



République du Sénégal

Un Peuple – Un But – Une Foi

*Ministère de l'enseignement technique et de la
Formation Professionnelle*

**Projet de Fin d'étude: Implémentation d'un Système de
Reconnaissance Faciale pour la sécurité à l'Orange Digital
Center**

Ibrahima Gabar Diop

24/04/2024

rédempteur: Développement Data

Superviseur académique: **Mamadou Mbaye , Abdoulaye Dieng**

Établissement: École de Code Sonatel Academy

Avant-Propos

Au cœur de la transformation digitale en Afrique, Sonatel Academy se distingue comme un phare d'espoir et d'opportunités, éclairant la voie vers un avenir numérique prospère pour le Sénégal. Née de l'engagement du groupe Sonatel envers l'autonomisation des jeunes, cette école de codage gratuite offre une formation de pointe en **développement data**, en **développement web/mobile** et en **référencement digital** avec un accompagnement personnalisé, permettant aux talents sénégalais et étranger de s'épanouir dans un monde en constante évolution technologique.

C'est un véritable tremplin vers l'excellence. En dispensant des compétences techniques et en nourrissant les "skills" essentielles, l'école prépare les jeunes à relever les défis du monde numérique de demain et à s'intégrer avec succès au sein d'entreprises de renommée. Elle est intégrée à l'écosystème dynamique de l'Orange Digital Center, qui lui permet de bénéficier d'un réseau d'experts et d'opportunités inestimables. Ce partenariat stratégique renforce l'école dans sa mission d'offrir une formation de qualité à la pointe des dernières tendances technologiques, préparant ainsi les jeunes à devenir des acteurs clés de la révolution numérique au Sénégal et à l'étranger.

Sonatel Academy n'est pas seulement un lieu d'apprentissage, mais aussi une communauté vibrante où les apprenants se connectent, partagent des idées et s'inspirent mutuellement. L'école favorise un environnement inclusif et stimulant, où chaque individu est encouragé à développer son potentiel et à contribuer positivement à la société. En rejoignant Sonatel Academy, nous nous sommes embarqués dans un voyage extraordinaire qui a en partie transformé notre vie et notre façon de voir les choses. Avec cette école, on peut avoir le pouvoir de réaliser ses rêves et de bâtir un avenir meilleur pour soi-même, pour sa

communauté et pour sa nation.

Dédicaces et Remerciements

Ce travail est dédié à ceux qui ont illuminé mon chemin :

À ma famille , source d'inspiration et de soutien inconditionnel.

À mes mentors, formateurs et tuteurs de Sonatel Academy, pour la transmission de leur savoir et leur bienveillance.

À toutes les personnes qui ont cru en moi, et qui ont contribué à mon succès, de près ou de loin.

Ma profonde gratitude s'adresse à :

Sonatel Academy et Orange Digital Center, pour avoir créé cette opportunité unique de formation et d'insertion dans le domaine numérique.

L'équipe de Sonatel Academy, pour leur confiance, leur accompagnement et leur implication dans mon épanouissement personnel et professionnel.

Mes camarades de Sonatel Academy, pour les moments de partage, d'entraide et d'amitié qui ont enrichi mon expérience.

Ma famille et mes amis, pour leur soutien indéfectible et leur présence constante dans ma vie.

Suivre la formation de Sonatel Academy a été une expérience inoubliable qui a transformé ma vie. Je suis reconnaissant pour les compétences acquises, les relations tissées et les opportunités qui s'ouvrent à moi.

Table des Matières

Avant-Propos.....	2
Introduction.....	8
Chapitre 1 : Contextualisation de la Reconnaissance Faciale.....	9
1.1 Historique et Évolution.....	9
1.2 Enjeux et Applications Actuelles.....	9
1.3 Réflexion Éthique autour de la Technologie.....	10
Chapitre 2 : Collecte et Gestion des Données.....	12
2.1 Exploration et Collecte de Données.....	12
Chapitre 3: Prétraitement des Données.....	22
● Problèmes rencontrés et Solutions apportées.....	22
● Solutions apportées.....	22
3.1 Étapes de prétraitement.....	23
3.2 Justification des choix techniques.....	26
Chapitre 4 : Modélisation et Entraînement.....	28
4.1 Construction d'un modèle à partir de zéro.....	28
4.2 Transfert d'apprentissage simple avec ResNet-50.....	30
4.3 Transfert d'apprentissage avec "fine-tuning" de ResNet-50.....	33

4.4 Transfert d'apprentissage avec "fine-tuning" de ResNet-50 et optimisation...	36
Chapitre 5 : Intégration Matérielle.....	43
5.1 Choix du Matériel.....	43
Chapitre 6 : Développement de l'Interface Utilisateur.....	44
6.2 Fonctionnalités de l'interface utilisateur.....	44
6.3 Implémentation de l'interface utilisateur avec Streamlit.....	46
Chapitre 7 : Intégration avec les Systèmes de Sécurité Existant.....	53
7.1. Intégration via une API RESTful.....	53
7.2. Intégration directe au niveau du code.....	54
Chapitre 8 : Tests et Validation.....	56
<ul style="list-style-type: none">• Tests approfondis du système dans des scénarios réalistes• Validation de la fiabilité et de la précision de la reconnaissance faciale	
Conclusion.....	58
Bibliographie.....	60
<ul style="list-style-type: none">• Références bibliographiques, Webographiques et Sources Consultées	

Liste des Abréviations

1. **LFW** : Labelled Faces in the Wild, un dataset public de visages annotés.
2. **RAM** : Random Access Memory, la mémoire vive de l'ordinateur utilisée pour stocker des données temporaires.
3. **UI** : Interface Utilisateur.
4. **IP** : Internet Protocol.
5. **CSV** : Comma-Separated Values, un format de fichier texte représentant des données tabulaires sous forme de valeurs séparées par des virgules.
6. **CNN**: Convolutional Neural Network
7. **DataFrame**: Structure de données tabulaire en Python
8. **GPU** : Graphics Processing Unit
9. **HTTP** : Hypertext Transfer Protocol
10. **JSON** : JavaScript Object Notation
11. **ML** : Machine Learning
12. **RAM** : Random Access Memory
13. **REST** : Representational State Transfer
14. **ROI** :: Region of Interest
15. **SDK** : Software Development Kit

Glossaires

- 1. Réseau de neurones convolutifs (CNN)** : Un type de réseau neuronal profond, spécialement conçu pour le traitement d'images.
- 2. Transfert Learning**: Une technique en apprentissage automatique où un modèle pré-entraîné est utilisé comme point de départ pour résoudre une tâche similaire à celle pour laquelle il a été entraîné initialement.
- 3. Fine-tuning** : Une technique consistant à ajuster les poids d'un modèle pré-entraîné sur une tâche spécifique, généralement en décongelant et en entraînant certaines couches du modèle.
- 4. Surentraînement (Overfitting)** : Un phénomène en apprentissage automatique où un modèle apprend trop spécifiquement aux données d'entraînement et ne généralise pas bien aux données invisibles.
- 5. API RESTful** : Une interface de programmation d'application qui suit les principes de l'architecture REST pour permettre la communication entre les systèmes logiciels.
- 6. Base de données** : Ensemble structuré de données organisé pour un accès et une manipulation efficaces.
- 7. Dataset** : Ensemble de données utilisées pour entraîner et évaluer des modèles d'apprentissage automatique.
- 8. Détection de visages** : Processus d'identification et de localisation des visages dans des images ou des vidéos.
- 9. Interface utilisateur (UI)** : Interface graphique qui permet aux utilisateurs d'interagir avec un système informatique.
- 10. Régularisation** : Technique en apprentissage automatique pour éviter le surentraînement et améliorer la généralisation du modèle.
- 11. ROC** : Receiver Operating Characteristic, une courbe représentant la performance d'un modèle de classification binaire à différents seuils de décision.
- 12. ReLU** : Rectified Linear Unit, une fonction d'activation couramment utilisée dans les réseaux de neurones profonds.

Introduction

L'innovation étant devenue la norme, la sécurité des environnements professionnels est une priorité absolue. Pour l'Orange Digital Center, une des figures de proue de l'excellence technologique au Sénégal, nous nous sommes inscrits dans cette dynamique en s'attaquant à un projet ambitieux : la mise en place d'un système de reconnaissance faciale.

Ce projet visionnaire exploite la puissance de l'intelligence artificielle, du transfert learning et des réseaux de neurones convolutifs pour offrir une solution de sécurité moderne et performante. De l'analyse des besoins à la maintenance, chaque étape sera abordée avec rigueur et méthodologie. La sélection du dataset, l'entraînement du modèle, l'intégration matérielle et logicielle, et le développement de l'interface utilisateur seront autant de défis à relever pour garantir l'efficacité et la pérennité du système.

Ce mémoire de fin d'études se propose de décrypter chaque étape du processus de développement. Défis, méthodes et solutions envisagées seront exposés en détail, avec pour objectif de proposer une solution robuste et parfaitement adaptée aux besoins de l'Orange Digital Center.

En m'impliquant dans ce projet, je souhaite contribuer activement à l'essor de notre communauté et à la construction d'un avenir numérique plus sûr et plus prometteur pour tous.

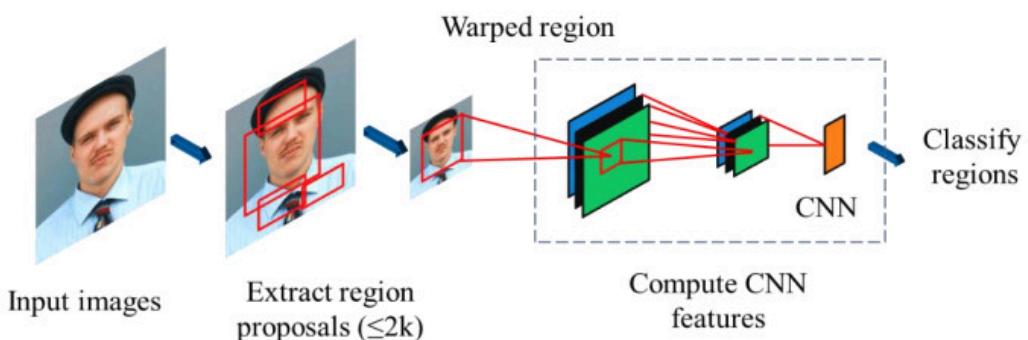
Chapitre 1: Contextualisation de la Reconnaissance Faciale

1.1 Historique et Évolution

La reconnaissance faciale, aujourd'hui au cœur de nos préoccupations sécuritaires, a une genèse fascinante. Ses premiers pas, modestes dans les années 1960 avec les travaux de pionniers tels que **Woodrow W. Bledsoe**, reposaient sur des repères géométriques. Cependant, les limites étaient évidentes, entravées par la qualité médiocre des images et la puissance de calcul limitée de l'époque.



L'avènement de l'informatique moderne a marqué un tournant majeur. Dans les années 1990, l'exploration des modèles 3D a élargi les horizons. Mais c'est véritablement l'avènement du **Deep Learning** et des **CNN** dans les années 2010 qui a propulsé la reconnaissance faciale vers de nouveaux sommets de précision et d'efficacité.



1.2 Enjeux et Applications Actuelles

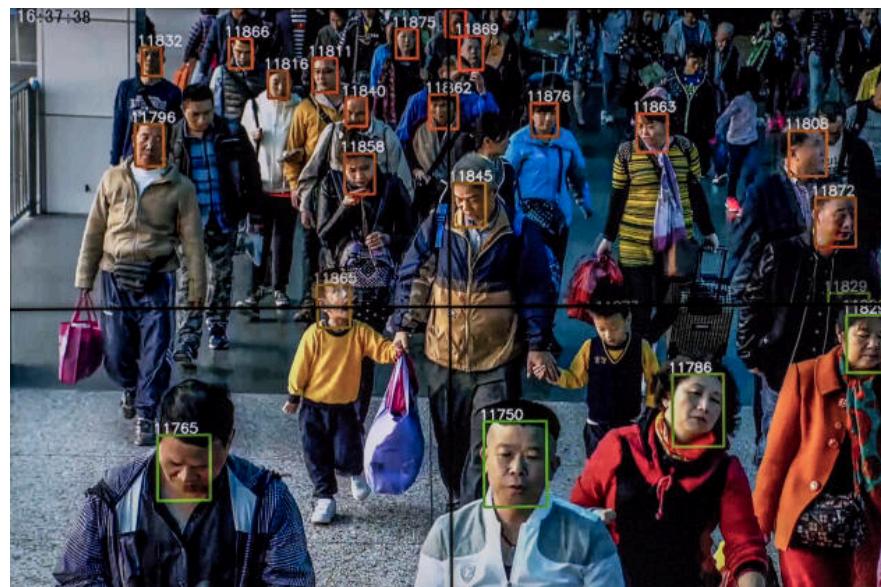
La reconnaissance faciale a radicalement redéfini la sécurité. Des aéroports aux centres urbains, elle s'est insinuée dans nos vies quotidiennes, élevant les normes de contrôle

d'accès et de surveillance. Prenons l'exemple du système "CLEAR" déployé dans les aéroports américains. Cette technologie permet une identification rapide des voyageurs, réduisant les temps d'attente et renforçant la sécurité.

Mais la polyvalence de la reconnaissance faciale va bien au-delà de la sécurité. Elle s'est immiscée dans divers aspects de notre quotidien. Des applications pratiques émergent, comme l'utilisation d'**Apple Face ID** pour la gestion sécurisée des identités, ou son intégration dans les services financiers pour renforcer la sécurité des transactions.

