République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieure et de la Recherche Scientifique
Université Abderrahmane Mira
Faculté de la Technologie

جامعة بجاية
Tasdawit n Bgayet
Université de Béjaïa

ATE Department

Département d'Automatique, Télécommunication et d'Electronique

Projet de Fin d'Etudes

Pour l'obtention du diplôme de Master

Filière : Télécommunication / Electronique

Spécialité : Réseaux / Instrumentation

Thème

Deep Learning :

Reconnaissance Faciale

Préparé par :

- Larbaoui Med Lotfi
- Siamer Djamyl

Dirigé par :

Mme S.GHENNAM

Examiné par :

Mr Sadji

Mr Kasmi

Année universitaire : **2023/2024**

Sommaire

Liste des Figures

Liste des Tableaux

Introduction :

La reconnaissance faciale est une technique qui a connu une évolution remarquable au fil du temps, passant d'une idée de science-fiction à une réalité quotidienne intégrée dans de nombreux aspects de notre vie. Initialement conceptualisée avant même l'invention d'Internet, elle a été popularisée par des œuvres de fiction comme Star Trek et Robocop. Aujourd'hui, elle est utilisée pour simplifier la vérification d'identité, créer des identités numériques sécurisées et améliorer l'expérience utilisateur tout en assurant une sécurité sans faille. Les progrès de l'intelligence artificielle, notamment l'apprentissage profond, ont permis d'améliorer considérablement la performance des outils de reconnaissance faciale, les rendant presque infaillibles et à la portée de tous.

L'importance de la reconnaissance faciale réside dans sa capacité à offrir des solutions efficaces pour l'identification et l'authentification des individus, ce qui est crucial dans la lutte contre les actes criminels et le terrorisme. Elle est également utilisée dans la délivrance de documents d'identité et lors des contrôles frontaliers, où la rapidité et la précision sont essentielles. Cependant, cette technologie n'est pas sans controverses, car elle soulève des questions éthiques et de respect de la vie privée, notamment en ce qui concerne la collecte et le stockage d'informations sensibles sur les individus.

L'évolution de la reconnaissance faciale a été marquée par plusieurs étapes clés. En 1964, des chercheurs ont étudié la programmation d'ordinateurs pour reconnaître des visages, ce qui a conduit à la création de la première technologie réussie de reconnaissance faciale en 1991, appelée Eigenfaces. L'essor du « deeplearning » en 2011 a accéléré le développement de cette technologie, permettant aux ordinateurs de sélectionner eux-mêmes les points à comparer et d'apprendre de manière autonome à partir d'un grand nombre d'images. En 2014, Facebook a développé Deepface, un algorithme capable d'identifier les visages avec une précision de 97%, rivalisant avec les performances de l'œil humain.

Malgré les avancées technologiques, la reconnaissance faciale a été longtemps ralentie dans le monde occidental en raison de préoccupations liées à la liberté individuelle. Cependant, la pandémiedu Covid-19 a contribué à une acceptation sociale plus large de cette technique, notamment en raison de son utilité dans le contrôle des mesures sanitaires. En France, par exemple, la reconnaissance faciale est encadrée par une réglementation stricte pour répondre aux inquiétudes concernant le respect de la vie privée.

Par ailleurs, en termes de développement de logiciels de reconnaissance faciale, Python est un choix populaire en raison de sa simplicité, de sa flexibilité et de la richesse de ses bibliothèques d'apprentissage automatique et de vision par ordinateur.

Notre travail aspire à contribuer à une meilleure compréhension de la reconnaissance faciale en tant que technique émergente et à fournir des pistes pratiques pour son développement et son utilisation. Nous nous baserons sur deux hypothèses :

1. **Hypothèse A :** L'intégration de modèles de réseaux neuronaux profonds permettra d'améliorer la performance du système face à des variations dans les images faciales en termes d'éclairage, d'angle de prise de vue et d'expression faciale.
2. **Hypothèse B :** En utilisant des techniques avancées d'apprentissage automatique et de vision par ordinateur, il est possible de développer un système de reconnaissance faciale précis et efficace avec Python, notamment grâce à ses bibliothèques et les outils disponibles dans ses archives.

Dans cette optique, le mémoire est rédigé en trois chapitres. Dans le premier nous exposerons un état de l'art des techniques de reconnaissance faciale, nous présenterons brièvement les principes fondamentaux, les applications actuelles et les défis. Dans le deuxième chapitre nous aborderons les réseaux neuronaux, le « deeplearning » notamment à travers l'architecture CNN, qui sera amplement détaillée. Le troisième chapitre sera consacré à la conception d'un système de reconnaissance faciale, basé sur une architecture CNN de type GoogleNet. Les étapes de son implémentation sous Python seront présentées, ainsi que les phases de son apprentissage et de test, Des résultats seront exposés et commentés, corroborés par des mesures de. Nous finirons par une conclusion axée sur les défis techniques et meilleures pratiques pour les surmonter, ainsi que les impacts potentiels sur la vie privée des systèmes de reconnaissance faciale.

Chapitre 1 : Systèmes de Reconnaissance Faciale

Introduction :

Dans ce chapitre nous allons explorer les différentes étapes de la conception et de la mise en œuvre d'un système de reconnaissance faciale. Nous aborderons les principes fondamentaux, les algorithmes utilisés, les différentes techniques, ainsi que les considérations éthiques et légales associées à cette technologie en pleine expansion.

1.1 Fondements théoriques de la reconnaissance faciale

1.1.1 Principes de base de la reconnaissance faciale

La reconnaissance faciale est une discipline de la vision par ordinateur qui vise à identifier ou vérifier l'identité d'une personne en analysant et en comparant les caractéristiques de son visage. Cette section explore les principes fondamentaux de la reconnaissance faciale, notamment sa définition, ses objectifs et les méthodes traditionnelles utilisées pour identifier les visages.

A Définition et objectifs de la reconnaissance faciale :

La reconnaissance faciale consiste à extraire et à analyser les caractéristiques distinctives d'un visage humain pour identifier une personne ou vérifier son identité. Les objectifs de la reconnaissance faciale peuvent varier en fonction du contexte d'application, notamment la sécurité, la surveillance, la biométrie, l'authentification, et même le divertissement. [1]

B Méthodes traditionnelles de reconnaissance faciale :

Pour ainsi pouvoir élaborer un système de reconnaissance faciale nous devons connaitre les différentes études existantes à ce jour

L'étude géométrique :

L'étude géométrique dans la reconnaissance faciale concerne l'analyse des caractéristiques géométriques du visage d'une personne pour identifier et reconnaître cette personne. Cela implique généralement de mesurer et d'analyser les proportions, les distances et les angles entre différentes parties du visage, telles que les yeux, le nez, la bouche et les contours du visage, elle peut se faire automatiquement comme manuellement.[2]

L'étude globale :

L'étude globale dans la reconnaissance faciale fait référence à l'analyse et à la prise en compte de l'ensemble du visage plutôt que de se concentrer uniquement sur des parties

spécifiques du visage contrairement à l'étude géométrique, dont la méthode la plus répandu sont les visages propre (eigenfaces), elle se basera sur l'utilisation d'algorithme de Fourier, les vecteurs poids, l'approche statistique de Markov, les réseaux neuronaux... etc.[2]

Ces algorithmes peuvent être utilisés pour :

- reconnaître un seul individu sur la connaissance de ce seul individu (on parle alors de reconnaissance 1 à 1 dans une base de données qui contient un seul individu)
- ou la reconnaissance d'un individu parmi tant d'autres sur une base de données (reconnaissance 1 à N dans une base de données contenant N individus)

L'approche hybride :

Qui exploitent en symbiose les deux techniques précédentes.

C Évolution vers la reconnaissance faciale basée sur l'apprentissage automatique :

Avec l'avènement de l'apprentissage automatique et en particulier de l'apprentissage profond, la reconnaissance faciale a connu une évolution significative. Les approches modernes de la reconnaissance faciale utilisent des réseaux neuronaux profonds pour apprendre automatiquement à extraire et à reconnaître les caractéristiques pertinentes du visage, ce qui permet d'atteindre des performances remarquables dans des conditions variables et complexes.[3]

1.1.2 Approches géométriques

L'étude géométriques abordent la problématique de représentation de visages afin d'en déterminer les traits particuliers du visage qui devraient être plus spécifiquement exploités afin de les distinguer. La difficulté principale reliée à ces techniques consiste à trouver une manière adéquate de représenter les interrelations qui existent entre ces caractéristiques, qu'il s'agisse de positions relatives sur le visage, d'interconnexions géométriques, ou de regroupements par graphes. Parmi les techniques les plus répandues, nous y retrouvons particulièrement les méthodes HMM, DCP, par landmarks et les approches purement géométriques. À titre d'exemples, les images représentées de la Figure 1.1 à la Figure 1.3 démontrent des applications appartenant à la famille des techniques locales.[3]

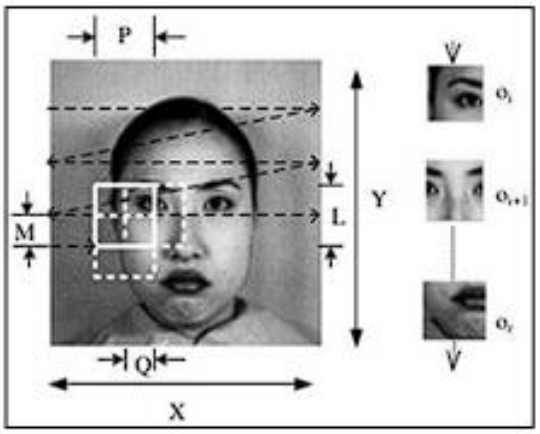


Figure 1-1 : Extraction de blocs géométriquement paramétrés[10]

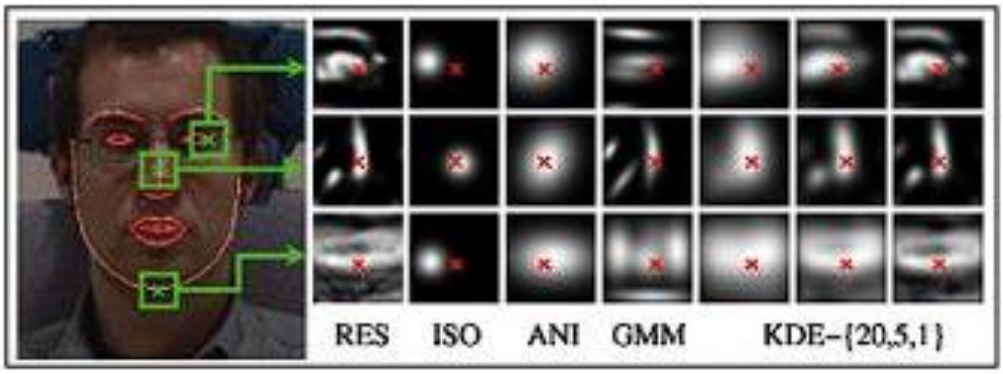


Figure 1-2 : Extraction de caractéristiques locales à partir de landmarks [11]

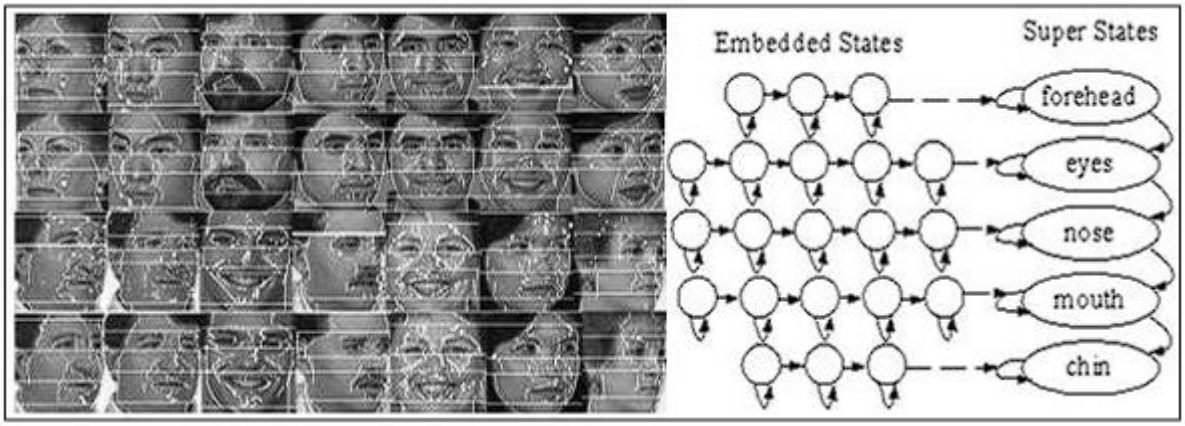


Figure 1-3 : Extraction des états ordonnés des sections du visage[12]

1.1.3 Approches algorithmiques

Les approches algorithmiques en reconnaissance faciale sont essentielles pour extraire et interpréter les informations visuelles nécessaires à l'identification ou à la vérification des visages. Cette section explore les techniques algorithmiques utilisées dans la reconnaissance faciale,

notamment le prétraitement d'image, l'extraction de caractéristiques et l'utilisation d'algorithmes de classification.

A Techniques de prétraitement d'image :

Le prétraitement d'image vise à améliorer la qualité et la cohérence des données d'image afin de faciliter la détection et l'identification des visages. Les techniques de prétraitement courantes incluent :

Normalisation d'image : Ajustement de l'échelle, de l'orientation et de l'illumination des images pour réduire les variations dues à des conditions d'acquisition différentes.

Filtrage : Application de filtres pour améliorer le contraste, réduire le bruit et extraire les contours et les textures des visages.

Détection et alignement des visages : Localisation des visages dans une image et alignement des visages détectés pour assurer une présentation cohérente des données.

B Extraction de caractéristiques :

À titre d'exemples pour l'étude globale, nous retrouvons particulièrement les techniques PCA, LDA, eigenfaces, fisherfaces, LBP et HOG qui sont extrêmement rependues en littérature, que pour en nommer que quelques-unes. À noter que cela n'est en effet qu'une liste non exhaustive étant donné non seulement les multiples variations et alternatives existantes à ces techniques mêmes, mais également à l'existence des multiples autres techniques se rapportant à cette catégorie.[2]

Local Binary Patterns (LBP) : Une méthode basée sur la texture qui encode les relations de luminance locales dans une image en comparant les valeurs de pixels voisins à un seuil donné.[2]

Histogram of Oriented Gradients (HOG) : Une technique qui mesure la distribution des orientations du gradient dans une image pour capturer les contours et les textures des visages.[4]

Eigenfaces : Une méthode basée sur l'analyse en composantes principales (PCA) qui représente les visages comme une combinaison linéaire de vecteurs propres, appelés "eigenfaces", qui capturent les variations les plus significatives des données d'image.

Le but consiste à extraire les caractéristiques sur l'ensemble du visage qui permettent de les discriminer entre eux, de sorte à pouvoir produire un modèle de visage particulier avec lequel un classificateur pourrait être entraîné à les différencier. Par exemple, la Figure 1.4 et la Figure 1.5 présentent des variations de visages représentés par différentes techniques. Nous pouvons y voir

que les techniques permettent de faire ressortir les points saillants du visage, de sorte à permettre d'identifier les caractéristiques discriminantes les plus probables de rendre la classification de différents visages efficace. [4]

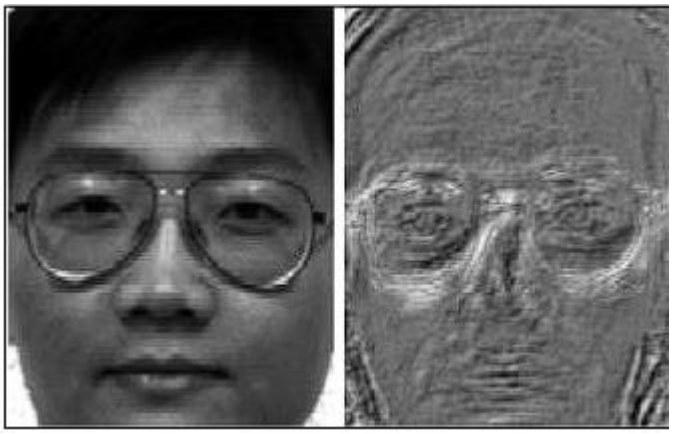


Figure 1-4 : Illustration d'une fisherface produite d'un visage [13]

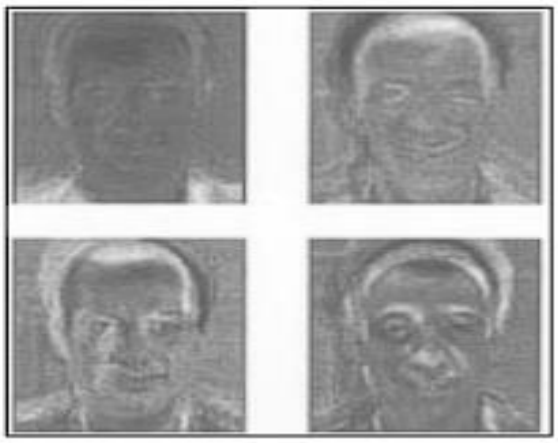


Figure 1-5 : Extraction de composantes principales d'un visage[14]

C Algorithmes de classification :

Une fois que les caractéristiques des visages ont été extraites, des algorithmes de classification sont utilisés pour identifier ou vérifier les visages en les comparants à des modèles existants. Les algorithmes de classification couramment utilisés incluent :

k plus proches voisins (k-NN) : Un algorithme simple qui classe les visages en fonction de la similarité de leurs caractéristiques avec les visages d'entraînement les plus proches.

