جامعة مولاي إسماعيل
UNIVERSITÉ MOULAY ISMAÏL

كلية العلوم والتقنيات
FACULTÉ DES SCIENCES ET TECHNIQUES

Filière : Cycle d'Ingénieurs en Génie Informatique
Module : Intelligence artificielle

Rapport de projet intitulé :

Système de Reconnaissance Faciale

Réalisé par :

MAHROUCH Mohamed

Encadré par :

Prof. Imad ZEROUAL

Présenté le : 21/06/2025

Année Universitaire : 2024-2025

Dédicaces

Toutes les lettres ne sauraient trouver les mots qu'il faut. Tous les mots ne sauraient exprimer la gratitude, l'amour, le respect, la reconnaissance. Avec un énorme plaisir, un cœur ouvert et une immense joie, que nous dédions ce travail :

À nos très chers, respectueux et magnifiques parents, qui nous ont offert sans condition leur soutien tout au long de notre vie.

À notre frère et notre sœur pour leur soutien aux moments difficiles de notre travail.

À tous nos chers amis, pour leurs encouragements permanents, et leur soutien Moral,

À tous nos enseignants, votre générosité et votre soutien nous obligent à vous témoigner notre profond respect et notre loyale considération

À toutes les personnes qui nous ont aidés ou encouragés tout au long de nos études, nos professeurs et nos encadrants.

À tous ceux qui nous sont chers.

Remerciements

En tout premier lieu, nous remercions le bon Dieu, tout puissant, de nous avoir donné la force pour survivre, ainsi que l'audace pour
Dépasser toutes les difficultés. Au nom du dieu le clément et le miséricordieux louange à ALLAH le tout puissant.

Nous tenons à remercier fortement la professeur IMAD ZEROUAL pour sa disponibilité et ses conseils qui nous ont permis toujours de poser
de nouvelles questions et ainsi pour son suivi et pour son énorme soutien,
qu'il n'a cessé de nous prodiguer tout au long de la période d'étude.
Nous vous remercions d'avoir partagé avec nous votre passion pour l'enseignement. Nous avons grandement apprécié votre soutien, votre implication et votre expérience.
Merci Infiniment !

Résumé

Ce rapport détaille le développement d'un système de reconnaissance faciale intégré à une maison intelligente, dont l'objectif est de permettre un accès sécurisé et automatisé aux résidents autorisés. Le projet s'appuie sur des technologies d'apprentissage profond, notamment les réseaux de neurones convolutifs (CNN), et combine plusieurs volets essentiels : la collecte d'un jeu de données personnalisé, la détection faciale en temps réel, l'entraînement d'un modèle de classification, ainsi que le déploiement d'une interface web permettant une interaction fluide avec le système.

L'ensemble du système repose sur la capture d'images via webcam, la détection des visages à l'aide d'un classificateur Haar, et l'organisation des données par individu. Ces images sont ensuite redimensionnées, normalisées, puis utilisées pour entraîner un modèle CNN capable de distinguer les visages autorisés. offrant une reconnaissance faciale en temps réel directement à partir du flux vidéo de la caméra.

Des techniques de régularisation ont été utilisées pour éviter le sur-apprentissage, et des pistes d'amélioration sont envisagées, telles que l'augmentation du dataset ou l'optimisation du modèle pour l'embarqué. Ce système constitue une solution innovante, sécurisée et évolutive pour la gestion d'accès dans un contexte domestique.

Table de matière

Table des matières

Introduction Générale

À l'ère de l'Internet des objets (IoT) et de l'intelligence artificielle, les maisons intelligentes deviennent une réalité de plus en plus accessible. Ces habitats connectés ont pour ambition de faciliter le quotidien des utilisateurs, d'optimiser la consommation énergétique, mais surtout d'améliorer la sécurité. Parmi les technologies émergentes les plus prometteuses dans ce domaine, la reconnaissance faciale se distingue comme un moyen efficace, naturel et sécurisé d'authentification.

Ce projet s'inscrit dans cette dynamique, avec pour objectif la conception et le développement d'un système de reconnaissance faciale intégré à une maison intelligente. Il vise à restreindre l'accès au domicile aux seules personnes autorisées, sans recours à des moyens traditionnels tels que les clés ou les codes d'accès. Cette approche permet de renforcer la sécurité tout en offrant une expérience utilisateur plus fluide et intuitive.

Le projet repose sur l'utilisation de techniques d'apprentissage profond, notamment les réseaux de neurones convolutifs (CNN), pour détecter et reconnaître les visages en temps réel. Il couvre l'ensemble du cycle de développement d'un système intelligent : acquisition de données personnalisées, entraînement d'un modèle de classification, et test en conditions réelles.

Ce document présente en détail les étapes de mise en œuvre, les choix technologiques, les résultats obtenus, ainsi que les perspectives d'amélioration de ce système de reconnaissance faciale appliqué à la sécurité domestique.

Chapitre 1 : Enivrenement De Travail et Outils Utilisée

1. *Technologies utilisées*

1.1. Environnements de travail

Figure 3:Logo de Google Colaboratory

Google Colab (Colaboratory) est un service gratuit proposé par Google qui permet d'écrire et d'exécuter du code Python directement dans votre navigateur. Conçu principalement pour le développement et l'expérimentation de modèles d'apprentissage automatique, Colab est un outil puissant pour les chercheurs, les ingénieurs de données, et les développeurs

1.2 Outils Utilisés :

1.2.1 Python :

Python est un langage de programmation interprété, orienté objet et de haut niveau, caractérisé par une sémantique dynamique. Ses structures de données intégrées de qualité supérieure, associées à un typage dynamique et à une liaison dynamique, en font un choix extrêmement attrayant pour le développement rapide d'applications, ainsi que pour son utilisation en tant que langage de script ou de liaison pour connecter des composants préexistants. La syntaxe simple et aisément compréhensible de Python met l'accent sur la
Lisibilité, réduisant ainsi les coûts de maintenance du code.

Python prend en charge les modules et les packages, favorisant la modularité et la Réutilisation du code. L'interpréteur Python, ainsi que la vaste bibliothèque standard, sont disponibles gratuitement sous forme de code source ou binaire pour toutes les principales plateformes et peuvent être largement diffusés.

Figure 5 :Python

1.2.2 OpenCV

OpenCV, acronyme de Open Source Computer Vision Library, est une bibliothèque open source spécialisée dans la vision par ordinateur et le traitement d'images. Elle est conçue pour aider les développeurs à créer des applications de vision par ordinateur performantes en fournissant un ensemble d'outils et de fonctions pour le traitement, la manipulation et l'analyse d'images et de vidéos.

OpenCV offre un large éventail de fonctionnalités, notamment la détection d'objets, la reconnaissance faciale, la correspondance de formes, le suivi d'objets, la stéréovision, le calibrage de caméra, la segmentation d'images, et bien plus encore. Elle prend en charge de nombreux langages de programmation, notamment C++, Python, et Java, tout en proposant des liaisons pour d'autres langages.

Figure 6 :opencv

L'un des avantages majeurs d'OpenCV est sa portabilité, ce qui signifie qu'elle peut être utilisée sur différentes plateformes telles que Windows, Linux, macOS, Android et iOS. Cette polyvalence en fait un choix populaire pour une variété d'applications, allant de la robotique à la vision par ordinateur embarquée.

OpenCV est largement utilisée dans l'industrie, la recherche universitaire et la communauté open source en raison de sa fiabilité, de ses performances et de sa facilité d'utilisation. Elle est constamment mise à jour et améliorée pour suivre les dernières avancées en vision par ordinateur, ce qui en fait une ressource inestimable pour les acteurs de ce domaine passionnant.

1.2.3 TensorF low

TensorFlow est une bibliothèque open source développée par Google, largement utilisée pour le développement de modèles d'apprentissage automatique et de réseaux de neurones. Elle offre un ensemble complet d'outils et de ressources pour la création, la formation et le déploiement de modèles d'IA dans une grande variété d'applications.

TensorFlow Lite, quant à lui, est une version allégée de TensorFlow spécialement conçue pour les appareils mobiles, les systèmes embarqués et les environnements à ressources limitées.
L'objectif principal de TensorFlow Lite est de permettre l'exécution de modèles d'apprentissage automatique sur des appareils dotés de capacités de calcul limitées, tels que les smartphones, les tablettes, les microcontrôleurs et les systèmes embarqués.

