# Face-recognition models comparison

Machine learning versus Deep Learning.

Giovanni Federico Poli, mat.766577

*University of Bari "Aldo Moro", Computer Science course, Machine Learning report*

November, 2022

**Abstract**

This work is proposed as a comparison between machine learning and deep learning models for the face recognition task. Here are implemented four ML models (K-NN, Random Forest, SVM, and Logistic Regression) and also a DNN model in particular a Convolutional Neural Network in order to assess which one performs better for the task. Haar cascade has been used for facial detection.

## 1. Introduction

Face recognition is an easy task for humans. Experiments have shown, that even one to three days old babies are able to distinguish between known faces. So, how hard could it be for a computer? Well, it's not that easy.
Face recognition could be used for a wide variety of tasks, nowadays companies like Amazon grasped the potential of face recognition to improve the customer experience. Just think about Amazon Prime Video X-Ray technology, which, in real-time, is able to show on screen some information about the actors playing in the scene currently displayed. With this work, we will try to understand the ratio behind those complex technologies, analyzing the behavior of some Machine Learning and Deep Learning models, focusing our attention on how and under what circumstances one approach is better than another.
The Machine Learning models here implemented are:

- Random Forest;
- K-Nearest Neighbors;
- Support Vector Machine;
- Logistic Regression;

The only Deep Learning model here discussed is:

- Convolutional Neural Network;

## 2. Related Works

The preliminary part of this work was dedicated to the exploration of various scientific papers' methodologies, such as "Face Recognition: A literature survey"[1] which states that even though current machine recognition systems have reached a certain level of maturity, their success is limited by the conditions imposed by many real applications, or "Face Recognition Based on Convolutional Neural Network"[2] that, at the 2017 International Conference on Modern Electrical and Energy Systems (MEES), proposed a modified Convolutional Neural Network (CNN) architecture by adding two normalization operations to two of the layers. These kinds of papers are the starting point for the idea of the comparison between ML and DL models that characterize this work .

## 3. Haarcascade

The first true step to facial recognition is facial detection. The face detection is performed with the Haar Cascade classifier. The starting dataset (which included more than 100 celebrities and 10000 images) has been cleaned because it included some "dirty data", and for simplicity reasons, it has been narrowed down to about 1000 images picturing the faces of 12 celebrities Haar Cascade is a machine learning-based approach where a lot of positive and negative images are used to train the classifier.

Hence initially, the algorithm train the classifier. Then it has to extract features from it and for this, Haar features shown in the below image are used. They are just like convolutional kernels. Each feature is a single value obtained by subtracting the sum of pixels under the white rectangle from the sum of pixels under the black rectangle. Now, each kernel's possible sizes and locations are used to calculate many features. But among all these features we calculated, most of them are irrelevant.
The top row shows two good features:

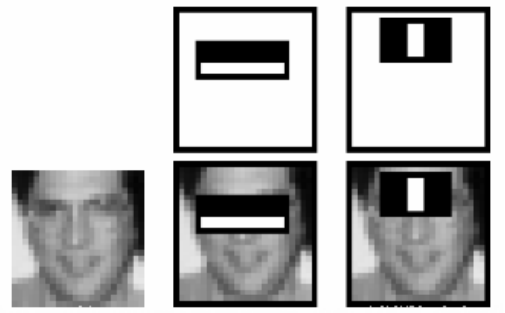


**Figure 1:** Haar features

- The first feature selected seems to focus on the property that the region of the eyes is often darker than the region of the nose and cheeks;
- The second feature selected relies on the property that the eyes are darker than the bridge of the nose;

But the same windows applied to cheeks or any other place are irrelevant. So how do we select the best features out of thousands of features? This is achieved by Adaboost.

### *3.1. Adaboost*

Each and every feature are applied to all the training images. Then the features with minimum error rate are selected, which means they are the features that most accurately classify the face and non-face images. (The process is not as simple as this. Each image is given equal weight in the beginning. After each classification, the weights of misclassified images are increased. Then the same process is done. New error rates are calculated as also new weights. The process is continued until the required accuracy or error rate is achieved or the required number of features are found).

