INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON

OT2
Machine Learning Project
Deep Learning for Face Recognition

HAFID Salim
GADACHA Amine
CUZMAR Florencia

Table of Contents

1. Introduction & Project goals

As part of the OT2 course at INSA de Lyon, this Machine Learning project aims at using Deep Learning technologies in various ways. These technologies have been shown to have the best results regarding Computer Vision tasks such as Image and Video Classification, Object Detection, Image Segmentation and Face Recognition.
In this specific project, we'll be using Deep Learning to perform Face Recognition on a large image dataset.

The project has two sets of objectives:

- Implementing an end-to-end face recognition pipeline that uses convolutional neural networks, using the following steps:
    - Choosing and Preprocessing the datasets
    - Choosing the neural network architecture
    - Implementing the face detector
    - Visualizing and Evaluating results

- Building competences around the Machine Learning and Data Science problems, some of which are the following:
    - The difficulty of extracting semantic information from images
    - The functioning of a Convolutional Neural Network, how it's trained and applied
    - The influence of training data, of the network's architecture and of overfitting on the final results
    - The complexity/accuracy tradeoff
    - The limitations of learning methodologies in the context of a face recognition task

2. Creating a face/no-face classifier

a. Datasets

The datasets used to train and test the face/no-face classifier model were provided as part of the course, they consist of the following:

- 2 distinct image classes: face and no-face
- each image is a B/W 36x36 pixel image

The train set contains the following :

- a total of 91 720 images
- class distribution : 64 770 faces / 26 950 no-faces

The test set contains the following :

- a total of 7628 images
- class distribution : 6831 faces / 797 no-faces

b. Deep Learning model : CNN

To learn how to detect a human face, our face/no-face classifier will be a Convolutional Neural Network (CNN) model.

The CNN model specification is the following :

	Layers	Layer order and params
CNN Model	2 Convolutional layers 1 Pool layer 3 fully-connected layers	Conv2d(1,6,5) MaxPool2d(2,2) Conv2d(6,16,5) Linear(16*6*6,120) Linear(120,84) Linear(84,2)

c. Training the model

For training the model, we used a backpropagation algorithm taken from the course, with 10 epochs and a batch size of 32.

d. Results

For a first version, we trained the model for only 10 epochs. The results were promising, as the following table shows:

Loss Function	Optimizer	Number of training epochs	Loss convergence	Accuracy	
				train set	test set
Cross Entropy	SGD lr = 0.001 momentum = 0.9	10	First loss = 0.40202 Last loss = 0.01709	0.99	0.92

3. Implementing a face detector

In order to detect one or more faces on a picture of any size, we need to apply our face/no-face classifier to every (x,y) position of the input picture. This process is called an image sliding window.
But because our CNN model has an input size of 36x36 pixels it can only detect faces of that size. In order to detect faces of larger sizes we'll be using a pyramidical sliding window.
Put simply, this process means that instead of going through every (x,y) position of our image to see if our face/no-face classifier detects any faces one time, we'll repeat the process several times, while downscaling the input image by a N factor at every iteration. That way, we ensure that if the input picture contains a face, that face will at some point match our 36x36 pixel classifier input size.

a. Datasets

The dataset used to implement the face detector was taken from a public dataset provided by the Cornell University
(available here : http://chenlab.ece.cornell.edu/people/Andy/ImagesOfGroups.html).

It consists of images containing groups of people (anything from 2 to 20+ people) in various situations and environments (family pictures, weddings, group pictures). The images vary in size and contain 3 color channels (RGB).

b. Algorithm

i. Preprocessing

Images were converted into 2-dimensional pictures (B/W) to match the input expected by our model and were resized to a standard 500x500 pixels size.

ii. Pyramidical sliding window

As described above, we implemented a pyramidical sliding window to ensure that our classifier matches any face in the picture with its valid face input size.

