INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON

# DEEP: L'apprentissage profond pour l'indexation d'une grande base d'images

novembre 2018
5IF - OT1

Dorian Lefeuvre, Lucas Marie, Fatima-Ezzahra Mezidi, Jiaye Pu

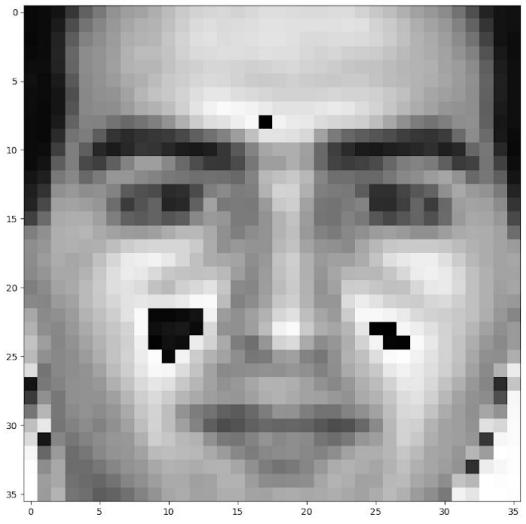
# Introduction

Ce document présente le travail réalisé dans le cadre du projet de conception ”DEEP: L'apprentissage profond pour l'indexation d'une grande base d'images”. Le projet consiste à développer un détecteur de visages basé sur un réseau de neurone à convolution (CNN). Le projet a été développé avec le framework pytorch, installé sans CUDA. Un jeu d'entraînement ainsi qu’un jeu de test ont été fournis. Nous utilisons ces données. Le code source du projet ainsi que les modèles présentés dans ce document sont accessibles sur github à l’adresse suivante : https://github.com/DoLuFaJi/DeepConvNet

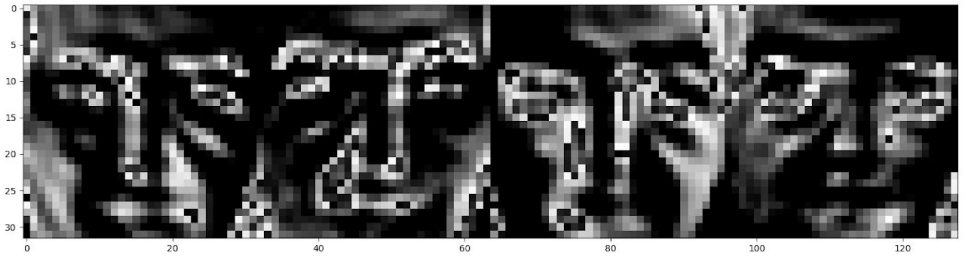
# Architecture du réseau de neurone

## Description

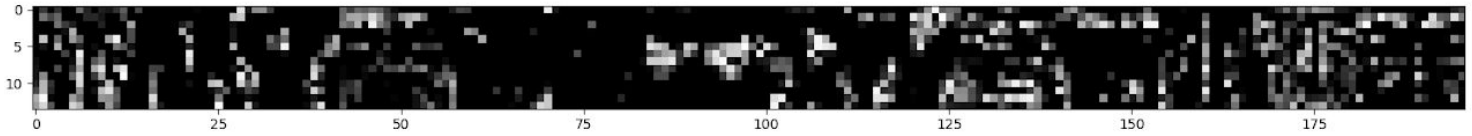
Notre réseau de neurone suit l’architecture suivante :
INPUT -> [CONV -> RELU -> POOL] * 2 -> FC -> RELU -> FC
L’entrée est un tenseur pytorch correspondant à une image en niveaux de gris au format 36x36 pixels.




La première couche de convolution extrait 4 features map. Le pas est de 1, la marge de 0 et les filtres de convolutions sont de 5x5 pixels. Les features maps en sortie ont donc une taille de 32x32 (36-5/1 + 1). Elles passent ensuite par une couche de correction ReLu.



Une couche de pooling applique un filtre de taille 2x2 sur les features afin de réduire leur taille. On obtient des features maps de taille 16x16. La deuxième couche de convolution va extraire 14 features maps. Le pas est de 1, la marge de 0, les filtres sont cette fois-ci de taille 3x3. Les features maps obtenues en sortie ont une taille de 14x14. On applique une couche de correction ReLu, puis une couche de pooling avec un filtre 2x2. On obtient donc 14 features maps de taille 7x7.



