



Compte Rendu Projet

DETECTION EMOTION DANS UN VISAGE

Realisé Par :

El Mostafa FADILI

Hamza EL GATIA

Moad ELKHAYATI

Oussama ANOUZID

Table des matières

0.1	Les utilisations de la reconnaissance des émotions	3
0.2	Quels sont les avantages de la reconnaissance des émotions ?	5
1	Dataset : FER-2013	6
1.1	Description	6
2	Détection de Visage Avec OpenCV	9
2.1	Detection visage avec haarcascade	9
3	Détection de Visage Avec Deep Learning	14
3.1	Module MTCNN	14
3.2	Exemple de detection visage avec MTCNN	15
3.3	Exemple de detection visage avec mediapipe	18
3.3.1	Les solutions proposé par mediapipe	18
3.3.2	Application du methode Face Mesh	19
3.4	Extract des visages	20
4	Reconnaissance des Emotions Faciales	23
4.1	Reconnaissance des Emotions Faciales avec Landmarks	23
4.2	Methode linéaire : KNN	24
4.2.1	KNN avec Landmarks	25
4.3	Deep Learning	25
4.3.1	Définition et Généralité	25
4.3.2	Historique	26
4.3.3	Applications du Deep Learning	27
4.3.4	Réseau de neurones artificiels	28
4.3.5	Réseaux de neurones à convolution	29
4.4	Data augmentation	32
4.4.1	Pourquoi le Data augmentation	32

4.4.2	Exemples du Data augmentation	33
5	Conclusion	38

Introduction

La reconnaissance des émotions est une méthode utilisée dans un logiciel qui permet à un programme d'« examiner » les sentiments d'un visage humain en utilisant une distribution d'images sophistiquée. Les entreprises ont testé une fusion de formules avancées avec des pratiques de traitement d'images qui se sont matérialisées au cours de la dernière décennie pour mieux comprendre ce qu'une vidéo ou une image du visage d'un individu nous dit sur ce qu'il ressent. Avec l'innovation actuelle, le logiciel d'identification des émotions s'est développé très habilement. De plus, son aptitude à suivre les premiers regards du visage pour des émotions comme le bonheur, la tristesse, la surprise, la colère, etc. sentiments dépourvus de leur connaissance.

La reconnaissance des émotions concorde également avec d'autres types de technologies de reconnaissance faciale et l'identification d'images biométriques. Ces deux types de technologies peuvent être appliqués dans de nombreux types de cas de sécurité. Par exemple, les autorités peuvent utiliser un logiciel de reconnaissance des émotions pour approfondir les efforts d'enquête concernant quelqu'un à un moment donné d'un entretien ou d'un interrogatoire. La détection des émotions continue d'avancer à égalité avec d'autres innovations telles que le traitement du langage naturel et ces signes de progrès sont pour la plupart rendus probables par le mélange de processeurs de plus en plus dominants, la croissance scientifique d'algorithmes complexes et d'autres technologies associées.

0.1 Les utilisations de la reconnaissance des émotions

La reconnaissance des émotions est déjà largement utilisée par différentes entreprises pour évaluer l'humeur des consommateurs envers leur produit ou

leur marque. Les opportunités apportées par cette technologie vont au-delà des études de marché et de la publicité numérique.

— **Reconnaissance des émotions dans les soins de santé :**

une industrie qui tire parti de cette technologie est celle des soins de santé, avec un logiciel de reconnaissance basé sur l'IA aidant à décider quand les patients ont besoin de médicaments ou à aider les médecins à déterminer qui consulter en premier.

— **Industrie automobile et reconnaissance des émotions :**

l'industrie automobile applique également la technologie de reconnaissance des émotions, car les constructeurs automobiles du monde entier se concentrent de plus en plus sur la fabrication de voitures plus personnelles et plus sûres pour les personnes à conduire. Ce dernier est un domaine qui attire principalement l'attention et que diverses entreprises ont déjà pris des mesures pour tester et rechercher. Dans leur quête pour créer plus de fonctionnalités de voitures intelligentes, il est logique que les constructeurs automobiles utilisent l'IA pour les aider à comprendre les émotions humaines. En utilisant la détection des émotions faciales, les voitures intelligentes peuvent alerter le conducteur lorsqu'il se sent somnolent.

— **Reconnaissance des émotions dans les tests de jeux vidéo :**

les jeux vidéo sont conçus pour un public cible spécifique et visent à évoquer un comportement et un ensemble d'émotions particuliers de la part des utilisateurs. Au cours de la phase de test, les utilisateurs sont invités à jouer au jeu pendant une période donnée et leurs commentaires sont intégrés pour créer le produit final. L'utilisation de la reconnaissance des émotions faciales peut aider à comprendre quelles émotions un utilisateur ressent en temps réel pendant qu'il joue sans analyser la vidéo complète manuellement.

0.2 Quels sont les avantages de la reconnaissance des émotions ?

Détecter les émotions avec la technologie est une tâche assez difficile, mais où les algorithmes d'apprentissage automatique se sont révélés très prometteurs. En utilisant la reconnaissance faciale des émotions , les entreprises peuvent traiter des images et des vidéos en temps réel pour surveiller les flux vidéo ou automatiser l'analyse vidéo, réduisant ainsi les coûts et améliorant la vie de leurs utilisateurs.

Chez Sightcorp, nous avons combiné la science de la psychologie, des expressions humaines et de l'intelligence artificielle pour reconnaître automatiquement différentes émotions sur le visage d'un individu. Notre algorithme d' analyse faciale peut identifier sept types différents d'états émotionnels en temps réel : bonheur, tristesse, dégoût, surprise, colère et peur.

Chapitre 1

Dataset : FER-2013

Il s'agit une compétition sur [kaggle](#), qui a été publié sur *International Conference on Machine Learning(ICML)* il y a 8 ans, pour apprendre les expressions faciales à partir d'une image.

1.1 Description

- **test**
 - angry
 - disgust
 - fear
 - happy
 - neutral
 - sad
 - surprise
- **train**

Les données consistent en des images en niveaux de gris de 48 x 48 pixels de visages.

Les visages ont été automatiquement enregistrés afin que le visage soit plus ou moins centré et occupe à peu près la même quantité d'espace dans chaque image.

La tâche consiste à classer chaque visage en fonction de l'émotion montrée dans l'expression du visage dans l'une des sept catégories (0 = en colère, 1 = dégoût, 2 = peur, 3 = heureux, 4 = triste, 5 = surprise, 6 = neutre).

Étiquettes d'émotion dans l'ensemble de données :

- 0 : -4593 images- En colère.
- 1 : -547 images- Dégoût.
- 2 : -5121 images- Peur.
- 3 : -8989 images- Heureux.
- 4 : -6077 images- Triste.