The relevant parameters associated with the pyramidical sliding window are the pyramid downscaling factor and the window step.
For a first version, we decided to use the following values :

Pyramid downscaling factor	Window step size
1.2	5 pixels

iii. Evaluation

To evaluate the face detector's accuracy, we implemented a visual evaluation system : for every face detected, the detector draws the 36x36 window inside which the face was detected.

This way we can visually determine not only its accuracy but also the false positives (i.e detecting a face when there is no face to be detected) and the false negatives (i.e not detecting a face when a face was to be detected)

c. Results

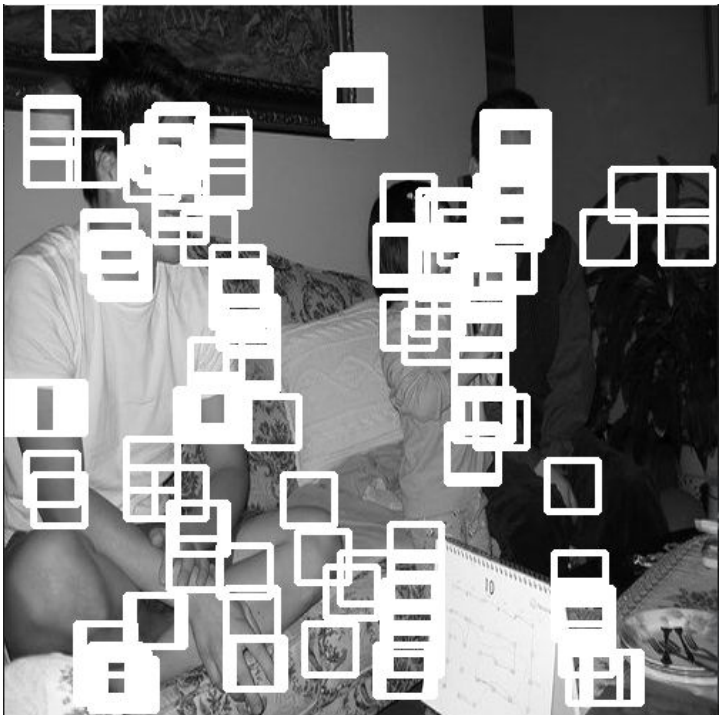

Figure 1 : First version of the face detector

As shown in the figure above (see Figure 1), the detector effectively detects every face on the picture, but several problems remain :

- it falsely detects many no-faces as faces
- it detects the same face multiple times

To solve these issues we'll be looking at various strategies, all of which are described in the following sections.

4.Enhancing the face detector

a. False-positives

As mentioned above, the first version of the face detector resulted in lots of false positives.

