality features based on image A.
- Columns 15-28 : 14 quality features based on image B.
- Columns 29-36 : 8 matching scores between images A and B.

Another data set is provided and contains the exact classification for each of the training example.

2 Exploratory data analysis

There are no large outliers, and no missing values. According to the correlation matrix, we notice that :

- the features extracted on each image are correlated one by one
- the common features S1 to S8 are highly correlated

We are also able to identify clusters on our data set from the cluster map that fit the previous interpretation of the correlation matrix heat map.

The principal component analysis lead on the data set reveals that the variance explained by the two first components reaches 41%. The scree plot of the eigenvalues illustrates the fact that the projection on a two dimensional plan should be efficient. The data set has been scaled before performing such analysis.

The circle on correlations highlights the directions of the different features along the two main axis. We observe that :

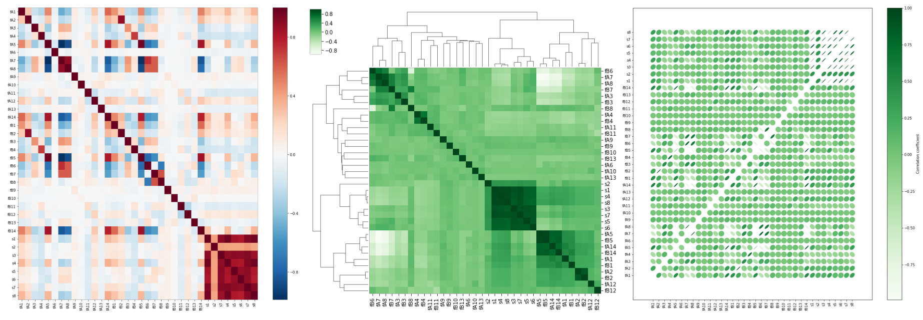


Fig. 1. Correlation matrix, correlation cluster and ellipse plot

- the features S1 to S8 are important features and all seem correlated
- the features extracted individually on each image are orthogonal on the PCA two-dimensional projection to the global direction of the features S1 to S8

The PCA scatter plot shows the two first PCA axis with a color type depending on the label. We can observe a clear separability of the data. We can overall expect high accuracies on such problem.

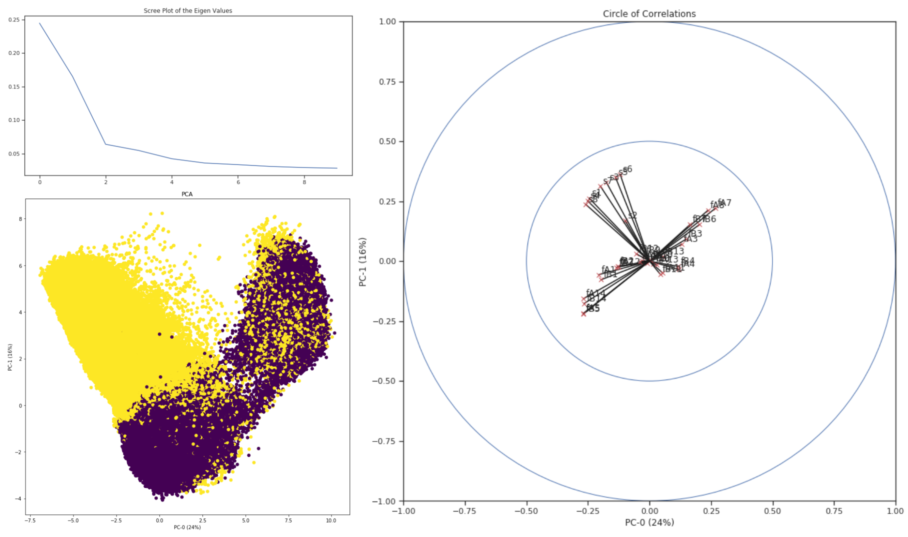


Fig. 2. PCA Analysis

We can also use t-distributed stochastic neighbor embedding (t-SNE), a dimension reduction technique based on probability distributions with random walk on neighborhood graphs that finds patterns in the data by identifying observed clusters based on similarity of data points with multiple features. The t-SNE is here applied on top of a PCA dimension reduction. The results confirm the separability of the datas.

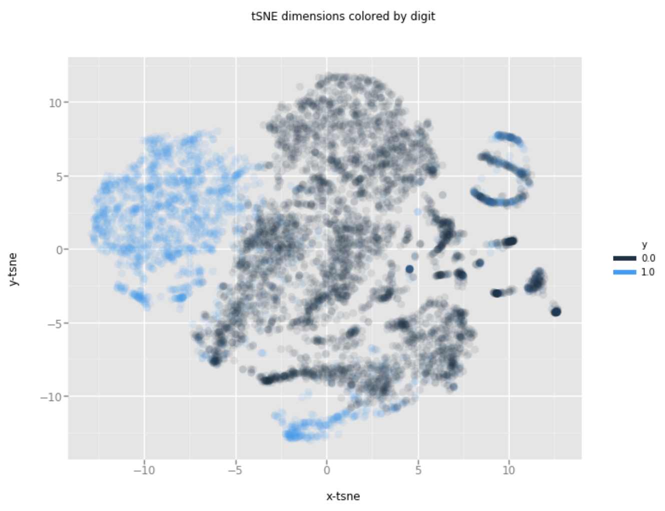


Fig. 3. t-distributed stochastic neighbor embedding

The feature importance has been evaluated by Random Forest Classifier without (left graph) and with (right graph) standardization. Overall, the standardization simply changes the order of importance of the variables S1 to S8, but does not bring any significant change otherwise. For the rest of the analysis, we will be working with non-standardized data.