FIGURE 1.1 – Exemples d’images de l’ensemble de données FER-2013

— 5 : -4002 images- Surprise.

— 6 : -6198 images- Neutre

L'ensemble de formation se compose de 28 709 exemples et l'ensemble de test public se compose de 3 589 exemples.

Ceci lien du competition : [FER-2013](#).

Chapitre 2

Détection de Visage Avec OpenCV

Les algorithmes de détection de visage basés sur des fonctionnalités sont rapides et efficaces et sont utilisés avec succès depuis des décennies.

L'exemple le plus réussi est peut-être une technique appelée classificateurs en cascade décrite pour la première fois par Paul Viola et Michael Jones et leur article de 2001 intitulé **Détection rapide d'objets à l'aide d'une cascade renforcée de fonctionnalités simples**.



Paul Viola



Michael Jones

Dans l'article, les fonctionnalités efficaces sont apprises à l'aide de l'algorithme **AdaBoost**, bien que plusieurs modèles soient organisés en hiérarchie ou en **cascade**.

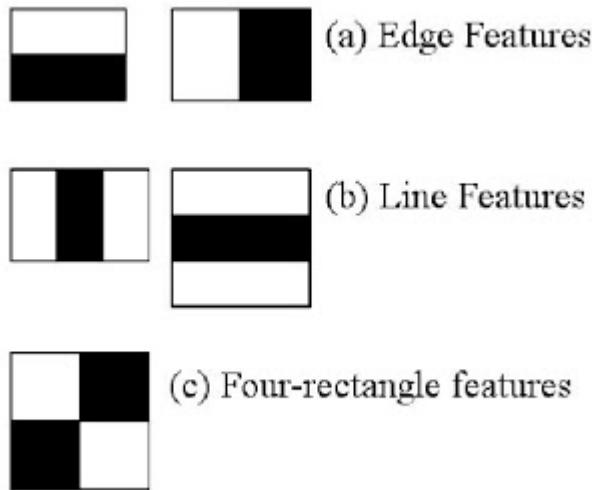
Dans l'article, le modèle AdaBoost est utilisé pour apprendre une gamme de caractéristiques très simples ou faibles dans chaque face, qui ensemble fournissent un classificateur robuste.

2.1 Detection visage avec haarcascade

La détection d'objets à l'aide de classificateurs en cascade basés sur les caractéristiques Haar est une méthode efficace proposée par Paul Viola

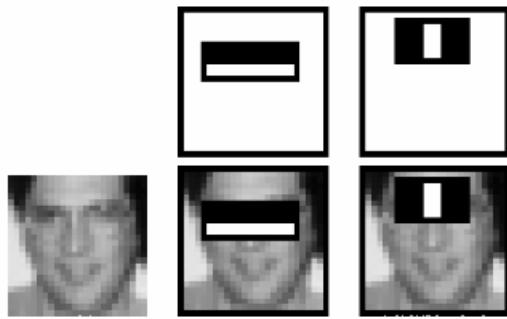
et Michael Jones dans l'article de 2001, "Détection rapide d'objets à l'aide d'une cascade améliorée de caractéristiques simples". Il s'agit d'une approche basée sur l'apprentissage automatique dans laquelle une fonction en cascade est formée à partir d'un grand nombre d'images positives et négatives. Il est ensuite utilisé pour détecter des objets dans d'autres images.

Ici, nous allons travailler avec la détection de visage. Au départ, l'algorithme a besoin de beaucoup d'images positives (images de visages) et d'images négatives (images sans visages) pour entraîner le classificateur. Ensuite, nous devons extraire des fonctionnalités. Pour cela, les fonctionnalités Haar présentées dans l'image ci-dessous sont utilisées. Ils sont comme notre noyau convolutif. Chaque caractéristique est une valeur unique obtenue en soustrayant la somme des pixels sous le rectangle blanc de la somme des pixels sous le rectangle noir.



Désormais, toutes les tailles et emplacements possibles de chaque noyau sont utilisés pour calculer de nombreuses fonctionnalités. Pour chaque calcul d'entité, nous devons trouver la somme des pixels sous les rectangles blanc et noir. Pour résoudre ce problème, ils ont introduit les images intégrales. Il simplifie le calcul de la somme des pixels, quelle peut être la taille du nombre de pixels, à une opération impliquant seulement quatre pixels.

Mais parmi toutes ces caractéristiques que nous avons calculées, la plupart d'entre elles ne sont pas pertinentes. Par exemple, considérez l'image ci-dessous. La rangée du haut montre deux bonnes fonctionnalités. La première caractéristique sélectionnée semble se concentrer sur la propriété que la région des yeux est souvent plus sombre que la région du nez et des joues. La deuxième caractéristique sélectionnée repose sur la propriété que les yeux sont plus foncés que l'arête du nez. Mais les mêmes fenêtres qui s'appliquent sur les joues ou à tout autre endroit ne sont pas pertinentes. Alors, comment sélectionner les meilleures fonctionnalités parmi plus de 160000 fonctionnalités ? Il est réalisé par Adaboost.



Pour cela, nous appliquons chaque fonctionnalité sur toutes les images de formation. Pour chaque fonction, il trouve le meilleur seuil qui classera les faces en positif et en négatif. Mais évidemment, il y aura des erreurs ou des classifications erronées. Nous sélectionnons les fonctionnalités avec un taux d'erreur minimum, ce qui signifie que ce sont les fonctionnalités qui classent le mieux les images faciales et non faciales. (Le processus n'est pas aussi simple que celui-ci. Chaque image reçoit un poids égal au début. Après chaque classification, les poids des images mal classées sont augmentés. Ensuite, le même processus est effectué. De nouveaux taux d'erreur sont calculés. De nouveaux poids sont également ajoutés. Le processus se poursuit jusqu'à ce que la précision ou le taux d'erreur requis soit atteint ou que le nombre requis de caractéristiques soit trouvé).

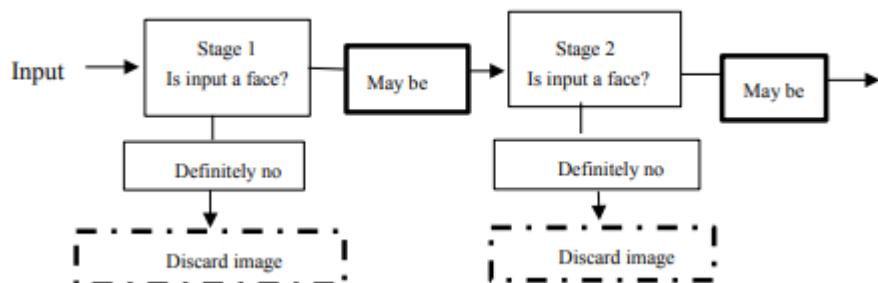
Le classificateur final est une somme pondérée de ces classificateurs faibles. Il est appelé faible parce qu'il ne peut pas à lui seul classer l'image, mais avec d'autres, il forme un classificateur puissant. Le document indique que même 200 fonctionnalités fournissent une détection avec une précision de

0.95. Leur configuration finale avait environ 6000 fonctionnalités. (Imaginez une réduction de 160000+ fonctionnalités à 6000 fonctionnalités. C'est un gros gain).