Figure 7 :TensorFlow

1.2.4 Keras :

Keras est une bibliothèque de logiciels open source qui fournit une interface Python pour les réseaux de neurones artificiels. Elle agit comme une interface pour la bibliothèque TensorFlow.

Keras contient de nombreuses implémentations de blocs de construction de réseaux neuronaux couramment utilisés tels que des couches, des objectifs, des fonctions d'activation, des optimiseurs et une multitude d'outils pour faciliter le travail avec des données d'image et de texte afin de simplifier le codage nécessaire à l'écriture de code de réseau neuronal profond.

2. *Machine Learning*

2.1. Définition et utilité

Le machine learning, ou apprentissage automatique en français, est une sous-discipline de l'intelligence artificielle (IA) qui se concentre sur le développement d'algorithmes et de modèles permettant à un système de "apprendre" à partir de données. Contrairement aux approches traditionnelles de la programmation, où les règles et les décisions sont explicitement codées par les développeurs, le machine learning permet aux ordinateurs d'améliorer leurs performances sur des tâches spécifiques en détectant des motifs dans des ensembles de données et en ajustant automatiquement leurs modèles en conséquence.

Les modèles de machine learning sont généralement classés en trois catégories principales :

Apprentissage supervisé : Le modèle est entraîné sur un ensemble de données étiquetées, où chaque exemple d'entraînement est associé à une réponse correcte. Le but est d'apprendre une fonction qui relie les entrées aux sorties afin de prédire la sortie pour de nouvelles entrées.

Apprentissage non supervisé : Le modèle travaille sur des données non étiquetées et cherche à découvrir des structures ou des motifs cachés dans les données, comme le regroupement (clustering) ou la réduction de dimensionnalité.

Apprentissage par renforcement : Le modèle apprend en interagissant avec un environnement et en recevant des récompenses ou des punitions en fonction de ses actions. Il cherche à maximiser sa récompense globale en apprenant une politique optimale d'actions.

2.2. La classification

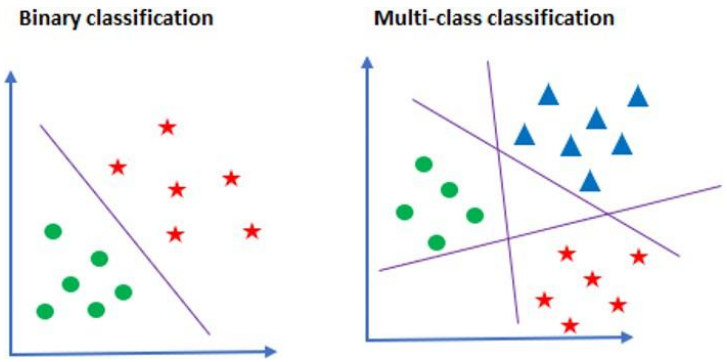


Figure 9 – Graphe de Classification binaire et Multiclasse

La classification est une tâche fondamentale en machine learning, où l'objectif est d'attribuer une étiquette ou une catégorie à une entrée donnée parmi un ensemble de catégories possibles. Cette tâche est considérée comme un problème d'apprentissage

supervisé, car le modèle est entraîné sur un ensemble de données étiquetées, où chaque exemple est associé à une catégorie précise.

Le processus de classification consiste généralement à :

Apprendre à partir de données d'entraînement : Un algorithme de classification est fourni avec des données étiquetées qui contiennent des exemples d'entrées et leurs catégories associées. L'algorithme utilise ces exemples pour apprendre les relations ou les motifs qui permettent de distinguer les différentes catégories.

Prédire les étiquettes pour de nouvelles données : Une fois le modèle entraîné, il est capable de prédire la catégorie d'une nouvelle entrée en fonction des motifs appris. Ces prédictions peuvent être utilisées pour diverses applications, telles que la reconnaissance d'objets, la détection de spam, ou la classification de textes.

Les algorithmes courants de classification incluent les machines à vecteurs de support (SVM), les arbres de décision, les réseaux de neurones, et les méthodes bayésiennes, chacun ayant ses avantages en fonction des caractéristiques des données.

1. *Deep Learning*

1.1. Définition du Deep Learning

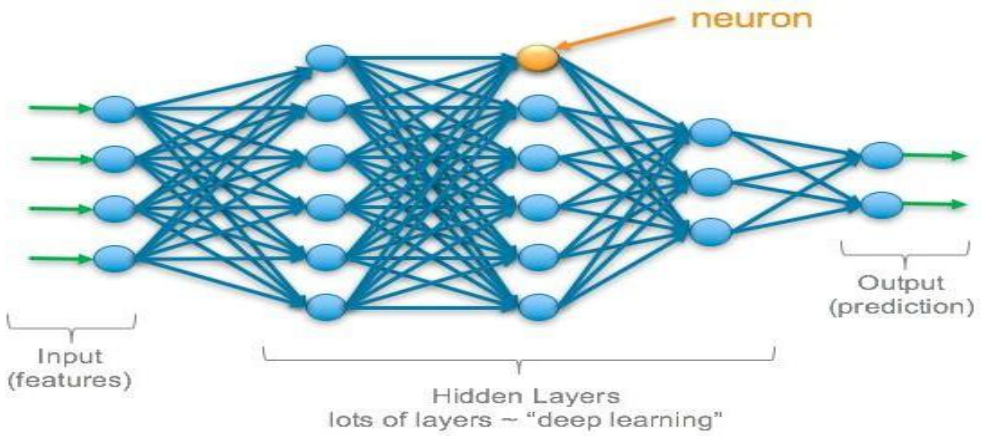


Figure 13 : Architecture typique d'un réseau de neurones profond

Le Deep Learning est une branche de l'intelligence artificielle (IA) qui se concentre sur l'apprentissage à partir de grandes quantités de données en utilisant des réseaux de neurones artificiels, appelés réseaux de neurones profonds. Il s'inspire vaguement de la structure et du fonctionnement du cerveau humain, où les neurones sont organisés en couches et traitent l'information de manière hiérarchique.

En deep learning, les réseaux de neurones sont composés de plusieurs couches intermédiaires, aussi appelées couches cachées, qui permettent au modèle d'apprendre des représentations complexes des données. Chaque couche transforme les données d'entrée de manière non linéaire pour extraire des caractéristiques plus abstraites à chaque niveau. Ce processus est particulièrement efficace pour des tâches telles que la classification d'images, la reconnaissance vocale, la traduction automatique, ou encore la

détection d'objets.

L'un des aspects clés du deep learning est sa capacité à apprendre directement à partir des données brutes, en évitant la nécessité de définir manuellement des caractéristiques spécifiques, ce qui le distingue des approches plus traditionnelles de l'apprentissage automatique. Les modèles de deep learning, comme les réseaux convolutifs (CNN) pour la vision par ordinateur ou les réseaux récurrents (RNN) pour le traitement du langage, sont particulièrement puissants grâce à leur capacité à traiter de vastes quantités de données et à exploiter des architectures complexes pour optimiser leurs performances.

2.2 CNN

2.2.1 Définition et utilisation

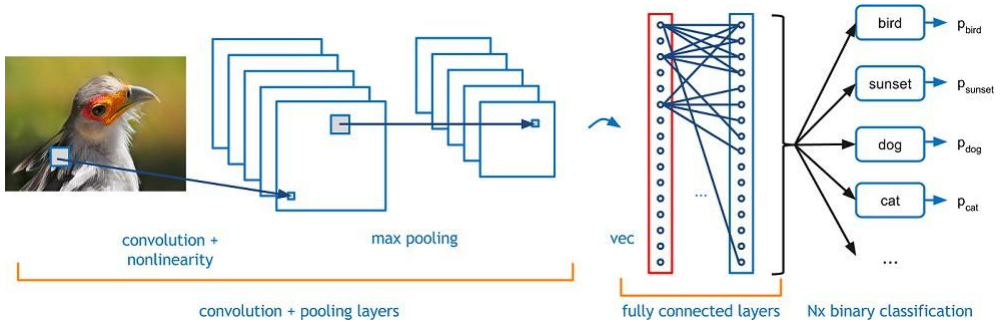


Figure 14 : Extraction des caractéristiques dans un réseau CNN

L'architecture notre projet est basée largement sur CNN. CNN(Convolutional Neural networks) sont une classe de réseaux de neurones profonds particulièrement efficaces pour le traitement des données ayant une structure de grille, comme les images. Introduits dans les années 1980 et popularisés par des avancées récentes en apprentissage profond, les CNN exploitent des mécanismes de convolution pour extraire automatiquement des caractéristiques à différents niveaux de complexité.

