REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

**Ministère de l'Enseignement Supérieur et de la Recherche Scientifique**
**Université Mohamed Khider – BISKRA**
**Faculté des Sciences Exactes, des Sciences de la Nature et de la Vie**
**Département d'informatique**

**N° d'ordre : IVA20/M2/2021**

# Mémoire

Présenté pour obtenir le diplôme de master académique en

# Informatique

Parcours : **Image et Vie Artificielle (IVA)**

---

# Titre : Reconnaissance d'émotions basée sur la vidéo en temps réel à l'aide d'un réseau neuronal convolutif.

---

**Par :**

ALLAOUA Youcef

Soutenu le 04/07/2021 devant le jury composé de :

| | | |
|---|---|---|
| Nom Prénom | grade | Président |
| Hattab Dalila | grade | Rapporteur |
| Nom Prénom | grade | Examinateur |

Année universitaire 2016-2017

## Résumé

L’informatique affective et la reconnaissance d’émotions ont connu un intérêt croissant dans plusieurs domaines de recherche durant ces dernières décennies. En particulier, les expressions faciales représentent un des moyens les plus efficaces pour le relevé des éléments caractéristiques du comportement humain et décrire un état émotionnel.

des algorithmes basés sur l'apprentissage en profondeur (Deep Learning) sont mis en œuvre pour classer les émotions exprimées par le visage en temps réel capturées via une webcam. En effet, Le Réseau de Neurones Convolutif (CNN) est utilisé pour détecter en temps réel les émotions exprimées par le visage en sept émotions différentes obtenues à partir des images traitées. Néanmoins, les CNNs utilisés ont succédé de realiser un taux de précision de reconnaissance au tour de 66% due du manque de processus de régularisation d’un coté, ainsi que l’utilisation unique de la base de données FER2013 d’autre coté.

Dans notre travail, nous avons ajouté un processus de régularisation dans le CNN ainsi que l’enrechissement de la base de données FER2013 par celle FER2013new afin de d’offrir un maximum de donnees. Les resultats obtenus sont prometteuses et traduit par le taux de précision de reconnaissance qui peut arriver jusqu’à 84%.

## Abstract

Affect computing and emotion recognition have shown an increased interest in several research areas for the past decades. Notably, facial expressions are one of the most powerful ways for depicting specific patterns in human behavior and describing human emotional state.

algorithms based on deep learning are implemented to classify the emotions expressed by the face in real time captured via a webcam. Indeed, the Convolutional Neural Network (CNN) is used to detect in real time the emotions expressed by the face in seven different emotions obtained from the processed images. Nonetheless, the CNNs used succeeded in achieving a lap recognition accuracy rate of 66% due to the lack of regulatory process on one side, as well as the unique use of the FER2013 database on the other side.

In our work, we added a regulation process in the CNN as well as the enrichment of the FER2013 database by the FER2013new one in order to offer a maximum of data. The results obtained are promising and reflected in the recognition accuracy rate which can reach 84%.

## ملخص

أظهرت الحوسبة المؤثرة والتعرف على المشاعر اهتمامًا متزايدًا بالعديد من مجالات البحث على مدى العقود الماضية. وتجدر الإشارة إلى أن تعبيرات الوجه هي واحدة من أقوى الطرق لتصوير أنماط معينة في السلوك البشري ووصف الحالة العاطفية للإنسان.

يتم تنفيذ الخوارزميات القائمة على التعلم العميق لتصنيف المشاعر التي يعبر عنها الوجه في الوقت الحقيقي التي يتم التقاطها عبر كاميرا الويب. في الواقع ، تُستخدم الشبكة العصبية التلافيفية (CNN) للكشف في الوقت الحقيقي عن المشاعر التي يعبر عنها الوجه في سبعة مشاعر مختلفة تم الحصول عليها من الصور المعالجة. ومع ذلك ، نجحت شبكات CNN المستخدمة في تحقيق معدل دقة التعرف على اللفات بنسبة 66٪ بسبب الافتقار إلى العملية التنظيمية من جانب ، وعن الاستخدام لقاعدة بيانات FER2013 الفريدة من جانب آخر.

في عملنا ، أضفنا عملية تنظيم في CNN بالإضافة إلى إثراء قاعدة بيانات FER2013 بواسطة FER2013 الجديدة من أجل تقديم أقصى قدر من البيانات. النتائج التي تم الحصول عليها واعدة وتنعكس في معدل دقة التعرف الذي يمكن أن يصل إلى 84٪.

# Remerciement

*Au terme de ce mémoire, je tiens à exprimer mes remerciements et ma profonde gratitude avant tout au bon DIEU de m'avoir donné la force et le courage pour mener à bien ce modeste travail, je saisis de cette occasion pour exprimer mon profond remerciement à ma enseignante encadreure Mm Hattab Dalila pour son soutien et son louable effort, durant toute la durée de ce travail, Je remercie également mon oncle Sahraoui Mohamed de m'avoir toujours soutenus et de m'avoir toujours encouragé à aller le plus loin possible. mes remerciements y vont aussi à l'endroit de tous les professeurs du département informatique de l'université Mohamed Khidher biskra et a tous ceux qui m'ont aidés à élaborer ce travail de près ou de loin.*

# Dédicace

*Je dédié ce mémoire à ma cher mère auquelle je dois ce que je suis aujoud'hui, à mon frère Fares avec qui je partage tous ma vie et à tous ceux qui m'aiment ou ceux qui m'aimeront.*

**Table des matières**

# Introduction Générale

Les émotions colorent notre vie, permettent d'exprimer les différentes facettes de la personnalité, et, les vivre pleinement, c'est s'autoriser une existence intense. On a beaucoup cru, au siècle précédent, à la toute-puissance de la raison en oubliant l'émotion. Et pour cause ! On la considérait comme un obstacle au travail de la raison. Grâce au neuroscientifique et à l'imagerie cérébrale, on sait désormais que l'être humain n'est pas un décideur rationnel et que l'émotion est un partenaire fondamental de la cognition humaine, de sa créativité et de sa prise de décision.

Doter la machine des capacités de reconnaissance d'état émotionnel, tel est le défi scientifique autour duquel se rassemblent différentes communautés (traitement du signal, traitement d'images, intelligence artificielle, robotique, interaction homme-machine, etc.).

L'état émotionnel des humains peut être obtenu à partir d'un large éventail d'indices comportementaux et des signaux qui sont disponibles par le biais d'une expression ou d'une présentation visuelle et physiologique de l'émotion :

L'état émotionnel à travers l'expression visuelle est évalué en fonction de la modulation des expressions faciales, gestes, postures et plus généralement le langage corporel. Les données sont capturées par une caméra, permettant des configurations non intrusives. Les systèmes sont généralement très sensibles à la qualité de la vidéo, l'éclairage, la pose et la taille du visage sur la vidéo.

L'état émotionnel à travers la représentation physiologique est estimé par la modulation de l'activité du système nerveux autonome. L'estimation peut être très fiable et est moins sensible à la qualité des émotions que celles extraites des modalités visuelles. La principale limitation est liée à l'intrusion des dispositifs de détection **[2].**

Doter la machine des capacités de reconnaissance d'état émotionnel, tel est le défi scientifique autour duquel se rassemblent différentes communautés (traitement du signal, traitement d'images, intelligence artificielle, robotique, interaction homme-machine, etc.).

La reconnaissance automatique des émotions faciale suit le schéma général d'un processus de la reconnaissance des formes qu'est défini par les étapes suivantes: extraction, apprentissage-test, classification .

Plusieurs travaux ont été développés dans le domaine de machine Learning sur la reconnaissance des émotions faciales en utilisant plusieurs algorithmes pour l'extraction des caractéristiques (statistiques ou structurelles) et des classifieurs. Ces travaux ont prouvé leurs puissances en termes du taux de reconnaissance sur les petites bases de données.

Le deep Learning et plus particulièrement les réseaux de neurones convolutionnels (CNN) ont apparu spécialement pour résoudre les problèmes rencontrés de la machine Learning. L'un des ingrédients les plus importants pour le succès de ces méthodes est la disponibilité de grandes quantités de données d'entraînement.

Le Convolutional Neural Networks (CNN) est l'une des structures réseau les plus représentatives de la technologie d'apprentissage en profondeur et a connu un grand succès dans le domaine du traitement et de la reconnaissance d'images.

L’objectif de ce mémoire consiste à proposer une approche la reconnaissance des émotions faciales en se basant sur la méthode des réseaux de neurones convolutifs qui se base sur le processus de régularisation et utilise un maximum de quantités de données d'entraînement présente par l’enrechissement de la base de données FER2013 par celle FER2013new.

La structure de ce mémoire est basé sur trois chapitres :

**-Le chapitre 1** nous présente un aperçu général sur la reconnaissance des émotions faciales

et du Deep Learning.

**-Le chapitre 2** est consacré à l’étude du modèle proposé à savoir le CNN et les différents algorithmes d’apprentissages utilisés.

**-Le chapitre 3** illustre l’implémentation et l’expérimentation de notre système, les outils et logiciels que nous avons eu à utiliser pour le développement, et la réalisation de notre système. En fin, nous terminerons ce mémoire par une conclusion générale et quelques perspectives.

# CHAPITRE 01 : la reconnaissance des émotions et Deep Learning

# 1 Introduction

Dans ce premier chapitre, nous présentons quelques notions concernant les émotions telles que leurs définitions, leurs différentes théories, et leurs composantes. Ensuite, nous décrivons un état de l'art sur les méthodes développées dans le domaine de reconnaissance des émotions et plus particulièrement nous abordons la notion du Deep Learning et ses différentes architectures existantes.

# 2 Émotions

Les émotions sont des expériences conscientes caractérisées par une intense activité mentale et un degré élevé de plaisir ou de déplaisir. De plus, elles sont des états de sentiment qui entraînent des changements physiques et psychologiques qui influencent notre comportement.

En parlant à d'autres personnes, nous avons tendance à montrer différentes émotions pour les aider à mieux comprendre ce que nous exprimons ou ressentons. C'est ce qu'on appelle la reconnaissance des émotions.

Intuitivement, le cerveau humain est capable de reconnaître les émotions en analysant des caractéristiques spécifiques telles que les expressions faciales.

Avec l'avancement en intelligence artificielle et l'apprentissage automatique, il y a eu diverses implémentations de prototypes de reconnaissance des émotions grâce à l'utilisation de différents algorithmes intelligents.

Mais les résultats les plus marquants en termes de précision et de validation ont été atteints jusqu'à présent grâce à l'utilisation des réseaux de neurones profonds **[3].**

# 3 Expression faciale

Une expression faciale est une manifestation visible d'un visage de l'état d'esprit (émotion, réflexion), l'activité cognitive, physiologique (fatigue, douleur), personnalité et psychopathologie d'une personne. Elle repose sur trois principales caractéristiques influençant sur la nature de l'expression faciale à savoir :

la bouche, les yeux et les sourcils.

Ainsi, caractérisant une déformation du visage par un changement perceptible visuel**[4]**.

# 4 Expressions faciales vs émotions

Les expressions faciales émotionnelles sont des changements faciaux traduisant des états émotionnels internes, des intentions ou des communications sociales d'une personne. L'analyse de l'expression faciale était un sujet de recherche actif pour les scientifiques du comportement depuis le travail de Darwin en 1872 . Il est important de souligner dès le début qu'il y a une distinction d'un point de vue de la vision par ordinateur entre la reconnaissance de l'expression faciale (REF) et la reconnaissance des émotions humaines.

Comme l'expliquent Fasel et Luettin **[5]**, la première traite la classification du mouvement facial et la déformation des traits faciaux en classes abstraites basées sur des informations visuelles. La deuxième est le résultat de nombreux facteurs différents et peut être révélée sur plusieurs

canaux, par exemple, la voix, la pose, les gestes, la direction du regard et l'expression faciale. En effet, la correspondance d'une expression faciale à une émotion implique la connaissance des catégories d'émotions humaines auxquelles des expressions faciales peuvent être attribuées.

## 5 Différents types et psychologie des émotions basales

| **Expressions** | **Distance entre paupières** | **Distance entre œil et sourcil** | **Distance entre les coins de la bouche** | **Distance entre lèvre supérieure et lèvre inférieure** | **Distance entre les coins de l'œil et de la bouche** |
|---|---|---|---|---|---|
| **Joie**  | accroît ou décroît | accroît ne change pas ou décroît | accroît | ne change pas ou accroît | décroît |
| **Surprise**  | Accroît | Accroît | ne change pas ou décroît | Accroît | ne change pas ou accroît |
| **Dégout**  | Décroît | décroît | Accroît ne change pas ou décroît | Accroît | accroît ne change pas ou décroît |
| **Colère**  | Décroît où accroît | décroît | ne change pas ou décroît | Accroît ne change pas ou décroît | ne change pas ou accroît |
| **Tristesse**  | Décroît | Accroît | ne change pas ou accroît | ne change pas ou accroît | ne change pas ou décroît |

| | | | | | |
|---|---|---|---|---|---|
| **Peur**   | ne change pas ou accroît | ne change pas ou accroît | Accroît ne change pas ou décroît | ne change pas ou accroît | Accroît ne change pas ou décroît |

***Tableau 1.1*** *Descriptions des six expressions faciales* ***[6].***

## 6 Synthèse des travaux développés pour la reconnaissance d'expressions faciales :

Le tableau suivant (tableau1.2) présente quelques techniques récentes développées pour la reconnaissance des expressions faciales :

| **Références** | **Techniques** | **Expressions faciales** | **Bases de données** | **Types** | **Sujets** | **Performance** |
|---|---|---|---|---|---|---|
| [7] | LPB, SVM | Six | CK+ | Images | 50 | 97% |
| [8] | EBGM, ELM | Six | CK+ | Images | 40 | 96% |
| [9], [10] | PCA, LDA | Six | CK+ | 193 images | 9 | 92%<br>75% |
| [11] | SVM | Sourire spontané et sourire Sournois | MMI | Vidéos | 52 | 94% |
| [12] | HOG, SVM | Six | CK+ | Images | | 90% |
| [13] | HMMS | Sept | -Propre base<br>Cohn-kanade | Vidéos | 5 sujets<br>53 sujets | 84,46%<br>58,63% |
| [14] | MAA | Sept | CMU | 166 images | 30 sujets | 84, 34% |
| [15] | Codification Des règles | Six | | Images | 8 sujets | 92% |

| | | | | | | |
|---|---|---|---|---|---|---|
| [16] | Multiboots, SVM linéaire | Six | BU-3DDFE | Scans 3D | 60 sujets | 97,75%<br>98,81% |
| [17] | CNN | Sept | Fer2013 | Images | 35000 sujets | 66 ,67% |
| [18] | CNN | Six | CK+ | Images | 123 sujets | 91% |

***Tableau 1.2*** *Synthèse des travaux développés.*

# 7 Réseaux de neurones:

## 7.1 Définition

Un réseau de neurones artificiels est un système dont la conception est à l'origine inspirée du fonctionnement des neurones biologiques, et qui par la suite s'est rapproché des méthodes statistiques **[W2]**. Les réseaux de neurones artificiels sont des réseaux fortement connectés par des processeurs élémentaires fonctionnant en parallèle. Chaque processeur élémentaire (neurone artificiel) calcule une sortie unique sur la base des informations qu'ils reçoivent.

## 7.2 Historique

Les réseaux neuronaux ont vu le jour qu'en 1943 par W.MCCulloch et W. Pitts du neurone formel qui est une abstraction du neurone physiologique. Par cette présentation, ils ont pu démontrer que le cerveau est équivalent à une machine de Turing, la pensée devient alors purement des mécanismes matériels et logiques. Ils déclarèrent en 1955 "Plus nous apprenons de choses au sujet des organismes, plus nous sommes amenés à conclure qu'ils ne sont pas simplement analogues aux machines, mais qu'en est-il de cette machine. La démonstration de McCulloch et Pitts a été l'un des acteurs importants de la création de la cybernétique.

En 1949, D.Hebb présenta dans son ouvrage "The Organization of Behavior" une règle d'apprentissage. De nombreux modèles de réseaux aujourd'hui s'inspirent encore de la règle de Hebb.

En 1958, F. Rosenblatt développe le modèle du Perceptron. C'est un réseau de neurones inspiré du système visuel. Il possède deux couches de neurones : une couche de perception et une couche liée à la prise de décision. C'est le premier système artificiel capable d'apprendre par expérience. Dans la même période, le modèle de L'Adaline (ADAptive LINar Element) a été présenté par B. Widrow, chercheur américain à Stanford. Ce modèle sera par la suite le modèle de base des réseaux multicouches.

En 1969, M. Minsky et S. Papert publient une critique des propriétés du Perceptron. Cela va avoir une grande incidence sur la recherche dans ce domaine. Elle va fortement diminuer jusqu'en 1972, où T. Kohonen présente ses travaux sur les mémoires associatives et propose des applications à la reconnaissance de formes.

C'est en 1982 que J. Hopfield présente son étude d'un réseau complètement rebouclé, dont il analyse la dynamique.

Aujourd'hui, les réseaux neuronaux sont utilisés dans de nombreux domaines (entre autres, vie artificielle et intelligence artificielle) à cause de leur propriété en particulier, leur capacité d'apprentissage, et qu'ils soient des systèmes dynamiques **[W3].**

### 7.3 Topologie

Chaque réseau de neurones est connecté entre eux de diverses manières. De la figure suivante.

