Facial Expression Recognition
Deep Learning vs Smart Features

Lazraq Yahya
yahya.lazraq@student.ecp.fr
Ben-Goumi Meryem
meryem.ben-goumi@student.ecp.fr
Olympie Gabriel
gabriel.olympie@student.ecp.fr

1 ABSTRACT

This is our report on the experiments and the research work done on Facial Emotion Recognition (FER). The associated code will be available on the following link github . The dataset used for our experiment is known as the FER-2013 Dataset and is available on the following link. The references and papers used to support and develop our approach are available on the last section of this report.

2 INTRODUCTION

2.1 Motivation

The goal of this project will be to build an algorithm able to detect a range of emotions on different test subjects. Those emotions are anger, disgust, fear, happiness, sadness, surprise and neutral. In order to perform the task at hand, we will try two different approaches, one based on smart features extraction, and the second based on a deep-neural network. Our experiments use the Fer-2013 set of images as a data input.
Since the problem of emotion recognition is a relatively easy task for humans, the datasets produced have a good precision. However, it turns out to be a quite challenging task for a computer. Emotion recognition have been a the focus of many research projects since the 90s, and even more in the recent past with the increasing tools available for image processing. Our bibliographic research work allowed us to understand that most of the proposed solution are built with a similar three-step-structure:

- Face detection
- Feature extraction
- Classification

The *face detection step* is a pre-processing step which importance varies depending on the problem at hand. Indeed, it is essential to be able to find where the face is, and to track it when the data is composed of a video stream. On static images, it is also useful but can be included in the feature extraction step. In our problem, we use pre-processed images that are directly centered on the face, making this pre-processing step unnecessary. The *feature extraction step* is essential, because it determines the quality of the data we try to process. In literature, there are several kinds of features that can be used for this purpose:

- **Raw Images:** they represent the highest level of feature that we could use, since they do not need any kind of processing except for adapting the picture to the classifier.
- **Geometry based features:** given a geometric representation of the face, it consists in extracting some key points from the face (for example: the top of the nose, corners of the eyes, etc.). An other approach relies on a surface modelisation of the face, that is used as an input of the classifier.
- **Psychology based features:** some theoreticians, like Paul Ekman, tried to find recurrent patterns in the expression of emotions on the faces. They generalized their theory by defining some landmarks that can be found on any human face, and whom combined activation reveals an emotion. Finding these landmarks on the face and being able to recognize the activation with an algorithm is the deepest level of feature representation that exists today.

The *classification step* is the last step of the modelisation, and is different depending on the kind of features extracted. In literature, the most widely used classifiers are Deep Convolution Networks for raw images inputs, and Support Vector Machines for smart features.

The problem of emotion recognition has many applications today. In marketing, it can allow retailers to better understand how their customers perceive their products. It can also be used by law enforcers to predict violent behaviors and prevent them. In smart cars, it can prevent an accident by analyzing the driver's behavior. For everyday life, it can help our devices to be more customized to our habits.

2.2 Problem Definition

For our project, we will base our work on the FER dataset, composed of static annotated images of faces displaying emotions. We will then structure our approach along with the framework described above.

- Face detection : this part was already performed by the creator of the dataset
- Feature extraction : we will try different kinds of features, mostly based on raw images and geometry based representations
- Classification : we will compare the performance of different classifiers, especially neural networks and the support vector machines.

Finally, we will compare our best results with the leaderboard presented on Kaggle, that hosted a competition based on the very same dataset. Our key metric to assess the quality of our model will be the accuracy, in order to have a comparison baseline with the contestants of Kaggle.

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ examples} \tag{1}$$

3 RELATED WORK

State-of-the-art researches on Facial Emotion Recognition can be grouped into three types of approaches:

- **Convential Approaches**, that rely on custom-made smart features and traditional Machine Learning classifiers
- **Deep Learning Approaches**, that requires a training of deep neural networks
- **Hybrid Approaches**, that input the custom-made smart features into a deep learning network

It should be noted that there is related work implementing these three approaches for both image and video inputs. However, we have decided to focus, for our project, on classifying images.