Caractéristiques des CNN :

Convolution : Les CNN utilisent des filtres ou des noyaux de convolution pour détecter des motifs locaux dans les images, comme des bords, des textures, et des formes. Ces filtres glissent sur l'image pour produire une carte de caractéristiques qui représente les informations extraites.

Pooling : Après la convolution, les CNN utilisent des opérations de pooling (par exemple, max pooling) pour réduire la taille des cartes de caractéristiques tout en conservant les informations essentielles. Cela aide à réduire la complexité computationnelle et à rendre le modèle plus robuste aux variations des images.

Architecture Hiérarchique : Les CNN sont composés de plusieurs couches convolutionnelles suivies de couches de pooling, puis de couches entièrement connectées pour effectuer des tâches de classification ou de régression. Chaque couche successive extrait des caractéristiques de plus en plus abstraites.

2.2.2 Architecture de CNN

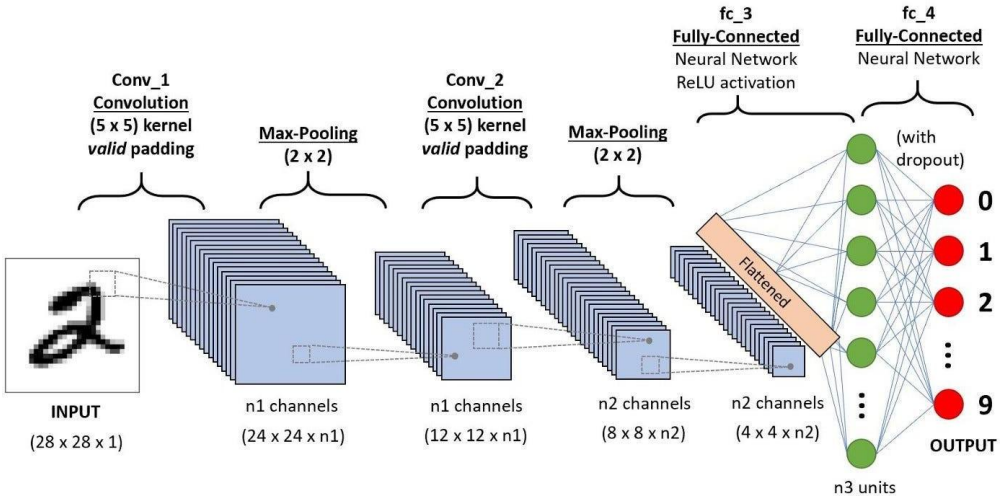


Figure 15 :Architecture du réseau CNN

Les réseaux de neurones convolutionnelles sont conçus pour traiter des données ayant une structure de grille, comme les images. Leur architecture repose sur une série de couches spécialisées qui permettent d'extraire des caractéristiques hiérarchiques à partir des donnéesd'entrée. Voici une description détaillée des principaux composants de l'architecture des CNN
:

1. *Couche de Convolution*

- **Fonctionnement :** La couche de convolution applique des filtres (ou noyaux) de petite taille à l'image d'entrée pour créer des cartes de caractéristiques (feature maps). Chaque filtre détecte un motif spécifique, comme des bords ou des textures. Le processus de convolution consiste à faire glisser le filtre sur l'image d'entrée et à calculer le produit scalaire entre le filtre et la région de l'image qu'il couvre.
- *Paramètres :*
    - **Taille du Filtre :** Les filtres ont généralement une petite taille, par exemple, 3x3 ou 5x5 pixels.
    - **Stride :** Le pas de déplacement du filtre sur l'image. Un stride de 1 signifie que le filtre se déplace d'un pixel à la fois.
    - **Padding :** Ajout de pixels autour des bordures de l'image pour conserver les dimensions de l'image après la convolution.
- **Objectif :** Extraire des caractéristiques locales de l'image, comme les bords, les coins, et les textures.

2. *Couche de Pooling*

- **Fonctionnement :** La couche de pooling (ou sous-échantillonnage) réduit la taille des cartes de caractéristiques tout en conservant les informations essentielles. Cela permet de diminuer la complexité computationnelle et de rendre le modèle plus invariant aux petites variations dans les images.
- *Types de Pooling :*

- **Max Pooling :** Prend la valeur maximale dans une région spécifiée. Par exemple, un pooling 2x2 sélectionne la valeur maximale parmi chaque groupe de 4 pixels (2x2).
- **Average Pooling :** Calcule la moyenne des valeurs dans une région spécifiée.

- **Objectif :** Réduire la dimensionnalité et extraire les caractéristiques les plus importantes tout en réduisant le risque de sur apprentissage.

3. *Couche de Normalisation*

- **Fonctionnement :** La normalisation par lot (Batch Normalization) normalise les sorties des couches de convolution en utilisant la moyenne et l'écart-type de chaque mini-lot d'entrées. Cela aide à stabiliser l'entraînement et à accélérer la convergence.
- **Objectif :** Améliorer la stabilité de l'apprentissage en réduisant le décalage de distribution interne et en accélérant l'entraînement.

4. *Couche d'Activation*

- **Fonctionnement :** Les couches d'activation introduisent la non-linéarité dans le modèle. La fonction d'activation la plus couramment utilisée est la ReLU (Rectified Linear Unit), qui remplace les valeurs négatives par zéro et laisse les valeurs positives inchangées.
- **Objectif :** Introduire des non-linéarités dans le réseau, permettant au modèle d'apprendre des relations complexes.

5. *Couche Complètement Connectée (Fully Connected Layer)*

- **Fonctionnement :** Les couches entièrement connectées (ou dense) suivent généralement les couches de convolution et de pooling. Chaque neurone dans une couche entièrement connectée est connecté à tous les neurones de la couche précédente. Ces couches sont utilisées pour effectuer des tâches de classification ou de régression.
- **Objectif :** Intégrer les caractéristiques extraites par les couches précédentes pour effectuer la prédiction finale.

6. *Couche de Sortie*

- **Fonctionnement :** La couche de sortie est généralement une couche entièrement connectée avec une fonction d'activation adaptée à la tâche spécifique, comme Softmax pour la classification multi-classes ou Sigmoid pour la classification binaire.
- **Objectif :** Produire les résultats finaux du modèle, comme les probabilités de classes pour la classification ou les coordonnées des boîtes pour la détection d'objets.

7. *Architecture Hiérarchique*

- **Principe :** Les CNN utilisent une architecture hiérarchique où chaque couche successive apprend des caractéristiques de plus en plus abstraites. Les

premières couches capturent des détails simples (bords, textures), tandis que les couches plus profondes détectent des objets complexes (formes, structures).

- **Objectif :** Permettre au modèle de capturer des motifs à différentes échelles et niveaux de complexité, ce qui est essentiel pour des tâches comme la reconnaissance d'objets et la segmentation.

Exemple d'Architecture CNN

Un exemple classique d'architecture CNN est LeNet-5, qui se compose de plusieurs couches de convolution, de pooling, et de couches entièrement connectées. Plus récemment, des architectures comme AlexNet, VGG, ResNet, et DenseNet ont été développées pour améliorer la performance des CNN dans des tâches de vision par ordinateur.

2.3. Fonction d'activation

2.3.1. Définition

Une fonction d'activation est un composant clé des réseaux de neurones artificiels, qui détermine si un neurone doit être activé ou non en fonction de la somme pondérée des entrées reçues par ce neurone. Elle introduit de la non-linéarité dans le réseau, permettant ainsi au modèle de résoudre des problèmes complexes et de capturer des relations non linéaires entre les données d'entrée et de sortie.

