RÉPUBLIQUE TUNISIENNE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

* * *

UNIVERSITÉ DE SOUSSE

* * *

Institut Supérieur d'Informatique et des Techniques de Communication

Rapport de Stage de Fin d'Études

Pour l'obtention du Diplôme National de Licence

en Sciences de l'Informatique
Spécialité : Informatique et Multimédia

IDENTICORE : SYSTÈME DE VÉRIFICATION D'IDENTITÉ

Réalisé par :
Abdel Aziz ANNABI

Encadré par :
Mme Faten ZHANI (Académique)
M. Youssef NAJJAR (Professionnel)

Année Universitaire : 2024 – 2025

Tél/Fax : +216 73 37 15 71 / +216 73 36 44 11

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université de Sousse
INSTITUT SUPÉRIEUR D'INFORMATIQUE
ET DES TECHNOLOGIES DE COMMUNICATION

Rapport de Stage de Fin d'Études

Présenté en vue de l'obtention du Diplôme de Licence en Sciences de l'Informatique
Spécialité : Informatique et Multimédia

IDENTICORE : SYSTÈME DE VÉRIFICATION D'IDENTITÉ

Réalisé par : Abdel Aziz ANNABI

Année Universitaire 2024 – 2025

INSTITUT SUPÉRIEUR D'INFORMATIQUE ET DES TECHNOLOGIES DE COMMUNICATION (ISITCom)
Tél/Fax : +216 73 37 15 71 / +216 73 36 44 11

Signature de l'encadrante académique : **Signature de l'encadrant professionnel :**

Dédicaces

À mes parents, Pour leur amour infini, leur patience inébranlable et leur soutien moral constant. Vous avez été mes premiers guides, mes premières inspirations et mes plus grands supporters. Votre soutien financier, tout aussi précieux que votre affection, a été essentiel pour me permettre de poursuivre mes rêves. Merci pour avoir toujours cru en moi, même dans les moments les plus difficiles, et pour m'avoir enseigné l'importance de l'effort, du respect et de l'humilité

À mes frères, Pour leur encouragement sans cesse renouvelé, leur présence rassurante et leur amitié sincère. Vous avez été mes premiers camarades, mes complices de toujours, et mes plus grands alliés. Votre énergie et votre soutien ont souvent été ma force, et pour cela, je vous en suis profondément reconnaissant.

À mes amis et collègues, Pour avoir été à mes côtés tout au long de cette aventure, pour les moments de partage, les discussions enrichissantes et les conseils précieux. Vous avez contribué à rendre ce parcours plus agréable et plus inspirant. Votre présence a été un moteur essentiel pour surmonter les défis et avancer avec confiance.

Merci à vous tous pour avoir été mes piliers, mes repères et mes sources de motivation

Remerciement

Je tiens à exprimer ma profonde gratitude à mon encadrante professionnelle, **Madame Mohamed Faten Zhani** , pour son accompagnement précieux, sa disponibilité et ses conseils avisés tout au long de cette expérience. Votre guidance m'a permis de progresser tant sur le plan professionnel que personnel, et je vous remercie sincèrement pour la confiance que vous m'avez accordée.

Je remercie également mon encadrant technique, **Monsieur Yusuf Najjar**, pour son expertise, sa patience et ses orientations techniques précieuses. Vous avez su me guider à chaque étape, en m'aidant à surmonter les défis techniques avec rigueur et persévérance.

Je ne saurais oublier toute **la famille Linqubit** , pour l'accueil chaleureux, l'esprit collaboratif et l'environnement professionnel stimulant que vous m'avez offert. Votre soutien et votre esprit d'équipe ont rendu cette expérience inoubliable et enrichissante.

Merci à vous tous pour votre générosité, votre encouragement et votre inspiration, qui ont été des moteurs essentiels pour la réussite de ce projet.

Table des matières

1 Introduction Générale **1**
1.1 Introduction 1
1.2 Cadre de stage et organisme d'accueil 1
1.3 Problématique 2
1.4 Solution proposé 2
1.5 Méthodologie 2
1.5.1 Méthodologie Kanban 3
1.5.2 Les composants de Kanban 4
1.5.3 Backlog produit du projet 4
1.6 Conclusion 4

2 État de l'art **6**
Introduction 6
2.1 Vision par ordinateur 7
2.2 Apprentissage profond 7
2.2.1 Réseaux de Neurones Convolutifs (CNN) 8
2.2.2 YOLO (You Only Look Once) 9
2.3 Extraction de données 10
2.3.1 EasyOCR 10
2.4 Interaction gestuelle 10
2.4.1 MediaPipe 10
2.5 Revue de littérature 11
2.6 Conclusion 12

3 Spécification des besoins **13**
Introduction 13
3.1 Analyse de besoin 14
3.1.1 Besoins fonctionnels 14

3.1.2 Besoins non fonctionnels 14
3.2 Scénario d'utilisation 15
3.3 Environnement de travail 15
3.3.1 Logiciels et langages de programmation 15
3.3.2 Bibliothèques 15
Conclusion 16

4 Conception et développement 17
Introduction 18
4.1 Architecture 18
4.1.1 Diagramme de flow 18
4.1.2 Architecture frontend/backend 24
4.2 reCAPTCHA Gestuel 24
4.2.1 Détection des gestes avec MediaPipe 24
4.2.2 Gestes supportés 25
4.2.3 Pipeline de Traitement des Gestes 25
4.3 Détection de la carte d'identité (CIN) 26
4.3.1 Prétraitement des données 26
4.3.2 Métriques d'évaluation des modèles 27
4.3.3 Modèle CNN 29
4.3.4 Entraînement du modèle (CNN) 29
4.3.5 Analyse des performances (CNN) 30
4.3.6 Modèle YOLO 33
4.3.7 Entraînement du modèle (YOLO) 33
4.3.8 Analyse des performances (YOLO) 34
4.3.9 Étude comparative 36
4.4 Extraction de texte 36
4.4.1 Problèmes 36
4.4.2 Solutions 37
4.5 Biometrique faciale 40
4.5.1 Détection des émotions 40
4.5.2 Comparaison de visages 41
4.6 Implémentation 42
4.6.1 Frontend 42
4.6.2 Back-end 47

Table des figures

1.1 Illustration de la solution proposée 3
1.2 Tablau Kanabn 3

2.1 Architecture d'un modèle d'apprentissage profond. 8
2.2 Exemple d'architecture typique d'un réseau de neurones convolutif (CNN). . . 9
2.3 Exemple d'architecture typique d'un modèle YOLO. 9

4.1 Diagramme de séquence du système de vérification. 19
4.2 Diagramme de séquence du reCAPTCHA gestuel. 20
4.3 Diagramme de séquence du téléversement et traitement de la CIN. 21
4.4 Diagramme de séquence de la vérification faciale. 23
4.5 Architecture simplifiée frontend/backend du système. 24
4.6 Pipeline de traitement des gestes avec MediaPipe. 26
4.7 Architecture du modèle. 30
4.8 Courbes de précision (*accuracy*) pour l'entraînement et la validation 31
4.9 Courbes de fonction de perte (*loss*) pour l'entraînement et la validation 31
4.10 confusion matrix 32
4.11 Matrice de confusion. 35
4.12 Courbes de fonction de perte pour l'entraînement et la validation du modèle YOLOv8 36
4.13 Détection des champs avec easyOcr 37
4.14 Detection des champs avec YOLO 38
4.15 mAP curves. 38
4.16 Loss curves. 39
4.17 Exemple. 40
4.18 Architecture du modèle. 41
4.19 Pipeline de vérification faciale. 42
4.20 Page d'accueil de l'application. 43
4.21 Interface reCAPTCHA. 44

4.22 Interface de téléversement du recto de la CIN. 45
4.23 Interface de téléversement du verso de la CIN. 45
4.24 Interface de détection des émotions. 46
4.25 Interface de vérification du visage. 47
4.26 Cryptage de données. 50

Liste des tableaux

1.1 Répartition des étapes du projet 5

4.1 Rapport de classification du modèle CNN 32

Chapitre 1

Introduction Générale

Contents

1.1 Introduction **1**

1.2 Cadre de stage et organisme d'accueil **1**

1.3 Problématique **2**

1.4 Solution proposé **2**

1.5 Méthodologie **2**

1.5.1 Méthodologie Kanban 3

1.5.2 Les composants de Kanban 4

1.5.3 Backlog produit du projet 4

1.6 Conclusion **4**

1.1 Introduction

Le premier chapitre de ce rapport présente le cadre global dans lequel s'inscrit le projet. Il vise à contextualiser l'environnement de travail, les motivations de départ, et les étapes méthodologiques adoptées.

1.2 Cadre de stage et organisme d'accueil

Ce projet a été réalisé dans le cadre de l'obtention du Diplôme National de Li cence en Sciences de l'Informatique à l'Institut Supérieur d'Informatique et des Techniques de Communication. Il s'est déroulé sur une période de quatre mois en collaboration avec l'entreprise tunisienne LINQUBIT, spécialisée dans la concep tion et le développement de sites web et d'applications mobiles. Fondée en 2022 et basée à Kairouan, LINQUBIT se distingue par sa capacité à intégrer les dernières tendances technologiques et par son engagement envers la satisfaction client.

L'objectif principal du stage était de concevoir et tester un système automatisé de vérification d'identité basé sur l'intelligence artificielle.

1.3 Problématique

La vérification d'identité est aujourd'hui une composante essentielle dans l'accès aux services numériques. Que ce soit pour ouvrir un compte, signer un contrat ou accéder à des services personnalisés, les utilisateurs doivent prouver leur identité de manière fiable et rapide. Les méthodes les plus courantes reposent généralement sur des vérifications par e mail, SMS ou applications d'authentification. Bien que pratiques, ces approches ne permettent pas toujours de confirmer l'identité réelle de la personne, notamment lorsque le service requiert un niveau de garantie plus élevé. Dans ce contexte, l'intégration de la vérification d'identité par document officiel et reconnaissance faciale apparaît comme une évolution naturelle. Toutefois, ce processus est souvent effectué manuellement, ce qui peut allonger les délais et mobiliser des ressources importantes. D'où la nécessité de réfléchir à une automa tisation intelligente et efficace de cette étape clé

1.4 Solution proposé

Afin de répondre aux besoins d'authentification dans un environnement numé rique de plus en plus exigeant, ce projet propose le développement d'un système automatisé de vérification d'identité basé sur l'intelligence artificielle. L'objectif est de permettre à une plateforme en ligne de traiter, de manière autonome, les documents d'identité soumis par les utilisateurs et d'assurer une reconnaissance faciale rapide et fiable. La solution repose sur l'intégration de plusieurs modules complémentaires : dé tection de carte d'identité via un modèle YOLO, extraction des informations textuelles à l'aide d'un moteur OCR, comparaison d'images faciales grâce à un algorithme de reconnaissance basé sur des embeddings, et ajout d'un module de captation gestuelle pour garantir une interaction en temps réel. Cette approche permet de fluidifier l'expérience utilisateur tout en renforçant la f iabilité du processus de vérification, sans intervention humaine directe.

1.5 Méthodologie

Pour assurer une gestion claire et structurée du projet, une méthode agile a été adoptée. L'objectif principal était de garantir une progression régulière avec des livrables partiels à chaque étape. Parmi les différentes approches agiles existantes, la méthodologie Kanban a été privilégiée en raison de sa flexibilité et de sa sim plicité de mise en œuvre dans un cadre de stage.

Fig. 1.1. Illustration de la solution proposée

1.5.1 Méthodologie Kanban

La méthode Kanban est une approche de gestion de projet visuelle permettant de suivre l'avancement des tâches en temps réel. Elle repose sur un tableau divisé en colonnes représentant les étapes du processus : À faire, En cours, et Terminé. Chaque tâche est représentée par une carte qui circule entre les colonnes en fonc tion de son état d'avancement. Cette méthode offre une vision claire du flux de travail et facilite la détection des blocages

Fig. 1.2. Tablau Kanabn

1.5.2 Les composants de Kanban

La méthodologie kanban dans ce projet se compose de :

1. **Product Owner** : Le Product Owner est responsable de définir les besoins et les exigences du projet du point de vue des utilisateurs et des parties prenantes

2. **Équipe de Développement** : L'équipe de développement est composée de développeurs et de testeurs qui travaillaient ensemble pour réaliser les fonctionnalités et les améliorations du produit.

3. **Coach d'agile** :Le coach agile assiste le Product Owner et l'équipe de développement dans le développement et le maintien des bonnes pratiques.

4. **Backlog Produit** : Un tableau qui contient toutes les tâches, fonctionnalités, améliorations et corrections qui doivent être réalisées pour compléter le projet

5. **Tableau Kanban** : Le tableau Kanban contient les étapes que les tâches doivent suivre

. **Àfaire** : liste initiale des tâches à planifier et exécuter

. **En cours** : tâches actuellement en développement

. **En attente** : tâches en pause ou dépendantes d'une autre

. **Terminé** : tâches finalisées et validées Chaque tâche était associée à une fonctionnalité spécifique du système ou à une étape technique

1.5.3 Backlog produit du projet

1.6 Conclusion

Ce premier chapitre a permis d'introduire le contexte général du projet, en présentant le cadre de stage, l'organisme d'accueil, ainsi que les objectifs à atteindre. Il a également mis en lumière la problématique liée à la vérification d'identité en ligne et a proposé une solution basée sur l'intelligence artificielle. Enfin, la méthodologie adoptée ainsi que les étapes de développement ont été détaillées pour assurer un déroulement structuré du projet. Le chapitre suivant présentera un état de l'art des technologies utilisées dans la mise en œuvre du système.

Étape	**Description**
Recherche et analyse	- Étude des méthodes existantes de vérification d'identité. - Exploration des outils IA (OCR, reconnaissance faciale, MediaPipe). - Définition de la solution technique adaptée.
Spécification des besoins	- Rédaction des besoins fonctionnels et non fonctionnels. - Élaboration des scénarios d'usage. - Prise en compte des contraintes techniques.
Développement	- Développement des modèles d'IA (détection de carte d'identité, reconnaissance faciale, détection gestuelle). - Intégration front-end et back-end. - Connexion entre les modules et les interfaces utilisateurs.
Documentation	- Rédiger la documentation technique du code. - Préparer le rapport final décrivant les étapes du projet.