Machines à vecteurs de support (SVM) : Un algorithme qui apprend à séparer les visages en utilisant un hyperplan dans un espace multidimensionnel défini par leurs caractéristiques.

Réseaux de neurones : Des modèles d'apprentissage automatique complexes qui apprennent à représenter et à classifier les visages à partir de données d'entraînement.[4]

1.2 État de l'art de la recherche en reconnaissance faciale :Tendances émergentes

Les récentes avancées en matière de technologie et d'intelligence artificielle ont ouvert de nouvelles perspectives passionnantes pour la reconnaissance faciale. Cette section explore certaines des tendances émergentes dans le domaine de la reconnaissance faciale, notamment l'utilisation de l'apprentissage profond, l'intégration de techniques de traitement du langage naturel (NLP) et les applications spécialisées dans divers domaines.

1.2.1 Utilisation de réseaux de neurones profonds (Deep Learning) pour la reconnaissance faciale :

L'apprentissage profond, ou deeplearning, a révolutionné la reconnaissance faciale en permettant l'apprentissage automatique de représentations hiérarchiques complexes des données d'image. Les réseaux neuronaux profonds, tels que les réseaux de neurones convolutifs (CNN) et les réseaux de neurones siamois, ont démontré une capacité remarquable à extraire des caractéristiques discriminatives des visages et à réaliser des tâches de reconnaissance avec une précision et une robustesse accrues.[5]

1.2.2 Intégration de techniques de traitement du langage naturel (NLP) pour une compréhension contextuelle des images faciales :

L'intégration de techniques de traitement du langage naturel dans la reconnaissance faciale vise à améliorer la compréhension contextuelle des images faciales. En analysant les descriptions textuelles associées aux images ou en exploitant des modèles de langage pré-entraînés, les systèmes de reconnaissance faciale peuvent mieux interpréter le contexte dans lequel les visages sont capturés, ce qui peut conduire à une identification plus précise et à une meilleure adaptation à des scénarios variables.[5]

1.2.3 Applications de la reconnaissance faciale dans des domaines spécifiques tels que la santé, le marketing et la gestion des ressources humaines :

La reconnaissance faciale trouve des applications dans divers domaines, allant de la sécurité et de la surveillance à la santé, au marketing et à la gestion des ressources humaines. Les applications spécialisées utilisent des techniques de reconnaissance faciale pour résoudre des problèmes spécifiques dans leur domaine respectif, tels que l'identification des personnes disparues, la détection des émotions, la personnalisation des publicités et la gestion des effectifs.

Voici un tableau ayant plus détails sur l'utilisation de la reconnaissance faciale dans ces domaines :

Domaine d'utilisation	*Objectif*
Sécurité et surveillance	-Identification et vérification des individus pour l'accès à des zones sécurisées.
	- Détection et suivi des personnes suspectes dans les lieux publics.
	- Prévention des crimes en identifiant les personnes recherchées par les forces de l'ordre.
Contrôle d'accès	- Contrôle d'accès aux bâtiments, aux ordinateurs ou aux systèmes informatiques.
	- Authentification biométrique pour les transactions sécurisées.
	- Suivi du temps de présence des employés dans les entreprises.
Identification et vérification	- Authentification des individus pour les applications gouvernementales (passeports, visas).
	- Vérification de l'identité des clients dans les banques et les institutions financières.
	- Accès sécurisé aux appareils mobiles et aux services en ligne.
Marketing et Publicité	- Analyse démographique des clients dans les magasins pour adapter les offres et les publicités.
	- Personnalisation des expériences utilisateur dans les applications et les sites web.
	- Mesure de l'efficacité des campagnes publicitaires en analysant les réactions faciales.
Santé et bien-être	- Suivi de la santé et du bien-être des patients en analysant les expressions faciales.
	- Détection des signes de fatigue, de stress ou de maladies mentales.
	- Assistance aux personnes handicapées en reconnaissant les expressions pour la communication.
Éducation et apprentissage	- Suivi de l'attention des élèves en classe pour adapter les méthodes d'enseignement.
	- Personnalisation des programmes éducatifs en fonction des styles d'apprentissage des élèves.
	- Évaluation automatique des réactions des élèves lors de sessions de formation en ligne.

Tableau 1 : Domaines d'utilisation de la reconnaissance faciale

1.3Méthodologie de recherche :

La méthodologie de recherche en reconnaissance faciale englobe les étapes de planification, de collecte de données, d'expérimentation et d'évaluation nécessaires pour mener à bien une

étude ou un projet dans ce domaine. Cette section détaille les différentes phases de la méthodologie de recherche, y compris la collecte de données, l'expérimentation et l'évaluation des modèles de reconnaissance faciale.

1.3.1 Collecte de données :

La collecte de données est une étape cruciale dans le développement d'un système de reconnaissance faciale, car la qualité et la diversité des données influencent directement la performance et la généralisation du système. Cette section explore les différentes sources de données disponibles pour l'entraînement et le test des modèles de reconnaissance faciale, ainsi que les considérations éthiques et légales associées à la collecte et à l'utilisation de données faciales.

- Sources de données publiques et privées pour l'entraînement et le test des modèles de reconnaissance faciale :

A Sources de données publiques :

Les sources de données publiques sont des ensembles de données largement accessibles et utilisés dans la recherche académique et industrielle en reconnaissance faciale. Ces ensembles de données comprennent généralement des images de visages capturées dans divers environnements et conditions, avec des annotations fournies pour l'identification des visages et la vérification de l'identité.[6]

Exemples de sources de données publiques :

Labeled Faces in the Wild (LFW) : Un ensemble de données comprenant des images de visages collectées à partir d'Internet, souvent utilisé pour l'évaluation de la reconnaissance faciale en conditions non contrôlées.[7]

CelebA : Un ensemble de données contenant des images de célébrités, utilisé pour l'apprentissage et l'évaluation de modèles de reconnaissance faciale.

YouTube Faces Database : Un ensemble de données composé de séquences vidéo de visages extraits de vidéos YouTube, utilisé pour l'entraînement de modèles de reconnaissance faciale vidéo.[8]

B Sources de données privées :

Les sources de données privées sont des ensembles de données collectés par des organisations ou des entreprises pour des applications spécifiques de reconnaissance faciale, telles que la gestion des ressources humaines, la sécurité et la surveillance. Ces ensembles de données peuvent être plus restreints en accès, mais ils offrent souvent une plus grande variabilité et une meilleure représentation des conditions réelles d'utilisation. [9]

Exemples de sources de données privées :

Ensembles de données d'entreprise : Des ensembles de données contenant des images de visages d'employés collectées à des fins d'identification et de gestion des présences.

Ensembles de données de sécurité : Des ensembles de données utilisés pour la surveillance et la sécurité, comprenant des images de visages capturées par des caméras de sécurité dans des environnements spécifiques.

C Considérations éthiques et légales liées à la collecte et à l'utilisation de données faciales :

La collecte et l'utilisation de données faciales soulèvent des préoccupations éthiques et légales importantes en matière de confidentialité, de consentement et de protection des données personnelles. Il est essentiel de respecter les réglementations et les directives en vigueur, telles que le Règlement Général sur la Protection des Données (RGPD) en Europe, et de prendre des mesures appropriées pour garantir la confidentialité et la sécurité des données collectées.[9]

1.3.2 Expérimentation et évaluation :Métriques d'évaluation de la performance des modèles :

A Précision :

La précision mesure le pourcentage de visages correctement identifiés parmi tous les visages prédits par le modèle.

Elle est calculée en divisant le nombre de vrais positifs par la somme des vrais positifs et des faux positifs.

B Rappel :

Le rappel mesure le pourcentage de visages correctement identifiés parmi tous les visages réels présents dans l'ensemble de test.

Il est calculé en divisant le nombre de vrais positifs par la somme des vrais positifs et des faux négatifs.

C F1-score :

Le F1-score est une mesure de la précision et du rappel combinés, donnant une moyenne harmonique des deux.

Il est calculé en prenant la moyenne harmonique de la précision et du rappel, ce qui permet de prendre en compte à la fois les faux positifs et les faux négatifs.

Conclusion :

Dans ce chapitre nous avons concentrés nos recherches sur les fondamentaux de la reconnaissance faciale, ses mécanismes et les évolutions dont elle est passée, pour arriver aux métriques d'évaluation de la performance des modèles basés sur cette technologie.

Nous enchainerons avec l'intelligence artificiel et comment elle est conçue, nous étudierons différentes architectures de Deep Learning.

Chapitre 2 : Intelligence Artificiel

Introduction :

Dans ce chapitre nous allons découvrir qu'est-ce que c'est que l'intelligence artificielle en quoi est composé l'IA, et ainsi connaitre la différence entre le Machine Learning et le Deep Learning, pour ainsi pouvoir nous approfondir davantage sur le Deep Learning et les réseaux neuronaux

2.1Les réseauxneuronaux

2.1.1 Introduction

L'intelligence artificielle (ia) est une méthodeinformatique qui permet à une machine quelconque d'effectuer des missions complexe que seul l'être humain est capable de résoudre. [15]

Pour pouvoir réaliser ses taches, la machine auras besoin d'être entrainée pour qu'elle puisse être capable de reproduire sa propre représentation du monde, c'est ce qu'on appelle couramment le « machine learning » (en français : apprentissage automatique). Le machine learning est un ensemble de méthode qui permettre à une machine d'apprendre à partir d'échantillons où le deeplearning(enfrançais : apprentissage profond) est son sous ensemble qui est basé sur les réseaux neuronaux, comme son nom laisse a suggéré que ce sont des réseaux de neurones artificiels imitant le système nerveux des êtres vivant.

2.1.2 Machinelearning

Le machine learning est un domaine qui se concentre sur le développement, l'analyse et la mise en œuvre de méthodes permettant à une machine d'évoluer par le biais d'un processus d'apprentissage. Cela lui permet d'accomplir des tâches qui seraient difficiles ou impossibles à réaliser avec des algorithmes traditionnels. L'objectif principal est d'extraire et d'utiliser l'information contenue dans un ensemble de données de manière automatique. La figure 2-1 illustre la relation entre l'intelligence artificielle (IA), l'apprentissage automatique (ML) et l'apprentissage profond (DL).

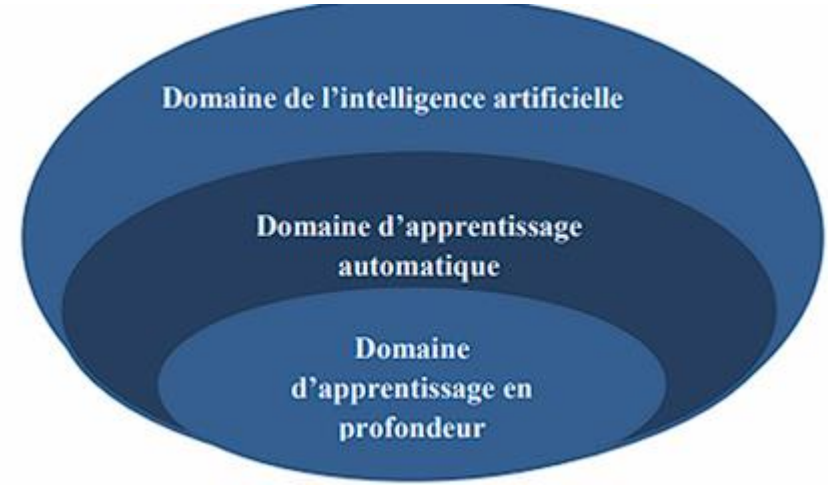


Figure 2-1 la relation entre IA, ML, DL

Le concept d'apprentissage automatique est basé sur l'idée qu'un programme est capable d'apprendre d'une expérience E pour accomplir une tâche T. Cela signifie que si la performance P du programme, évaluée par la tâche T, s'améliore avec l'expérience E, alors le programme est considéré comme ayant appris. Autrement dit, le programme est capable de reproduire et de généraliser un comportement pour de nouvelles situations qui sont similaires à l'expérience E [16].

A La tâche T représente une progression qui regroupe une ou plusieurs fonctions.

À savoir :

- **Régression :** On peut voir la régression comme une technique d'apprentissage automatique qui construit un modèle pour prédire une valeur cible en se basant sur des prédicteurs indépendants. C'est principalement un outil statistique qui permet de comprendre la relation entre une variable dépendante et une variable indépendante [17].

- **Classification :** La classification est le processus qui consiste à diviser un ensemble de données en différentes classes. Ce processus peut être appliqué à des données structurées ou non structurées. Il commence par la prédiction de la classe des points de données.

- **Segmentation :** La segmentation est le processus qui consiste à diviser les données en différents groupes, souvent appelés segments ou catégories (en anglais : Labels). Un des

défis majeurs de la création de segments est de déterminer le nombre réel de segments présents dans les données [18].

B L'expérience E représente l'algorithme de l'apprentissage. Les algorithmes d'apprentissage peuvent être classés en trois catégories principales :

- **Apprentissage supervisé :** C'est un type d'apprentissage où les classes sont déjà définies et les exemples utilisés pour l'apprentissage sont déjà étiquetés avec la classe à laquelle ils appartiennent. Le système apprend sur un ensemble de données d'apprentissage et est ensuite testé sur un ensemble de données de test différent. Il s'agit de classer ou d'évaluer des données en fonction de règles bien définies à partir d'un modèle basé sur les hiérarchies de données présentées lors des étapes précédentes. Plusieurs algorithmes sont associés à ce modèle, comme l'apprentissage par arbre de décision, la régression logistique ou les réseaux bayésiens. La qualité du modèle est évaluée en fonction du taux d'erreurs commises sur l'ensemble de données de test.

- **Apprentissage non supervisé :** À l'inverse de l'apprentissage supervisé, ici, nous n'avons pas d'étiquettes préalablement connues pour nos données. L'algorithme doit trouver seul des structures dans un ensemble qui ne lui en donne pas directement (par exemple en regroupant des clusters) et donc sans exemple de sortie particulier par lui-même (la structure : datamining) ou dans un ensemble. Larecherche étendue non supervisé (en anglais : plus ou moins extendeddiscovery) fait partie des méthodes non paramétriques.

- **Apprentissage par renforcement :** C'est une méthode d'apprentissage où une machine ou un logiciel, appelé agent, interagit avec son environnement et apprend au fur et à mesure de ces interactions jusqu'à ce qu'il découvre les comportements qui produisent les meilleures récompenses.
  La figure 2-2 montre la différence entre l'apprentissage supervisé et non supervisé.

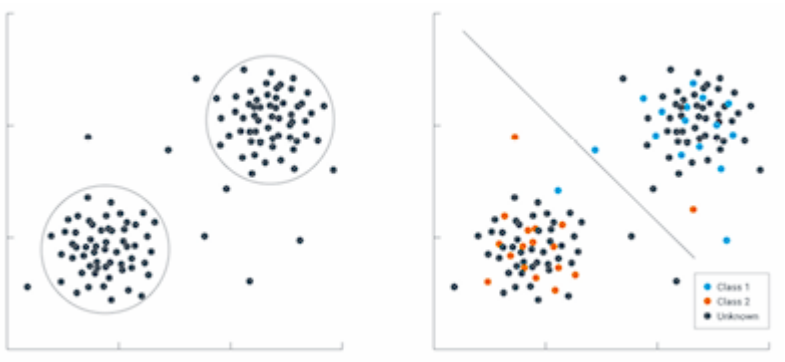


Figure 2-2 : l'apprentissage supervisé et non supervisé

C La performance P désigne des mesures quantitatives qui servent à évaluer l'algorithme de l'apprentissage effectué par la tâche.