La première couche entièrement connectée prend donc 686 (14*7*7) entrées et réduit vers 84 sorties, passant ensuite par une couche ReLu. La seconde couche entièrement connectée réduit de 84 entrées à 2 sorties. Une sortie correspond à une classe, visage ou non-visage.

## Choix

Nous avons choisi de garder une architecture simple afin de prévenir le sur-apprentissage et de ne pas extraire des features trop spécifiques au jeu de données.
L'architecture est inspirée de l'article de C. Garcia et de M. Delakis ainsi que d'exemples provenant de la documentation de pytorch. La taille du premier filtre de convolution nous semble approprié pour la taille des données en entrée. Nous réduisons la taille sur le deuxième filtre, car les données d'entrée sont plus petites.

# Processus d'apprentissage

## Description

Nous avons divisé le jeu de données en deux parties, une d'entrainement et une de validation. Le jeu d'entraînement est composé de 55 719 images dont 39 312 visages, soit 70% du jeu. Le jeu de validation est composé de 36 001 images dont 25 458 visages, soit 70% du jeu.
Chaque itération se déroule en deux phases, l'entraînement et la validation. Lors de la phase d'entraînement le CNN apprend à partir des données d'entraînement. Lors de la phase de validation on teste le CNN, sans qu'il apprenne, sur les données de validation.
Le meilleur modèle, celui avec le taux de précision le plus élevé lors de la phase de validation, est enregistré. Cela nous permet d'avoir, après toutes les itérations, un modèle qui n'a pas sur-appris.

## Choix

La fonction de perte utilisée est la cross-entropy loss car le problème est un problème de classification.
On utilise SGD pour la descente des gradients. Nous avons essayé différents taux d'apprentissage et de momentum. Nous obtenons les meilleurs résultats avec un taux d'apprentissage à 0.01 et un momentum de 0.2.
Les données sont traitées par paquet de 16, c'est une taille assez petite pour converger plus vite et assez grande pour être représentative.
On constate que notre CNN obtient généralement de bons résultats après une douzaine d'itérations. Au-delà de 30 il devient plus rare de voir une amélioration et celle-ci est minime.

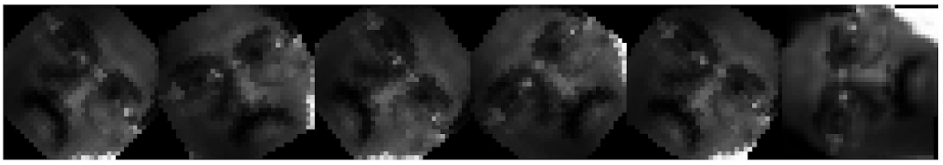
## Augmentation du jeu de données

Pytorch nous permet d'appliquer des transformations sur chaque image permettant de modifier le contenu de celles-ci. Nous utilisons cette option pour augmenter artificiellement la taille de notre jeu de donnée. Cela permet à notre réseau de neurone de mieux généraliser le visage humain et par conséquent d'avoir un taux de détection plus élevé. Nous effectuons 4 transformations, listées ci-dessous :

- Retournement horizontal

- Retournement vertical

- Rotation aléatoire entre + 90 degrés et - 90 degrés par rapport au centre de l'image



- Fluctuation du contraste avec un facteur entre 0.25 et 1.75 et de la luminosité avec un facteur entre 0.5 et 1.5.

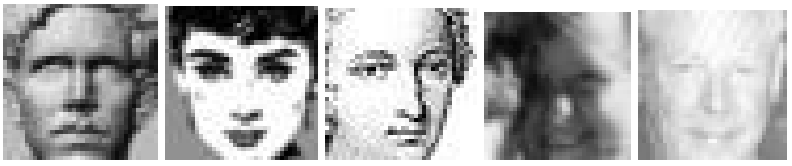
Chaque transformation est appliquée aléatoirement avec un taux de 50%. L'image ci-dessous présente six visages ayant subis six fois les transformations mentionnées plus haut.

# Processus de test

Une fois l'apprentissage terminé nous testons notre réseau de neurone sur plusieurs jeu de données. Le premier test se fait sur l'ensemble d'entraînement et de validation, il nous permet de rapidement évaluer le réseau. Si le résultat n'est pas bon ici, il ne sera pas bon pour les autres.