Tab. 1.1. Répartition des étapes du projet

Chapitre 2

État de l'art

Contents

Introduction **6**

2.1 Vision par ordinateur **7**

2.2 Apprentissage profond **7**

2.2.1 Réseaux de Neurones Convolutifs (CNN) 8

2.2.2 YOLO (You Only Look Once) 9

2.3 Extraction de données **10**

2.3.1 EasyOCR 10

2.4 Interaction gestuelle **10**

2.4.1 MediaPipe 10

2.5 Revue de littérature **11**

2.6 Conclusion **12**

Introduction

Ce chapitre a pour objectif de présenter les concepts et les technologies clés mobilisés dans la réalisation du projet. Il s'agit notamment de l'intelligence artificielle, de la vision par ordinateur, des réseaux de neurones profonds, ainsi que des outils permettant la reconnaissance de texte et la détection gestuelle. Cette revue de l'état de l'art permet de justifier les choix technologiques adoptés par la suite.

Les différents outils étudiés ici incluent des algorithmes comme CNN, YOLO pour la détection d'objets, EasyOCR pour l'extraction de texte, ArcFace pour la reconnaissance faciale, et MediaPipe pour la détection de gestes. Ces solutions sont aujourd'hui largement utilisées dans les systèmes de vérification d'identité modernes et constituent une base solide pour développer une plateforme fiable et automatisée.

2.1 Vision par ordinateur

La vision par ordinateur est une branche de l'intelligence artificielle qui vise à permettre aux machines de comprendre et d'interpréter des informations visuelles à partir d'images ou de vidéos. Elle s'inspire du fonctionnement de la vision humaine, en tentant de reproduire la perception visuelle grâce à des algorithmes capables d'identifier, localiser et analyser des objets dans une scène.
Dans le cadre de ce projet, la vision par ordinateur joue un rôle central. Elle intervient notamment dans la détection automatique de la carte d'identité dans une image, la reconnaissance faciale pour identifier l'utilisateur, ainsi que dans la reconnaissance gestuelle pour valider certaines actions à l'écran. Grâce à l'évolution des modèles d'apprentissage profond, ces tâches sont aujourd'hui réalisées avec un haut degré de précision. Les principales tâches en vision par ordinateur se répartissent en trois catégories : la **classification**, qui consiste à attribuer une étiquette à une image dans son ensemble (ex. : image contenant un visage ou non), la **détection**, qui localise un ou plusieurs objets dans l'image en les entourant de boîtes englobantes, et la **segmentation**, qui associe chaque pixel de l'image à une classe pour délimiter précisément les objets. L'utilisation de frameworks performants comme OpenCV, combinée à des modèles d'apprentissage profonds comme CNN , YOLO ou ArcFace, a considérablement amélioré la capacité des systèmes à traiter des images en temps réel avec efficacité.

2.2 Apprentissage profond

L'apprentissage profond (ou *deep learning*) est une sous-catégorie de l'apprentissage automatique qui repose sur des réseaux de neurones artificiels à plusieurs couches. Ces architectures sont capables de traiter de grandes quantités de données, d'apprendre des représentations complexes et de s'adapter à divers types de tâches comme la classification, la détection ou la reconnaissance.

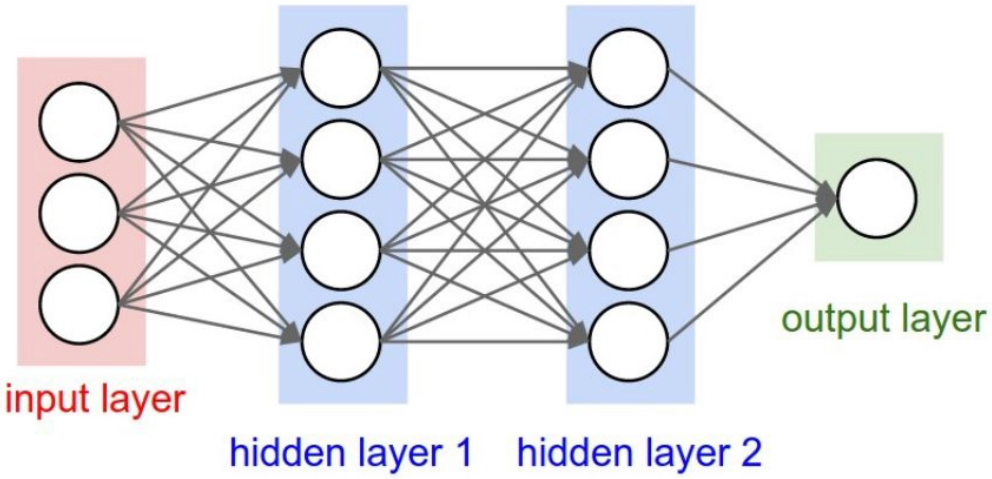

Fig. 2.1. Architecture d'un modèle d'apprentissage profond.

Contrairement aux méthodes classiques reposant sur des règles prédéfinies ou des extractions manuelles de caractéristiques, le deep learning permet à la machine d'apprendre automatiquement les caractéristiques pertinentes à partir des données brutes. Cette approche a révolutionné des domaines comme la vision par ordinateur, le traitement du langage naturel, et plus récemment les systèmes d'authentification biométrique.

2.2.1 Réseaux de Neurones Convolutifs (CNN)

Les réseaux de neurones convolutifs (CNN – *Convolutional Neural Networks*) sont des architectures d'apprentissage profond largement utilisées pour le traitement d'images. Leur force réside dans leur capacité à extraire automatiquement des caractéristiques visuelles à différents niveaux (contours, textures, formes) grâce à l'empilement de couches convolutives.
Un CNN est généralement composé de trois types de couches principales : les couches convolutives (qui appliquent des filtres pour détecter des motifs locaux), les couches de regroupement (*pooling*) qui réduisent la dimension des données, et les couches entièrement connectées pour la classification finale.

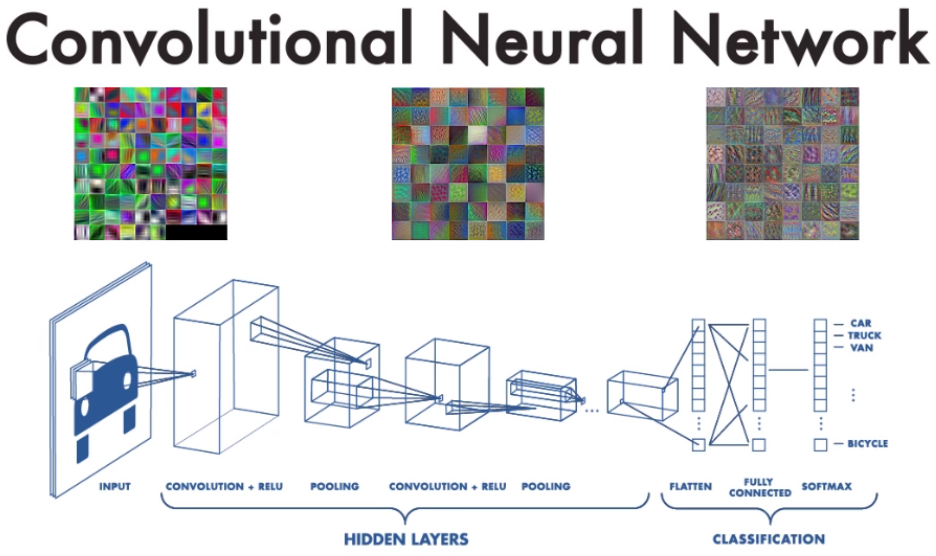

Fig. 2.2. Exemple d'architecture typique d'un réseau de neurones convolutif (CNN).

2.2.2 YOLO (You Only Look Once)

YOLO (pour *You Only Look Once*) est un algorithme de détection d'objets en temps réel basé sur l'apprentissage profond. Contrairement à d'autres approches qui analysent une image en plusieurs étapes, YOLO traite l'image entière en une seule passe, ce qui le rend extrêmement rapide et efficace.

Le principe de YOLO repose sur le découpage de l'image en une grille, où chaque cellule prédit les coordonnées des boîtes englobantes potentielles et leur probabilité d'appartenance à une classe donnée. Cette approche unifiée permet d'atteindre un bon compromis entre précision et rapidité, ce qui la rend idéale pour des applications en temps réel.

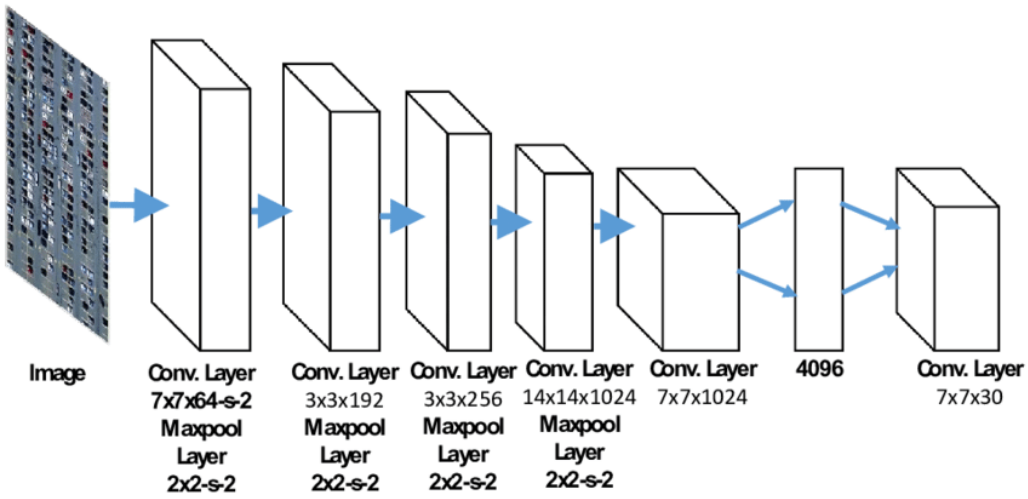

Fig. 2.3. Exemple d'architecture typique d'un modèle YOLO.

2.3 Extraction de données

L'extraction de données est un processus visant à identifier et récupérer des informations pertinentes à partir de sources non structurées, telles que des documents, des images ou des vidéos. Dans le domaine de la vision par ordinateur, cela se traduit souvent par l'identification de zones contenant du texte ou des éléments visuels, puis leur conversion en données exploitables.
Les techniques d'extraction s'appuient généralement sur des méthodes de détection (pour localiser les zones d'intérêt) combinées à des approches de reconnaissance, comme l'OCR (*Optical Character Recognition*), pour transformer l'image en texte numérique. Ces outils permettent par exemple de lire automatiquement des informations sur des documents administratifs, des factures, ou des cartes d'identité.
Grâce aux progrès de l'intelligence artificielle et de l'apprentissage profond, l'extraction de données devient plus rapide, plus précise et adaptable à des documents de formats très variés.

2.3.1 EasyOCR

EasyOCR est une bibliothèque open-source de reconnaissance optique de caractères (OCR) basée sur l'apprentissage profond. Elle est conçue pour extraire du texte à partir d'images ou de documents scannés et prend en charge plus de 80 langues, y compris les langues à écriture complexe comme l'arabe, le chinois ou l'hindi.
Grâce à l'utilisation de réseaux de neurones pré-entraînés, EasyOCR permet d'obtenir des résultats fiables même en présence d'images de faible qualité, de texte incliné ou de bruit visuel. Elle est particulièrement utile pour automatiser la lecture d'informations contenues dans des factures, des formulaires, des panneaux signalétiques ou des documents administratifs.
Facile à intégrer dans des applications Python, cette bibliothèque offre une interface simple et rapide pour convertir des zones de texte en données exploitables, sans nécessiter d'entraînement préalable de modèles personnalisés.

2.4 Interaction gestuelle

L'interaction gestuelle désigne l'ensemble des techniques permettant à un utilisateur de contrôler une application ou d'interagir avec un système informatique à travers des mouvements du corps, sans utiliser d'interface physique classique comme un clavier ou une souris. Ces gestes peuvent être capturés à l'aide de caméras, de capteurs de profondeur, ou de solutions logicielles de vision par ordinateur.

2.4.1 MediaPipe

MediaPipe est un framework open-source développé par Google, dédié à la création de pipelines de traitement multimédia en temps réel. Il permet de construire des solutions performantes pour

l'analyse vidéo, la détection d'objets, la reconnaissance faciale, la détection de gestes ou encore l'analyse de la posture.
L'une des principales forces de MediaPipe réside dans sa capacité à détecter et suivre les mouvements des mains, du visage ou du corps humain avec une grande précision, tout en fonctionnant en temps réel sur des appareils aux ressources limitées comme les smartphones. Il fournit des modèles pré-entraînés et optimisés, intégrables facilement dans des applications Python, Android ou Web.
MediaPipe est particulièrement utilisé dans les interfaces gestuelles, la réalité augmentée, ou encore les systèmes de vérification d'identité renforcée par détection de vivacité. Son architecture modulaire et sa compatibilité multiplateforme en font un outil puissant pour développer des interactions naturelles et intuitives entre l'humain et la machine.

2.5 Revue de littérature

Le projet *Identicore* s'inscrit dans une dynamique technologique actuelle où la vérification d'identité à distance devient une nécessité croissante, notamment dans les domaines de la finance, des plateformes de services en ligne et de l'administration numérique. De nombreuses études récentes ont exploré les méthodes d'authentification automatisées, en s'appuyant sur la vision par ordinateur et les réseaux de neurones profonds.
Concernant la reconnaissance faciale, plusieurs architectures comme FaceNet (Schroff et al., 2015) ou ArcFace (Deng et al., 2019) ont démontré leur efficacité dans la génération d'embeddings faciaux robustes. Ces modèles sont aujourd'hui largement utilisés pour des tâches de comparaison faciale à des fins d'identification ou de vérification. La littérature indique que ces systèmes offrent un taux de précision élevé, même en présence de variations d'éclairage ou d'angle, ce qui les rend adaptés à une utilisation dans des environnements non contrôlés.
Pour la détection de documents et d'éléments visuels, YOLO s'impose comme une référence incontournable. Redmon et al. ont introduit YOLO comme une méthode unifiée de détection rapide, capable d'identifier et de localiser plusieurs objets dans une image en une seule passe. Des travaux ultérieurs ont intégré YOLO dans des pipelines de traitement documentaire pour détecter des cartes d'identité, des numéros de série ou des visages à partir de vidéos en direct, avec d'excellents résultats en temps réel.
Du côté de l'OCR, les solutions traditionnelles comme Tesseract ont été largement dépassées par les approches modernes basées sur l'apprentissage profond. EasyOCR, par exemple, repose sur une architecture de type CRNN (Convolutional Recurrent Neural Network) combinée à un mécanisme d'attention, ce qui lui permet de reconnaître des textes dans des contextes complexes. Plusieurs études ont validé son efficacité pour l'extraction de texte sur des documents officiels, y compris dans des langues à script non latin.
Enfin, l'interaction gestuelle et la preuve de vivacité (*liveness detection*) sont de plus en plus

explorées comme couches de sécurité supplémentaires. MediaPipe, une bibliothèque développée par Google, a été utilisée dans plusieurs projets pour détecter les gestes de la main, les expressions faciales ou la posture en temps réel. Dans le contexte de la vérification d'identité, cette technique permet de s'assurer que l'utilisateur est bien une personne physique présente au moment de la vérification, et non une image ou une vidéo préenregistrée.
En conclusion, les technologies utilisées dans le projet *Identicore* s'appuient sur des fondations solides et éprouvées dans la littérature. L'intégration de ces outils permet d'élaborer un système de vérification d'identité complet, intelligent et automatisé, répondant aux besoins actuels en matière de sécurité et d'expérience utilisateur.