As explained in paper [1], for **Conventional Approaches**, it is critical to compute smart features that capture the expression in the input face image. Once computed, such features are used to train traditional Machine Learning classifiers such as a Support Vector Machine (SVM) or a Random Forest. This type of implementation, using a SVM, is available in [4]. It turns out that two types of features are particularly useful for such emotion classification. The first one relies on a *Face Landmarks Localization* method, that is implemented in `dlib` library. The second one, *Hog features* (histogram of Oriented Gradients), is widely used in object detection for its ability to reflect local shape of objects in an image. It is known to provide particularly significant results when used to train a linear SVM [5].

However, the results obtained with such traditional classifiers are not really satisfying, as variations in face expressions from an emotion to another happen to be quite subtle. For this reason, more researchers have been actively exploring **Deep Learning Approaches** and **Hybrid Approaches**, that use Convolutional Neural Networks (CNNs) to extract features and predict expressions. [2]

Deep Learning Approaches rely on filters of convolution layers to convolve the input images and build feature maps, which resolution is then reduced using max-pooling layers.

The performance can then be further improved with **Hybrid Approaches**, that leverage also on smart features to better train the neuronal networks.

Differences in performance between the different CNN-based methods are due to variations in:

- *Image Preprocessing*: face detection, face registration, correction of illumination
- *CNN Architecture*: different layers, depths, and parameters
- *Training*: smart features, size of training and test sets, number of epochs.

The best state-of-the-art results on FER-1023 dataset have an accuracy of 75%, that were obtained with *"a Convolutional Neural Network used during several hours on GPUs"* [2].

4 METHODOLOGY

This section will begin by an introduction of the dataset used and challenges associated. Then, the classification methodologies will be presented.

4.1 The FER-2013 Dataset

The FER2018 (Facial Emotion Recognition) is a publicly available dataset that have been submitted on Kaggle for a competition.

The data consists of 48x48 pixel grayscale images of faces. The faces have been automatically registered so that the face is more or less centered and occupies about the same amount of space in each image. The task is to categorize each face based on the emotion shown in the facial expression into one of seven categories (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral).

The dataset is already separated between a training set, composed from 28709 labelled image, and a validation set, composed from 3589 labelled image.

The dataset can be challenging to process for several reasons:

- The images are not aligned
- Some images are mislabelled
- Some samples do not contain a face
- The classes of emotions are strongly unbalanced

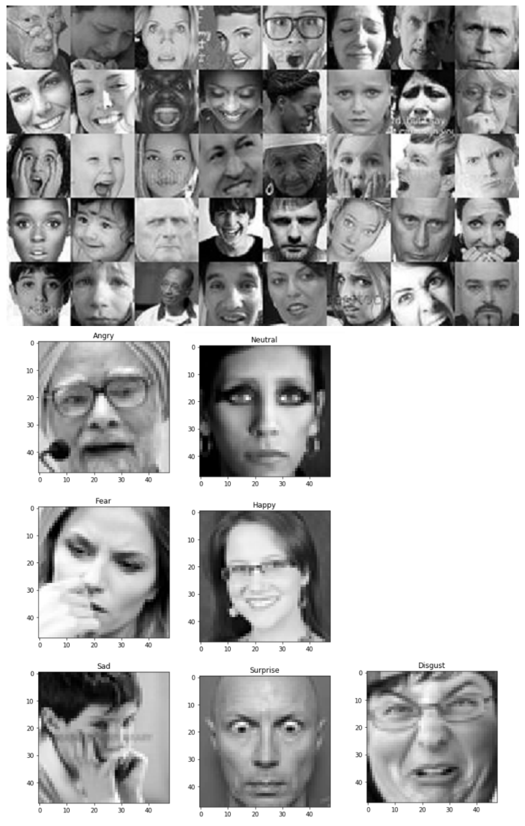


Figure 1: Sample extracted from the dataset

4.2 Features Extraction

Face Landmarks. Traditional FER approaches rely on the detection of facial landmarks, i.e. notable regions of the face:

- eyes
- nose

- jawline
- mouth
- eyebrows
- chin …

Detecting these regions is a particular instance of a wider *shape detection* problem, that localizes points of key attributes within an image. The process that consists in the extraction of such smart features, called *Face Alignment* or *Face Landmark Localization*, can be achieved using the `shape_predictor` function of the `dlib` library. This function takes as an input a trained model that we downloaded from dlib's website (link).