Nous pouvons distinguer deux familles de réseaux de neurones :

1- Non bouclés ou statiques (a) et (b), et bouclés (dynamiques) (c) et (d**)[19].**

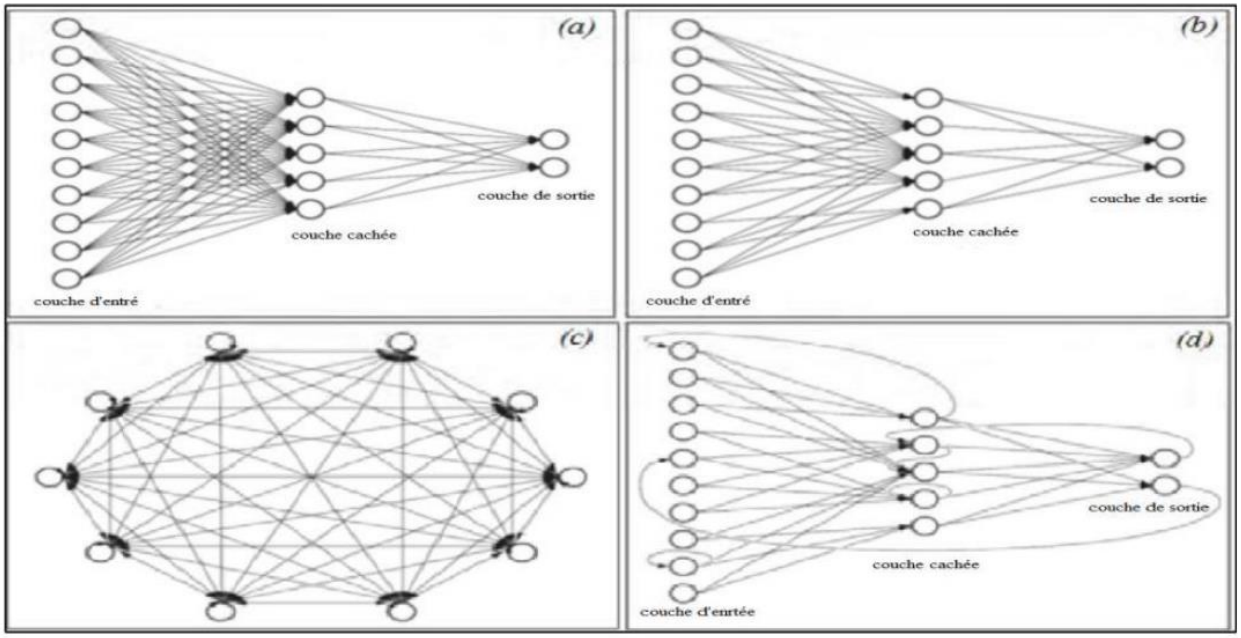



***Figure 1.2*** *Topologie des Réseaux de neurones artificiels* ***[19].***

## 8 L'apprentissage en profondeur (deep Learning):

Le Deep Learning est un nouveau domaine de recherche de la machine Learning (ML), qui a été introduit dans le but de rapprocher le ML de son objectif principal à savoir : l'intelligence artificielle. Il concerne les algorithmes inspirés par la structure et du fonctionnement du cerveau. Ils peuvent apprendre plusieurs niveaux de représentation dans le but de modéliser des relations complexes entre les données **[19].**

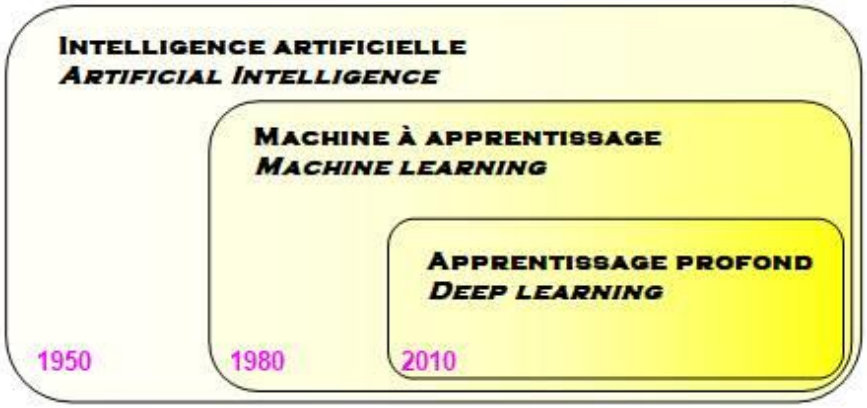



***Figure 1.3*** *La relation entre l'intelligence artificielle, le ML et l'apprentissage profond* ***[w6].***

## 8.1 Définition

L'apprentissage profond est une notion issue du fait que les réseaux neurones disposaient de plus en plus de couches cachées et que le nombre élevé de couches devenait une source de problèmes. En effet, à partir d'un nombre de couches, le réseau neuronal n'était plus capable d'assimiler les informations et d'apprendre correctement.

Des solutions ont été apportées à ces problèmes et les réseaux de neurones sont de plus en plus dotées de couches multiples et capables d'apprendre. Tous ces types de réseaux de neurones peuvent être regroupés sous la notion « deep learning » **[19].**

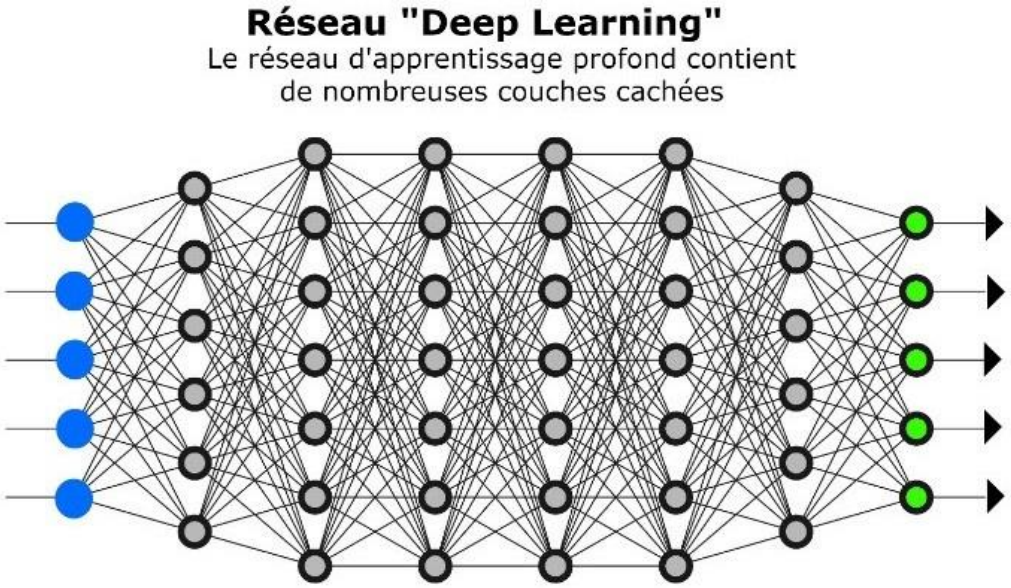



***Figure 1.4*** *Schéma illustratif d'apprentissage profond avec plusieurs couches* ***[w5].***

## 8.2 Historique

L'idée du Deep Learning n'est pas une idée récente, mais elle date en réalité des années 1980, plus particulièrement suite aux travaux de réseaux de neurones multicouches et aux travaux de certains pionniers du machine Learning et du Deep Learning comme le français Yann Le Cun. En collaboration avec deux autres informaticiens, Kunihiko Fukushima et Geoffrey Hinton, ils mettent au point un type d'algorithme particulier appelé Convolutional neural network.

Bien que cette approche donne des résultats, ses progrès et son évolution sont limités par les progrès technologiques en matière de micro-processeurs, de puissance de calculs, et du manque d'accessibilités à des données afin de pouvoir entraîner les neurones. Cependant certains chercheurs ont continué à travailler sur ce modèle pendant environ deux décennies et avec l'aide des évolutions en matière de technologies, mais surtout avec la disponibilité toujours plus grande de données ont pu améliorer cette technique.

Afin de développer un système d'apprentissage performant il faut pouvoir l'exercer et cela requiert un nombre important de données à tester. C'est dans ce contexte qu'en 2007 le STANFORD VISION LAB, avec Fei-Fei Li à sa tête, développent un agrégateur d'images où sont consignés et étiquetés quelques millions de photos : ImageNet. En 2010, ImageNet regroupe 15 000 000 d'images toutes catégorisées en fonction de leurs caractéristiques propres (véhicules, animaux, ...) **[W4].**

En 2012, le Deep Learning est remis au goût du jour avec un succès retentissant au ImageNet Large Scale Visual Recognition Challenge (ILSVRC) qui est un concours annuel de reconnaissance d'image fondée par l'université de Stanford, dans le cadre de son laboratoire STANFORD VISION LAB [W3]. Plusieurs équipes de chercheurs en informatique s'affrontent dans ce concours tous les ans afin de décerner la victoire au programme ayant eu le plus faible taux d'échec. Et alors que les algorithmes d'apprentissage profond sont absents de la compétition, en 2012 c'est bel et bien un algorithme de Deep Learning qui va remporter l'édition 2012 à la surprise générale **[19].**

## 8.3 Pour quoi le choix "apprentissage profond"

Tout d'abord les différents algorithmes du deep Learning ne sont apparus qu'à l'échec de l'apprentissage automatique tentant de résoudre une grande variété de problèmes de l'intelligence artificielle (l'IA) :

- Afin d'améliorer le développement des algorithmes traditionnels dans de telles tâches de l'IA.

- De développer une grande quantité de données telle que les big data.

- De s'adapter à n'importe quel type de problème.

- D'extraire les caractéristiques de façon automatique **[19][20].**

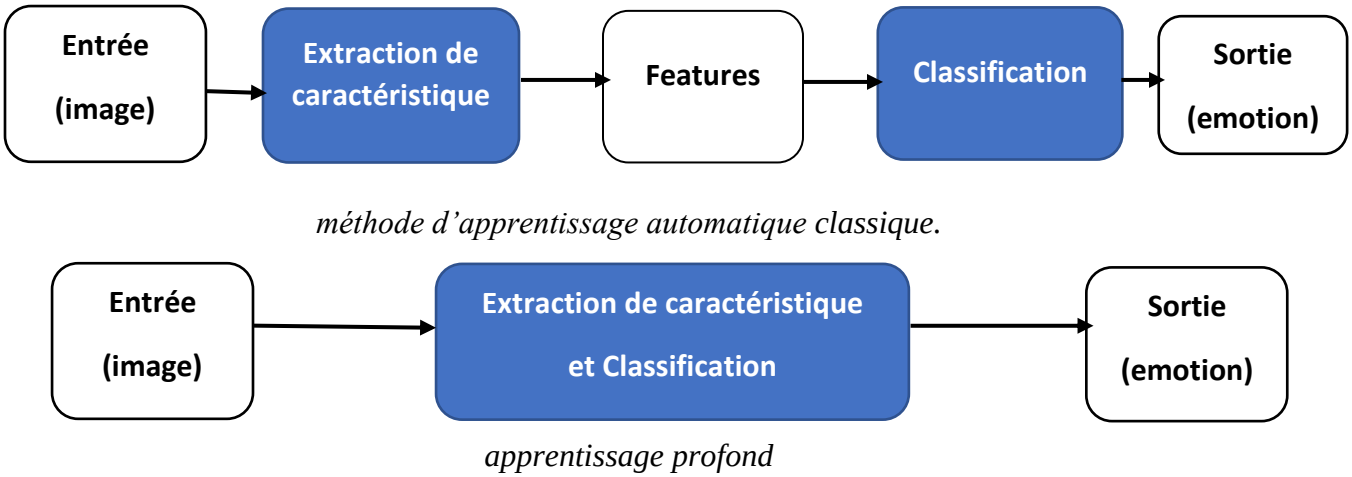



***Figure 1.5*** *Comparaison entre la machine Learning et le Deep Learning* **[19][20].**

## 8.4 Les différentes Architectures du "apprentissage profond"

### 8.4.1 Les réseaux de neurones convolutifs (CNN)

(CNNs) sont un type de réseau de neurones spécialisés pour le traitement de données ayant une topologie semblable à une grille. Qui se sont avérés très efficaces dans des domaines tels que la reconnaissance et la classification d'images et vidéos. CNN a réussi à identifier les visages, les objets, panneaux de circulation et auto-conduite des voitures. Récemment, les CNN ont été efficaces dans plusieurs tâches de traitement du langage naturel (telles que la classification des phrases) **[21] [19].**

Dans le ML, un réseau convolutif est un type de réseau de neurones feed-forward, il a été inspiré par des processus biologiques [w1]. Il existe quatre (4) principales opérations illustrées dans le CNN à savoir :

- La couche convolution.
- La couche Rectified Linear Unit.
- La couche Pooling.
- La couche entièrement connectée.

### 8.4.2 Réseau de neurones récurrents

L'idée derrière les RNN est d'utiliser des informations séquentielles. Dans un réseau neuronal traditionnel, nous supposons que toutes les entrées (et les sorties) sont indépendantes les unes des autres. Mais pour de nombreuses tâches, c'est une très mauvaise idée. Si on veut prédire le prochain mot dans une phrase, il faut connaître les mots qui sont venus avant. Les RNN sont appelés récurrents, car ils exécutent la même tâche pour chaque élément d'une séquence, la sortie étant dépendante des calculs précédents.

Une autre façon de penser les RNN est qu'ils ont une « mémoire » qui capture l'information sur ce qui a été calculé jusqu'ici. En théorie, les RNN peuvent utiliser des informations dans des séquences arbitrairement longues. mais dans la pratique, on les limite à regarder seulement quelques étapes en arrière. **[22] [23] [19]** Il est utilisé pour :

- La modélisation du langage et génération de texte.

- La traduction automatique.

- La reconnaissance vocale.

- Et la description des images.

### 8.4.3 Modèle génératif

Si les modèles discriminatifs comme (CNN, RNN) sont utilisés pour prédire les données du label et de l'entrée, tant disque le modèle génératif décrit comment générer les données, il apprend et fait des prédictions en utilisant la loi de Bayes **[19]**.

Cependant les modèles génératifs sont capables de bien plus que la simple classification comme par exemple générer de nouvelles observations. Voici quelques exemples de modèle génératif :

• Boltzmann Machines

• Restricted Boltzmann Machines

• Deep Belief Networks

• Deep Boltzmann Machines

• Generative Adversarial Networks

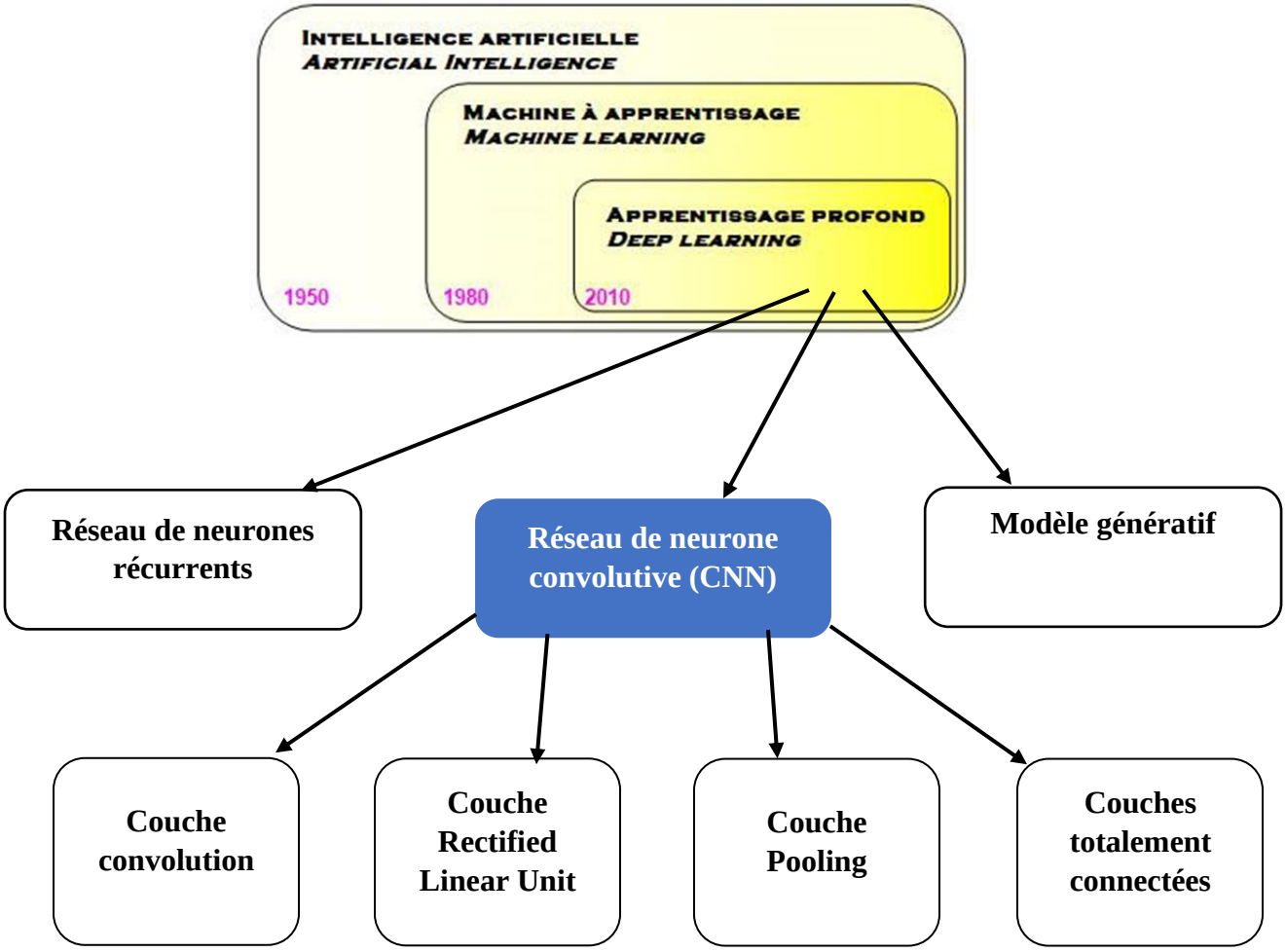



***Figure 1.6*** *Différents modèles du Deep Learning et les couche du CNN.*

### 8.4.4 Exemples d'application de "apprentissage profond"

Les applications du Deep Learning sont utilisées dans divers secteurs, de la conduite automatisée aux dispositifs médicaux **[W2].**

Grace au deep Learning nous pouvons maintenant:

- Faire une colorisation des images en noir et blanc.
- Ajouter des sons à des films silencieux.
- Faire de la traduction automatique.
- Faire de classification des objets en photographies.
- Générer d'écriture automatique.
- Génération de légende d'image.
- Jeu automatique.

## 9 Analyser les expressions faciales, puis déterminer l'émotion automatique

Afin de pouvoir synthétiser un système de reconnaissance automatique des émotions en se basant sur les expressions faciales, trois étapes cruciales sont nécessaires, à savoir :

- La détection du visage, qui va permettre de délimiter la zone du visage,
- L'extraction des caractéristiques facials.
- Et enfin, la classification qui vas déterminer à quelle émotion la combinaison d'expressions faciales extraites correspond.

Les figures suivantes (2 et 3) représentent la différence d'étapes entre un système de reconnaissance automatique des émotions classique et pour un CNN.