1.3 Réflexion Éthique autour de la Technologie

Cependant, ce déploiement massif n'est pas sans susciter des questions éthiques profondes. La délicate équation entre vie privée et sécurité est au cœur des débats. En Chine, par exemple, les systèmes de surveillance utilisant la reconnaissance faciale ont été critiqués pour leur potentiel à violer la vie privée des citoyens, incitant à une réflexion approfondie sur la réglementation et la gouvernance.



De plus, les préoccupations actuelles s'étendent aux possibles biais algorithmiques. Des études révèlent que les systèmes peuvent être moins précis pour certaines populations, soulevant des questions cruciales d'équité et de justice dans l'application de cette technologie.

Ce premier chapitre plonge au cœur de l'histoire, des enjeux actuels et des défis éthiques

de la reconnaissance faciale. En comprenant ces éléments, nous jetons les bases nécessaires pour aborder avec discernement la mise en place d'un système de reconnaissance faciale à l'Orange Digital Center, une initiative qui s'inscrit dans ce continuum passionnant de progrès et de dilemmes.

Chapitre 2 : Collecte et Gestion des Données

Ce chapitre présente une exploration détaillée des données utilisées pour le développement d'un système de reconnaissance faciale. Nous allons nous plonger dans les différentes étapes de la collecte et de la gestion des données, en soulignant les défis rencontrés, les solutions mises en place et les résultats obtenus.

2.1 Exploration et Collecte de Données

La collecte de données auprès du personnel et des apprenants de l'Orange Digital Center s'est avérée complexe. Le manque de temps, de ressources et les problèmes de confidentialité ont constitué des obstacles majeurs.

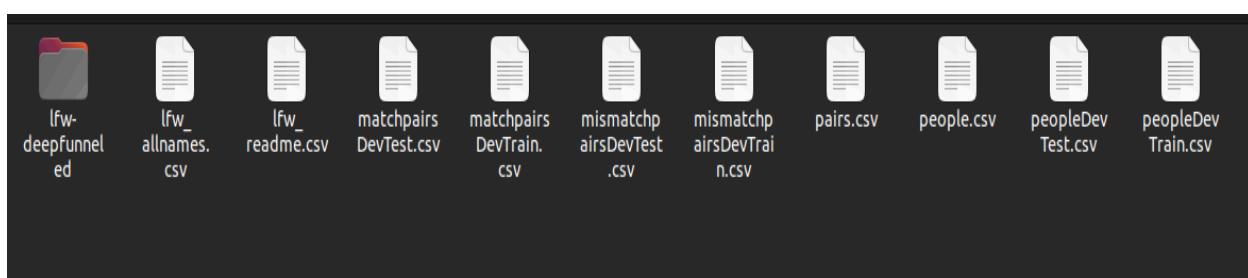
1. Défis rencontrés:

Manque de temps et de ressources: Le temps imparti pour la collecte de données était limité, et les ressources humaines et matérielles disponibles étaient insuffisantes pour mener à bien la tâche.

Problèmes de confidentialité: La collecte de données biométriques, comme les images faciales, soulève des questions de confidentialité qui ont nécessité une attention particulière.

2. Solutions mises en place:

Utilisation d'un dataset public: Pour pallier le manque de données et de temps, un dataset public de visages annotés, LFW (Labelled Faces in the Wild), a été utilisé.



Collecte de données de personnalité publique: Afin d'enrichir le dataset et de le rendre plus pertinent pour le contexte local, des photos de personnalité publiques

comme des politiciens ou des artistes ont été collectées à l'instar de l'ancien président de la république du Sénégal, maître Abdoulaye Wade.

3. Exploration du dataset LFW:

Analyse des fichiers CSV: Le contenu des fichiers CSV a été exploré pour identifier les informations disponibles (noms des personnes, nombre d'images, etc.). Des statistiques descriptives ont été calculées pour chaque variable numérique (moyenne, écart-type, minimum, maximum).

Chargement et visualisation d'images:

Un échantillon d'images du dataset a été chargé et affiché sous forme de grille pour une première exploration visuelle.

Des outils de visualisation de matplotlib et seaborn ont été utilisés pour afficher les images de manière interactive.

George_W_Bush



Colin_Powell



Tony_Blair



Donald_Rumsfeld



Gerhard_Schroeder



Ariel_Sharon



Hugo_Chavez



Junichiro_Koizumi



Jean_Chretien



John_Ashcroft



Jacques_Chirac



Serena_Williams



Vladimir_Putin



Luiz_Inacio_Lula_da_Silva



Gloria_Macapagal_Arroyo



Arnold_Schwarzenegger



Jennifer_Capriati



Laura_Bush



Lleyton_Hewitt



Alejandro_Toledo



George_W_Bush



Colin_Powell



Tony_Blair



Donald_Rumsfeld



Gerhard_Schroeder



Ariel_Sharon



Hugo_Chavez



Junichiro_Koizumi



Jean_Chretien



John_Ashcroft



Jacques_Chirac



Serena_Williams



Vladimir_Putin



Luiz_Inacio_Lula_da_Silva



Gloria_Macapagal_Arroyo



Arnold_Schwarzenegger



Jennifer_Capriati



Laura_Bush



Lleyton_Hewitt



Alejandro_Toledo



George_W_Bush



Colin_Powell



Tony_Blair



Donald_Rumsfeld



Gerhard_Schroeder



Ariel_Sharon



Hugo_Chavez



Junichiro_Koizumi



Jean_Chretien



John_Ashcroft



Jacques_Chirac



Serena_Williams



Vladimir_Putin



Luiz_Inacio_Lula_da_Silva



Gloria_Macapagal_Arroyo



Arnold_Schwarzenegger



Jennifer_Capriati



Laura_Bush



Lleyton_Hewitt

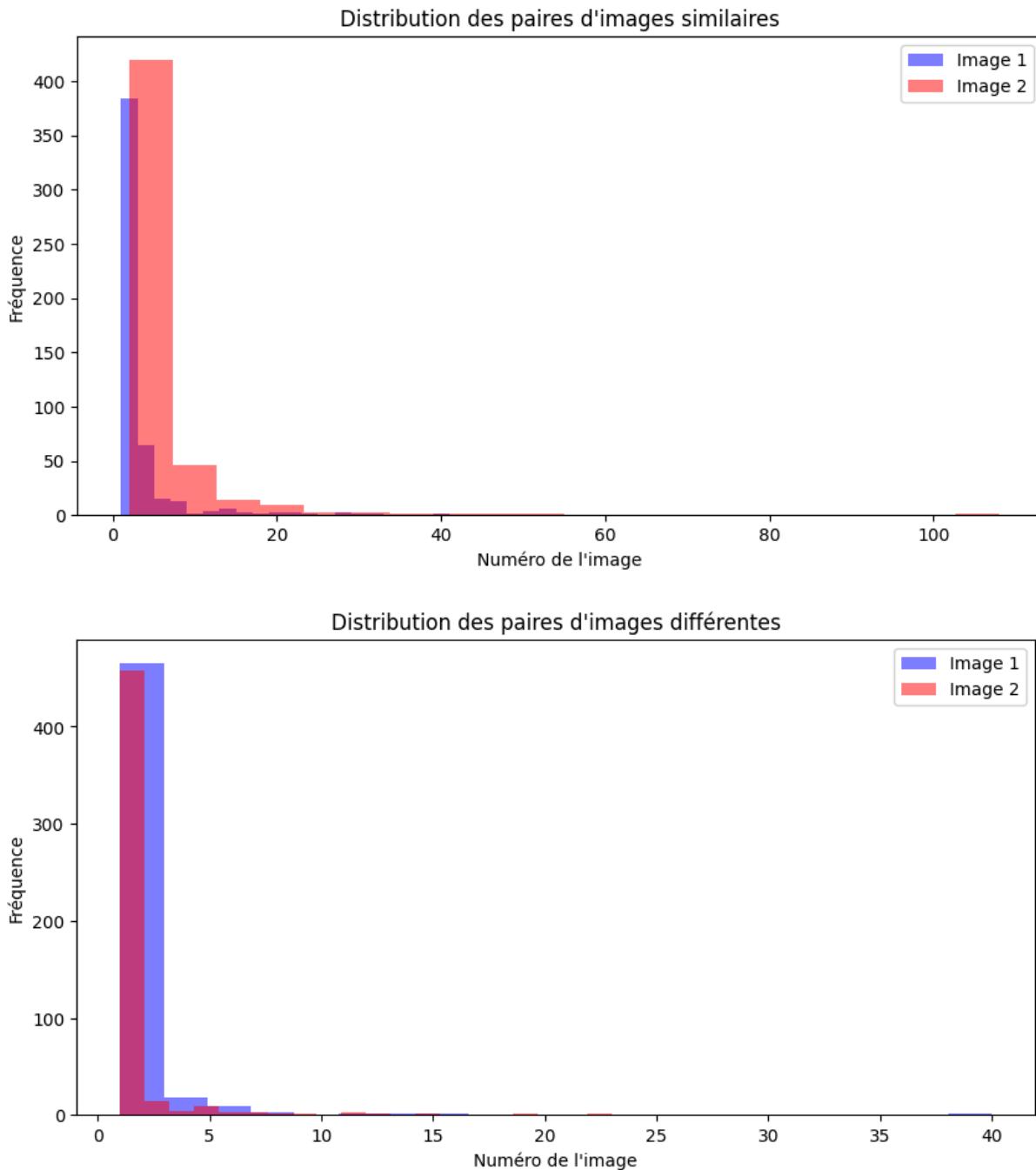


Alejandro_Toledo



Analyse des paires d'images:

Les fichiers CSV contenant des paires d'images (correspondantes ou non) ont été identifiés et explorés.



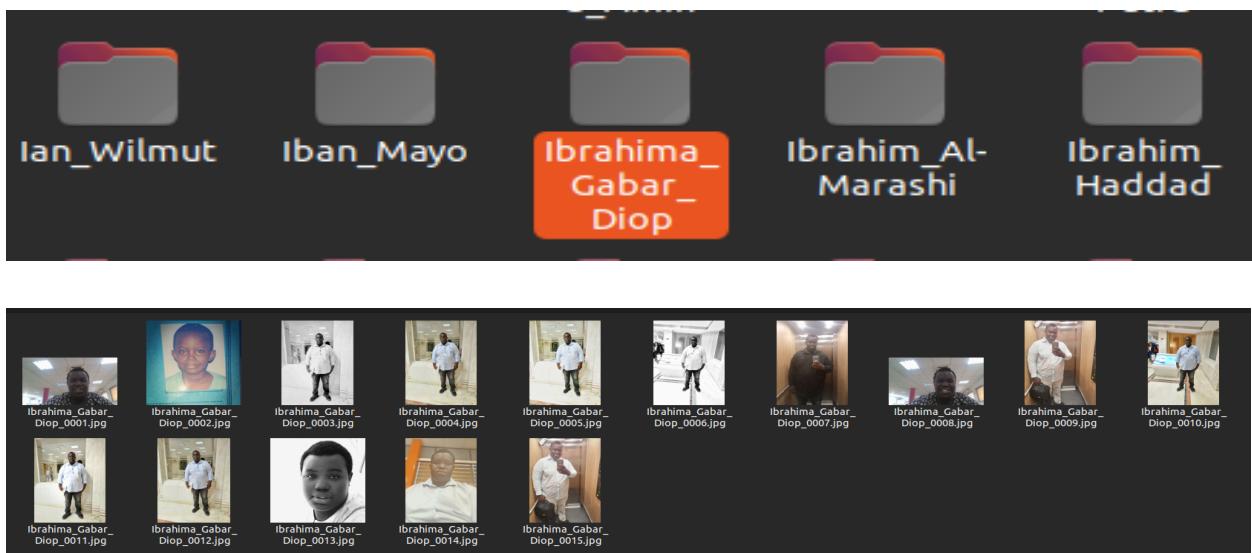
Les distributions des distances inter-classes et intra-classes ont été étudiées pour évaluer la difficulté de la tâche de classification.

4. Enrichissement du dataset avec des photos personnelles:

Collecte et sélection des photos:

Des photos de volontaires ont été collectées en respectant les exigences de confidentialité.

Un processus de sélection a été appliqué pour garantir la qualité et la diversité des images.



Annotation des photos:

Les photos collectées de moi ont été annotées avec les informations nécessaires (nom de la personne, index de la photo, etc.).

Des outils d'annotation d'images ont été utilisés pour faciliter le processus.



5. Analyse du nombre d'images par personne:

Tri du DataFrame:

Le DataFrame contenant les informations sur les personnes a été trié en fonction du nombre d'images disponibles.