The final classifier is a weighted sum of these weak classifiers. They're called weak because they alone can't classify the image but together form a strong classifier, able to detect a face.

## 4. Methodology

### *4.1. Set-up*

The data manipulation process is one of the most important tasks in order to fully exploit any model, and different models need different set-ups. For this purpose the project goes through every folder and every file in the dataset root, filtering the images: It discards every picture in which does not find any face and overrides the images in which there is at least one face with the ROI of the face cropping out the background, in order to feed the algorithms with cleaner data. Created this dataset, it is split into 75-25 for training and testing, but the methodology applied on the machine learning models will differ for the convolutional neural network.

### *4.2. Machine Learning set-up*

The Machine Learning workflow starts with relevant features being extracted manually. For this reason here are extracted the so-called HOG Features. The Histogram of Oriented Gradients, is a simple and powerful feature descriptor. The basic idea of HOG is to divide the image into small connected cells and quantize the edge direction of each cell starting by generating a histogram for each one of them by using gradients value[3]. In such manner, our features will be robust against illumination variance, but the only disadvantage with HOG-based face detection is that it doesn't work well on faces at odd angles. To solve this problem it has been implemented a data preprocess pipeline, in which every image is not only analyzed for detecting faces but, once a face is detected and the associated ROI generated, it detects also the eyes. In this way, we can rotate the whole image in order to align the eyes according to a horizontal line.


**Figure 2:** Training samples

#### *4.2.1. PCA*

Now we have a clean dataset, but since we realigned some of the images, those can present a certain amount of black edges. Because of this, it is also important to select the right features to identify a face, thus it has been implemented also the Principal Component Analysis. PCA is an unsupervised machine learning algorithm, so it ignores all the labels and treats the dataset as a whole[4].

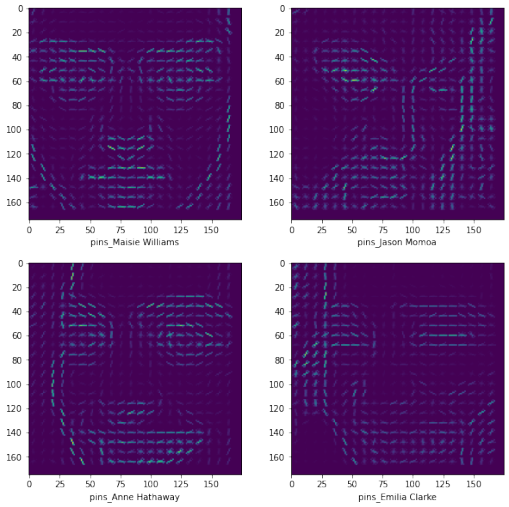



**Figure 3:** HOG features of training samples

The idea is that high-dimensional datasets are often described by correlated variables and therefore only a few meaningful dimensions account for most of the information. The PCA method finds a way to project the samples into a lower subspace, in which only the most meaningful features are retained. Our data is now ready to be fed to the machine-learning models.

### *4.3. Deep Learning set-up*

Deep Learning is a specialized version of Machine Learning. In Deep Learning workflows relevant features are automatically extracted from the plain data, which is the only thing that the DL models receive as input. If it is true that Deep Learning algorithms scale with the data, it is also true that in order to work as intended such algorithms need a certain amount of data and a thousand images divided into 12 classes are not enough for a convolutional neural network.

#### *4.3.1. Data Augmentation*

Thereby, a different data manipulation process has been implemented, including data augmentation. Data augmentation techniques, such as rotation, translation, zoom, flips, etc., address the limited training data issue by enriching the training set with transformed original samples. For images with small intra-class variability (such as changing the viewpoint, flipping, or cropping) in face images, the traditional data augmentation techniques improve the model performance[5], in fact, without them, our DL model did not achieve any better performance than a random classifier.
Our data augmentation pipeline starts normalizing the images, then we'll generate new samples by rotating every image in a range of 8° and changing their brightness from 0.8 to 1.2 in order to help the model extract important features.
Please note that this kind of data augmentation won't work at all with the data manipulation performed for the ML models, on the contrary, it will be not only useless but also harmful: first for the computational cost, but more importantly, rotating the images at various degrees would imply losing the robustness of the HOG features.