The model mentioned above is publicly available. It relies on an Ensemble of Regression Trees (ERT) [3]: a cascade of regressors is trained, using gradient boosting, to detect the positions of 68 key points in faces. The key points are deducted from differences in pixel intensities. Hence, the cascade of regressors allows an iterative improvement, at each level, of the alignment error. This model is particularly renown for its impressive accuracy/running time trade-off: it takes less than 3ms to generate a set of 68 (x,y) coordinates representing the landmarks.

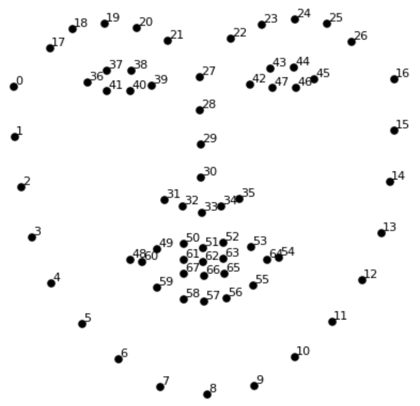


Figure 2: 68 facial key points that the model is trained to detect. Source: https://ibug.doc.ic.ac.uk/resources/facial-point-annotations/

Let's note that this dlib predictor is particularly useful because it can be directly used to extract landmarks from any dataset of images. In figure 3, we can visualize the landmarks detected in some of our images.

However, as mentioned in section 4.1, the images in this dataset are often blurry, and they do not always contain faces in frontal view. As a result, the positions of landmarks are not always optimal, in particular when the faces are in profile view (figure 4).

To help solve this, we added a processing step to the output of our function `compute_dlib_landmarks`, that centers the 68 key points around the 30th point (ie the nose). This way, we make sure to avoid face alignment errors that, for instance, localize the eyes in the forehead when the image is too dark.

Another possible issue is the treatment of an input image that does not contain any face, or, on the contrary, too many faces. To check if we might have any anomalies of this type, we ran a

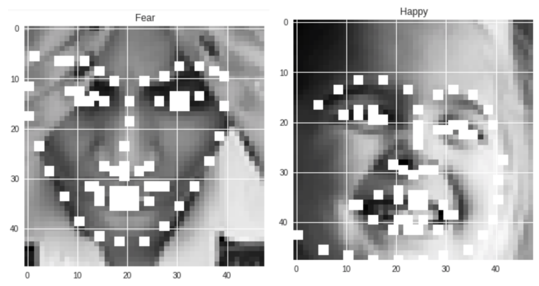


Figure 3: Screenshots of Facial Landmarks detected in images from FER2013 dataset

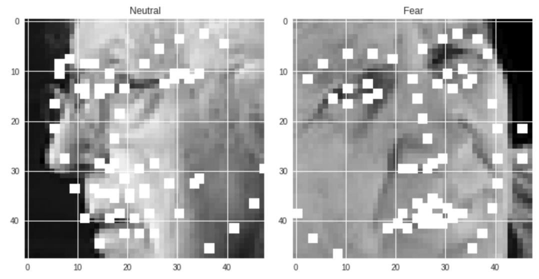


Figure 4: Screenshots of Facial Landmarks detected in images from FER2013 dataset

code (link) that uses the `dlib` library to raise errors when such exceptions occur. As the test did not raise any exceptions, we can fairly assume that the vast majority of the dataset contains only one face image. These findings are coherent with the fact that the dataset was made for an Emotion Recognition contest.

Hog feature. The Hog Feature, or *Histogram of Oriented Gradients*, is a widely used feature in computer vision for image analysis and object detection. This feature relies on the idea according to which the distribution of gradient intensity reflects the appearance of objects within an image. Indeed, it turns out that gradient orientations are able to capture the notion of "shape".

To extract this feature, we used the `feature.hog` function of `skimage`. It splits each image into cells, and then computes a Histogram of Oriented Gradients for each pixel within the cells. The histograms obtained for each pixel of the same cell are then grouped into *orientation bins*. Hence, predominant gradients contribute to the bin with a higher weight, limiting thus the impact of noise. Therefore, the *orientation bins*, called accordingly, capture the strong orientation of the cell.

The output for each image, called *hog*, is obtained after concatenating the histograms of all cells in the image. It reflects shape variations in objects detected.