2.1.3 Les réseaux neuronaux

Les réseaux neuronaux artificiel aussi appelés RNA, sont des modèles de traitement de l'information qui simulent le fonctionnement d'un système nerveux vivant (figure 2-3). C'est similaire à la façon dont le cerveau manipule l'information au niveau du fonctionnement. Tous les réseaux neuronaux sont constitués de neurones interconnectés qui sont organisés en couches [19]

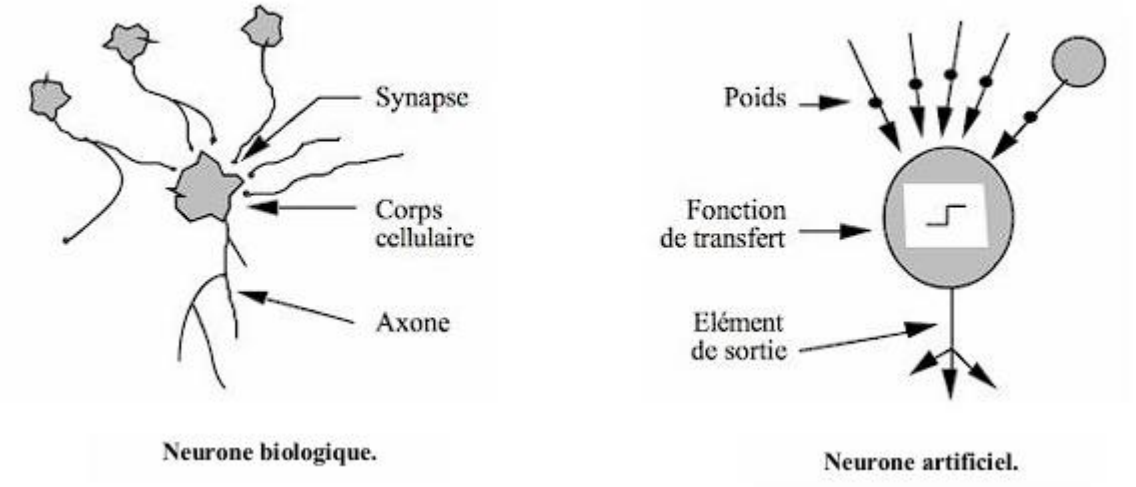


Figure 2-3 : similitude entre les neurones vivantes et neurones artificiel.

Le but, est de réaliser des calculs complexes et de trouver, par apprentissage, une relation non linéaire entre des données numériques et des paramètres [19].L'un des neurones artificiels simplifié c'est le perceptron (figure 2.4) qui a été introduit par Frank Rosenblatt en 1957 [20]. Il contient plusieurs entrées ou chacune possède un poids. Les entrées sont toutes connectées à une seule sortie en passant par une fonction de transfert qui est appelée aussi fonction d'activation [21].

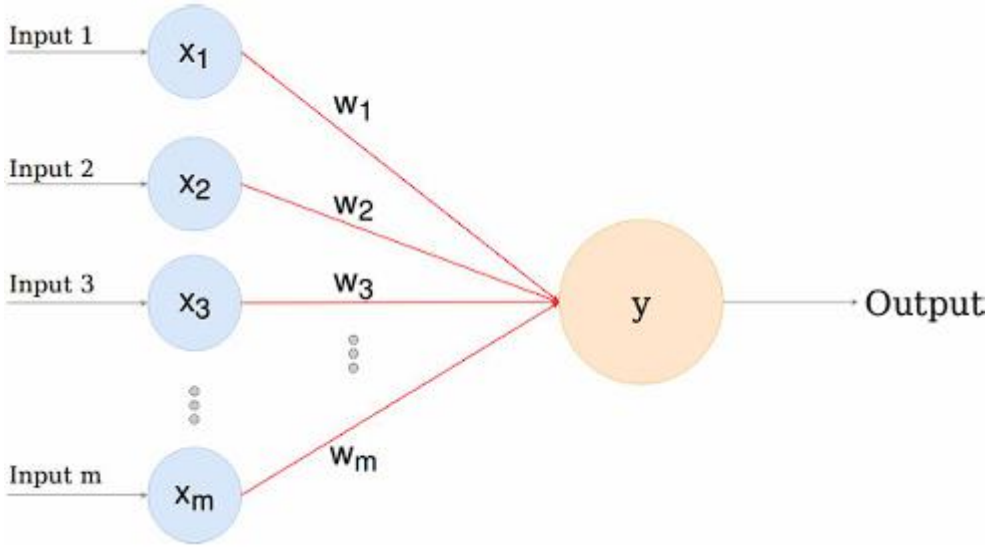


Figure 2-4 : Représentation d'un perceptron.

Ou :

Xi : représente les entrées de perceptron.

Wi : représente les poids associes a ces entrées.

Y : c'est la fonction de transfert ou la fonction d'activation.

2.2Le Deep Learning

L'apprentissage en profondeur est un ensemble d'algorithmes d'apprentissage automatique qui tirent d'apprendre à plusieurs niveaux, correspondant à différents niveaux d'abstraction. Il a la capacité d'extraire des caractéristiques à partir des données brutes grâce aux multiples couches de traitement composé de multiples transformations linéaires et non linéaires et apprendre sur ces caractéristiques petites à petites à travers chaque couche avec une intervention humaine minimale [22]. La figure 2-5 montre un schéma d'un apprentissage profond avec plusieurs couches.

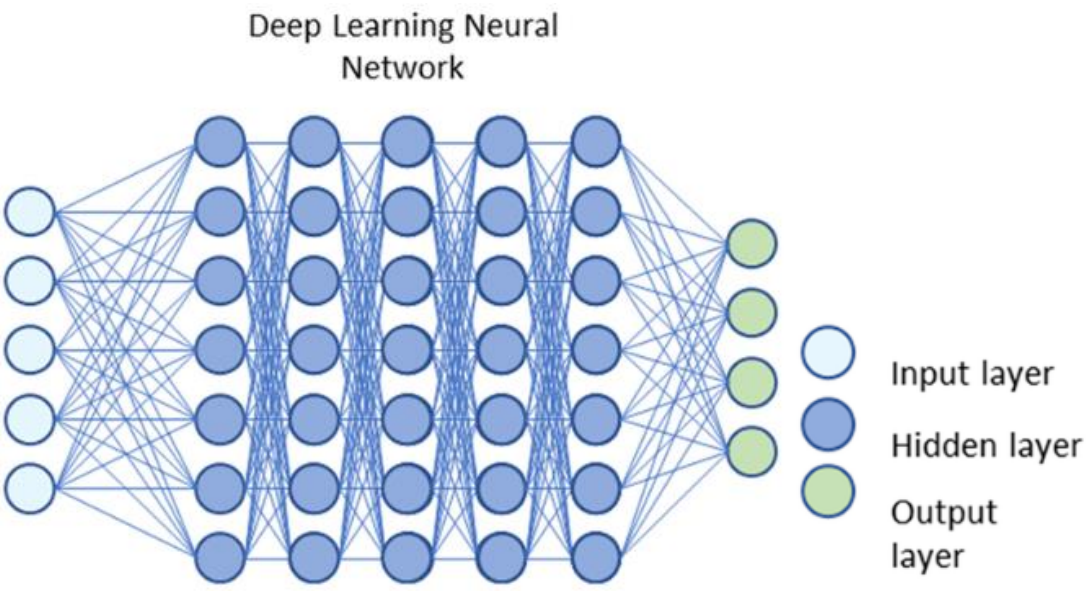


Figure 2-5 : schéma illustratif de deeplearning avec plusieurs couches. [49]

2.2.1 Pourquoi le Deep Learning ?

Tout d'abord les différents algorithmes du Deep Learning ne sont apparus qu'àl'échec de l'apprentissage automatique tentant de résoudre une grande variété de problèmes de l'intelligence artificielle [23] :

- de développer une grande quantité de données telle que lebig data
- de s'adapter àn'importequelproblème
- d'extraire les caractéristiques de façon automatique.

  La figure ce dessous montre une comparaison entre le machine learninng et le Deep Learning

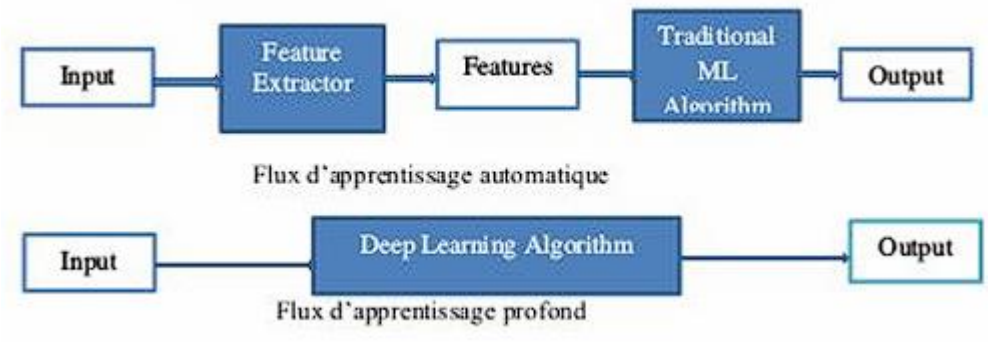


Figure 2-6 comparaisons entre machine Learning et Deep Learning

2.2.2 Lesdifférentsmodelés du réseau profond :

Il existe plusieurs modèles pour un réseau de neurones profond

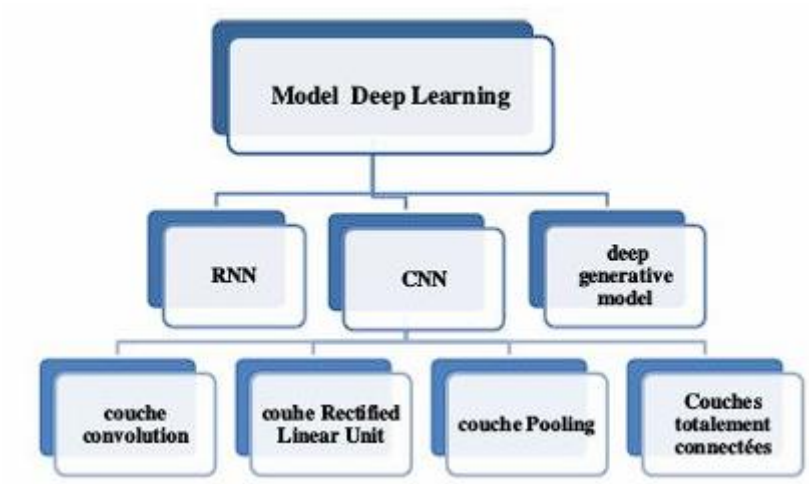


Figure 2-7: different models de deep learning

A Réseau de neuronesrécurant (RNN) :

Un réseau de neurones récurrent (Recurrent Neural Network (RNN)) est un réseau de neurones dont le graphe de connexion contient au moins un cycle. De nombreux types de RNN ont été développés au cours des 30 dernières années telles que les réseaux d'Elman, les réseaux de Jordan et les Echo State Networks [24].

Au cours des dernières années, un type de RNN qui est le plus performant à élever challenge Long Short-Term Memory (LSTM) est devenu une sorte de référence pour les obtenus des applications aussi nombreuses que variées.

B Modèlegénératif :

Si les modèles CNN et RNN sont utilisés pour prédire les données du label et de l'entrée, le modèle génératif est un modèle probabiliste qui décrit comment générer les données, il apprend et fait des prédictions en utilisant la loi de Bayes [25]. En plus, les modèles génératifs sont capables de faire une simple classification comme par exemple générer de nouvelles observations. Voici quelque exemple de modèle génératif :

- Boltzmann Machines [26, 27].
- Restricted Boltzmann Machines [28, 29].

- Deep Belief Networks [30, 31].
- Deep Boltzmann Machines [32, 33].
- Generative Adversarial Networks [34, 35, 36].
- Generative Stochastic Networks [37].
- Adversarialautoencoders [38].

C Réseau de neuronesconvolutif (CNN)

Les réseaux de neurones convolutifs sont à ce jour les modèles les plus performants pourclasser des images. Désignés par l'acronyme CNN, de l'anglais Convolutional Neural Networkqui est un réseau constitué de plusieurs couches de convolutions dont chacune un traitement surl'image est effectué [39]. En entrée, une image est fournie sous la forme d'une matrice de pixels.Elle a deux dimensions pour une image en niveaux de gris. La couleur est représentée par unetroisième dimension, de profondeur 3 pour représenter les couleurs fondamentales [Rouge, Vert,Bleu]. L'image est passée à travers une succession de filtres, ou noyaux de convolution, créantde nouvelles images appelées cartes de convolutions. Certains filtres intermédiaires réduisent larésolution de l'image par une opération de maximum local. Au final, les cartes de convolutionssont mises à plat et concaténées en un vecteur de caractéristiques. La figure 2-8 représente unréseau de neurone CNN

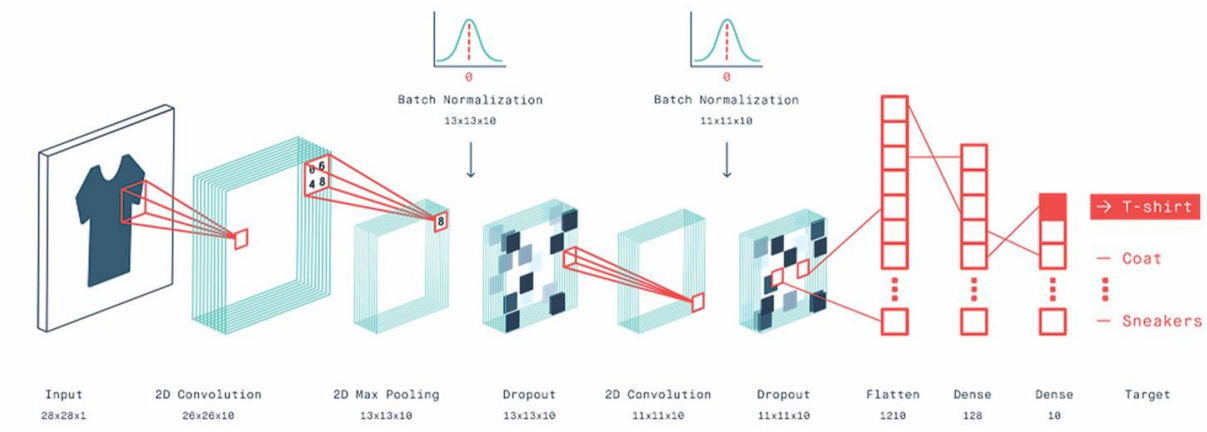


Figure 2-8 : une representation d'un reseau de neuronne CNN

2.2.3- Architecture de réseau de neurones convolutif

Il existe plusieurs architectures des CNNs qui varient des plus basiques aux plus complexes. Une architecture spécifique est plus performante dans certains domaines que d'autres. Une architecture CNN est formée par un empilement de couches de traitement indépendantes :

- La couche de convolution (CONV), qui traite les données d'un champ récepteur.

- La couche de pooling (POOL), qui permet de compresser l'information en réduisant la taille de l'image intermédiaire (souvent par sous-échantillonnage).
- La couche de correction (ReLU) qui applique la fonction d'activation (Unité de rectification linéaire).
- La couche 'entièrement connectée' (FC), qui est une couche de type perception.
- La couche de perte (LOSS) responsable pour la correction des erreurs.

A Couche de convolution(CONV)

La couche de convolution est la composante principale des réseaux de neurones convolutifs. Son but est de repérer la présence d'un ensemble de caractéristiques dans les images reçues en entrée [40].

Pour cela, on réalise un filtrage par convolution : le principe est de faire glisser une fenêtre représentant la caractéristique sur l'image, et de calculer le produit de convolution entre la caractéristique et chaque portion de l'image balayée et produire une carte de caractéristiques (featuresmap) a sortie [41].

Avant l'étape de la convolution, il faut déterminerquelques paramètres, ces paramètres font partie du calculde la taille des images à la sortie de chaque couche (la carte des caractéristiques), cesparamètres sont :

- Les Strides (Pas) : Stride est le nombre de pixels décalés sur la matrice d'entrée. Lorsque le pas est de 1, nous déplaçons les filtres d'1 pixel à la fois. Lorsque le pas est de 2, nous déplaçons les filtres à 2 pixels à la fois et ainsi de suite [42].
- Le Padding : Le padding est l'ajout de pixels aux bords de l'image pour pouvoir appliquer la convolution sur les bordures de l'image (figure 2-9). Le nombre de lignes et colonnes rajoutées dépend de la taille du filtre de la convolution [15].