Sans fonction d'activation, un réseau de neurones se comporterait comme une combinaison linéaire simple des entrées, ce qui limiterait sa capacité à modéliser des phénomènes complexes. En introduisant cette non-linéarité, les fonctions d'activation permettent au réseau de mieux comprendre et représenter les structures complexes des données.

Quelques exemples courants de fonctions d'activation sont :

Sigmoïde : Transforme la sortie dans une plage comprise entre 0 et 1. Utile pour des tâches de classification binaire.

- **ReLU (Rectified Linear Unit) :** Active le neurone en renvoyant la valeur d'entrée si elle est positive, sinon elle renvoie zéro. Très populaire en raison de ses performances dans les réseaux profonds.
- **Tanh :** Transforme la sortie entre -1 et 1, ce qui peut être utile pour des situations où les données sont centrées autour de zéro.
- **Softmax :** Utilisée principalement dans la classification multi-classes, elle normalise les sorties pour représenter des probabilités.

Chaque fonction d'activation a ses avantages et ses inconvénients, et le choix de la fonction d'activation peut avoir un impact significatif sur les performances d'un modèle de deep learning.

2.3.2. Fonctions d'activation du réseau CNN

Dans la plupart des architectures typiques d'un réseau CNN, on voit l'emploi des fonctions d'activation suivantes :

1. *ReLU (Rectified Linear Unit)*

- **Description** : La fonction ReLU est l'une des plus populaires dans les réseaux de neurones, en particulier dans les réseaux convolutifs (CNN). Elle est définie comme étant une fonction linéaire par morceaux qui renvoie la valeur d'entrée si elle est positive, et zéro si elle est négative.

- **Formule** : $f(x)=\max(0,x)$
  - *Caractéristiques :*
    - Simplicité de calcul : ReLU est très simple et rapide à calculer.

- Activation parcimonieuse : Elle produit des activations éparses, c'est-à-dire que seule une partie des neurones est activée, ce qui améliore l'efficacité du réseau.
- Non-linéarité : Malgré sa simplicité, elle introduit une non-linéarité essentielle pour modéliser des relations complexes.

- **Utilité** : ReLU permet de résoudre le problème du "vanishing gradient" souvent rencontré avec des fonctions comme Sigmoid et Tanh, ce qui facilite l'entraînement de réseaux profonds. Cependant, elle peut souffrir du problème des "neurones morts", où certains neurones ne s'activent plus, ce qui peut être corrigé avec des variantes comme Leaky ReLU.

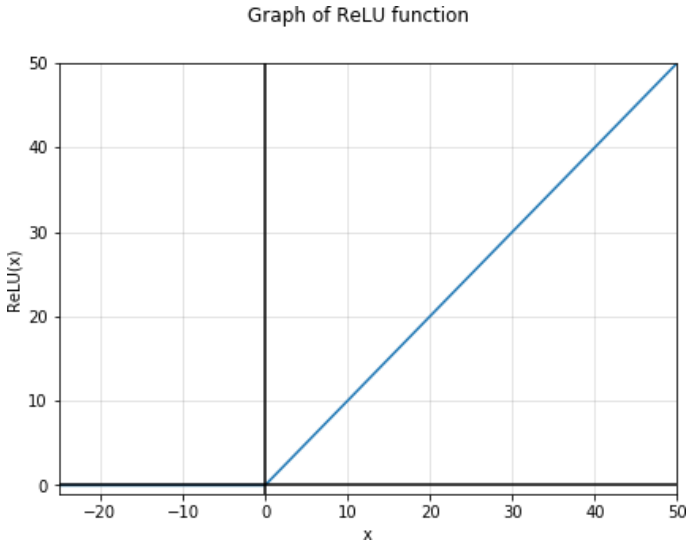


Figure 16 :fonction d'activation Relu

2. *Softmax*

- **Description** : Softmax est une fonction d'activation utilisée principalement dans les couches de sortie des réseaux de neurones, en particulier pour les tâches de **classification multi-classes**. Elle convertit les sorties d'un réseau de neurones en probabilités, où la somme de toutes les sorties est égale à 1.

Formule : $\text{softmax}(x_i)=\exp(x_i)/(\sum_j \exp(x_j))$ Où x_i est l'entrée pour la classe i, et la somme est effectuée sur toutes les classes.

- *Caractéristiques :*
  - Génère une distribution de probabilités : Les sorties de Softmax sont comprises entre 0 et 1, représentant des probabilités.
  - Normalisation : Elle s'assure que la somme des probabilités pour toutes les classes est égale à 1, ce qui permet d'interpréter la sortie du modèle comme une classification probabiliste.

- **Utilité** : Softmax est couramment utilisée dans les tâches de classification multi-classes, par exemple dans les réseaux convolutifs pour des problèmes de classification d'images où l'on cherche à attribuer une image à l'une des classes possibles. La classe avec la probabilité la plus élevée est choisie comme prédiction.

3. *Sigmoid*

- **Description** : La fonction Sigmoid, également appelée fonction logistique, est une fonction d'activation qui prend une entrée quelconque et la transforme en une sortie comprise entre 0 et 1. Elle est souvent utilisée dans les réseaux de neurones pour les problèmes de **classification binaire**.
- **Formule** : $\sigma(x)=1/(1+\exp(-x))$
- *Caractéristiques :*
    - Sortie bornée entre 0 et 1 : Sigmoid produit des valeurs qui peuvent être interprétées comme des probabilités.
    - Lissage : Elle est continue et dérivable, ce qui en fait une bonne option pour des sorties probabilistes.
    - Saturation : Dans les régions où xxx est très grand ou très petit, les gradients deviennent très faibles, ce qui peut ralentir l'entraînement des réseaux de neurones en profondeur (problème du "vanishing gradient").
- **Utilité** : Sigmoid est couramment utilisée dans les **réseaux de neurones multicouches** et est particulièrement adaptée pour les tâches de classification binaire, où la sortie est interprétée comme une probabilité. Elle est aussi parfois utilisée dans des couches internes de modèles de classification plus complexes.

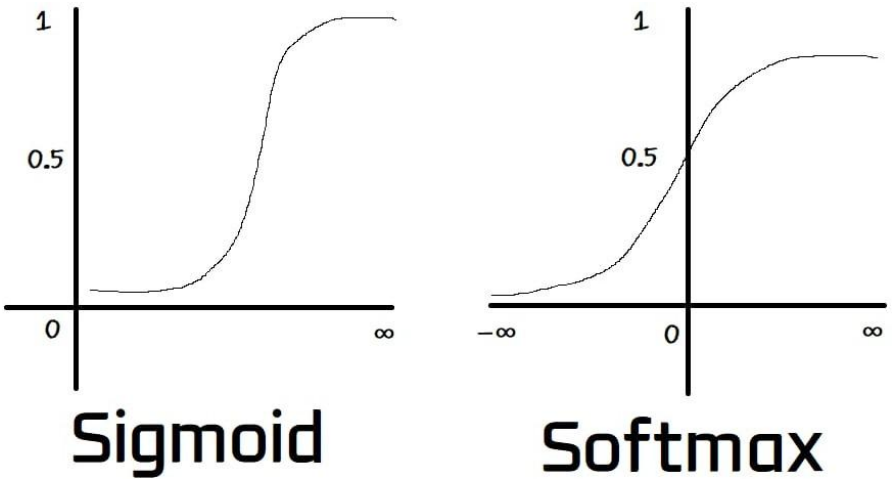
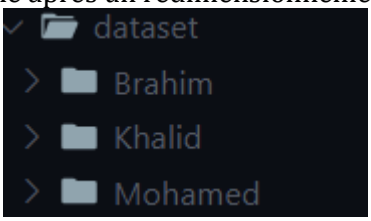


Figure 17 :fonctions d'activations Sigmoid et Softmax

Chapitre 2: Préparation des données

1. Collecte des images avec OpenCV

Dans un premier temps, un script en Python utilisant OpenCV a été développé pour permettre la collecte des images de visages à l'aide d'une webcam. L'utilisateur est invité à saisir son nom, ce qui permet de stocker automatiquement les images capturées dans un dossier portant son nom. Le système détecte les visages en temps réel et enregistre 1000 images par personne après un redimensionnement à une taille uniforme.