2.6 Conclusion

Ce chapitre a présenté les principales technologies utilisées dans le projet, notamment la vision par ordinateur, la reconnaissance faciale, l'OCR et l'interaction gestuelle. Ces outils, issus de recherches récentes, constituent la base du système. Le chapitre suivant décrira les besoins à satisfaire pour la mise en œuvre de la solution.

Chapitre 3

Spécification des besoins

Contents

Introduction **13**
3.1 Analyse de besoin **14**
3.1.1 Besoins fonctionnels 14
3.1.2 Besoins non fonctionnels 14
3.2 Scénario d'utilisation **15**
3.3 Environnement de travail **15**
3.3.1 Logiciels et langages de programmation 15
3.3.2 Bibliothèques 15
Conclusion **16**

Introduction

La réussite d'un projet technologique repose sur une définition claire et précise des besoins à satisfaire. Ce chapitre vise à formaliser les attentes fonctionnelles et techniques du système de vérification d'identité proposé, en s'appuyant sur les objectifs identifiés lors de la phase d'analyse.

Il s'agira d'identifier les fonctionnalités principales que le système doit offrir, mais aussi de prendre en compte les contraintes non fonctionnelles comme la sécurité, la fluidité de l'expérience utilisateur, et la compatibilité avec les plateformes mobiles. Cette étape permet de poser les fondations du développement à venir en garantissant que la solution réponde aux exigences du contexte réel.

3.1 Analyse de besoin

3.1.1 Besoins fonctionnels

Les besoins fonctionnels se concentrent sur les capacités spécifiques que le système doit posséder. Ainsi, nous avons identifié les besoins suivants :

- **Test anti-bot (reCAPTCHA gestuel)** : vérification que l'utilisateur est humain en lui demandant d'afficher un chiffre avec ses doigts devant la caméra.
- **Téléversement de la carte d'identité** : importation du recto et du verso de la carte d'identité nationale pour permettre l'analyse.
- **Extraction des données textuelles** : lecture automatique des informations présentes sur la carte à l'aide de l'OCR.
- **Vérification d'émotion (liveness check)** : demande à l'utilisateur d'afficher un sourire pour confirmer sa présence réelle.
- **Capture d'une photo avec la carte** : prise d'une photo montrant l'utilisateur tenant sa carte pour valider la correspondance.
- **Vérification faciale biométrique** : comparaison du visage capturé en direct avec celui figurant sur la carte d'identité.
- **Retour de l'état de vérification** : notification à l'utilisateur sur le résultat de la vérification.

3.1.2 Besoins non fonctionnels

Les besoins non fonctionnels définissent les critères de qualité attendus du système, au-delà des fonctionnalités principales. Ils garantissent la performance, la sécurité et la bonne expérience utilisateur du système.

- **Fiabilité** : le système doit fonctionner de manière stable, sans erreurs fréquentes ni interruptions durant les vérifications.
- **Performance** : les étapes de vérification doivent s'exécuter rapidement, avec un temps de réponse acceptable pour l'utilisateur.
- **Sécurité des données** : toutes les données personnelles (images, textes extraits) doivent être protégées contre les accès non autorisés.
- **Accessibilité** : l'application doit être simple d'utilisation, même pour des utilisateurs ayant peu d'expérience technique.

- **Compatibilité mobile** : le système doit être utilisable sur des smartphones (Android et iOS), avec une interface adaptée.

3.2 Scénario d'utilisation

Le scénario d'utilisation du système commence par une vérification anti-bot où l'utilisateur doit effectuer un geste demandé à l'écran, comme montrer un chiffre avec ses doigts. Ensuite, il est invité à téléverser le recto et le verso de sa carte d'identité nationale. Le système procède alors à l'extraction automatique des données textuelles via un module OCR. Pour s'assurer que l'utilisateur est bien présent physiquement, une vérification d'émotion est déclenchée, lui demandant d'afficher un sourire. Enfin, une photo est prise avec la carte d'identité visible, permettant une comparaison faciale entre le visage détecté en temps réel et celui présent sur la carte. Le système analyse l'ensemble des informations et fournit à l'utilisateur un retour sur la réussite ou l'échec de la vérification.

3.3 Environnement de travail

3.3.1 Logiciels et langages de programmation

— **Python** : Python est un langage de programmation interprété, simple et polyvalent.

— **React Native** : React Native est un framework open-source qui permet de créer des applications mobiles natives en utilisant JavaScript et React

— **Expo** : Expo est une plateforme et un ensemble d'outils pour simplifier le développement d'applications React Native.

— **Visual Studio Code** :est un éditeur de code source léger, gratuit et multiplateforme développé par Microsoft..

— **Google Colab / Jupyter Notebook** : plateformes de prototypage et d'entraînement pour les modèles IA.

— **Roboflow** : plateforme en ligne permettant de gérer, annoter et préparer des datasets d'images pour l'entraînement de modèles de vision par ordinateur.

3.3.2 Bibliothèques

— **EasyOCR** : bibliothèque Python open-source pour la reconnaissance de texte dans des images via OCR (reconnaissance optique de caractères).

- **DeepFace** : bibliothèque Python open-source pour la reconnaissance et l'analyse faciale à l'aide de deep learning.
- **ArcFace** : méthode de reconnaissance faciale basée sur les réseaux de neurones qui optimise la séparation angulaire pour améliorer la précision.
- **MediaPipe** : framework open-source de Google pour la détection en temps réel de visages, mains, corps et objets dans les flux vidéo.
- **OpenCV** : bibliothèque open-source de traitement d'images et de vision par ordinateur.
- **Pillow** : bibliothèque Python pour la manipulation et le traitement d'images.
- **Ultralytics** : bibliothèque Python permettant de faciliter l'utilisation de modèles YOLO (You Only Look Once)
- **TensorFlow** : bibliothèque open-source de Google pour le développement et l'entraînement de modèles de machine learning et deep learning.
- **PyTorch** : bibliothèque open-source développée par Facebook pour le calcul scientifique et le deep learning, très utilisée pour les projets de vision par ordinateur.
- **Matplotlib** : bibliothèque de traçage et de visualisation de données en 2D.
- **Seaborn** : bibliothèque de visualisation basée sur Matplotlib, offrant des graphiques statistiques attractifs.
- **Pandas** : bibliothèque de manipulation et d'analyse de données en Python, notamment via des DataFrames.
- **CUDA** : plateforme de calcul parallèle développée par NVIDIA pour accélérer l'exécution de calculs sur GPU, utilisée avec TensorFlow et PyTorch.

Conclusion

Ce chapitre a permis de formaliser l'ensemble des besoins liés au développement du système de vérification d'identité. Les exigences fonctionnelles ont été identifiées de manière détaillée, tout comme les contraintes non fonctionnelles qui garantissent la qualité du service. L'environnement technique, incluant les langages, outils et bibliothèques, a également été précisé afin de poser une base solide pour la phase de conception et de développement présentée dans le chapitre suivant.

Chapitre 4

Conception et développement

Contents

Introduction **18**

4.1 Architecture **18**

4.1.1 Diagramme de flow 18

4.1.2 Architecture frontend/backend 24

4.2 reCAPTCHA Gestuel **24**

4.2.1 Détection des gestes avec MediaPipe 24

4.2.2 Gestes supportés 25

4.2.3 Pipeline de Traitement des Gestes 25

4.3 Détection de la carte d'identité (CIN) **26**

4.3.1 Prétraitement des données 26

4.3.2 Métriques d'évaluation des modèles 27

4.3.3 Modèle CNN 29

4.3.4 Entraînement du modèle (CNN) 29

4.3.5 Analyse des performances (CNN) 30

4.3.6 Modèle YOLO 33

4.3.7 Entraînement du modèle (YOLO) 33

4.3.8 Analyse des performances (YOLO) 34

4.3.9 Étude comparative 36

4.4 Extraction de texte **36**

4.4.1 Problèmes 36

4.4.2 Solutions 37

4.5 Biometrique faciale **40**

4.5.1 Détection des émotions 40

4.5.2 Comparaison de visages 41

4.6 Implémentation **42**

4.6.1 Frontend 42
4.6.2 Back-end 47

Introduction

Ce chapitre présente l'architecture globale du système, ainsi que les étapes techniques de sa mise en œuvre. Il couvre les différents modules développés pour assurer la détection, l'extraction et la vérification d'identité. Chaque partie du système a été pensée pour être modulaire, performante et facilement intégrable dans un environnement mobile et web. Des choix technologiques précis ont été faits pour garantir l'efficacité du traitement et la fiabilité des résultats.

4.1 Architecture

L'architecture du système a été conçue de manière modulaire afin de permettre une intégration fluide entre les différents composants fonctionnels. Elle repose sur une communication entre un frontend mobile (React Native) et un backend (Django) qui regroupe l'ensemble des traitements IA, OCR et biométriques. Chaque étape du processus de vérification est traitée par un module dédié, garantissant ainsi évolutivité et maintenabilité.
Les deux sous-sections suivantes détaillent la structure fonctionnelle sous forme de diagramme de flux, puis l'architecture logicielle utilisée pour connecter les interfaces utilisateur avec les services d'analyse.

4.1.1 Diagramme de flow

Le diagramme de flow ci-dessous illustre les différentes étapes du processus de vérification d'identité tel qu'il a été conçu dans ce projet. Chaque utilisateur suit un parcours structuré, depuis l'ouverture de l'application jusqu'à la validation finale de son identité, en passant par plusieurs contrôles automatisés.
Le flux met en évidence les interactions entre les modules de reCAPTCHA gestuel, de capture de carte d'identité et de reconnaissance faciale Cette modélisation permet de visualiser clairement les dépendances et la logique de traitement globale du système.

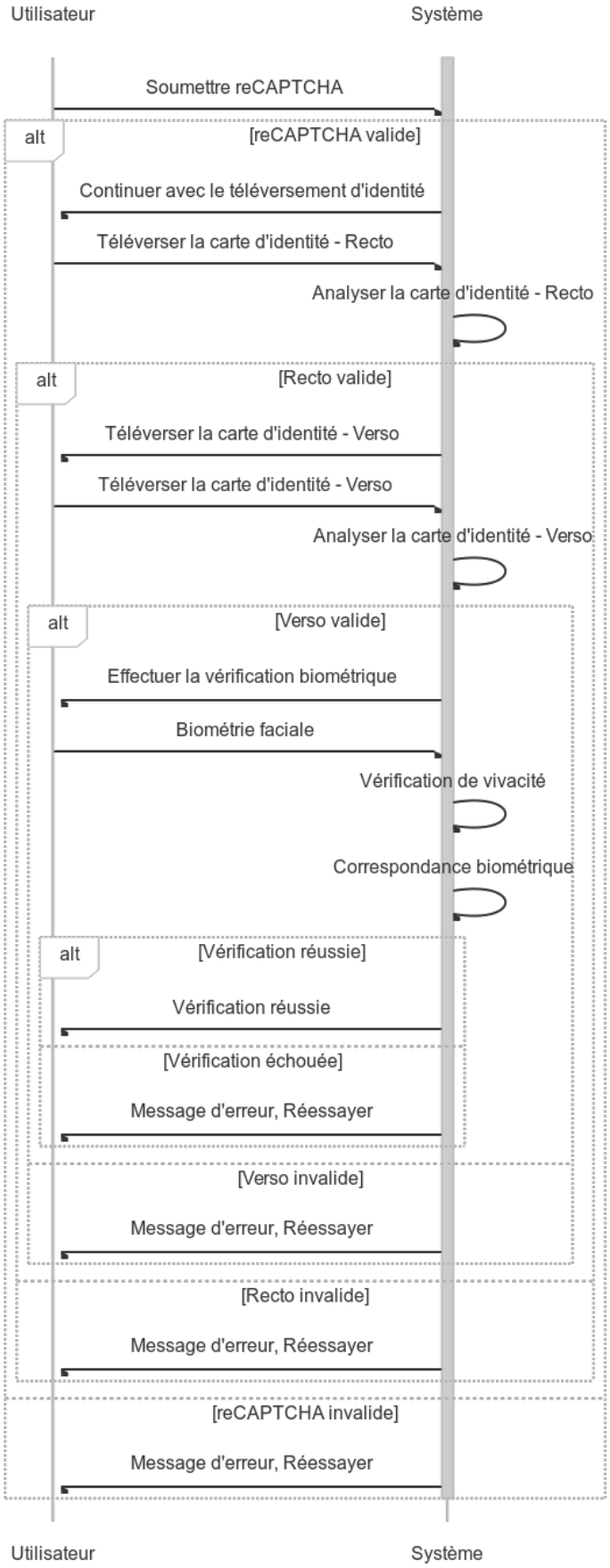

Fig. 4.1. Diagramme de séquence du système de vérification.

reCAPTCHA gestuel

Cette étape vise à vérifier que l'utilisateur est bien un humain à l'aide d'un test gestuel. L'utilisateur est invité à montrer un chiffre avec ses doigts via la caméra frontale. Le système utilise MediaPipe pour détecter et reconnaître la position des doigts en temps réel.