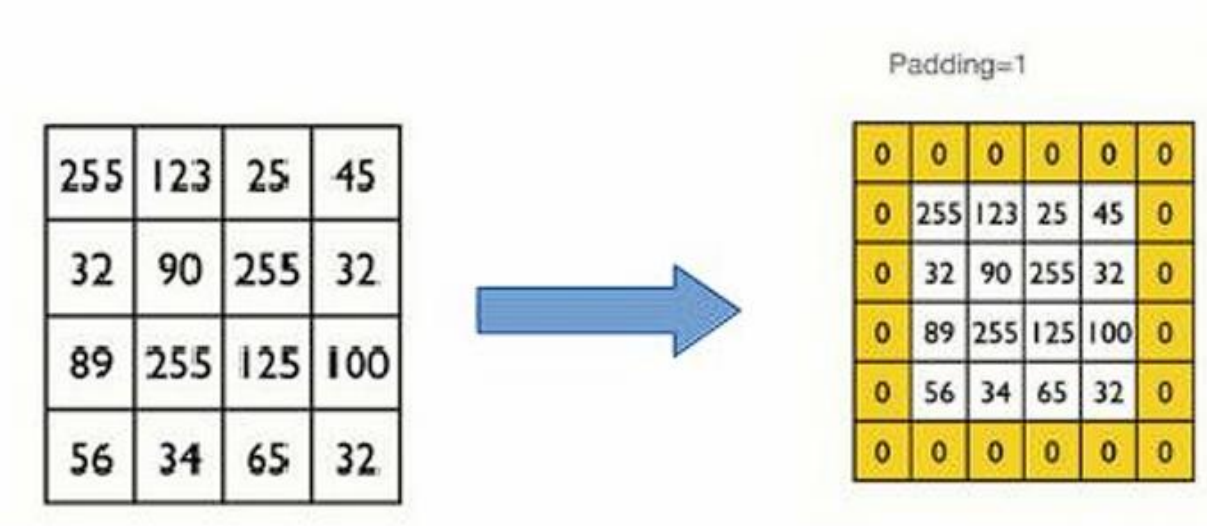


Figure 2-9 : représentation d'un padding

Le principe de la convolution est représenté de la figure ci-dessous :

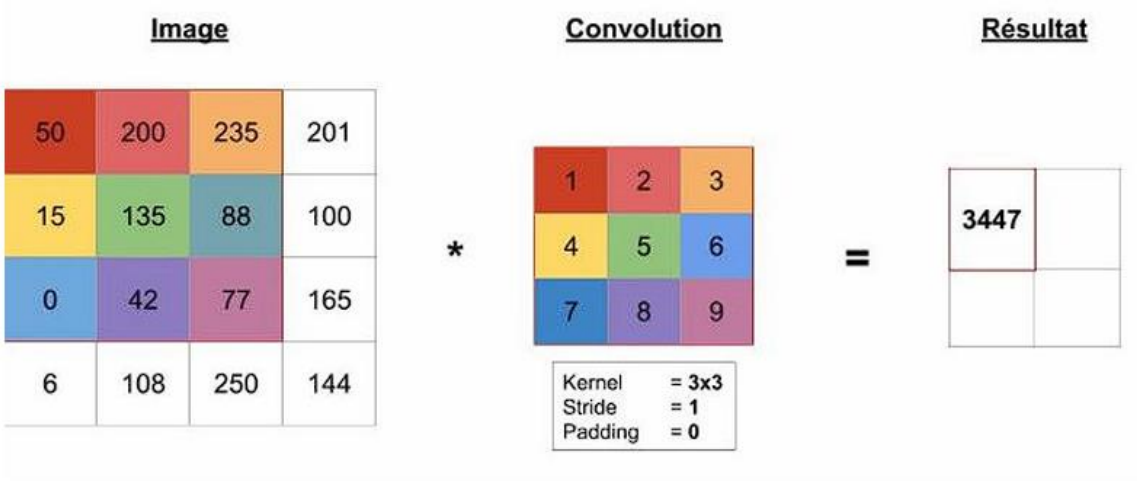


Figure 2-10 : principe de la convolution

Selon les valeurs de la figure ce dessus, le calcul de la convolution est effectué comme suit :

50*1+200*2+235*3+15*4+135*5+88*6+0*7+42*8+77*9=3447.

$$Taille\ sortie\ covolution = 1 + \frac{W - F + 2P}{S}$$

A la sortie de chaque couche de convolution taille de l'image est réduite avec la formule suivante (2-1), on peut savoir la taille de l'image en sortie d'une convolution a partie de la taille de l'image(W), la taille du filtre(F), le padding(P) et le stride(S) [43].

B Couche de pooling(pool)

C'est une méthode mathématique permettant de réduire la taille de l'image sans perdre les informations les plus importantes dans l'image. Son principe consiste à diviser l'image en petites matrices par exemple 2x2 et choisir la valeur maximale de chaque matrice, pour obtenir une image réduite [15].

La figure 2-11 suivante montre un exemple de l'application d'un sous échantillonnage 2x2 avec un pas égal à 2.

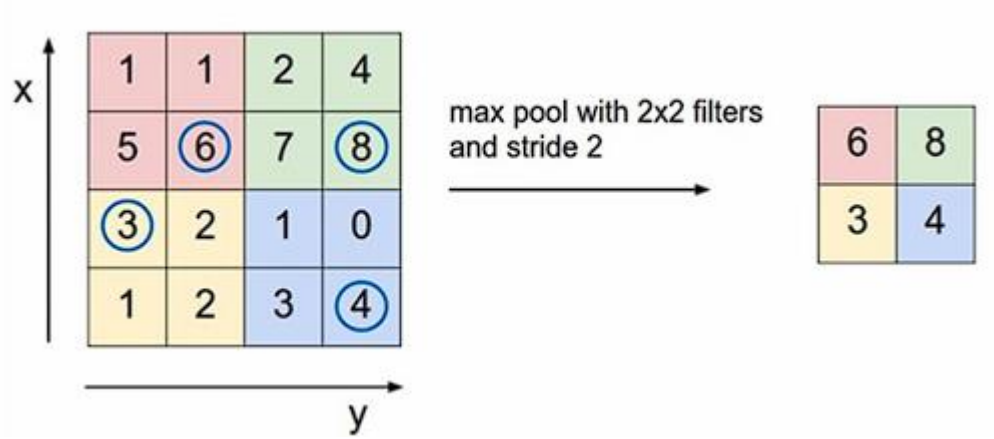


Figure 2-11 : principe de pooling

C Couche de correction (ReLu)

Souvent, il est possible d'améliorer l'efficacité du traitement en intercalant entre les couches de traitement une couche qui va opérer une fonction mathématique (fonction d'activation) sur les signaux de sortie. La couche de correction ReLu remplace donc toutes les valeurs négatives reçues en entrées par des zéros [39].

D Couche de « entièrement connectée » (FC)

La couche entièrement connectée est un traditionnel perceptron multicouche (Multi Layer Perceptron). Le terme « entièrement connecté » implique que chaque neurone dans la couche précédente est connecté à chaque neurone sur la couche suivante. La figure ci-dessous représente un exemple d'une couche entièrement connectée [43].

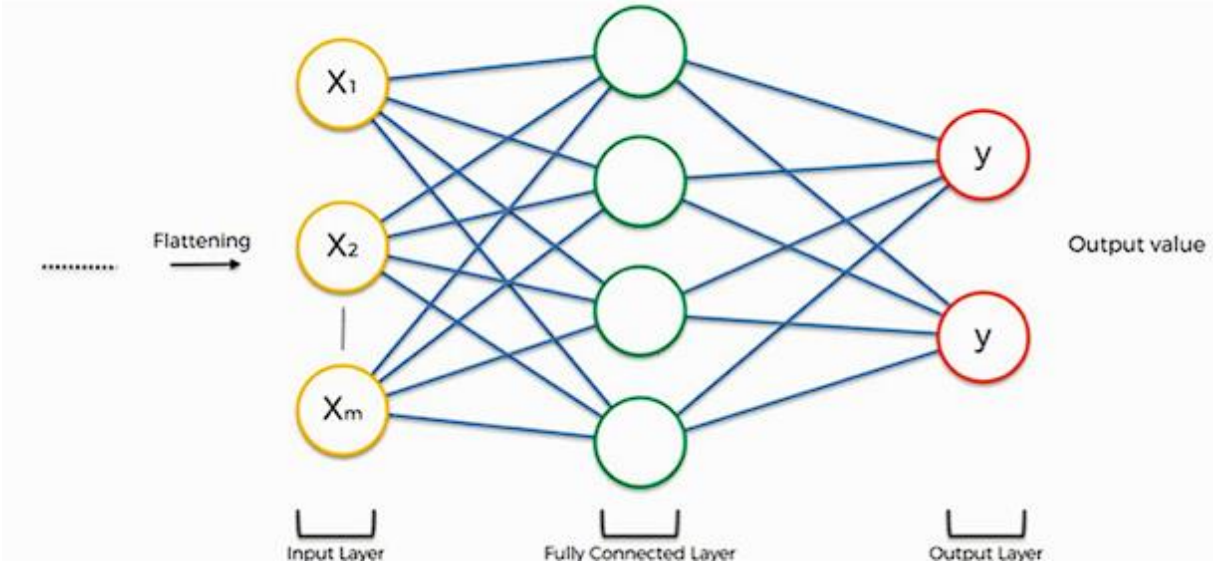


Figure 2-12: exemple des couches entièrement connectées

La sortie des couches de convolution et de Pooling représente les fonctions de haut niveau de l'image d'entrée. Le but de la couche entièrement connectée est de pouvoir utiliser ces fonctions pour classer l'image d'entrée dans différentes classes en fonction de l'ensemble de données d'apprentissage [43].

E Couche de perte (LOSS)

La couche de perte spécifie comment l'entraînement du réseau pénalise l'écart entre le signal prévu et réel. Elle est normalement la dernière couche dans le réseau. Diverses fonctions de perte adaptées à différentes tâches peuvent y être utilisées. La fonction «Softmax» permet de calculer la distribution de probabilités sur les classes de sortie pour la correction des erreurs [39].

F La Normalisation

Il est connu depuis longtemps que l'apprentissage du réseau converge plus rapidement si ses entrées sont centrées réduites (moyenne = 0, variance = 1). Comme chaque couche observe des entrées produites par des couches qui la précèdent, il serait avantageux d'obtenir des entrées centrées et réduites pour chaque couche.

Batch normalisation est une composante qui se trouve entre les couches du réseau de neurones et qui prend continuellement la sortie d'une couche particulière et la normalise avant de l'envoyer à la couche suivante comme entrée [15].

G Le Dropout

Le Dropout est une technique où des neurones sélectionnés au hasard sont ignorés pendant l'apprentissage. Cela signifie que leur contribution à l'activation des neurones qui leur succède est temporairement supprimée lors de la phase de propagation et toutes les mises à jour de poids ne sont pas appliquées aux neurones lors de la phase de rétro-propagation [15].

2.2.4 Les architecture des CNN les plus connus :

Plusieurs architectures dans le domaine des réseauxconvolutifs existent, les plus utilisées sont :

A LeNet-5 :

LeNet est un réseau de convolution à 7 niveaux pionnier de LeCun et al en 1998, qui classifie les chiffres, a été appliqué par plusieurs banques pour reconnaître les numéros manuscrits sur les chèques numérisés en 32x32 pixels. La capacité à traiter des images à plus haute résolution nécessite des couches plus convolutives et plus grandes, cette technique est donc limitée par la disponibilité des ressources informatiques [44]. La figure 2-13 montre l'architecture du LeNet-5.

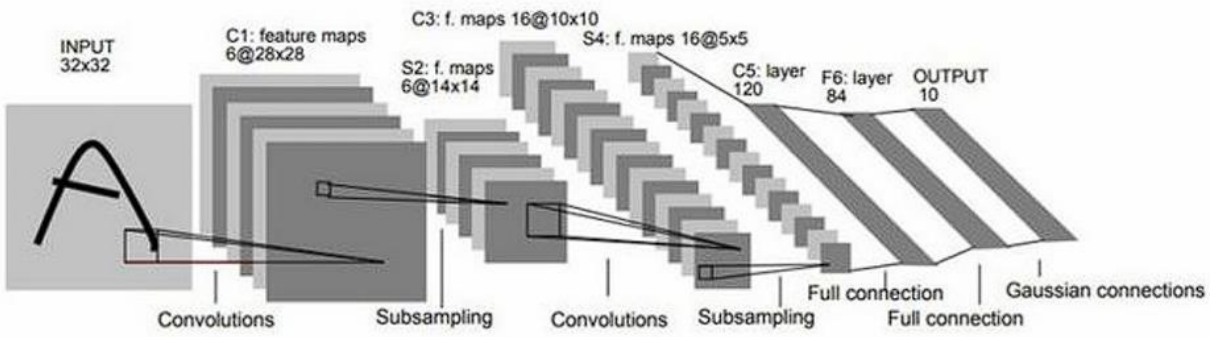


Figure 2-13 : architecture LeNet.

Description des couches du réseauLeNet [45] :

- Première couche : Cette première couche est appliquée sur une image en niveau de gris de taille 32x32, avec six filtres de convolution de taille 5x5 et un pas (stride) égal à 1. La dimension de l'image passe de 32x32x1 à 28×28×6.
- Deuxième couche : Le LeNet-5 applique une couche de sous-échantillonnage avec une taille de filtre de 2×2 et un pas de deux. Les dimensions de l'image résultante seront réduites à 14×14×6.

- Troisième couche : C'est une deuxième couche de convolution avec 16 cartes de caractéristiques (filtres) de taille 5x5 et un pas de 1, les dimensions de l'image passent à 10x10x16.
- Quatrième couche : La quatrième couche est à nouveau une couche de sous échantillonnage avec une taille de filtre de 2 x 2 et un pas de 2. Cette couche est identique à la deuxième couche, la différence est qu'elle comporte 16 cartes de caractéristiques de sorte que la sortie soit réduite à 5×5×16.
- Cinquième couche : La cinquième couche (C5) est une couche convolutionnelle entièrement connectée avec 120 cartes de caractéristiques de taille 1x1. Chacune des 120 unités est connectée à tous les 400 nœuds (5x5x16) de la quatrième couche.
- Sixième couches : La sixième couche est une couche entièrement connectée avec 84 unités.
- Couche de sortie : Enfin, il existe une couche de sortie softmax entièrement connectée avec 10 valeurs possibles correspondant aux chiffres de 0 à 9.

B AlexNet :

Le premier travail qui a popularisé les réseaux convolutifs dans la vision par ordinateur,développé par Alex Krizhevsky. Le réseau avait une architecture très similaire à celle de LeNet,mais il était plus profond, plus grand et comportait des couches convolutives empilées les unespar-dessus les autres (auparavant, il était courant de n'avoir qu'une seule couche CONVimmédiatement suivie d'une couche POOL) [46]. La figure 2-14 montre l'architecture desréseaux AlexNet.

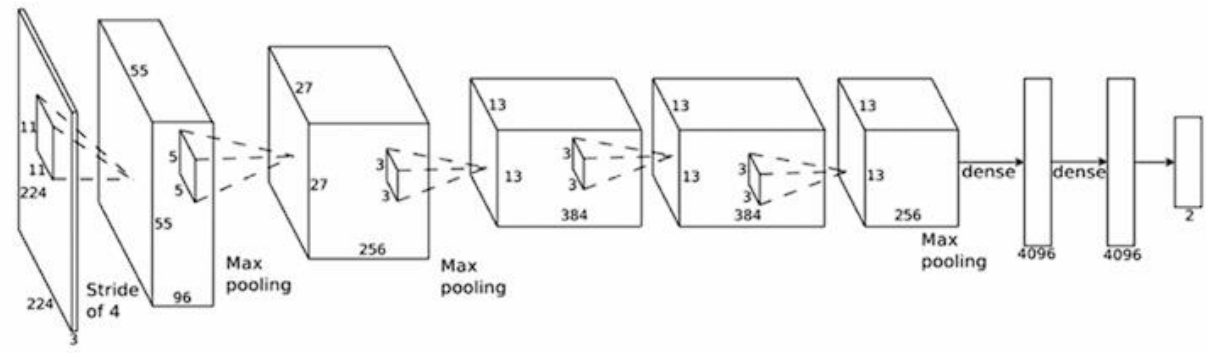


Figure 2-14: architecture AlexNet.