3 Models

3.1 Approaches

Several approaches have been tested and can be summarized the following way :

1. Single Model : Fitting a single classifier on the whole set
2. Splitted Data : Fitting a first classifier on the individual image features, a second on the common features already built, a third on the whole set and creating a voting classifier
3. Embedded Model : Fitting a first model, extracting the data points for which the predicted probabilities were the weakest, and fitting a second classifier on those data points

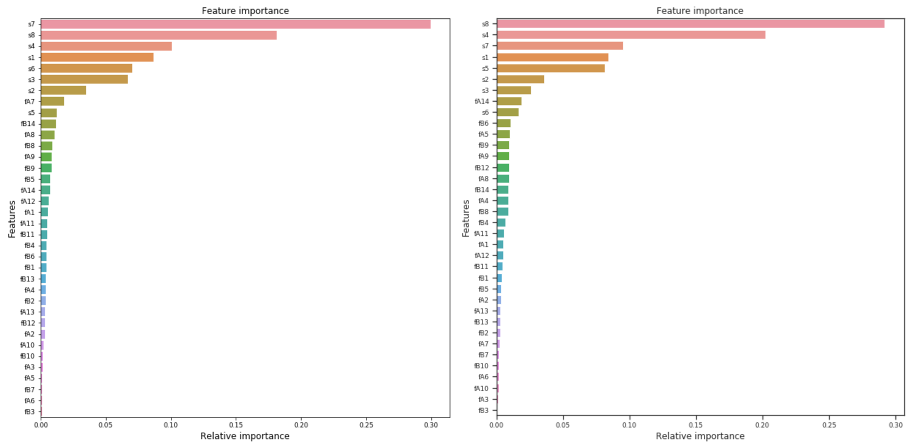


Fig. 4. Feature importance

4. Ensemble Model : Fitting 3 different models in an ensemble model, and adding the predicted probabilities to the test set in order to fit a last classifier
5. Deep Learning Model : Fit a neural network on each image and the common features, and a last neural network on the output of the first two networks.

The classifiers that have been considered in such models were :

- a logistic regression
- a support vector machine
- tree based classifiers (random forest, extra trees..)
- gradient boosting methods (XGBoost, Light Gradient Boosting Model (LGBM))

The results of the different combinations tested are presented in Table 1.

Table 1. Classification accuracy per model

Best Classifier/Structure	Accuracy
Single Model : LGBM	99.39%
Splitted Data Model	No improvement
Embedded Model	99.45%
Ensemble Model	99.33 %
Deep Learning Model	98.10%

3.2 In depth

Let's go in depth in some of the best performing structures.

3.2.1 Embedded Model Figure 5 illustrates the architecture of the Embedded Model :

- a first classifier is fitted on the whole training set. LGBM was chosen for this step.
- by plotting the predicted probabilities by the first classifier, we are able to identify the thresholds at which we should consider that a given data point might be miss-classified.
- the accuracy on the data points for which the predicted probability was the highest or the lowest reached 99.71%, whereas the accuracy for the data points selected as the ”weakly” classified was 61%.
- those data points are extracted, and a second classifier is fitted. Random Forest reached an accuracy of more than 70%.
- a similar procedure is adopted for the prediction.