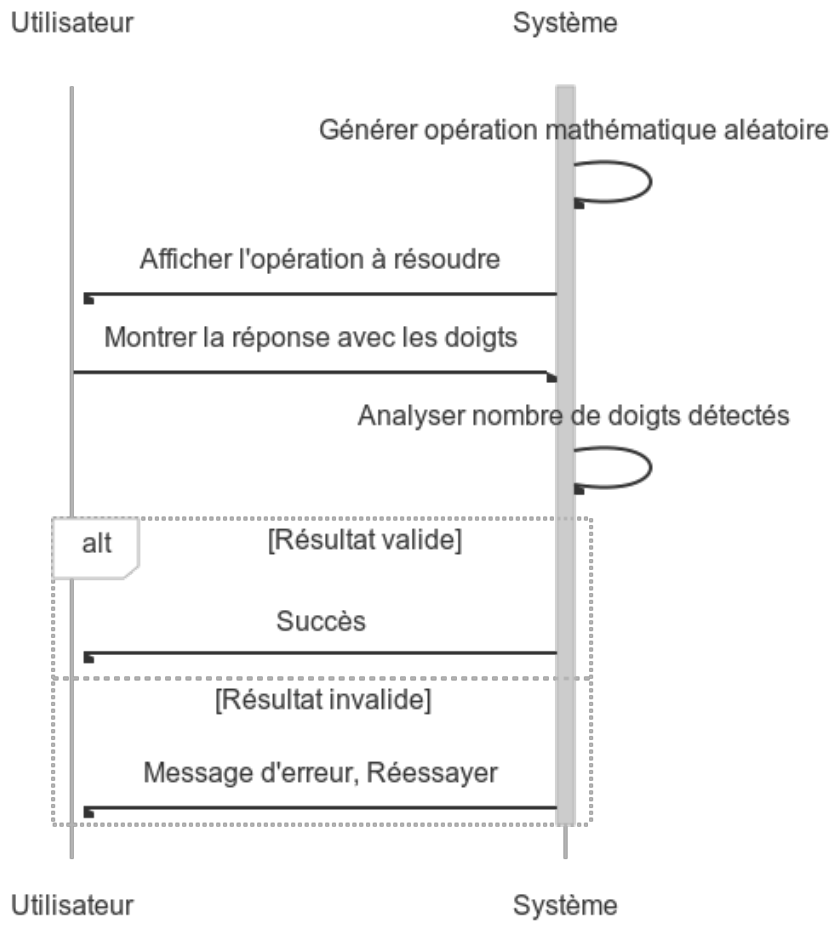

Fig. 4.2. Diagramme de séquence du reCAPTCHA gestuel.

Téléversement de la carte d'identité

L'utilisateur est ensuite invité à téléverser le recto et le verso de sa carte d'identité. Un modèle d'IA vérifie que les images sont valides et reconnaissables avant d'envoyer les données vers le module d'extraction des données

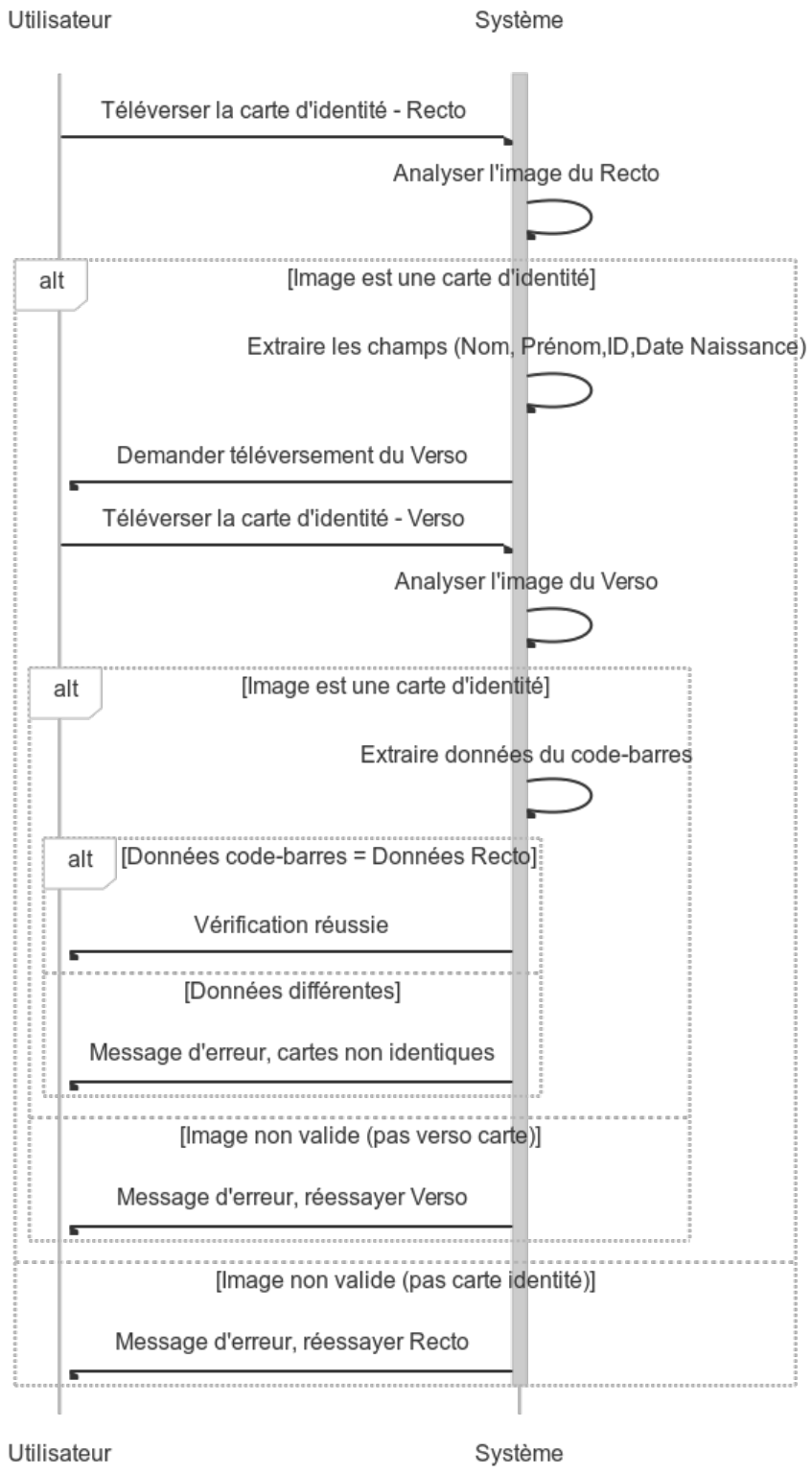

Fig. 4.3. Diagramme de séquence du téléversement et traitement de la CIN.

Vérification faciale biométrique

Après avoir extrait les informations de la carte d'identité, une vérification biométrique est effectuée. L'utilisateur est invité à prendre une photo de son visage avec la carte d'identité visible. Le système compare ensuite cette photo avec la photo extraite de la CIN à l'aide d'un modèle ArcFace.

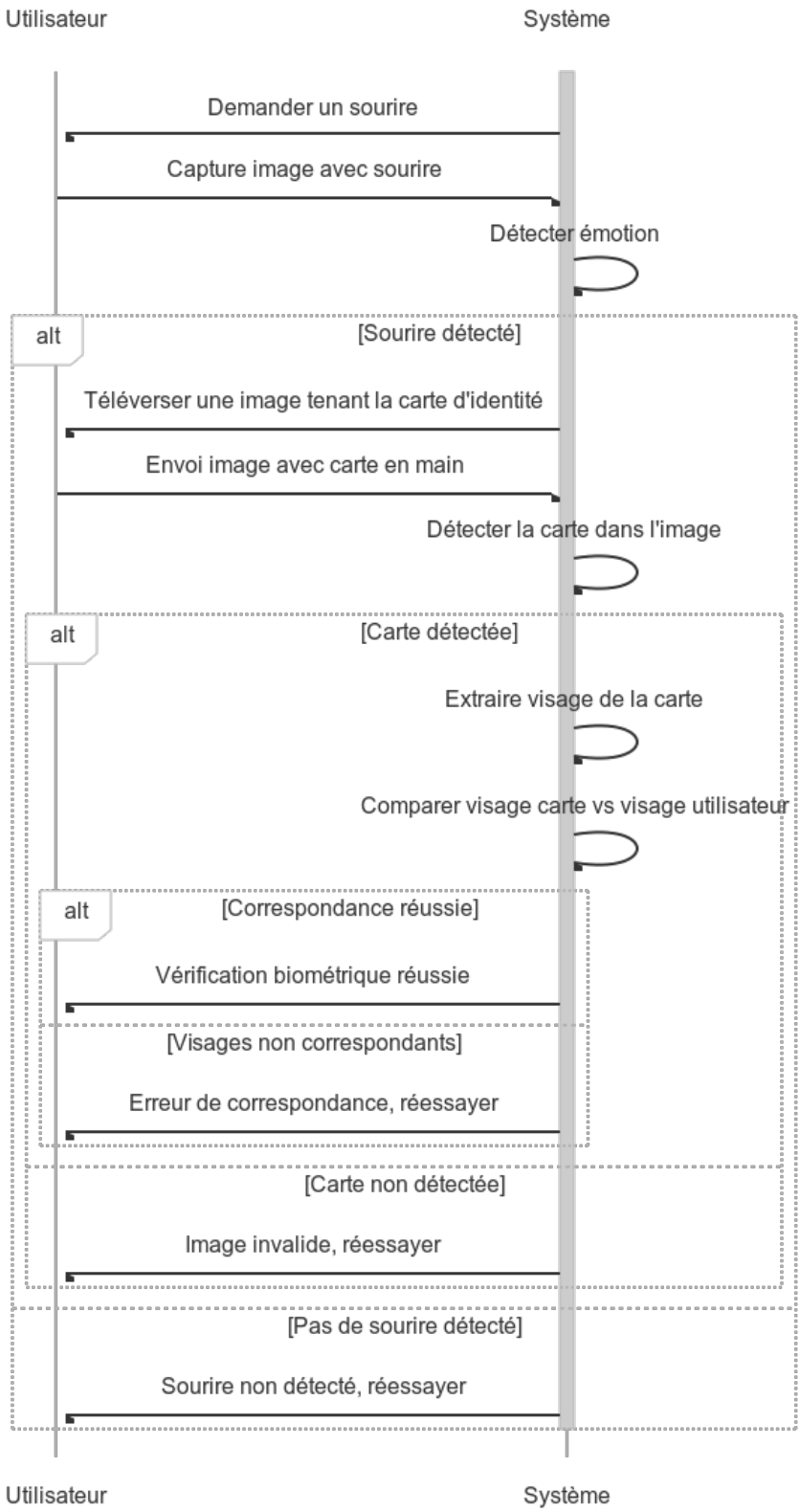

Fig. 4.4. Diagramme de séquence de la vérification faciale.

4.1.2 Architecture frontend/backend

L'architecture du système repose sur deux parties principales : le frontend et le backend. Le frontend gère l'interface utilisateur et la capture des données, tandis que le backend assure le traitement des images et la vérification d'identité à l'aide des modèles d'IA.

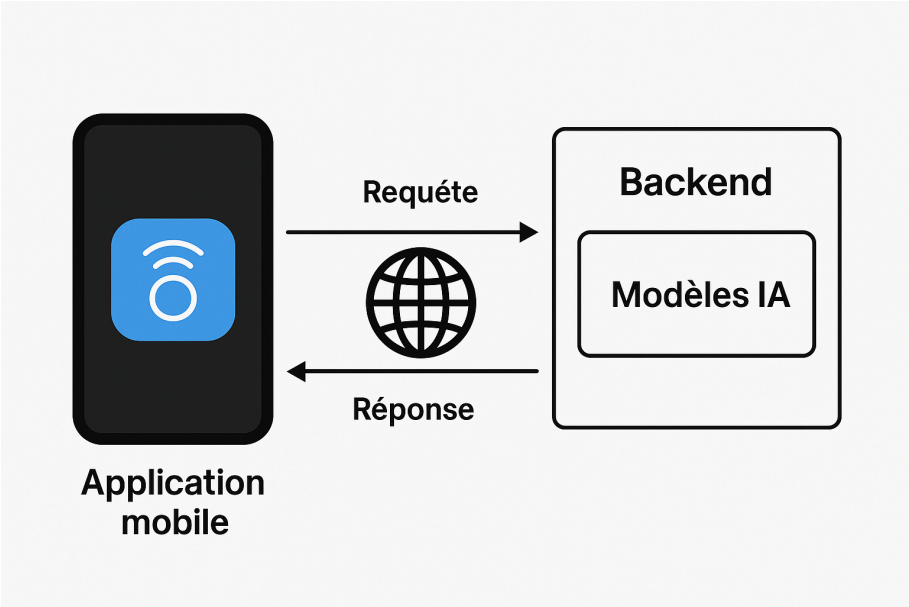

Fig. 4.5. Architecture simplifiée frontend/backend du système.

4.2 reCAPTCHA Gestuel

Pour renforcer la sécurité et s'assurer que l'utilisateur est bien une personne réelle, nous avons intégré une étape de vérification gestuelle avant la capture des documents d'identité. Fini les anciens CAPTCHA où il fallait cliquer sur des feux de signalisation ou déchiffrer des lettres floues !
À la place, notre système invite simplement l'utilisateur à effectuer quelques gestes faciles de la main devant la caméra. Ces mouvements sont reconnus en temps réel, prouvant ainsi qu'il s'agit bien d'une interaction humaine.
On appelle ça un IntelligentCAPTCHA : une vérification moderne et intuitive qui combine reconnaissance gestuelle et détection de mouvement pour offrir plus de sécurité sans casser l'expérience utilisateur. C'est simple, fluide et beaucoup plus naturel

4.2.1 Détection des gestes avec MediaPipe

MediaPipe est un framework modulaire et optimisé pour la capture et l'analyse des mouvements de la main en temps réel. S'appuyant sur des modèles de vision par ordinateur légers, il détecte jusqu'à deux mains simultanément et en extrait vingt et un repères tridimensionnels correspondant aux articulations et au poignet. À partir de ces données géométriques, MediaPipe reconnaît

une large variété de gestes — chiffres, signes de validation, alphabet gestuel, etc. — avec une latence minimale et une grande fiabilité.
Facile à intégrer aussi bien sur des applications web que mobiles ou embarquées, MediaPipe assure un suivi fluide malgré les variations d'éclairage et de position. Cette polyvalence en fait une solution de référence pour la reconnaissance gestuelle dans des domaines tels que la réalité augmentée, l'accessibilité, la robotique ou l'identité numérique. C'est ce moteur performant qui alimente notre système de vérification gestuelle, garantissant à la fois simplicité d'usage et robustesse face aux tentatives d'automatisation.

4.2.2 Gestes supportés

Afin de garantir une expérience utilisateur à la fois simple et naturelle, notre module se concentre exclusivement sur la reconnaissance de **gestes numériques** : l'utilisateur n'a qu'à lever de 0 à 5 doigts pour exprimer un chiffre.
Cette approche présente plusieurs avantages :

— **Intuitivité** : pas de formes complexes à mémoriser, l'utilisateur exprime directement le nombre souhaité.