The parameters to specify as an input for the hog function and the values that we chose for each are:

- `orientations`, the number of orientation bins: 8
- `pixels_per_cell`: (8,8)
- `cells_per_block`: (1,1)

The last parameter, `cells_per_block`, allows to perform a normalization step along blocks of cells, in order to avoid orientation disparities provoked by lightning fluctuations. This step is meant to increase the robustness of the computed features and thus to improve results. However, as it was already time-consuming to compute the hog features for the whole dataset, we chose not to include the normalization, by setting the `cells_per_block` at (1,1). Nonetheless, let's note that including such normalization can allow a further improvement of our results, provided that the features would take longer to compute.

Hog feature with a Sliding Window. The ability for the hog feature to capture variations of shape can be improved by computing the oriented gradients for a sliding window rather than doing it directly for the whole image. As explained in [5], this feature has proven to be particularly efficient for training linear classifiers such as a Support Vector Machine. With a `Window Size` of 24 and an exploration of the image with a `Window Step` of 6, we obtain hog features that can be visualized for some of our images below:

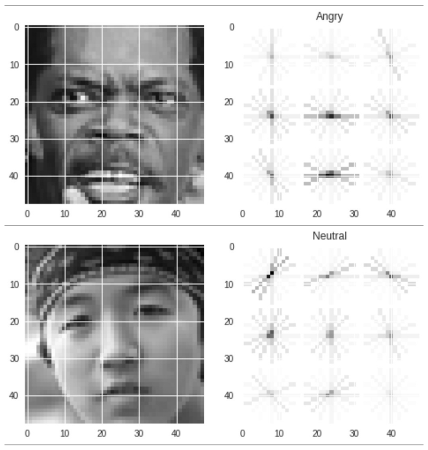


Figure 5: Images and Associated Hog Features computed with a Sliding Window

As it can be seen in the screenshots above, variations in expression (from angry to neutral here) are captured by variations in the Histograms of Oriented Gradients.

4.3 Prediction Models

SVM. As more sophisticated/complex classification methods are more demanding in terms of computational power and time, we have decided to start with linear and basic classification methods. SVM, seemed to be one of the most appropriate.
Basically, the SVM tries to find a set of hyper-planes splitting the point based on their class. Each space between hyper-planes corresponds to a certain class. The particularity of the SVM is that it tries to maximize the distance between the hyper-plan and the nearest training data points. To deal with the non linearity of data, we have decided to use `rbf` kernel to add complexity to the model. Still, we should note that as the model becomes more complex, the risk of over-fitting raise.

Random Forest. Thinking about the features extracted and analyzing them suggested that some may be correlated and an algorithm selecting features intrinsically would work well. We needed also a non linear algorithm, robust and able to avoid overfitting. We decided then to try Random Forest. Random Forest constructs a multitude of decision trees. The training is performed through the extraction of random N lines (based on bagging method) and p features (sometimes called feature bagging) from the original set. The output is then a sort of mean of all the results of those multiple trees.

Deep Learning. After trying a features-based approach, we decided to use neural networks for this classification task and evaluate the performance of Convolutional Neural networks (CNN) on extracting relevant information for the needed task.
We started by a network composed of 5 Convolutional layers separated from each other by max-pooling/average-pooling layers. The whole is followed by a fully connected layer of 2 dense layers; the last layer being a 7-neurons Dense layer with sigmoid function. This last layer enables to determine a probability of belonging to each of the 7 emotions. A dropout was added on the fully connected network to reduce over-fitting. The input of this network is a picture from the FER dataset and the output is a 7-elements array, with the highest value element giving the emotion associated to the picture. A summary of the architecture is presented in Figure 6.