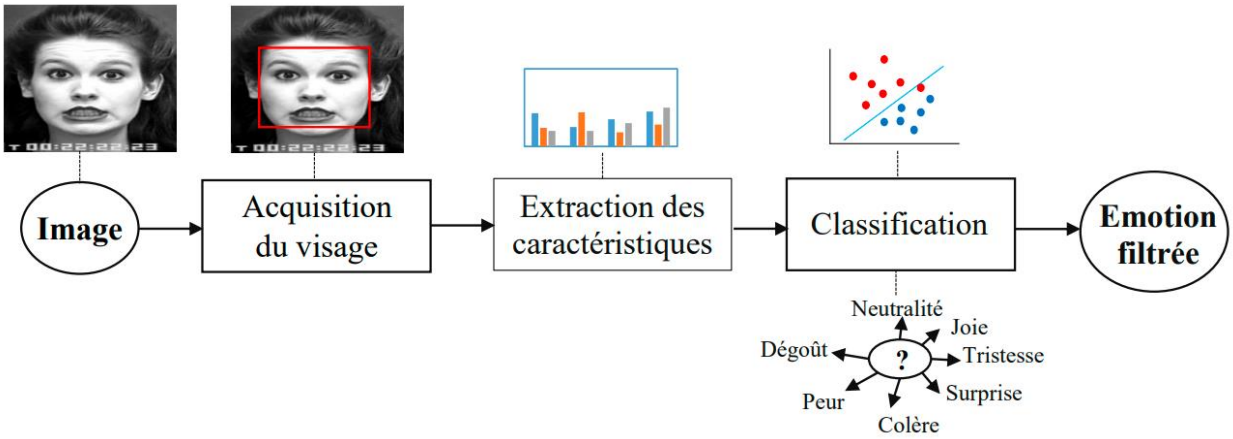



***Figure 1.7*** *Système de reconnaissance automatique des émotions* ***[24].***

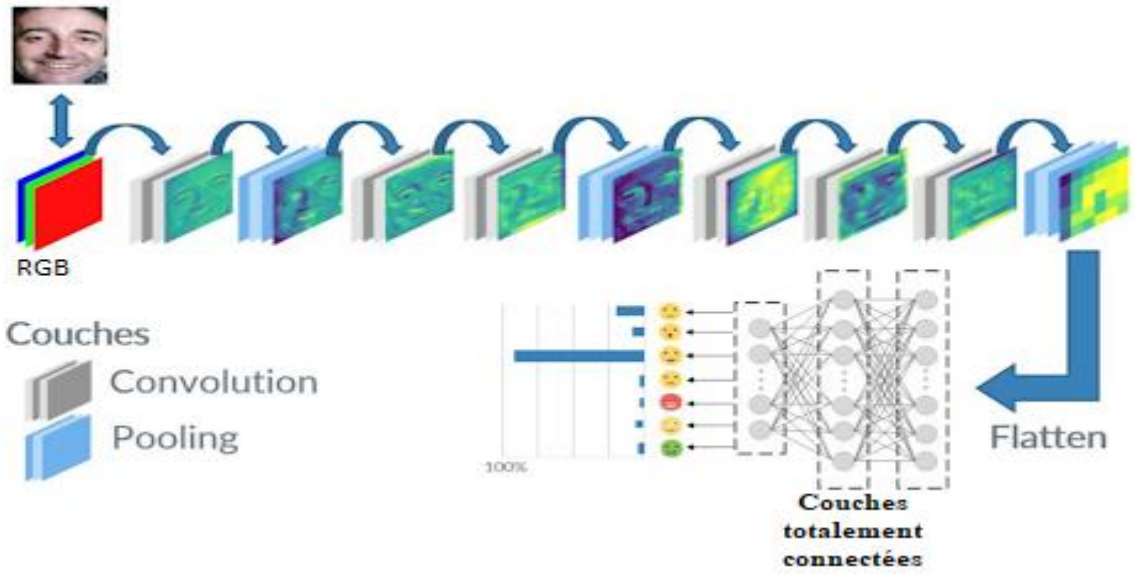



***Figure 1.8*** *Reconnaissance automatique des émotions avec les CNNs****[w1].***

## 9.1 Détection du visage

La première étape d'un système d'analyse des expressions faciales entièrement automatique consiste à localiser la région du visage et ses limites. Le but de la détection du visage est de déterminer si un visage est présent ou non sur l'image et, le cas échéant, de localiser son emplacement (voir Figure 1.7, le rectangle rouge englobe le visage détecté). Une étude exhaustive sur les algorithmes de détection de visages dans a regroupé les différentes méthodes dont :

- Les principaux algorithmes de cette catégorie comprennent l'algorithme de détection de visage Viola-Jones (VJ) et ses variations, les algorithmes basés sur (CNN) et CNN profonds (Deep CNN, DCNN), et les méthodes qui appliquent des stratégies inspirées de l'extraction d'images (image-retrieval) et la transformée généralisée de Hough **[24][1].**

## 9.2 Extraction des caractéristiques facials

**Les différentes méthodes développées pour la reconnaissance des émotions faciale**

Ils existent différentes approches qui ont été développées dans le domaine de reconnaissance des expressions faciales. Elles peuvent être séparées en deux grandes familles à savoir les méthodes globales (ou holistiques) et les méthodes locales. **[19]**

### 9.2.1 Méthodes globales

Elles sont basées sur des techniques d'analyse statistique bien connues. Il n'est pas nécessaire de repérer certains points caractéristiques du visage (comme les centres des yeux, les narines, le centre de la bouche, etc.) à part pour normaliser les images. Dans ces méthodes, les images de visage (qui peuvent être vues comme des matrices de valeurs de pixels) sont traitées de manière globale et sont généralement transformées en vecteurs, plus faciles à manipuler. L'avantage principal des méthodes globales est qu'elles sont relativement rapides à mettre en œuvre et que les calculs de base sont d'une complexité moyenne. En revanche, elles sont très sensibles aux variations d'éclairement, de pose et d'expression faciale. Ceci se comprend aisément puisque la moindre variation des conditions de l'environnement ambiant entraîne des changements inéluctables dans les valeurs des pixels qui sont traités directement. Ces méthodes utilisent principalement une analyse de sous-espaces de visages. L'utilisation de techniques de modélisation de sous-espace a fait avancer la technologie de reconnaissance faciale de manière significative.

Nous pouvons distinguer deux types de techniques parmi les méthodes globales : les techniques linéaires et les techniques non linéaires.

Les techniques linéaires projettent linéairement les données d'un espace de grande dimension (par exemple, l'espace de l'image originale) sur un sous-espace de dimension inférieure. Malheureusement, ces techniques sont incapables de préserver les variations non convexes des variétés (géométriques donc au sens mathématique du terme) de visages afin de différencier des individus. Dans un sous-espace linéaire, les distances euclidiennes et plus généralement les distances de Mahalanobis, qui sont normalement utilisées pour faire comparer des vecteurs de donnés, ne permettent pas une bonne classification entre les classes de formes et entre les individus eux-mêmes. La technique linéaire la plus connue et sans aucun doute l'analyse en composantes principales (PCA), également appelée transformée de Karhunen-Loeve. Le PCA fut d'abord utilisé afin de représenter efficacement des images de visages humains. Bien que ces méthodes globales linéaires basées sur l'apparence évitent l'instabilité des toutes premières méthodes géométriques

qui ont été mises au point, elles ne sont pas assez précises pour décrire les subtilités des variétés (géométriques) présentes dans l'espace de l'image originale.

Ceci est dû à leurs limitations à gérer la non linéarité en reconnaissance faciale les déformations de variétés non linéaires peuvent être lissées et les concavités peuvent être remplies, causant des conséquences défavorables. Afin de pouvoir traiter ce problème de non-linéarité sur la reconnaissance des expressions faciales, de telles méthodes linéaires ont été étendues à des techniques non linéaires basées sur la notion mathématique de noyau ("kernel") comme le Kernel PCA et le Kernel LDA **[25].**

### 9.2.2 Méthodes locales

Elles sont basées sur des modèles, utilisant des connaissances a priori que l'on possède sur la morphologie du visage et s'appuie en général sur des points caractéristiques de celui-ci. Kanade a présenté un des premiers algorithmes de ce type **[26]** en détectant certains points ou traits caractéristiques d'un visage puis en les comparants avec des paramètres extraits d'autres visages. Ces méthodes constituent une autre approche pour prendre en compte la non-linéarité en construisant un espace de caractéristiques local et en utilisant des filtres d'images appropriés, de manière à ce que les distributions des visages soient moins affectées par divers changements.

Les approches bayésiennes, les machines à vecteurs de support (SVM) **[27],** la méthode des modèles actifs d'apparence (AAM) **[25]** ou encore la méthode "local binary pattern"(LBP) ont été utilisées dans ce but.

Toutes ces méthodes ont l'avantage de pouvoir modéliser plus facilement les variations de pose, d'éclairage et d'expression par rapport aux méthodes globales. Toutefois, elles sont plus lourdes à utiliser puisqu'il faut souvent placer manuellement un assez grand nombre de points sur le visage alors que les méthodes globales ne nécessitent de connaître que la position des yeux afin de normaliser les images, ce qui peut être fait automatiquement et de manière assez fiable par un algorithme de détection **[28]**

### 9.2.3 Méthodes hybrides

Ce sont une combinaison des méthodes globales et locales en combinant la détection de caractéristiques géométriques (ou structurales) avec l'extraction de caractéristiques d'apparence locales. Elles permettent d'augmenter la stabilité de la performance de reconnaissance lors de changements de pose, d'éclairement et d'expressions faciales **[28] [19].**

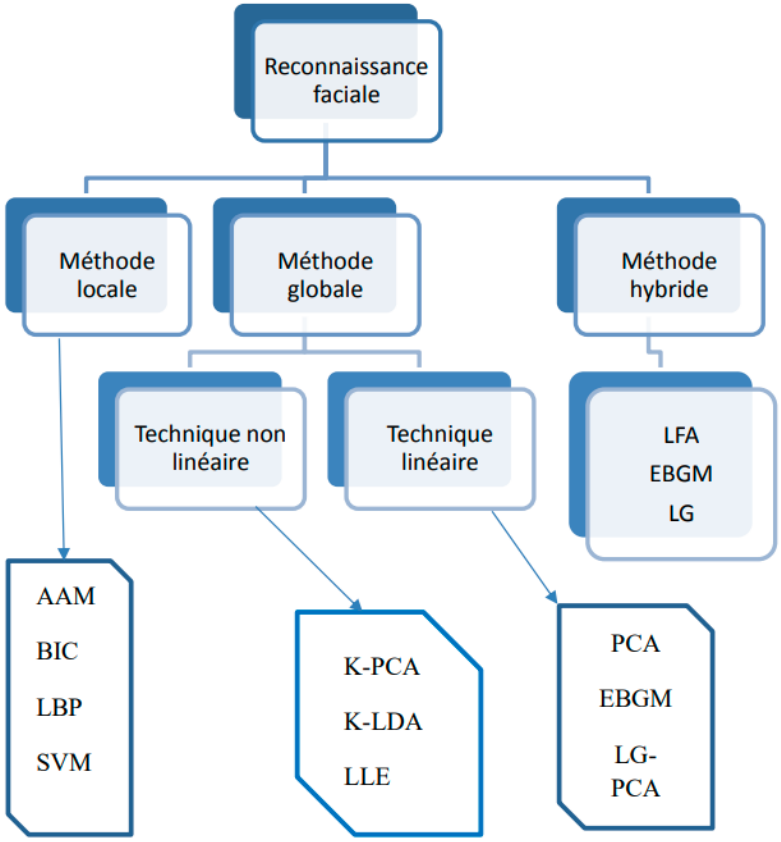



***Figure 1.1*** *Classification des algorithmes principaux utilisés en reconnaissance faciale* ***[19][26].***

### 9.3 La classification

La dernière étape d'un système de reconnaissance automatique des émotions est la classification des émotions selon les caractéristiques faciales extraites. Elle se divise en deux types :( La reconnaissance d'expressions basée sur des images statiques et la reconnaissance d'expressions basée sur des séquences vidéo). De nombreux classifieurs ont été adoptés pour la reconnaissance des émotions, parmi cux **[1]** .

- réseaux de neurone (Neural Networks, NN).

## 10 En temp réel (Pré-traitement vidéo)

Tous les frames sont initialement extraits du signal visuel pour des étapes ultérieures. Étant donné que ces images extraites continent toujours des informations redondantes considérables pour la détection des émotions, nous extrayons uniquement les régions du visage comme suit :

- Toutes les boîtes englobantes contenant des régions de visage dans chaque image ont été extraites en utilisant l'algorithme de Viola-Jones [Viola et Jones, 2004] et une région de visage a ensuite été détectée **[29].**

## 11 Applications possible et les avantages de la reconnaissance d'émotions

La reconnaissance et la classification de l'expression faciale peuvent être applicables dans divers aspects de notre vie quotidienne ci-après une liste non exhaustive des domaines d'application susceptibles de bénéficier de la capacité de détection des expressions faciales émotionnelles **[5].**

- **Interaction homme-machine :** Les expressions faciales constituent un moyen de communication comme de nombreuses autres manières (par ex. le signal vocal). La détection des émotions est naturelle pour les humains mais c'est une tâche très difficile pour les machines.

- **Interaction homme-robot :** Pour les robots sociaux, il est également important qu'ils puissent reconnaître différentes expressions et agir en conséquence afin d'avoir des interactions efficaces [176, 176, 166]. Pour atteindre une telle interaction homme-robot, il est primordial que le robot comprenne les expressions faciales des humains.

- **Neuromarketing :** La mesure automatisée des préférences des consommateurs à partir de leurs expressions faciales en réponse aux publicités des produits aurait un impact profond sur l'analyse des études de marché, car cela permettrait aux entreprises de mieux connaître le consommateur et proposer de nouveaux produits et services à leur clients. McDuff et al. Ont réussi à déterminer si les gens aimaient certaines publicités en analysant leur comportement facial.

- **Psychologie :** La détection d'expressions est extrêmement utile pour l'analyse de la psychologie humaine. Ainsi, la reconnaissance de l'incapacité d'une personne à exprimer certaines expressions faciales peut aider à diagnostiquer les troubles psychologiques précoces.

## 12 Conclusion

Dans cette partie, nous avons défini l'émotion ainsi que les concepts principaux qui lui sont associés tel que l'expression faciale, les types d'émotions basales. De plus, nous avons démontré les travaux développés pour la reconnaissance faciale en donnant un intérêt particulier l'explication de l'approche Deep Learning et comment il se différencie des algorithmes de ML traditionnelles. Nous avons également parlé des trois familles majeures de modèle à savoir les réseaux convolutifs (CNN) les réseaux récurrents ainsi que les modèles génératifs.

Dans le chapitre qui suit, nous rentrerons plus en détail sur les réseaux de neurones convolutifs (CNN).

# Chapiter 02 :CNN pour la reconnaissance d'émotions

# 1 Introduction

Les CNNs sont une catégorie de réseaux de neurones qui se sont avérés très efficaces dans des domaines tels que la reconnaissance et la classification d'images. ils ont réussi à identifier les visages, les objets, panneaux de circulation et auto-conduite des voitures. Dans ce chapitre nous étudions l'approche CNN dans ses diffèrent angles.

## Réseaux de neurones convolutifs

Un réseau de neurones convolutif ou réseau de neurones à convolution (CNN ou ConvNet en anglais pour Convolutional Neural Networks) sont une forme particulière de réseau neuronal multicouches dont l'architecture des connexions est inspirée de celle du cortex visuel des mammifères **[31].** Ces réseaux sont capables de catégoriser les informations des plus simples aux plus complexes. Ils consistent en un empilage multicouche de neurones et fonctions mathématiques à plusieurs paramètres ajustables, qui prétraitent de petites quantités d'informations.


***Figure 2.1*** *Architecture générale d'un CNN.*

Le premier réseau de neurones convolutif (CNN) a été introduit à la fin des années 80 par LeCun **[35] [32].** C'est le premier réseau de neurones pour la reconnaissance d'images, ce réseau permettait la reconnaissance des chiffres manuscrits. L'idée est de passer l'image dans une succession de filtres convolutifs apportant une description réduite et pertinente de l'image. Ces

caractéristiques sont, par la suite, envoyées à un perceptron multicouches composé de couches cachées et d'une couche de sortie complètement connectées permettant la classification du chiffre présent dans l'image. Par la suite, en 1998, Yann LeCun et al **[36]** ont proposé une structure de réseaux de convolutions LeNet-5 représenter dans la **figure 2.2**, associé à un algorithme d'apprentissage pour la classification des chiffres manuscrits.

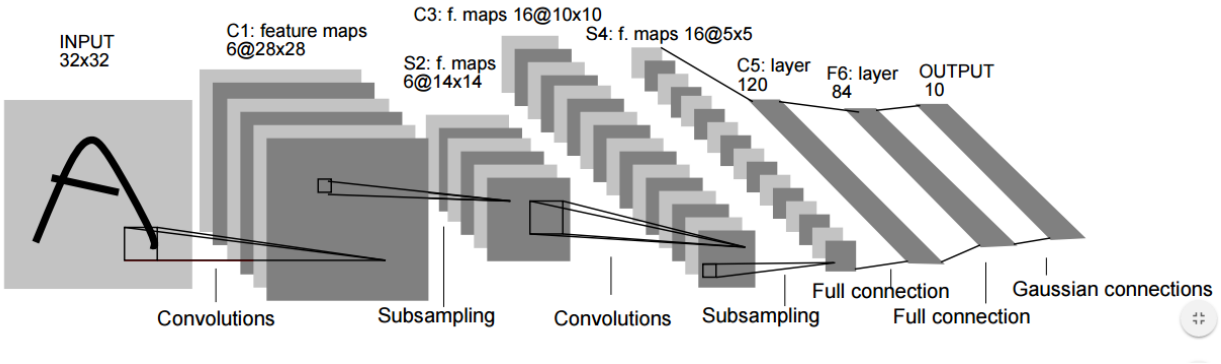



***Figure 2.2**- Architecture d'un réseau de neurones convolutifs LeNet-5 **[36].***

Les réseaux CNN sont particulièrement utiles pour identifier des modèles sur des images afin de reconnaître des objets, des visages et des scènes. Ils apprennent directement à partir de données d'entrées (images) et utilisent des modèles pour les classer, éliminant ainsi la nécessité d'effectuer une extraction manuelle des caractéristiques **[36]**.

Le terme convolutif vient de l'opération de convolution des matrices utilisées dans le traitement de signal. Dans les convNet, deux nouveaux types de couche ont été ajouter dans le réseau par rapport aux réseaux de neurones conventionnels :

La couche convolutif (convolutional layer) et la couche de mise en commun ou de sous-échantillonnage (pooling layer) **[33],** nous les décrivons dans les parties suivantes.

Les CNN sont un type particulier de réseaux de neurones, applicables facilement sur des images pour capter spatialement de l'information **[32].** De plus, de par leur structure convolutive, ils permettent de prendre en entrée des données de grande dimension ce qui est une limite du perceptron multicouches. Une image à trois canaux (RGB) de taille 224 × 224 pixels représente un vecteur d'entrée de taille (150528) pour un perceptron multicouche.