## 5. Models

### *5.1. Machine Learning*

The Machine Learning models are been implemented with the GridSearchCV technique which is used in order to identify the optimal hyperparameters for each model. What it actually does is "brute-forcing" every possible combination of the given hyperparameters and returns the best set of them according to the model accuracy score, so it simply makes a complete search over a given subset of the hyperparameters space (grid)[6].
Moreover, this technique also implements the so-called cross-validation to evaluate the performance of the models, measuring how well a model generalizes itself to a test set. This gives us an estimation of the model's goodness. With this method, we have a pair of training and testing sets. We can partition a single dataset to yield the two sets. These partitions are of the same size and are referred to as folds. A model in consideration is trained on all folds, except one, that is used to test the model. This process is repeated until all folds are used as the test set and the average performance of the model on all folds is then used to estimate the actual model's performance.

#### *5.1.1. K-Nearest Neighbors*

The KNN is one of the basic ML models. It is a non-parametric (number of parameters grows with the amount of data), supervised classifier that looks at k points in the training set that are nearest to the test input, counts how many members of each class are in this set, and assign a class label on basis of majority vote. It is based on the assumption that similar points can be found near one another, however, before classification can be made, the distance must be defined. Euclidean distance is most commonly used. In our case the model works with the distance of the 6 nearest neighbors, and the closer they are the more influence they have ['n neighbors':6, 'weights':'distance'].

| Model | Val accuracy | Macro avg f1 | Correct guesses out of 12 |
|---|---|---|---|
| KNN | 55.5% | 0.55 | 8 |

*5.1.2. Random Forest*

The Random Forest is an ensemble method. The goal of those methods is to combine the predictions of various base estimators (in this case we average the predictions of several Decision Trees) chosen randomly in order to decorrelate the base learners and improve generalizability and robustness over a single estimator. Our model uses 500 decision trees and Gini impurity (frequency at which any element of the dataset will be mislabelled when it is randomly labeled).
['criterion':'gini', 'max features':'sqrt', 'n estimators':500]

| Model | Val accuracy | Macro avg f1 score | Correct guesses out of 12 |
|---|---|---|---|
| Random Forest | 52.8% | 0.51 | 6 |

*5.1.3. Support Vector Machine*

The SVM adopts a supervised learning paradigm and can be considered a classifier that constructs a decision boundary (according to training examples) in the form of an hyperplane that is able to separate the data. Hence it is fundamentally a binary classification model, hence we need a way to rephrase our problem in order to fit its needs. The technique used here is the SVC default decision function for multiclass problems, i.e. one-vs-rest. So the classifier grabs a class and creates a binary label for whether a point is or isn't in that specific class. With a not-so-strong regularization (l2 penalty) and the RBF kernel which computes the similarity or how close two points are to each other (using also the euclidean distance) the model gets these results:
['C':1, 'gamma':'scale', 'kernel':'linear']

| Model | Val accuracy | Macro avg f1 | Correct guesses out of 12 |
|---|---|---|---|
| SVM | 69.8% | 0.70 | 10 |

*5.1.4. Logistic Regression*

Logistic regression belongs to the family of supervised machine learning models. It is also considered a discriminative model, which means that it attempts to distinguish between classes, it is also used to estimate the relationship between a dependent variable and one or more independent variables, but the same problem with the SVM for multiclass-classification is also found here. Binary logistic regression models can be fitted using either the Logistic Regression procedure (one vs rest) or the Multinomial Logistic Regression procedure. In our case, we will exploit the Multinomial procedure which is an extension of logistic regression that adds native support for multi-class classification problems. This kind of model is to be preferred when there is a fixed set of classes, and they are mutually exclusive, it is similar to logistic regression, but it is more general because the dependent variable is not restricted to two categories.
['penalty': 'l2', 'solver': 'saga']

| Model | Val accuracy | Macro avg f1 | Correct guesses out of 12 |
|---|---|---|---|
| Logistic Regression | 70.2% | 0.70 | 10 |