Résumé de l'architecture AlexNet est representer dans le tableau 2 ci-dessous [47] :

Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	227x227x3	-	-	-
1	Convolution	96	55 x 55 x 96	11x11	4	relu
	Max Pooling	96	27 x 27 x 96	3x3	2	relu
2	Convolution	256	27 x 27 x 256	5x5	1	relu
	Max Pooling	256	13 x 13 x 256	3x3	2	relu
3	Convolution	384	13 x 13 x 384	3x3	1	relu
4	Convolution	384	13 x 13 x 384	3x3	1	relu
5	Convolution	256	13 x 13 x 256	3x3	1	relu
	Max Pooling	256	6 x 6 x 256	3x3	2	relu
6	FC	-	9216	-	-	relu
7	FC	-	4096	-	-	relu
8	FC	-	4096	-	-	relu
Output	FC	-	1000	-	-	Softmax

Tableau 2 : Résumé de l'architecture AlexNet

C VGGNet :

Il a été développé par Simonyan et Zisserman et il se compose de 16 couches convolutives etest très attrayant en raison de son architecture très uniforme. Similaire à AlexNet, seulement 3x3convolutions, mais beaucoup de filtres. Formé sur 4 GPU pendant 2 à 3 semaines. C'estactuellement le choix préféré de la communauté pour l'extraction de fonctionnalités à partird'images. La configuration de poids du VGGNet est accessible au public et a été utilisée dans denombreuses autres applications et défis comme extracteur de fonctionnalités de base. Cependant,VGGNet se compose de 138 millions de paramètres, ce qui peut être un peu difficile à gérer [44]. La figure 2-15 montre l'architecture de VGGNet.

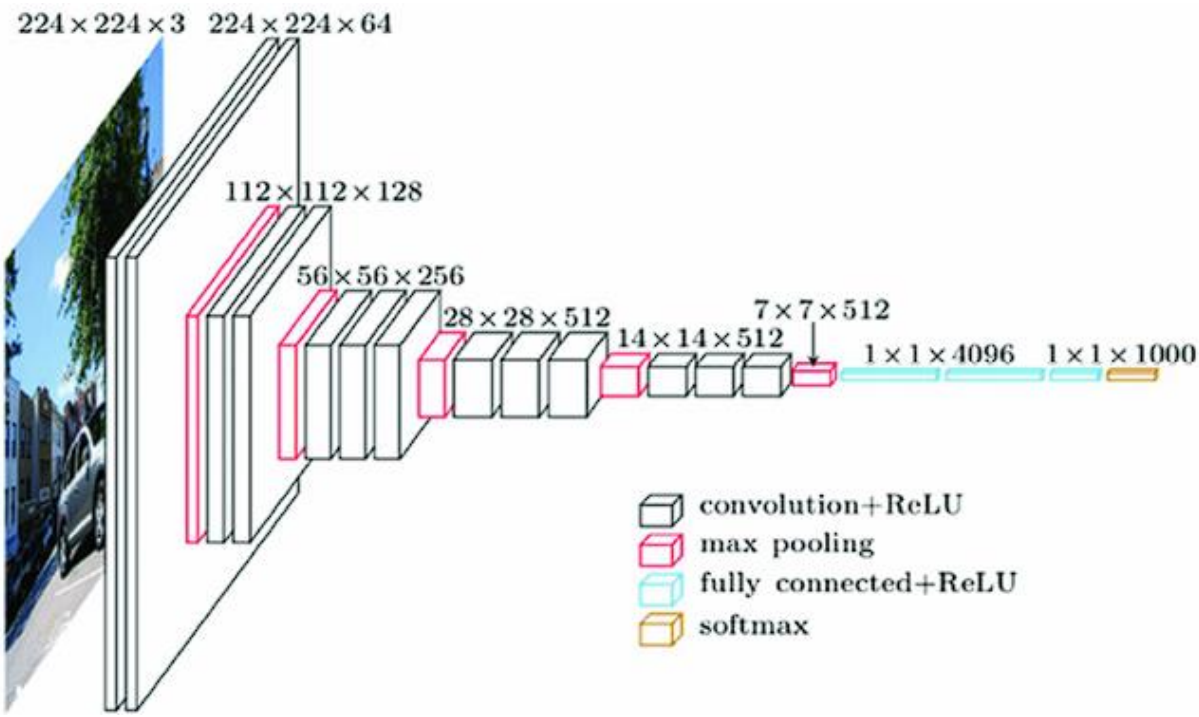


Figure 2-15 : architecture du réseau.

Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	224 x 224 x 3	-	-	-
1	2 X Convolution	64	224 x 224 x 64	3x3	1	relu
	Max Pooling	64	112 x 112 x 64	3x3	2	relu
3	2 X Convolution	128	112 x 112 x 128	3x3	1	relu
	Max Pooling	128	56 x 56 x 128	3x3	2	relu
5	2 X Convolution	256	56 x 56 x 256	3x3	1	relu
	Max Pooling	256	28 x 28 x 256	3x3	2	relu
7	3 X Convolution	512	28 x 28 x 512	3x3	1	relu
	Max Pooling	512	14 x 14 x 512	3x3	2	relu
10	3 X Convolution	512	14 x 14 x 512	3x3	1	relu
	Max Pooling	512	7 x 7 x 512	3x3	2	relu
13	FC	-	25088	-	-	relu
14	FC	-	4096	-	-	relu

Tableau 3 : Résumé de l'architecture VGG16

Résumé de l'architecture VGG16 est presenté dans le tableau 3 [48] :

D GoogleNet

GoogleNet appelé aussi Inception Module, il contient 22 couches, ces couches ne sont pas toutes séquentielles comme dans les architectures précédentes. Il comporte des couches qui se traitent en parallèle. Son architecture est montrée sur la figure 2-16 [49].

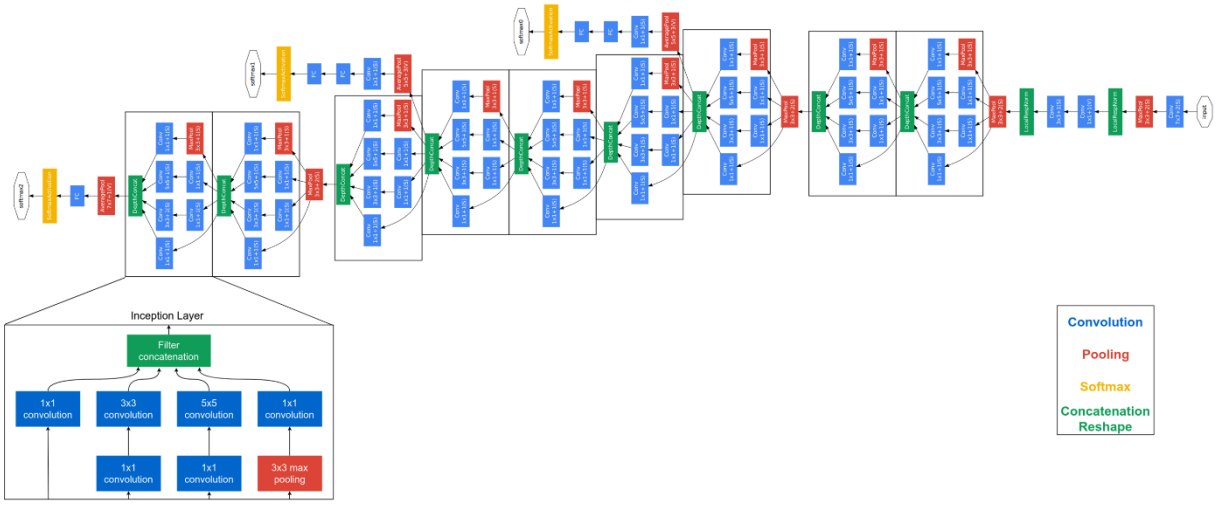


Figure 2-16 Architecture GoogleNet.

Les parties encadrées s'appelle « inception layer »(ou bien « inception module ») et lafigure ci-dessous énonce l'architecture d'un « module inception » avec plus de précisions.

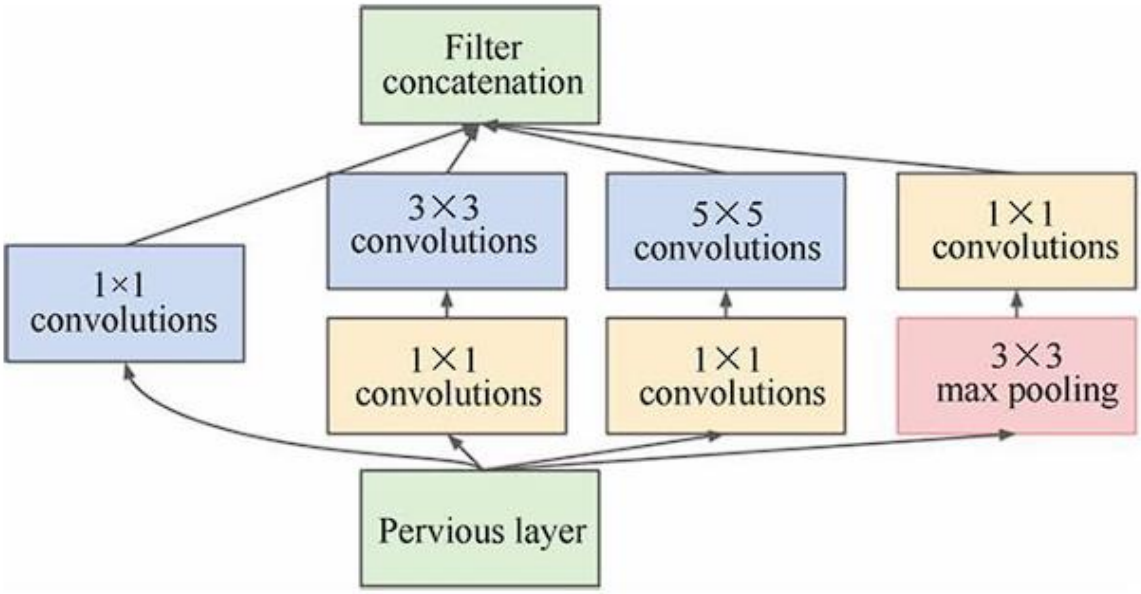


Figure 2-17 architecture d'un module "inception"

CONCLUSION :

Dans ce chapitre nous avons présenté en détails le principe de l'apprentissage automatique et les réseaux de neurones, ainsi que les réseaux de neurones profonds et les différents modèles existants. Par la suite, nous avons introduit le principe et les architectures des réseaux révolutionnaires utilisés dans le domaine de la vision par ordinateur qui sont les CNN.

Ou dans le prochain chapitre nous utiliserons une architecture CNN basée sur le model **GoogleNet**dont on a parlé juste avant.

Chapitre 3 : Réalisation d'un Système de Reconnaissance Faciale

3.1 Sélection des outils et des technologies

3.1.1 Choix du langage de programmation et des bibliothèques

Le choix du langage de programmation pour le développement d'un système de reconnaissance faciale est crucial pour assurer à la fois l'efficacité du développement et la performance du système final. Dans cette section, nous justifions l'utilisation de Python comme langage principal pour le développement de notre système de reconnaissance faciale, et nous présentons les principales bibliothèques Python pertinentes pour ce projet.

- **Justification de l'utilisation de Python pour le développement du système :**

Simplicité et lisibilité : Python est réputé pour sa syntaxe claire et sa facilité de lecture, ce qui en fait un choix idéal pour le développement de projets complexes comme la reconnaissance faciale, où la compréhension du code est essentielle.

Large disposition de bibliothèques : Python bénéficie d'un vaste nombre de bibliothèques spécialisées dans le traitement d'images et la vision par ordinateur, ce qui facilite le développement de fonctionnalités avancées de reconnaissance faciale.

Support pour l'apprentissage automatique : Python est largement utilisé dans le domaine de l'apprentissage automatique, avec des bibliothèques populaires telles que TensorFlow, PyTorch et scikit-learn, ce qui nous permet d'intégrer facilement des modèles d'apprentissage automatique dans notre système de reconnaissance faciale.

Portabilité : Python est un langage multiplateforme, ce qui signifie que notre système de reconnaissance faciale peut être facilement déployé sur différentes plateformes, y compris les systèmes Windows, macOS et Linux.

- **Présentation des bibliothèques Python pertinentes pour la reconnaissance faciale :**

-OpenCV (Open Source Computer Vision Library) : OpenCV est une bibliothèque open-source largement utilisée pour le traitement d'images et la vision par ordinateur. Elle offre des fonctionnalités avancées telles que la détection de visages, la reconnaissance d'objets et le suivi de mouvement, ce qui en fait un choix incontournable pour le développement de systèmes de reconnaissance faciale.

-dlib :dlib est une bibliothèque Python populaire qui propose des implémentations hautement optimisées d'algorithmes de vision par ordinateur et de machine learning. Elle offre

notamment des fonctionnalités de détection de visages et de reconnaissance faciale basées sur des modèles pré-entraînés.

-**TensorFlow :**TensorFlow est une bibliothèque d'apprentissage automatique développée par Google, largement utilisée pour la création et l'entraînement de réseaux neuronaux profonds. Nous pouvons l'utiliser pour développer des modèles de reconnaissance faciale basés sur l'apprentissage profond, tels que les réseaux de neurones convolutifs (CNN) et les réseaux de neurones siamois.

-scikit-image :scikit-image est une bibliothèque Python spécialisée dans le traitement d'images, offrant une large gamme de fonctionnalités pour la manipulation, la transformation et l'analyse d'images. Bien que moins axée sur la reconnaissance faciale que les autres bibliothèques mentionnées, elle peut être utile pour certains aspects du prétraitement d'images.

En choisissant Python comme langage de programmation principal et en utilisant les bibliothèques mentionnées ci-dessus, nous pouvons bénéficier d'un environnement de développement riche en fonctionnalités et de ressources pour la mise en œuvre efficace de notre système de reconnaissance faciale.

3.1.2 Environnement de développement

- Configuration de l'environnement de développement : installation des bibliothèques, gestion des dépendances, etc.

  (screen installation de bibliothèque python ,CMake ….)

  Installation de Python :

Nous installerons La version 3.11 64bit Windows de python, pour l'installer il suffit de télécharger l'installeur sur le site https://www.python.org/

Installation de l'éditeur de code VSC (Visual Studio Code):

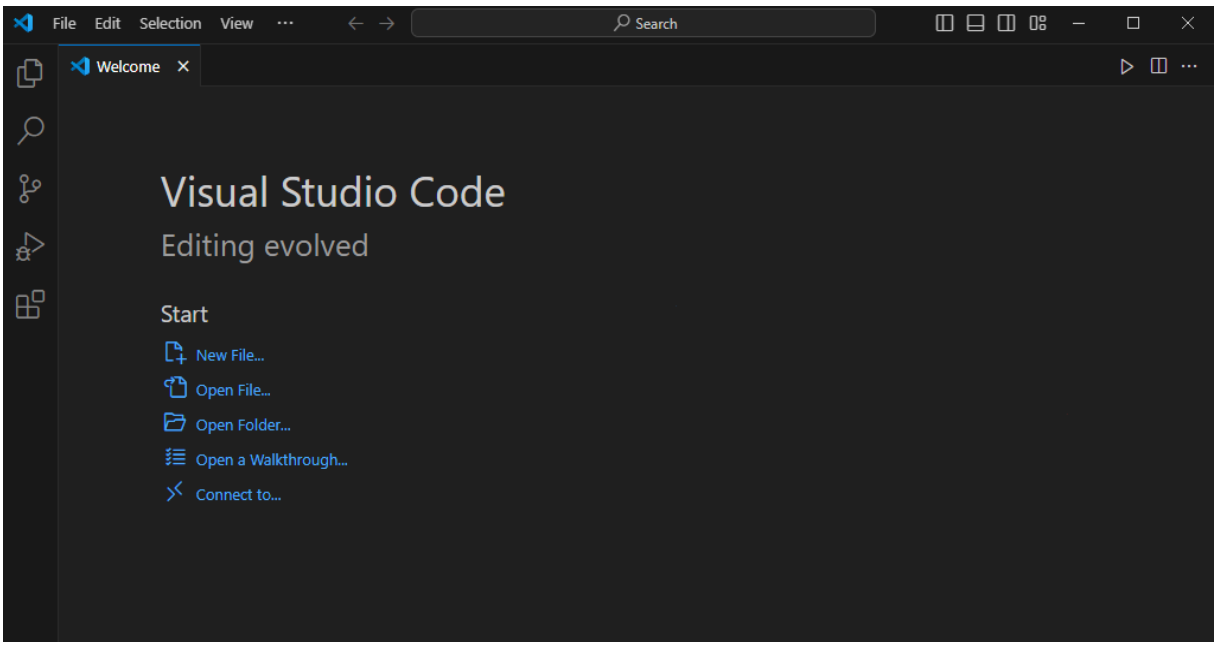


Figure 3-1 Editeur de code Visual Studio Code

Visual Studio Code appartenant à Microsoft, nous servira d'éditeur pour écrire notre code d'application et gérer nos terminaux.

Installation de Visual Studio :

L'installation de Visual Studio a pour but de collecter certains packages Python et C++ pour pouvoir installer les bibliothèques nécessaire pour notre réalisation.