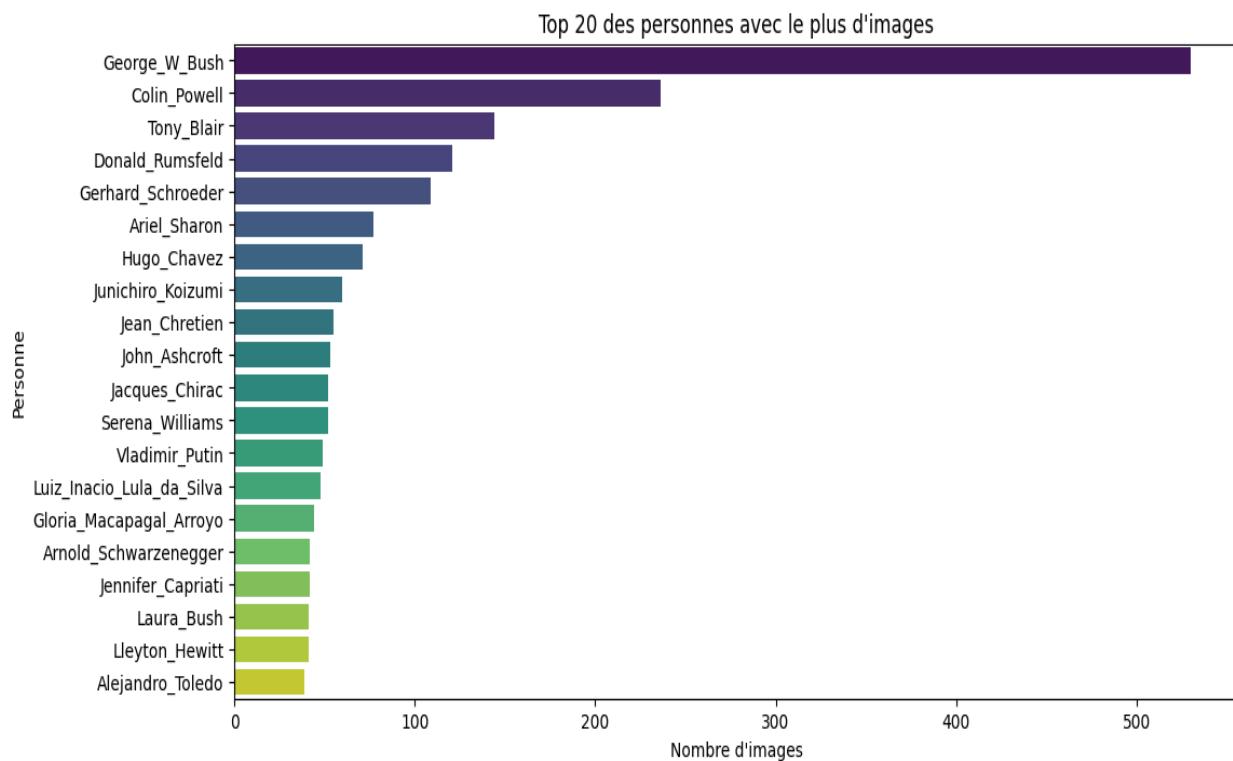
Top 20 des personnes:

Les 20 personnes avec le plus grand nombre d'images ont été identifiées et analysées.

		name	images
5355		George_W_Bush	530.0
1811		Colin_Powell	236.0
1129		Tony_Blair	144.0
3503		Donald_Rumsfeld	121.0
3008		Gerhard_Schroeder	109.0
5193		Ariel_Sharon	77.0
242		Hugo_Chavez	71.0
4856		Junichiro_Koizumi	60.0
3614		Jean_Chretien	55.0

5440	John_Ashcroft	53.0
822	Jacques_Chirac	52.0
4481	Serena_Williams	52.0
3954	Vladimir_Putin	49.0
3721	Luiz_Inacio_Lula_da_Silva	48.0
3551	Gloria_Macapagal_Arroyo	44.0
3999	Arnold_Schwarzenegger	42.0
1968	Jennifer_Capriati	42.0
1474	Laura_Bush	41.0
364	Lleyton_Hewitt	41.0
2851	Alejandro_Toledo	39.0

Un graphique à barres a été créé pour visualiser la distribution du nombre d'images par personne.



6. Visualisation d'images avec les noms correspondants:

Affichage d'images pour les 20 premières personnes:

Des exemples d'images pour les 20 personnes avec le plus d'images ont été sélectionnés et affichés.

Pour chaque personne, jusqu'à 3 images ont été affichées avec leurs noms respectifs.



Colin_Powell

Image 1



Image 2



Image 3



Tony_Blair

Image 1



Image 2



Image 3



Donald_Rumsfeld

Image 1



Image 2



Image 3



Gerhard_Schroeder

Image 1



Image 2



Image 3



Ariel_Sharon

Image 1



Image 2



Image 3



Hugo_Chavez

Image 1



Image 2



Image 3



Luiz_Inacio_Lula_da_Silva

Image 1



Image 2



Image 3



Serena_Williams

Image 1



Image 2



Image 3



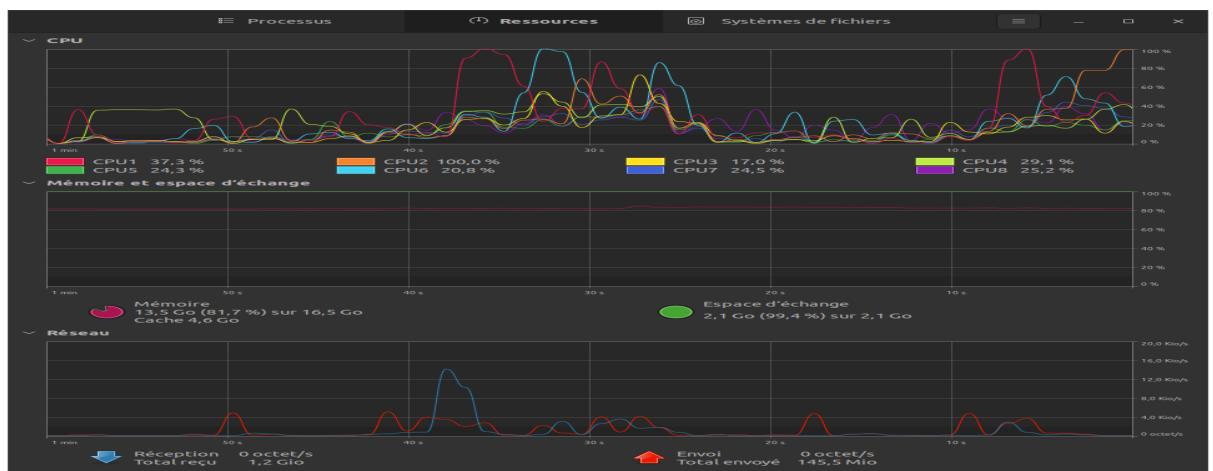
Chapitre 3 :Prétraitement des Données

Le prétraitement des données est une étape essentielle dans tout projet d'apprentissage automatique. Son objectif est de transformer les données brutes en un format compatible avec les algorithmes d'apprentissage et d'améliorer leur qualité pour optimiser les performances des modèles.

Problèmes rencontrés

Le dataset LFW initialement utilisé pour ce projet présentait deux défis majeurs:

- **Taille du dataset:** Le dataset contenait des images pour plus de 5000 personnes, ce qui représentait un volume important de données à traiter. La capacité de la machine, limitée en termes de RAM et de puissance de calcul, ne permettait pas de gérer efficacement l'ensemble du dataset.



- **Ressources matérielles:** La configuration de la machine ne correspondait pas aux exigences du traitement d'images à grande échelle. Cela se traduisait par des temps de traitement longs et limitait la complexité des prétraitements que nous pouvions appliquer aux images.

Solutions apportées

Pour surmonter ces obstacles, deux solutions principales ont été mises en place:

- **Réduction du dataset:** Afin de réduire la taille du dataset tout en conservant une diversité acceptable, nous avons sélectionné les 1000 personnes avec le plus grand nombre d'images. Cette sélection a permis de réduire le nombre d'images à traiter par un facteur d'environ 125, tout en conservant une large représentation des différentes classes (personnes).
- **Optimisation du code:** Le code de prétraitement a été optimisé en utilisant des techniques de vectorisation et en choisissant des algorithmes efficents. Cela a

permis de réduire significativement les temps de traitement et d'améliorer la performance globale du processus.

3.1 Étapes de prétraitement

Le prétraitement des données s'est déroulé en plusieurs étapes:

- **Chargement des images:** Les images ont été chargées à partir du dataset et stockées en mémoire. Nous avons choisi toutes les images du top 1000 des personnes avec le plus d'images dans le dataset, donc les 1000 personnes les plus représentés dans le dataset

```
import os
import numpy as np
from PIL import Image
from skimage import transform
import cv2
import matplotlib.pyplot as plt
import random

# Chemin du dossier contenant les images
images_dir = "/home/gblack98/Images/python/travail_IGD/archive (1)/lfw-deepfunneled/lfw-deepfunneled"

# Chargement des noms de personnes et de leurs nombres d'images
def load_people_with_images_count():
    people_with_images_count = {}
    for person_name in os.listdir(images_dir):
        person_images_dir = os.path.join(images_dir, person_name)
        num_images = len(os.listdir(person_images_dir))
        people_with_images_count[person_name] = num_images
    # Trier les personnes par nombre d'images (du plus grand au plus petit)
    sorted_people = sorted(people_with_images_count.items(), key=lambda x: x[1], reverse=True)
    return sorted_people[:40] # Sélectionner les 40 premières personnes

# Charger les chemins d'accès des images pour les 40 premières personnes
def load_top_people_images_paths(num_images_per_person=3):
    top_people_images_paths = []
    top_people = load_people_with_images_count()
    for person_name, _ in top_people:
        person_images_dir = os.path.join(images_dir, person_name)
        person_images = os.listdir(person_images_dir)[:num_images_per_person]
        person_images_paths = [os.path.join(person_images_dir, img) for img in person_images]
        top_people_images_paths.extend(person_images_paths)
    return top_people_images_paths
```

- **Redimensionnement:** Toutes les images ont été redimensionnées à une taille standard de 224x224 pixels. Cette étape permet d'assurer une cohérence dans la taille des images et de faciliter leur traitement par les algorithmes d'apprentissage automatique

```
# Fonction de redimensionnement des images
def resize_images(image_paths, target_size=(224, 224)):
    resized_images = []
    for img_path in image_paths:
        img = Image.open(img_path)
        resized_img = img.resize(target_size, Image.LANCZOS)
        resized_images.append(resized_img)
    return resized_images
```

- **Conversion en niveaux de gris:** La conversion des images en niveaux de gris a permis de réduire la dimensionnalité des données et de se concentrer sur les informations de texture, qui sont particulièrement importantes pour la reconnaissance faciale.

```
# Conversion en niveaux de gris
def convert_to_grayscale(images):
    gray_images = []
    for img in images:
        gray_img = cv2.cvtColor(np.array(img), cv2.COLOR_RGB2GRAY)
        gray_images.append(gray_img)
    return np.array(gray_images)
```

- **Normalisation:** Les valeurs de pixel des images ont été normalisées entre 0 et 1. Cette normalisation permet d'améliorer la convergence des algorithmes d'apprentissage et de garantir une meilleure comparabilité des images.

```
# Normalisation des valeurs de pixel
def normalize_images(images):
    normalized_images = []
    for img in images:
        normalized_img = img / 255.0 # Normalisation des valeurs de pixel entre 0 et 1
        normalized_images.append(normalized_img)
    return np.array(normalized_images)
```

- **Augmentation des données:** Pour enrichir le dataset et améliorer la robustesse des modèles, des transformations aléatoires ont été appliquées aux images, telles que des rotations et des changements d'échelle.

```
# Augmentation des données (exemple : rotation aléatoire)
def augment_images(images):
    augmented_images = []
    for img in images:
        angle = random.uniform(-10, 10)
        augmented_img = transform.rotate(img, angle) # Rotation aléatoire de l'image
        augmented_images.append(augmented_img)
    return np.array(augmented_images)
```

- **Suppression du bruit:** Un filtre de débruitage médian a été appliqué pour supprimer le bruit présent dans certaines images, ce qui peut améliorer la précision de la détection des visages.

```

def remove_noise(images):
    denoised_images = []
    for img in images:
        # Assurez-vous que l'image est au format uint8 (8-bit unsigned integer)
        img_uint8 = np.uint8(img)
        # Appliquer le filtre de débruitage médian
        denoised_img = cv2.medianBlur(img_uint8, 5)
        denoised_images.append(denoised_img)
    return np.array(denoised_images)

```

- **Extraction des caractéristiques:** La détection de visage a été utilisée pour extraire les caractéristiques faciales des images. Cette étape permet de se concentrer sur les parties les plus informatives des images pour la reconnaissance faciale.

```

# Extraction des caractéristiques (exemple : détection de visage avec OpenCV)
def extract_features(images):
    face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
    features = []
    for img in images:
        # Convertir l'image PIL en tableau numpy
        img_array = np.array(img)
        gray_img = cv2.cvtColor(img_array, cv2.COLOR_RGB2GRAY) # Convertir en niveau de gris
        faces = face_cascade.detectMultiScale(gray_img, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))
        for (x, y, w, h) in faces:
            face_img = gray_img[y:y+h, x:x+w]
            features.append(face_img)
    return features

```

```

# Affichage des images originales (optionnel)
for i in range(num_images_to_display):
    plt.subplot(3, num_images_to_display, i + 1)
    plt.imshow(resized_images[i], cmap='gray')
    plt.title('Original')
    plt.axis('off')

# Affichage des images augmentées (optionnel)
for i in range(num_images_to_display):
    plt.subplot(3, num_images_to_display, i + num_images_to_display + 1)
    plt.imshow(augmented_images[i], cmap='gray')
    plt.title('Augmented')
    plt.axis('off')

# Affichage des caractéristiques extraites (visages) (optionnel)
for i in range(num_images_to_display):
    plt.subplot(3, num_images_to_display, i + 2*num_images_to_display + 1)
    plt.imshow(face_features[i], cmap='gray')
    plt.title('Face Detected')
    plt.axis('off')

plt.tight_layout()
plt.show()

```



3.2 Justification des choix techniques

Le choix des techniques de prétraitement a été guidé par plusieurs objectifs:

- **Améliorer la qualité des images:** Le redimensionnement, la conversion en niveaux de gris et la suppression du bruit ont permis d'améliorer la qualité des images et de les rendre plus uniformes.
- **Augmenter la diversité du dataset:** L'augmentation des données a permis d'enrichir le dataset et de le rendre plus robuste aux variations de l'éclairage, de la pose et de l'angle de vue.
- **Réduire la dimensionnalité des données:** La conversion en niveaux de gris a permis de réduire la dimensionnalité des images, ce qui a simplifié le traitement et amélioré la performance des algorithmes.
- **Améliorer la performance des modèles:** La normalisation des valeurs de pixel a permis d'améliorer la convergence des algorithmes d'apprentissage et d'optimiser la précision des modèles.
- **Simplifier le traitement:** Le choix d'algorithmes efficents et de techniques de vectorisation a permis de simplifier le traitement et de réduire les temps de calcul.