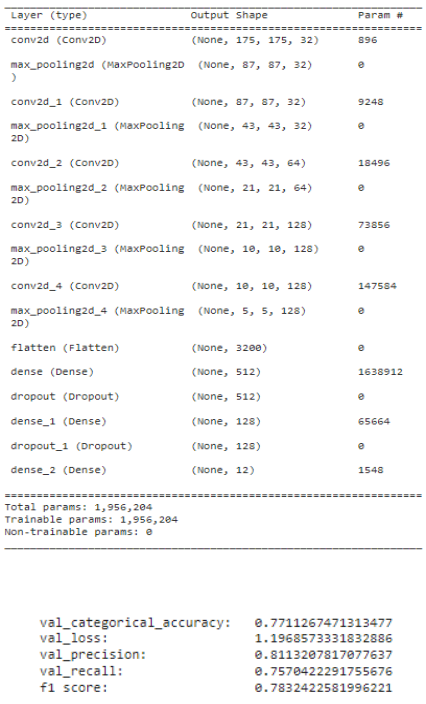
*5.2. Deep Learning*

The Deep Learning approach, as already said, is totally different because it does not need to receive the features directly, but it is able to extract them by itself. Here we perform some data augmentation based on brightness and rotation range, in order to let our DL model train correctly. Our model is a Convolutional Neural Network, which is a specialized Neural Network for processing grid-like data such as images. A NN consists of a hierarchy of layers each of which transforms the input data into a more abstract representation and the output layers combine those representations into making predictions. In the CNN neurons are locally connected, and it applies specific convolutional kernels (or filters) to a window (called receptive field) of the input image until high-level features are extracted.
We can identify four main operations:

1. Convolution:
   The main building block of a CNN that applies multiple trainable filters and produces an activation map(feature) for every filter. Each filter has its own weight and it's shared between every neuron related to the filter (# of filters = depth).
2. Non-linearity (ReLu):
   This is needed after every convolutional operation to learn a non-linear representation of data. The most used function is the ReLu (rectified Linear Unit). It is an element-wise operation that replaces all negative pixel values in the feature with 0.
3. Pooling:
   Pooling reduces the dimensionality of each feature map retaining most of the important information, resulting in a reduced resolution output feature map robust to small variations. So it makes the representation smaller and more manageable.

4. Classification:
   the classification of the input image into one of the classes on which the CNN is trained.

Before the classification, the input has been flattened to a 1D array and three dense layers have been implemented. Those layers use the features coming from the convolution and pooling for classifying the input image through a multi-layer perceptron that uses both ReLu and Soft max functions in the output layer. Between those three dense layers, we also use a dropout function that randomly sets the input to 0 at a frequency of 0.25 in order to prevent overfitting.



```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 175, 175, 32)      896

max_pooling2d (MaxPooling2D  (None, 87, 87, 32)        0
)

conv2d_1 (Conv2D)            (None, 87, 87, 32)        9248

max_pooling2d_1 (MaxPooling  (None, 43, 43, 32)        0
2D)

conv2d_2 (Conv2D)            (None, 43, 43, 64)        18496

max_pooling2d_2 (MaxPooling  (None, 21, 21, 64)        0
2D)

conv2d_3 (Conv2D)            (None, 21, 21, 128)       73856

max_pooling2d_3 (MaxPooling  (None, 10, 10, 128)       0
2D)

conv2d_4 (Conv2D)            (None, 10, 10, 128)       147584

max_pooling2d_4 (MaxPooling  (None, 5, 5, 128)         0
2D)

flatten (Flatten)            (None, 3200)              0

dense (Dense)                (None, 512)               1638912

dropout (Dropout)            (None, 512)               0

dense_1 (Dense)              (None, 128)               65664

dropout_1 (Dropout)          (None, 128)               0

dense_2 (Dense)              (None, 12)                1548

=================================================================
Total params: 1,956,204
Trainable params: 1,956,204
Non-trainable params: 0
_________________________________________________________________
```

```
val_categorical_accuracy:   0.7711267471313477
val_loss:                   1.1968573331832886
val_precision:              0.8113207817077637
val_recall:                 0.7570422291755676
f1 score:                   0.7832422581996221
```