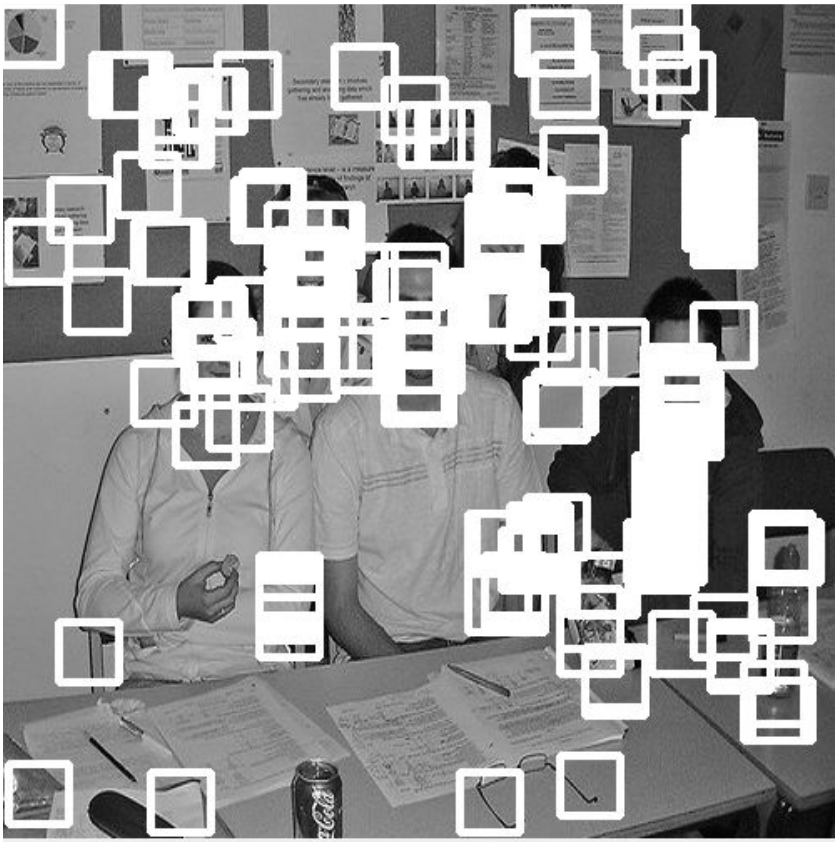

Figure 2 : False positives on the face detector

To decide if the window is detecting a face or not, the detector computes a probability of detecting a face thanks to the softmax function. By default, the softmax threshold is at 0,5. It means that above 0,5, the detector decides that the sliding window is detecting a face.
The low threshold explains why there are so many false-positives.

By simply switching the softmax threshold to a very high value (0.99) we get the following results :

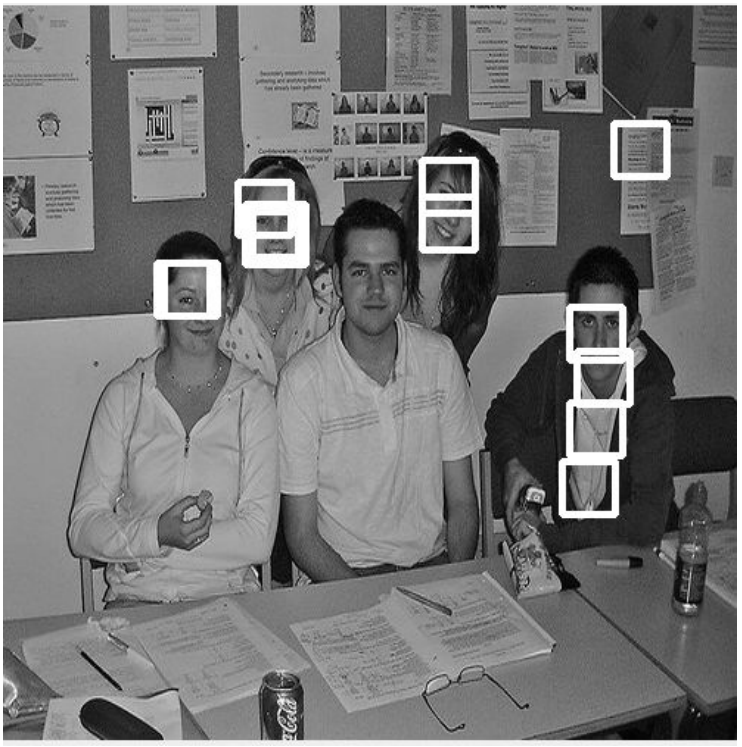

Figure 3 : Face detector with softmax threshold at 0.99

We succeeded in clearing most of the false positives out of the image but a few of them persisted. Furthermore, the threshold does not work equally for all images in our dataset. For some it has to be less or more than 0,99 in order to correctly clear all of the false positives out of the image.

A low threshold means many false positives and no false negatives, but a too high threshold means no false positives but potentially some false negatives.

Nevertheless, we found the 0.998 threshold value to give the best all-in-all results (see Figure 3).

b. Multiple detections of the same face

As shown in the previous results (see Figure 3), the detector can sometimes detect the same face more than once. As we want our final face detector to be able to determine the number of people present in a picture, that behaviour is to be eliminated.