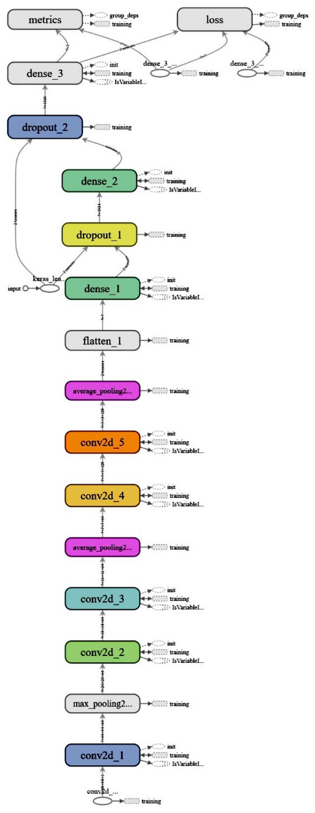


Figure 6: CNN network taking as an input raw pictures

The network was trained on all the extracted FER training dataset, and the performance was evaluated on the testing set. We used an SGD optimizer, and we set the loss to be minimized as a categorical cross-entropy.
Therefore, this network takes as input raw pictures and performs predictions on it.
Hence, We thought that if we could improve our results by combining both approaches mentioned above, i.e. using the features extracted for the SVM as an input for the CNN.
Thus, we decided to build a second network, this time taking two different inputs: raw pictures and "smart" features (Face Landmarks and HOG features with a sliding window). To extract information from raw pictures, we built an other convolutional neural network, less deep this time due to computational power limitations. This network is composed of 3 consecutive convolutional layers, each one followed by a Max-pooling and a batch-normalization layer. This CNN is followed by 2 dense layers with dropout layers to limit overfitting. On parallel of that, the "smart features" are given as an input of a fully connected network made of 2 dense layers. The result of each network is then combined and the output layer is a 7 neurons dense layer with a sigmoid activation function. Again, we used the SGD optimizer and categorical cross-entropy. A representation of the neural network used on this second approach is presented in Figure 7.

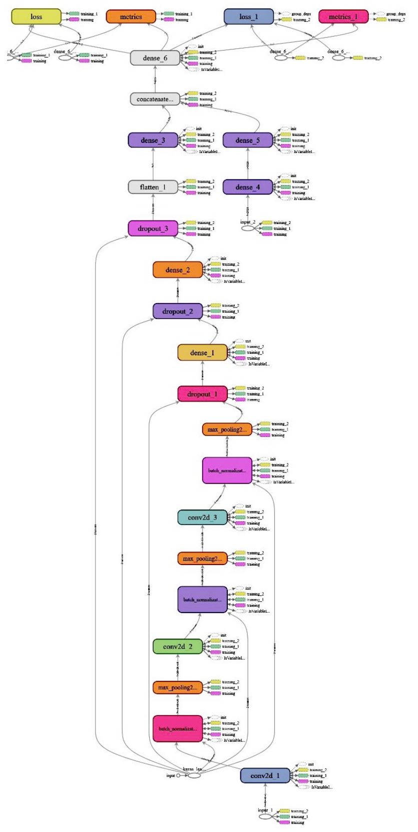


Figure 7: CNN + FC networks taking 2 inputs: raw pictures + "smart" features

5 EVALUATION

To be able to check the performance of our classification, we will first compare the global accuracy of the approaches and the F1-score split-up between emotions. Additionally, We will analyze the computational time for each approach as it is also an important element to look at when choosing a method.

5.1 Accuracy

The accuracy corresponds to the *'ratio of true outcomes (both true positive to true negative) to the total number of cases examined.'* To analyze the performance of our approaches, we will start by analyzing the accuracy. Here is a summary of all the results obtained:

Method used	Accuracy
SVM with smart features	0.48
Random Forest with smart features	0.49
CNN with raw pictures	0.55
CNN + FC on raw pictures + smart features	0.60

As we can deduce it from the table above, the best results are obtained using the Convolutional Neuronal Networks mixed with a Fully Connected layer and taking as input raw pictures and smart extracted features. To have a point of comparison, a Kaggle competition has been used to respond to the same problematic of FER on the same dataset. This Kaggle is available here. With our model performance, we would have been scored among the top 25% best scores of the competition. Considering that this was a rewarded Kaggle, and given the amount of time and computational power available, we could rate our results as satisfying.