2. Chargement des images et étiquetage

Une fois les images collectées, elles sont chargées depuis une structure de dossiers où chaque sous-dossier représente une personne. Chaque image est convertie en niveaux de gris et redimensionnée à une taille fixe afin de standardiser les données. Les images sont stockées dans une liste `X` et leurs étiquettes correspondantes (les noms des personnes) dans une liste `y`.

3. Augmentation des données

Pour enrichir le dataset et améliorer la robustesse du modèle, une étape d'augmentation des données est effectuée. Elle consiste à générer plusieurs variantes de chaque image en appliquant des transformations telles que la rotation, la translation et le retournement horizontal. Chaque image originale génère trois nouvelles images augmentées, ce qui permet d'accroître la diversité du jeu de données.

```python
def augment_image(img):
    rows, cols = img.shape

    # Rotation aléatoire entre -15 et 15 degrés
    angle = random.uniform(-15, 15)
    M_rot = cv2.getRotationMatrix2D((cols/2, rows/2), angle, 1)

    # Translation aléatoire
    tx = random.uniform(-0.1*cols, 0.1*cols)
    ty = random.uniform(-0.1*rows, 0.1*rows)
    M_trans = np.float32([[1, 0, tx], [0, 1, ty]])

    # Appliquer rotation puis translation
    rotated = cv2.warpAffine(img, M_rot, (cols, rows))
    translated = cv2.warpAffine(rotated, M_trans, (cols, rows))

    # Flip horizontal aléatoire
    if random.choice([True, False]):
        translated = cv2.flip(translated, 1)

    return translated
```

4. Visualisation des données

Un outil de visualisation a été mis en place pour afficher un échantillon des images avec leurs étiquettes correspondantes. Cela permet de s'assurer de la qualité des images collectées et de leur bonne répartition par classe (personne).

5. Prétraitement final et encodage des étiquettes

Avant de passer à l'entraînement du modèle, les images sont normalisées (valeurs des pixels entre 0 et 1) et remodelées pour convenir à l'entrée du réseau de neurones convolutif. Les étiquettes textuelles sont ensuite

converties en entiers grâce à un encodeur, ce qui facilitera leur traitement par le modèle d'apprentissage automatique.

Chapitre 3: Entraînement du modèle

3.1 Préparation des données

Avant d'entraîner le modèle, les données ont été préparées. Les étiquettes (labels) ont été encodées en one-hot pour permettre une classification multi-classes. Les images ont également été normalisées et converties dans le format requis par le réseau neuronal.

3.2 Architecture du modèle

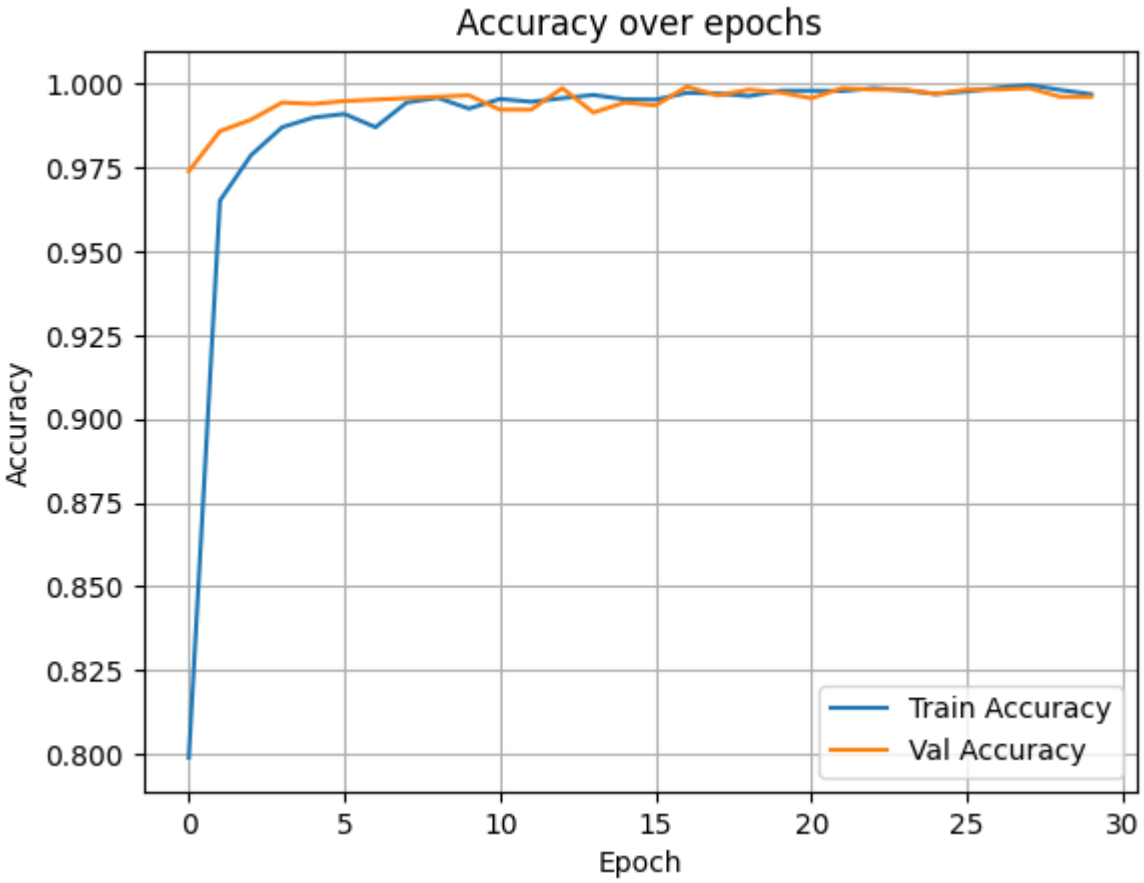
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 198, 198, 32)	320
max_pooling2d (MaxPooling2D)	(None, 99, 99, 32)	0
conv2d_1 (Conv2D)	(None, 97, 97, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 48, 48, 64)	0
conv2d_2 (Conv2D)	(None, 46, 46, 128)	73,856
max_pooling2d_2 (MaxPooling2D)	(None, 23, 23, 128)	0
flatten (Flatten)	(None, 67712)	0
dense (Dense)	(None, 128)	8,667,264
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 3)	387

3.3 Compilation et entraînement

Le modèle a été compilé avec l'optimiseur Adam et la fonction de perte categorical_crossentropy. La métrique utilisée pour l'évaluation pendant l'entraînement est l'accuracy. L'entraînement a été effectué sur 30 époques, avec une validation à chaque époque