Le deuxième test se fait sur le jeu de données de Yale, il contient 165 visages et 0 non visage. Les images sont assez uniformes et semblable à celles de notre jeu d'apprentissage. Les visages sont centrés. Il y a quelques variations d'expression, comme des bouches ouvertes ou des clignements d'oeil. Le taux de détection devrait être bon.


Le troisième test se fait sur le jeu de données nommé googlefaces. Il contient 632 visages et 0 non visage. Les visages sont plus variés, pas forcément centré et avec des éclairages différents. Certains visages sont difficiles à détecter pour nous. Enfin on peut débattre sur la présence de certaines images comme un dessin ou une statue en bronze. C'est un jeu de données difficile. Si le taux de détection est bon, cela veut dire que notre réseau de neurone est capable de bien généraliser et de détecter des visages pas facilement détectable.



Le quatrième test se fait sur le jeu de données nommé google_images02. Il contient 6831 non visages et 0 visage. Les images sont variées, certaines sont sombres, d'autres très claires. Nous n'avons pas relevé d'images ressemblant à un visage pour un Homme. Un taux de précision élevé nous permettra d'avoir moins de faux-positif sur le détecteur de visages.

Le cinquième test se réalise à partir de ces trois jeux de données. Il nous permet d'avoir une vision d'ensemble de notre réseau de neurone. Le jeu n'étant pas équilibré il faut prêter une attention relative aux résultats. Par exemple un taux de précision de 90% peut simplement signifier que notre réseau de neurone qualifie tout comme étant un non visage. C'est pour cela que nous avons mis en place les autres tests.

# Détecteur de visages

Le but ici est de détecter plusieurs visages de tailles différentes dans une image, et de les entourer.

## Étapes :

1. Découper l'image **img** en 36*36 pixels avec un pas d'avance de 9 pixels de chaque côté
2. Créer un dataset basé sur ces images, pour ce qui sont prédites comme un visage avec une confiance plus grande que **CONFIANCE** ( = 5.9 ), on les entoure sur l'image finale **imgout**
3. Redimensionner l'image **img** avec une échelle prédéfinie **scale** ( = 1.2 )
4. Aller à l'étape 1 jusqu'à un des deux côtés de l'image **img** est plus petit que **IMG_TAILLE_MIN** ( = 200 ) pixels
5. Afficher l'image finale **imgout** avec les carrés autour du visage

# Résultats

## Jeux de tests

Tous les tests ont été effectué avec des paquets de 16. TRAINSET correspond au premier test décrit plus haut, YALE au deuxième, GOOGLE au troisième, GOOGLE2 au quatrième et TEST au dernier.

### Itérations : 15; Taux apprentissage : 0.01; Momentum : 0.2

TRAINSET: Accuracy of the network: 98 % [90004 / 91720]
TRAINSET: NOT FACE : 95 % [25807 / 26950]
TRAINSET:  FACE : 99 % [64197 / 64770]
YALE: Accuracy of the network: 88 % [146 / 165]
YALE: NOT FACE :  0 % [0 / 0]
YALE:  FACE : 88 % [146 / 165]
GOOGLE: Accuracy of the network: 78 % [495 / 632]
GOOGLE: NOT FACE :  0 % [0 / 0]
GOOGLE:  FACE : 78 % [495 / 632]
GOOGLE2: Accuracy of the network: 97 % [6637 / 6831]
GOOGLE2: NOT FACE : 97 % [6637 / 6831]
GOOGLE2:  FACE :  0 % [0 / 0]
TEST: Accuracy of the network: 95 % [7278 / 7628]
TEST: NOT FACE : 97 % [6637 / 6831]
TEST:  FACE : 80 % [641 / 797]

### Itérations : 30; Taux apprentissage : 0.01; Momentum : 0.2

TRAINSET: Accuracy of the network: 99 % [90822 / 91720]
TRAINSET: NOT FACE : 98 % [26466 / 26950]
TRAINSET:  FACE : 99 % [64356 / 64770]
YALE: Accuracy of the network: 93 % [154 / 165]
YALE: NOT FACE :  0 % [0 / 0]
YALE:  FACE : 93 % [154 / 165]
GOOGLE: Accuracy of the network: 70 % [444 / 632]
GOOGLE: NOT FACE :  0 % [0 / 0]
GOOGLE:  FACE : 70 % [444 / 632]
GOOGLE2: Accuracy of the network: 98 % [6711 / 6831]
GOOGLE2: NOT FACE : 98 % [6711 / 6831]
GOOGLE2:  FACE :  0 % [0 / 0]
TEST: Accuracy of the network: 95 % [7309 / 7628]
TEST: NOT FACE : 98 % [6711 / 6831]
TEST:  FACE : 75 % [598 / 797]