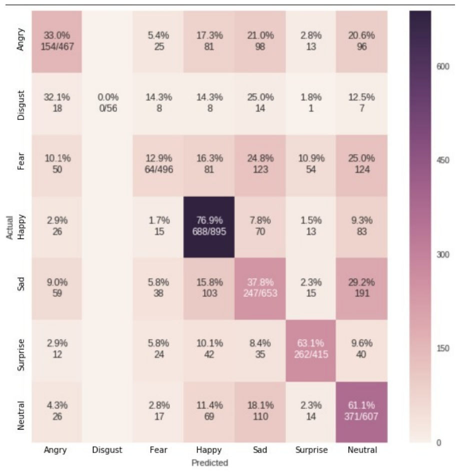


Figure 8: Confusion matrix for the SVM classification

In figure 8, the confusion matrix for the SVM prediction (the diagonal corresponding to the accuracy per emotion) shows that the SVM was not able to predict low represented classes (disgust). We decided to extract then, as an experiment, from the initial dataset, a set of pictures that would have balanced classes of emotions. The extracted dataset has 3436 rows; each class having almost the same number of pictures in each emotion. We followed the same approach on the Testing and Validation dataset. Consequently, the accuracy results slightly improved but not significantly.
On the other hand, we thought that a Random Forest might help us solve the problem of un-predicted classes, since as it uses bagging methods and trains each tree on a subset of the dataset. In figure 9, the confusion matrix of the Random Forest prediction shows that not only it improved our score, but it also helped to predict the "disgust" emotion, whereas SVM failed to do so.
We performed the same type of analysis for the two networks and we obtained the results in figure 10. The graph on the left corresponds to the CNN and the one on the right to the CNN + FC.

The CNN + FC demonstrates the best performances for the 7 emotions. We can also notice that neural networks perform much better than traditional classification methods on low represented classes.

5.2 F1 Score

The F1 score is calculated based on 2 elements :

- **Recal:** This metric correspond to "the correct recognition of emotion i over the actual number of images with emotion i."
- **Precision:** This metrics correspond to "the fraction of automatic annotation of emotion i that are correctly recognized"

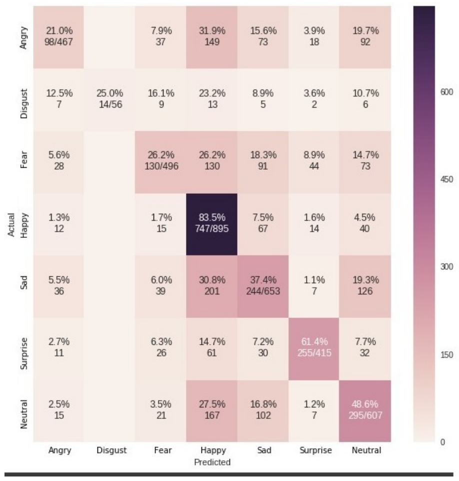


Figure 9: Confusion matrix for the Random Forest classification

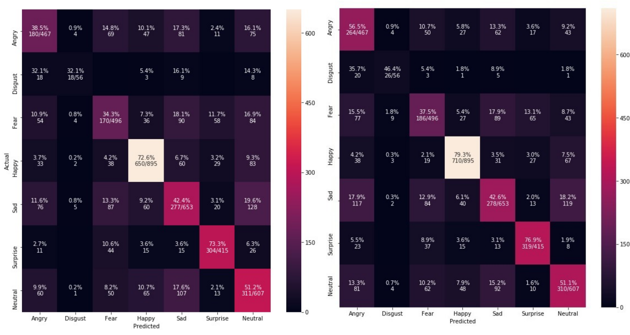


Figure 10: Confusion matrix for the networks (left: CNN; right: CNN + FC)

The F1 score for each emotion is then obtained using the following formula :

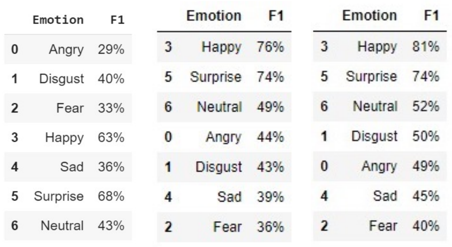
$$F1 = 2 * \frac{precision * recall}{precision + recall} \tag{2}$$

Below is a summary of all the F1 scores obtained for the different methods except for the SVM, which was excluded due to non-prediction of the disgust emotion (from the left to the right : Random Forest, CNN network, CNN + FC network) :

As we can see it here again, the CNN + FC is the top performer.

5.3 Computational Time

In addition the the accuracy and the F1 score, the SVM had clearly under-performed in term of computational time, as it took more than 1 hour to run. This is due to the high dimension of the input data. Indeed, there was more than 2000 features presented to the SVM. With such dimensions, performing distance calculation is really demanding and drives the slow computing. Using a dimension reduction method as PCA could clearly improve this computational time and even help improve the results as noise would have less impact.



