# Implémantation d'un système de détection et de reconnaissance faciale

Alexandre Bonhomme
Université Paul Sabatier, Toulouse
Florent Maufras
EFREI, Paris
Timothée Planté
EISTI, Pau

## 1 Introduction

La reconnaissance faciale est actuellement un domaine en plein essort. Elle rentre petit à petit dans nos vies au travers de nos téléphones mobiles ou de nos ordinateurs portables par exemple. Malgré l'amélioration du taux de détection elle reste actuellement l'objet de nombreuses études.

L'objectif de notre projet sera de mettre en oeuvre un système complet permettant la détection et la reconnaissance de visage issue de plusieurs bases de données. Pour ce faire nous implémenterons différents algorithmes de reconnaissance et nous utiliserons la bibliothèque OpenCV [3] et son implémentation de l'algorithme de Viola & Jones pour la détection de visage.

## 2 Algorithmes mis en oeuvre

Au cours de notre projet nous avons confronté différents algorithmes afin de tester leurs aptitudes sur plusieurs ensembles de données différents.

### 2.1 K Plus Proches Voisins et Fenêtres de Parzen

L'algorithme des «K plus proches voisins» (KPPV) utilise le voisinage du point de test afin de déterminer sa classe. C'est la classe majoritaire parmi ces $K$ points qui sera attribuée au point de test (1).

Pour cet algorithme il n'y a pas d'apprentissage des paramètres sur l'ensemble d'entrainement, seulement une optimisation des l'hyperparamètres $K$ sur l'ensemble de validation.

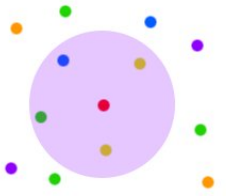


**Figure 1** – K Plus Proches Voisins

Dans la Figure 1 le point de test (en rouge) va recevoir comme prédiction la classe jaune car c'est elle la plus présente dans les $K = 4$ voisins les plus proches. Le disque ici n'est que représentatif pour mettre en évidence les voisins. Les points voisins peuvent être à une distance plus ou moins importantes de notre point test.

#### 2.1.1 Extension avec les fenêtres de Parzen

L'application que l'on a fait ici de Parzen nous permet de pondérer les points du voisinage. Plus un point sera distant du point à l'étude plus son poids sera faible. Cela est calculé à partir d'une gaussienne centrée sur le point à l'étude et d'une largeur fixée par $\sigma$, dans notre cas $\sigma$ sera fixé par l'utilisateur.

### 2.2 Réseau de Neurones

Le réseau de neurone utilisé dans ce projet est de type Perceptron Multicouche. Ce modèle est entrainé par descente de gradient de bacth sur un ensemble d'entrainement afin d'optimiser les paramètres du réseau en réduisant le risque empirique régularisé. On tâchera ensuite d'optimiser les hyperparamètres de contrôle de capacité sur un ensemble de validation séparé.

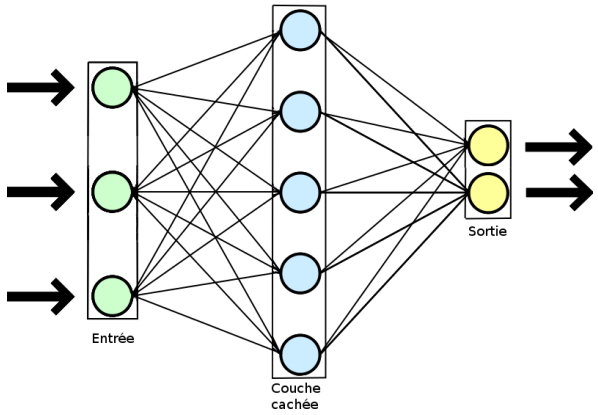



**Figure 2** – Réseau de neurones

Dans notre cas nous utiliserons une architechture monocouche (cf. Figure 2) et une pénalité de type «weight decay» (1) pour la régularisation car elle défavorisera les poids les plus lourds lors de l'apprentissage des paramètres.

$$\Omega(\theta) = \|\mathbf{W}^{(1)}\|^2 + \|\mathbf{W}^{(2)}\|^2 \qquad (1)$$

Cette pénalité sera pondérée par un hyperparamètre $\lambda$.

## 2.3 Algorithme de Viola et Jones

Afin d'effectuer efficacement la détection de visage nous avons utilisé une implémentation de l'algorithme de Viola et Jones [4] proposé par la célèbre bibliothèque de traitement d'image *OpenCV* [3].

Cet algorithme est basé sur le principe de «boosting» qui consiste à assembler plusieurs classifieurs faibles afin d'obtenir un classifieur à forte capacité. Plutôt que de travailler directement sur les pixels de l'image, Viola P. et Jones M., introduisent des caractéristiques appelées «pseudo-Haar» fortement inspirées des ondelettes de Haar. De plus il utilise une variante d'un algorithme de boosting très célèbre qui est l'«AdaBoost».