Cela implique (150528) poids à apprendre pour chaque neurone de la couche cachée connectée aux entrées, ce qui est énormes et compliqué à apprendre. Les CNN peuvent être vus comme un assemblage de modules en série permettant l'extraction de caractéristiques de manière hiérarchique à partir des pixels d'une image **[32].**

Au final, les CNN sont composés de couches non rebouclées, chaque couche est composée de neurones qui sont connectés à la couche précédente. Dans l'architecture standards d'un réseau ConvNet (la **figure 2.3**), l'image d'entré (input image) est convolutée avec un filtre a noyaux (kernels) du réseau convolutif pour l'obtention de la carte des caractéristiques de l'image (feature maps) **[36]**. Ces caractéristiques sont ensuite sou échantillonnées à l'étape de pooling, ce processus est répété à plusieurs reprise en alternance (convolution et pooling). Pour la dernière phase celle de sortie, les résultats sont obtenus en utilisant un perceptron multicouche pour aboutir à la classification finale **[32] .**

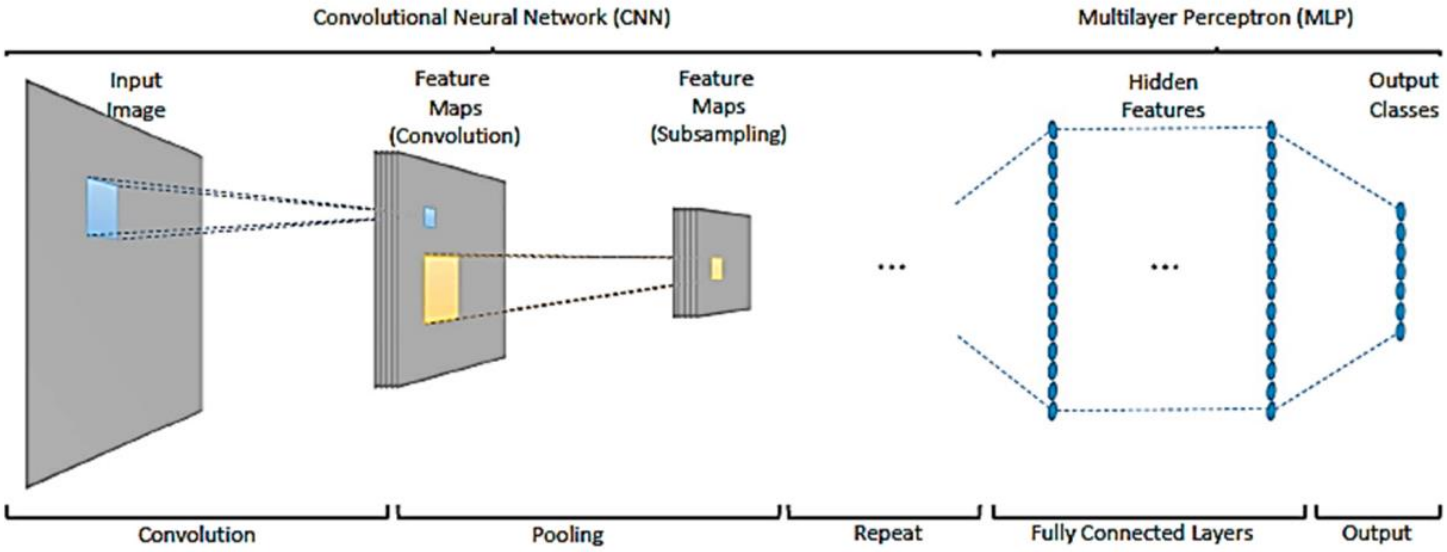



***Figure 2.3**- Les différentes couches d'un réseau de neurones convolutif standard **[37].***

## 2.1 Différents modules d'un réseau de neurones convolutif

Au vu de cette structure globale, nous allons donc nous intéresser maintenant aux différents couches (module) classiquement utilisés dans les réseaux CNN.

### 2.1.1 La convolution

La pièce maîtresse d'un réseau CNN est la couche de convolution. La sortie en résultant est appelée carte de caractéristiques **[32]**. Une couche convolutive est constituée de plusieurs filtres (ou noyaux) de convolution appliquée sur une matrice d'entrée (une image ou une carte de caractéristique précédente). La couche de convolution fait passer les images d'entrée par un ensemble de filtres convolutifs, chacun de ces filtres activant certaines caractéristiques des images.

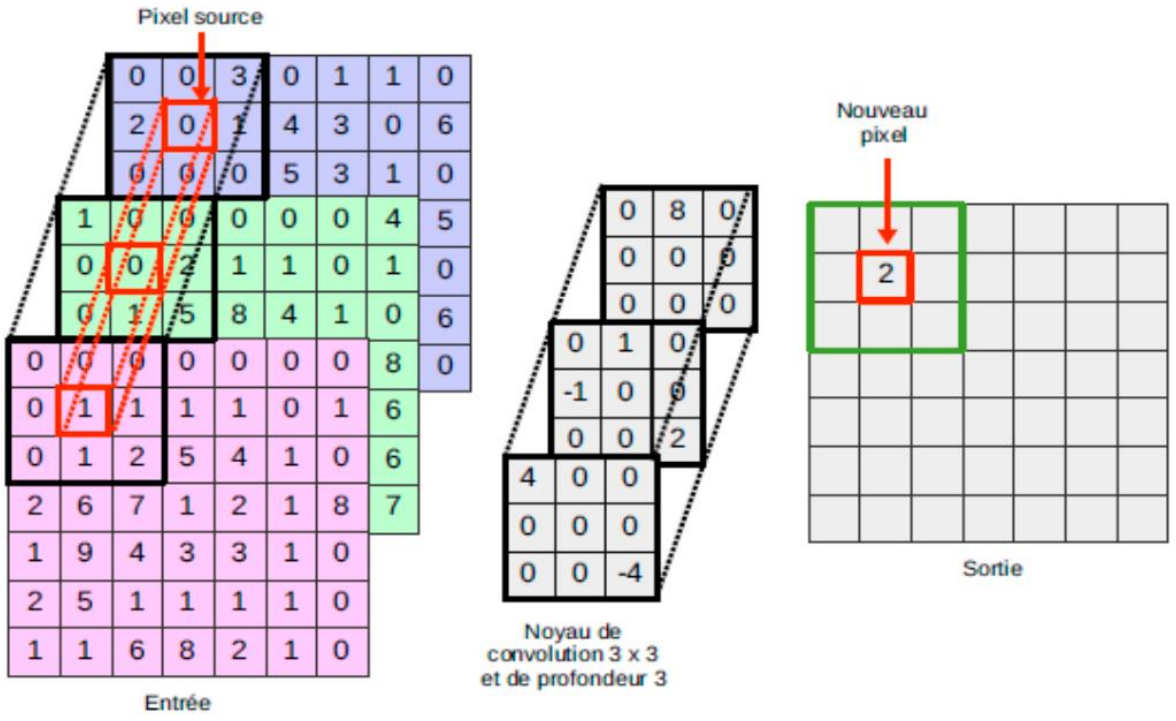



***Figure 2.4**- Illustration de la convolution **[33].***

Étant donné en entrée, un filtre de convolution (ou noyau de convolution) est appliqué pour chaque position de la matrice d'image en question. La profondeur du noyau dépend de de l'image d'entrée sur laquelle il est appliqué. Dans l'exemple illustré dans la **figure 2.4** l'image d'entrée possède trois canaux (image couleur RGB) donc la profondeur du noyau est de trois. Le résultat pour une position donnée correspond à la somme de la multiplication des éléments du noyau par ceux de l'entrée. Dans l'exemple précédant (2×-4) +(5×2) = 2 représente le résultat de la multiplication élément par élément de la matrice par le noyau de convolution. Dans le cadre des CNN, la sortie d'une étape de convolution est appelée carte de caractéristiques. Le nombre de cartes de caractéristiques dépend du nombre de filtres appliqués sur l'image d'entrée **[32] [33].**

Un filtre dans l'étape de convolution sert à faire ressortir certaines caractéristiques d'une image donnée (couleur, contour, luminosité, netteté, etc..). Un filtre d'une couche convolutive est appliqué à toutes les positions de la matrice d'entrée. Ce filtre va être déplacé par pas successifs sur l'ensemble d'images. Pour chaque position du filtre, les valeurs des deux matrices en superposition (filtre et image à traiter) sont multipliées puis projetée dans une nouvelle matrice. Pour cela il faut utiliser un paramètre de la convolution appelé Stride ou pas de convolution. Il représente de combien de pixels dans l'image d'entrée on va se décaler pour réappliquer la convolution **[38]**. Cependant, il peut arriver que la convolution dépasse l'image si on choisit mal le kernel et le stride. Par exemple, pour une image de 4×4 (pixels), un kernel de 2×2 et un stride de 3, on va avoir un problème après la première convolution. De plus, le résultat de la convolution est une image plus petite que l'image d'entrée. A ces deux problèmes, il y a un paramètre qu'on

appelle Padding qui est utilisé dans ce cas de figure en ajoutant tout simplement des 0 autour de l'image d'entrée, pour éviter les dépassements **[32]**.

### 2.1.2 Le pooling

Le but de la couche pooling ou sous–échantillonnage (subsampling) est de réduire la taille des images après le procédé de la convolution, elle permet de rajouter de l'invariance spatiale lors de l'extraction de caractéristiques tout en réduisant la dimension des entrées. Elle peut être de différentes natures mais les types de pooling les plus utilisés sont le Max Pooling (illustré dans la **Figure 2.5)**. Le Max Pooling renvoie l'élément maximum sur une fenêtre de calcul (la matrice issue de la convolution) **[32].**

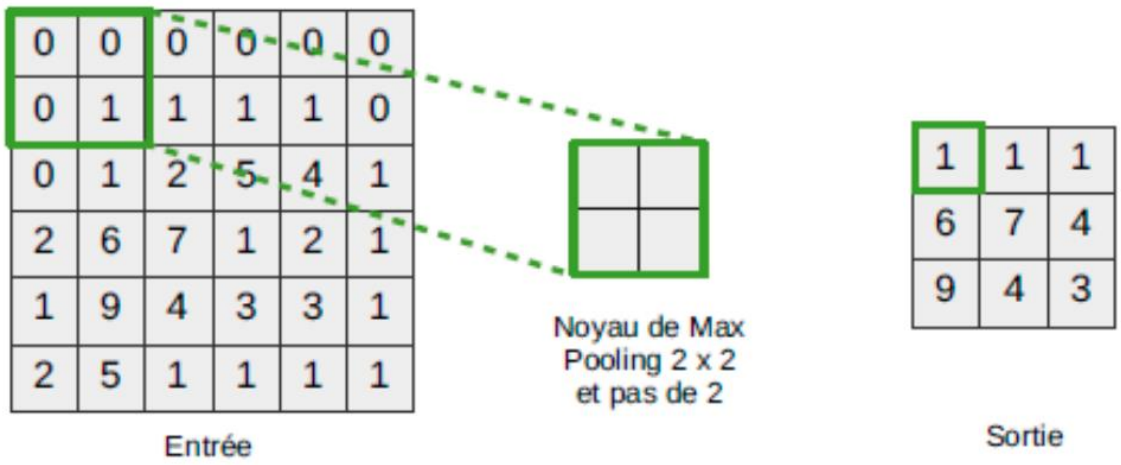



***Figure 2.5****- Illustration du Max Pooling* ***[32].***

En extrayant les valeurs importantes des pixels, cette étape permet de réduire une image tout en conservant les caractéristiques pertinentes. Dans cet exemple (de la **figure 2.5**), le noyau de pooling est de taille 2 × 2 et est appliqué tous les deux pixels (stride = 2). Le maximum des quatre éléments sur une fenêtre de la matrice d'entrée est gardé.

Il existe plusieurs types de pooling **[38]**:

**-Le max-pooling:** qui revient à prendre la valeur maximale de la sélection. C'est le type le plus utilisé car il est rapide à calculer (immédiat), et permet de simplifier efficacement l'image.

**-Le mean-pooling (ou average-pooling):** soit la moyenne des pixels de la fenêtre sélectionné, on calcule la somme de toutes les valeurs et on divise par le nombre de valeurs. On obtient ainsi une valeur intermédiaire pour représenter ce lot de pixels.

**- Le sum-pooling:** représente la moyenne sans avoir divisé par le nombre de valeurs (on ne calcule que leur somme) **[32]**.

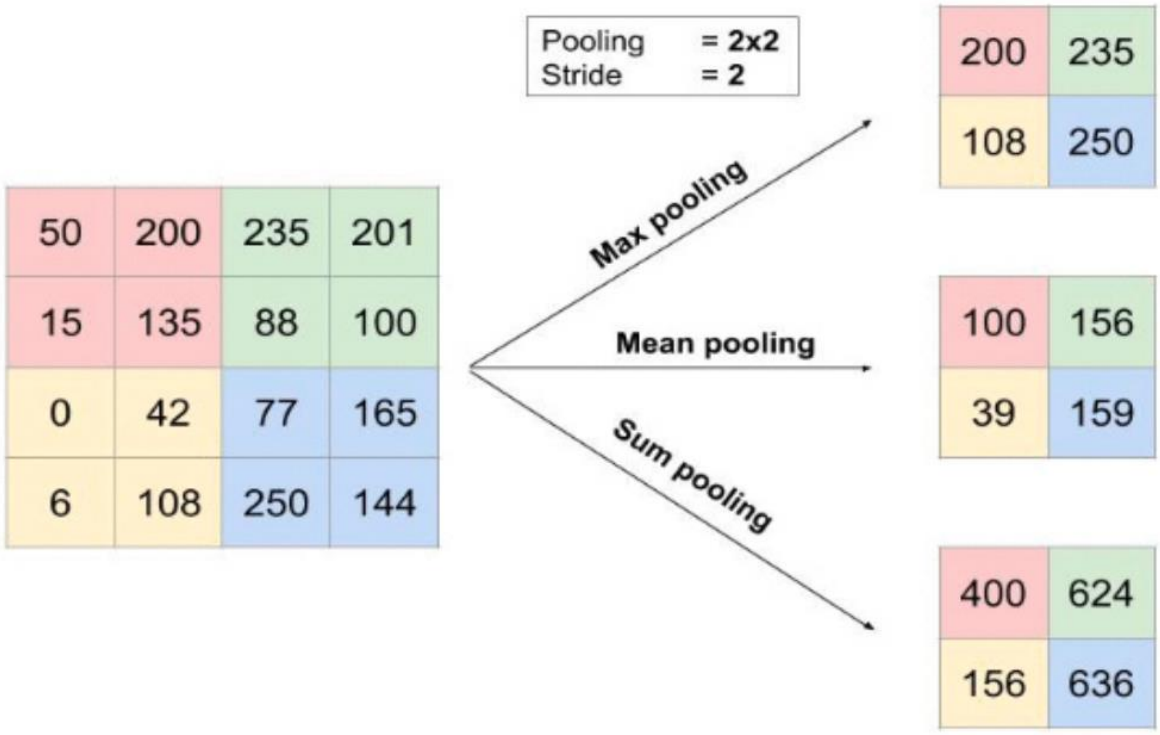



***Figure 2.6****- les différents types de Pooling avec un filtre 2x2 et un pas de 2.*

### 2.1.3- Les fonctions d'activation :

Il est possible d'améliorer l'efficacité du réseau CNN, en intercalant entre les couches de traitement une fonction qui va opérer comme une couche de correction sur les signaux de sortie. Cela permet d'introduire des non-linéarités pour permettre au réseau d'apprendre des systèmes complexes non linéaires. Il existe différentes fonctions d'activation permettant la non-linéarité dans les différentes couches des CNN. Parmi les plus connues : le sigmoïde (logistique), la tangente hyperbolique et la fonction d'unité de rectification linéaire (Rectified Linear Unit) ReLU **[32] [33].**

La fonction ReLU (abréviation de Unités Rectifié linéaires : F(x)=max (0, x)), est une fonction qui vient briser (une partie de) la linéarité en supprimant une partie des valeurs (toutes celles négatives). En supprimant une partie des données, ReLU permet également d'accélérer les calculs. En ne modifiant pas les données positives, ReLU n'impacte pas les caractéristiques misent en évidence par la convolution, au contraire elle les met davantage en évidence en creusant l'écart (valeurs négatives) entre deux caractéristiques. Celle-ci est certainement la fonction la plus utilisée dans les réseaux CNN profonds **[32],** car elle permet une optimisation plus facile. Elle a pour avantage de fournir des réponses parcimonieuses, ne sature pas et permet de réduire les problèmes de disparition de gradient. La fonction ReLU renvoie un gradient constant pour une entrée, permettant ainsi d'apprendre plus rapidement (en particulier les réseaux d'une certaine profondeur) **[38].**

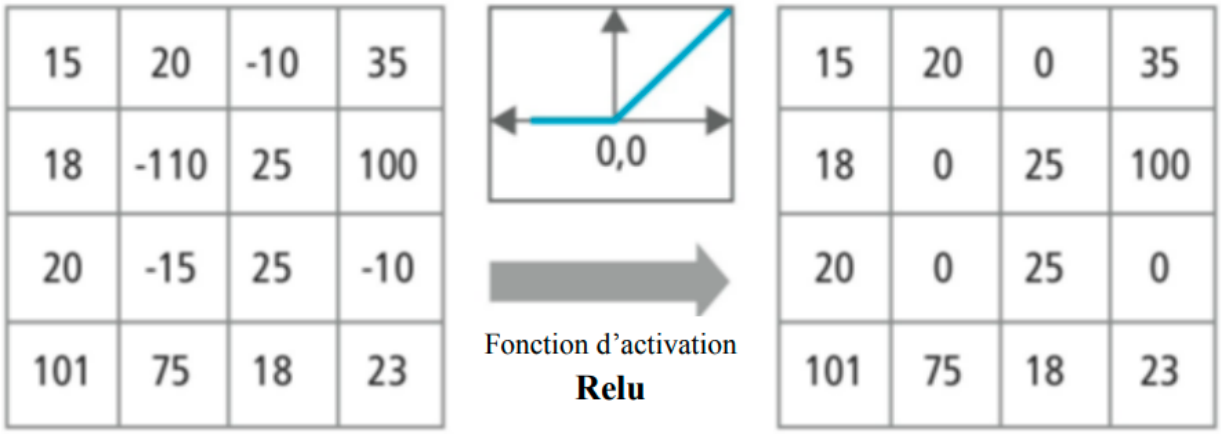



***Figure* 2.7**- *Exemple de fonctionnement d'une couche Relu. (Tous les nombres négatifs valorisés dans la case de gauche ont été mis à zéro après la fonction de d'activation appliquée, toutes les autres valeurs restent inchangées).*

### 2.1.4 Couche entièrement connectée (Fully Connected Layer (FC))

Après plusieurs couches de convolution et de max-pooling, La couche fully-connected constitue toujours la dernière couche d'un réseau de neurones convolutif. Les neurones dans une couche entièrement connectée ont des connexions vers toutes les sorties de la couche précédente (comme on le voit régulièrement dans les réseaux conventionnels de neurones perceptron multicouches) **[34]**. En outre, la couche entièrement connectée est la dernière couche où la classification est effectuée. Ici, nos images filtrées et réduites passent par une étape dite de "**Flatten**" (ou aplatissement). Cette opération consiste à mettre toutes les données dans un seul vecteur. Ce vecteur permettra la création d'une première couche de neurones entièrement connectée. C'est à dire que chacune des valeurs de ce vecteur sera connectée aux neurones de la première couche du réseau permettant la classification de l'image.

La taille de cette couche dépendra de la taille de la sortie des couches précédentes. Par exemple, si nous avons besoin de dix étiquettes de classe différentes, la taille de cette couche serait de 10. La probabilité pour chacune des classes est calculée sur cette couche. Selon les besoins, les couches entièrement connectées peuvent être modélisées pour la régression ou la classification **[32].**

La forme la plus commune d'une architecture de réseau de neurones convolutifs empile quelques couches Convolution-ReLU, les suit avec des couches Pooling, et répète ce schéma jusqu'à ce que l'entrée soit réduite dans un espace d'une taille suffisamment petite. À ce moment, il est fréquent de placer des couches entièrement connectées (FC) **[33] [34]**. La dernière couche entièrement connectée est reliée vers la sortie (la **figure 2.8** présente l'architectures communément utilisés dans un modèle de réseau de neurones convolutifs).