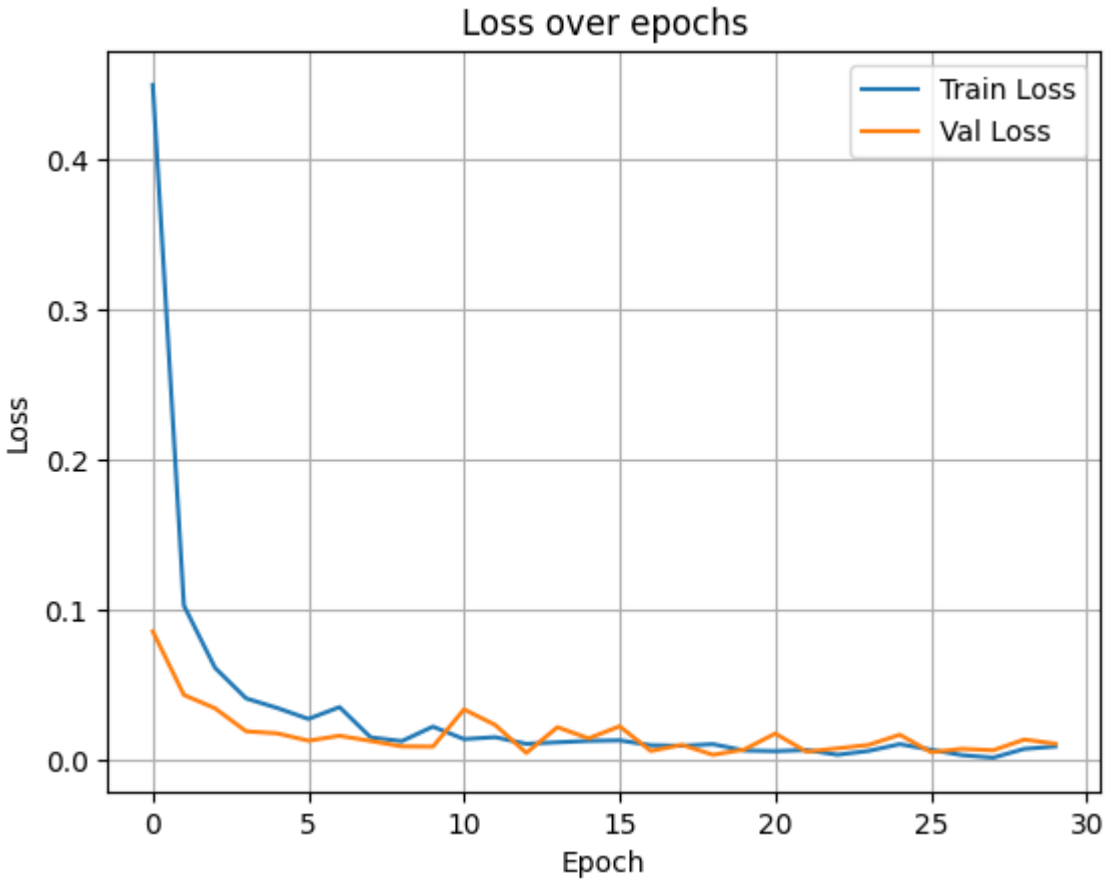
Les résultats :

```
Epoch 1/30
293/293 ━━━━━━━━━━━━━━━━━━━━ 190s 629ms/step - accuracy: 0.6248 - loss: 0.7605 - val_accuracy: 0.9739 - val_loss: 0.0855
Epoch 2/30
293/293 ━━━━━━━━━━━━━━━━━━━━ 150s 513ms/step - accuracy: 0.9570 - loss: 0.1293 - val_accuracy: 0.9859 - val_loss: 0.0431
Epoch 3/30
293/293 ━━━━━━━━━━━━━━━━━━━━ 147s 501ms/step - accuracy: 0.9815 - loss: 0.0552 - val_accuracy: 0.9893 - val_loss: 0.0342
Epoch 4/30
293/293 ━━━━━━━━━━━━━━━━━━━━ 146s 497ms/step - accuracy: 0.9869 - loss: 0.0449 - val_accuracy: 0.9944 - val_loss: 0.0189
Epoch 5/30
293/293 ━━━━━━━━━━━━━━━━━━━━ 146s 497ms/step - accuracy: 0.9911 - loss: 0.0291 - val_accuracy: 0.9940 - val_loss: 0.0175
Epoch 6/30
293/293 ━━━━━━━━━━━━━━━━━━━━ 151s 514ms/step - accuracy: 0.9905 - loss: 0.0306 - val_accuracy: 0.9949 - val_loss: 0.0126
Epoch 7/30
293/293 ━━━━━━━━━━━━━━━━━━━━ 145s 494ms/step - accuracy: 0.9839 - loss: 0.0434 - val_accuracy: 0.9953 - val_loss: 0.0159
Epoch 8/30
293/293 ━━━━━━━━━━━━━━━━━━━━ 145s 494ms/step - accuracy: 0.9959 - loss: 0.0135 - val_accuracy: 0.9957 - val_loss: 0.0124
Epoch 9/30
293/293 ━━━━━━━━━━━━━━━━━━━━ 145s 496ms/step - accuracy: 0.9953 - loss: 0.0123 - val_accuracy: 0.9962 - val_loss: 0.0088
Epoch 10/30
293/293 ━━━━━━━━━━━━━━━━━━━━ 150s 511ms/step - accuracy: 0.9917 - loss: 0.0230 - val_accuracy: 0.9966 - val_loss: 0.0086
Epoch 11/30
293/293 ━━━━━━━━━━━━━━━━━━━━ 149s 508ms/step - accuracy: 0.9963 - loss: 0.0116 - val_accuracy: 0.9923 - val_loss: 0.0335
Epoch 12/30
293/293 ━━━━━━━━━━━━━━━━━━━━ 151s 514ms/step - accuracy: 0.9927 - loss: 0.0188 - val_accuracy: 0.9923 - val_loss: 0.0231
Epoch 13/30
...
Epoch 29/30
293/293 ━━━━━━━━━━━━━━━━━━━━ 152s 519ms/step - accuracy: 0.9989 - loss: 0.0036 - val_accuracy: 0.9962 - val_loss: 0.0134
Epoch 30/30
293/293 ━━━━━━━━━━━━━━━━━━━━ 176s 601ms/step - accuracy: 0.9975 - loss: 0.0074 - val_accuracy: 0.9962 - val_loss: 0.0106
```



Analyse des résultats :

1. **Départ (Epoch 0-1)** :
   - La précision sur les données d'entraînement est assez faible (~80 %), ce qui est normal au début.
   - La précision sur le jeu de validation est meilleure (~97 %), ce qui peut indiquer que les premières données sont bien généralisées ou que le modèle commence rapidement à apprendre les bonnes caractéristiques.
2. **Montée rapide (Epoch 1 à 5)** :
   - La précision du modèle s'améliore très vite sur le jeu d'entraînement, atteignant plus de 98 %.
   - Celle de la validation aussi reste élevée, ce qui est bon signe.
3. **Stabilisation (après Epoch 10)** :
   - Les deux courbes sont très proches (>99 %) et quasi parallèles.
   - Cela montre que le **modèle généralise bien** : il apprend sans sur-apprendre (pas de surapprentissage visible ici).



Ce graphique reflète un entraînement **très réussi** du modèle :

1 Apprentissage rapide :

Dans les premières époques (0 à 5), la perte diminue rapidement pour l'entraînement comme pour la validation. Cela montre que le modèle, initialement avec des poids aléatoires, parvient vite à apprendre les motifs essentiels présents dans les données.

2 Bonne convergence :

À partir de la 10e à la 30e époque, les courbes de perte se stabilisent à un niveau très bas, proche de zéro. Cela indique que le modèle a atteint un point de convergence, et poursuivre l'entraînement n'apporterait que très peu de gains.

3 Pas de surapprentissage :

Un des points les plus importants : il n'y a **aucun signe de surapprentissage**. Habituellement, on le détecte si la perte sur les données d'entraînement continue de baisser tandis que celle sur la validation remonte. Ici, les deux courbes restent proches et basses, signe que le modèle **généralise bien** à des données jamais vues.

4 Bon équilibre (Good Fit) :

Le fait que la perte sur la validation soit aussi faible que celle de l'entraînement indique un **bon ajustement** : le modèle n'est ni trop simple (sous-apprentissage), ni trop complexe (surapprentissage), mais bien adapté à la tâche.

3.4 Sauvegarde du modèle

Une fois l'entraînement terminé, le modèle entraîné a été sauvegardé au format '.h5'. L'encodeur des labels a également été sauvegardé à l'aide de Joblib pour être réutilisé lors de la prédiction.