— **Robustesse** : le calcul d'angles d'ouverture de doigts tolère mieux les variations d'orientation et de luminosité qu'une détection de mouvements dynamiques.

— **Mono-main** : en ne traitant qu'une seule main, on limite les confusions et on accélère le traitement en temps réel.

— **Gamme 0–5** : correspond exactement au nombre de doigts disponibles, ce qui minimise les erreurs de reconnaissance.

4.2.3 Pipeline de Traitement des Gestes

Le processus de détection repose sur un pipeline structuré permettant d'acquérir, traiter et analyser les gestes en continu. Ce pipeline se compose des étapes suivantes :

— **Capture en temps réel** : acquisition des flux vidéo depuis la caméra du terminal utilisé.

— **Détection des mains** : localisation des mains et des articulations à l'aide des modèles pré-entraînés de MediaPipe.

— **Extraction des points clés** : repérage des 21 points clés pour chaque main, représentant les articulations des doigts et le poignet.

— **Calcul des angles et distances** : détermination des angles entre les doigts et des distances relatives à partir des points extraits, servant à caractériser le geste.

— **Classification des gestes** : attribution d'une étiquette au geste détecté (par exemple : "1", "2", "5") selon la position des doigts et les valeurs angulaires mesurées.

— **Validation du geste** : comparaison entre le geste détecté et celui attendu pour autoriser l'utilisateur à poursuivre l'action.

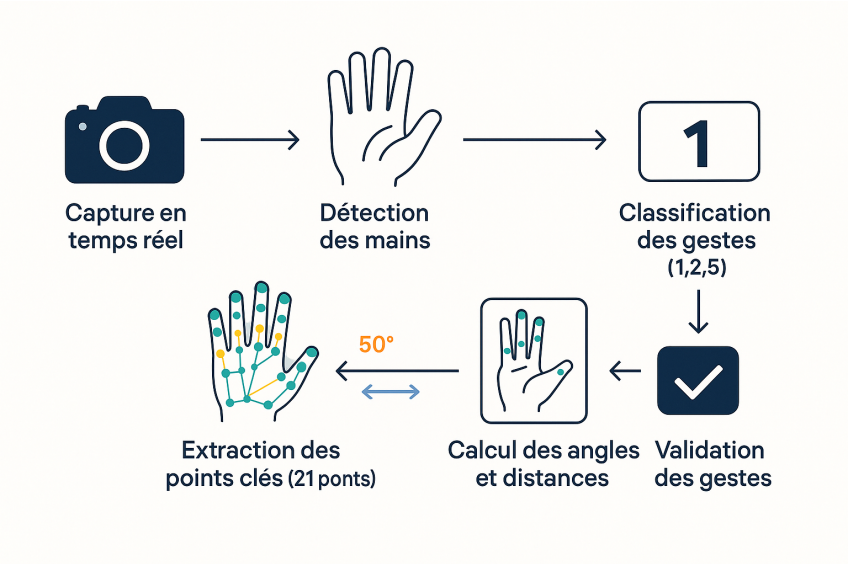

Fig. 4.6. Pipeline de traitement des gestes avec MediaPipe.

Ce pipeline optimise la reconnaissance gestuelle en assurant à la fois rapidité et précision, contribuant à renforcer la sécurité et l'ergonomie du système.

4.3 Détection de la carte d'identité (CIN)

le processus de vérification d'identité consiste à déterminer si l'image envoyée par l'utilisateur contient bien une carte d'identité nationale (CIN). Pour cela, un modèle de classification d'images a été entraîné afin de distinguer les images contenant une CIN de celles qui n'en contiennent pas. Ce modèle permet de filtrer automatiquement les images invalides, garantissant que seules les images pertinentes sont ensuite analysées pour l'extraction des don nées. Cette étape joue un rôle fondamental dans la fiabilité du système. Deux approches ont été explorées pour cette tâche : un modèle basé sur les réseaux de neurones convolutifs (CNN) et un autre utilisant l'architecture YOLO. Chacune de ces méthodes a été entraînée et testée dans des conditions variées afin de déterminer la plus efficace en termes de précision, de rapidité et de robustesse

4.3.1 Prétraitement des données

Avant l'entraînement du modèle de classification, une étape de prétraitement a été nécessaire pour garantir la qualité du jeu de données. Cette phase n'impliquait pas de transformations techniques complexes, mais un travail rigoureux de collecte et de filtrage manuel des images.

Les images utilisées représentent des cartes d'identité tunisiennes, et il était primordial de conserver leurs dimensions et leurs couleurs d'origine afin de préserver leur aspect réel. Aucun recadrage ni traitement automatique n'a été appliqué pour ne pas altérer les caractéristiques visuelles importantes.
Le prétraitement s'est concentré sur les aspects suivants :

— **Collecte d'images réalistes** : les images ont été sélectionnées pour représenter fidèlement des cas d'usage concrets, incluant différentes conditions d'éclairage et d'arrière-plan.

— **Filtrage des images floues ou partiellement visibles** : toutes les images dont la carte d'identité n'était pas clairement lisible ont été retirées afin d'éviter d'introduire du bruit dans l'apprentissage.

— **Suppression des doublons** : afin de garantir la diversité du jeu de données, les images en double ont été identifiées et supprimées.

— **Organisation par classe** : les images ont été triées en trois catégories distinctes : celles contenant des cartes d'identité tunisiennes, celles présentant des objets divers et celles affichant d'autres documents qui ne sont pas des cartes d'identité.

Cette étape de prétraitement a permis d'obtenir un jeu de données propre, équilibré et représentatif des différentes situations possibles.

4.3.2 Métriques d'évaluation des modèles

Afin de mesurer la performance des modèles développés, plusieurs métriques d'évaluation ont été utilisées. Ces métriques permettent d'analyser la précision et la robustesse des modèles de classification et de détection d'images.

Métriques pour le modèle CNN

Le modèle CNN a été évalué à l'aide des indicateurs suivants :

— **Accuracy (taux de bonne classification)** :

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

où TP : True Positive, TN : True Negative, FP : False Positive, FN : False Negative. Cette métrique donne la proportion globale de bonnes classifications.

— **Loss (fonction de perte)** : mesure l'écart entre la prédiction et la valeur réelle. Dans le cadre de la classification multi-classes par CNN, la fonction utilisée est la *Cross-Entropy Loss*, définie par :

$$Loss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij}\log(\hat{y}_{ij})$$

où N est le nombre d'échantillons, C le nombre de classes, y_{ij} la valeur réelle (1 ou 0) et $\hat{y}_{ij}$ la probabilité prédite pour la classe j de l'échantillon i. Cette fonction de perte n'est pas directement liée à la matrice de confusion et est calculée à partir des probabilités de sortie du modèle. Aucune opération dérivée (comme des ratios) n'est réalisée à partir de cette valeur : elle est optimisée en continu durant l'entraînement.

— **Matrice de confusion** : tableau récapitulatif permettant de visualiser les performances de classification pour chaque classe, en séparant les vrais positifs, faux positifs, vrais négatifs et faux négatifs.

— **Rappel (Recall)** :

$$Recall = \frac{TP}{TP + FN}$$

Mesure la capacité du modèle à identifier correctement les images positives. Cette valeur est dérivée des résultats de la matrice de confusion.

— **Précision (Precision)** :

$$Precision = \frac{TP}{TP + FP}$$

Évalue la proportion de prédictions positives correctes par rapport à toutes les prédictions positives. Issue également de la matrice de confusion.

— **F1-score** :

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

C'est la moyenne harmonique entre la précision et le rappel, offrant un compromis entre ces deux métriques. Ce score est utile en cas de classes déséquilibrées.

Métriques pour le modèle YOLO

Le modèle YOLO (You Only Look Once), qui effectue à la fois la détection et la classification d'objets dans les images, a été évalué selon les indicateurs suivants :

— **mAP (mean Average Precision)** :

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i$$

où AP_i est l'Average Precision de la classe i, calculée à partir des courbes précision-rappel pour plusieurs seuils d'Intersection over Union (IoU), et N le nombre de classes. Le mAP reflète la qualité des prédictions de localisation et de classification sur l'ensemble des objets détectés.

— **Loss (fonction de perte)** : YOLO utilise une perte composite, calculée comme la somme pondérée de plusieurs sous-losses :

$$Loss_{total} = \lambda_{coord} \times Loss_{coord} + \lambda_{obj} \times Loss_{conf}^{obj} + \lambda_{noobj} \times Loss_{conf}^{noobj} + \lambda_{class} \times Loss_{class}$$

avec :

— $Loss_{coord}$: erreur quadratique (MSE) sur les coordonnées des boîtes englobantes :

$$Loss_{coord} = \sum_{i=1}^{S^2} \sum_{j=1}^{B} 1_{ij}^{obj} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 + (w_i - \hat{w}_i)^2 + (h_i - \hat{h}_i)^2 \right]$$

— $Loss_{conf}$: erreur sur la confiance prédite (présence d'un objet ou non)

— $Loss_{class}$: erreur de classification des objets détectés, souvent sous forme de Cross-Entropy ou MSE

Les coefficients λ permettent d'ajuster l'importance de chaque composant. Contrairement aux métriques comme Recall ou Précision, le Loss est une mesure continue issue directement des sorties du modèle et des cibles réelles, sans passer par une matrice de confusion.

4.3.3 Modèle CNN

Le premier modèle exploré pour la détection de cartes d'identité est un réseau de neurones convolutifs (CNN). Cette architecture est particulièrement efficace pour l'analyse d'images, car elle extrait automatiquement les caractéristiques visuelles les plus pertinentes. Le modèle utilisé dans ce projet se compose des couches suivantes :

— **Couches convolutives** : application de filtres (noyaux) sur l'image d'entrée pour détecter motifs visuels (bords, textures, formes). Chaque filtre apprend une caractéristique spécifique.

— **Couches de pooling** : réduction de la taille des cartes de caractéristiques générées par les convolutions, diminuant la complexité tout en conservant l'information essentielle.

— **Couches de normalisation** : standardisation des activations pour accélérer l'apprentissage et stabiliser le réseau.

— **Couches entièrement connectées** : transformation des caractéristiques extraites en prédictions binaires (présence ou absence de CIN).

— **Fonctions d'activation** : introduction de la non-linéarité (ReLU, softmax) pour permettre l'apprentissage de relations complexes entre pixels.

Ce CNN a été entraîné sur un jeu d'images annotées contenant des cartes d'identité dans divers contextes (angles de prise de vue, luminosité, arrière-plans variés). Les résultats obtenus ont validé la faisabilité de la classification binaire avant d'envisager une approche de détection plus fine.

4.3.4 Entraînement du modèle (CNN)

Une fois le jeu de données préparé, l'entraînement du modèle CNN a été lancé avec l'objectif de classifier les images selon qu'elles contiennent ou non une carte d'identité. Le modèle a été

construit avec une architecture simple mais efficace, composée de plusieurs couches convolutives suivies de couches de regroupement (pooling) et d'une couche entièrement connectée en sortie.

— **Chargement des données** : les images ont été réparties en un ensemble d'entraînement et un ensemble de validation, avec un équilibre entre les deux classes.

— **Compilation du modèle** : le modèle a été compilé avec une fonction de perte adaptée à la classification binaire, et optimisé à l'aide de l'algorithme Adam.

— **Phase d'apprentissage** : l'entraînement a été réalisé sur plusieurs époques, permettant au réseau d'ajuster ses poids progressivement pour améliorer sa capacité de prédiction.

— **Suivi des performances** : à chaque époque, la précision et la perte ont été surveillées pour évaluer la convergence du modèle et éviter tout surapprentissage.

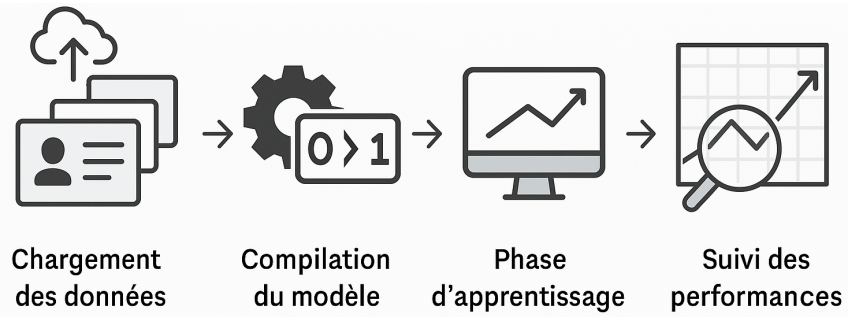

Fig. 4.7. Architecture du modèle.

4.3.5 Analyse des performances (CNN)

Les performances du modèle CNN ont été analysées à travers plusieurs indicateurs. Chaque indicateur fournit une vision complémentaire de l'efficacité du modèle à identifier correctement les cartes d'identité. Nous commençons par la précision (*accuracy*), dont les courbes d'évolution seront présentées ci-après.

Précision (accuracy)

D'après la Figure 4.8, les courbes de précision (*accuracy*) montrent une progression significative au fil des époques. L'accuracy d'entraînement a commencé autour de 0.55 et a atteint environ 0.85 à la fin de l'entraînement. De même, l'accuracy de validation a progressé de 0.55 à environ 0.83, indiquant que le modèle a amélioré sa capacité à généraliser sur des données inconnues sans signe de surapprentissage.

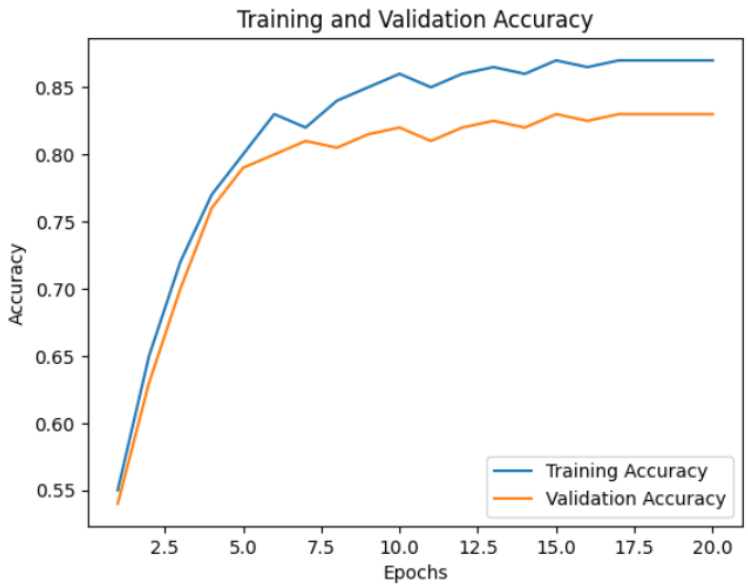

Fig. 4.8. Courbes de précision (*accuracy*) pour l'entraînement et la validation

Fonction de perte (Loss)

D'après la Figure 4.9, les courbes de fonction de perte (*loss*) montrent une diminution régulière au fil des époques. La perte d'entraînement a commencé autour de 0.9 et a rapidement chuté pour atteindre environ 0.3, tandis que la perte de validation a également diminué de manière constante, se stabilisant autour de 0.4. Cela indique une amélioration progressive de la capacité du modèle à faire des prédictions précises sans surajustement notable.