Alors maintenant, vous prenez une image. Prenez chaque fenêtre 24x24. Appliquez-lui 6000 fonctionnalités. Vérifiez si c'est le visage ou non. Wow... Wow... N'est-ce pas un peu inefficace et chronophage ? Oui c'est le cas. Les auteurs ont une bonne solution pour cela.

Dans une image, la majeure partie de la région de l'image est une région sans visage. Il est donc préférable d'avoir une méthode simple pour vérifier si une fenêtre n'est pas une région de visage. Si ce n'est pas le cas, jetez-le en un seul coup. Ne le traitez plus. Concentrez-vous plutôt sur la région où il peut y avoir un visage. De cette façon, nous pouvons trouver plus de temps pour vérifier une région de visage possible.

Pour cela, ils ont introduit le concept de cascade de classificateurs. Au lieu d'appliquer toutes les 6000 fonctionnalités sur une fenêtre, regroupez les fonctionnalités en différentes étapes de classificateurs et appliquez-les une par une. (Normalement, les premières étapes contiendront très moins de fonctionnalités). Si une fenêtre échoue à la première étape, supprimez-la. Nous ne considérons pas les fonctionnalités restantes dessus. S'il réussit, appliquez la deuxième étape des fonctionnalités et poursuivez le processus. La fenêtre qui passe toutes les étapes est une région de visage. Comment est le plan !!!



Le détecteur des auteurs avait plus de 6000 fonctionnalités avec 38 étapes avec 1, 10, 25, 25 et 50 caractéristiques dans les cinq premières étapes. (Deux fonctionnalités dans l'image ci-dessus sont en fait obtenues comme les deux meilleures fonctionnalités d'Adaboost). Selon les auteurs, en moyenne,

10 fonctionnalités sur plus de 6000 sont évaluées par sous-fenêtre.

Voici donc une explication simple et intuitive du fonctionnement de la détection de visage Viola-Jones. Lisez l'article pour plus de détails.

Chapitre 3

Détection de Visage Avec Deep Learning

Un certain nombre de méthodes d'apprentissage en profondeur ont été développées et démontrées pour la détection des visages.

L'une des approches les plus populaires s'appelle peut-être le **réseau neuronal convolutif en cascade multitâche**, ou **MTCNN** en abrégé, décrit par *Kaipeng Zhang* et al. dans l'article de 2016 intitulé **Face Detection and Alignment Using Multitask Cascaded Convolutional Networks** ([lien](#)).



3.1 Module MTCNN

Le **MTCNN** est populaire car il a obtenu des résultats à la pointe de la technologie sur une gamme d'ensembles de données de référence, et parce qu'il est également capable de reconnaître d'autres caractéristiques faciales telles que les yeux et la bouche, appelée détection de point de repère. Le réseau

utilise une structure en cascade avec trois réseaux ; l'image est d'abord redimensionnée à une gamme de tailles différentes (appelée pyramide d'images), puis le premier modèle (Proposal Network ou P-Net) propose des régions

faciales candidates, le second modèle (Refine Network ou R-Net) filtre les cadres de délimitation , et le troisième modèle (Output Network ou O-Net) propose des repères faciaux.

3.2 Exemple de detection visage avec MTCNN

```
# Importing libraries
from mtcnn.mtcnn import MTCNN
import cv2

# Read an image
image = cv2.imread('test1.jpg')

# display the image
cv2.imshow('test1.jpg')
```



```
# create mtcnn model detector
detector = MTCNN()
```

```
# predict faces
faces = detector.detect_faces(image)
for face in faces:
    print(face)
```

Une fois le modèle est chargé, il peut être utilisé directement pour détecter des visages sur des photographies en appelant la fonction `detect_faces()`.

Cela renvoie une liste d'objets dict, chacun fournissant un certain nombre de clés pour les détails de chaque visage détecté, notamment :

- **boîte** : Fournir le x, y du coin inférieur gauche de la boîte englobante, ainsi que la largeur et la hauteur de la boîte.
- **confiance** : la probabilité de confiance de la prédiction.
- **keypoints** : Fournir un dict avec des points pour le *left_eye*, *right_eye*, *nose*, *mouth_left* et *mouth_right*.

L'exécution de l'exemple charge la photographie, charge le modèle, effectue une détection de visage et imprime une liste de chaque visage détecté.

```
{
    'box': [186, 71, 92, 115],
    'confidence': 0.9999665021896362,
    'keypoints': {
        'left_eye': (209, 112),
        'right_eye': (252, 120),
        'nose': (221, 142),
        'mouth_left': (203, 152),
        'mouth_right': (244, 160)
    }
}
{
    'box': [371, 86, 94, 124],
    'confidence': 0.9582215547561646,
    'keypoints': {
        'left_eye': (392, 131),
        'right_eye': (437, 137),
        'nose': (406, 164),
        'mouth_left': (389, 176),
        'mouth_right': (434, 181)
    }
}
```

```

def draw_boxes(image, multi_faces=False):
    faces = detector.detect_faces(image)
    for face in faces:
        # get face landmarks coordinates
        x, y, width, height = face['box']
        keypoints = face['keypoints']

        # Draw a face box
        cv2.rectangle(image,
                      (x, y),
                      (x+width, y+height),
                      (0,0,255),
                      2)
        # Draw circles for the eyes, nose, and mouth
        cv2.circle(image, (keypoints['left_eye']), 2, (0,255,0),2)
        cv2.circle(image, (keypoints['right_eye']), 2, (0,255,0),2)
        cv2.circle(image, (keypoints['nose']), 2, (0,255,0),2)
        cv2.circle(image, (keypoints['mouth_left']), 2, (0,255,0),2)
        cv2.circle(image, (keypoints['mouth_right']), 2, (0,255,0),2)

    if not multi_faces:
        break

return image

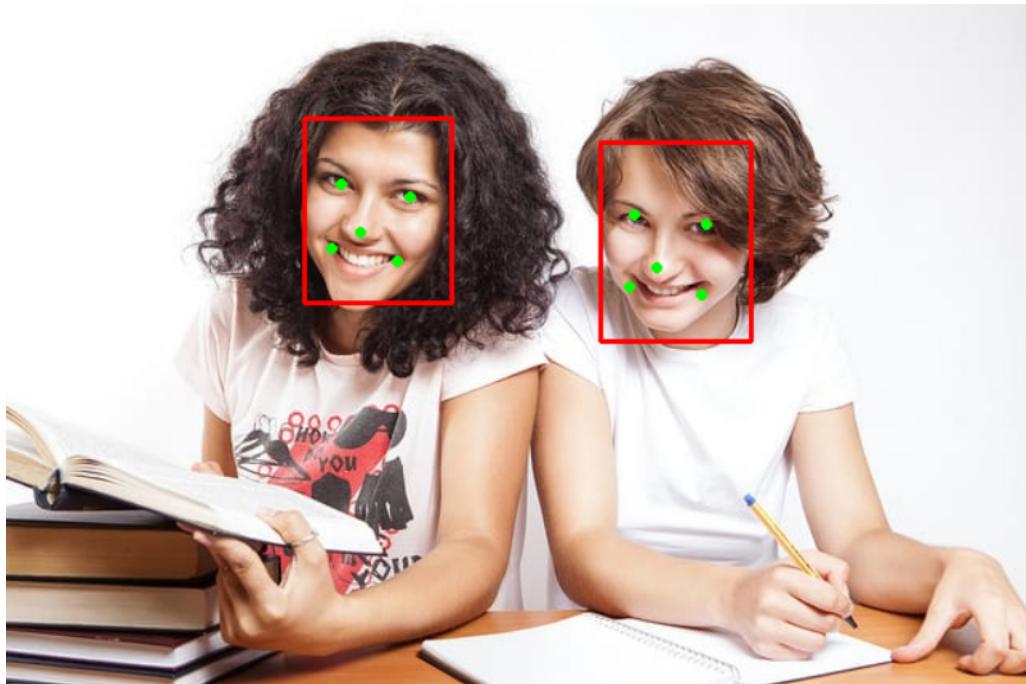
marked_image = draw_boxes(image)

cv2.imshow(marked_image)