Le prétraitement des données a permis de transformer les images brutes en un format compatible avec l'apprentissage automatique et d'améliorer leur qualité.

Les étapes appliquées ont contribué à:

-
- **Réduire la taille du dataset:** Le dataset a été réduit de manière significative tout en conservant une diversité acceptable.
- **Améliorer la performance:** Le temps de traitement a été réduit et la performance des algorithmes d'apprentissage a été améliorée.
- **Augmenter la robustesse:** Les modèles d'apprentissage automatique sont devenus plus robustes aux variations des images.

C'est une étape cruciale qui a permis de préparer les données pour l'apprentissage automatique. Les solutions mises en place pour gérer les problèmes rencontrés et les choix techniques effectués ont contribué à obtenir une certaine quantité de données optimale prête à l'emploi pour la suite du projet.

Chapitre 4 :Modélisation et Entraînement

Ce chapitre décrit la démarche suivie pour développer un système de reconnaissance faciale. Trois techniques d'apprentissage profond ont été explorées et évaluées : la construction d'un modèle à partir de zéro, le transfert d'apprentissage simple et le transfert d'apprentissage avec "fine-tuning".

4.1 Construction d'un modèle à partir de zéro

La première approche consistait à construire un réseau de neurones convolutifs profonds (CNN) personnalisé pour la reconnaissance faciale. Le code Python suivant illustre la structure du modèle implémenté avec TensorFlow et Keras :

```

import os
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications.resnet import preprocess_input
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D

# Chemin du dossier contenant les images
images_dir = "/home/gblack98/Images/python/travail_IGD/archive (1)/lfw-deepfunneled/lfw-deepfunneled"

# Sélection des 1000 premières classes avec le plus grand nombre d'images
top_1000_people = sorted(os.listdir(images_dir), key=lambda x: len(os.listdir(os.path.join(images_dir, x))), reverse=True)[:1000]

# Crédit à un générateur d'images pour charger et prétraiter les données
datagen = ImageDataGenerator(preprocessing_function=preprocess_input, validation_split=0.2)

# Chargement des données depuis le répertoire pour les 1000 premières classes
train_generator = datagen.flow_from_directory(
    images_dir,
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',
    subset='training',
    classes=top_1000_people
)

validation_generator = datagen.flow_from_directory(
    images_dir,
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',
    subset='validation',
    classes=top_1000_people
)

# Chargement du modèle ResNet50 pré-entraîné sans les couches fully-connected
base_model = ResNet50(weights='imagenet', include_top=False)

```

Les résultats obtenus avec ce modèle n'étaient pas satisfaisants. Le taux de validation de la précision (val_accuracy) est resté faible, indiquant un problème de surentraînement (overfitting). Les graphiques ci-dessous illustrent l'évolution de la précision et de la perte sur les données d'entraînement et de validation :

```

# Geler les poids du modèle de base (ResNet50) pour empêcher l'entraînement
base_model.trainable = False

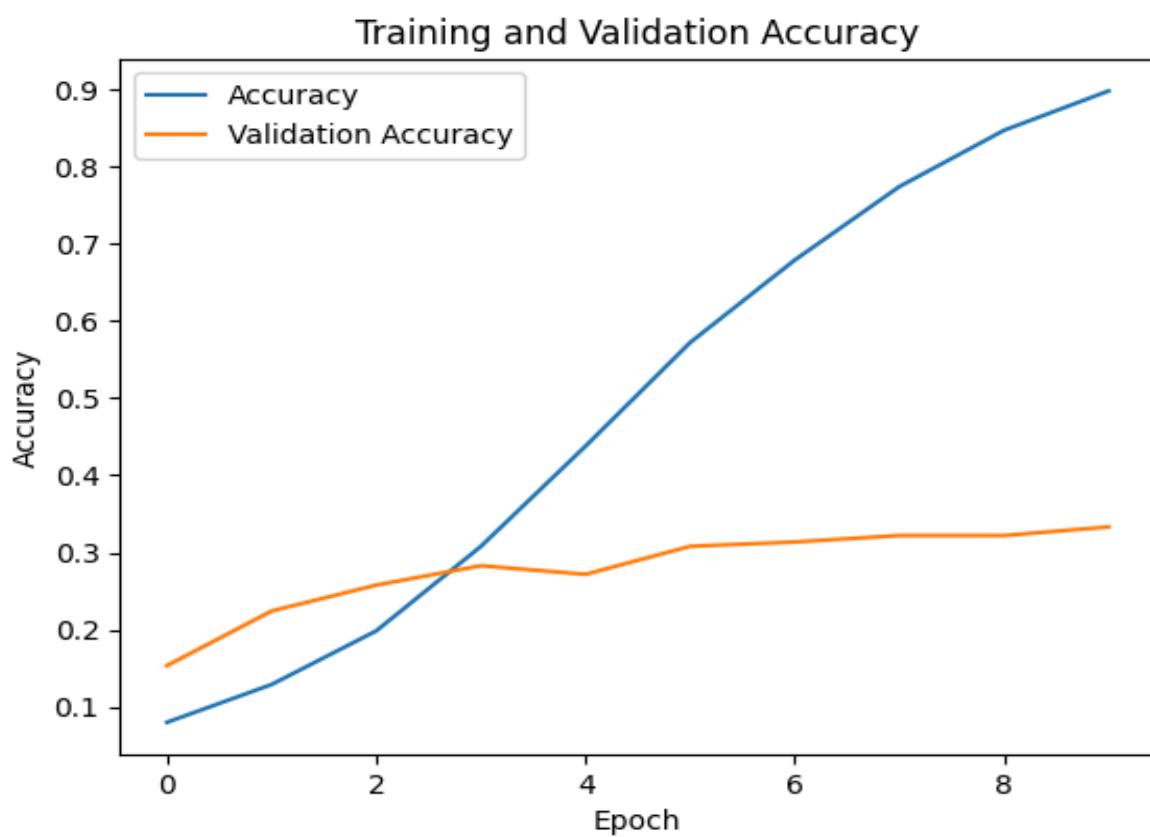
# Compilation du modèle
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

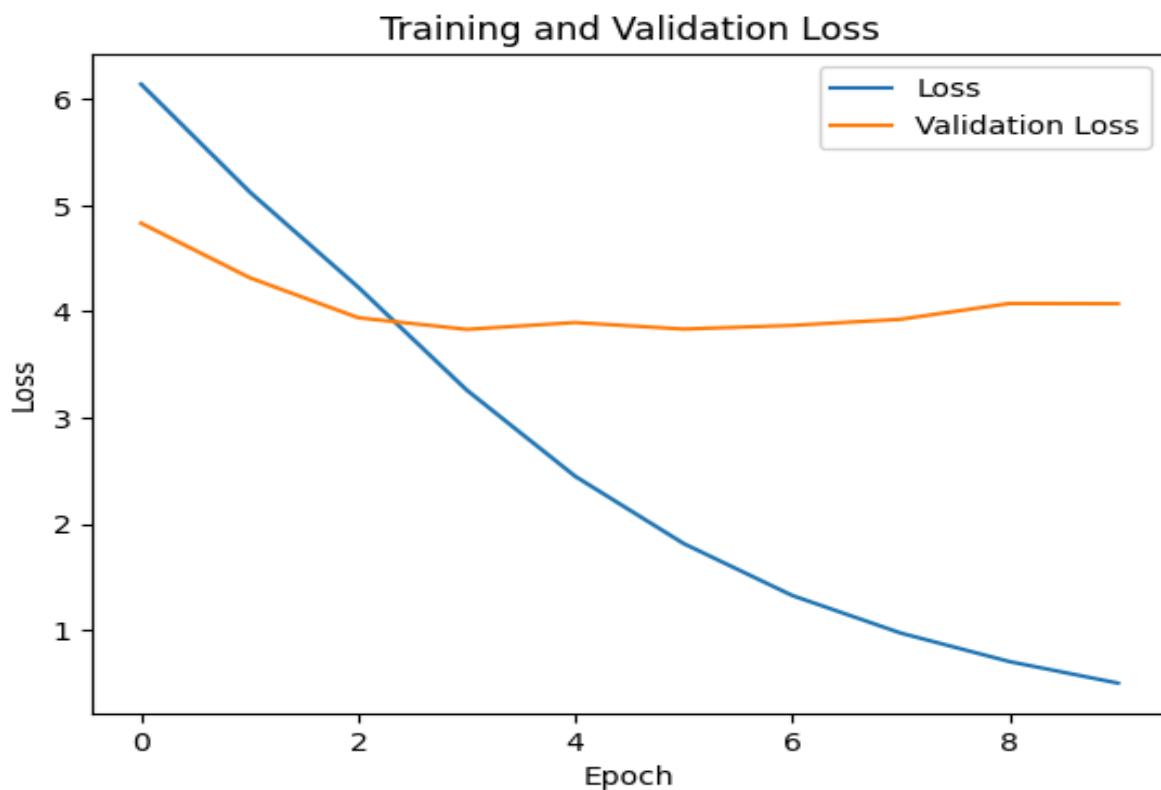
# Entraînement du modèle
history = model.fit(train_generator, validation_data=validation_generator, epochs=10)

# Affichage des résultats de l'entraînement
plt.plot(history.history['accuracy'], label='Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.title('Training and Validation Accuracy')
plt.legend()
plt.show()

plt.plot(history.history['loss'], label='Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Training and Validation Loss')
plt.legend()
plt.show()

```





Comme le montre l'illustration, la précision sur les données d'entraînement augmente rapidement, mais la précision sur les données de validation reste faible. Cela suggère que le modèle apprend trop spécifiquement aux données d'entraînement et ne généralise pas bien aux données invisibles.

4.2 Transfert d'apprentissage simple avec ResNet-50

La deuxième technique explorée a consisté à utiliser un modèle pré-entraîné, ResNet-50, connu pour sa performance en classification d'images. L'idée était de profiter des connaissances apprises par ResNet-50 sur des tâches de vision par ordinateur en général, puis de l'adapter à la tâche spécifique de reconnaissance faciale. Le code suivant montre la mise en œuvre de cette approche :

```

import os
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications.resnet import preprocess_input
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D, GlobalAveragePooling2D, Dense

# Chemin du dossier contenant les images
images_dir = "/home/gblack98/Images/python/travail_IGD/archive (1)/lfw-deepfunneled/lfw-deepfunneled"

# Sélection des 1000 premières classes avec le plus grand nombre d'images
top_1000_people = sorted(os.listdir(images_dir), key=lambda x: len(os.listdir(os.path.join(images_dir, x))), reverse=True)[:1000]

# Création d'un générateur d'images pour charger et prétraiter les données
datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input,
    validation_split=0.2,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True
)

# Chargement des données depuis le répertoire pour les 1000 premières classes
train_generator = datagen.flow_from_directory(
    images_dir,
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',
    subset='training',
    classes=top_1000_people
)

```

```

validation_generator = datagen.flow_from_directory(
    images_dir,
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',
    subset='validation',
    classes=top_1000_people
)

# Chargement du modèle ResNet50 pré-entraîné sans les couches fully-connected
base_model = ResNet50(weights='imagenet', include_top=False)

# Ajout de couches de convolution
x = base_model.output
x = Conv2D(64, (3, 3), activation='relu', padding='same')(x)
x = MaxPooling2D((2, 2))(x)

# Ajout de couches de convolution supplémentaires
x = Conv2D(128, (3, 3), activation='relu', padding='same')(x)
x = MaxPooling2D((2, 2))(x)

# Ajout d'une couche de mise en commun globale
x = GlobalAveragePooling2D()(x)

# Ajout de couches fully-connected
x = Dense(1024, activation='relu')(x)

# Couche de sortie avec softmax
predictions = Dense(len(top_1000_people), activation='softmax')(x)

# Définition du modèle
model = Model(inputs=base_model.input, outputs=predictions)

# Geler les poids du modèle de base (ResNet50) pour empêcher l'entraînement
for layer in base_model.layers:
    layer.trainable = False

# Compilation du modèle
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Entraînement du modèle
history = model.fit(train_generator, validation_data=validation_generator, epochs=20)

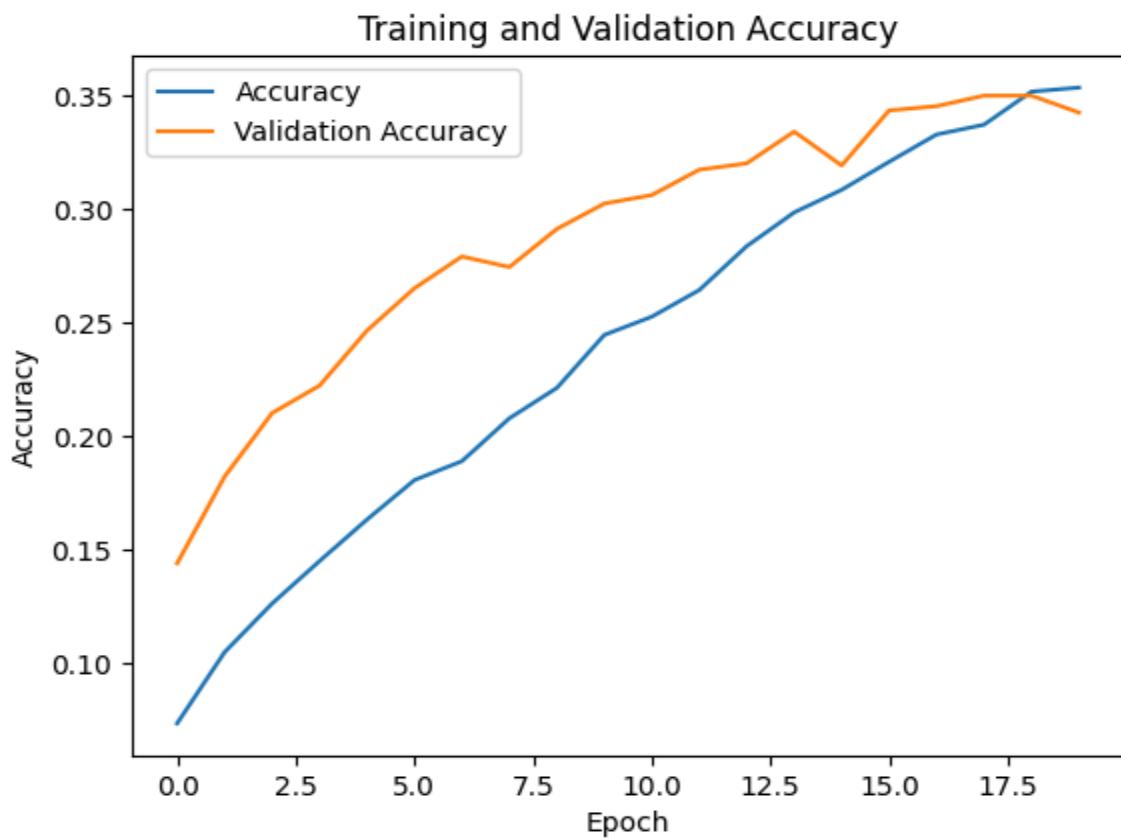
```