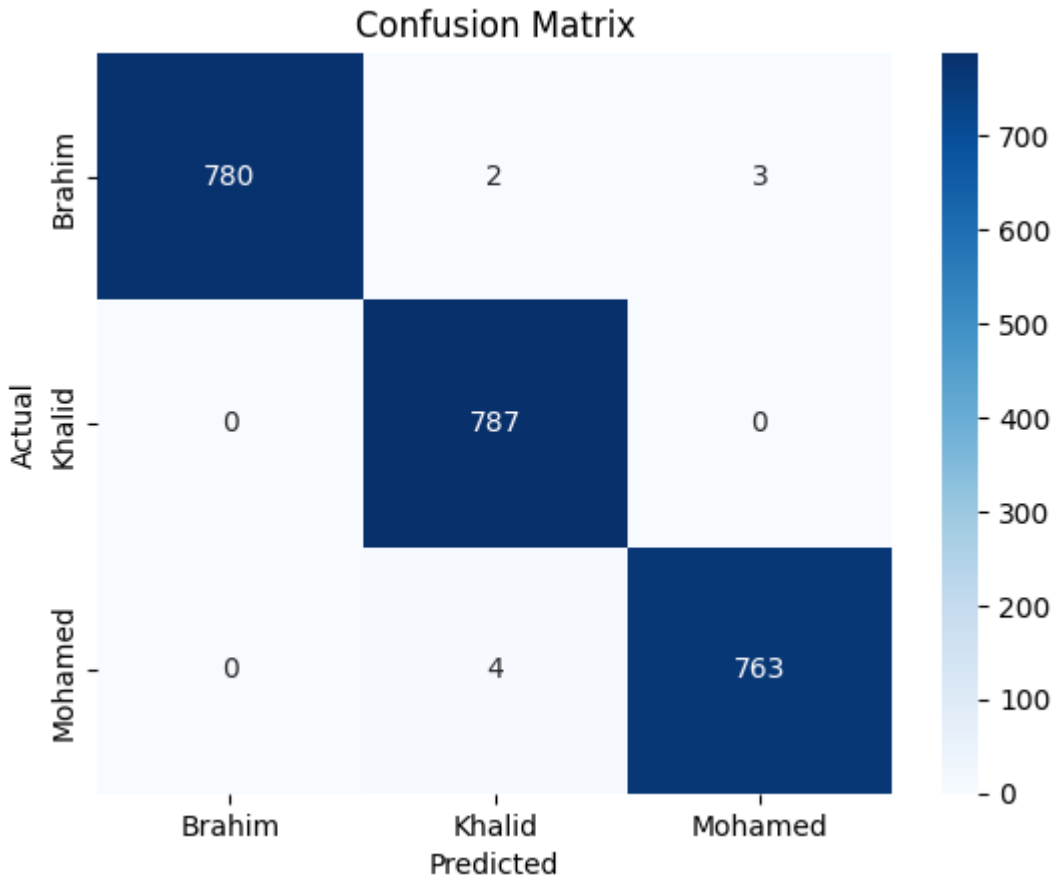
```
# Chargement du modèle et de l'encodeur
model = load_model('model_faces.h5')
le = joblib.load('label_encoder.pkl')

✓ 0.2s
```

3.5 Évaluation du modèle

Après l'entraînement, le modèle a été évalué sur un ensemble de test. Les métriques observées sont :

- Une matrice de confusion pour visualiser les performances du modèle sur chaque classe



- La précision (accuracy) sur les données de test

- Un rapport de classification avec précision, rappel et F1-score pour chaque classe

	precision	recall	f1-score	support
Brahim	1.00	0.99	1.00	785
Khalid	0.99	1.00	1.00	787
Mohamed	1.00	0.99	1.00	767
accuracy			1.00	2339
macro avg	1.00	1.00	1.00	2339
weighted avg	1.00	1.00	1.00	2339

Test accuracy :

```
Test accuracy: 99.62%
```

Chapitre 4: Application Web

1. Résumé Exécutif

Cette application web de reconnaissance faciale allie un backend performant développé en Python (Flask, TensorFlow, OpenCV) à un frontend (HTML, CSS, JavaScript). Elle permet :

- l'identification de visages en temps réel via webcam,
- l'analyse d'images statiques uploadées par l'utilisateur,
- la gestion d'un historique dynamique des reconnaissances.

2. Objectifs d'application

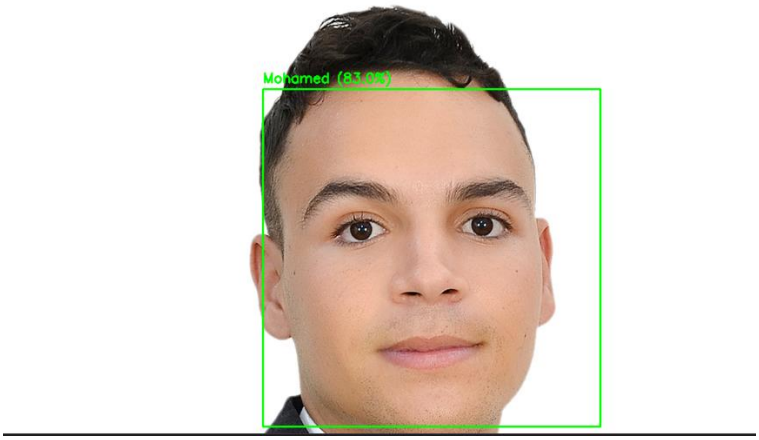
Les objectifs étaient de :

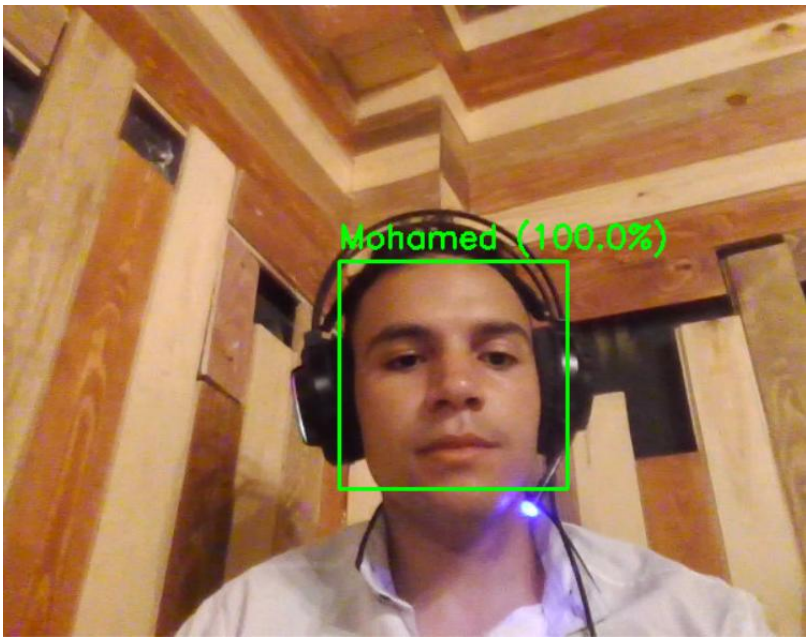
- Concevoir un système de reconnaissance faciale à l'aide d'un modèle de deep learning qui déjà entrainé
- Intégrer ce système dans une application web utilisable via navigateur.
- Permettre la reconnaissance faciale en temps réel (flux webcam).
- Analyser des images statiques soumises.
- Offrir une interface intuitive, responsive et agréable.