```

L'exécution de l'exemple trace la photographie puis dessine un cadre de délimitation pour chacun des visages détectés.

Nous pouvons voir que les deux visages ont été détectés correctement.



3.3 Exemple de detection visage avec mediapipe

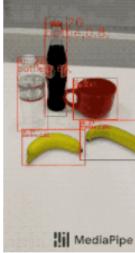
Mediapipe est un framework concus par [Google AI](#) pour la perception simultanée en temps réel de la pose humaine, des repères du visage et du suivi des mains sur les appareils mobiles peut permettre une variété d'applications percutantes, telles que l'analyse de la condition physique et du sport, le contrôle des gestes et la reconnaissance de la langue des signes, les effets de réalité augmentée, etc.



3.3.1 Les solutions proposé par mediapipe

Mediapipe propose plusieurs solutions pour les problèmes de machine learning dans les applications mobiles ou web avec une performance sur le temps et

de qualité.

Face Detection	Face Mesh	Iris	Hands	Pose	Holistic
					
Hair Segmentation	Object Detection	Box Tracking	Instant Motion Tracking	Objectron	KNIFT
					

Pour notre application nous allons utiliser la méthode pour détection **Face Mesh**

3.3.2 Application du méthode Face Mesh

La méthode **Face Mesh** cherche dans une image les points essentiels dans une image(448points) qui détermine tous les détails de profondeur et détection des yeux, du nez, de la bouche.

- Exemple d'application du méthode de Face Mesh.



3.4 Extract des visages

Nous pouvons vouloir extraire les visages détectés et les transmettre en entrée à un autre système.

Ceci peut être réalisé en extrayant les données de pixels directement de la photographie ; par exemple :

```
from PIL import Image
from numpy import asarray

def extract_face(filename, required_size=(224,224)):
    # load image from the file
    image = cv2.imread(filename)
    # create detector
    detector = MTCNN()
    # detect faces in the image
    faces = detector.detect_faces(image)

    # extract the bounding box from the first face
    x1, y1, width, height = faces[0]['box']
    x2, y2 = x1+width, y1+height

    # extract the face
    face = image[y1:y2, x1:x2]
```

```

# resize pixels to the model size
face_img = Image.fromarray(face)
face_img = face_img.resize(required_size)
face_array = asarray(face_img)
return face_array

face = extract_face('test1.jpg')
cv2_imshow(face)

```



Nous pouvons le démontrer en extrayant chaque face et en les traçant sous forme de sous-parcelles séparées. Nous pouvez tout aussi bien les enregistrer dans un fichier. **Le draw_faces()** cidessous extrait et trace chaque visage détecté dans une image.

```

import matplotlib.pyplot as plt
def extract_faces(filename, required_size=(224,224)):
    # load image from the file
    image = plt.imread(filename)
    # create detector
    detector = MTCNN()
    # detecte faces in the image
    faces = detector.detect_faces(image)

    # iterate on each face
    for i in range(len(faces)):
        # extract the bounding box from the first face
        x1, y1, width, height = faces[i]['box']
        x2, y2 = x1+width, y1+height

```

```
# extract the face
face = image[y1:y2, x1:x2]

# define subplot
plt.subplot(1, len(faces), i+1)
plt.axis('off')

# plot face
plt.imshow(face)
```



Chapitre 4

Reconnaissance des Emotions Faciales

Nous avons utilisé deux méthodes pour la détection des émotions :

- Reconnaissance des Emotions Faciales avec **CNN**
- Reconnaissance des Emotions Faciales avec **Landmarks**

Les trois principaux composants de la détection des émotions sont les suivants :

1. Prétraitement d'image
2. Extraction de caractéristiques
3. Classification des caractéristiques

4.1 Reconnaissance des Emotions Faciales avec Landmarks

Les étapes de détection des émotions avec la méthode de landmarks et CNN.



- Detection de visage :
La détection de visage est une étape importante dans la détection des émotions. Il supprime les parties de l'image qui ne sont pas pertinentes.
- Landmarks de visage :
Landmarks de visage sont un ensemble de points clés sur les images de visage humain. Les points sont définis par leurs coordonnées (x,y) sur l'image. Ces points sont utilisés pour localiser et représenter les régions saillantes du visage, telles que les yeux, les sourcils, le nez, la bouche et la mâchoire.
Le modèle de repère facial que nous avons utilisé était le modèle de détection de repère facial pré-entraîné de Dlib qui détecte 68 points bidimensionnels sur le visage humain.
- Classificateur de Machine learning : On trouve alors la distance euclidienne entre toutes les combinaisons des 68 points de repère, ce qui nous donne $68 \times 68 = 4624$ distances. Nous ne savons peut-être pas lesquelles de ces distances sont les plus importantes, nous laissons donc les models d'apprentissage le comprendre.
Il'y a deux methods de machine learning de classifications :
 - Methode linéaire : **Le plus proche voisin(KNN)**
 - Methode Non linéaire : **CNN**

4.2 Methode linéaire : KNN

Nous avons test classification des images avec l'algorithme de plus proche voisin, qui est un algorithme de classification supervisé.