**Figure 4:** CNN Model and mean scores

### *5.3. ML and DL results table*

| Model | Val accuracy | Macro avg f1 | Test accuracy |
|---|---|---|---|
| Random Forest | 52.8% | 0.51 | 50% |
| K-Nearest Neighbors | 55.5% | 0.55 | 66.6% |
| Support Vector Machine | 69.8% | 0.70 | 83.3% |
| Logistic Regression | 70.2% | 0.70 | 83.3% |
| Conv Neural Network | 77.1% | 0.78 | 66.6% |

## 6. Conclusions

First of all, we have to make some considerations: facial recognition is not a trivial task, and the dataset used in this work was no ordinary dataset, such as the AT&T database of faces where every person has their photos taken from predefined angles and with the same technical settings (brightness, height, distance, etc.). Our dataset is closer to a dataset of faces in the wild, which is good because it means that it will give our models some robustness against the variance in the data making them more generalizable.
Having said that, it is true that with this dataset we can attempt to generalize more, but it is also true that the models will experience more difficulty in the training phase, and that's due to the low number of samples and their large variance. For this reason, we try to normalize the data by cropping only the face, horizontally aligning it through the eyes, and using PCA in order to decrease the variance of the HOG features. Now, let's think about the Random Forest and KNN classifiers: since their results, they can't really extract significant patterns in the features from so few samples for each class, because they consider each class individually and compare it to every other class.
It is a different matter for the SVM and Logistic Regression. They are both born as binary classifiers which means that they have to rephrase the problem to a binary form like the SVM that performs the one class vs the rest or like the Logistic Regression which applies the Multinomial Logistic regression in order to extend its binary properties to a multi-class problem, and, in practice, this translates into a big increment of their performance compared to the RF and KNN models, this also gives us the indication that the HOG features are a good estimator in the facial recognition task.
Regarding the Convolutional Neural Network the state of the art states that those kinds of NN are the best tool for this problem and the result on the validation set validates this statement, but we have to keep in mind that in order to make this result possible we had to use some data augmentation. In fact, this kind of model is more similar to the RF and KNN model since they all share the same vision of classes, meaning that they compare every class to every other class, hence they need more samples in order to be able to generalize their predictions.

## 7. Bibiliography

[1] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld. 2003. Face recognition: A literature survey. ACM Comput. Surv. 35, 4 (December 2003), 399–458. https://doi.org/10.1145/954339.954342

[2] M. Coşkun, A. Uçar, Ö. Yildirim and Y. Demir, "Face recognition based on convolutional neural network," 2017 International Conference on Modern Electrical and Energy Systems (MEES), 2017, pp. 376-379, doi: 10.1109/MEES.2017.8248937.

[3] Mohammed, M.G. and Melhum, A.I., 2020. Implementation of HOG feature extraction with tuned parameters for human face detection. International Journal of Machine Learning and Computing, 10(5), pp.654-661.

[4] Dadi, Harihara Mohan, P.G.. (2016). Improved Face Recognition Rate Using HOG Features and SVM Classifier.IOSR Journal of Electronics and Communication Engineering(IOSR-JECE). 11. 34-44. 10.9790/2834-1104013444.

[5] Dongmei Han, Qigang Liu, Weiguo Fan, A new image classification method using CNN transfer learning and web data augmentation, Expert Systems with Applications, Volume 95,2018, Pages 43-56, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2017.11.028.

[6] Liashchynskyi, Petro et al. "Grid Search, Random Search, Genetic Algorithm: A Big Comparison for NAS." (2019).