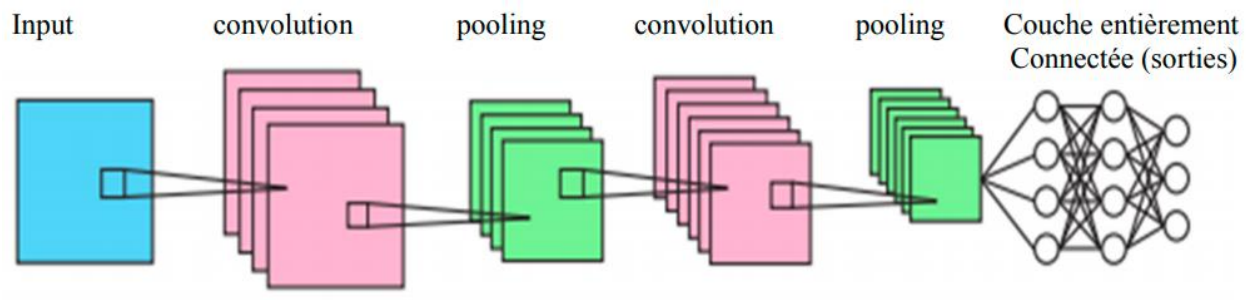



***Fig.2.8**- Architecture standard d'un réseau convolutifs.*

## 2.2 Outils d'optimisation des réseaux convolutifs

Les réseaux de neurones convolutifs utilisent différents paramètres d'optimisation à la différence des réseaux de neurones standard multicouche. Dans cette section, nous nous attachons à décrire des méthodes permettant une meilleure optimisation associée au réseau CNN. La plupart de ces méthodes ont été mises en œuvre lors de la thèse pour le développement et l'apprentissage des réseaux de neurones convolutifs **[38].**

### 2.2.1 La batch normalisation

Dans le cadre d'un apprentissage automatique, l'ordre de présentation des échantillons est très important puisque les paramètres du réseau sont mis à jour après chaque passage d'un échantillon d'apprentissage. Cependant, il n'y a pas de méthodes efficaces pour déterminer à priori quel est l'échantillon qui apportera le plus d'information, sauf peut–être la recherche exhaustive (et donc coûteuse) **[39]**. Une méthode simple consiste donc à mélanger l'ordre de passage des échantillons après chaque passage de l'ensemble d'apprentissage. Ainsi, les échantillons successifs n'appartiennent pas à la même classe, et contiennent donc potentiellement plus d'information **[33]**.

Une technique a été introduite récemment en 2015 **[39]** afin de rendre l'apprentissage des CNN plus rapide et efficace. Elle part de l'observation suivante, pendant l'apprentissage, la distribution des entrées des différentes couches du réseau change à chaque itération. Cela induit une adaptation permanente des paramètres du CNN à ces différentes distributions, ce qui augmente le temps d'apprentissage. L'idée de la batch normalisation (**batch_normalization**) **[32]**, est de normaliser les entrées de chaque couche afin que les distributions de celles-ci soient de moyenne nulle et de variance unitaire. Durant l'apprentissage, les couches de batch normalisation utilisent des paramètres (un facteur d'échelle et un biais) permettant d'ajuster cette normalisation, ces paramètres permettent d'appliquer une transformation sur la distribution normalisée. Autrement dit, durant l'apprentissage, si le réseau considère que la distribution normalisée n'est pas adaptée pour une couche donnée, il apprend les paramètres permettant de l'ajuster.

### 2.2.2 Les fonctions de perte

La fonction (ou couche) de perte spécifie comment l'entrainement du réseau pénalise l'écart entre le signal prévu et réel. Il existe plusieurs fonctions de perte (ou fonctions d'objectif) utilisables pour l'apprentissage des réseaux de neurones. Ces fonctions sont dépendantes de la

tâche que le réseau doit effectuer (classification, régression...). Nous listons ici, les fonctions de perte les plus utilisées **[32]**.

• La fonction de perte **Softmax** : On utilise une couche de Softmax pour transformer des scores logistiques calculés dans la couche dense en une distribution probabiliste en sortie. Cette sortie n'est pas uniforme ou proportionnelle. En fait, la couche Softmax est exponentielle et peut élargir les différences. Par exemple, cette fonction peut rapprocher la probabilité d'un résultat de 1 et d'un autre de 0**[w1].**

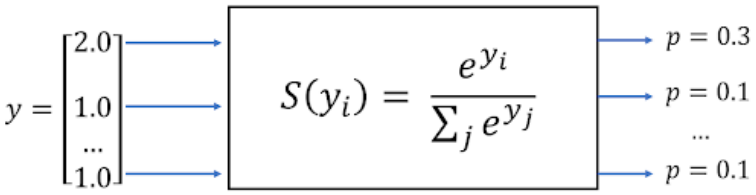



***Fig.2.9*** *La fonction de perte Softmax* ***[w1].***

Par conséquent, le modèle est capable de montrer la distribution de probabilité des émotions faciales.

• La fonction de perte par **entropie croisée** (sigmoïde) : permettant une régression sur des probabilités.

• La perte **quadratique** (squared loss) : Cette fonction calcule les carrés de la différence entre la valeur prédite d'un modèle pour un exemple étiqueté et la valeur réelle de l'étiquette. Une autre fonction associée avec ce type de perte, est l'erreur quadratique moyenne (Mean Squared Error (MSE)) qui est calculée en divisant la perte quadratique par le nombre d'exemples.

### 2.2.3 Méthodes de régularisation

Le nombre d'entrées et de sorties dans un réseaux est généralement détermine par les données d'apprentissage, mais le nombre total de neurones des couches intermédiaires et le nombre de ces couches est un paramètre qui doit être ajusté pour éviter à la fois le sur-apprentissage (overfitting) et le sous apprentissage (underfitting) **[39]**. En apprentissage automatique, la régularisation est un procédé visant à améliorer les performances de généralisation d'un algorithme d'apprentissage, autrement dit à diminuer l'erreur sur les nouvelles données si elles suivent les mêmes lois que celles d'apprentissage. La régularisation est introduite pour réduire l'erreur de généralisation sans perturber l'erreur d'apprentissage. Différents types de régularisation peuvent être envisagées **[39] [40]** :

• **Le dropout :** Pour éviter le sur-apprentissage (overfitting), la couche de dropout a été introduite. Cette couche est utilisée pendant l'apprentissage. Elle permet de désactiver

aléatoirement des neurones durant les différentes itérations de l'apprentissage. Cette manière de faire permet d'apprendre des paramètres plus génériques qui ne se focalisent pas sur des détails de la base d'apprentissage. Une fois l'apprentissage terminé, tous les neurones sont réactivés **[32]**.

• **Early stopping :** consiste à entrainer le réseau en utilisant à la fois une base d'entrainement et une base de test, et à stopper l'entrainement lorsque l'erreur de test se met à ré-augmenter.

• **Augmentation de données :** consiste à augmenter la taille de la base d'apprentissage en lui ajoutant des données obtenues par transformations (ajout de bruit, transformations géométriques, etc.) des données de la base de départ.

• **La régularisation par norme 1 :** la spécificité de cette régulation est de diminuer le poids des entrées aléatoires et faibles et d'augmenter le poids des entrées « importantes ». Le système devient moins sensible au bruit **[32]**.

• **La régularisation par norme 2 :** (norme euclidienne) La spécificité de cette régulation est de diminuer le poids des entrées fortes, et de forcer le neurone à plus prendre en compte les entrées de poids faible **[32] [33].**

## 2.3 Les architectures neuronales convolutifs

Nous présentons ici les architectures de réseaux convolutifs profonds utilisées couramment dans les travaux de recherche en vision par ordinateur, reconnaissance de formes et notamment dans des taches d'identification faciale. Il est important de remarquer que les architectures proposées dans la littérature ont une forte tendance à devenir de plus en plus profonde avec les années. Autrement dit, il semble que plus le réseau est profond, plus les performances sont meilleures. Néanmoins, cette profondeur implique de faire face à certaines difficultés notamment en termes de temps de calcul et d'optimisation durant l'apprentissage. C'est pourquoi la communauté reste très active sur la problématique de conception d'architectures neuronales **[32] [33].** Plusieurs architectures CNN ont déjà été proposées, certaines seront décrites dans cette partie.

**-LeNet (1998) :** LeCun **[36]** a été le premier à mettre en œuvre avec succès une application d'un réseaux CNN, utilisé pour la reconnaissance de l'écriture manuscrite. Il se compose de sept couches, sans compter la couche d'entrée. Les images d'entrée utilisées étaient de taille 32×32. La première couche est composée de six filtres 5×5, qui après la convolution amènent la taille à 28×28. Après la convolution vient une couche de sou échantillonnage max-pooling, puis seize autres filtres 5×5 pour la deuxième couche de convolution, suivie de la dernière couche de sous-échantillonnage avant d'être projetés dans une couche entièrement connectée .

**-AlexNet (2012) :** Cette architecture proposée par Krizhevsky et al **[30]**. C'est l'un des premiers travaux à avoir popularisé les réseaux convolutifs en informatique, grâce à la victoire lors de la compétition de classification d'images ImageNet. Cette architecture utilise cinq couches de convolution et trois couches de pooling. La taille des noyaux de convolution est variable (11×11, 5×5, 3×3) en fonction de la couche considérée. La fonction d'activation utilisée entre chaque couche est la fonction ReLU. Après le passage de l'image dans les couches de convolution, de

pooling et d'activation, une carte de caractéristiques est obtenue. Celle-ci est envoyée dans un perceptron multicouches composé de deux couches cachées et d'une couche de sortie **[32].**

**-VGGNet (2014) :** Ce CNN a été introduit par Simonyan et al **[42].** Au lieu d'utiliser une seule convolution par niveau de profondeur comme AlexNet, cette architecture utilise des séquences de convolution. De plus, les filtres convolutifs sont de plus petite taille que dans AlexNet (noyau de taille 3 × 3) **[32].**

**-GoogLeNet (2015) :** Cette architecture de CNN a été créée par Szegedy et al **[43],** elle permet une réduction du temps de calcul par rapport à l'architecture VGG présentée précédemment. Pour cela, le GoogLeNet est composé de plusieurs couches appelées couches d'inception. Elles sont composées de plusieurs modules de convolution exécutés en parallèle sur la carte de caractéristiques résultant de la couche précédente d'autres modules d'inception ont par la suite été proposés notamment Inception V2 et V3 **[32].**

**-ResNet (2016) :** Ce type de réseau CNN (réseau résiduel) permet l'apprentissage de réseaux très profonds (plus de 150 couches) **[32].** L'idée développée dans ResNet est l'utilisation de connexions résiduelles permettant une meilleure optimisation des réseaux très profonds. Une connexion résiduelle permet de passer l'entrée dans deux filtres de convolution mais également de passer directement cette entrée aux couches suivantes.

## 3 Conclusion :

Dans ce chapitre, nous avons détaillé le CNN a travers ces quatre principales opérations : convolution, la fonction non-linéarité (ReLU), Pooling et couche entièrement connectée. De plus, nous avons expliqué les architectures de CNN utilisés.

Dans le chapitre suivant nous allons présenter notre conception ainsi que la réalisation de notre système avec l'étude comparative.

# CHAPITRE 03 : Implémentation et résultats expérimentaux

# 1 Introduction

L'objectif de ce chapitre est de présenter les étapes de l'implémentation de notre approche dans le cadre d'un système de reconnaissance des émotions. Nous commençons tout d'abord par la présentation des ressources, du langage et de l'environnement de développement que nous avons utilisé. Puis les étapes de la réalisation du modèle et on termine par les tests effectués.

# 2 Environnement de développement

## 2.1 Spyder(anaconda)

Spyder est un puissant environnement scientifique écrit en Python, pour Python, et conçu par et pour des scientifiques, des ingénieurs et des analystes de données, Spyder offre une intégration intégrée avec de nombreux packages scientifiques populaires, notamment NumPy, Pandas, IPython, QtConsole, Matplotlib, etc. Au-delà de ses nombreuses fonctionnalités intégrées, Spyder peut être étendu encore plus via des plugins tiers**[w7]**, des applications d'apprentissage approfondi à l'aide de bibliothèques populaires telles que Keras, TensorFlow, PyTorch et Open CV.

Spyder présente une combinaison unique des fonctionnalités avancées d'édition, d'analyse, de débogage et de profilage d'un outil de développement complet avec l'exploration de données, l'exécution interactive, l'inspection approfondie et les belles capacités de visualisation d'un package scientifique **[w7].**

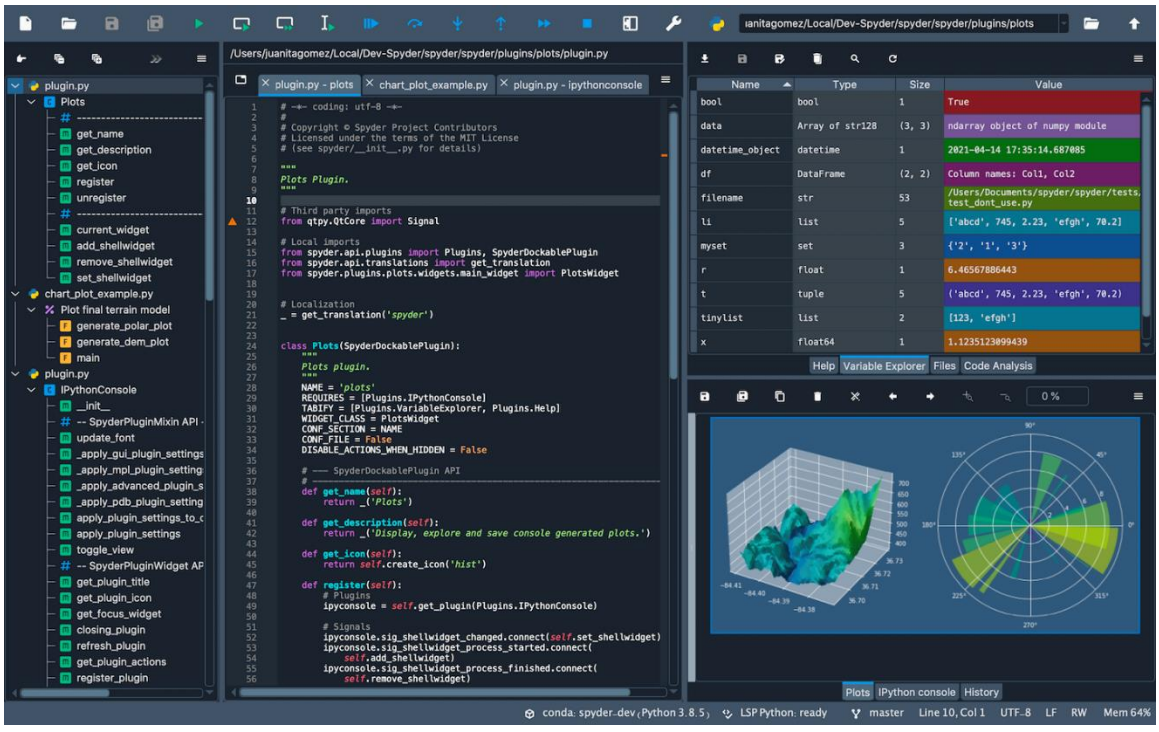


***Figure 3.1*** *Environnement de Spyder* ***[w8].***

### 1.2 Python

Python est un langage de programmation de haut niveau utilisé pour la programmation générale. Créé par Guido van Rossum et sorti en 1991, Python a une philosophie de conception qui met l'accent sur la lisibilité du code, notamment en utilisant des espaces importants.

Il fournit des constructions qui permettent une programmation claire à petite et à grande échelle. Python dispose d'un système de type dynamique et d'une gestion automatique de la mémoire. Il prend en charge de multiples paradigmes de programmation, y compris orientés objet, impératifs, fonctionnels et procéduraux, et dispose d'une bibliothèque standard vaste et complète. Les interpréteurs de Python sont disponibles pour de nombreux systèmes d'exploitation **[W9].**

## 3 Bibliothèques utilisées

### 3.1 OpenCV (Open Source Computer Vision Library)

Est une bibliothèque proposant un ensemble de plus de 2500 algorithmes de vision par ordinateur spécialisé dans le traitement d'images, accessible au travers d'API pour les langages C, C++, et Python. Elle est distribuée sous une licence BSD (libre) pour les plateformes Windows, GNU/Linux, Android et MacOS **[W10]**, nous avons utilisé cette bibliothèque pour la détection du visage à partir des images introduites.

### 3.2 Numpy

Est une bibliothèque permettant d'effectuer des calculs numériques avec Python. Elle introduit une gestion facilitée des tableaux de nombres, des fonctions sophistiquées (diffusion), on peut aussi l'intégrer le code C / C ++ et Fortran. **[W11]**

### 3.3 Matplotlib

Est une bibliothèque de traçage pour le langage de programmation Python et son extension mathématique numérique NumPy . Il fournit une API orientée objet permettant d'incorporer des graphiques dans des applications à l'aide de kits d'outils d'interface graphique à usage général tels que Tkinter , wxPython , Qt ou GTK + .

### 3.4 Keras

Keras est une bibliothèque open source écrite en python et permettant d'interagir avec les algorithmes de réseaux de neurones profonds et de machine Learning, notamment Tensorflow et Theano. Elle a été initialement écrite par François Chollet **[W12].**

### 3.5 Pandas

Pandas est une bibliothèque open source sous licence BSD fournissant des structures de données hautes performances et faciles à utiliser, ainsi que des outils d'analyse des données pour le langage de programmation Python. Pandas est un projet sponsorisé par NumFOCUS. Cela contribuera au succès du développement de pandas en tant que projet open source de classe mondiale et permettra de faire un don au projet **[W13].**

# 4 Base de données

## 4.1 FER2013

La base de données d'images des expressions faciales que nous avons eu à utiliser est celle de la base de données de Fer2013 elle contient 35887 images expressions faciales.

Toutes ces images ont été déjà résolues par kaggle et contiennent chacune une taille de (48x48 pixels).

| | emotion | pixels | Usage |
|---|---|---|---|
| **0** | 0 | 70 80 82 72 58 58 60 63 54 58 60 48 89 115 121... | Training |
| **1** | 0 | 151 150 147 155 148 133 111 140 170 174 182 15... | Training |
| **2** | 2 | 231 212 156 164 174 138 161 173 182 200 106 38... | Training |
| **3** | 4 | 24 32 36 30 32 23 19 20 30 41 21 22 32 34 21 1... | Training |
| **4** | 6 | 4 0 0 0 0 0 0 0 0 0 0 0 3 15 23 28 48 50 58 84... | Training |
| **...** | ... | ... | ... |
| **35882** | 6 | 50 36 17 22 23 29 33 39 34 37 37 37 39 43 48 5... | PrivateTest |
| **35883** | 3 | 178 174 172 173 181 188 191 194 196 199 200 20... | PrivateTest |
| **35884** | 0 | 17 17 16 23 28 22 19 17 25 26 20 24 31 19 27 9... | PrivateTest |
| **35885** | 3 | 30 28 28 29 31 30 42 68 79 81 77 67 67 71 63 6... | PrivateTest |
| **35886** | 2 | 19 13 14 12 13 16 21 33 50 57 71 84 97 108 122... | PrivateTest |

35887 rows × 3 columns

***Figure 3.2*** *Vue du bloc de données Fer2013.*

## 4.2 FER2013new

En raison du grand nombre d'images mal étiquettes dans l'ensemble de données **FER2013**, nous avons constaté que l'utilisation des étiquettes **FER2013new** est une meilleure option pour entraîner le modèle afin d'obtenir de meilleures performances.Voici quelques exemples de labels FER2013 vs FER2013new extraits (FER2013 haut, FER2013new bas) :


***Figure 3.3*** *FER2013 VS FER2013new* ***[w4]****.*

Les annotations FER2013new fournissent un ensemble de Le nouveau étiquettes pour l'ensemble de données Emotion FER2013 standard. Dans FER2013new, chaque image a été étiquetée par 10 tagueurs participatifs, qui fournissent une vérité terrain de meilleure qualité pour l'émotion des images fixes que étiquettes FER2013 d'origine. Le fait d'avoir 10 marqueurs pour chaque image permet aux chercheurs d'estimer une distribution de probabilité d'émotion par visage. Cela permet de construire des algorithmes qui produisent des distributions statistiques ou des sorties multi- étiquette au lieu de la sortie conventionnelle à une seule etiquette , grâce à fer2013new, la précision augmente de **3%.**

## 5 Implémentation codage du système

Pour ce faire, nous allons écrire un script python. Nous utiliserons le Spyder dans notre système local :

### 5.1 Formation du model avec keras

- La première des choses à faire consiste à tout d'abord configurer l'environnement

- Ensuite, installer toutes les bibliothèques citez ci-dessus

- Importer les bibliothèques installées (pandas, numpy, matplotlib) voir (**chapitre 3 page 31**).