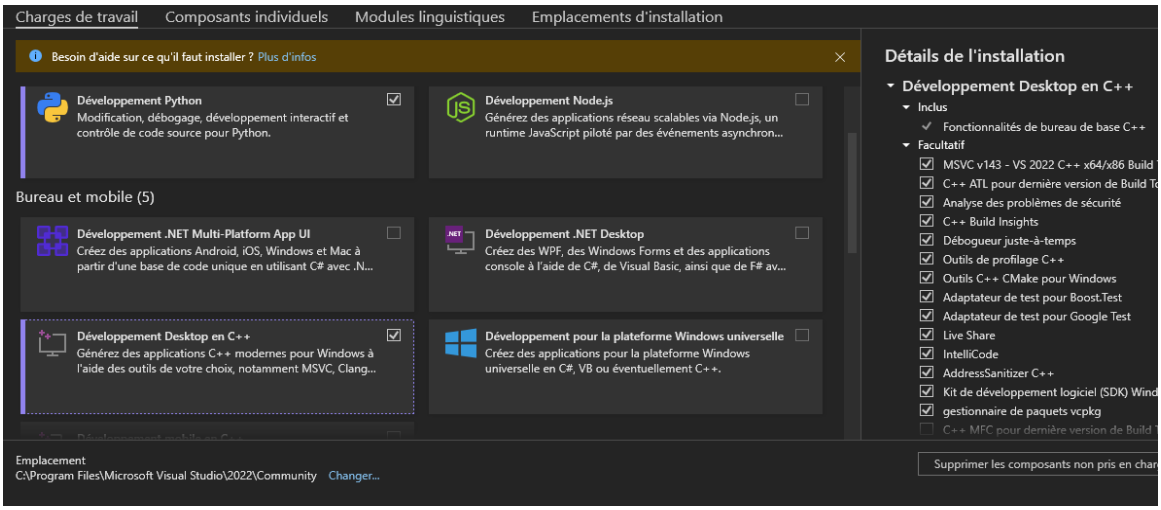


Figure 3-2 Installation de packages Visual Studio

Installation de bibliothèque python nécessaire :

CMake :

CMake est une distribution binaire en C++ obligatoire pour le fonctionnement des fonctions intégrés dans les bibliothèques utilisés on commence par télécharger le package CMake sur https://cmake.org/download/ Version 3.29.3 Win86_64bit.

Binary distributions:

Platform	Files
Windows x64 Installer:	cmake-3.29.3-windows-x86_64.msi

Figure3-3 : Package CMake

Puis dans le CMD (Command Prompt) :

Pour intégrer une distribution à python on utilise la commande **pipinstall (distribution)**

```
C:\Users\Sego>pip install cmake
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: cmake in c:\users\sego\appdata\roaming\python\python311\site-packages (3.29.2)
```

Figure3-4 : Installation de bibliothèque sur Python

Dlib :

Dlib est une bibliothèque logicielle en C++ avec des interfaces pour Python. Elle est largement utilisée dans les domaines de la vision par ordinateur, de l'apprentissage automatique et de la reconnaissance de formes.

```
C:\Users\Sego>pip install dlib
Defaulting to user installation because normal site-packages is not writeable
Collecting dlib
  Using cached dlib-19.24.4.tar.gz (3.3 MB)
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Building wheels for collected packages: dlib
  Building wheel for dlib (pyproject.toml) ... done
  Created wheel for dlib: filename=dlib-19.24.4-cp311-cp311-win_amd64.whl size=2865
6b818c805abfe9a6941daea716b8db615ac031
  Stored in directory: c:\users\sego\appdata\local\pip\cache\wheels\0d\98\d7\6152ef
cf451bf
Successfully built dlib
Installing collected packages: dlib
Successfully installed dlib-19.24.4
```

Figure3-5 : Intégration de la bibliothèque Dlib

Après avoir intégré les bibliothèques nous intégrons les extensions requises sur l'éditeur Visual Studio Code :

Extension	Définition
C/C++	Ajoute le support pour le développement en C et C++ avec des fonctionnalités telles que l'autocomplétion et le debugging.
CMake	Offre des outils pour travailler avec des fichiers CMake, utilisés pour gérer le processus de construction du projet.
CodeRunner	Permet d'exécuter du code en plusieurs langages directement depuis l'éditeur.
IntelliCode	Fournit des suggestions de code intelligentes basées sur les pratiques de codage courantes et les algorithmes de machine learning.
OpenCVSnippets	Offre des extraits de code pour simplifier l'utilisation de la bibliothèque OpenCV dans vos projets.
Prettier	Un formateur de code qui permet de maintenir un style de code cohérent en le formatant automatiquement.
Pylance	Un serveur de langage Python performant, offrant des fonctionnalités avancées de type checking et d'autocomplétion.
Python	Ajoute le support pour le développement en Python, avec des fonctionnalités comme l'autocomplétion, le linting et le debugging.
Python Debugger	Extension pour déboguer les applications Python directement depuis Visual Studio Code.
Python Image Preview	Permet de pré visualiser les images directement dans l'éditeur lors de la manipulation de fichiers d'image en Python.

Tableau 4 : Extensions Optimisant la programmation.

Ces extensions nous permettent d'interpréter les images et les lignes de code plus facilement ayant pour but d'enrichir notre environnement de travail

3.2 Implémentation du système de reconnaissance faciale

3.2.1 Schématisation du réseau neuronal : Création du modèle CNN :

```
importtensorflow as tf
fromtensorflow.keras.models import Sequential
fromtensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
```

Initialisation du modèle séquentiel :

```
model = Sequential()
```

Première couche de convolution :

```
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(28, 28, 1)))
```

Première couche de pooling :

```
model.add(MaxPooling2D(pool_size=(2, 2)))
```

Deuxième couche de convolution :

```
model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
```

Deuxièmecouche de pooling :

```
model.add(MaxPooling2D(pool_size=(2, 2)))
```

Couche de flattening pour transformer les matrices 2D en un vecteur 1D :

```
model.add(Flatten()) Couche dense (complètement connectée)

model.add(Dense(128, activation='relu'))
```

Couche de sortie avec 10 neurones (correspondant aux 10 classes, par exemple pour MNIST) :

```
model.add(Dense(10, activation='softmax'))
```

Compilation du modèle :

```
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

Affichage du résumé du modèle :

```
model.summary()
```

3.2.2 Entrainement du modèle :

```
fromtensorflow.keras.datasets import mnist
```

Chargement des données MNIST

```
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

Prétraitement des données

```
x_train = x_train.reshape(x_train.shape[0], 28, 28,
1).astype('float32') / 255

x_test = x_test.reshape(x_test.shape[0], 28, 28, 1).astype('float32')
/ 255
```

Entraînement du modèle

```
model.fit(x_train, y_train, epochs=10, batch_size=32,
validation_data=(x_test, y_test))
```

Évaluation du modèle

```
test_loss, test_acc = model.evaluate(x_test, y_test)

print(f"Test accuracy: {test_acc}")
```

Cet exemple montre comment construire, compiler, entraîner et évaluer un modèle CNN basique pour la classification d'images.

1. Convolution :

Les couches convolutionnelles sont la brique de base des CNN. Elles appliquent des filtres (ou noyaux) sur les images d'entrée pour extraire des caractéristiques locales telles que les contours, les textures, et d'autres motifs.

Opération de Convolution : Un filtre de petite taille (comme 3x3 ou 5x5) est appliqué sur l'image d'entrée. Chaque filtre glisse sur l'image et calcule un produit scalaire entre les valeurs du filtre et les pixels de l'image, produisant une carte de caractéristiques.

Activation ReLU : Une fonction d'activation non-linéaire, souvent ReLU (RectifiedLinear Unit), est appliquée pour introduire des non-linéarités, permettant au modèle de capturer des relations plus complexes.

2. Pooling :

Les couches de pooling réduisent les dimensions spatiales de la carte de caractéristiques, tout en conservant les informations importantes. Cela permet de réduire le nombre de paramètres et la charge computationnelle, et d'introduire une invariance à la translation.

Max Pooling : La méthode la plus courante est le max pooling, qui prend le maximum d'une région (comme 2x2) de la carte de caractéristiques.

3. Flattening :

Après plusieurs couches de convolution et de pooling, la sortie est aplatie (flattened) en un vecteur 1D. Cela prépare les données pour les couches pleinement connectées (dense layers).

4. FullyConnectedLayers :

Les couches pleinement connectées ressemblent aux réseaux de neurones traditionnels. Chaque neurone est connecté à tous les neurones de la couche précédente. Ces couches apprennent des combinaisons de caractéristiques extraites par les couches convolutionnelles.

Dense Layers : Elles combinent les caractéristiques extraites pour prendre des décisions. Par exemple, elles peuvent reconnaître des motifs spécifiques d'un visage.

Dropout : Une technique de régularisation utilisée pour éviter le surapprentissage. Pendant l'entraînement, une fraction aléatoire des neurones est ignorée (mise à zéro).

5. Output Layer :

La couche de sortie produit les prédictions finales.

Softmax : Pour la classification multi-classes, une fonction d'activation softmax est souvent utilisée pour produire des probabilités de chaque classe. Pour la reconnaissance faciale, chaque classe pourrait correspondre à une personne différente.

Fonctionnement Global pour la Reconnaissance Faciale

Prétraitement de l'Image : L'image du visage est redimensionnée à une taille standard (par exemple, 64x64 pixels).

Extraction de Caractéristiques : Les couches convolutionnelles et de pooling extraient des caractéristiques importantes de l'image (contours, textures, etc.).

Classification : Les couches pleinement connectées combinent ces caractéristiques pour classifier l'image dans une des classes prédéfinies (personnes connues).

Prédiction : Le modèle donne la classe avec la probabilité la plus élevée, indiquant l'identité prédite du visage.

Schéma Résumé :

Entrée	Image d'un visage (64x64x3)
Couches Convolutionnelles	Extraction de caractéristiques locales
Couches de Pooling	Réduction de la dimensionnalité, invariance à la translation
Flattening	Transformation en vecteur 1D
Couches FullyConnected	Combinaison de caractéristiques pour classification
Sortie	Prédiction de la classe (identité du visage)

Tableau 5 : Schéma Résumé du modèle CNN

3.2.3 Configuration des Générateurs d'Images :

Les générateurs d'images sont utilisés pour prétraiter et augmenter les images en temps réel pendant l'entraînement. Cela permet d'améliorer la généralisation du modèle en créant des variations des images d'entraînement.

Générateur pour les Données d'Entraînement :

```
fromtensorflow.keras.preprocessing.image
importImageDataGenerator

train_datagen = ImageDataGenerator(
rescale=1./255,          # Normalise les valeurs des pixels entre 0 et
1
shear_range=0.2,         # Applique des cisaillements
zoom_range=0.2,          # Applique des zooms
horizontal_flip=True     # Permet les retournements horizontaux
)
```

- `rescale=1./255` : Cette opération normalise les pixels de l'image pour qu'ils aient des valeurs comprises entre 0 et 1 (au lieu de 0 à 255). Cela aide le modèle à converger plus rapidement.
- `shear_range=0.2` : Applique une transformation de cisaillement aux images. Cela peut aider le modèle à être invariant aux variations d'angle dans les images.
- `zoom_range=0.2` : Applique un zoom aléatoire aux images. Cela rend le modèle plus robuste aux variations de taille des objets dans les images.
- `horizontal_flip=True` : Retourne aléatoirement les images horizontalement. Cela permet de simuler les variations dans la direction des objets.

3.2.4Générateur pour les Données de Test :

```
test_datagen = ImageDataGenerator(rescale=1./255)
```

Pour les données de test, seules les valeurs de pixel sont normalisées. On n'applique pas d'augmentations pour éviter d'introduire des variations qui ne sont pas présentes dans les données réelles de test.

3.2.5Chargement des Données

Ces générateurs sont ensuite utilisés pour charger les images à partir des répertoires spécifiés et pour créer des lots d'images et d'étiquettes.

Donnéesd'Entraînement :

```
training_set = train_datagen.flow_from_directory(
'dataset/training_set',
target_size=(64, 64),  # Redimensionne les images à 64x64 pixels
batch_size=32,         # Nombre d'images par lot
class_mode='categorical'     # Indique que les étiquettes sont
catégorielles (pour classification multi-classes)
)
```

- `flow_from_directory` : Cette méthode charge les images à partir du répertoire spécifié (`'dataset/training_set'`).
- `target_size=(64, 64)` : Redimensionne toutes les images à 64x64 pixels. Cela assure une taille uniforme des images pour l'entrée du modèle.
- `batch_size=32` : Définit la taille des lots d'images qui seront passés au modèle pendant l'entraînement.
- `class_mode='categorical'`: Indique que les étiquettes sont catégorielles, ce qui signifie que chaque image appartient à une des classes multiples. Cela est utilisé pour la classification multi-classes.

Données de Test :

```
test_set = test_datagen.flow_from_directory(
'dataset/test_set',
target_size=(64, 64),
batch_size=32,
class_mode='categorical'
)
```

- `flow_from_directory` : Charge les images à partir du répertoire spécifié(`'dataset/test_set'`).
- `target_size=(64, 64)` : Redimensionne toutes les images à 64x64 pixels.
- `batch_size=32` : Définit la taille des lots d'images pour la validation.

- `class_mode='categorical'`: Indique que les étiquettes sont catégorielles pour la classification multi-classes.

- **Sauvegarde et chargement du modèle :**

```
model.save('face_recognition_model.h5')  #Sauvegarde
```

```
fromtensorflow.keras.modelsimportload_model

model = load_model('face_recognition_model.h5') #Chargement
```

3.2.6Developpement du programme de reconnaissance faciale :

- **Acquisitions de données d'entrée :**

  La première partie du programme sera responsable de lire les entrées et séparer les noms et les images en deux listes séparés. Ce qui nous amène à créer une classe spécifique qui sera responsable à accomplir cette tâche.

1. Import des Modules :

Le programme commence par importer les modules nécessaires. Cependant, dans le code que tu as fourni, les modules ne sont pas explicitement importés. Assurons-nous d'importer les modules requis :

```
import os

import cv2
```

`os` : Ce module fournit des fonctions pour interagir avec le système d'exploitation, comme la navigation dans les répertoires.

`cv2`: Ce module est OpenCV, une bibliothèque populaire pour la vision par ordinateur.

2. Définition du Chemin :

`path =` 'persons'

`path` : La variable path contient le chemin du répertoire où se trouvent les images des personnes.

3. Initialisation des Listes :

```
images = []

classNames = []
```

`images` : Cette liste va stocker les images lues à partir du répertoire.

`classNames` : Cette liste va stocker les noms des fichiers d'image sans leurs extensions.

4. Lecture de la Liste des Fichiers :

```
personsList = os.listdir(path)
```

`os.listdir(path)`: Cette fonction retourne une liste de tous les fichiers et dossiers dans le répertoire spécifié par path. Dans ce cas, elle retourne une liste de noms de fichiers d'image dans le répertoire`'persons'`.

5. Boucle sur les Fichiers :

```
for cl inpersonsList:

curPersonn = cv2.imread(f'{path}/{cl}')

images.append(curPersonn)

classNames.append(os.path.splitext(cl)[0])
```

Boucle for : La boucle itère sur chaque fichier dans `personsList`.

`cl` : Représente le nom actuel du fichier d'image dans la liste.

Lecture de l'Image :

`cv2.imread(f'{path}/{cl}')` : Utilise OpenCV pour lire l'image à partir du chemin complet(`f'{path}/{cl}'`). Cela combine le chemin de base path avec le nom du fichier`cl`.

`curPersonn` : Contient l'image lue.

`images.append(curPersonn)` : Ajoute l'image lue à la liste images.

Extraction du Nom de Fichier Sans Extension :

`os.path.splitext(cl)[0]` : Sépare le nom du fichier et son extension. [0] sélectionne la partie sans l'extension.

`classNames.append(os.path.splitext(cl)[0])` : Ajoute le nom de fichier sans l'extension à laliste`classNames`.

6. Affichage des Noms des Fichiers :

```
print(classNames)
```

Affiche la liste des noms de fichiers sans leurs extensions.

Voici le programme complet avec des commentaires supplémentaires :

```
import os

import cv2

# Chemin du répertoire contenant les images des personnes

path = 'persons'

# Initialisation des listes pour stocker les images et les noms des fichiers

images = []

classNames = []

# Récupération de la liste des fichiers dans le répertoire

personsList = os.listdir(path)

# Parcours de chaque fichier dans la liste

for cl inpersonsList:

# Lecture de l'image actuelle à partir du chemin complet

curPersonn = cv2.imread(f'{path}/{cl}')

# Ajout de l'image à la liste des images

images.append(curPersonn)

# Extraction du nom du fichier sans l'extension et ajout à la liste des noms

classNames.append(os.path.splitext(cl)[0])

# Affichage des noms des fichiers sans leurs extensions

print(classNames)
```

- ***Encodage et conversion d'images format BGR vers RGB :***

Fonction`findEncodeings :`

```
deffindEncodeings(images):
encodeList = []
forimgin images:
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
encode = face_recognition.face_encodings(img)[0]
encodeList.append(encode)
```

```
returnencodeList
```

1. **Définition de la Fonction :**

```
deffindEncodeings(images):
```

- La fonction`findEncodeings`prend une liste d'images en entrée (`images`).