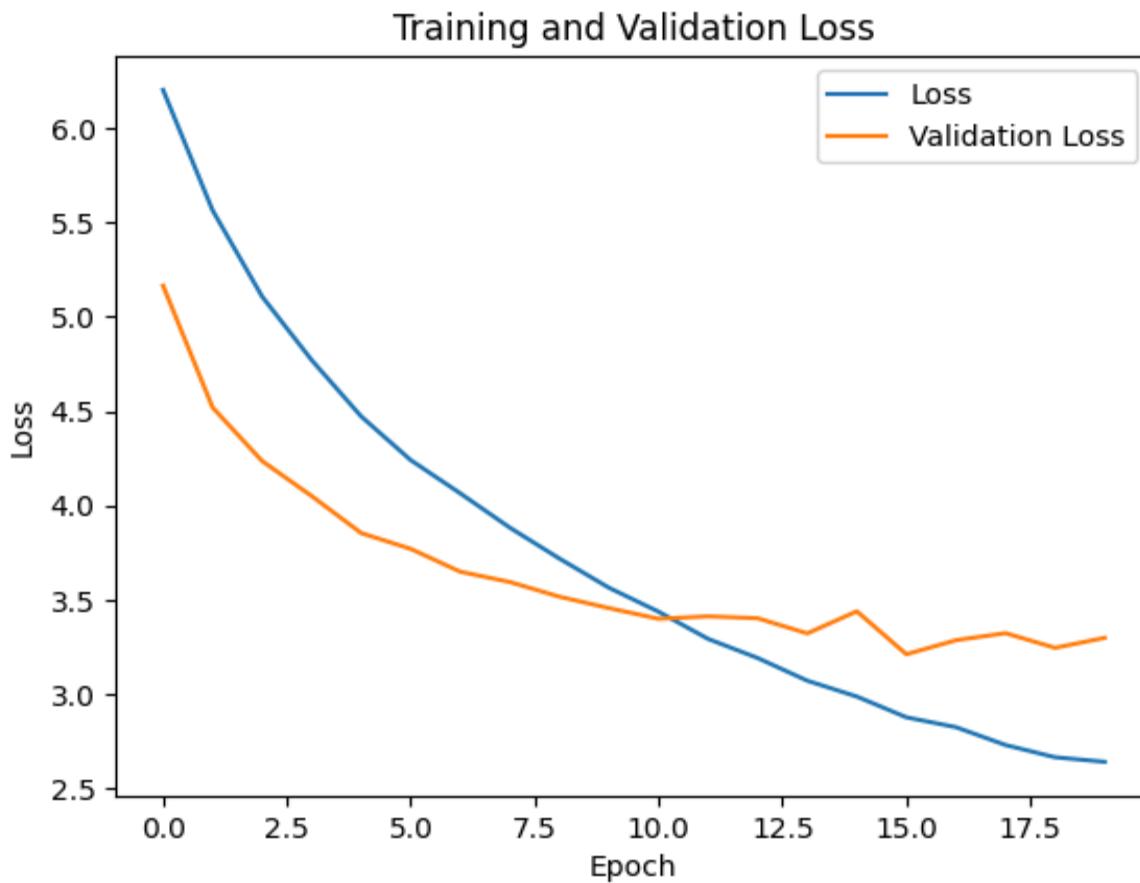
Dans ce cas, les poids des couches de ResNet-50 ont été gelés pendant l'entraînement, ce qui signifie que leurs paramètres n'ont pas été mis à jour. Seules les couches nouvellement ajoutées ont été entraînées sur le dataset de reconnaissance faciale.

Les résultats obtenus avec le transfert d'apprentissage simple ont montré une légère amélioration par rapport au modèle construit à partir de zéro. L'écart entre la précision sur les données d'entraînement et la précision sur les données de validation s'est réduit, comme le montre l'illustration suivante :

```
# Affichage des résultats de l'entraînement
plt.plot(history.history['accuracy'], label='Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.title('Training and Validation Accuracy')
plt.legend()
plt.show()

plt.plot(history.history['loss'], label='Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Training and Validation Loss')
plt.legend()
plt.show()
```





Cependant, le score de précision global restait relativement faible et le temps d'exécution de l'entraînement était très important (environ 2 heures).

4.3 Transfert d'apprentissage avec "fine-tuning" de ResNet-50

Étant donné les limitations des deux premières approches, la troisième technique explorée consistait à effectuer un transfert d'apprentissage avec "fine-tuning" de ResNet-50. Cette technique consiste à décongeler et à entraîner les dernières couches de ResNet-50 tout en gelant les couches inférieures. Cela permet au modèle d'ajuster les connaissances pré-entraînées de ResNet-50 à la tâche spécifique de reconnaissance faciale.

Le code d'implémentation de cette technique est similaire à celui du transfert d'apprentissage simple, à la différence que certaines couches de ResNet-50 sont dégelées et entraînées :

```

import os
import numpy as np
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications.resnet import preprocess_input
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D, GlobalAveragePooling2D, Dense

# Define paths and hyperparameters
images_dir = '/home/gblack98/Images/python/travail_IGD/archive (1)/lfw-deepfunneled/lfw-deepfunneled'
top_n_classes = 1000 # Adjust depending on the number of classes
target_size = (224, 224)
batch_size = 32
epochs = 20

# Data augmentation for training data
datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input,
    validation_split=0.2,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True
)

# Generators for training and validation data
train_generator = datagen.flow_from_directory(
    images_dir,
    target_size=target_size,
    batch_size=batch_size,
    class_mode='categorical',
    subset='training',
    classes=os.listdir(images_dir)[:top_n_classes] # Select top N classes
)

```

```

validation_generator = datagen.flow_from_directory(
    images_dir,
    target_size=target_size,
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation',
    classes=os.listdir(images_dir)[:top_n_classes]
)

# Load pre-trained ResNet50 and freeze base layers
base_model = ResNet50(weights='imagenet', include_top=False)
for layer in base_model.layers:
    layer.trainable = False

# Add custom layers for fine-tuning
x = base_model.output
x = Conv2D(64, (3, 3), activation='relu', padding='same')(x)
x = MaxPooling2D((2, 2))(x)
x = Conv2D(128, (3, 3), activation='relu', padding='same')(x)
x = MaxPooling2D((2, 2))(x)
x = GlobalAveragePooling2D()(x)
x = Dense(1024, activation='relu')(x)
predictions = Dense(top_n_classes, activation='softmax')(x)

# Create the final model
model = Model(inputs=base_model.input, outputs=predictions)

# Compile the model with Adam optimizer and categorical crossentropy loss
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Train the model
history = model.fit(train_generator, validation_data=validation_generator, epochs=epochs)

```

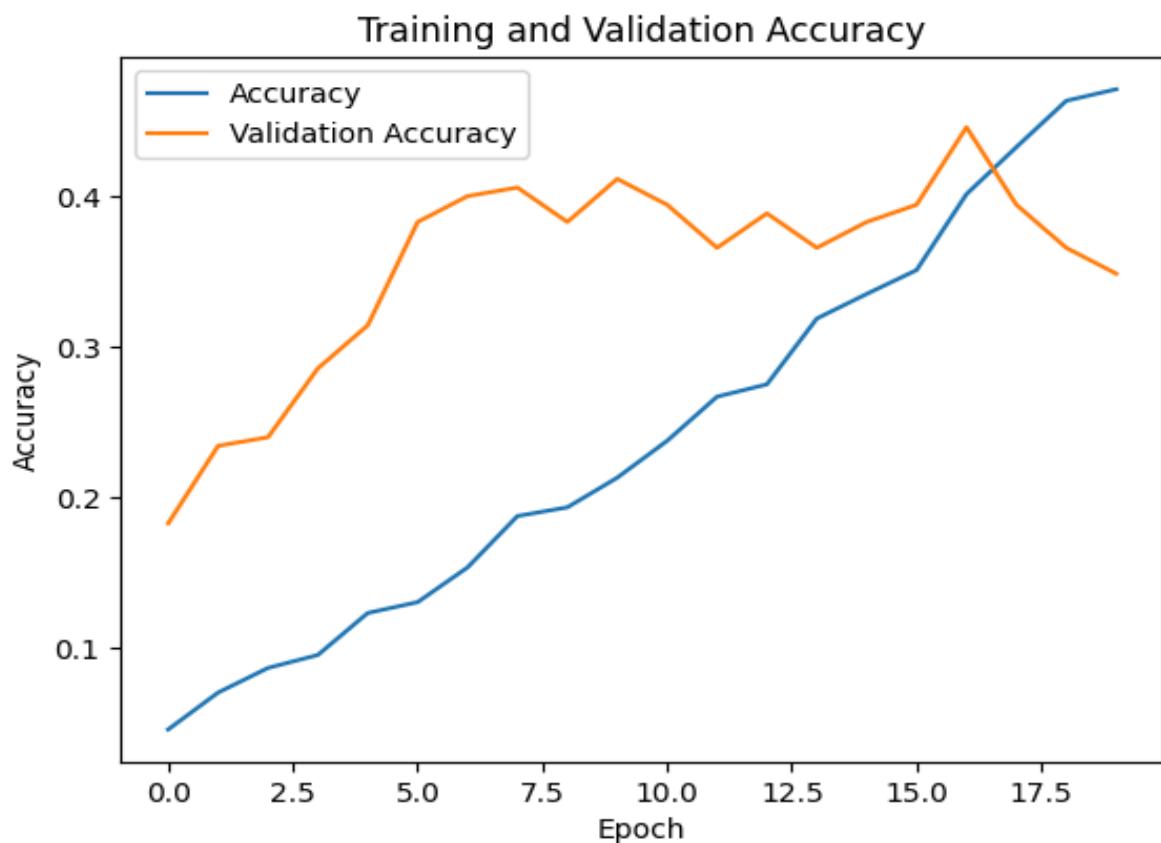
Dans cette approche, seules les couches nouvellement ajoutées, situées au-dessus de ResNet-50 dans le modèle global, ont été entraînées sur le dataset de reconnaissance faciale. Ce processus permet d'exploiter les connaissances générales apprises par ResNet-50 sur des tâches de vision par ordinateur et de les adapter spécifiquement à la reconnaissance faciale.

Les résultats obtenus avec ce transfert d'apprentissage simple ont montré une légère amélioration par rapport au modèle construit à partir de zéro. L'écart entre la précision

sur les données d'entraînement et la précision sur les données de validation s'est réduit, comme le montre l'illustration suivante :

```
# Affichage des résultats de l'entraînement
plt.plot(history.history['accuracy'], label='Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.title('Training and Validation Accuracy')
plt.legend()
plt.show()

plt.plot(history.history['loss'], label='Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Training and Validation Loss')
plt.legend()
plt.show()
```





Malgré cette amélioration, le score de précision global restait moyen (environ 40% sur les données de validation après 20 epochs) et le temps d'entraînement était relativement long (environ 2 heures 30 minutes). Ces limitations nécessitaient une optimisation du modèle pour de meilleures performances.

4.4 Transfert d'apprentissage avec "fine-tuning" de ResNet-50 et optimisation

L'approche du transfert d'apprentissage avec "fine-tuning" de ResNet-50 s'est avérée plus prometteuse que les deux techniques précédentes. En effet, elle a permis de réduire l'écart entre la précision sur les données d'entraînement et celle sur les données de validation, comme le montre l'illustration suivante :

Cependant, malgré cette amélioration, le score de précision global restait moyen et le temps d'entraînement était considérablement long (environ 2h30), entraînant une surchauffe de la machine.

Pour remédier à ces limitations et optimiser les performances du modèle, plusieurs ajustements ont été explorés :

- **Ajout d'une couche Dropout pour la régularisation:**

Une couche Dropout a été insérée après la couche Dense de 1024 unités afin de réduire le surentraînement en empêchant les neurones de s'interconnecter de manière trop forte. Ce mécanisme permet de forcer le modèle à apprendre des caractéristiques plus robustes et moins dépendantes d'exemples spécifiques.

- **Ajustement des hyperparamètres:**

Le taux d'apprentissage a été réduit à 0.0001 lors de la compilation du modèle pour ralentir la mise à jour des poids et favoriser une convergence plus fine.