The multiple detections of the same face are due to two major reasons :

- Overlapping bounding boxes (i.e sliding windows) are never merged

- When printing the final results in the original size image, bounding boxes are not rescaled according to the downscale factor of the pyramid-level in which they were created

To solve the overlapping bounding boxes issue, we use the Non-Max-Suppression algorithm (NMS). The algorithm successfully merges all overlapping boxes within an area ratio threshold. Through iteration, we have found the optimal threshold to be equal to 0.01.

Furthermore, to correctly take into account the bounding boxes of every pyramid level, we only use the NMS algorithm once, and we feed it every bounding box found through all the pyramid iterations. To ensure that the bounding boxes match the original image size, we rescale them according to their downscaled factor size before giving them as an input to the NMS algorithm.

The results were as follows :

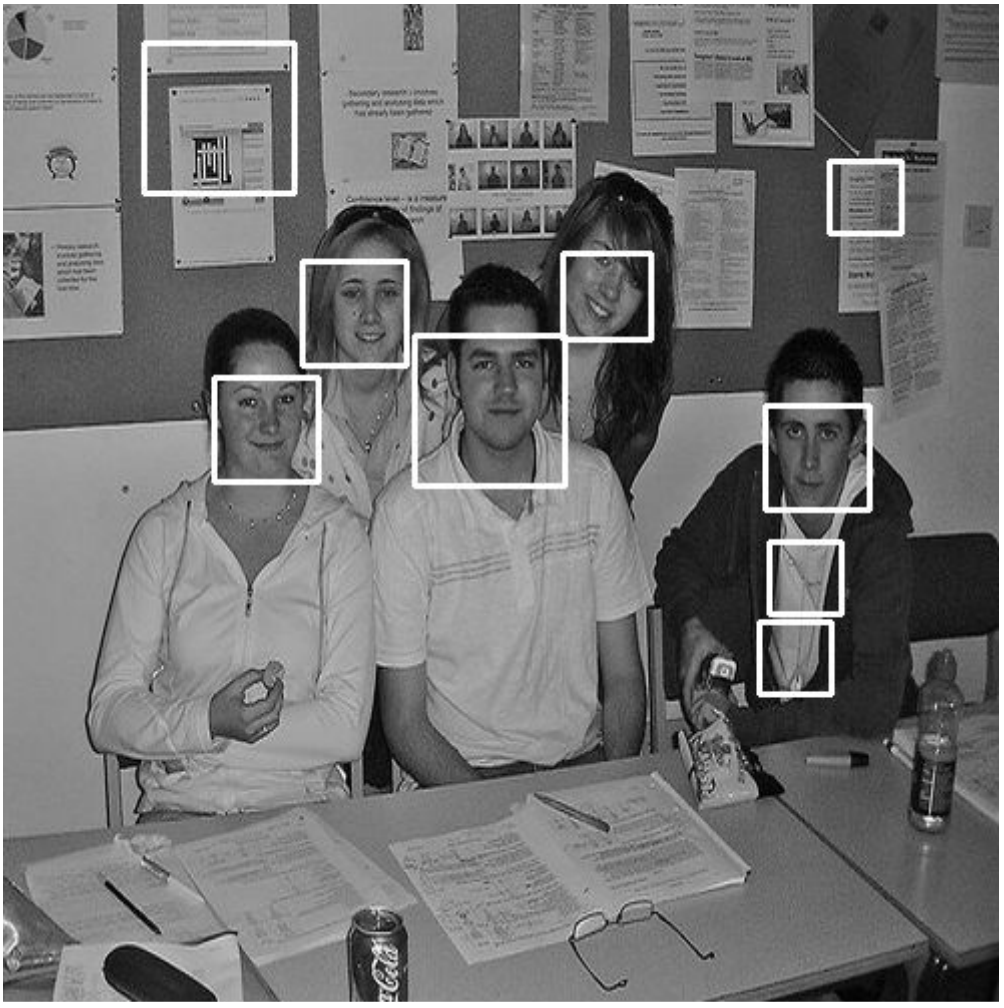

Figure 4 : Face detector with softmax threshold at 0.998 and NMS algorithm to merge overlapping bounding boxes

The detector successfully detects every face of the picture once (see Figure 4), which is the correct expected behaviour.