Nous avons lui donné les images sous forme d'une ligne de pixels dans une images avec les label de emotions. Les resultat qu'on a trouver on les resume dans une matrice de confusion et le facteur de précision :

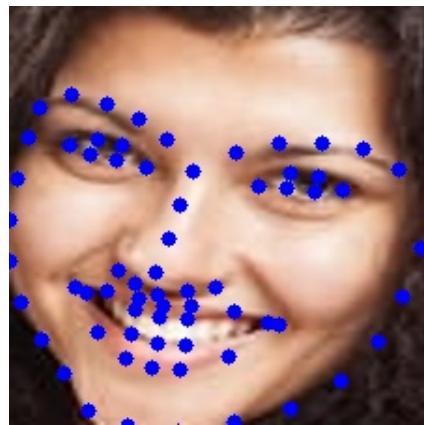
- matrice de confusion :

	0	1	2	3	4	5	6
0	196	11	60	94	39	16	82
1	9	25	4	5	4	1	4
2	92	10	207	83	57	34	62
3	122	28	139	361	60	28	143
4	109	11	84	107	164	16	97
5	69	7	49	80	21	155	33
6	89	17	84	135	47	21	218

— precision : **36.946224575090554**

4.2.1 KNN avec Landmarks

Pour la detection des landmarks on a utiliser le module **Dlib** qui detecte 68 points de landmarks sur le visage.



Ensuite on calcule les distances entre tout les points, donc on aura 4624 distance.

4.3 Deep Learning

4.3.1 Définition et Généralité

L'apprentissage profond, ou apprentissage en profondeur (en anglais : deep learning, deep structured learning, hierarchical learning) est un ensemble de méthodes d'apprentissage automatique tentant de modéliser avec

un haut niveau d'abstraction des données grâce à des architectures articulées de différentes transformations non linéaires. Ces techniques ont permis des progrès importants et rapides dans les domaines de l'analyse du signal sonore ou visuel et notamment de la reconnaissance faciale, de la reconnaissance vocale, de la vision par ordinateur, du traitement automatisé du langage. Dans les années 2000, ces progrès ont suscité des investissements privés, universitaires et publics importants, notamment de la part des GAFAM (Google, Apple, Facebook, Amazon, Microsoft).

Les techniques d'apprentissage profond constituent une classe d'algorithmes d'apprentissage automatique qui :

Utilisent différentes couches d'unité de traitement non linéaire pour l'extraction et la transformation des caractéristiques ; chaque couche prend en entrée la sortie de la précédente ; les algorithmes peuvent être supervisés ou non supervisés, et leurs applications comprennent la reconnaissance de modèles et la classification statistique ; Fonctionnent avec un apprentissage à plusieurs niveaux de détail ou de représentation des données ; à travers les différentes couches, on passe de paramètres de bas niveau à des paramètres de plus haut niveau, où les différents niveaux correspondent à différents niveaux d'abstraction des données. Ces architectures permettent aujourd'hui de conférer du « sens » à des données en leur donnant la forme d'images, de sons ou de textes.

L'apprentissage profond utilise des couches cachées de réseaux de neurones artificiels, des « machines de Boltzmann restreintes », et des séries de calculs propositionnels complexes. Les algorithmes d'apprentissage profond s'opposent aux algorithmes d'apprentissage peu profonds du fait du nombre de transformations réalisées sur les données entre la couche d'entrée et la couche de sortie, où une transformation correspond à une unité de traitement définie par des poids et des seuils.

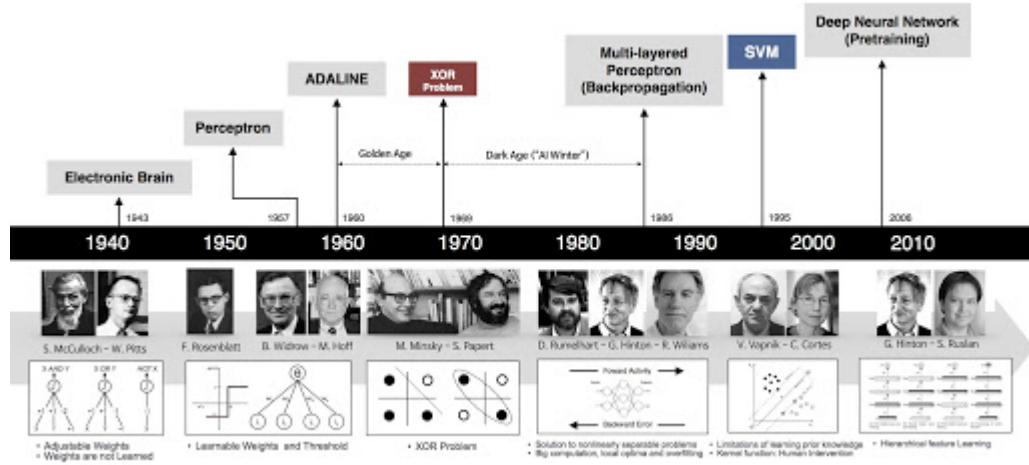
4.3.2 Historique

Le concept d'apprentissage profond prend forme dans les années 2010, avec la convergence de quatre facteurs :

- Des réseaux de neurones artificiels multicouches (eux-mêmes issus entre autres du concept de perceptron, datant de la fin des années 1950) ;
- Des algorithmes d'analyse discriminante et apprenants (dont l'émergence remonte aux années 1980) ;

- Des machines dont la puissance de traitement permet de traiter des données massives ;
- Des bases de données suffisamment grandes, capables d'entraîner des systèmes de grandes tailles.

En octobre 2015, le programme AlphaGo, à qui l'on a « appris » à jouer au jeu de go grâce à la méthode de l'apprentissage profond, bat le champion européen Fan Hui par 5 parties à 0. En mars 2016, le même programme bat le champion du monde Lee Sedol par 4 parties à 1.



4.3.3 Applications du Deep Learning

Le deep Learning est utilisé dans de nombreux domaines :

- reconnaissance d'image
- traduction automatique
- voiture autonome
- diagnostic médical
- recommandations personnalisées
- modération automatique des réseaux sociaux
- prédiction financière et trading automatisé
- identification de pièces défectueuses
- détection de malwares ou de fraudes
- chatbots (agents conversationnels)
- exploration spatiale
- robots intelligents

C'est aussi grâce au deep Learning que l'intelligence artificielle de Google

Alpha Go a réussi à battre les meilleurs champions de Go en 2016. Le moteur de recherche du géant américain est lui-même de plus en plus basé sur l'apprentissage par deep Learning plutôt que sur des règles écrites.

