# Gender image categorization using NNs

**Aitor Diaz de Otazu**

## Abstract

In this document I report my proposal for the application of supervised classification methods to the gender recognition problem. The classifier that I have selected for the classification tasks is: Multilayer Perceptron[3]. I have implemented the classification process using the sklearn, keras and scipy library. I have learned the classifiers using the train data and computing the accuracy in the test data. In addition, I have implemeted a CNN to compare the results between MLP and CNN.

## 1 Description of the problem

The task we have to solve is the classification of human faces in two groups: "Males" and "Females". For this problem I have used "The IMDB-WIKI dataset".

The dataset includes images of human faces and their date of birth, name and gender. The goal of the project is to predict the gender of the person.

The database has the following characteristics:

- $16,384$ attributes.
- 2 classes: Male '1', Female '0'.
- $62,328$ instances.
- The data is almost perfectly balanced. The number of instances in each class are $10,000$.

## 2 Description of our approach

I organized the implementation of the project according to the tasks:

1. Preprocessing and clear the dataset.
2. Define and learn the MLP classifier using the training data.
3. Design the validation method to evaluate the accuracy of the proposed classification approaches.

### 2.1 Preprocessing

I used the scipy[4] library to read the struct of the dataset as data frame. I splitted the gender information and images paths in two arrays. In the load process, broken images were removed and useful images were resized to 128x128 pixels. The data was split into two different sets: train and test, for validation. We use the same train set to learn MLP Classifier and CNN, the same test set for evaluating their accuracy.

### 2.2 Used Neural Networks

I used MLP for the main goal and CNN for compare the results.

1. MLPClassifier parameters:
   - hidden_layer_sizes=(32,32,64,128)
   - random_state=1,
   - batch_size=2000
   - max_iter=200
   - shuffle=True
   - activation='logistic'

#### 2.2.1 MLPClassifier

In this case I defined a MLP with 4 hidden layer of: 32, 32, 64 and 128 hidden neurons. In addition, the input layer is composed by 16.384 neuron, one for each input feature, and the output layer is composed by 1 neuron.

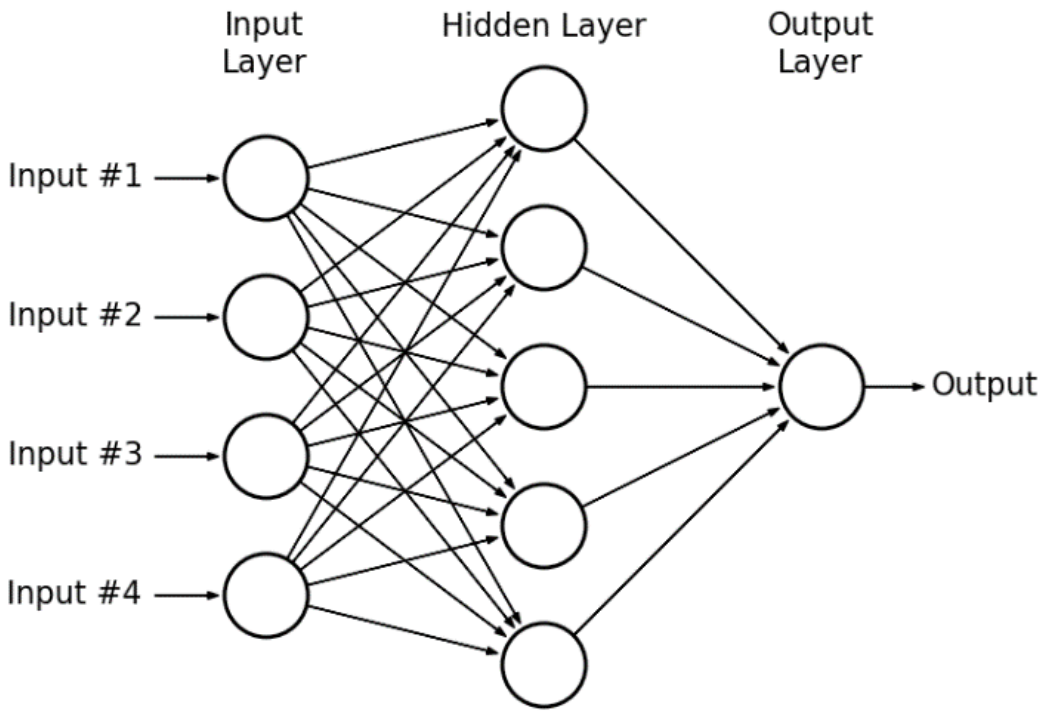



Figure 1: MLP architecture

MLP is a simply NN composed by dense layers (fully connected layers), so this MLP classifier has 535.932 parameters: (16384 + 1)$\cdot 32 + (32 + 1) \cdot 32 + (32 + 1) \cdot 64 + (64 + 1) \cdot 128 + (128 + 1) \cdot 1$

In this case I used sigmoid activation fuction because the task to resolve is the binary classification.

### 2.2.2 CNN

Convolutional neural network (CNN) is a class of deep neural networks, most commonly applied to analyzing visual imagery.