**(a)** Image initiale **(b)** Visage extrait

**Figure 3** – Détection d'un visage par l'algorithme de Viola et Jones [4]

Cette technique est particulièrement efficace et offre d'excelents résultats pour la détection de visages (cf. Figure 3) et par extension la détection de motifs (ou d'objets) au sein d'une image.

# 3 Ensembles de données utilisés

Afin d'entrainer nos algorithmes et de mesurer leurs perfomances nous avons essentiellement utilisé deux jeux de données.

## 3.1 Our Database of Faces (ORL [2])

Le premier ensemble de données utilisé est issu des laboratoires de l'entreprise *AT&T*. C'est un ensemble de taille moyenne qui regroupe au total 40 sujets différents avec 10 images par sujet. Les photos ont été prises sous différentes postures et durant plusieurs années et ont une taille de $92 \times 112$ pixels.

Les images sont en niveaux de gris (NdG) et au format PGM.

## 3.2 Labeled Faces in the Wild Home (LFW [5])

Pour le second jeu de données, il est mis a diposition par l'université du Massachusetts et il s'agit de photographies tirées d'Internet et sur lesquels l'algorithme de Viola et Jones [4] a été appliqué afin d'obtenir une image de taille $250 \times 250$ pixels centré sur le visage.

Cet ensemble est relativement grand il contient 13233 images de 5749 sujets différents. Les images sont en couleur et au format JPG.

## 3.3 Prétraitements des données

### 3.3.1 Cas particulier : l'ensemble LFW

Contrairement à l'ensemble ORL, LFW nécessite des prétraitement supplémentaires. En effet, nous avons utilisé l'ensemble LFW pour simuler une entrée réelle dans notre système. Pour ce faire nous avons appliqué l'algorithme de Viola et Jones afin de détecter les visages au sein des images présentes dans l'ensemble.
Nous avons ainsi pu extraire (après divers traitements de redimensionnement) une image de taille $92 \times 115$ pixels en plusieurs NdG.

### 3.3.2 Cas général

Appartir des images en différents NdG nous construisons une représentation vectorielle de celle-ci (cf. Figure 4).

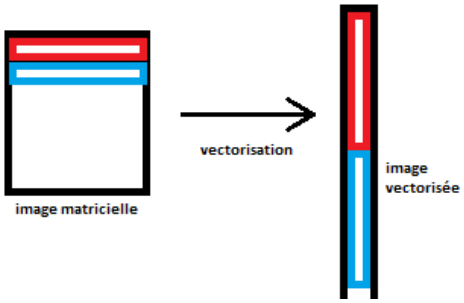



**Figure 4** – Vectorisation d'une image

Cependant cette représentation est très lourde car elle contient de nombreuses caractéristiques. Afin de reduire la dimentionalité de nos données d'entrainement nous avons appliqué l'agorithme du *PCA* («Principal Components Analysis»), plus connu sous le nom de «Eigenfaces» [1] lorsqu'il est appliqué à la reconnaissance facial.

### 3.3.3 Réduction de la dimentionnalité : PCA

Nous avons implémenté le PCA dans ca forme classique.
On normalise l'ensemble de données en sustrayant le «visage» moyen :

$$X_N = X - \bar{x} \tag{2}$$

A la suite de quoi nous utilisons une «astuce mathématique» qui consiste à calculer la Décomposition en Valeurs Singulières (ou SVD pour «Singular Value Decomposition») afin de déterminer les vecteurs propres de la matrice de covariance. On obtient alors :

$$X_N = USU^T \tag{3}$$

Ou les colonnes de $U$ contiennent les vecteurs propres et $S^2$ les valeurs propres associées. On ne conserve alors qu'un nombre $M$ de vecteurs propres (ie. $M$ colonnes de la matrice $U$). Après réduction du nombre de colonnes on obtient une matrice $U'$ (aussi appelée, dans notre cas, «Eigenspace») dans laquelle on projete l'ensemble de données $X$ afin de déterminer la matrice de poids $W$ associée :

$$W = \langle (U')^T, X \rangle \tag{4}$$

Ainsi, pour chaque nouvel exemple $x$ on calculera son vecteur de poids $w$ par projection dans le Eigenspace $U'$.

# 4 Vérifications des algorithmes

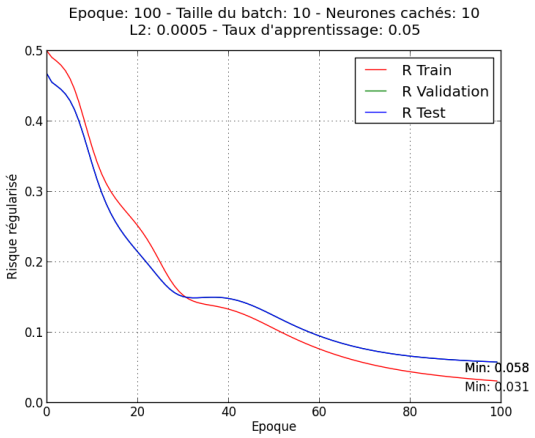
Afin de vérifier le bon fonctionnement des algorithmes implémentés nous avons mis en oeuvre quelques procédures de vérification.