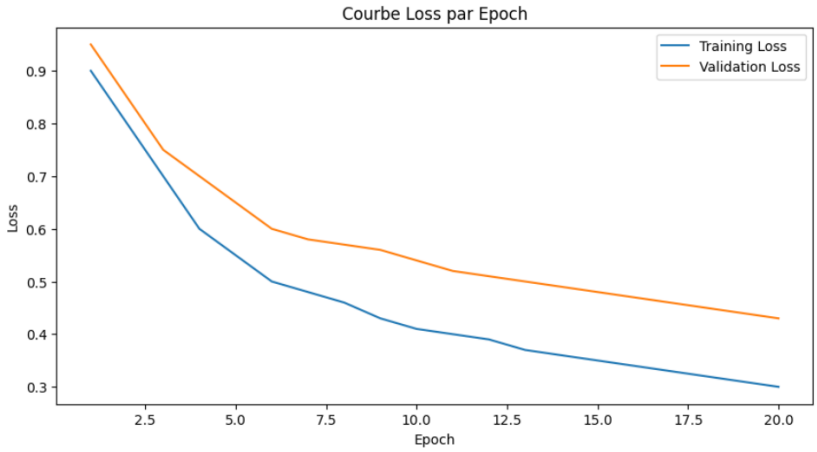

Fig. 4.9. Courbes de fonction de perte (*loss*) pour l'entraînement et la validation

Matrice de confusion

En analysant la matrice de confusion de la Figure 4.8 , on peut observer les performances de classification du modèle sur les trois classes suivantes : `cin`, `no_tunisian_cin` et `tunisian_cin`. Voici un résumé des résultats :

— `cin` : 85 instances correctement classées, 10 mal classées comme `no_tunisian_cin` et 9 mal classées comme `tunisian_cin`.

— `no_tunisian_cin` : 91 instances correctement classées, 5 mal classées comme `cin` et 8 mal classées comme `tunisian_cin`.

— `tunisian_cin` : 89 instances correctement classées, 11 mal classées comme `cin` et 4 mal classées comme `no_tunisian_cin`.

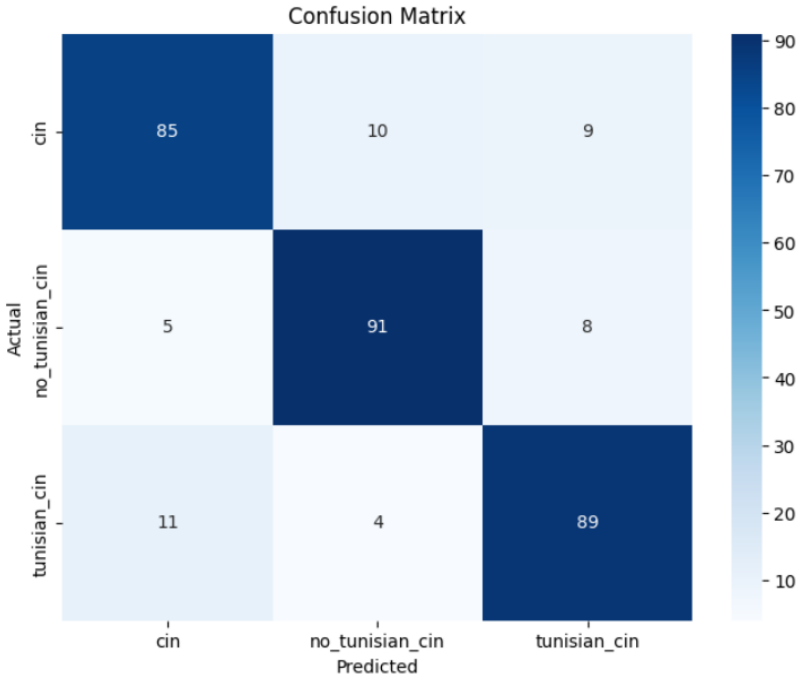

Fig. 4.10. confusion matrix

Cette matrice révèle une performance solide, avec un nombre élevé de classifications correctes pour chaque classe, en particulier pour les catégories `no_tunisian_cin` et `tunisian_cin`, qui présentent des taux d'erreurs plus faibles. Cela montre une nette amélioration par rapport aux versions précédentes, avec une réduction des erreurs inter-classes.

Classe	**Précision**	**Rappel**	**F1-score**	**Support**
`cin`	0.84	0.82	0.83	104
`no_tunisian_cin`	0.87	0.88	0.87	104
`tunisian_cin`	0.84	0.86	0.85	104
Moyenne pondérée	0.85	0.85	0.85	312

Tab. 4.1. Rapport de classification du modèle CNN

— **Précision** : mesure la capacité du modèle à prédire correctement une classe donnée parmi toutes les prédictions faites pour cette classe.

— **Rappel** : évalue la capacité du modèle à trouver tous les exemples pertinents d'une classe donnée.

— **F1-score** : combine précision et rappel pour donner une mesure unique de performance, utile en cas de classes déséquilibrées.

— **Support** : représente le nombre total d'exemples réels pour chaque classe.

Ces métriques fournissent une évaluation plus complète des performances du modèle, en tenant compte à la fois de la capacité à prédire correctement et à couvrir l'ensemble des exemples pertinents.

4.3.6 Modèle YOLO

Le second modèle exploré pour la détection de cartes d'identité est un réseau de détection d'objets de type YOLOv8 (*You Only Look Once version 8*). Cette architecture est particulièrement efficace pour la localisation précise des objets dans les images, tout en maintenant une vitesse d'inférence élevée, ce qui la rend idéale pour les applications en temps réel.
Le modèle YOLOv8 utilisé dans ce projet se compose des composants suivants :

— **Backbone** : une série de couches convolutionnelles profondes pour extraire des caractéristiques visuelles complexes à différents niveaux (bords, textures, formes). Le backbone utilise des connexions CSP (*Cross Stage Partial*) pour améliorer l'efficacité des calculs.

— **Neck (PANet)** : une structure en forme de pyramide pour fusionner les caractéristiques extraites à différentes échelles, permettant une meilleure détection des petits et grands objets.

— **Head (Détection)** : produit les prédictions finales, comprenant les coordonnées des boîtes englobantes, les scores de confiance et les classes prédictes. Utilise des couches de convolution finales pour affiner les résultats.

— **Fonctions d'activation (SiLU/Leaky ReLU)** : introduit de la non-linéarité pour permettre au modèle d'apprendre des relations complexes entre les caractéristiques visuelles.

— **Anchor-free Head** : contrairement aux versions précédentes de YOLO, YOLOv8 utilise une approche sans ancres pour une détection plus rapide et plus précise.

Ce modèle a été entraîné sur un jeu d'images annotées contenant des cartes d'identité dans divers contextes (angles de prise de vue, luminosité, arrière-plans variés). Les résultats obtenus ont validé la capacité du modèle à détecter et à localiser précisément les cartes d'identité, offrant une alternative plus avancée par rapport au CNN classique.

4.3.7 Entraînement du modèle (YOLO)

Une fois le jeu de données préparé, l'entraînement du modèle YOLOv8 a été lancé avec l'objectif de détecter et classifier les cartes d'identité. Contrairement aux réseaux CNN classiques,

YOLOv8 est conçu pour prédire à la fois la localisation des objets et leur classe en une seule étape, offrant ainsi une efficacité accrue.
Le processus d'entraînement du modèle YOLOv8 s'est déroulé en plusieurs étapes :

— **Chargement des données** : les images ont été réparties en un ensemble d'entraînement et un ensemble de validation, avec un équilibre entre les classes. Les annotations incluent les coordonnées des boîtes englobantes et les étiquettes de classe.

— **Compilation du modèle** : le modèle a été compilé avec une fonction de perte adaptée à la détection et à la classification des objets, en intégrant des métriques comme la *Binary Cross-Entropy* pour les classes et la *IoU (Intersection over Union)* pour les boîtes englobantes. L'optimisation a été réalisée à l'aide de l'algorithme AdamW pour une convergence rapide et stable.

— **Phase d'apprentissage** : l'entraînement a été réalisé sur plusieurs époques, permettant au réseau d'ajuster ses poids progressivement pour améliorer à la fois la précision des classifications et la qualité des localisations.

— **Suivi des performances** : à chaque époque, les métriques comme la précision, la perte totale et l'IoU ont été surveillées pour évaluer la convergence du modèle et éviter le surapprentissage. Des techniques comme l'*early stopping* et le *learning rate scheduler* ont été utilisées pour optimiser l'entraînement.

Ce protocole d'entraînement, combiné à une gestion rigoureuse des hyperparamètres, a permis de créer un modèle robuste capable de détecter et classer les cartes d'identité avec une précision élevée.

4.3.8 Analyse des performances (YOLO)

mAP (Mean Average Precision)

Le **mAP** (*Mean Average Precision*) est une métrique couramment utilisée pour évaluer les performances des modèles de détection d'objets comme YOLOv8. Il mesure la précision moyenne des prédictions en tenant compte à la fois de la localisation correcte des objets (Intersection over Union, IoU) et de leur classification correcte. Le mAP est calculé en moyennant les précisions à différents seuils d'IoU, typiquement entre 0.5 et 0.95, offrant ainsi une évaluation plus complète de la qualité du modèle.

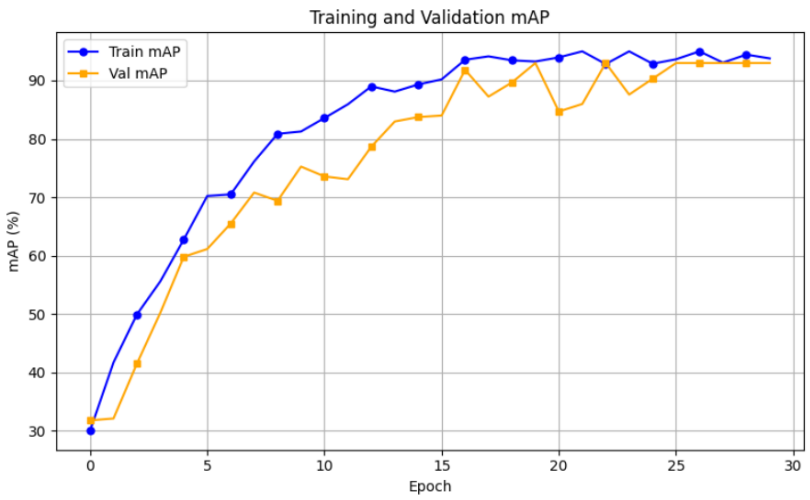

Fig. 4.11. Matrice de confusion.

Dans la Figure 4.11, on peut observer une forte progression du mAP pour l'entraînement et la validation au fil des époques, atteignant environ 90% en fin de formation. La courbe d'entraînement monte rapidement, indiquant une convergence rapide, tandis que la courbe de validation montre une légère fluctuation autour de 93%, ce qui est typique lorsque le modèle commence à atteindre sa capacité maximale de généralisation. Cette stabilité autour de 93% est un signe que le modèle a bien appris les caractéristiques des cartes d'identité, sans signe évident de surapprentissage.

Fonction de perte (Loss)

Les courbes de fonction de perte (*Loss*) sont des indicateurs clés de la performance d'un modèle, car elles montrent comment l'erreur du modèle diminue au fil des époques. Dans la Figure 4.12, on peut voir l'évolution des pertes pour l'entraînement (gauche) et la validation (droite). Les courbes incluent trois types de pertes :

— **Box Loss** (Bleu) : mesure l'erreur de localisation des boîtes englobantes. Elle diminue rapidement au début, se stabilisant autour de 0 après 20 époques, montrant que le modèle a bien appris à localiser les cartes d'identité.

— **Cls Loss** (Orange) : correspond à l'erreur de classification. Cette perte diminue également de manière significative, indiquant que le modèle devient plus précis pour différencier les classes.

— **DFL Loss** (Vert) : la perte associée à la régularisation des bords des boîtes (distribution focal loss), essentielle pour améliorer la précision spatiale des prédictions. Cette perte se stabilise également, confirmant une convergence réussie.

Dans l'ensemble, les courbes montrent une réduction constante des erreurs, suggérant que le modèle a atteint une convergence stable sans signe de surapprentissage.

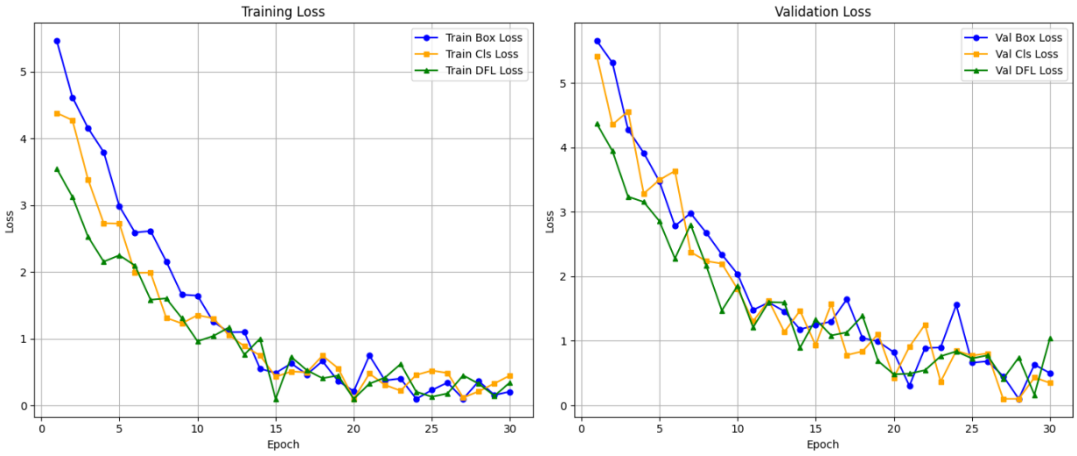

Fig. 4.12. Courbes de fonction de perte pour l'entraînement et la validation du modèle YOLOv8

4.3.9 Étude comparative

En comparant les performances du CNN et de YOLOv8, il apparaît que YOLOv8 surpasse largement le CNN en termes de précision ,et du rapidité, grâce à son architecture optimisée pour la détection et la classification simultanées. De plus, YOLOv8 maintient des pertes plus faibles et une meilleure stabilité au fil des époques, réduisant les erreurs de classification observées avec le CNN. Ce choix est également justifié par la nécessité de localiser précisément les cartes d'identité avant l'étape de biométrie faciale, où une détection exacte est essentielle pour assurer la cohérence entre l'image de la CIN et la photo capturée en temps réel.