However, a few problems persist :

- The existence of a few false positives due to the very high softmax threshold (0.998)
- More rarely, the existence of a few false negatives (i.e faces not detected as so)

These two problems are not within the reach of the face detector. To solve them, we need to improve the accuracy of the face/no-face classifier, and that's what we'll be doing in the following sections.

5. Enhancing the face/no-face classifier

a. Class imbalance

The dataset for testing is considerably unbalanced, which can cause problems with the accuracy.

For the model, the results of the classification by class in the test set were the following:

	Face	No Face
Classified as Face	37% of all faces	1% of all no faces
Classified as no face	62% of all faces	99% of all no faces

As these results show, even if the model has a high accuracy, it seems to have trouble detecting faces.

To address this problem, we decided to give weight to the training samples, giving more importance to the no face class since it has less samples.
After adding the different weights, the results were:

	Face	No Face
Classified as Face	41% of all faces	1% of all no faces
Classified as no face	58% of all faces	99% of all no faces

This is a small improvement over the old model, with a total accuracy of 93%

b. Overfitting

To more accurately train and evaluate our model, we can implement an evaluation over a validation set in every epoch. In this case, the validation set is composed of a 20% of the training set and selected at random.
During training, at every epoch the performance in the validation set is calculated. If the performance doesn't improve in a few iterations, we can stop the training earlier so that the model doesn't overfit.
We also save the model that has the best performance in the validation set, that may or may not be the model obtained after the last iteration of the training.
While these modifications didn't have a noticeable impact on the accuracy of the model, they help better control the training and make sure we are saving the best possible model.

c. Number of epochs and other parameters

Another way of improving the model is to experiment with different epochs and batch_size. We can also experiment with different optimizers parameters.
First, augmenting the number of epochs can make better models, but only up to a point. After that, the improvements are very small and there can even be overfitting. Using more epochs also has the disadvantage of requiring more training time.
For our model, good accuracy is obtained at around 11 epochs, as shown below:

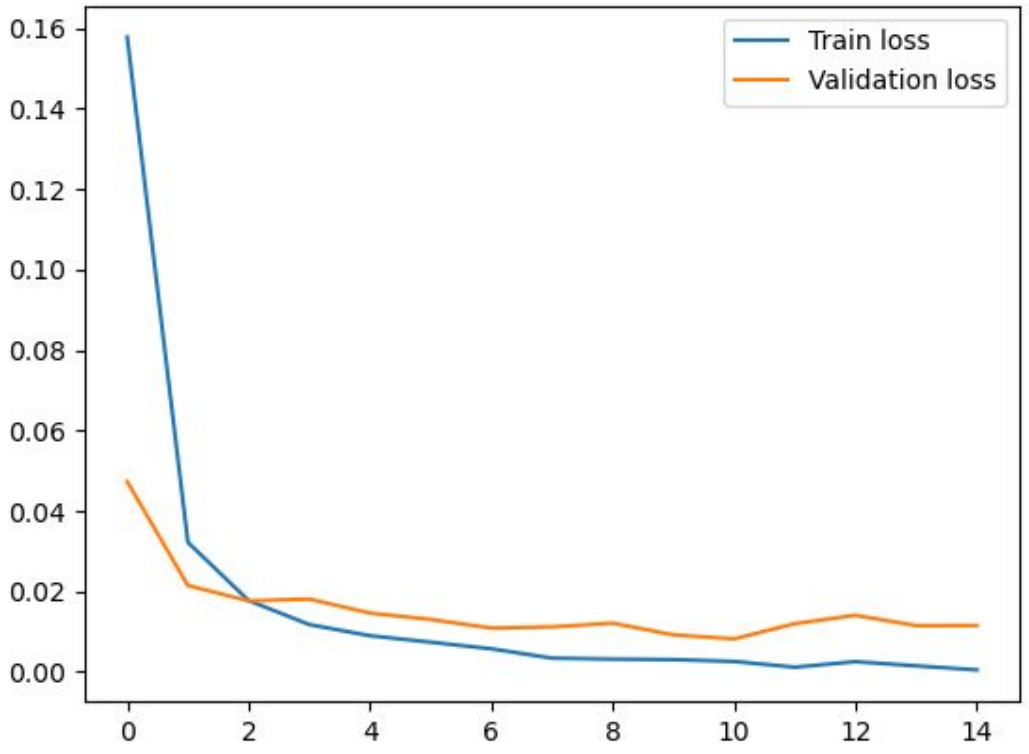

Figure 5 : validation and training loss during 15 epochs

Also, the training will often be made to stop around the 15th epoch, since the performance on the validation set stops increasing.

The other parameter that proved to have an effect on the accuracy is the learning rate of the optimizer. Varying this parameter between 0.001 and 0.1, the best model was obtained with 0.01 with an accuracy of 94%.

d. Fine-tuning the CNN model

Another way to enhance the model is by fine-tuning, this involves re-training the model we already had with some new data, but instead of letting it be trained from zero again, we just train the parameters from the last layer.

The initial results obtained by this method were not good, with accuracies several points below those obtained before, so further experimentation would be needed with this method to see if it can be used to improve the accuracy in a significant way.

6.Final results

After improving the face detector by reducing false positives and merging overlapping bounding boxes, and improving the face classifier by handling issues such as class imbalance, overfitting and the number of training epochs, the results have improved greatly (See Figure 6).

Our final system successfully detects every face once, it rarely gives false negatives.

However, a few issues still remain, the biggest of which being the existence of some false positives and the limitations of the NMS algorithm to merge overlapping bounding boxes.

For the false positives, the solution would be to improve the accuracy of the face classifier (94% at the moment). One of the solutions that’s not been explored enough is to programmatically fine-tune the model.

For the NMS algorithm, it could also be improved as it tends to occasionally not merge the correct bounding boxes (See picture in the middle Figure 5, the correct bounding box for the baby face was probably merged because it overlapped with the mom’s face). This behaviour is especially visible on pictures with very close faces.

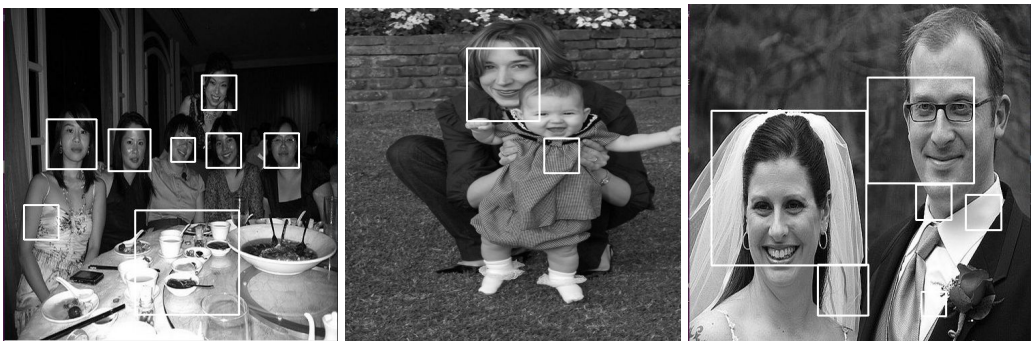

Figure 6 : Final results

7. Conclusion

The project resulted in the implementation of an end-to-end face recognition system that uses convolutional neural networks, using the following steps :

- Choosing and Preprocessing the datasets
- Choosing the neural network architecture
- Implementing the face detector
- Visualizing and Evaluating results

It also resulted in all of us getting a better grasp of the difficulties surrounding Machine Learning and Data Science problems such as the difficulty of extracting semantic information from images, the functioning of a CNN, the influence of training data and overfitting on the final results, the importance of the network's architecture, the complexity/accuracy tradeoff and the limitations of learning models in the context of a face recognition task.