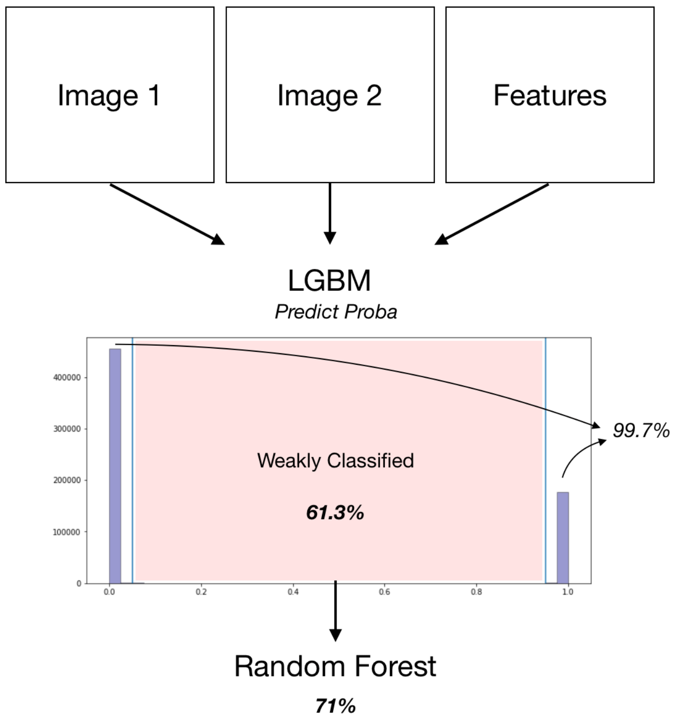


Fig. 5. Embedded Classifier structure

3.2.2 Ensemble Model The ensemble model tested combined :

- a LGMB
- a Cat-Boost Classifier
- a XGBoost Classifier

Such ensemble models come at a certain computational cost. Since we are training with a large amount of data, the computation time is typically tripled. The predicted probabilities were concatenated, and added to the existing test data. A new classifier, a Random forest, was then fitted to the extended data set. The performance remained high although few hyper parameter tuning was done due to the long computation time.

3.2.3 Deep Learning Model A deep learning model was also developed. The architecture included :

- a first network on the first image and the common features
- a second network on the second image and the common
- a third network on the output of the first two

The accuracy did not exceed 98% and would require a better pre-processing of the inputs.

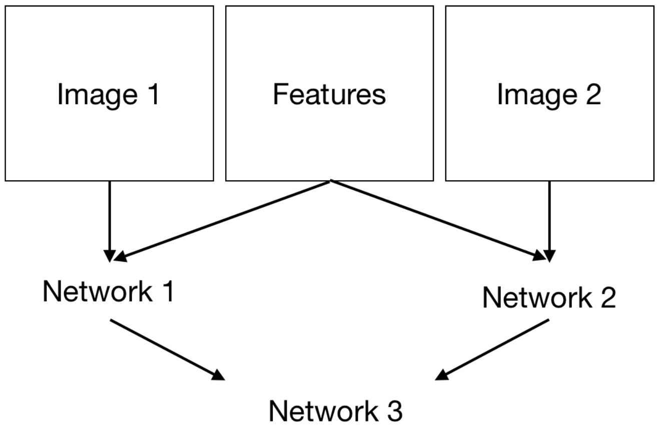
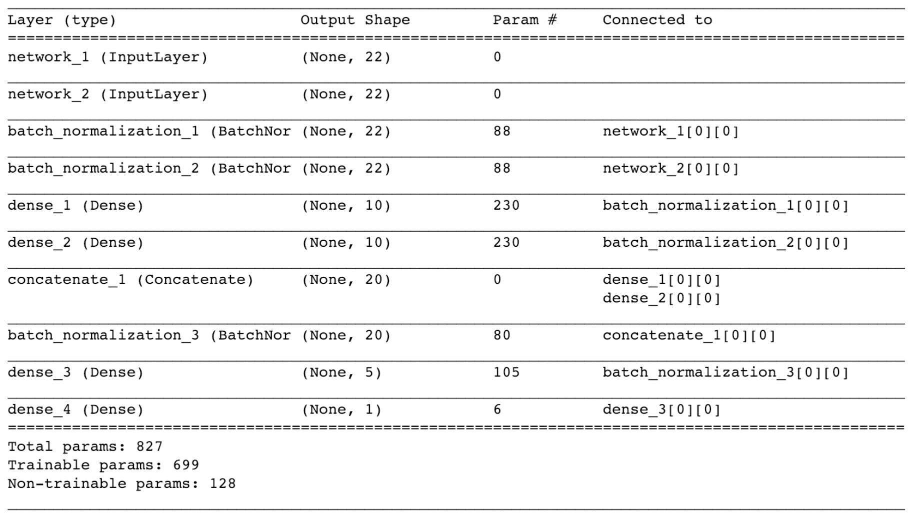


Fig. 6. Deep Learning structure

The summary of the model is presented in figure 7. There are overall only 699 parameters. An easy way to improve the model would be for example to use one hot encoding on the input data.



```
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
network_1 (InputLayer)          (None, 22)           0
__________________________________________________________________________________________________
network_2 (InputLayer)          (None, 22)           0
__________________________________________________________________________________________________
batch_normalization_1 (BatchNor (None, 22)           88          network_1[0][0]
__________________________________________________________________________________________________
batch_normalization_2 (BatchNor (None, 22)           88          network_2[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 10)           230         batch_normalization_1[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, 10)           230         batch_normalization_2[0][0]
__________________________________________________________________________________________________
concatenate_1 (Concatenate)     (None, 20)           0           dense_1[0][0]
                                                                 dense_2[0][0]
__________________________________________________________________________________________________
batch_normalization_3 (BatchNor (None, 20)           80          concatenate_1[0][0]
__________________________________________________________________________________________________
dense_3 (Dense)                 (None, 5)            105         batch_normalization_3[0][0]
__________________________________________________________________________________________________
dense_4 (Dense)                 (None, 1)            6           dense_3[0][0]
==================================================================================================
Total params: 827
Trainable params: 699
Non-trainable params: 128
__________________________________________________________________________________________________
```

Fig. 7. Model summary

4 Conclusion

We have shown that in the context of image similarity, the embedded classifier structure improved the overall accuracy. Deep learning models show encouraging with simple architectures results but are harder to tune.