4.4 Extraction de texte

L'extraction précise des informations à partir des documents d'identité est un défi technique majeur en raison de la diversité des formats, des langues et des variations d'alignement des images. Les systèmes comme EasyOCR, qui combinent détection et reconnaissance des caractères, peuvent rencontrer des difficultés dans la segmentation précise des champs et l'interprétation des textes complexes comme l'arabe. Ces limitations peuvent entraîner des erreurs critiques lors de l'analyse des données, nécessitant des solutions sur mesure pour garantir une extraction fiable et cohérente.

4.4.1 Problèmes

Problème de Détection des Champs

EasyOCR se compose de deux étapes principales : une couche de détection des champs de texte et une couche de reconnaissance des lettres. Le problème rencontré provient souvent de la première couche, qui doit isoler précisément chaque champ avant l'étape de reconnaissance.

Parfois, cette couche ne parvient pas à détecter certains champs essentiels ou n'extrait qu'une partie du texte, ce qui peut entraîner des erreurs critiques lors de l'extraction des informations nécessaires pour les étapes de vérification ultérieures.

Fig. 4.13. Détection des champs avec easyOcr

Problème avec le texte arabe

Un autre problème rencontré avec EasyOCR concerne l'extraction du texte en arabe, qui se lit de droite à gauche et présente des ligatures complexes. Lorsque l'image de la carte d'identité est prise sous un angle incorrect ou est inclinée, EasyOCR peut échouer à extraire correctement les caractères, créant ainsi des erreurs de lecture.

4.4.2 Solutions

Solution pour la Détection des Champs

Pour surmonter cette limitation, nous avons décidé de remplacer la première couche de détection des champs d'EasyOCR par un modèle YOLO spécialement entraîné pour cette tâche. Nous avons créé un jeu de données dédié, composé d'images de cartes d'identité, où chaque champ nécessaire (nom, prénom, numéro de CIN, date de naissance, etc.) a été annoté individuellement. Le modèle YOLO ainsi entraîné pourra détecter précisément ces champs, les découper (crop) et les envoyer ensuite à la seconde couche d'EasyOCR pour la reconnaissance des lettres. Cette approche permet de réduire les erreurs de segmentation et d'améliorer la précision globale de l'extraction des informations.

Fig. 4.14. Detection des champs avec YOLO

Analyse des performances

mAP (Mean Average Precision) :

La Figure 4.15 montre l'évolution de la précision (mAP) pour l'entraînement et la validation du modèle YOLO utilisé pour détecter les champs sur les cartes d'identité. On observe que les courbes de précision augmentent rapidement au début de l'entraînement, atteignant une précision stable autour de 80% après environ 10 époques. Cette convergence rapide indique que le modèle a bien appris à localiser les champs nécessaires, avec une différence minimale entre les courbes d'entraînement et de validation, suggérant une bonne généralisation sans surapprentissage notable.

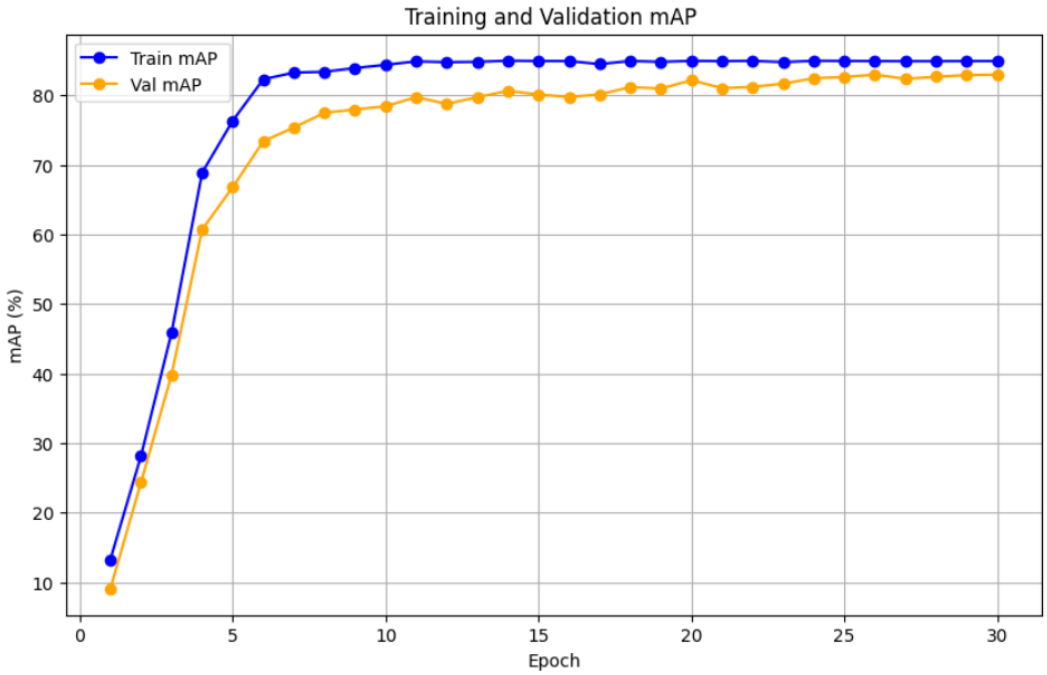

Fig. 4.15. mAP curves.

Fonction de perte (Loss)

La Figure 4.16 montre l'évolution des courbes de perte (*Loss*) pour l'entraînement (gauche) et la validation (droite) du modèle YOLO utilisé pour détecter les champs sur les cartes d'identité. Les courbes incluent trois types de pertes :

— **Box Loss** (Bleu) : mesure l'erreur de localisation des boîtes englobantes. Cette perte diminue rapidement au début, puis se stabilise autour de 0.5 après environ 10 époques, indiquant que le modèle a appris à localiser les champs de manière cohérente.

— **Cls Loss** (Orange) : correspond à l'erreur de classification des objets détectés. Elle diminue rapidement au début, se stabilisant autour de 0.5, ce qui montre que le modèle a rapidement appris à classer les champs correctement.

— **DFL Loss** (Vert) : représente l'erreur associée à la régularisation des bords des boîtes (distribution focal loss), essentielle pour améliorer la précision spatiale des prédictions. Elle reste relativement basse et stable après les premières époques, indiquant une bonne convergence.

Les courbes montrent une réduction rapide des erreurs au début de l'entraînement, suivie d'une stabilisation, ce qui suggère que le modèle a atteint une convergence stable sans signe de surapprentissage.

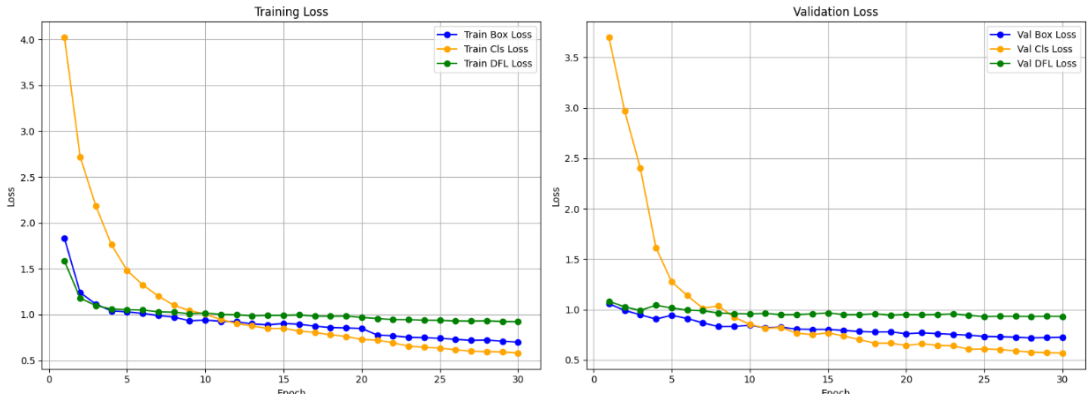

Fig. 4.16. Loss curves.

Solution pour le Texte Arabe

Pour surmonter ce problème, une solution consiste à réorienter automatiquement l'image avant l'extraction pour trouver l'angle optimal, assurant ainsi que le texte est correctement aligné pour la reconnaissance. Cette approche permet de maximiser les chances d'extraire correctement les informations en arabe, en ajustant dynamiquement l'orientation des cartes avant l'analyse.

Fig. 4.17. Exemple.

4.5 Biometrique faciale

Après avoir développé les modèles et résolu les problèmes liés à l'extraction de texte, le système Identicore intègre désormais la biométrie faciale comme étape clé de vérification. Cette approche repose sur deux étapes principales : la détection des émotions pour s'assurer de la présence réelle de l'utilisateur et la comparaison faciale pour vérifier la correspondance entre le visage capturé et la photo sur la carte d'identité

4.5.1 Détection des émotions

La détection des émotions est une étape cruciale pour vérifier que l'utilisateur est bien une personne réelle et non une tentative de fraude utilisant une image ou une vidéo. Cette approche renforce la sécurité en s'assurant que l'utilisateur présente des signes de vie, comme des expressions faciales naturelles, ce qui est difficile à imiter avec des photos ou des masques.
Pour cette tâche, nous utilisons **DeepFace**, une bibliothèque avancée de reconnaissance faciale qui intègre des modèles comme VGG-Face, Facenet et OpenFace pour l'extraction d'embeddings faciaux. DeepFace est particulièrement adapté à cette application car il peut détecter les micro-expressions et analyser les émotions en temps réel, offrant ainsi une analyse plus précise des états émotionnels.

Prétraitement des données

Avant l'analyse des émotions, il est essentiel de préparer les données faciales pour garantir des résultats précis. Ce prétraitement inclut l'alignement du visage pour corriger les variations d'angle, la normalisation des couleurs pour réduire l'impact des conditions d'éclairage, et la

réduction du bruit pour améliorer la qualité des images. Ces étapes sont cruciales pour s'assurer que les caractéristiques extraites sont représentatives des véritables émotions de l'utilisateur.

Comparaison des embeddings

Une fois les données prétraitées, DeepFace extrait les caractéristiques faciales sous forme d'embeddings, des vecteurs numériques représentant les traits distinctifs du visage. Pour détecter l'émotion de l'utilisateur, ces embeddings sont comparés à des vecteurs de référence correspondant à différentes émotions comme la joie, la tristesse, la colère ou la surprise. Cette comparaison permet de déterminer l'émotion dominante en calculant la distance entre les embeddings extraits et les vecteurs émotionnels de référence, généralement en utilisant des métriques comme la distance cosinus.

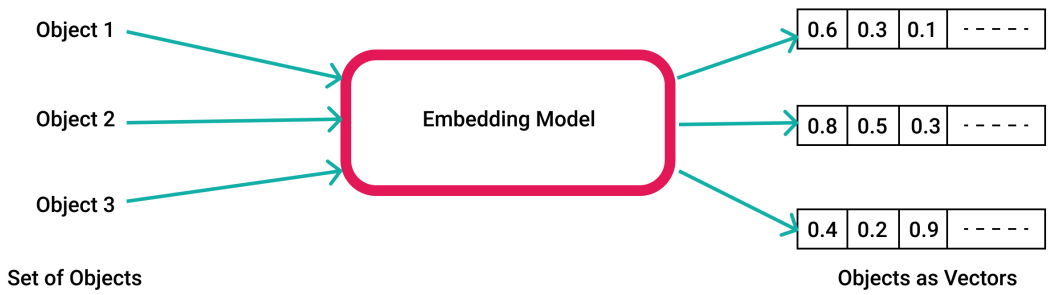

Fig. 4.18. Architecture du modèle.

Cette combinaison de prétraitement des images et de comparaison des embeddings permet de détecter les émotions avec une grande précision, assurant ainsi une authentification plus robuste et sécurisée.

4.5.2 Comparaison de visages

Après la détection des émotions, l'étape suivante consiste à vérifier que l'utilisateur est bien la même personne que celle représentée sur la carte d'identité. Pour cette tâche, nous utilisons **ArcFace**, un modèle avancé de comparaison de visages basé sur des embeddings faciaux. ArcFace est connu pour sa capacité à créer des représentations vectorielles précises des visages, permettant une comparaison fiable même dans des conditions d'éclairage et d'angle variables.

Prétraitement des données

Comme pour la détection des émotions, un prétraitement des données est nécessaire pour garantir une comparaison précise. Cette étape inclut l'extraction des deux visages : celui de l'utilisateur capturé en temps réel et celui présent sur la carte d'identité. Les images sont ensuite

alignées, normalisées et recadrées pour s'assurer que les caractéristiques faciales sont correctement centrées et mises à l'échelle.

Comparaison des embeddings

Une fois les données prétraitées, ArcFace extrait les embeddings des deux visages sous forme de vecteurs numériques représentant les traits distinctifs du visage. La comparaison est effectuée en calculant la distance entre ces deux ensembles d'embeddings. Les distances couramment utilisées pour cette tâche sont la distance euclidienne et la distance cosinus, qui mesurent le degré de similarité entre les deux vecteurs. Si la distance est inférieure à un certain seuil prédéfini, les deux visages sont considérés comme appartenant à la même personne, sinon, l'authentification est refusée.