2. **Initialisation de la Liste d'Encodages :**

```
encodeList = []
```

- `encodeList`est une liste vide qui sera utilisée pour stocker les encodages de visage.

3. **Boucle sur les Images :**

```
forimginimages:
```

- La boucle for itère sur chaque image dans la liste`images`.

4. **Conversion de l'Image en RGB :**

```
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
```

- OpenCV charge les images en format BGR par défaut. Cette ligne convertit l'image du format BGR au format RGB, car la bibliothèque face_recognition utilise le format RGB.

5. **Encodage du Visage :**

```
encode = face_recognition.face_encodings(img)[0]
```

- `face_recognition.face_encodings(img)`renvoie une liste d'encodages de visage pour chaque visage détecté dans l'image. [0] sélectionne le premier encodage, supposant qu'il y a un seul visage par image.

6. **Ajout de l'Encodage à la Liste :**

```
encodeList.append(encode)
```

- L'encodage obtenu est ajouté à la liste`encodeList`.

7. **Retour de la Liste des Encodages :**

```
returnencodeList
```

- La fonction renvoie la liste des encodages après avoir traité toutes les images.

Utilisation de la Fonction :

```
encodeListKnown = findEncodeings(images)
print('Encoding Complete.')
```

1. **Appel de la Fonction :**

```
encodeListKnown = findEncodeings(images)
```

- La fonction`findEncodeings`est appelée avec la liste images (qui a été remplieprécédemment avec les images des visages). Le résultat, qui est une liste d'encodages de visage, est stocké dans `encodeListKnown`.

2. **Affichage d'un Message de Confirmation :**

```
print('Encoding Complete.')
```

- Affiche un message pour indiquer que le processus d'encodage des visages est terminé.

Voici le programme complet avec des commentaires supplémentaires :

```
import os
import cv2
importface_recognition

# Chemin du répertoire contenant les images des personnes
path = 'persons'

# Initialisation des listes pour stocker les images et les noms
des fichiers
```

```
images = []
classNames = []

# Récupération de la liste des fichiers dans le répertoire
personsList = os.listdir(path)

# Parcours de chaque fichier dans la liste
for cl inpersonsList:
# Lecture de l'image actuelle à partir du chemin complet
curPersonn = cv2.imread(f'{path}/{cl}')

# Ajout de l'image à la liste des images
images.append(curPersonn)

# Extraction du nom du fichier sans l'extension et ajout à la liste des noms
classNames.append(os.path.splitext(cl)[0])

# Affichage des noms des fichiers sans leurs extensions
print(classNames)

# Définition de la fonction pour trouver les encodages des images
deffindEncodeings(images):
encodeList = []
forimgin images:
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
encode = face_recognition.face_encodings(img)[0]
encodeList.append(encode)
returnencodeList

# Utilisation de la fonction pour trouver les encodages des images chargées
encodeListKnown = findEncodeings(images)
print('Encoding Complete.')
```

- **Acquisition du signal vidéo et de son traitement :**

  Nous définissons un capteur vidéo .Suite à la réception du signal vidéo, nous convertirons le signal en format RGB puis nous commencerons son traitement pour ainsi détecter la position des visages, leurs formes et angles.

1. Initialisation de la Capture Vidéo :

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

- **cv2.VideoCapture(0)** : Initialise la capture vidéo à partir de la webcam. L'argument 0 fait référence à la première webcam connectée à l'ordinateur. Si tu avais plusieurs caméras, tu pourrais utiliser 1, 2, etc.

2. Boucle Principale :

```
while True:

    _, img = cap.read()
```

- **whileTrue:** : Boucle infinie qui permet de capturer des images en continu à partir de la webcam.
- **_, img = cap.read()** : Lit une image de la webcam. _ est utilisé pour ignorer la première valeur retournée (qui est un booléen indiquant si la capture a réussi), et img contient l'image capturée.

3. Redimensionnement et Conversion de l'Image :

```
imgS = cv2.resize(img, (0,0), None, 0.25, 0.25)

imgS = cv2.cvtColor(imgS, cv2.COLOR_BGR2RGB)
```

- **cv2.resize(img, (0,0), None, 0.25, 0.25)** : Redimensionne l'image img à 25% de sa taille d'origine pour accélérer le traitement.
- **cv2.cvtColor(imgS, cv2.COLOR_BGR2RGB)** : Convertit l'image redimensionnée du format BGR (par défaut dans OpenCV) au format RGB (utilisé par face_recognition).

4. Détection et Encodage des Visages :

```
faceCurentFrame = face_recognition.face_locations(imgS)

encodeCurentFrame = face_recognition.face_encodings(imgS,
faceCurentFrame)
```

- **face_recognition.face_locations(imgS)** : Trouve les emplacements des visages dans l'image redimensionnée imgS. Cette fonction retourne une liste de positions des visages.
- **face_recognition.face_encodings(imgS, faceCurentFrame)** : Calcule les encodages de visage pour chaque visage détecté dans imgS en utilisant les emplacements trouvés précédemment (faceCurentFrame).

5. Comparaison et Annotation des Visages :

```
forencodeface, faceLoc in zip(encodeCurentFrame, faceCurentFrame):

matches = face_recognition.compare_faces(encodeListKnown, encodeface)

faceDis = face_recognition.face_distance(encodeListKnown, encodeface)

matchIndex = np.argmin(faceDis)
```

- **for encodeface, faceLoc in zip(encodeCurentFrame, faceCurentFrame):** : Itère simultanément sur les encodages des visages détectés et leurs positions.
- **face_recognition.compare_faces(encodeListKnown, encodeface)** : Compare l'encodage du visage détecté (encodeface) avec les encodages connus (encodeListKnown) et retourne une liste de booléens indiquant les correspondances.
- **face_recognition.face_distance(encodeListKnown, encodeface)** : Calcule les distances entre l'encodage du visage détecté et les encodages connus. Une distance plus petite indique une meilleure correspondance.
- **matchIndex = np.argmin(faceDis)** : Trouve l'indice de la distance la plus petite, correspondant au visage le plus proche dans encodeListKnown.

6. Annotation de l'Image :

```
if matches[matchIndex]:

name = classNames[matchIndex].upper()

print(name)

y1, x2, y2, x1 = faceLoc
```

```
            y1, x2, y2, x1 = y1*4, x2*4, y2*4, x1*4
cv2.rectangle(img, (x1, y1), (x2, y2), (0,0,255), 2)
cv2.rectangle(img,     (x1,y2-35),     (x2,y2),     (0,0,255),
cv2.FILLED)
cv2.putText(img,          name,          (x1+6,          y2-6),
cv2.FONT_HERSHEY_COMPLEX, 1, (255,255,255), 2)
```

- **if matches[matchIndex]:** : Vérifie si le visage détecté correspond à un visage connu (avec une distance minimale).
- **name = classNames[matchIndex].upper()** : Récupère le nom correspondant à l'encodage trouvé et le met en majuscules.
- **print(name)** : Affiche le nom de la personne reconnue.

- **Coordonnées de la boîte de délimitation** :
    - y1, x2, y2, x1 = faceLoc : Récupère les coordonnées du visage détecté.
    - y1, x2, y2, x1 = y1*4, x2*4, y2*4, x1*4 : Multiplie les coordonnées par 4 pour revenir à l'échelle originale de l'image (car l'image a été redimensionnée à 25% de sa taille d'origine).
- **Dessiner des rectangles et afficher le nom** :
    - cv2.rectangle(img, (x1, y1), (x2, y2), (0,0,255), 2) : Dessine un rectangle autour du visage reconnu en rouge.
    - cv2.rectangle(img, (x1,y2-35), (x2,y2), (0,0,255), cv2.FILLED) : Dessine un rectangle rempli pour le fond du texte du nom.
    - cv2.putText(img, name, (x1+6, y2-6), cv2.FONT_HERSHEY_COMPLEX, 1, (255,255,255), 2) : Affiche le nom de la personne reconnue dans le rectangle dessiné.

7. Affichage et Attente :

```
cv2.imshow('Face Recognition', img)
cv2.waitKey(1)
```

- **cv2.imshow('Face Recognition', img)** : Affiche l'image annotée dans une fenêtre intitulée "Face Recognition".
- **cv2.waitKey(1)** : Attends 1 milliseconde avant de capturer la prochaine image. Cela permet de rafraîchir l'image affichée et de maintenir la boucle en cours d'exécution.

3.3 Test et Résultats Acquis :

Dans cette partie nous allons réaliser différents tests avec deux camera différentes pour ainsi obtenir un résultat plus étendu et également réaliser des tests sous différentes conditions :

	Première camera	**Deuxième camera**
Type	Webcam	Caméra de Téléphone
Résolution	1280x720 pixels	1920x1080 pixels
Fréquence de capture	25 Img/s	30 img/s

Tableau 6 : Caractéristiques des capteurs utilisés

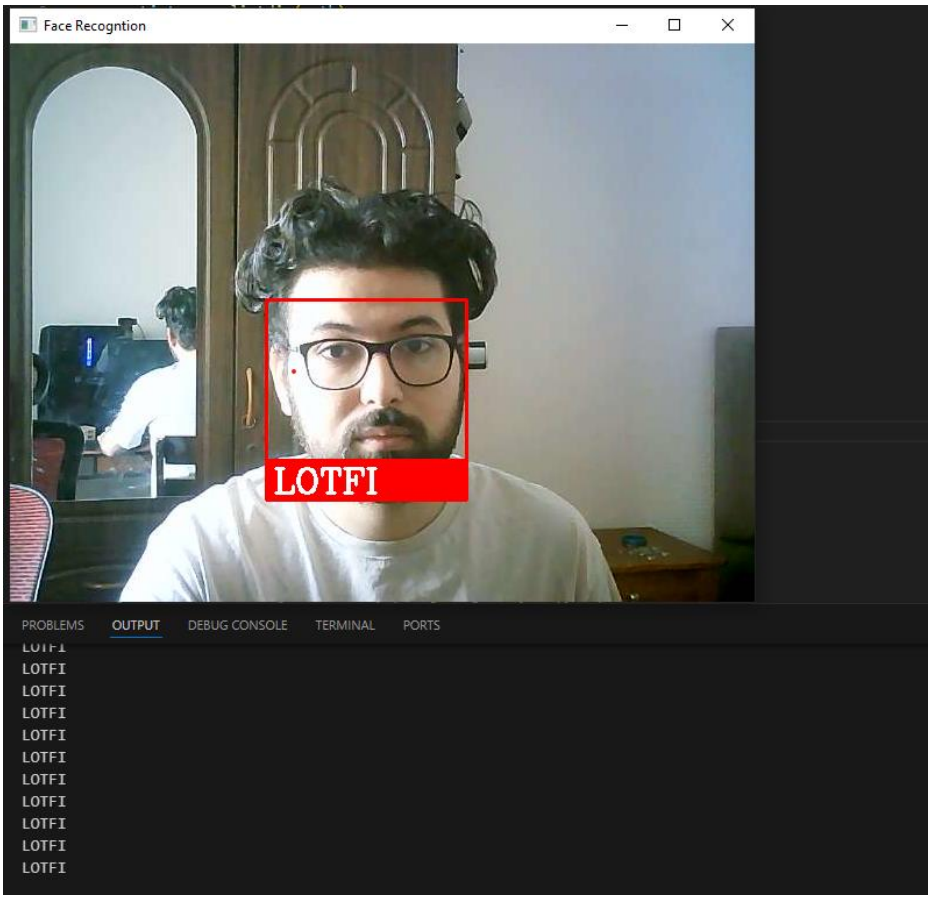


Figure 3-6 Première personne

Résultat d'une personne :

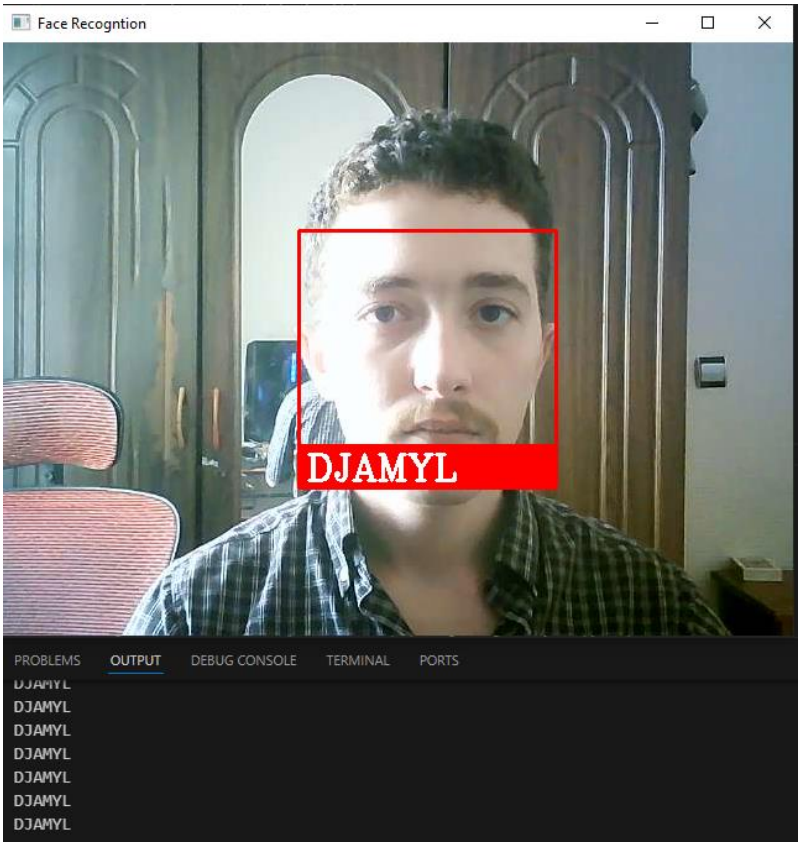


Figure 3-7 Deuxièmes personnes

Test sur la deuxième personne :

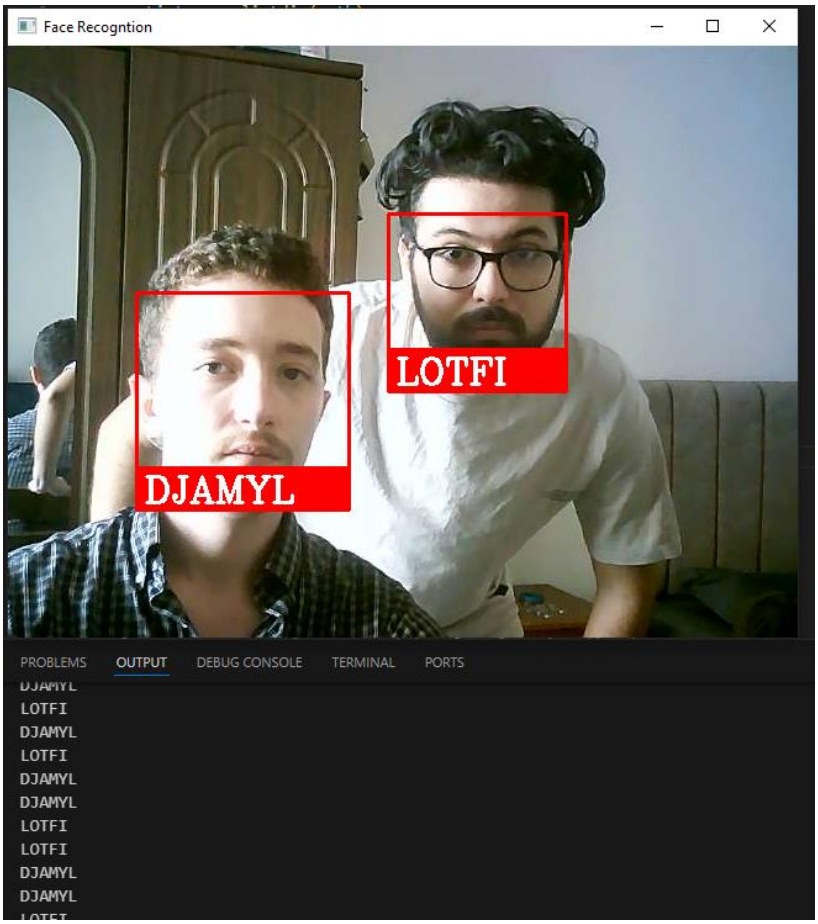
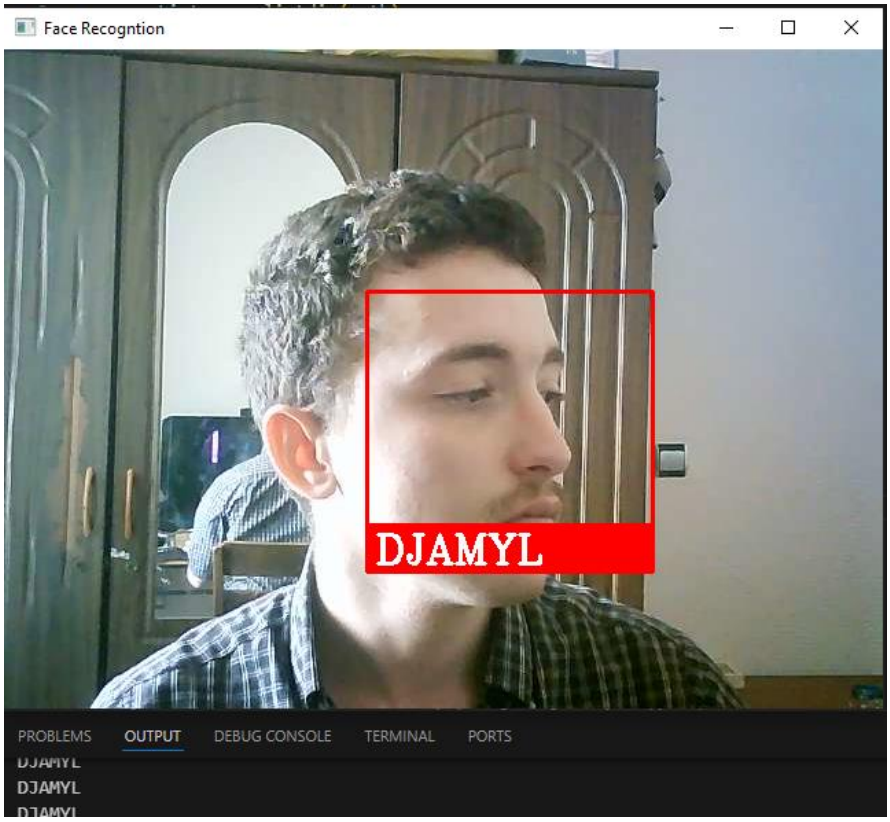