## 4.1 Perceptron Multicouche

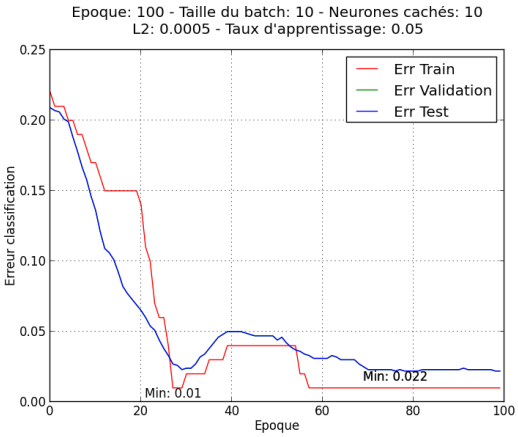
Pour le réseau de neurones nous avons vérifié nos résultats sur l'ensemble de données deux dimensions «MOONS» et nous avons obtenu les résultats de la Figure 5.

**Figure 5** – Vérification du réseau de neurones

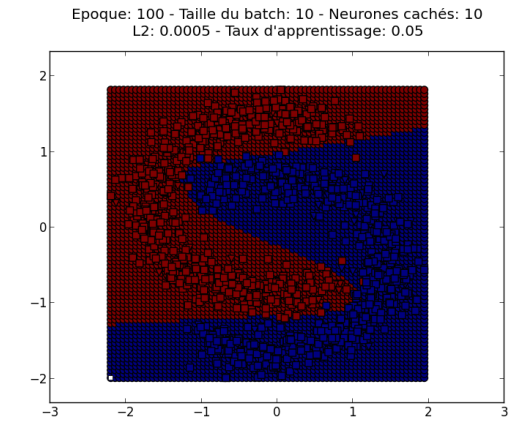
**(a)** Risque empirique




**(b)** Erreur de classification




**(c)** Frontières de décision




## 4.2 Détection de visages

Pour vérifier le bon fonctionnement de la détection de visage nous avons effectué des essais préalables des

images de test telle que celle de la Figure 3.

# 5 Résultats

Afin d'évaluer la performance des deux algorithmes sur les deux ensembles de données nous les avons chacun divisés en trois lots de la manière suivante :

**60%** Ensemble d'entrainement

**20%** Ensemble de validation

**20%** Ensemble de test

Les lots son généré de manière aléatoire à chaque lancement de l'application.

Pour chaque ensemble de test nous avons évalué l'algorithme du KNN et le Perceptron Multicouche avec différentes valeurs d'hyperparamètres.

## 5.1 Ensemble ORL

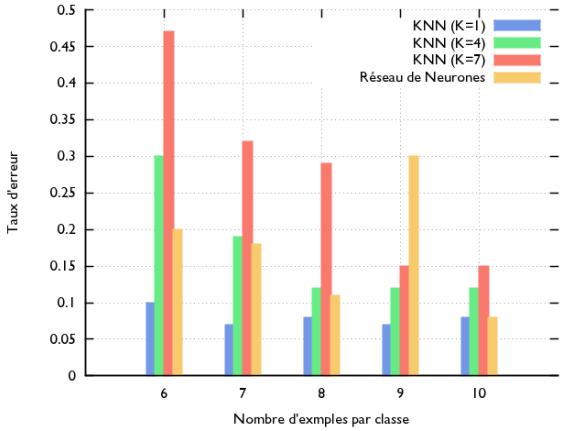



**Figure 6** – Meilleurs résultats pour l'ensemble ORL

Comme le montre la Figure 6 sur le jeu de donnée ORL l'algorithme du KNN est très performant avec pret de 90% de précision pour $K = 1$. En revanche on observe que l'augmentation de nombre de voisins n'est pas très efficase, voir très inefficase pour un faible nombre d'exemples.

Le Réseau de Neurone quand à lui donne des résultats mitigés et parfois aléatoirement très mauvais. Les hyperparamètres sont beaucoup plus difficiles à optimiser car plus nombreux ce qui entraine parfois des résultats abérants. Nous n'avons pas réussit à optenir de bon résultats constants avec cet algorithme sur ce jeu d'entrainement.

En conclusion pour cet ensemble «simple» la technique des K plus proches voisins est de loin la meilleur et la plus stable avec un $K = 1$.