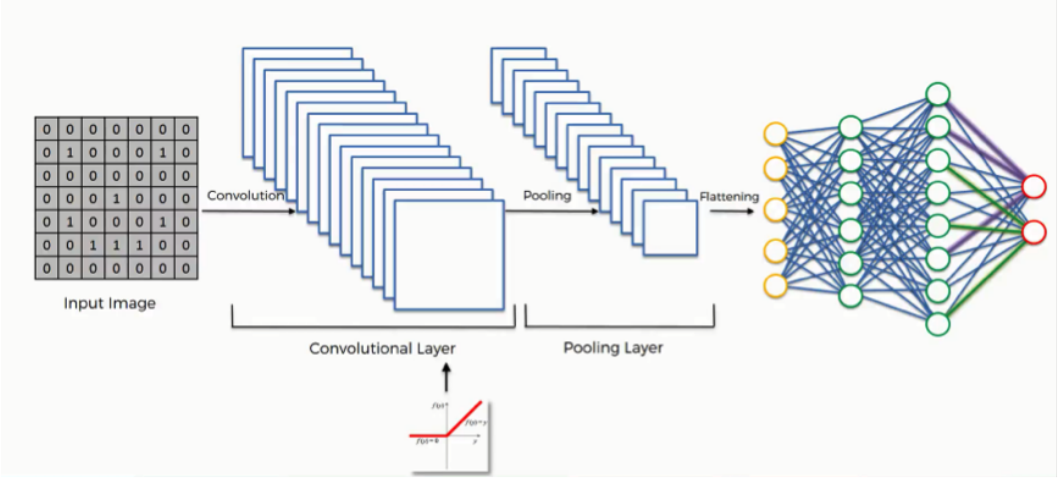



Figure 2: CNN architecture

The CNN was implemented using Keras library [5]. This CNN includes two convolutional layers using ReLU activaction function following by a pooling layer for each convolutional layer, a Dropout layer with 0.25 drop rate and Dense layer (like MLP) of 128 neurons with Sigmoid activation function.

This kind of NN improve the classification applying filters to capture patterns includes on the images, this increases the computing time but the results are good enough to use them.

### 2.3 Validation

To validate the results I compute the classifier accuracy in the test data. Another possibility was to compute the cross-validation in the complete dataset but I used the split between train and test because it was simpler.

As an additional validation step I computed the confusion matrices for the MLP and CNN.

| | |
|---|---|
| 1051 | 1049 |
| 1036 | 1064 |

Table 1: Confusion matrix produced by the MLP classifier

| | |
|---|---|
| 1597 | 503 |
| 407 | 1693 |

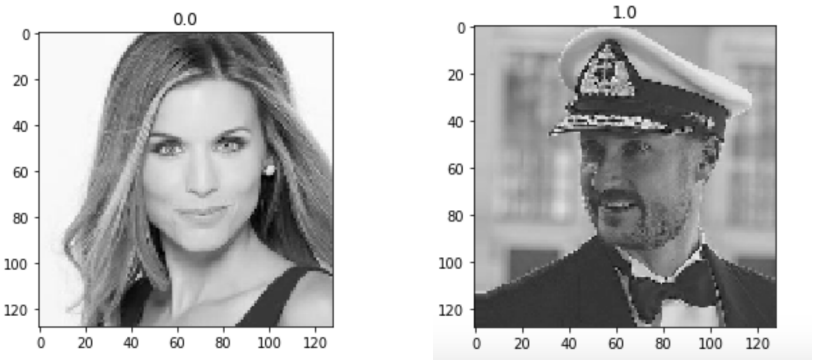
Table 2: Confusion matrix produced by the CNN classifier

## 3 Implementation

All the project steps were implemented in Python. I used scipy library to read the structucture of dataset, and keras and sklearn to implement MLP and CNN. I illustrate how the implementation works in the Python notebook

## 4 Results

The results with MLP are decent, I have obtained results around 60% accuracy. Use multilayer perceptron to classify it fulfills the goal but in the case of CNN we can see how we achieved a result of almost 80% training it with a small amount of data compared to what is commonly used in this kind of NN.




(a) Female prediction (b) Male prediction

Figure 3: MLP prediction example

## 5 Conclusions

In the project I used a Multilayer Perceptron implementation to resolve the problem of gender classification. This implementation give us an accucary over 60% but no enough if you need an accurate prediction. In addition, I have implemented a Convolutional Neural Network to compare the results and these are significantly better, around 80%. In conclusion, I recommend use CNN to resolve this type of task but I have to comment that MLP uses less time than CNN to train (MLP around 10 minutes comparing the 5 hours in CNN).

## References

[1] IMDB-Wiki dataset:
`https://data.vision.ee.ethz.ch/cvl/rrothe/imdb-wiki/`

[2] Numpy:
`https://numpy.org`

[3] Sicikt-learn library:
`https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html`

[4] Scipy.io:
`https://docs.scipy.org/doc/scipy/reference/tutorial/io.html`

[5] Keras library:
`https://keras.io/models/sequential/`

[6] Gender classification based on feedforward backpropagation neural network:
`https://link.springer.com/content/pdf/10.1007/978-0-387-74161-1_32.pdf`