	Emotion	F1
0	Angry	29%
1	Disgust	40%
2	Fear	33%
3	Happy	63%
4	Sad	36%
5	Surprise	68%
6	Neutral	43%

	Emotion	F1
3	Happy	76%
5	Surprise	74%
6	Neutral	49%
0	Angry	44%
1	Disgust	43%
4	Sad	39%
2	Fear	36%

	Emotion	F1
3	Happy	81%
5	Surprise	74%
6	Neutral	52%
1	Disgust	50%
0	Angry	49%
4	Sad	45%
2	Fear	40%

Figure 11: Summary of F1 scores (from the left to the right : Random Forest, CNN network, CNN + FC network)

On the contrary, the Random Forest has outperformed the other methods in term of computational time as it produces results in less than 4 minutes.
In Figure 12, when moving to deep-learning methods, we observe that above 20 epochs, the score stops improving for the simple CNN whereas we needed 28 epochs for the other network to reach its best performance. As more parameters had to be tuned onto the CNN + FC network, the epoch for this network took more time to run (41s vs 12s) but the overall time taken by the networks is still competitive, as we reached our best score in almost 19 minutes for the CNN + FC network and 8 minutes for the simple CNN.

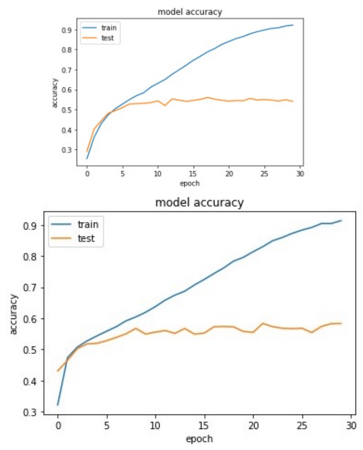


Figure 12: Accuracy evolution on the train set and test set for networks during training (on the left: CNN; on the right: CNN + FC)

On the following, a summary of the different computational times :

Method used	Time
SVM with smart features	4566s
Random Forest with smart features	187s
CNN with raw pictures	440s
CNN + FC on raw pictures + smart features	1148s

6 CONCLUSIONS

In conclusion, this exercise has been extremely formative in terms of algorithmic methods and process implementation. The task was also intuitively challenging in terms accuracy results. It was, for us, a very interesting dive into image processing methods and applications. The task of Emotion Recognition is so subtle that we were fairly impressed with what Visual Computing techniques allows to do with it.
Overall, we retain two main outlines. First of all, for the Facial Expression Recognition task, deep learning performs better than traditional classification methods. Secondly, it should be noted that a hybrid approach with networks taking as an input handcrafted features and raw pictures delivers strong results on reasonable computational time.

7 REFERENCES

(1) A very well-detailed post: Byoung Chul Ko. [January 2018]. A Brief Review of Facial Emotion Recognition Based on Visual Information.
(2) Christopher Pramerdorfer, Martin Kampel. Facial Expression Recognition using Convolutional Neural Networks: State of the Art.
(3) A paper explaining the approach behind dlib's implementation of face landmarks detection: Vahid Kazemi, Josephine Sullivan [2014]. *One Millisecond Face Alignment with an Ensemble of Regression Trees.* Available at: link.
(4) A Git by Amine Bendahmane implementing the Conventional Expression Detection Method, using smart Features and a trained SVM classifier, as well as a Deep Learning Method link.
(5) A paper detailing the benefits of Hog features when training a SVM. Hilton Bristow and Simon Lucey. Why do linear SVMs trained on HOG features perform so well?
(6) Moon Hwan Kim, Young Hoon Joo, and Jin Bae Park. Emotion Detection Algorithm Using Frontal Face Image. July 2015
(7) PaweÅĆ Tarnowski, Marcin KoÅĆodziej, Andrzej Majkowski, Remigiusz J. Rak. Emotion recognition using facial expressions. June 2017
(8) Mengyi Liu, Shaoxin Li, Shiguang Shan, Xilin Chen. AU-inspired Deep Networks for Facial Expression Feature Learning. February 2015.