## 5.2 Ensemble LFW

L'ensemble LFW s'est avéré beaucoup plus difficile à appréander. En effet les images sont de moins bonnes qualités, les visages sont souvent de profil et peuvent comporter divers expressions.

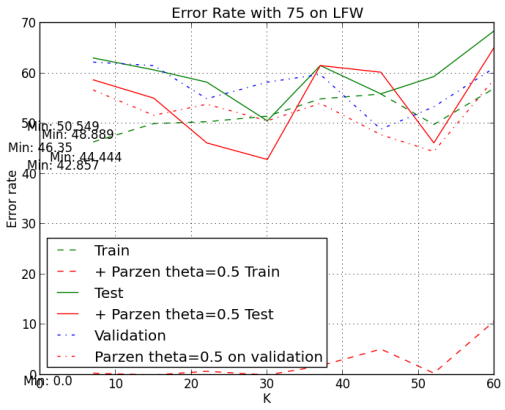



**Figure 7** – Observation des résultats du KNN

Sur cet ensemble on vois clairement (cf. Figure 7) que l'algorithme des K plus proches voisins est totalement inéfficasse. Celui-ci ne s'adapte pas du tout à cet ensemble compliqué à cause de sa faible capacité.

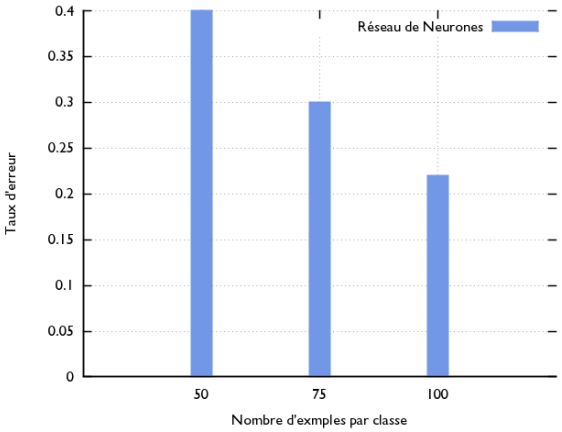



**Figure 8** – Meilleurs résultats pour l'ensemble LFW

Le réseaux de neurones donne des résultats plus prométeurs. Il nécessite en revanche un grand nombre d'exemples par classe pour être efficase et atteindre un niveau de reconnaissance acceptable. Les résultats de la Figure 8 on été obtenus avec pret de 900 neurones dans la couche caché et une moyenne de 300 époques d'entrainement afin d'optimiser les paramètres. Une pénalité de $\lambda \simeq 0.0025$ à aussi été appliqué afin de limiter le surapprentissage. Le taux d'apprentissage quant à lui se trouve au alentoire de 0.0005 pour un mini batch de taille 10.

# 6 Conclusion

Une des caractéristiques de notre projet est qu'il a été réalisé par une équipe de trois personnes ayant suivi des formations différentes. De plus nous avons dû mettre en place un gestionnaire de version (Git) afin de pouvoir travailler efficacement malgré nos diponibilités très disparates.

L'usage du langage Python a aussi ajouté son lot de complications car nous avons tous les trois découvert ce langage au début du semestre.

La dernière difficulté, et non des moindres, concerne le sujet en lui-même. En effet, la reconnaissance faciale est un sujet d'étude actuel et aucun système n'est encore réellement fiable dans ce domaine.

Nous avons cependant obtenu des résultats concluants sur le jeu de données ORL, confirmant l'efficacité de notre programme dans le domaine de la reconnaissance faciale. Ce dernier se montre moins concluant lorsqu'il s'agit d'un jeu de données comportant des portaits moins conventionnels comme le jeu de données LFW contennant nombre d'éléments perturbatoires (lunettes, micros, certaines images présentant aussi plusieurs visages), et marque ainsi une perspective d'améliorations futures.

Notre étude a abouti au fait qu'encore à ce jour la reconnaissance faciale n'est pas un outils fiable et qu'aucune des méthodes présentées dans ce document ne permet de reconnaître de façon assuré une personne parmi d'autres.e

# Références


[1] Turk M. and Pentland A., *Eigenfaces for recognition.* J. Cognitive Neuroscience, 1991.

[2] Samaria F. and Harter A., *Parameterisation of a stochastic model for human face identification.* 2nd IEEE Workshop on Applications of Computer Vision, 1994.

[3] Bradski G., *The OpenCV Library.* Dr. Dobb's Journal of Software Tools, 2000.

[4] Viola P. and Jones M., *Rapid object detection using a boosted cascade of simple features.* Computer Vision and Pattern Recognition, 2001.

[5] Huang G. B., Ramesh M., Berg T. and Learned-Miller E., *Labeled Faces in the Wild : A Database for Studying Face Recognition in Unconstrained Environments.* University of Massachusetts, Amherst, 2007.