## Itérations : 100; Taux apprentissage : 0.01; Momentum : 0.2

TRAINSET: Accuracy of the network: 99 % [91308 / 91720]
TRAINSET: NOT FACE : 99 % [26683 / 26950]
TRAINSET:  FACE : 99 % [64625 / 64770]
YALE: Accuracy of the network: 96 % [159 / 165]
YALE: NOT FACE :  0 % [0 / 0]
YALE:  FACE : 96 % [159 / 165]
GOOGLE: Accuracy of the network: 69 % [437 / 632]
GOOGLE: NOT FACE :  0 % [0 / 0]
GOOGLE:  FACE : 69 % [437 / 632]
GOOGLE2: Accuracy of the network: 99 % [6767 / 6831]
GOOGLE2: NOT FACE : 99 % [6767 / 6831]
GOOGLE2:  FACE :  0 % [0 / 0]
TEST: Accuracy of the network: 96 % [7363 / 7628]
TEST: NOT FACE : 99 % [6767 / 6831]
TEST:  FACE : 74 % [596 / 797]

## Itérations : 15; Taux apprentissage : 0.01; Momentum : 0.9

TRAINSET: Accuracy of the network: 98 % [90269 / 91720]
TRAINSET: NOT FACE : 97 % [26326 / 26950]
TRAINSET:  FACE : 98 % [63943 / 64770]
YALE: Accuracy of the network: 63 % [104 / 165]
YALE: NOT FACE :  0 % [0 / 0]
YALE:  FACE : 63 % [104 / 165]
GOOGLE: Accuracy of the network: 49 % [315 / 632]
GOOGLE: NOT FACE :  0 % [0 / 0]
GOOGLE:  FACE : 49 % [315 / 632]
GOOGLE2: Accuracy of the network: 98 % [6713 / 6831]
GOOGLE2: NOT FACE : 98 % [6713 / 6831]
GOOGLE2:  FACE :  0 % [0 / 0]
TEST: Accuracy of the network: 93 % [7132 / 7628]
TEST: NOT FACE : 98 % [6713 / 6831]
TEST:  FACE : 52 % [419 / 797]

## Itérations : 30; Taux apprentissage : 0.01; Momentum : 0.9

TRAINSET: Accuracy of the network: 98 % [90409 / 91720]
TRAINSET: NOT FACE : 97 % [26253 / 26950]
TRAINSET:  FACE : 99 % [64156 / 64770]
YALE: Accuracy of the network: 61 % [102 / 165]
YALE: NOT FACE :  0 % [0 / 0]
YALE:  FACE : 61 % [102 / 165]

GOOGLE: Accuracy of the network: 47 % [299 / 632]
GOOGLE: NOT FACE : 0 % [0 / 0]
GOOGLE: FACE : 47 % [299 / 632]
GOOGLE2: Accuracy of the network: 98 % [6707 / 6831]
GOOGLE2: NOT FACE : 98 % [6707 / 6831]
GOOGLE2: FACE : 0 % [0 / 0]
TEST: Accuracy of the network: 93 % [7108 / 7628]
TEST: NOT FACE : 98 % [6707 / 6831]
TEST: FACE : 50 % [401 / 797]

## Itérations : 100; Taux apprentissage : 0.01; Momentum : 0.9

TRAINSET: Accuracy of the network: 99 % [90937 / 91720]
TRAINSET: NOT FACE : 98 % [26588 / 26950]
TRAINSET: FACE : 99 % [64349 / 64770]
YALE: Accuracy of the network: 59 % [98 / 165]
YALE: NOT FACE : 0 % [0 / 0]
YALE: FACE : 59 % [98 / 165]
GOOGLE: Accuracy of the network: 40 % [257 / 632]
GOOGLE: NOT FACE : 0 % [0 / 0]
GOOGLE: FACE : 40 % [257 / 632]
GOOGLE2: Accuracy of the network: 99 % [6767 / 6831]
GOOGLE2: NOT FACE : 99 % [6767 / 6831]
GOOGLE2: FACE : 0 % [0 / 0]
TEST: Accuracy of the network: 93 % [7122 / 7628]
TEST: NOT FACE : 99 % [6767 / 6831]
TEST: FACE : 44 % [355 / 797]

# Détecteur de visages

Nous utilisons le meilleur modèle pour notre détecteur de visages.