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

- Ensuite, nous allons importer le type de modèle séquentiel de Keras. Il s'agit simplement d'un empilement linéaire de couches de réseaux de neurones à travers la commande :

```
from keras.models import Sequential,
```

- Nous importerons les couches "principales" de Keras. Voici les couches utilisées dans presque tous les réseaux de neurones : (Dense, Activation, Flatten)

```
from keras.layers import Conv2D, MaxPooling2D, Dense, Activation, Flatten, Dropout, BatchNormalization
```

- Pour un entrainement plus efficace sur les données d'image nous allons importer les couches CNN de Keras. Ce sont les couches convolutives qui nous aideront à former notre modèle ( nvolution2D, MaxPooling2D) :

```
from keras.layers import Conv2D, MaxPooling2D, Dense, Activation, Flatten, Dropout, BatchNormalization
```

De la base de donnée modèle contient :

* (1, 48, 48) qui correspond à la (profondeur, largeur, hauteur) de chaque image de la base de données de Fer2013.

- Le Data Augmentation est utilisée pour créer artificiellement des images, ces images sont ajoutées aux images d'entraînement d'origine pour augmenter la taille totale de l'ensemble d'entraînement, Nous avons implémenté le Data Augmentation avec la classe keras **ImageDataGenerator** et ajusté ses paramètres. Ce faisant, nous avons pu augmenter la précision

du test d'environ ~ **7 %**.

```
from keras.preprocessing.image import ImageDataGenerator
```

L'astuce consistait à ne pas en abuser pour que le modèle puisse encore apprendre des images d'entraînement.

- Batch Normalization and Dropout Layers (Batch Normalization que nous avons ajouté dans l'architecture finale ,on verra ça) :

```
from keras.layers import Conv2D, MaxPooling2D, Dense, Activation, Flatten, Dropout, BatchNormalization
```

## 5.2 Formation du modele

Cette section décrit les expériences réalisées de notre système sur la reconnaissance des emitions faciales. Elle contient deux models (MODELE 1) et (MODELE 2) pour examiner la performance de l'information :

### 5.2.1 • modele (1) :

le modele (1) représente le model initial que nous avons utilisése:

- uniquement une dataset fer2013 que nous avons divise en deux parties (train = 28709 images ,et test = 7178 images ) .
- Construire une structure CNN avec trois couches constituées respectivement de : filtre convolutif, ReLu et de regroupement maximum (Max Pooling). Cette couche est répétée 3 fois , voir la **figure** (**2.1)** dans (le **chapiter 2 page 18**).
  l'architecture du modèle sera :
  [2 x CONV (3x3)] — MAXP (2x2) DROPOUT (0.25)
  [1 x CONV (3x3)] — MAXP (2x2)
  [1 x CONV (3x3)] — MAXP (2x2) DROPOUT (0.25)
  Dense (1024) — DROPOUT (0.5)

```
# Create the model (1)
def define_model(input_shape=(48, 48, 1), classes=7):
    num_features = 64
    model = Sequential()
    model.add(Conv2D(num_features / 2 , kernel_size=(3, 3), activation='relu', input_shape=input_shape))
    model.add(Conv2D(num_features, kernel_size=(3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))

    model.add(Conv2D(2 * num_features, kernel_size=(3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Conv2D(2 * num_features, kernel_size=(3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))

    model.add(Flatten())
    model.add(Dense(1024, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(classes, activation='softmax'))
    return model
```

***Figure 3.4*** *concevoir le modèle CNN pour la détection des émotions avec différentes couches (model 1).*

```
Found 28709 images belonging to 7 classes.
Found 7178 images belonging to 7 classes.
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 46, 46, 32)        320
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 44, 44, 64)        18496
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 22, 22, 64)        0
_________________________________________________________________
dropout (Dropout)            (None, 22, 22, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 20, 20, 128)       73856
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 10, 10, 128)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 8, 8, 128)         147584
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 4, 4, 128)         0
_________________________________________________________________
dropout_1 (Dropout)          (None, 4, 4, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 2048)              0
_________________________________________________________________
dense (Dense)                (None, 1024)              2098176
_________________________________________________________________
dropout_2 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_1 (Dense)              (None, 7)                 7175
=================================================================
Total params: 2,345,607
Trainable params: 2,345,607
Non-trainable params: 0
```

***Figure 3.5*** *Résultat du modèle (1) obtenu.*

### 5.2.2 • modele 2:

le modele (2) représente le model finale que nous avons utilisése:

- Augmente des étiquettes **FER2013new** avec le dataset **fer2013** , Le nouveau fichier d'etiquette est nommé fer2013new.csv et contient le même nombre de lignes que le fichier d'étiquettes d'origine fer2013.csv avec le même ordre, de sorte que vous puissiez en déduire quelle label d'émotion appartient à quelle image,

```python
def preprocess_data():
    data = pd.read_csv('fer2013.csv')
    labels = pd.read_csv('fer2013new.csv')
```

***Figure 3.6*** *Fonction pour telecharger les dataset (FER2013, FERPlus).*

nous avons divise en 3 parties (**train** ,**validation** et **test** ) .

```
def split_data(X, y):
    test_size = ceil(len(X) * 0.1)

    # Split Data
    x_train, x_test, y_train, y_test = model_selection.train_test_split(X, y, test_size=test_size, random_state=42)
    x_train, x_val, y_train, y_val = model_selection.train_test_split(x_train, y_train, test_size=test_size,
                                                                      random_state=42)
    return x_train, y_train, x_val, y_val, x_test, y_test
```

***Figure 3.7*** *Fonction pour divise dataset en 3 parties (train, validation et test).*

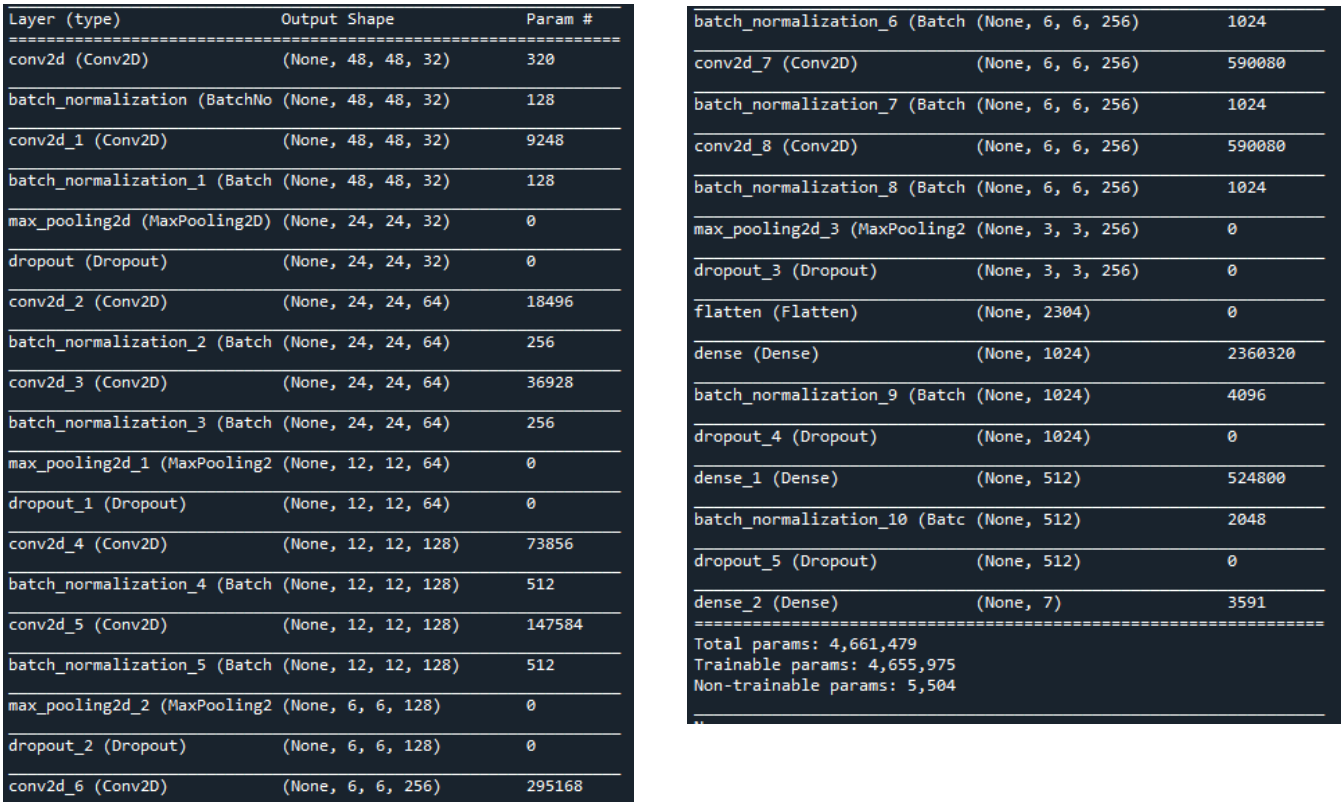
- Nous avons ajouté plusieurs couches dans le modelee (2) l'architecture du modèle sera :

[2 x CONV (3x3)] — DROPOUT (0.5)
[2 x CONV (3x3)] — MAXP (2x2)
[2 x CONV (3x3)]
[2 x CONV (3x3)] — MAXP (2x2)
[2 x CONV (3x3)]
Dense (1024) — DROPOUT (0.2)
Dense (1024) — DROPOUT (0.2)

- Dans toutes les couches convolutives, la couche Batch_Normalization a été ajoutée, il applique une transformation qui maintient la sortie moyenne proche de 0 et l'écart type de sortie proche de 1, ce qui rend la formation plus rapide et plus stable ,

```python
def define_model(input_shape=(48, 48, 1), classes=7):
    num_features = 64
    model = Sequential()

    model.add(Conv2D(num_features, kernel_size=(3, 3), input_shape=input_shape))
    model.add(BatchNormalization())
    model.add(Activation(activation='relu'))
    model.add(Conv2D(num_features, kernel_size=(3, 3)))
    model.add(BatchNormalization())
    model.add(Activation(activation='relu'))
    model.add(Dropout(0.5))

    model.add(Conv2D(num_features, (3, 3), activation='relu'))
    model.add(Conv2D(num_features, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

    model.add(Conv2D(2 * num_features, kernel_size=(3, 3)))
    model.add(BatchNormalization())
    model.add(Activation(activation='relu'))
    model.add(Conv2D(2 * num_features, kernel_size=(3, 3)))
    model.add(BatchNormalization())
    model.add(Activation(activation='relu'))

    model.add(Conv2D(2 * num_features, (3, 3), activation='relu'))
    model.add(Conv2D(2 * num_features, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

    model.add(Conv2D(4 * num_features, kernel_size=(3, 3)))
    model.add(BatchNormalization())
    model.add(Activation(activation='relu'))
    model.add(Conv2D(4 * num_features, kernel_size=(3, 3)))
    model.add(BatchNormalization())
    model.add(Activation(activation='relu'))

    model.add(Flatten())
    # Fully connected neural networks
    model.add(Dense(1024, activation='relu'))
    model.add(Dropout(0.2))
    model.add(Dense(1024, activation='relu'))
    model.add(Dropout(0.2))
    model.add(Dense(classes, activation='softmax'))

    return model
```

***Figure 3.8*** *Concevoir le modèle CNN pour la détection des émotions avec différentes couches.*



| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d (Conv2D) | (None, 48, 48, 32) | 320 |
| batch_normalization (BatchNo | (None, 48, 48, 32) | 128 |
| conv2d_1 (Conv2D) | (None, 48, 48, 32) | 9248 |
| batch_normalization_1 (Batch | (None, 48, 48, 32) | 128 |
| max_pooling2d (MaxPooling2D) | (None, 24, 24, 32) | 0 |
| dropout (Dropout) | (None, 24, 24, 32) | 0 |
| conv2d_2 (Conv2D) | (None, 24, 24, 64) | 18496 |
| batch_normalization_2 (Batch | (None, 24, 24, 64) | 256 |
| conv2d_3 (Conv2D) | (None, 24, 24, 64) | 36928 |
| batch_normalization_3 (Batch | (None, 24, 24, 64) | 256 |
| max_pooling2d_1 (MaxPooling2 | (None, 12, 12, 64) | 0 |
| dropout_1 (Dropout) | (None, 12, 12, 64) | 0 |
| conv2d_4 (Conv2D) | (None, 12, 12, 128) | 73856 |
| batch_normalization_4 (Batch | (None, 12, 12, 128) | 512 |
| conv2d_5 (Conv2D) | (None, 12, 12, 128) | 147584 |
| batch_normalization_5 (Batch | (None, 12, 12, 128) | 512 |
| max_pooling2d_2 (MaxPooling2 | (None, 6, 6, 128) | 0 |
| dropout_2 (Dropout) | (None, 6, 6, 128) | 0 |
| conv2d_6 (Conv2D) | (None, 6, 6, 256) | 295168 |
| batch_normalization_6 (Batch | (None, 6, 6, 256) | 1024 |
| conv2d_7 (Conv2D) | (None, 6, 6, 256) | 590080 |
| batch_normalization_7 (Batch | (None, 6, 6, 256) | 1024 |
| conv2d_8 (Conv2D) | (None, 6, 6, 256) | 590080 |
| batch_normalization_8 (Batch | (None, 6, 6, 256) | 1024 |
| max_pooling2d_3 (MaxPooling2 | (None, 3, 3, 256) | 0 |
| dropout_3 (Dropout) | (None, 3, 3, 256) | 0 |
| flatten (Flatten) | (None, 2304) | 0 |
| dense (Dense) | (None, 1024) | 2360320 |
| batch_normalization_9 (Batch | (None, 1024) | 4096 |
| dropout_4 (Dropout) | (None, 1024) | 0 |
| dense_1 (Dense) | (None, 512) | 524800 |
| batch_normalization_10 (Batc | (None, 512) | 2048 |
| dropout_5 (Dropout) | (None, 512) | 0 |
| dense_2 (Dense) | (None, 7) | 3591 |

Total params: 4,661,479
Trainable params: 4,655,975
Non-trainable params: 5,504

***Figure 3.9*** *Résultat du modèle (2) obtenu.*

## 5.2 Entraînement:

Dans cette partie nous nous intéressons aux résultats d'exécution de l' approche développée .

Une fois que le modèle est formé, nous allons essayer de lui attribuée certains paramètres :

| ***parameters*** | ***Utilisation*** |
|---|---|
| **Epoch** | Désigne le nombre d'itération dans notre base de données. |
| **Optimiser** | Permet de réduire le poids des erreurs. |
| **Loss** | Désigne le taux d'erreur. |
| **Accuracy** | Désigne le taux de précision. |
| **History** | Elle permet de donner au modèle toutes les images dont elle aura besoin en fonction du nombre d'itération (epochs) définit, dans le but de réduire le taux d'erreur. |

***Tableau 3.1*** *Tableau des paramètres utilise.*

Après avoir compilé le modèle, nous ajustons ensuite les données pour l'entraînement et la validation. Ici, nous prenons la taille du batch à 64 avec 100 époques (modele 2) :

Epoch 1/100

443/443 - 1615s - loss: 0.3472 - accuracy: 0.4110 - val_loss: 0.3785 - val_accuracy: 0.3891

: : :

: : :

Epoch 100/100

443/443 - 1683s - loss: 0.0569 - accuracy: 0.8960 - val_loss: 0.1971 - val_accuracy: 0.8321

56/56 [==============================] - 40s 639ms/step - loss: 0.0569 - accuracy: 0.8357

***Figure 3.10*** *Résultat de l'entrainement du modèle 2.*

## 5.3 Résultats expérimentaux et analyse de performance

Cette section décrit les résultats expérimentaux de notre système sur la reconnaissance des emitions faciales, Tracer la précision du modèles (modele 1 et modele 2) entraîné est toujours la première étape pour analyser les performances du modèle. Voici deux images illustrant la différence de performances entre les architectures initiales et l'architecture finale.

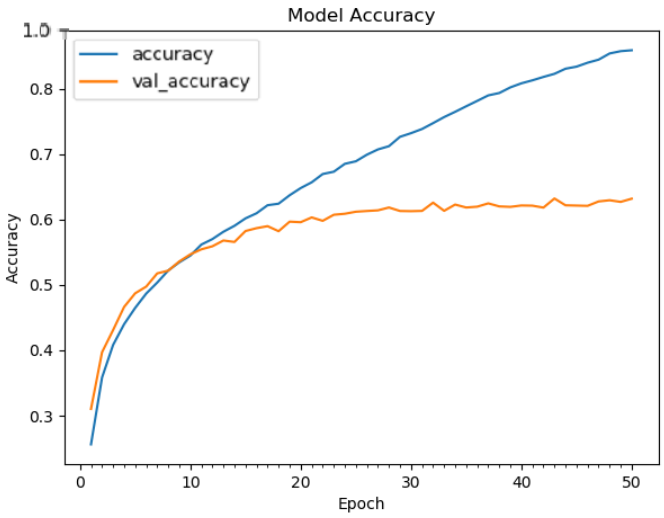



***Figure3.11*** *Le taux de précision initiale (modele 1).*

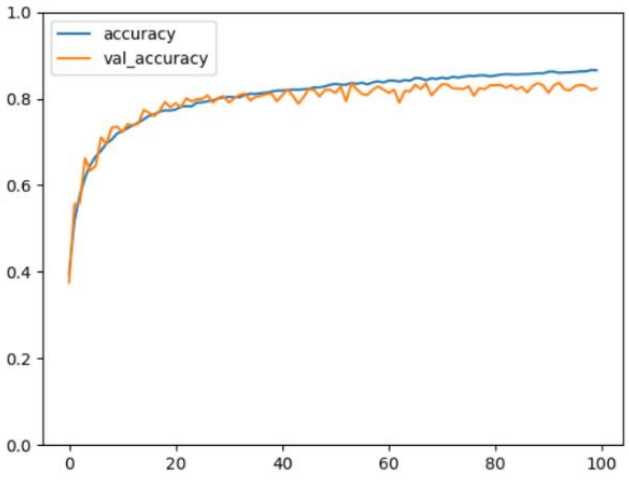



***Figure3.12*** *Le taux de précision finale (modele 2).*

### 5.4.1 Discussion 1

La courbe sur la gauche est pour notre architecture initiale (modele 1), nous pouvons voir que le modèle a commencé à surajuster dans les premières époques, ce qui signifiait que soit ce modèle n'était pas le mieux adapté à l'ensemble de données, soit que l'ensemble de données lui-même n'était pas suffisant pour le modèle pour apprendre suffisamment de caractéristiques pour pouvoir prédire avec une grande précision.