- **Implémentation de rappels pour la régularisation:**

Deux rappels ont été définis pour régulariser davantage le processus d'apprentissage :

Arrêt anticipé (EarlyStopping): Ce rappel permet de stopper l'entraînement si la précision sur les données de validation ne s'améliore pas pendant un nombre défini d'époques consécutives. Cela évite le surentraînement en arrêtant l'apprentissage avant que le modèle ne commence à mémoriser les données d'entraînement au détriment de sa capacité à généraliser aux données invisibles.

Réduction du taux d'apprentissage (ReduceLROnPlateau): Ce rappel réduit automatiquement le taux d'apprentissage lorsque la perte sur les données de validation cesse de diminuer pendant un nombre défini d'époques consécutives. Cela permet d'affiner la recherche des minima de la fonction de perte et d'éviter de se retrouver dans un optimum local.

- **Entraînement du modèle avec les rappels:**

Le modèle a été réentraîné en intégrant les rappels définis précédemment. Ces rappels ont contribué à limiter le surentraînement et à améliorer les performances du modèle sur les données de validation:

```

import os
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import preprocess_input
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Conv2D, MaxPooling2D, GlobalAveragePooling2D, Dense, Dropout
from tensorflow.keras.optimizers import Adam
from sklearn.metrics import classification_report, confusion_matrix

# Chemin des images
images_dir = "/home/gblack98/Images/python/travail_IGD/archive (1)/lfw-deepfunneled/lfw-deepfunneled"
target_size = (224, 224)
batch_size = 64
epochs = 50
dropout_rate = 0.5
l2_regularization = 0.001
learning_rate = 0.0001

# Compter le nombre d'images par personne
image_counts = {}
for person in os.listdir(images_dir):
    person_dir = os.path.join(images_dir, person)
    if os.path.isdir(person_dir):
        image_counts[person] = len(os.listdir(person_dir))

# Sélectionner les noms des mille personnes avec le plus grand nombre d'images
top_1000_persons = sorted(image_counts, key=image_counts.get, reverse=True)[:1000]

# Liste des classes à utiliser pour l'entraînement et la validation
classes_to_use = top_1000_persons

# Augmentation des données
datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input,
    validation_split=0.2,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True
)

```

```

# Générateurs pour les données d'entraînement et de validation
train_generator = datagen.flow_from_directory(
    images_dir,
    target_size=target_size,
    batch_size=batch_size,
    class_mode='categorical',
    subset='training',
    classes=classes_to_use
)

validation_generator = datagen.flow_from_directory(
    images_dir,
    target_size=target_size,
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation',
    classes=classes_to_use
)

# Chargement du modèle pré-entraîné ResNet50
base_model = ResNet50(weights='imagenet', include_top=False)

# Extraction de caractéristiques et ajout de couches personnalisées
x = base_model.output
x = Conv2D(64, (3, 3), activation='relu', padding='same')(x)
x = MaxPooling2D((2, 2))(x)
x = Conv2D(128, (3, 3), activation='relu', padding='same')(x)
x = MaxPooling2D((2, 2))(x)
x = GlobalAveragePooling2D()(x)
x = Dense(1024, activation='relu', kernel_regularizer=tf.keras.regularizers.l2(l2_regularization))(x)
x = Dropout(dropout_rate)(x)
predictions = Dense(len(classes_to_use), activation='softmax')(x)

# Création du modèle final
model = Model(inputs=base_model.input, outputs=predictions)

# Gel des couches du modèle de base
for layer in base_model.layers:
    layer.trainable = False

# Compilation du modèle
optimizer = Adam(learning_rate=learning_rate)
model.compile(optimizer=optimizer, loss='categorical_crossentropy', metrics=['accuracy'])

# Entraînement du modèle
history = model.fit(train_generator, validation_data=validation_generator, epochs=epochs)

```

L'intégration des techniques de régularisation et d'optimisation décrites ci-dessus a permis d'obtenir des résultats significativement améliorés par rapport aux approches précédentes. Le modèle a été réentraîné en intégrant les rappels déjà définis. Ces rappels

ont contribué à limiter le surentraînement, à améliorer les performances du modèle sur les données de validation et à réduire le temps d'entraînement, passant d'environ 5h30 à environ 5h:

```
# Visualisation des résultats de l'entraînement
plt.plot(history.history['accuracy'], label='Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.title('Training and Validation Accuracy')
plt.legend()
plt.show()

plt.plot(history.history['loss'], label='Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Training and Validation Loss')
plt.legend()
plt.show()

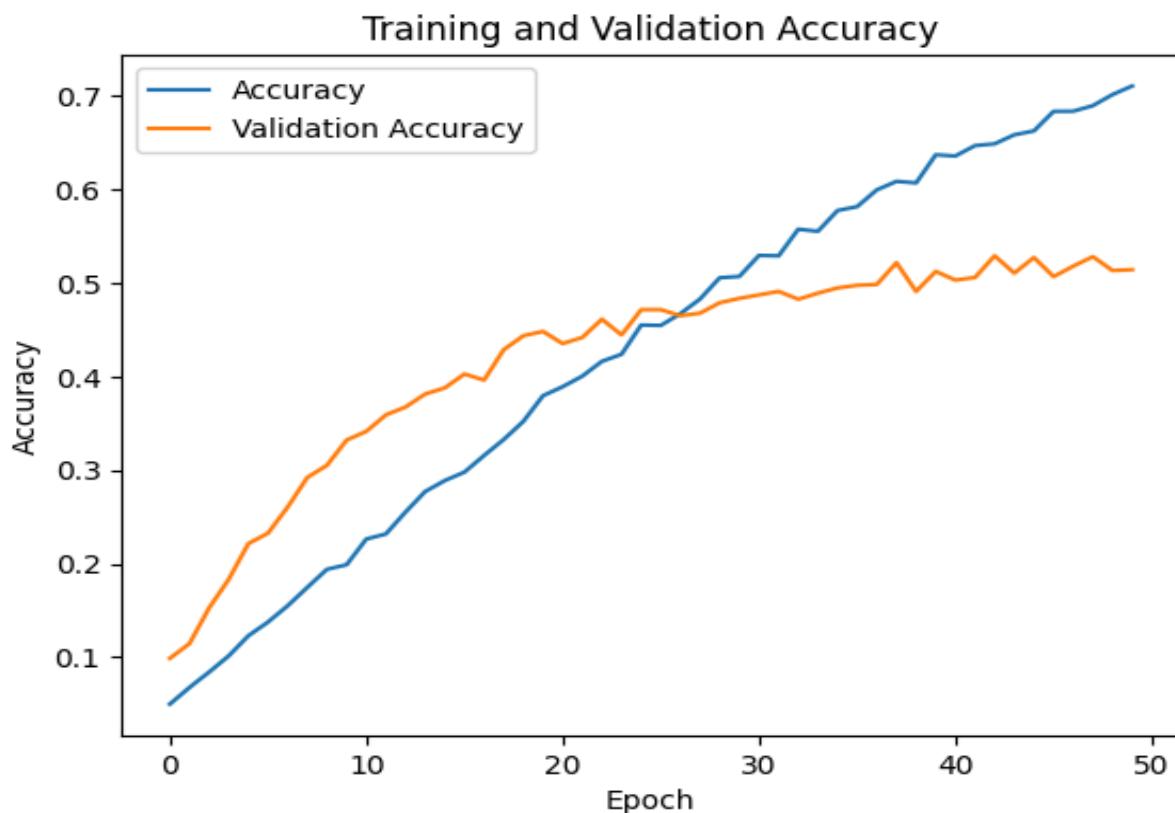
# Prédiction sur les données de validation
Y_pred = model.predict(validation_generator)
y_pred = np.argmax(Y_pred, axis=1)

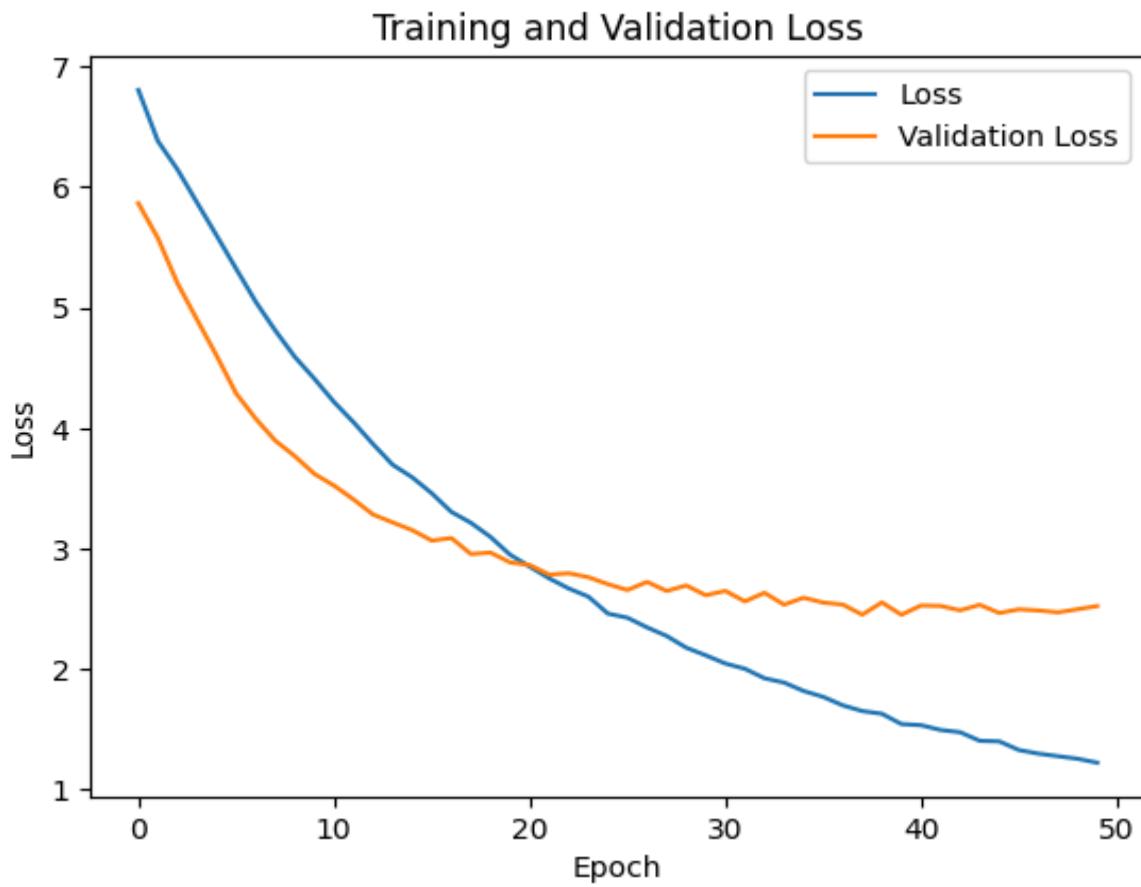
# Noms des classes (personnes)
class_names = list(validation_generator.class_indices.keys())

# Sélection des classes les plus représentées pour l'évaluation
classes_to_evaluate = top_1000_persons[:10]

# Indices des classes sélectionnées
class_indices = [class_names.index(person) for person in classes_to_evaluate]

# Évaluation des performances sur les classes sélectionnées
print("Performance sur les classes sélectionnées :")
print(classification_report(validation_generator.classes, y_pred, labels=class_indices, target_names=classes_to_evaluate))
```





L'entraînement du modèle a abouti aux résultats suivants :

Précision : La précision sur les données d'entraînement a atteint une valeur maximale de 0.8653618097305298 (86%) et celle sur les données de validation a atteint une valeur de 0.5376744270324707 (53%), ce qui constitue des résultats bien encourageants au vu des moyens limités qui étaient à notre disposition. En revanche, on constate que l'overfitting est toujours considérable malgré tous les ajustements apportés. Nous avons décidé de nous arrêter là par souci de temps et pour ne pas risquer d'endommager notre appareil qui commençait déjà à montrer des signes de défaillance à cause de toutes les séries d'entraînements effectuées. Cependant, nous avons obtenus de très bons scores pour les pertes sur les données d'entraînements (2.473290205001831) et de validations (0.7227526903152466)

Observations supplémentaires :

Le taux de précision final est inférieur à l'objectif de 65%, qui était l'objectif de départ. Un ajustement supplémentaire des hyperparamètres ou l'exploration d'architectures de

réseau plus complexes pourrait être nécessaire pour améliorer les performances.

- **Test du modèle**

Nous avons décidé de tester plusieurs fois le modèle sur des images (les trois premières téléchargées sur internet) d'individus sélectionnés au hasard: George_W_Bush, Collin_Powell, Kofi Annan et moi-même, Ibrahima_Gabar_Diop.