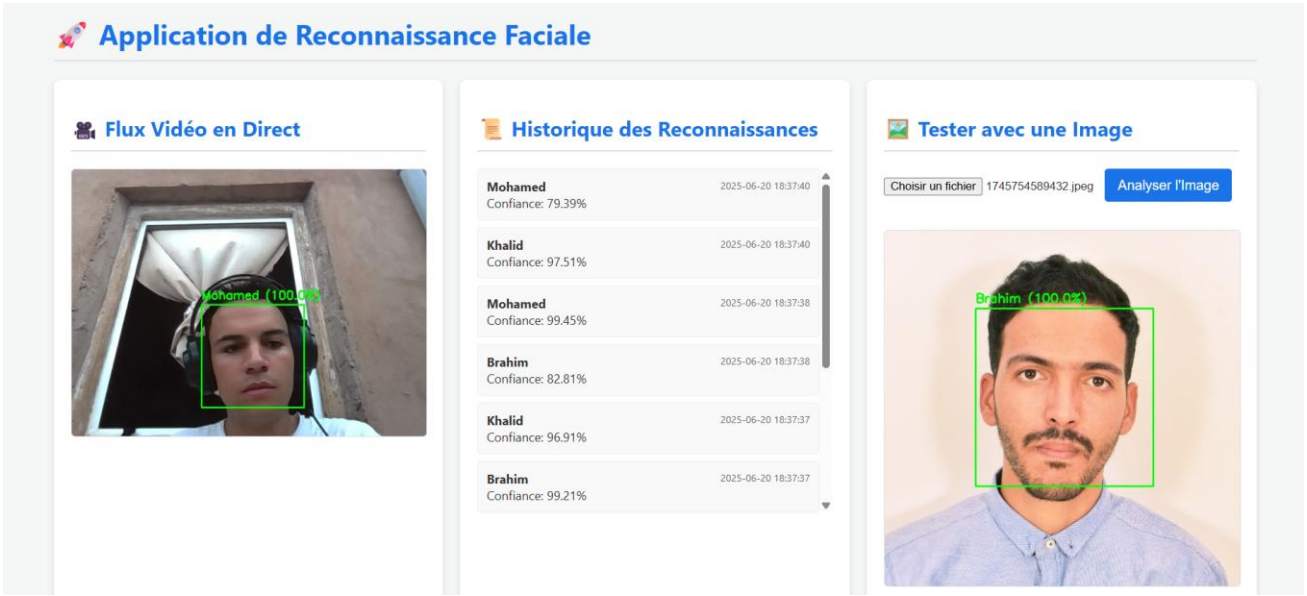
3. Fonctionnalités Clés

- 📽 **Flux Vidéo en Direct** : Affichage en temps réel du flux webcam avec identification (nom et confiance).
- 🖼 **Analyse d'Image par Upload** : Upload d'images (JPG, PNG), détection des visages, annotation.
- 📜 **Historique Dynamique** : Liste des 10 dernières reconnaissances, mise à jour automatique toutes les 3 sec.
- 📱 **Interface Responsive** : Adaptation à tous types d'écrans pour une UX optimale.

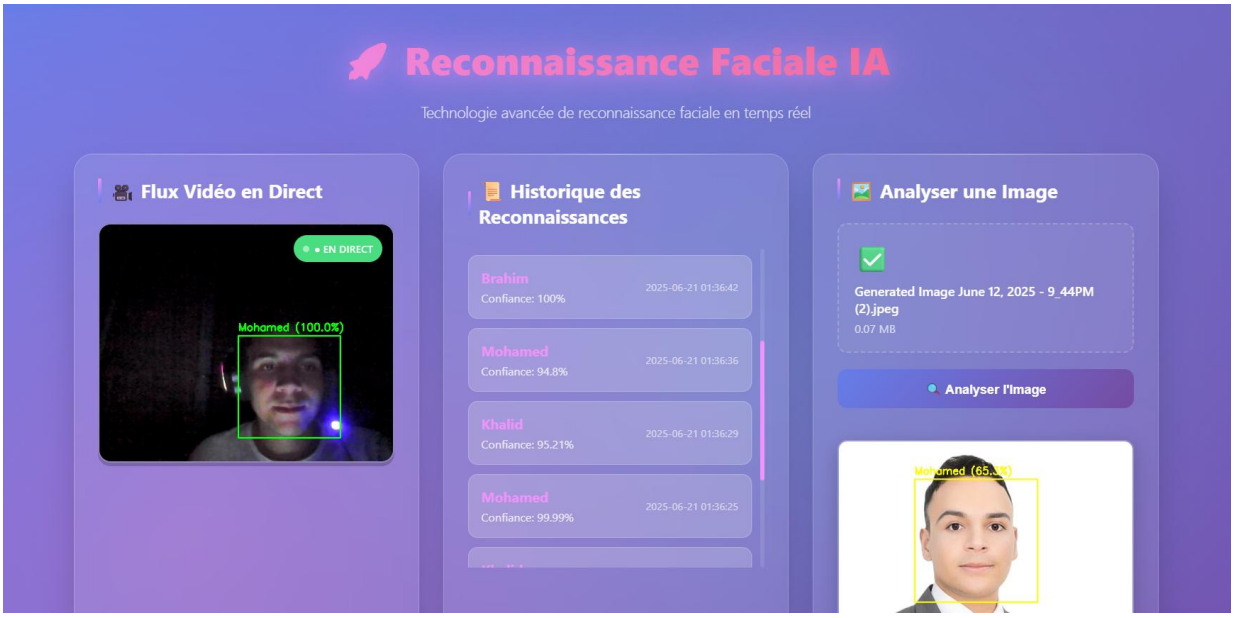
4.les tests et évaluation d'application web :




Mohamed (100.0%)


Application de Reconnaissance Faciale
Flux Vidéo en Direct
Historique des Reconnaissances
Tester avec une Image
Mohamed
Confiance: 79.39%
2025-06-20 18:37:40
Khalid
Confiance: 97.51%
2025-06-20 18:37:40
Mohamed
Confiance: 99.45%
2025-06-20 18:37:38
Brahim
Confiance: 82.81%
2025-06-20 18:37:38
Khalid
Confiance: 96.91%
2025-06-20 18:37:37
Brahim
Confiance: 99.21%
2025-06-20 18:37:37
Choisir un fichier
1745754589432.jpeg
Analyser l'Image
Brahim (100.0%)


Application de Reconnaissance Faciale
Flux Vidéo en Direct
Historique des Reconnaissances
Tester avec une Image
Mohamed
Confiance: 99.89%
2025-06-13 22:36:02
Brahim
Confiance: 99.33%
2025-06-13 22:36:00
Mohamed
Confiance: 100%
2025-06-13 22:35:52
Choisir un fichier
Capture d…0614.png
Analyser
Mahrouch mohame
Mohamed Mahrouch
Résultats :
Brahim (99.68%)
Mohamed (99.01%)


Reconnaissance Faciale IA
Technologie avancée de reconnaissance faciale en temps réel
Flux Vidéo en Direct
EN DIRECT
Mohamed (100.0%)
Historique des Reconnaissances
Brahim
Confiance: 100%
2025-06-21 01:36:42
Mohamed
Confiance: 94.8%
2025-06-21 01:36:36
Khalid
Confiance: 95.21%
2025-06-21 01:36:29
Mohamed
Confiance: 99.99%
2025-06-21 01:36:25
Analyser une Image
Generated Image June 12, 2025 - 9_44PM (2).jpeg
0.07 MB
Analyser l'Image

Conclusion

Le projet de reconnaissance faciale réalisé a permis de mettre en œuvre plusieurs techniques avancées d'intelligence artificielle et de vision par ordinateur. Grâce à l'utilisation d'un modèle de réseau de neurones convolutionnels (CNN), l'application est capable d'identifier avec précision des visages à partir d'images ou d'un flux vidéo en temps réel. Le développement a également intégré des étapes essentielles telles que la collecte et le nettoyage des données, l'entraînement et la validation du modèle, ainsi que la création d'une interface utilisateur simple et fonctionnelle.

Ce projet illustre bien le potentiel des technologies d'apprentissage profond dans des applications concrètes de sécurité et d'automatisation, comme la gestion d'accès dans une maison intelligente. Les résultats obtenus démontrent une bonne précision et une robustesse satisfaisante face aux variations d'éclairage, d'angle et d'expression faciale. Pour aller plus loin, il serait intéressant d'explorer des architectures de modèles plus sophistiquées, d'améliorer la base de données d'entraînement et d'intégrer des mécanismes de protection contre les tentatives de fraude par photos ou vidéos.

En somme, ce travail constitue une base solide pour le développement futur de systèmes de reconnaissance faciale à usage personnel ou professionnel, tout en soulignant l'importance de l'éthique et de la confidentialité dans l'utilisation de ces technologies.

Références

1. **Goodfellow, I., Bengio, Y., & Courville, A.** (2016). *Deep Learning*. MIT Press. https://www.deeplearningbook.org/
2. **Krizhevsky, A., Sutskever, I., & Hinton, G. E.** (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems (NIPS)*.
3. **Zhao, W., Chellappa, R., Phillips, P. J., & Rosenfeld, A.** (2003). Face Recognition: A Literature Survey. *ACM Computing Surveys*, 35(4), 399–458.
4. **OpenCV Documentation** — Bibliothèque open source pour la vision par ordinateur. https://docs.opencv.org/
5. **TensorFlow Documentation** — Framework pour le développement et le déploiement de modèles d'apprentissage profond. https://www.tensorflow.org/
6. **FaceNet: A Unified Embedding for Face Recognition and Clustering** (Schroff et al., 2015) https://arxiv.org/abs/1503.03832
7. Tutoriels et ressources utilisées pour l'entraînement du modèle :
    - Tutoriel TensorFlow Keras CNN pour reconnaissance faciale
    - Tutoriel OpenCV pour la détection de visage en temps réel