D'un autre côté, la courbe de droite (modele 2) montre que la précision (accuracy) de la validation (val_accuracy) croisée suivait la précision de l'entraînement jusqu'aux 80s, ce qui est un bon signe et c'est certainement une amélioration des performances par rapport à celle de gauche.

```
def plot_acc_loss(history):
    # Plot accuracy graph
    plt.plot(history.history['accuracy'], label='accuracy')
    plt.plot(history.history['val_accuracy'], label='val_accuracy')
    plt.xlabel('Epoch')
    plt.ylabel('accuracy')
    plt.ylim([0, 1.0])
    plt.legend(loc='upper left')
    plt.show()

    # Plot loss graph
    plt.plot(history.history['loss'], label='loss')
    plt.plot(history.history['val_loss'], label='val_loss')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    # plt.ylim([0, 3.5])
    plt.legend(loc='upper right')
    plt.show()
```

***Figure 3.13*** *Affichage du Taux d'erreurs et Taux de précision.*

### 5.4.2 Architecture

Notre architecture finale (modele 2 ) avait une précision de test d'environ **84 %**. L'architecture est une combinaison de ces 3 blocs :

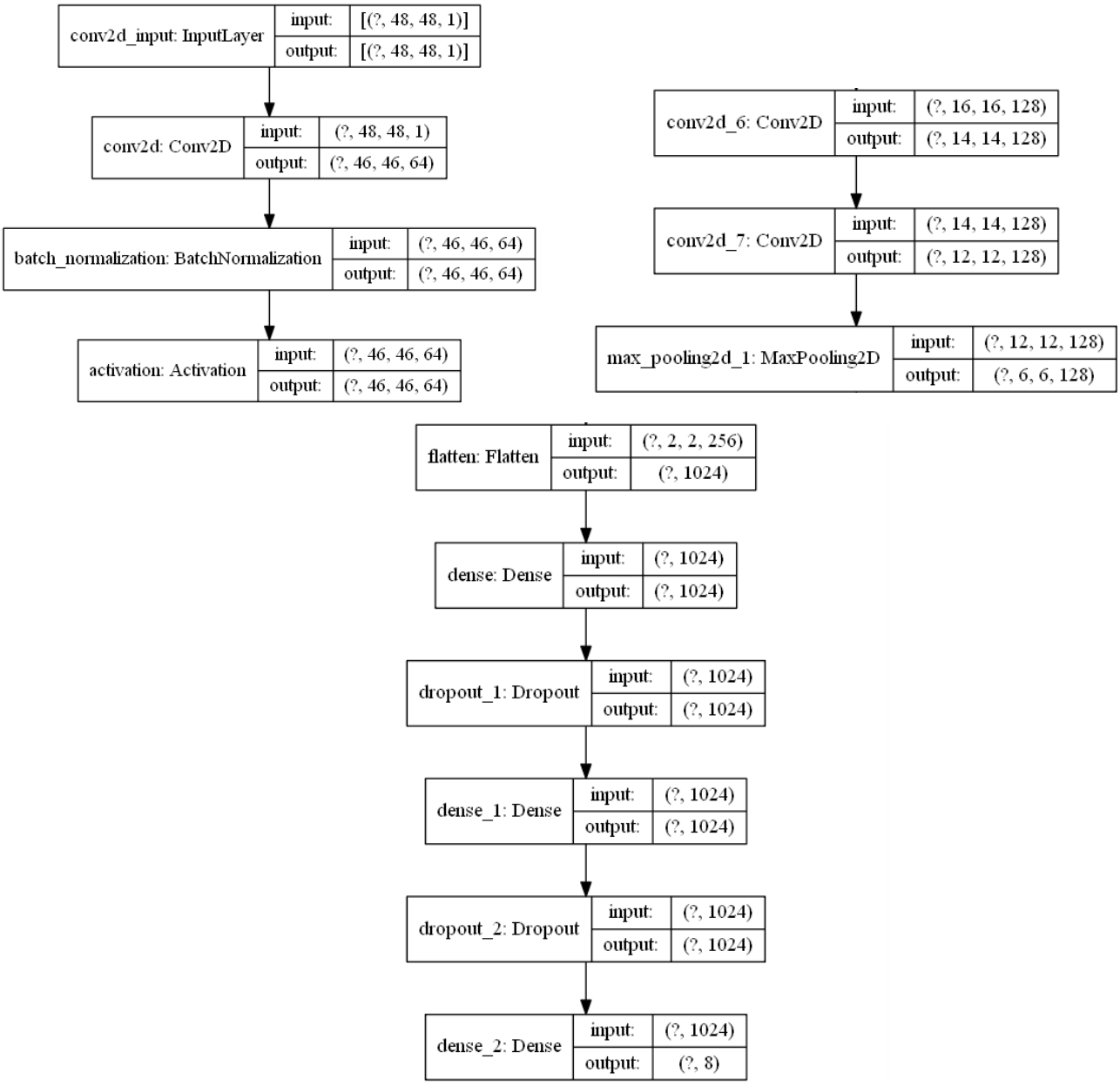



***Figure 3.14*** *les composons de L'architecture (modèle 2).*

Cependant, dépendre uniquement de la précision (accuracy) et de la perte (loss) du modèle entraîné ne donne pas toujours une compréhension complète des performances du modèle.

### 5.4.3 Matrice de confusion

Il existe des mesures plus avancées qui peuvent être utilisées comme le score F1 que nous avons décidé d'utiliser. Le score F1 est calculé à l'aide de deux métriques pré-calculées : la précision et le rappel. Ces deux mesures utilisent les exemples prédits vrais positifs, faux positifs et faux négatifs qui sont mieux visualisés à l'aide de la matrice de confusion.

Puisque nous avons conçu notre modèle pour reconnaître les 7 émotions faciales universelles et que l'ensemble de données **FER2013new** avait une 8e classe pour les émotions de « contempt », nous avons décidé d'ajouter tous les exemples de classe de mépris à la classe « neutre » plutôt que de jeter ces données.

F1 score = 0,8.

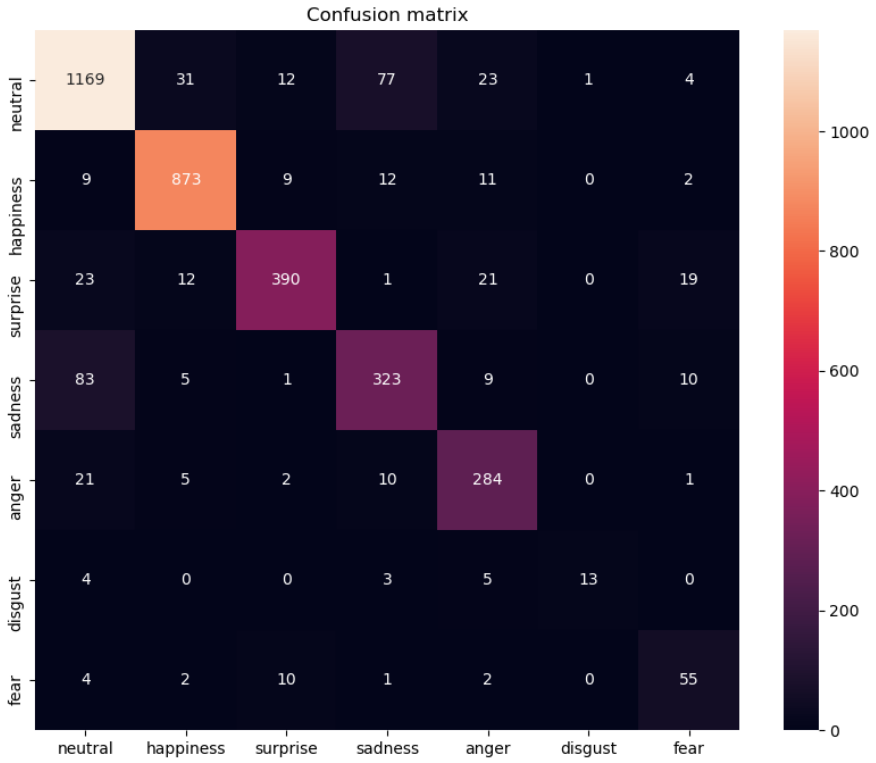



***Figure 3.15*** *matrice de confusion pour les 7 classes, l'axe X est pour les étiquettes prédites et l'axe Y est pour les vraies.*

# 6 Test:

## 6.1 Tester le modèle en temps réel avec OpenCV et WebCam:

Nous allons maintenant tester le modèle que nous construisons pour la détection des émotions en temps réel à l'aide d'OpenCV et d'une webcam. Nous allons installer quelques bibliothèques nécessaires :

```
import cv2
import numpy as np
from keras.models import model_from_json
from keras.preprocessing import image
```

***Figure 3.16*** *Avoir importation les bibliothèques.*

Après avoir importé toutes les bibliothèques requises, nous chargerons les poids du modèle que nous avons enregistrés plus tôt après la formation. Utilisez le code ci-dessous pour charger votre modèle enregistré "**blocks_4.h5**". Après avoir importé les poids du modèle, nous avons importé un fichier en "***haar cascade"*** conçu par open cv pour détecter la face frontale :

```
json_file = open('blocks_4.json', 'r')
loaded_model_json = json_file.read()
json_file.close()
model = model_from_json(loaded_model_json)

model.load_weights('blocks_4.h5')

face_haar_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
```

***Figure 3.17*** *Chargement les poids du modèle et importé un fichier haar cascade.*

Après avoir importé le fichier haar cascade nous aurons écrit un code pour détecter les visages et classer les émotions souhaitées pour chaque frame respectivement. Nous avons attribué les étiquettes qui seront différentes émotions comme (colère, heureuse, triste, surprise, neutre, dégoute, peureux). Dès que vous exécutez le code, une nouvelle fenêtre apparaîtra et votre webcam s'allumera.

Il détectera ensuite le visage de la personne, tracera un cadre de délimitation sur la personne détectée, puis convertira l'image RVB en niveaux de gris et la classera en temps réel. Veuillez vous référer au code ci-dessous pour les mêmes et exemples de sorties qui sont affichés dans les images. Pour arrêter le code, vous devez appuyer sur « q ».

```python
cap = cv2.VideoCapture(0)

while True:
    ret, img = cap.read()
    if not ret:
        break

    gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces_detected = face_haar_cascade.detectMultiScale(gray_img, 1.1, 6, minSize=(150, 150))

    for (x, y, w, h) in faces_detected:
        cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), thickness=2)
        roi_gray = gray_img[y:y + w, x:x + h]
        roi_gray = cv2.resize(roi_gray, (48, 48))
        img_pixels = image.img_to_array(roi_gray)
        img_pixels = np.expand_dims(img_pixels, axis=0)
        img_pixels /= 255.0

        predictions = model.predict(img_pixels)
        max_index = int(np.argmax(predictions))

        emotions = ['neutre', 'heureuse', 'surprise', 'triste', 'colère', 'degoute', 'peureux']
        predicted_emotion = emotions[max_index]

        cv2.putText(img, predicted_emotion, (int(x), int(y)), cv2.FONT_HERSHEY_SIMPLEX, 0.75, (255, 255, 255), 2)

        resized_img = cv2.resize(img, (1000, 700))
        cv2.imshow('Facial Emotion Recognition', resized_img)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
```

***Figure 3.18*** *Exécution la reconnaissance d'émotions en temps réel.*

## 6.2 le modèle en video.mp4 :

Après avoir importé les poids du modèle, nous avons importé un fichier .mp4 (vidéo.mp4), une nouvelle fenêtre s'ouvrira et votre vidéo s'allumera. On fait le même travail qu'avant en webcam en temps réel jusque a la fin de la vidéo pour chaque frame respectivement :

```python
cap = cv2.VideoCapture('C:\\Users\\allaw\\.spyder-py3\\Facial-emotion-recognition-master\\test.mp4')
if (cap.isOpened() == False):
    print('error opening')
#cap = cv2.VideoCapture(args['video'])

while(cap.isOpened()):
    ret, img = cap.read()

    if not ret:
        break

    gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces_detected = face_haar_cascade.detectMultiScale(gray_img, 1.2, 6)
    for (x, y, w, h) in faces_detected:
        cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), thickness=2)
        roi_gray = gray_img[y:y + w, x:x + h]
        roi_gray = cv2.resize(roi_gray, (48, 48))
        img_pixels = image.img_to_array(roi_gray)
        img_pixels = np.expand_dims(img_pixels, axis=0)
        img_pixels /= 255.0
        predictions = model.predict(img_pixels)
        max_index = int(np.argmax(predictions[0]))

        emotions = ['neutre', 'heureuse', 'surprise', 'triste', 'colere', 'degoute', 'peureux']
        predicted_emotion = emotions[max_index]
        cv2.putText(img, predicted_emotion, (int(x), int(y)), cv2.FONT_HERSHEY_SIMPLEX, 0.75, (255, 100, 0), 2)

    resized_img = cv2.resize(img, (800, 468))
    cv2.imshow('Facial Emotion Recognition sur le video', resized_img)
```

***Figure 3.19*** *Insertions des vidéos externes et applique le test.*

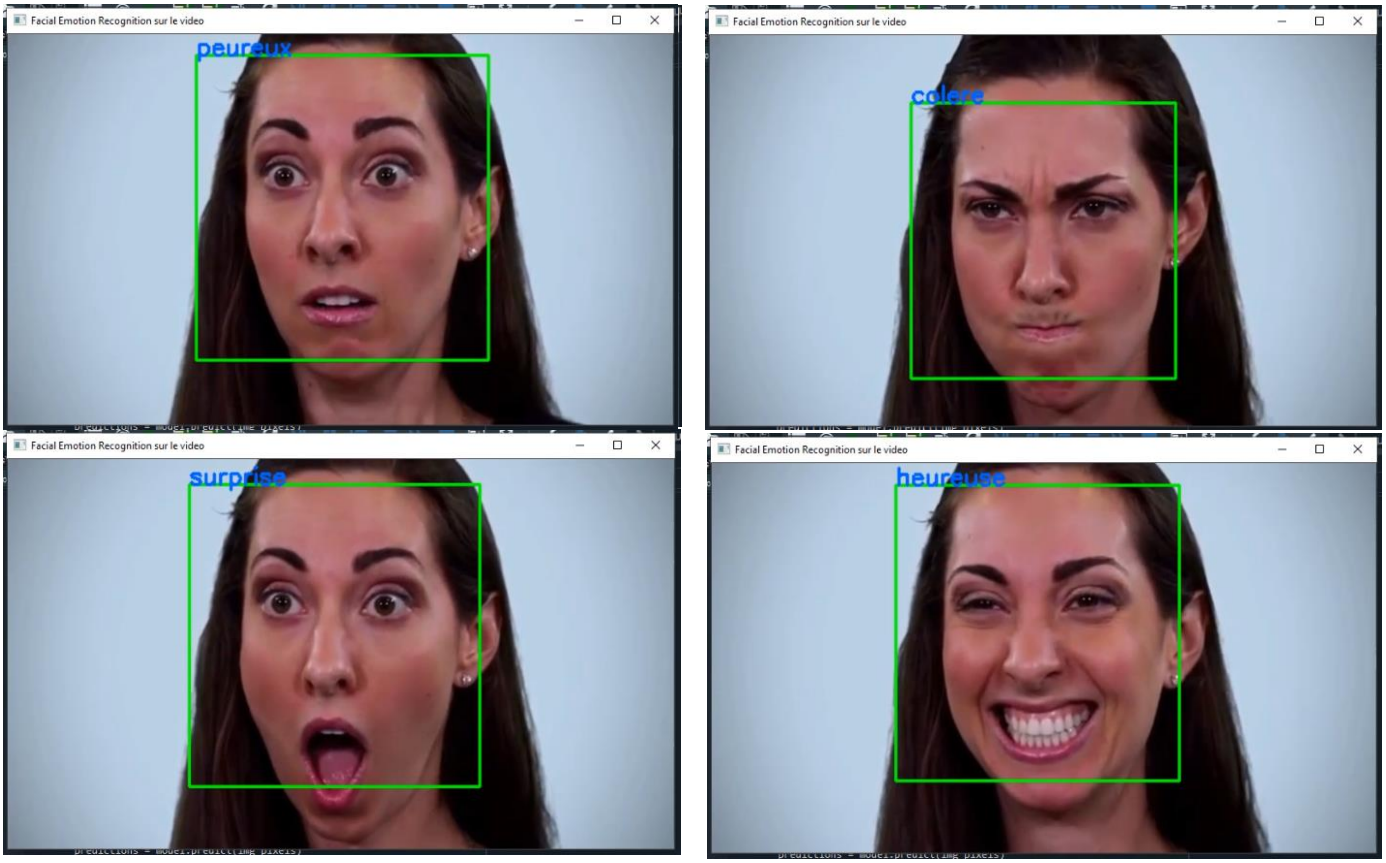



*Figure 3.20 Résultats d'une vidéo externe.*

## 6.3 le modèle en image.png :

Importé le fichier image.png, une nouvelle fenêtre s'ouvrira et votre photo s'allumera avec les visages détectés et les émotions sélectionnées pour chaque visage (un seul frame) :

```
img_path = os.path.abspath('C:\\Users\\allaw\\Desktop\\22.png')
img = cv2.imread(img_path)
gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
faces_detected = classifier.detectMultiScale(gray_img, 1.18, 5)

for (x, y, w, h) in faces_detected:
    cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)
    roi_gray = gray_img[y:y + w, x:x + h]
    roi_gray = cv2.resize(roi_gray, (48, 48))
    img_pixels = image.img_to_array(roi_gray)
    img_pixels = np.expand_dims(img_pixels, axis=0)
    img_pixels /= 255.0

    predictions = model.predict(img_pixels)
    print(model.summary())
    max_index = int(np.argmax(predictions))

    emotions = ['neutre', 'heureuse', 'surprise', 'triste', 'colere', 'degoute', 'peureux']
    predicted_emotion = emotions[max_index]
    cv2.putText(img, predicted_emotion, (int(x), int(y)), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 2)

resized_img = cv2.resize(img, (1500, 658))
cv2.imshow('Facial Emotion Recognition sur image', resized_img)
```

***Figure 3.21*** *Insertions des images externes et applique le test.*

Considérez ceci comme un exemple de la différence entre( modele 1 et modele 2 )