Itérations : 100; Taux apprentissage : 0.01; Momentum : 0.2
avec confiance[FACE] - confiance[NO_FACE] > 5.9

Itérations : 100; Taux apprentissage : 0.01; Momentum : 0.2
avec confiance[FACE] - confiance[NO_FACE] > 5.9

Itérations : 100; Taux apprentissage : 0.01; Momentum : 0.2

avec confiance[FACE] - confiance[NO_FACE] > 5.9

Itérations : 100; Taux apprentissage : 0.01; Momentum : 0.2
avec confiance[FACE] - confiance[NO_FACE] > 5.9
et découpe l'image en 18*18 pixels -> convert vers 36*36 pixels -> teste

Itérations : 100; Taux apprentissage : 0.01; Momentum : 0.2
avec confiance[FACE] - confiance[NO_FACE] > 5.9


# Évaluation des résultats

Nos meilleurs résultats sont obtenus avec un momentum de 0.2 et un taux d'apprentissage de 0.01. On remarque qu'au cours de l'apprentissage le taux de détection de visages baisse et que le taux de détection de non-visages augmente. L'apprentissage se fait en grande partie dans les premières itérations avant d'être quasiment stable à la trentième. Comme prévu nous obtenons de bons résultats sur le jeu de test yale et le plus mauvais résultat est celui du jeu de test googlefaces.
Nous sommes satisfaits des résultats obtenus, du taux de détection de non-visages. Nous expliquons le "faible" taux de détection de visages par la difficulté du jeu de test googlefaces. Les résultats obtenus avec le détecteur de visages nous confortent dans cette idée. Cependant celui-ci a été testé avec des photos assez simple, avec des visages en grande partie de face.

# Limites du système

En analysant les faux-négatifs issus du jeu de test on remarque que notre système détecte mal les visages avec une barbe, des lunettes de soleil et une peau trop sombre. Cette dernière peut être dû à la couleur de peau ou à la lumière. Les visages clairs, sur un fond clair ont aussi du mal à être détecté. Enfin les visages éclairés de manière non uniforme sont moins bien détectés également.

En analysant les faux-positifs issus du jeu de test on remarque qu'ils ont tous une grande tâche blanche au milieu. Cependant nous n'avons pas analysés l'ensemble des images du set google_images02, ils présentent peut être tous cette même caractéristique. Notre taux de faux-positifs étant relativement bas, nous n'avons pas prêté une grande attention à ceux-ci.

En analysant les images issues du détecteur on remarque que les caractères et les chaussures sont souvent détectées comme des visages.

# Amélioration

## Élargissement du jeu de données d'apprentissage

Il est encore possible d'élargir le jeu de données, afin d'améliorer la détection de visages. On peut ajouter des transformations pour continuer à augmenter artificiellement la taille du jeu de données. Par exemple une transformation pourrait ajouter des carrés noirs de manière aléatoire sur le visage, pour simuler une paire de lunette de soleil, une barbe ou des cheveux ou sourcils volumineux. Une transformation plus simple serait de réaliser une translation de quelques pixels dans une direction aléatoire. Cela permettra à notre réseau de mieux détecter les visages coupés sur une photo.
On peut également élargir le jeu de données en ajoutant une base de visages et de non-visages ou en utilisant les faux-positifs et les faux-négatifs des tests.

## Architecture du réseau de neurones

Une autre possibilité pour améliorer la précision serait de revoir l'architecture du réseau et de faire plusieurs tests afin de trouver la meilleure. On peut notamment revoir le nombre de features maps à extraire après chaque convolution. Ou bien implémenter une architecture d'un autre type comme : INPUT -> [CONV -> RELU -> CONV -> RELU -> POOL] * 3 -> [FC -> RELU] * 2 -> FC.

# Conclusion

Ce projet nous a permis de découvrir et comprendre les réseaux de neurones à convolution. Nous avons également pu rencontrer tous les problèmes types d'un projet "deep learning" tel que l'importance d'un bon jeu d'entrainement, de validation et de test, le réglage de paramètre et éviter le sur-apprentissage. Cela nous permettra d'être prêt et de gagner du temps si l'on doit travailler sur un projet similaire plus tard. C'était un projet très enrichissant.