Figure 3-8 Test de reconnaissance de deux personnes en même temps

Figure 3-9 Test de profil



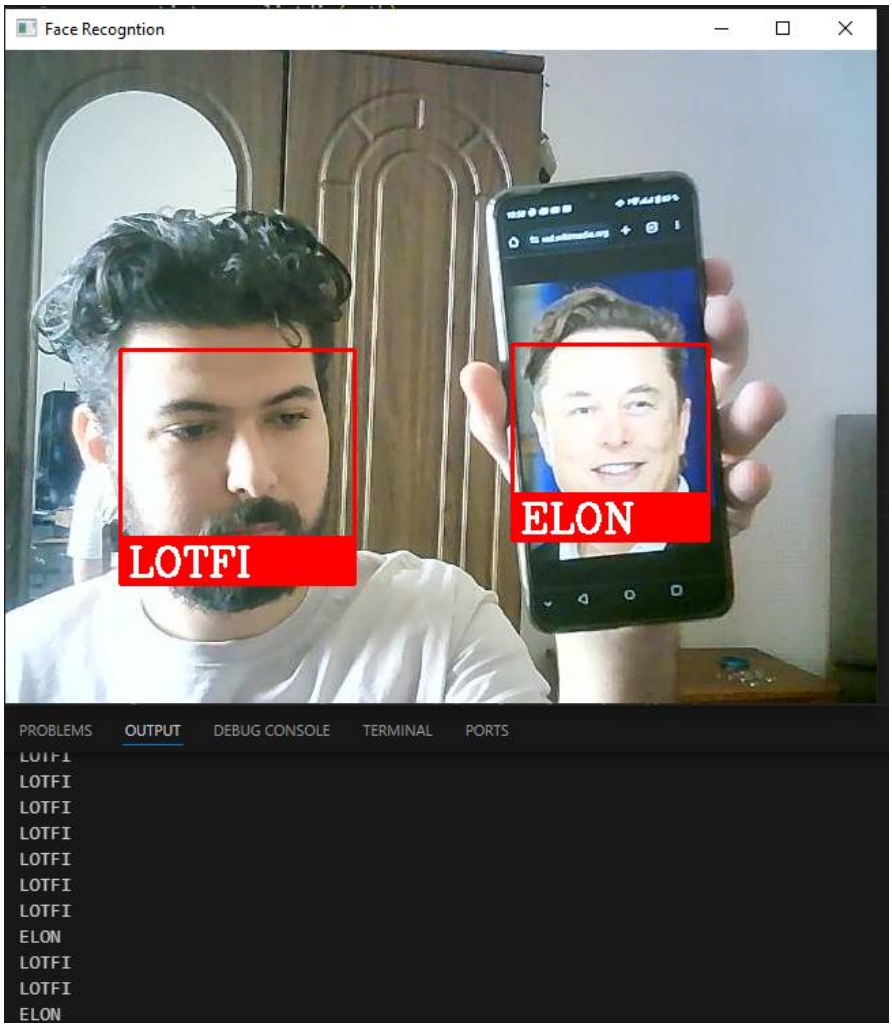


Figure 3-10 Test reconnaissance à travers téléphone

D'après nos résultats nous pouvons constater que :

1- La reconnaissance faciale s'effectue même si la personne porte des lunettes
2- On obtient les noms des personnes dans la console.
3- La reconnaissance faciale s'effectue même avec deux personnes en même temps voir même plus.
4- En effet la reconnaissance s'effectue avec succès même si l'angle de la camera ne laisse montres qu'une face du visage.
5- On remarque que la reconnaissance s'effectue même sur des visages qui se trouvent sur le téléphone.

Résultats :

	Webcam	**Caméra Téléphone**
Pourcentage de précision annoncé	99.38%	99.38%
Pourcentage de précision réel	97.52%	98.79%
Distance de détection annoncée	0.6 m	0.6 m
Distance de détection réelle	1m	0.9 m
Rappel	100%	100%
F1-score	98.74%	99.39%

Tableau 7 : Résultats des tests

Remarque :

Après des entrainements quotidiens réalisésd'une durée de 2 mois àl'aide de 50 images de différentes personnes nous avons pu optimiser la machine pour avoir de tels résultats en temps réel :

1. Nous constatons que les résultats dépendent intrinsèquement de la qualité du capteur utilisé lors de nos tests

2. Nous constatons également que l'exactitude des résultats dépendent de la distance par rapport à l'appareille de mesure

Conclusion :

Dans ce chapitre nous avons réalisé un système de reconnaissance faciale fonctionnel et précis, dont les tests sont aboutis avec succès dans plusieurs situations possibles. Ainsi nous répondons à notre problématique principale qui est de : « Comment réaliser un système de reconnaissance faciale précis ? »

Conclusion :

La reconnaissance faciale, en tant que technique émergente, présente un potentiel significatif pour diverses applications pratiques, mais elle est également accompagnée de défis techniques et éthiques. À travers ce travail, nous avons exploré les méthodes, les défis et les implications de la reconnaissance faciale, en nous concentrant sur trois questions clés.

1. Méthodes et techniques efficaces pour la reconnaissance faciale en Python :

Nous avons identifié plusieurs techniques pour la reconnaissance faciale, telles que les algorithmes basés sur les réseaux de neurones convolutifs (CNN) et les modèles de deeplearning comme le système de reconnaissance faciale développé par le framework d'OpenCV. La mise en œuvre de ces techniques nécessite une combinaison de bibliothèques Python comme OpenCV, dlib, et TensorFlow, offrant ainsi une flexibilité et une précision accrues pour détecter et reconnaître les visages à partir de flux vidéo en temps réel.

2. Défis techniques et meilleures pratiques pour les surmonter :

Le développement d'un système de reconnaissance faciale comporte plusieurs défis, notamment la gestion de variations d'éclairage, d'angles de prise de vue, et de la qualité des images. De plus, la précision peut être affectée par la diversité ethnique et l'âge des individus. Pour surmonter ces obstacles, il est crucial d'avoir des bases de données diversifiées pour l'entraînement des modèles et d'utiliser des techniques d'augmentation de données pour simuler différentes conditions. L'optimisation des modèles et la validation croisée sont également essentielles pour améliorer la performance et réduire les biais.

3. Impacts potentiels sur la vie privée et les droits individuels :

L'adoption de la reconnaissance faciale soulève des préoccupations majeures concernant la vie privée et les droits individuels. Les risques incluent la surveillance de masse, la mauvaise utilisation des données biométriques, et les violations de la vie privée. Pour atténuer ces risques, il est important de mettre en place des mesures de protection des données, telles que l'anonymisation des informations, le cryptage des données, et des politiques de confidentialité rigoureuses. De plus, la transparence dans l'utilisation des techniques de reconnaissance faciale et l'engagement des parties prenantes dans le développement de réglementations éthiques sont essentiels pour équilibrer efficacité et respect des droits individuels.

Pour conclure, la reconnaissance faciale représente une avancée technologique significative avec de vastes applications, mais elle nécessite une approche équilibrée pour maximiser ses avantages tout en minimisant ses risques. En adoptant des techniques avancées, en surmontant les défis techniques par des pratiques rigoureuses, et en respectant les principes éthiques et légaux, nous pouvons développer des systèmes de reconnaissance faciale efficaces et responsables. Il est impératif de continuer à évoluer avec une vigilance accrue pour les impacts sur la société et les droits des individus, afin de garantir que cette technologie soit utilisée de manière bénéfique et équitable.

Bibliographie :

-[1] Handbook of Face Recognitionpar Stan Z. Li et Anil K. Jain.

-[2] IEEE Transactions on Neural Networks and Learning Systems.Eigenfaces for Recognition" par Turk et Pentland(1991).

-[3] IEEE Transactions on Neural Networks and Learning Systems.Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection. ParBelhumeur, P. N., Hespanha, J. P., &Kriegman, D. J. (1997).

- [4] A Comparative Study of Texture Measures with Classification Based on Featured Distributions. Pattern Recognition. Par Ojala, T., Pietikäinen, M., & Harwood, D. (1996).

-[5] Histograms of Oriented Gradients for Human Detection. Par Dalal, N., &Triggs, B. (2005).

- [6] Faces in the Wild: University of Massachusetts, Amherst. Par Huang, G. B., Ramesh, M., Berg, T., & Learned-Miller, E. (2007).

- [7] Deep Learning Face Attributes in the Wild. In Proceedings of the IEEE International Conference on Computer Vision (ICCV). Par Liu, Z., Luo, P., Wang, X., & Tang, X. (2015).

- [8] Face Recognition in Unconstrained Videos with Matched Background Similarity. Par Wolf, L., Hassner, T., &Maoz, I. (2011).

-[9] J.M, F.C, and C.Wang "Research on Emotion Recognition for Facial Expression" - 2014Livres

-[10] J.F.C, "Deformable Model Fitting" by Regularized Landmark Mean-Shift -2013

-[11] « Maximum Confidence » M Modeling for Face Recognition -2008

-[12] P.N.B, J.P.H, and D. J.K Eigenfaces vs. Fisherfaces -1997

-[13] Z. J, Y.Y, and M.L Face Recognition:Eigenface -1997

-[14] I. Chaibeddra et S. Madene 'Détection visuelle d'objets statiques et dynamiques dans un environnement de type route et classification en exploitant l'apprentissage profond' Université des Sciences et de la Technologie Houari Boumediene – 2019

-[15] L. Nehemy « Estimation de modèles autorégressifs vectoriels a noyaux a valeur operateur », docteur de l'université d'Évry val d'Essonne – 2015

-[16] A. Freedman. « statistical models : Theory and practice" Cambridge University Press – 2012

-[17] J.Dean "Big Data, Data Mining, and Machine Learning: Value Creation for Business Leaders - and Practitioners" -2014: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118691786.ch6.

-[18] C.Touzer "Les réseaux de neurones artificiels, introduction au connexionnisme » Collection de l'EERIE – 1992.

-[19] F.Rosenbaltt « The perceptron: a probabilistic model for information storage and organization in the brain" – 1988

-[20] G.Saporito "What is a perceptron?" – 2019 :https://towardsdatascience.com/what-is-a-perceptron-210a50190c3b

-[21] L.Deng, D.Yu "Deep Learning: methods and applications" -2014

-[22] D. Moualek " Deep Learning pour la classification des images » Thèse de doctorat, Université Abou BakrBelkaid – Tlemcen – 2017

-[23] M. Gregory Gelly. « Réseaux de neurones récurrents pour le traitement automatique de la parole '', Thèse de doctorat de l'Université Paris-Saclay préparée a l'Université Paris-Sud – 2017

-[24] Y.Andrew, M.Jordan « on discriminative vs generative classifiers : a comparison of logistic regression and naïve bayes. In advances in neural information processing systems" -2002

-[25] G. Hinton « Boltzmann Machines » University of Toronto, Toronto, ON, Canada - 2014.

-[26] H. Ackley, E. Hinton, J. Sejnowski, « A learning algorithm for boltzmann machines » Cognitive science - 1985.

-[27] G. Hinton « A Practical Guide to Training Restricted Boltzmann Machines » - 2010.

-[28] R. Salakhutdinov, A. Mnih, G. Hinton « Restricted boltzmann machines for collaborative filtering » - 2007.

-[29] R. Khandelwal. « Deep learning – deep belief network » - 2018: https://medium.com/datadriveninvestor/deep-learning-deep-belief-network-dbn-ab715b5b8afc

-[30] G. Hinton, S. Osindero « A fast learning algorithm for deep belief nets » Neural computation - 2006.

-[31] G. Hinton, R. Salakhutdinov. « Deep Boltzmann Machines » in Artificial Intelligence and Statistics - 2009.

-[32] N. Srivastava, R. Salakhutdinov « Multimodal learning with deep boltzmann machines » in Advances in neural information processing systems - 2012.

-[33] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio « Generative adversarial nets » in Advances in neural information processing systems - 2014.

-[34] L. Denton, S. Chintala, R. Fergus, al « Deep generative image models using a laplacian pyramid of adversarial networks » in Advances in neural information processing systems - 2015.

-[35] A. Radford, L. Metz, and S. Chintala, « Unsupervised representation learning with deep convolutional generative adversarial networks » - 2015.

-[36] Y. Bengio, E. Laufer, G. Alain, and J. Yosinski « Deep generative stochastic networks trainable by backprop » in International Conference on Machine Learning - 2014.

-[37] A.Makhzani, J. Shlens, N. Jaitly, I.Goodfellow, and B. Frey "Adversarial autoencoders" – 2015.

-[38] S.Albawi, T.Mohammed and S. Al-Zawi "Understanding of a convolutional neural network" – 2017

-[39] P. Monasse, K. Nadjahi " Découvrez les différentes couches d'un CNN » - 2020 : https://openclassrooms.com/fr/courses/4470531-classez-et-segmentez-des-donnees-visuelles/5083336-decouvrez-les-differentes-couches-dun-cnn

-[40] Prabhu « Understanding of convolutional neural network (CNN) Deep Learning" – 2018 : https://medium.com/@RaghavPrabhu/Understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148

-[41] R. Lambert " réseau de neurones convolutifs »-2019 : https://www.aspexit.com/reseau-de-neurones-on-va-eesayer-de-demystifier-un-peu-tout-ca-3/

-[42] D. Nene, A. Diane. « La reconnaissance des expressions faciales » Université 8 Mai 1945 – Guelma – 2019

-[43] S. Das "CNN Architectures: LeNet, AlexNet, VGG, GoogLeNet, ResNet and more….." – 2017 : https://medium.com/analytics-vidhya/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5

-[44] Y. LeCun, L.Bottou, Y. Bengio, P.Haffner « LeNet-5 – A Classic CNN Architecture" – 1990 : https://engmrk.com/lenet5-a-classic-cnn-architecture/

-[45] N.Foued " reconnaissance d'expression faciale à partir d'un visage réel » Université de 8 mai 1945 – Guelma -2019

-[46] A. Krizhevsky, G. Hinton, I. Sutskever " ImplementationAlexNet a l'aide de Keras" – 2012:https://engmrk.com/alexnet-implementation-using-keras/

-[47] K.Simonyan, A.Zisserman « VGG16 – Implementation Using Keras" – 2018: https://engmrk.com/vgg16-implementation-using-keras/

-[48] D.Boukhlouf " Resolution de problémes par ecosystemes : Application au traitement d'images. » Université de biskra -2005

-[49] New frontiers in embryoselection - Scientific Figure on ResearchGate. Availablefrom: https://www.researchgate.net/figure/Depiction-of-the-various-layers-of-2-models-of-artificial-intelligence-A-represents-a_fig3_366945597