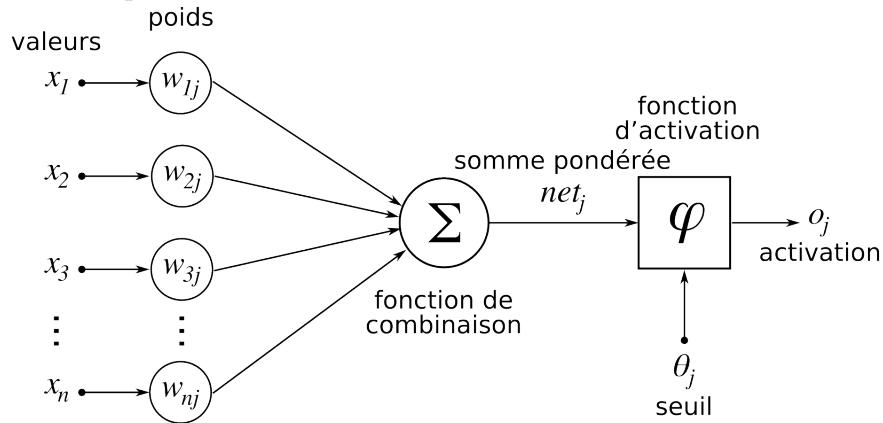
4.3.4 Réseau de neurones artificiels

Un réseau de neurones artificiels, ou réseau neuronal artificiel, est un système dont la conception est à l'origine schématiquement inspirée du fonctionnement des neurones biologiques, et qui par la suite s'est rapproché des méthodes statistiques.

Les réseaux de neurones sont généralement optimisés par des méthodes d'apprentissage de type probabiliste, en particulier bayésien. Ils sont placés d'une part dans la famille des applications statistiques, qu'ils enrichissent avec un ensemble de paradigmes permettant de créer des classifications rapides , et d'autre part dans la famille des méthodes de l'intelligence artificielle auxquelles ils fournissent un mécanisme perceptif indépendant des idées propres de l'implémenteur, et des informations d'entrée au raisonnement logique formel (voir Deep Learning).

Un Réseau de neurones artificiels se caractérise par :

- Une fonction d'activation(relu,softmax,tanh...)
- Une méthode pour la modification des weights w_i (gradient descent,stochastic gradient descent...)
- une loss function (pour la comparaison des résultats .Ex : $(y - y_{pred})^2$)
- Le bon traitement de la matrice des données
- La précision du nombre des couches, ainsi que le nombre des neurones a chaque couche.



4.3.5 Réseaux de neurones à convolution

Motivation

Les réseaux de neurones à convolution sont les véritables superstars des réseaux de neurones. Ces réseaux sont capables d'accomplir des tâches relativement complexes en exploitant des données de type image, son, texte, vidéo etc. Les premiers réseaux de convolution à succès ont été mis au point à la fin des années 90 par le professeur Yann LeCunn pour les laboratoires Bell.

Dans les problèmes de classification style MNIST, on peut utiliser, des perceptrons multicouches qui fournissent de très bons résultats et le temps d'entraînement reste raisonnable. Cependant pour des bases de données plus importantes avec des images de plus grande taille, le nombre de paramètres du réseau croît très rapidement, ce qui rend l'entraînement plus long, et dégrade les performances du modèle.

Limitations des perceptrons multicouches

Les perceptrons multicouches sont redoutablement efficaces pour résoudre plusieurs catégories de problème : classification, estimation, approximation, reconnaissance etc. Cependant, ces derniers peuvent rapidement s'avérer limités, en particulier avec des données de très grande dimension.

Exemple :

On veut créer un réseau de neurones pour réaliser une tâche de classification de type chien ou chat. Le modèle est entraîné sur une base de données d'images de taille 420x420px grayscale ; le but étant qu'après entraînement , le modèle soit capable à partir d'une image de dire s'il s'agit d'un chien ou d'un chat. Une première approche serait d'utiliser un perceptron multicouche avec une couche d'entrée comportant par exemple 128 neurones. Puisque chacun des neurones reçoit en entrée les 420×420 pixels et attribue un poids par pixel, on a par neurone $420 \times 420 = 176400$ poids, multiplié par les 128 neurones auxquels s'ajoutent finalement 128 biais ; un biais par neurone. Notre tout petit modèle devra donc apprendre plus de 22 millions de paramètres. Cela pour seulement 128 neurones et une couche intermédiaire. Si on veut un modèle performant, il faudra plusieurs couches supplémentaires et plus de neurones par couche. Le nombre de paramètres explose. Par exemple pour un modèle 3 couches (128 - 256 - 1) on a presque 23 millions de paramètres. Un tel modèle nécessite une énorme quantité de données pour être entraîné, et puisque les paramètres à ajuster sont beaucoup plus nombreux, l'entraî-

nement est plus long et les performances se dégradent ; disons-le clairement, même si on disposait d'une puissance de calcul suffisante, il serait impossible d'entrainer ce correctement ce modèle, d'autant plus que ce n'est pas le seul problème.

Feature extraction

Les réseaux de neurones de convolution utilisent le principe de la convolution réaliser des modèles permettant de résoudre une grande variété de problème à partir de données d'entraînement. Voyons dans le détail un réseau de convolution dans un problème de classification.

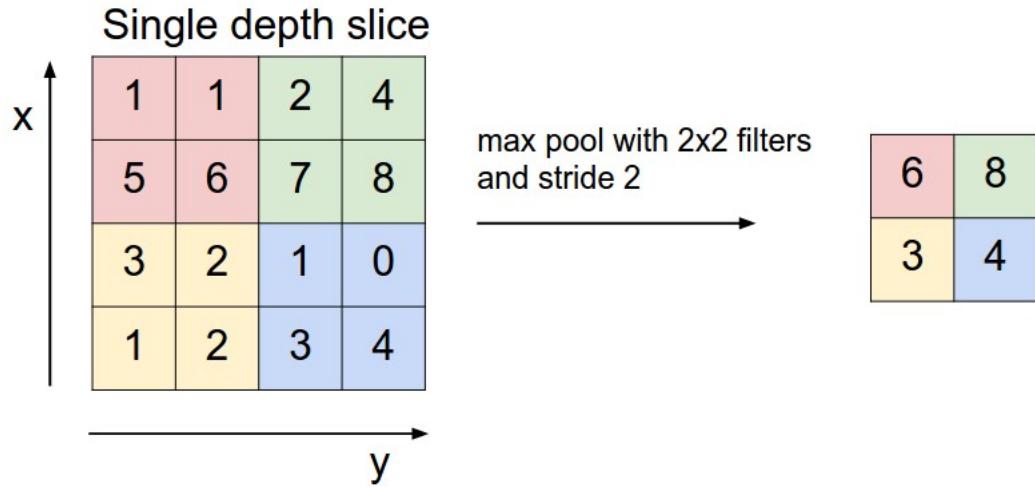
Donc à la place de travailler sur une image 420x420 on va travailler (par exemple) sur une image 10x10 en se basant sur la convolution et le MaxPooling pour faire du **Feature extraction** et **réduction** pour avoir une image réduite qui contient le maximum possible de l'information.