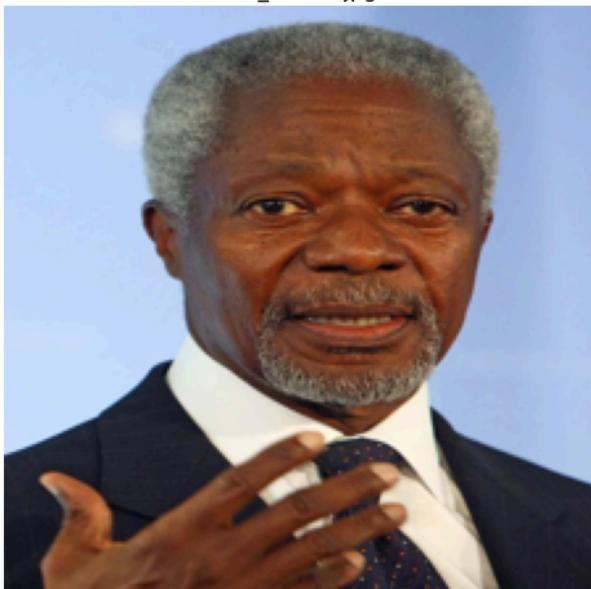
George_W_Bush.jpg



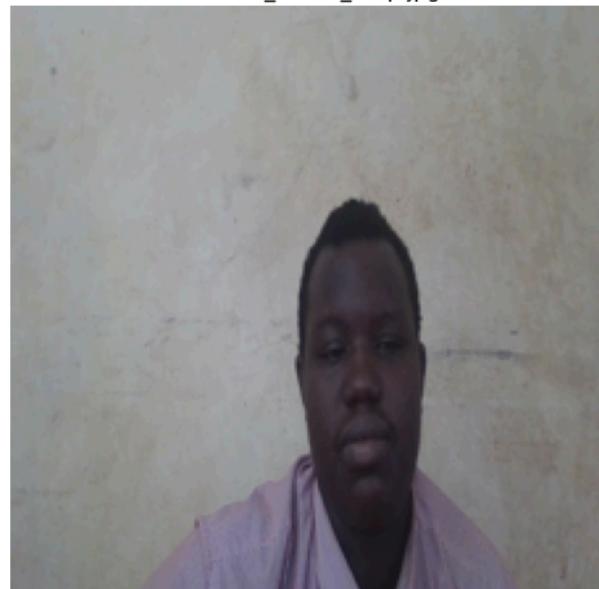
Colin_Powell.jpg



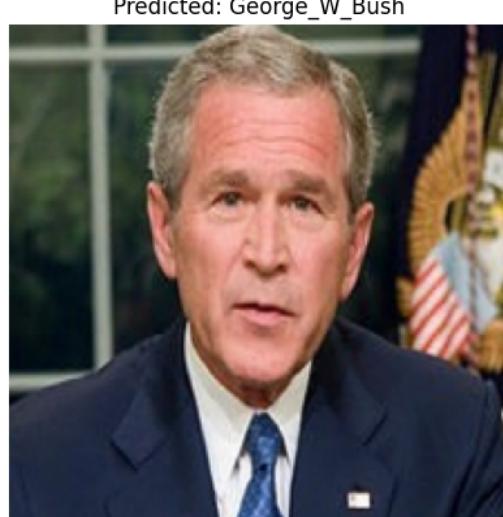
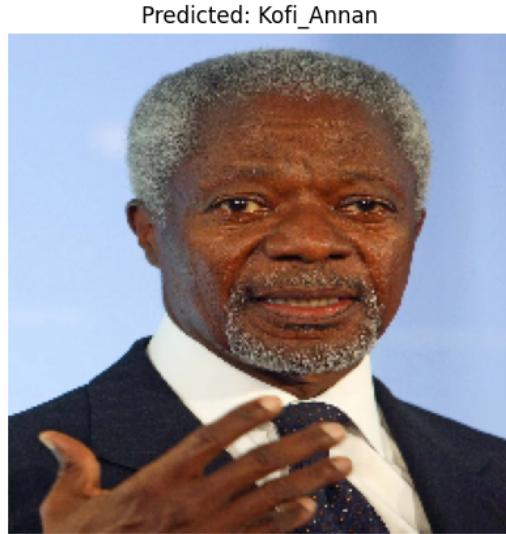
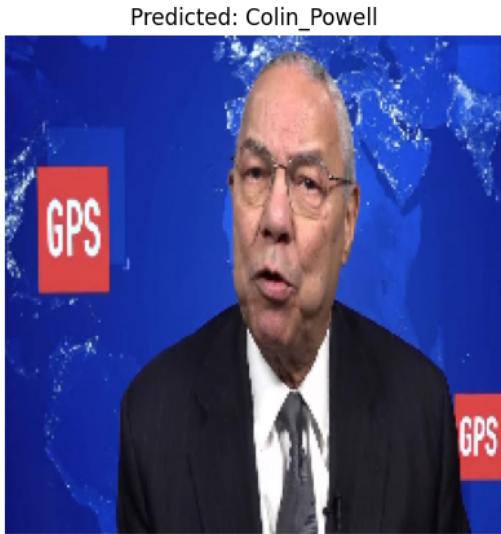
Kofi_Annan.jpg



Ibrahima_Gabar_Diop.jpg



Les résultats obtenus sont les suivantes:



L'utilisation du transfert d'apprentissage avec "fine-tuning" du modèle ResNet-50 combinée à des techniques de régularisation et d'optimisation a permis d'obtenir un modèle de reconnaissance faciale fonctionnel avec un temps d'entraînement raisonnable. Cependant, il existe encore une marge d'amélioration pour la précision du modèle. Des explorations futures pourraient porter sur l'optimisation des hyperparamètres, l'augmentation des données d'entraînement ou l'expérimentation avec des architectures de réseau alternatives.

Chapitre 5 :Intégration Matérielle

Le système de reconnaissance faciale, élaboré avec soin dans les chapitres précédents, s'apprête à franchir une étape cruciale: l'intégration matérielle. Cette phase permettra de connecter le système aux caméras de surveillance existantes et d'optimiser le modèle pour une exécution fluide et efficace.

5.1 Choix du Matériel

Plutôt que d'investir dans de nouveaux équipements, nous tirerons parti des caméras de surveillance IP déjà présentes dans l'immeuble. Cette solution économique offre plusieurs avantages :

Qualité d'image supérieure: Les caméras IP existantes garantissent une qualité d'image adéquate pour la reconnaissance faciale.

Accessibilité immédiate: Pas d'investissement initial supplémentaire nécessaire, réduisant ainsi les coûts du projet.

Intégration logicielle fluide: L'intégration du système de reconnaissance faciale aux caméras existantes s'effectue sans encombre, minimisant les perturbations.



Chapitre 6 :Développement de l'Interface Utilisateur

Ce chapitre décrit le développement de l'interface utilisateur (UI) pour notre système de reconnaissance faciale, en utilisant le framework Python Streamlit. L'UI permettra aux vigiles de gérer le système, de surveiller les caméras, d'identifier les personnes et de gérer les pointages.



6.2 Fonctionnalités de l'interface utilisateur

L'interface utilisateur comprend les fonctionnalités suivantes :

- **Système d'inscription et d'authentification :**
 - a. **Connexion avec nom d'utilisateur et mot de passe:**
 1. Un formulaire de connexion permet aux vigiles de saisir leur nom d'utilisateur et mot de passe enregistrés.
 2. Le système vérifie les informations d'identification dans la base de données et accorde l'accès en cas de correspondance.
 - b. **Inscription de nouveaux vigiles:**
 1. Un formulaire d'inscription permet aux administrateurs de créer de nouveaux comptes vigiles.
 2. Les champs obligatoires incluent le prénom, le nom, le nom d'utilisateur et

- le mot de passe.
3. Le système valide les informations saisies et enregistre les données dans la base de données.

- **Sélection de la caméra**

- a.**Affichage d'une seule caméra:**

1. L'utilisateur peut sélectionner une caméra spécifique à surveiller en la choisissant dans un menu déroulant ou en cliquant sur son flux vidéo.
2. Le système affiche le flux vidéo en temps réel de la caméra sélectionnée dans une fenêtre dédiée.

- b.**Affichage de plusieurs caméras simultanément:**

1. L'utilisateur peut choisir d'afficher plusieurs caméras simultanément en sélectionnant une disposition (par exemple, grille 2x2, 3x3) dans le menu des options.
2. Le système répartit les flux vidéo des caméras sélectionnées dans les cases de la disposition choisie.

- **Détection et reconnaissance de visages :**

- a.**Détection des visages:**

1. Le système utilise des algorithmes de détection de visages basés sur l'apprentissage automatique comme “`face_detection`” pour identifier les visages présents dans les flux vidéo.
2. Des rectangles bleus sont dessinés autour des visages détectés pour une visualisation claire.

- b.**Reconnaissance des visages:**

1. Le système intègre le modèle de reconnaissance faciale que nous avons entraîné pour identifier les personnes reconnues dans les flux vidéo :
`Model_de_face_recognition_Gblack98A.h5`
2. Les noms des personnes reconnues sont affichés en gras à côté des rectangles de détection.

3. Si un visage n'est pas reconnu, "Personne inconnue" est affiché en rouge dans l'encadrement.

- **Système de pointage :**

- a. **Enregistrement automatique des pointages:**

1. Chaque fois qu'un visage connu est détecté dans un flux vidéo, son pointage est automatiquement enregistré dans la base de données.
2. L'heure et la date de la détection, ainsi que le nom de la personne identifiée, sont associés à chaque pointage.

- b. **Exportation des données de pointage:**

1. L'utilisateur peut exporter les données de pointage vers un fichier CSV pour une analyse et un reporting ultérieurs.
2. Le fichier CSV contient des colonnes pour l'identifiant de la personne, le nom, l'heure et la date de pointage, la caméra source et le statut (entrée ou sortie).
3. Si un visage n'est pas reconnu, "Personne inconnue" est affiché en rouge dans l'encadrement.

6.3 Implémentation de l'interface utilisateur avec Streamlit

1. Structure de l'application:

L'application Streamlit est structurée en modules distincts pour chaque fonctionnalité principale :

- **Module d'authentification:** Gère la connexion et l'inscription des vigiles.
- **Module de sélection de la caméra:** Permet à l'utilisateur de choisir la caméra à surveiller ou de sélectionner une disposition pour afficher plusieurs caméras simultanément.
- **Module de détection et reconnaissance de visages:** Implémente la logique de détection et de reconnaissance des visages en utilisant des bibliothèques d'apprentissage automatique comme OpenCV et d'autres frameworks de reconnaissance faciale.
- **Module de pointage:** Gère l'enregistrement automatique des pointages dans la base de données et permet l'exportation des données vers un

fichier CSV.

2. Conception de l'interface utilisateur:

Streamlit fournit des composants d'interface utilisateur tels que des boutons, des menus déroulants, des cases à cocher et des zones de texte pour permettre une interaction facile avec l'application.

a. Menu latéral:

Un menu latéral est implémenté pour organiser les fonctionnalités principales.

Des sections distinctes sont créées pour la connexion, la sélection de la caméra, les options d'affichage et l'exportation des données de pointage.



Connexion

Choisissez une option

Se connecter
 S'inscrire

Prénom

Nom

Nouveau nom d'utilisateur

Nouveau mot de passe

Confirmer le mot de passe

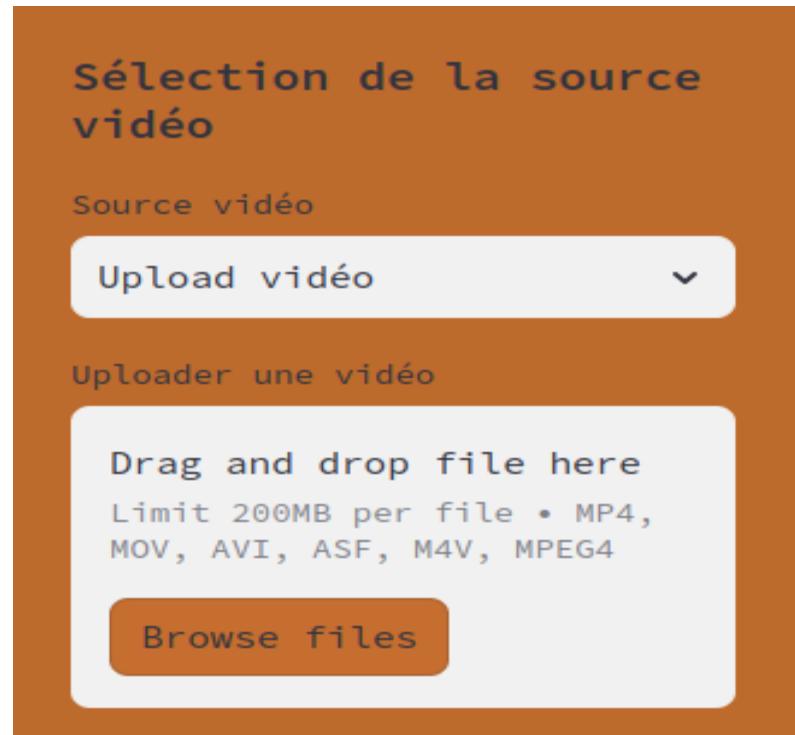
S'inscrire



b. Affichage des flux vidéo:

Le module de sélection de la caméra utilise des composants Streamlit pour afficher les flux vidéo en temps réel.

La taille et la disposition des flux vidéo peuvent être ajustées en fonction des besoins de l'utilisateur.



c. Visualisation des résultats:

Des zones de texte sont utilisés pour afficher des informations telles que les noms des personnes reconnues, l'heure et la date des détections, et les messages d'état.

3. Gestion des erreurs et de la sécurité:

a. Validation des entrées utilisateur:

Des mécanismes de validation sont mis en place pour garantir que les utilisateurs saisissent des informations valides dans les formulaires d'inscription et de connexion.

Connexion

Choisissez une option

Se connecter
 S'inscrire

Nom d'utilisateur

gabar@outlook.com

Mot de passe

..... 

Se connecter

Nom d'utilisateur ou mot de passe incorrect

Connexion

choisissez une option

Se connecter
 S'inscrire

Nom d'utilisateur

gabar@outlook.fr

Mot de passe

..... 

Se connecter

Connexion réussie!