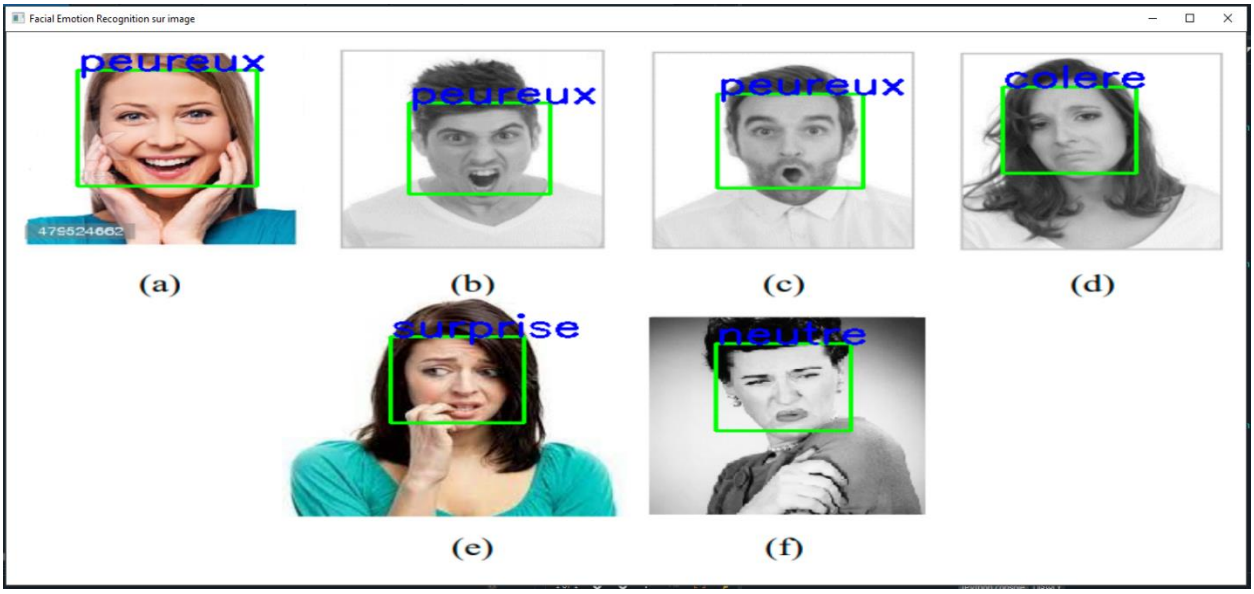



***Figure 3.22*** *Résultats de quelques images dans une seule image externe (modele 1).*

Il y a des images qu'il n'a pas détectées correctement, contrairement au model 2 (FER2013new):

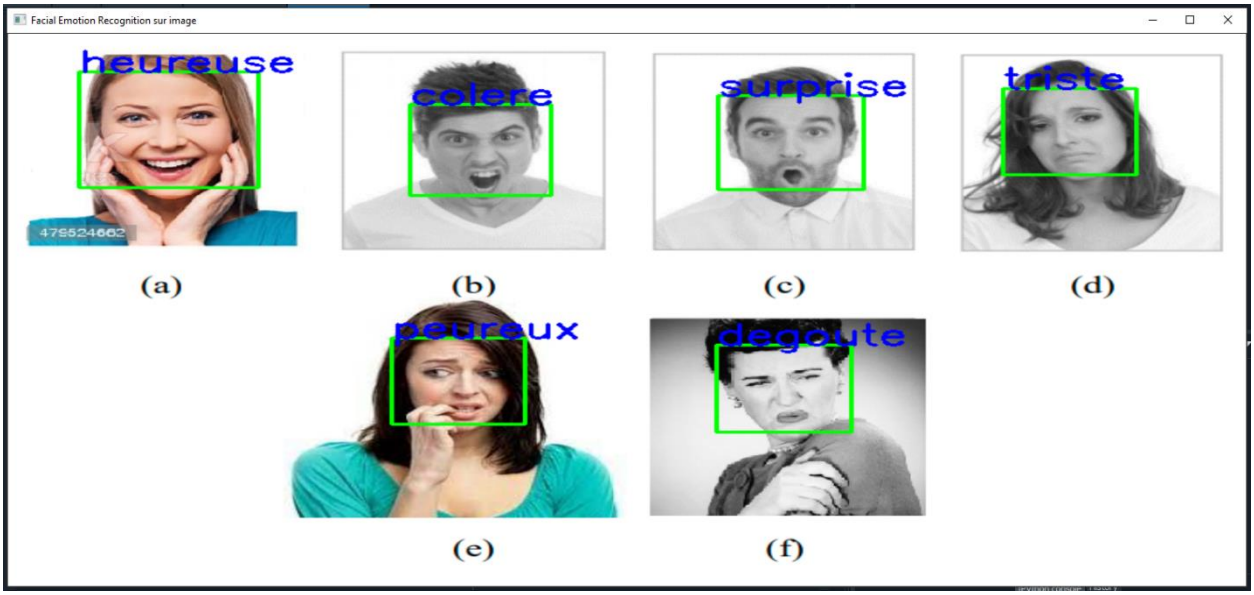
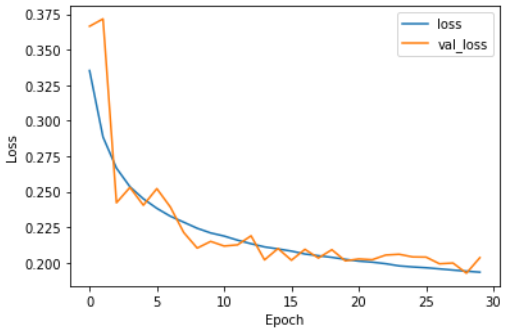


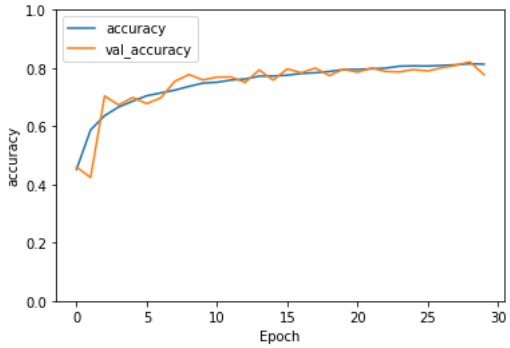

***Figure 3.23*** *Résultats de quelques images dans une seule image externe (modele 2).*

## 7 Discussion 2 :

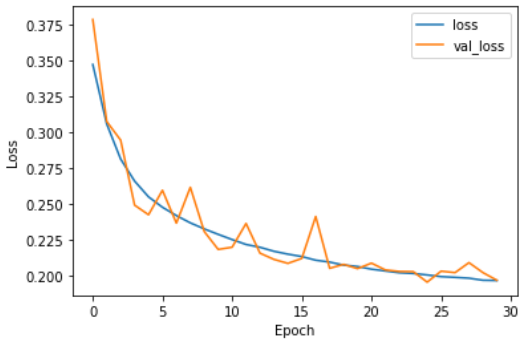
**Expérience :** Pour mieux tester l'efficacité du modèle développé dans ce projet, nous allons essayer de comparer entre le MaxPooling et l'AveragePooling , Pour cela nous avons créé un autre modèle de CNN pour l'AveragePooling , Le modèle entraîne pour 30 époques et afficher les rsultats sous forme plots de (loss , accuracy ) Voir la (Figure 3.24)



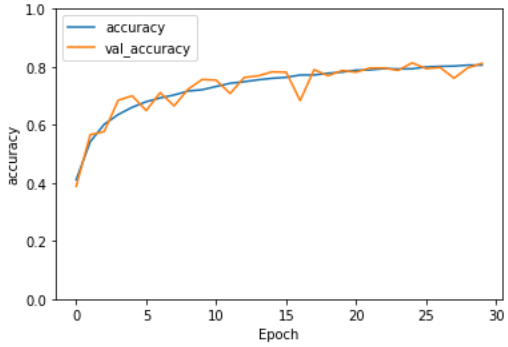

Averagepoolin perte




Averagepooling précision




Maxpooling perte




Maxpooling précision

***Figure 3.24*** *comparaison entre le maxpooling et le averagepooling dans ce travail.*

Nous voyons que les résultats sont similaires , Sauf qu'il y a des plages de diminution au début de la courbe Averagepooling ([0-2 epoch] , [3-4epoch] , ..) Et cela indique un manque de précision. Car lorsque les images sont pauvres, on les mutualise en Averagepooling (Moyenne), la résolution diminue et il est difficile de distinguer les choses remarquables dans l'image.

Contrairement au maxpooling, il détermine les choses remarquables (max) et il a plus de précision.

## 8 Conclusion

Nous avons présenté dans ce chapitre l'implémentation de notre approche de reconnaissance des émotions faciales basée sur un réseau de neurones convolutif, considéré comme extension intéressante de l'approche CNN. En raison de comparaison, nous avons utilisé deux architectures particulières de CNN, la second diffère de la première par le volume de données, le nombre de

couches augmenté et l'organisation des caractéristiques. Puis nous avons comparé leurs performances, Les resultats obtenus sont prometteuses et traduit par le taux de précision de reconnaissance qui peut arriver jusqu'à 84% dans le second architecture par rapport à 66% comme facteur principale.

# Conclusion Générale

Dans ce travail présenté à l'aide de ce mémoire, nous avons met l'accent sur l'importance de la reconnaissance émotionnelle et plus particulièrement les expressions faciales via l'approche Deep Learning qui a connu un grand succès dans le domaine du traitement et de la reconnaissance d'images. En effet, la méthode que nous avons utilisée est le réseau de neurones convolutifs pour la reconnaissance des émotions à l'aide des expressions faciales en temps réel. Pour ceci, nous avons utilisé deux architectures particulières de CNN, la second diffère de la première par le volume de données présenté par la base de données Fer2013new en plus celle Fer2013, le nombre de couches augmenté et l'organisation des caractéristiques. Puis nous avons comparé leurs performances, Les resultats obtenus sont prometteuses et traduit par le taux de précision de reconnaissance qui peut arriver jusqu'à 84% dans le second architecture par rapport à 66% comme facteur principale.

Malgré les difficultés que nous avons u durant l'élaboration de ce travail dans ses deux partie : théorique et pratique, ce travail nous a permis de découvrir un domaines très important présenté par les réseaux de neurones et plus particulièrement le CNN et de le mettre en pratique pour enrichir nos connaissances sur les réseaux de neurones et d'en acquérir d'autres en pratique mais aussi en théorie à travers le temps passé à lire des articles et des mémoires qui nous a servi d'une bonne initiation à la recherche.

Comme perspectives de recherches futures, nous envisageons de :

1. Tester sur notre modèle d'autre base de données autre que celle de Fer2013

2. Augmenter de nouvelle couche afin de voir ce que cava donnée.

3. Travailler avec les modèles déjà entrainés tels que le VGG16 et faire une comparaison.

# Bibliographie

[1] F.Khalfi : Reconnaissance automatique des émotions par données multimodales : expressions faciales et des signaux physiologiques. Autre. Université Paul Verlaine - Metz, 2010.

[2] P. Pal et A.N. Iyer et R.E. Yantorno : Emotion detection from infant facial expressions and cries. Proc. IEEE Int'l Conf. Acoustics, Speech and Signal Processing (ICASSP'06), 2, 2006.

[3] J. Colletta et A. Tcherkassof : Les émotions : cognition, langage et développement. Pierre Mardaga, 2003.

[4] A. Fathallah, L. Abdi et A. Douik : Facial expression recognition via deep learning. In 2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA) (pp. 745-750), 2017.

[5] Kh.Lekdioui : Reconnaissance d'états émotionnels par analyse visuelle du visage et apprentissage machine. Synthèse d'image et réalité virtuelle [cs.GR]. Université Bourgogne Franche-Comté; Université Ibn Tofail. Faculté des sciences de Kénitra, 2018.

[6] C.Padgett et G.Cottrell : Representing face images for emotion classification. In Advances in neural information processing systems, pages 894–900, 1997

[7] C.Qi et M.Li et Q.Wang, H.Zhang et J.Xing et Z.Gao et H.Zhang : Facial expressions recognition based on cognition and mapped binary patterns, 2018.

[8] D.Ghimire et J.Lee et Z. Li et et S.Jeong : Recognition of facial expressions based on salient geometric features and support vector machines. Multimedia Tools and Applications, 76(6), 7921-7946. 2017.

[9] M. J.Lyons et J.Budynek et S.Akamatsu : Automatic classification of single facial images. IEEE transactions on pattern analysis and machine intelligence, 21(12), 1357- 1362. 1999.

[10] M.Lyons et S.Akamatsu et M.Kamachi et J.Gyoba : Coding facial expressions with gabor wavelets. In Proceedings Third IEEE international conference on automatic face and gesture recognition (pp. 200-205). (1998, April)

[11] M. F.Valstar, H.Gunes et M.Pantic : How to distinguish posed from spontaneous smiles using geometric features. In Proceedings of the 9th international conference on Multimodal interfaces (pp. 38-45). ACM. 2007, November.

[12] U.Mlakar, I.Fister, J.Brest et B.Potočnik: Multi-objective differential 2017.

[13] I.Cohen, N.Sebe, A.Garg, L. S.Chen et T. S.Huang : Facial expression recognition from video sequences: temporal and static modeling. Computer Vision and image understanding, 91(1-2), 160-187. 2003.

[14] F.Davoine, B.Abboud et V. M. Dang,: Analyse de visages et d'expressions faciales par modèle actif d'apparence. traitement du signal, 1(3). 2004.

[15] M.Pantic, et L. J.Rothkrantz: Automatic analysis of facial expressions: The state of the art. IEEE Transactions on Pattern Analysis & Machine Intelligence, (12), 1424- 1445. 2000.

[16] Maalej, A., Amor, B. B., et M.Daoudi : Analyse locale de la forme 3D pour la reconnaissance d'expressions faciales. 2011, June.

[17] M.Quinn, G.Sivesind, G.Reis," Real-time Emotion Recognition From Facial Expressions", Stanford University 2015.

[18] H.Ding, S.K.Zhou, R.Chellappa : Facenet2expnet: Regularizing a deep face recognition net for expression recognition. In 2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017) (pp. 118-126). 2017, May.

[19] D.Dian: La reconnaissance des expressions faciales, Université 8 Mai 1945– Guelma-, Système Informatique, 2019.

[20] D. Y .Moualek : Deep Learning pour la classification des images (Thèse de doctorat), Université Abou Bakr Belkaid– Tlemcen. 2017.

[21] R.Collobert et J.Weston : A unified architecture for natural language processing: Deep neural networks with multitask learning. In Proceedings of the 25th international conference on Machine learning (pp. 160-167). 2008, July.

[22] S.Liu, N.Yang, Li et M.Zhou : A recursive recurrent neural network for statistical machine translation. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (pp. 1491-1500). 2014, June .

[23] A.Graves, N.Jaitly : Towards end-to-end speech recognition with recurrent neural networks. In International conference on machine learning (pp. 1764-1772). 2014, January.

[24] M. Paleari, R. Benmokhtar et B. Huet : Evidence theory-based multimodal emotion recognition. Springer-Verlag Berlin Heidelberg 2009 ,(Eds.) : MMM 2009, LNCS 5371, 2009.

[25] T.Ahonen, A.Hadid et M.Pietikäinen : Face recognition with local binary patterns. In European conference on computer vision (pp. 469-481). Springer, Berlin, Heidelberg. 2004, May.

[26] G.Guo, et K.Chan : Face recognition by support vector machines. In Proceedings fourth IEEE international conference on automatic face and gesture recognition (cat. no. PR00580) (pp. 196-201). 2000, March

[27] T. F.Cootes, G. J.Edwards et C. J. Taylor,: Active appearance models. IEEE Transactions on Pattern Analysis & Machine Intelligence, (6), 681-685. 2001.

[28] V.Perlibakas : Face recognition using principal component analysis and loggabor filters. arXiv preprint cs/0605025 evolution for feature selection in facial expression recognition systems. Expert Systems with Applications, 89, 129-137. 2006.

[29] Dung Nguyen M.Sc., B.Sc : Multimodal Emotion Recognition Using Deep Learning Techniques, School of Electrical Engineering and Computer Science Science and Engineering Faculty Queensland University of Technology 2020

[b1-30] A.Krizhevsky , I.Sutskever, et G.E. Hinton : ImageNet classification with deep convolutional neural networks. In Proc. Advances in Neural Information Processing Systems ,2012.

[b2-31] Reseaude neurones convolutifs : Document disponible sur : https://dataanalyticspost.com, Consulté 05/06/2021.

[b3-32] F. Chabot : Analyse fine 2D/3D de véhicules par réseaux de neurones profonds, université Clermont auvergne, France ,2018.

[b4-33] R. M. Ignace : Détection de concepts et annotation automatique d'images médicales par apprentissage profond, Université d'Antananarivo, 2018.

[b5-34] M.K. Benkaddour ,S. Beggari et K. Khamra : Système de reconnaissance de visage par un réseau de neurone convolutionnel, Universite kasdi merbah de ouargla, Algerie,2017.

[b6-35] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard et L. D. Jackel: Handwritten digit recognition with a backpropagation network. NIPS, 1989.

[b7-36] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner : Gradient-based learning applied to document recognition, Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.

[b9- 37] M. Zaltzhendler : A Deep-Learning Convolutional Neural Network Framework for Multiple Sclerosis Lesion Detection and Segmentation in Patient Brain Images, McGill University, Montreal, Canada, November 2015.

[b10-38] R.Lambert .Focus : Le Réseau de Neurones Convolutifs. Document disponible sur : http://penseeartificielle.fr. Consulté le 05/06/2021.

[b11-39] S. Ioffe et C. Szegedy : Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. ICML, 2015.

[b12-40] J. Delon : Introduction aux réseaux de neurones et à l'apprentissage profond. Université paris Descartes, août 2018.

[b13-41] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever et R. Salakhutdinov : Dropout: A Simple Way to Prevent Neural Networks from Overfitting. JMLR, 2014.

[b14-42] K. Simonyan et A. Zisserman : Very Deep Convolutional Networks for Large-Scale Image Recognition. ICLR, 2014.

[b15-43] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich : Going Deeper with Convolutions. CVPR, 2015.

[w1] https://meritis.fr/reconnaissance-des-emotions/,consultée le : 6 june 2021.

[w2] www.fr.wikipedia.org. Consultée le : 15 mai 2021

[w3] https://www.futura-sciences.com/tech/dossiers/robotique-presentation-historiquereseaux-neuronaux-31/ consultée le : 15 mai 2021.

[w4] www.image-net.org consultée le : 12 avril 2021.

[w5] https://wdrfree.com/stock-vector/deep-learning consultée le : 10 june 2021.

[w6] http://villemin.gerard.free.fr/Wwwgvmm/Logique/IAHisto.htm consultée le : 10 june 2021.

[w7] https://anaconda.org/anaconda/spyder consultée le : 19 june 2021.

[w8] https://pypi.org/project/spyder/ consultée le : 19 june 2021.

[w9] https://www.python.org./about/ consultée le : 19 june 2021.

[w10] https://opencv.org/about/ consultée le : 19 june 2021.

[w11] https://numpy.org/ consultée le : 19 june 2021.

[w12] https://fr.wikipedia.org/wiki/Keras consultée le : 19 june 2021.

[w13] https://pandas.pydata.org/ consultée le : 19 june 2021.

[w14]https://medium.com/@reachraktim/emotion-recognition-on-the-fer-dataset-using-pytorch-835ce93d52a5 consultée le : 22 june 2021.