Pooling

Le Max-Pooling est un processus de discréétisation basé sur des échantillons. Son objectif est de sous-échantillonner une représentation d'entrée (image, matrice de sortie de couche cachée, etc.) en réduisant sa dimension. De plus, son intérêt est qu'il réduit le coût de calcul en réduisant le nombre de paramètres à apprendre et fournit une invariance par petites translations (si une petite translation ne modifie pas le maximum de la région balayée, le maximum de chaque région restera le même et donc la nouvelle matrice créée restera identique).

Pour rendre plus concret l'action du Max-Pooling, voici un exemple : imaginons que nous avons une matrice 4×4 représentant notre entrée initiale et un filtre d'une fenêtre de taille 2×2 que nous appliquerons sur notre entrée. Pour chacune des régions balayées par le filtre, le max-pooling prendra le maximum, créant ainsi par la même occasion une nouvelle matrice de sortie où chaque élément correspondra aux maximums de chaque région rencontrée.

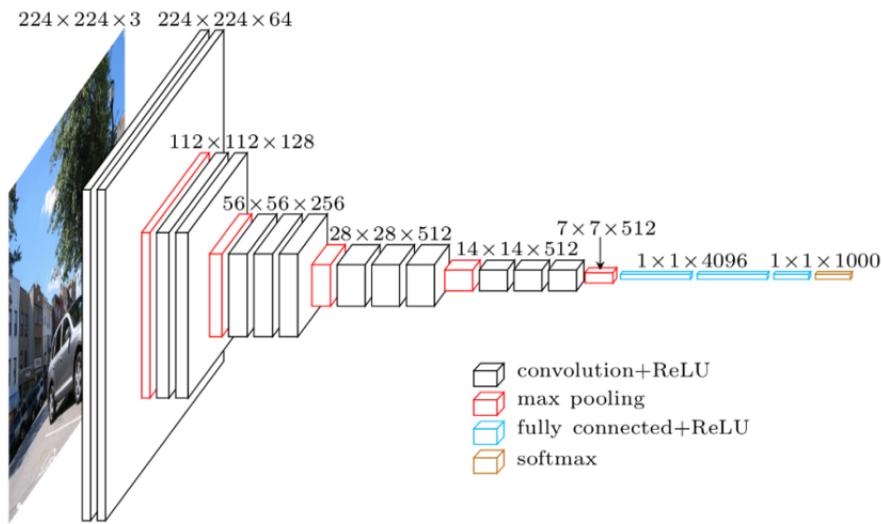
Illustrons le processus :

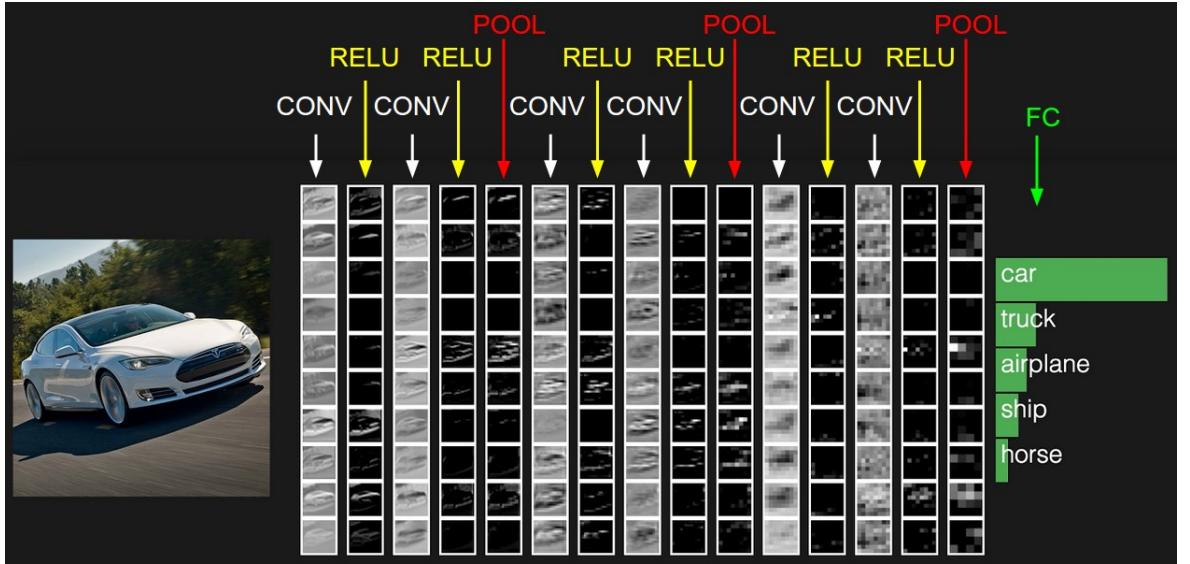


Démarche algorithmique

Alors, comme nous avons présenté, l'utilité des CNN consiste dans la réduction des données (seulement les importants qui vont être conserver).

Avant de créer un modèle ML, il faut appliquer la convolution+pooling plusieurs fois au limite d'un nombre raisonnable :





4.4 Data augmentation

Lorsque l'on s'attaque à un sujet de reconnaissance d'image, il arrive très souvent que le jeu d'apprentissage soit petit, voire inexistant. Embêtant quand on sait que les algorithmes d'aujourd'hui raffolent de gros volumes d'images labellisées! Une des méthodes les plus utilisées pour répondre à ce problème est la Data Augmentation (DA) – comprendre l'augmentation artificielle de la taille du dataset en utilisant des méthodes de manipulation d'image. Cette DA est réalisée en effectuant des opérations modifiant l'aspect de l'image, sans pour autant en modifier la sémantique : par exemple, en diminuant la luminosité, en effectuant une rotation, etc...

4.4.1 Pourquoi le Data augmentation

L'entraînement d'un réseau de neurones profond sur très peu d'images est souvent challengeant : le modèle n'ayant accès qu'à un nombre limité d'observations, il va avoir tendance à faire de “l'overfitting”, c'est à dire sur-apprendre à partir de l'échantillon d'entraînement, sans pour autant être capable d'émettre des prédictions pertinentes sur de nouvelles images – dans ce cas les performances sont faibles sur l'échantillon de test alors qu'elles étaient bonnes sur les images d'entraînement. Ce phénomène est bien connu

des Data Scientists, qui le résolvent souvent en augmentant la taille du dataset et/ou réduisant le nombre de paramètres du modèle. La première méthode est souvent difficile à mettre en place car le travail de recueil/labellisation de nouvelles observations est laborieux. La seconde possibilité est envisageable pour un problème de reconnaissance d'images, cependant les modèles même les moins complexes peuvent contenir des centaines de milliers de paramètres, donc difficile à réaliser ! Comme la Data Augmentation permet de générer de nouvelles images labellisées à partir de celles déjà disponibles, c'est une solution relativement facile à mettre en place, et les résultats peuvent être surprenants.