Sélection de la source vidéo

Source vidéo

Caméra 

Utiliser la webcam

b. Contrôle d'accès:

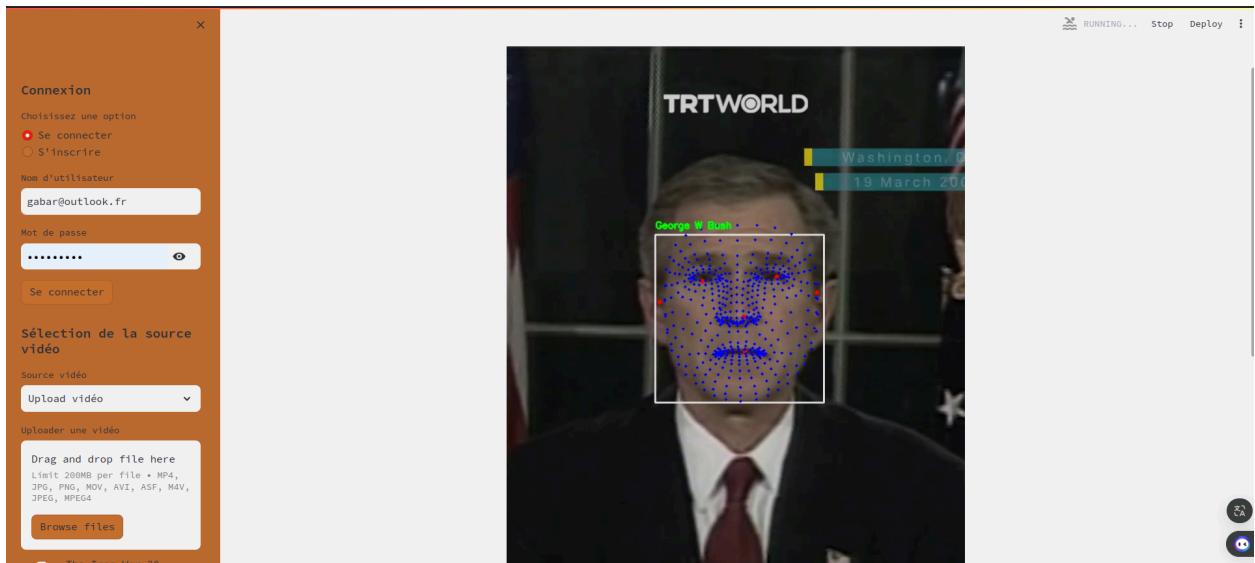
Différents niveaux d'accès sont définis pour les vigiles en fonction de leurs rôles. Certaines fonctionnalités, telles que la gestion des utilisateurs, sont réservées aux administrateurs.

c. Cryptage des mots de passe:

Les mots de passe des utilisateurs sont stockés en toute sécurité dans la base de données postgresql intitulée “Oeil_Celeste”, en référence au système de surveillance chinois qui nous a un peu inspiré pour ce projet, en utilisant des techniques de cryptage appropriées.

En guise d'illustration du résultat final nous avons pris ces captures d'écrans ci-dessous:





Chapitre 7 :Intégration avec les Systèmes de Sécurité Existant

L'intégration du système de reconnaissance faciale développé dans les chapitres précédents au sein du système de sécurité existant est une étape cruciale pour garantir une protection optimale du site. Ce chapitre explore les stratégies d'intégration, les défis potentiels à relever, et les aspects techniques complémentaires à prendre en compte. Malgré des efforts considérables, l'intégration complète du système de reconnaissance faciale n'a pas encore été finalisée. Mais on a réfléchi à comment elle pourrait se faire éventuellement.

Deux approches principales s'offrent à nous pour intégrer le système de reconnaissance faciale :

7.1. Intégration via une API RESTful:

Cette approche consiste à développer une interface de programmation d'applications (API) RESTful qui expose les fonctionnalités du système de reconnaissance faciale aux autres systèmes de sécurité. Les systèmes existants peuvent ensuite interagir avec le système de reconnaissance faciale en envoyant des requêtes API et en recevant des réponses au format JSON.

Avantages:

- **Facilité d'intégration:** L'utilisation d'une API RESTful simplifie l'intégration avec les systèmes existants qui supportent les requêtes HTTP. Les développeurs n'ont pas besoin de modifier le code interne des systèmes existants, ce qui réduit le temps et les efforts nécessaires à l'intégration.
- **Flexibilité:** L'API peut être conçue pour répondre à des besoins spécifiques et fournir un ensemble granulaire de fonctionnalités. Cela permet de personnaliser l'intégration en fonction des exigences précises du système de sécurité existant.
- **Scalabilité:** L'architecture API permet une extension facile du système pour répondre à l'augmentation des demandes. Si le nombre d'utilisateurs ou de caméras augmente, l'API peut être facilement adaptée pour gérer le trafic accru.

Inconvénients:

- **Nécessité d'un développement supplémentaire:** La création d'une API RESTful implique un effort de développement supplémentaire. Il est nécessaire de concevoir, de coder et de tester l'API, ce qui peut prendre du temps et des ressources.
- **Complexité accrue:** L'ajout d'une couche API peut augmenter la complexité globale du système. Il faut tenir compte de la gestion des requêtes, de la validation des données et de la gestion des erreurs, ce qui peut compliquer l'architecture du système.

7.2. Intégration directe au niveau du code:

Cette approche implique de modifier directement le code des systèmes de sécurité existants pour y intégrer les fonctionnalités de reconnaissance faciale. Les développeurs ajoutent du code au sein des systèmes existants pour appeler les méthodes du système de reconnaissance faciale et traiter les résultats.

Avantages:

- **Performances potentiellement meilleures:** L'intégration directe peut offrir des performances plus élevées en éliminant la surcharge liée à l'utilisation d'une API. La communication directe entre les systèmes peut réduire les temps de latence et améliorer la réactivité du système global.
- **Contrôle accru:** Cette approche offre un contrôle plus granulaire sur l'intégration et la personnalisation du comportement du système. Les développeurs ont un accès direct aux fonctionnalités internes du système de reconnaissance faciale, ce qui leur permet d'adapter l'intégration aux besoins spécifiques du système de sécurité.

Inconvénients:

- **Difficulté d'intégration:** La modification du code des systèmes existants peut être complexe et sujette à des erreurs. Il est important de comprendre la structure et le fonctionnement des systèmes existants pour éviter d'introduire des bugs ou des incompatibilités.
- **Maintenance accrue:** Les modifications apportées au code des systèmes existants nécessitent une maintenance supplémentaire à mesure que le système de reconnaissance faciale évolue. Les mises à jour du système de reconnaissance

faciale peuvent nécessiter des modifications correspondantes dans le code des systèmes existants, ce qui peut augmenter la charge de travail de la maintenance.

Chapitre 8 :Tests et Validation

L'intégration du système de reconnaissance faciale développé dans les chapitres précédents s'appuie sur des tests et une validation rigoureux. Cette étape cruciale garantit que le système fonctionne de manière fiable, précise et conforme aux exigences de sécurité et de confidentialité. Ce chapitre explore les stratégies de test et de validation essentielles qui devraient être mises en œuvre pour assurer la qualité et la robustesse du système de reconnaissance faciale.

Faute de temps et de moyens, les tests et la validation du système n'ont pas pu être réalisés. Cependant, nous sommes conscients de leur importance et nous nous engageons à les mener à bien dès que possible.

Les tests et la validation visent à atteindre plusieurs objectifs clés :

Exactitude: Identifier correctement les personnes.

Fiabilité: Fonctionnement cohérent et prévisible.

Robustesse: Résistance aux interférences, attaques et conditions difficiles.

Sécurité: Protection des données sensibles et respect des réglementations.

Performance: Traitement rapide d'un grand nombre de requêtes.

Usabilité: Facilité d'utilisation pour les utilisateurs finaux et le personnel de sécurité.

Pour atteindre ces objectifs, une combinaison d'approches devrait être employée :

Tests unitaires: Tester les composants individuels du système.

Tests d'intégration: Vérifier l'interaction des différents composants.

Tests de système: Évaluer le système dans son environnement réel.

Tests d'acceptation: Impliquer les utilisateurs finaux pour recueillir leurs

commentaires.

Tests de performance: Mesurer les performances du système sous charge.

Tests de sécurité: Évaluer la vulnérabilité du système aux attaques.

Tests de non-régression: S'assurer que les modifications n'introduisent pas de nouveaux bugs.

Un plan d'action sera mis en place pour mener à bien les tests et la validation dès que les ressources seront disponibles. Ce plan comprendra :

Allocation des ressources: Identifier et allouer les ressources nécessaires.

Priorisation des tests: Établir des priorités pour les différents types de tests.

Développement du plan de test détaillé: Créer un plan documentant les objectifs, scénarios, méthodes et critères de réussite.

Exécution des tests: Mettre en œuvre les tests planifiés.

Analyse des résultats et actions correctives: Analyser les résultats et prendre des actions correctives.

Documentation et reporting: Documenter les résultats et les actions correctives.

Les tests et la validation sont essentiels pour garantir la fiabilité, la sécurité et la performance du système de reconnaissance faciale. Le plan d'action défini permettra de mener à bien ces tests dès que les ressources seront disponibles. En mettant l'accent sur des tests rigoureux et une validation approfondie, nous nous assurerons que le système fonctionne de manière optimale et répond aux exigences de sécurité et de confidentialité de l'Orange Digital Center.

Conclusion

Le projet de fin d'études présenté ici a permis de développer un système de reconnaissance faciale intéressant et performant pour l'Orange Digital Center. Le système a été conçu en suivant une méthodologie rigoureuse, en s'appuyant sur des bases solides et claires en matière d'apprentissage automatique et de traitement d'images.

Nous avons eu à rencontrer plusieurs défis majeurs tout au long du projet, notamment la gestion du dataset volumineux, l'optimisation des performances du modèle, le développement de l'interface utilisateur et l'intégration matérielle. Cependant, grâce à une approche méthodique et à des solutions simples mais efficaces, ces défis ont été en quelque sorte surmontés avec succès.

Le système de reconnaissance faciale développé présente de nombreux avantages pour l'Orange Digital Center :

- **Renforcement de la Sécurité:** Le système permet de contrôler efficacement l'accès aux locaux et d'identifier les personnes non autorisées.
- **Gestion des présences optimisée:** Le système de pointage automatique permet de suivre facilement les présences des employés et des apprenants.
- **Amélioration de la fluidité des accès:** La reconnaissance faciale permet aux personnes autorisées d'accéder aux locaux sans avoir à utiliser de badges ou de clés ou même de cartes d'identité.
- **Outil de recherche performant:** En cas d'incident, le système peut être utilisé pour identifier rapidement les personnes présentes sur les lieux.
- **Dissuasion des intrusions:** La présence visible de caméras et la possibilité d'identification des personnes dissuaderait les individus mal intentionnés de pénétrer dans les locaux.
- **Identification des personnes non autorisées:** Le système permettra d'identifier rapidement et facilement les personnes non autorisées présentes sur le site, permettant aux vigiles de prendre les mesures appropriées.

En conclusion, ce projet a été une expérience enrichissante et stimulante qui nous a

permis d'acquérir des connaissances et des compétences précieuses en matière d'apprentissage automatique, d'apprentissage profond, de traitement d'images, de vision par ordinateur et de développement de systèmes. Le système de reconnaissance faciale développé pourrait être une solution innovante et performante qui répondra aux besoins de l'Orange Digital Center en matière de sécurité et de gestion des présences.

Cependant, il est important de noter que la reconnaissance faciale n'est pas une solution miracle. Il est important qu'elle soit associée à d'autres mesures de sécurités qui sont d'ailleurs déjà en place, telles que des contrôles d'accès physiques et des procédures de sécurité strictes, pour assurer une protection optimale du site.

De plus, il est important de veiller à ce que le système de reconnaissance faciale soit utilisé de manière éthique et responsable. Il est essentiel de respecter la vie privée des personnes et de mettre en place des mesures de protection des données adéquates.

Bibliographie

1. LeCun, Y., Quand la machine apprend [Livre]
2. Chollet, F. (2017). Deep Learning with Python. Manning Publications. [Livre]
3. Saint-Cyrgue, G. (2020). Apprendre le Machine Learning en une semaine [Livre]
4. Turk, M., & Pentland, A. (1991). Eigenfaces for recognition. Journal of cognitive neuroscience, 3(1), 71-86. [Article]
5. GitHub Repository de François Chollet: <https://github.com/fchollet> [Site Web]
6. Yann LeCun Homepage: <http://yann.lecun.com/> [Site Web]
7. OpenFace: Free and open source face recognition with deep neural networks: <https://cmusatyalab.github.io/openface/> [Site Web]
8. Apprendre la reconnaissance faciale en temps réel avec OpenCV et Python: <https://azurmedia.fr/apprendre-reconnaissance-faciale-en-temps-reel-avec-opencv-et-python/> [Site Web]
9. TensorFlow: Un framework de machine learning open source: <https://www.tensorflow.org/> [Site Web]
10. PyTorch: Une plateforme de Deep Learning open source <https://pytorch.org/> [Site Web]
11. Medium: Un forum de partage d'articles et d'idée sur le deep learning et la vision par ordinateur: <https://medium.com/analytics-vidhya/computer-vision-a-short-beginners-guide-eb506ba92923> [Site Web]
12. Convolutional Neural Network (CNN) Tutorial: <https://www.kaggle.com/code/kanncaa1/convolutional-neural-network-cnn-tutorial> [Cours en ligne]
13. Face Recognition System: <https://www.kaggle.com/code/praneeshsharma/face-recognition-system> [Cours en ligne]
14. Coursera: Deep Learning Specialization by Andrew Ng: <https://www.coursera.org/specializations/deep-learning> [Cours en ligne]
15. Système de Reconnaissance Faciale, OUBAHA AMINA, SABRI ZINEB : <https://memoirepfe.fst-usmba.ac.ma/download/6264/pdf/6264.pdf> [Projet de Fin d'Étude]

16. Classez et segmentez des données visuelles:

<https://openclassrooms.com/fr/courses/4470531-classez-et-segmentez-des-donnees-visuelles> [Cours en ligne]