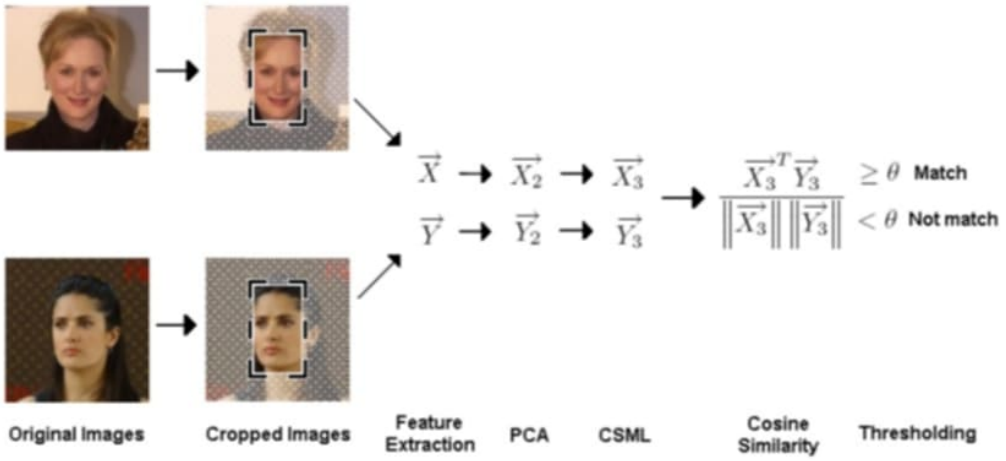

Fig. 4.19. Pipeline de vérification faciale.

Cette méthode assure une vérification précise de l'identité, combinant prétraitement et comparaison avancée pour garantir une correspondance fiable entre le visage réel et celui présent sur la carte d'identité.

4.6 Implémentation

L'implémentation de notre système de vérification d'identité repose sur une architecture front-end moderne, conçue pour offrir une expérience utilisateur fluide et sécurisée.

4.6.1 Frontend

Le frontend de l'application Identicore est développé en React Native, permettant de créer une interface utilisateur mobile fluide et performante, compatible avec iOS et Android. Grâce à React Native, l'application propose des fonctionnalités interactives telles que la capture d'images,

la vérification faciale en temps réel et l'affichage des résultats de manière intuitive. Cette approche garantit une expérience utilisateur optimale tout en tirant parti des performances natives des dispositifs mobiles .

Les interfaces

Les interfaces utilisateur d'Identicore sont conçues pour être intuitives et réactives, offrant une expérience fluide et sécurisée. Elles intègrent des fonctionnalités clés comme le téléversement des documents, la vérification faciale et l'analyse des émotions, tout en assurant une navigation simple et rapide pour les utilisateurs.

Page d'accueil

L'application démarre avec une page d'accueil accueillante, présentant les fonctionnalités principales et guidant l'utilisateur à travers les différentes étapes du processus de vérification d'identité. Cette page sert de point d'entrée pour l'expérience utilisateur et inclut des boutons pour commencer le processus ou pour accéder à des informations supplémentaires.

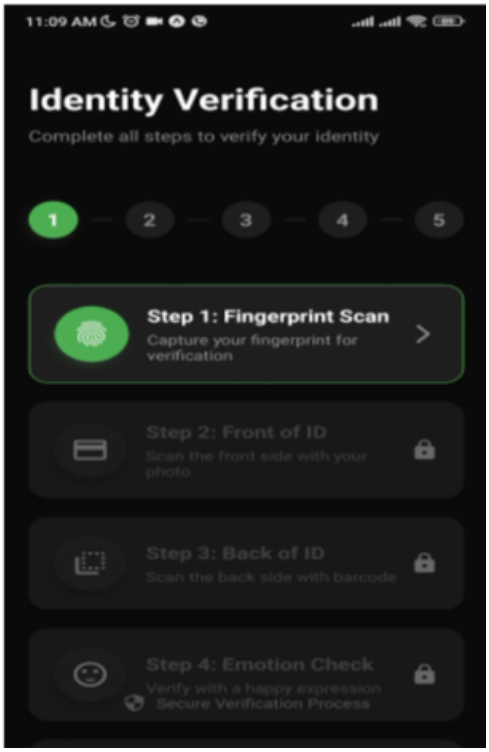

Fig. 4.20. Page d'accueil de l'application.

Interface reCAPTCHA

Dans cette interface, l'utilisateur doit effectuer une opération affichée à l'écran en utilisant ses doigts, comme montrer un nombre spécifique ou effectuer un geste particulier. Cette étape sert de vérification supplémentaire pour s'assurer que l'utilisateur est bien une personne réelle.

Fig. 4.21. Interface reCAPTCHA.

Téléversement de la CIN

L'application propose deux interfaces distinctes pour le téléversement de la carte d'identité nationale (CIN) :

- **Recto** : pour capturer l'image de la face avant de la carte.

Fig. 4.22. Interface de téléversement du recto de la CIN.

- **Verso** : pour capturer l'image de la face arrière, souvent nécessaire pour extraire des informations supplémentaires comme le code QR ou le code à barres.

Fig. 4.23. Interface de téléversement du verso de la CIN.

Biométrie faciale

Cette étape comprend deux interfaces :
- **Détection des émotions** : l'utilisateur est invité à montrer une expression spécifique (comme un sourire) pour valider sa présence en temps réel.

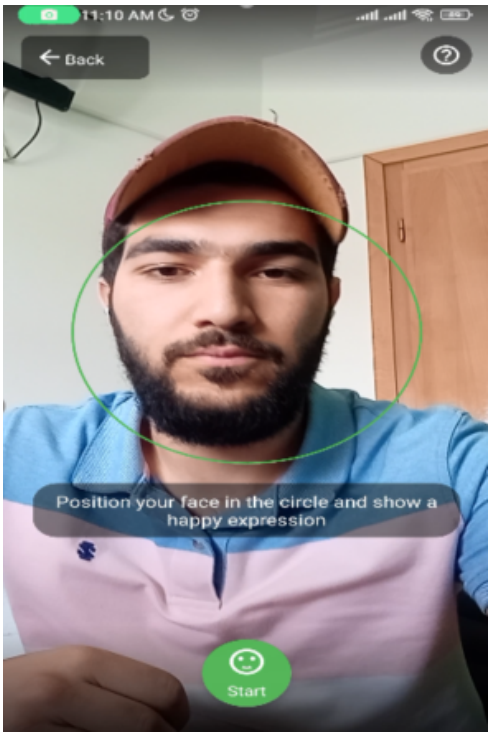

Fig. 4.24. Interface de détection des émotions.

- **Vérification du visage** : une fois l'émotion détectée, l'utilisateur est invité à comparer son visage avec celui présent sur la carte d'identité pour confirmer son identité.

Fig. 4.25. Interface de vérification du visage.

Ces interfaces sont conçues pour être intuitives et réactives, garantissant une expérience utilisateur fluide tout en maintenant un haut niveau de sécurité pour le processus de vérification.

4.6.2 Back-end

Pour le développement du back-end, nous avons utilisé **Django REST Framework**, une puissante bibliothèque pour créer des API robustes et sécurisées en Python. Le back-end est responsable de la gestion des flux de données entre le front-end et les modèles d'IA, ainsi que de la logique métier pour chaque étape du processus de vérification d'identité.

Services API REST

Les services API REST sont les points d'entrée pour les communications entre le front-end et le back-end. Ils sont conçus pour être légers, réactifs et sécurisés, avec des messages d'erreur détaillés pour chaque étape du processus :
- API ReCAPTCHA
- API de classification des images (CIN)
- API de détection des émotions
- API de vérification du visage
Ces API sont conçues pour recevoir des images, traiter les données en temps réel et retourner des réponses rapides pour garantir une expérience utilisateur fluide.

API ReCAPTCHA

Cette API reçoit en temps réel une image de la main de l'utilisateur, capturée via l'interface ReCAPTCHA. Pour garantir une faible latence et une réponse instantanée, nous utilisons une technique de **buffering** pour transmettre les images sans délai. Le buffering est une méthode qui consiste à stocker temporairement des données dans une mémoire tampon avant de les traiter ou de les transmettre, ce qui permet de réduire les temps de latence en envoyant les données en flux continu plutôt que par paquets distincts.
L'API analyse ensuite la main de l'utilisateur pour vérifier si le geste attendu est correct, en utilisant des techniques de détection des doigts pour compter le nombre affiché ou reconnaître des signes spécifiques. Cette approche permet de valider rapidement que l'utilisateur est bien une personne réelle et non une tentative de fraude utilisant une image ou une vidéo.

API de classification des images (CIN)

Cette API a pour rôle de vérifier si l'image reçue est bien une carte d'identité nationale (CIN) et si tous les champs nécessaires sont visibles. Lorsqu'une image est envoyée, elle est accompagnée d'un mot-clé précisant si elle correspond au "recto" ou au "verso" de la carte. L'API effectue alors plusieurs vérifications :
- **Validation de la carte** : elle vérifie si l'image contient bien une carte d'identité en utilisant un modèle YOLO spécialement entraîné pour cette tâche.
- **Vérification des champs** : une fois l'image validée comme étant une carte, l'API utilise également le modèle YOLO pour détecter et vérifier que tous les champs essentiels (nom, prénom, numéro de CIN, date de naissance) sont présents et correctement visibles. Cette approche permet de s'assurer que les informations sont complètes avant de procéder à l'extraction du texte.
- **Extraction de texte** : si tous les champs sont visibles, ils sont ensuite découpés (crop) et envoyés à EasyOCR pour la reconnaissance des caractères. Cette méthode réduit les erreurs de segmentation et améliore la précision des résultats.
Cette approche centralisée avec YOLO pour la validation des cartes et des champs permet de garantir une extraction plus fiable et cohérente des données présentes sur les cartes d'identité.

API de détection des émotions

Cette API gère l'étape de détection des émotions, vérifiant si l'utilisateur montre les expressions correctes pour garantir qu'il est bien présent physiquement. Elle retourne un message de succès si l'émotion détectée correspond à celle attendue, sinon, elle renvoie un message d'erreur.

API de vérification du visage

Enfin, l'API de vérification du visage compare les embeddings du visage de l'utilisateur avec ceux extraits de la carte d'identité pour confirmer l'identité. Cette étape utilise des modèles

comme ArcFace pour calculer les distances entre les vecteurs d'embeddings et déterminer si les visages appartiennent à la même personne.
L'API reçoit une image contenant à la fois le visage de l'utilisateur et la carte d'identité. Elle doit d'abord détecter et segmenter ces deux régions distinctes : - Le visage de l'utilisateur, pour l'extraction des embeddings en temps réel. - Le visage présent sur la carte d'identité, pour l'extraction des embeddings de référence.
De plus, l'API vérifie également que la carte d'identité est bien visible dans les mains de l'utilisateur avant de procéder à la comparaison des visages. Cette double vérification renforce la sécurité et l'authenticité du processus de reconnaissance faciale.

Gestion des messages

Pour chaque API, un système de gestion des messages a été mis en place pour retourner des réponses claires et compréhensibles. En cas de succès, un message de validation est envoyé, tandis qu'en cas d'erreur (comme une image floue, un geste incorrect ou une correspondance faciale insuffisante), un message spécifique est retourné pour guider l'utilisateur vers une nouvelle tentative.
Cette architecture back-end centralisée permet de simplifier la gestion des flux de données, tout en offrant une flexibilité pour ajouter de nouvelles fonctionnalités à l'avenir.

Gestion des données utilisateur

Une fois que l'utilisateur a validé toutes les étapes du processus de vérification et que son identité a été confirmée, le système doit garantir la sécurité des données personnelles collectées. Pour cela, nous utilisons l'algorithme de cryptographie asymétrique **RSA** pour chiffrer les informations sensibles avant de les stocker ou de les transférer.
L'algorithme RSA repose sur un système de clés publiques et privées :
- **Clé publique** : utilisée pour chiffrer les données de l'utilisateur avant leur stockage.
- **Clé privée** : utilisée pour déchiffrer les données, garantissant que seules les parties autorisées peuvent accéder aux informations sensibles.
Après la validation finale de l'identité, le back-end génère un fichier JSON contenant toutes les informations extraites et validées, comme le nom, le prénom, le numéro de CIN et d'autres données personnelles. Ce fichier JSON est ensuite chiffré avec l'algorithme RSA et transmis à l'équipe de développement en charge de l'intégration des données dans la base de données principale.
Pour l'utilisateur, seul un message de succès est retourné, confirmant que son identité a été vérifiée avec succès, sans exposer les données sensibles directement.

```
Encrypted user data: {
  "firstName": "gAAAAABoEJ2Co8VUb9junNuX67F2JnjwVatXMh6aaW2zUeydJ3-rg5oMS1vyeYDePcwh_fxfMG4i2xic9x5IOV3ONhhcMwWVPlhafVEm5TkwMKH
D5yLVmRw=",
  "lastName": "gAAAAABoEJ2Ck5Ohs8w9BFUhQ0b7rskvyzzol3zc52t1zvy7rq5Pg1xO4yuMzo-SKapg4FcxqgPtmnLkby7zr8n3IQmlFPWhEg==",
  "idNumber": "gAAAAABoEJ2CELg8JtnunUzg6j8LRZG7_0BtlRL3nB2j__kXJ8V8PYrFxQzib2FepR79xXtyShTXlUW0sVMddV3ZJvBi4O7Pgg==",
  "birthDate": "gAAAAABoEJ2CokVJhZ-YPozx6ntYtytRl6TasXDW_6WCoE6xN8ygoO2AbV8S7W6UdEl98UmvFW-ThouuUWn9L9OpvrDRDkr-dSNzuHwXm5j3wsZ
f9-JR76M=",
  "birthPlace": "gAAAAABoEJ2CkOhsOh0FcmsE4nOmxqfLTrWMF9qelpU1C7bq2Z3f9noYKhyVCOIxzi9rQNvgaZJtrO7EbbxxOhfB_GxEVTesM3lJ1S9_KBdh5F
WDALp49bk=",
  "job": "gAAAAABoEJ2CcZ6BDrt3E0gjw99QBWXRiE6XjMAmWEEUXvs6XMTwfWRiEjpxsMQyIAAhTGeiWnKiIILBymog0C25vSFuMkvJ1A==",
  "address": "gAAAAABoEJ2CP_e4RwxW3w5VzZKwnP4rNVjOw6Jx3BB4brw_pQu9a_qiOlV9ANj_mSrSz3cTI6kQ0YHrkWmyRxIRppUW8JCqm3LeqGyIWoRi6HTmE
uXeQ40="
}

Verification - Decrypted data matches original: True
```


Fig. 4.26. Cryptage de données.

Conclusion

Identicore représente une solution innovante pour la vérification d'identité, combinant les technologies avancées de détection de texte, reconnaissance faciale et analyse des émotions. En intégrant des modèles de machine learning et des interfaces réactives, il offre une expérience utilisateur sécurisée et fluide, adaptée aux exigences modernes de sécurité numérique. Grâce à cette approche, Identicore permet de renforcer la confiance et la sécurité des processus d'identification, tout en simplifiant l'expérience utilisateur.