4.4.2 Exemples du Data augmentation

Dans la plupart des techniques utilisées on DA, il est possible de perdre des régions de notre image, c'est pour ça on va utiliser une méthode pour remplir les régions perdu.

Il y a plusieurs méthodes(`reflect`,`wrap`,`nearest`,`constant..`) ,mais en va présenter la méthode **nearest** et **constant** :



(a) La méthode complet

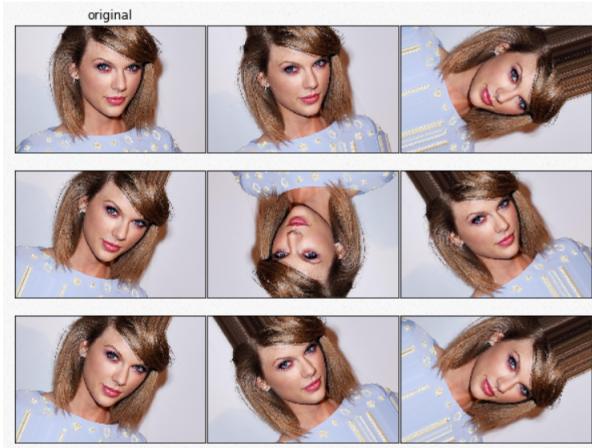


(b) La méthode nearest

FIGURE 4.1 – Comparaison entre les méthodes de filling les plus utilisées

NB : Il est recommandé d'utiliser la méthode **nearest**.

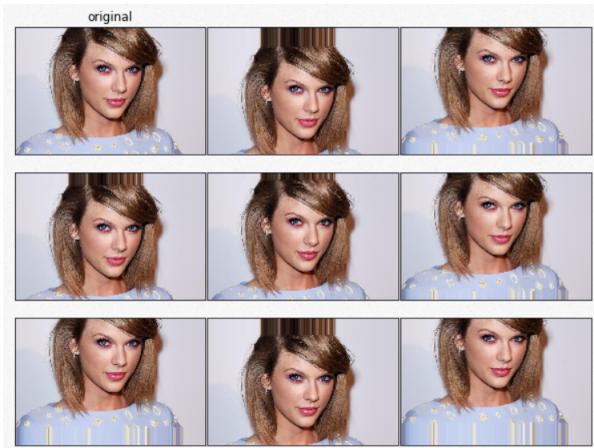
Rotation



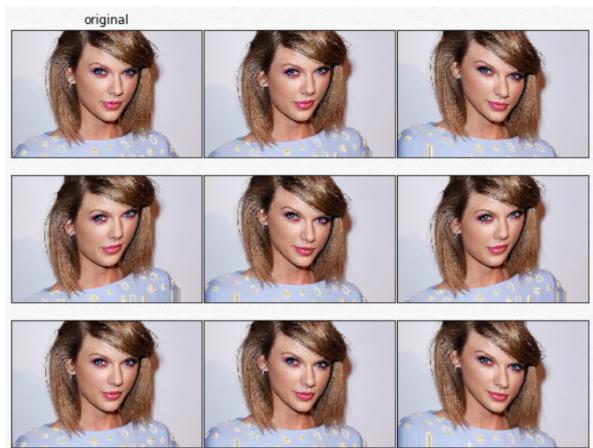
Width shift



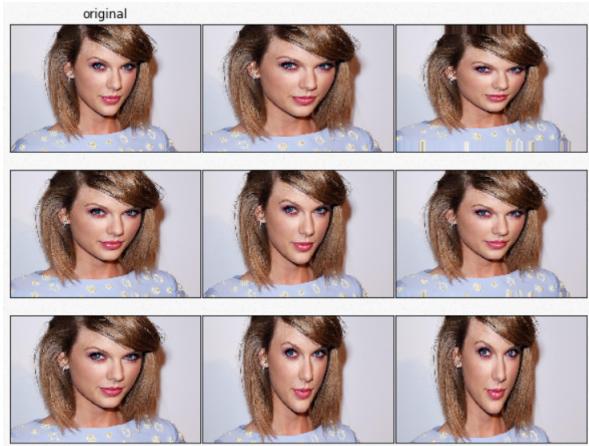
Height shift



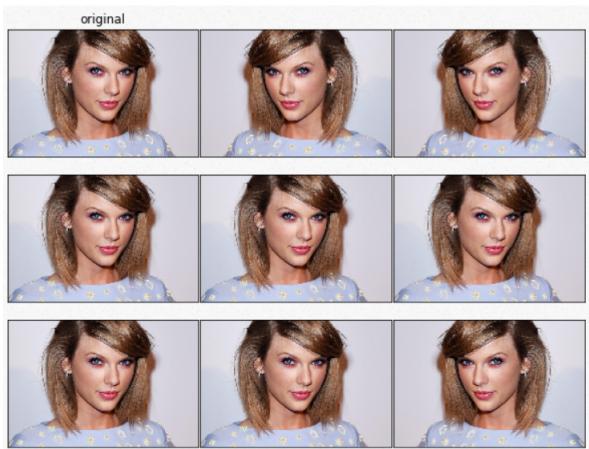
Shearing



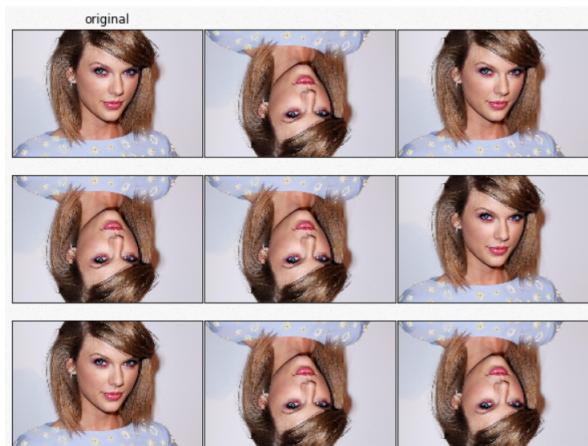
Zoom



Horizontal flip



Vertical flip



Conclusion

Les travaux présentés dans ce projet ont montré l'intérêt de deep learning et ses méthodes(Data augmentation, Réseau de nueron, ..) pour la reconnaissance des émotions Et on a remarqué qu'il a donné des resultat qui sont très acceptable Vue que L'utilisation de Base de donné qui n'est pas très grand et des ressources matérielle simples.