UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES

COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN MATHÉMATIQUES ET INFORMATIQUE APPLIQUÉES

PAR
MOUAD KRIM

Reconnaissance biométrique de l'iris : Modélisation hybride pour la segmentation et la classification utilisant des réseaux profonds

DÉCEMBRE 2024

Résumé

L'iris humain est une caractéristique biométrique unique et complexe, offrant un potentiel élevé pour les systèmes d'identification et d'authentification. Grâce à sa variabilité interpersonnelle, la reconnaissance de l'iris est devenue un domaine de recherche clé. Cependant, les conditions d'acquisition dans des environnements non coopératifs posent des défis importants. Ce mémoire propose une approche hybride combinant segmentation et classification pour la reconnaissance de l'iris. Pour la segmentation, un modèle U-Net modifié, intégrant une architecture ResNet34 dans le décodeur, a été utilisé afin d'améliorer la précision de la délimitation des contours de l'iris. Dans la phase de classification, cinq modèles de réseaux de neurones convolutifs (CNN) ont été exploités pour capturer les caractéristiques distinctives de l'iris : ResNet50, VGG16, DenseNet201, ResNet34, et InceptionV3. Afin d'améliorer la qualité des images et de maximiser les performances des modèles, trois techniques de prétraitement des images ont été appliquées : l'égalisation d'histogramme, l'utilisation de filtres de Gabor et l'extraction de motifs binaires locaux (LBP). Les expérimentations ont été menées sur cinq sous-ensembles de la base de données CASIA-IrisV4. Les résultats obtenus montrent une amélioration notable des performances, en termes de stabilité et de précision, grâce à l'approche proposée. Cette étude met en évidence l'efficacité des méthodes de segmentation et de classification utilisées ainsi que leur contribution à l'avancement des systèmes de reconnaissance biométrique de l'iris.

Mots-clés : Iris, Biométrie, CNN, U-Net, ResNet34, LBP.

Mouad Krim

Mhamed Mesfioui

Abstract

The human iris is a unique and complex biometric characteristic, offering significant potential for identification and authentication systems. Due to its high interpersonal variability, iris recognition has become a key research area. However, acquisition conditions in non-cooperative environments pose significant challenges. This thesis proposes a hybrid approach combining segmentation and classification for iris recognition. For segmentation, a modified U-Net model integrating a ResNet34 architecture in the decoder was used to improve the precision of iris boundary delineation. In the classification phase, five convolutional neural network (CNN) models were employed to capture the distinctive features of the iris : ResNet50, VGG16, DenseNet201, ResNet34, and InceptionV3. To enhance image quality and maximize model performance, three image preprocessing techniques were applied : histogram equalization, Gabor filters, and local binary pattern (LBP) extraction. The experiments were conducted on five subsets of the CASIA-IrisV4 database. The results demonstrate a notable improvement in performance, in terms of stability and accuracy, thanks to the proposed approach. This study highlights the effectiveness of the segmentation and classification methods employed and their contribution to advancing biometric iris recognition systems.

Keywords : Iris, Biometrics, CNN, U-Net, ResNet34, LBP.

Remerciements

Je tiens à exprimer ma profonde gratitude à mon professeur, Mhamed Mesfioui, dont l'encadrement, les conseils avisés et le soutien indéfectible ont été essentiels tout au long de la rédaction de ce mémoire. Son expertise et son approche pédagogique m'ont permis d'approfondir mes connaissances et d'affiner ma réflexion sur le sujet. Je suis reconnaissant pour le temps qu'il a consacré à m'orienter et à m'encourager dans mes recherches.

Je souhaite également remercier les évaluateurs "Prof. Nadia Ghazzali" et "Prof. Boucif Amar Bensaber" pour leurs commentaires constructifs et leur rigueur lors de la correction de mon mémoire. Leurs retours ont grandement contribué à l'amélioration de la qualité de mon travail, et je les en remercie chaleureusement.

Je ne peux pas oublier le soutien inestimable de ma famille, qui a toujours cru en moi et m'a encouragé à poursuivre mes études avec détermination. Leur amour et leur compréhension durant les moments difficiles ont été une source de motivation inestimable.

Enfin, un grand merci à mon ami Mohamed Elamine Khoudour. Sa présence et son aide précieuse tout au long de ce parcours ont été d'une grande importance pour moi. Je lui suis reconnaissant pour ses encouragements constants et pour avoir partagé avec moi ses idées et réflexions, ce qui a enrichi mon travail.

Table des matières

Liste des tableaux

Table des figures

Chapitre 1

Introduction

La reconnaissance biométrique consiste à identifier automatiquement des individus en se basant sur des caractéristiques biologiques uniques. Ces identifiants biométriques peuvent être anatomiques, tels que les empreintes digitales ou l'iris, ou comportementaux, comme la voix. En comparaison avec les méthodes d'authentification traditionnelles, telles que les mots de passe et les cartes d'identité, les systèmes biométriques offrent une sécurité et une fiabilité supérieures. Les caractéristiques biométriques, étant inhérentes à chaque individu, présentent une difficulté significative à falsifier, contrairement aux mots de passe, qui peuvent être compromis, ou aux cartes d'identité, qui peuvent être volées.

Parmi les différentes modalités biométriques, l'iris de l'œil se distingue par sa fiabilité et sa complexité anatomique, résultant en une grande variabilité interindividuelle [1]. Cette spécificité en fait un choix privilégié pour les systèmes de contrôle d'accès et d'authentification, tant dans les secteurs civils que gouvernementaux. Les technologies de reconnaissance de l'iris, qui s'appuient sur des algorithmes avancés d'apprentissage automatique et d'analyse d'images, garantissent une précision élevée et permettent une identification rapide.

L'iris, qui est une membrane circulaire située entre la cornée et le cristallin, joue un rôle essentiel dans la régulation de la lumière grâce à l'ajustement de la pupille. En plus de sa fonction physiologique, l'iris possède des motifs uniques qui restent constants tout au long de la vie d'un individu, ce qui le rend particulièrement adapté à l'identification individuelle [2]. Comme illustré dans la Figure 1.1.A, l'iris se compose de plusieurs zones, chacune ayant des fonctions spécifiques :

— **Zone ciliaire** : En contact avec le corps ciliaire, elle contribue à l'adaptation visuelle.
— **Zone clorette** : Région claire entourant la pupille, améliorant le contraste visuel.
— **Zone pupilaire** : Entoure la pupille et régule la quantité de lumière entrant dans l'œil.

Ces zones s'organisent en cercles fonctionnels qui illustrent les processus visuels, comme le montre la Figure 1.1.B :

— **Cercle de digestion** : Facilite l'adaptation à la luminosité ambiante.
— **Cercle d'utilisation** : Optimise l'accommodation pour une vision claire.
— **Cercle d'élimination** : Protège la rétine des lumières excessives.

La littérature sur l'authentification biométrique révèle une utilisation croissante des techniques d'apprentissage automatique, notamment dans la reconnaissance faciale. Ces approches ont suscité un intérêt notable au sein de la communauté scientifique, visant à résoudre des problèmes tant d'apprentissage supervisé que non supervisé [3]. Les réseaux de neurones convolutionnels (CNN) sont parmi les méthodes les plus répandues, utilisés pour l'extraction des caractéristiques, la segmentation et la classification. Ces techniques ont démontré leur efficacité et leur performance dans divers domaines d'application, allant de la sécurité aux systèmes de santé.

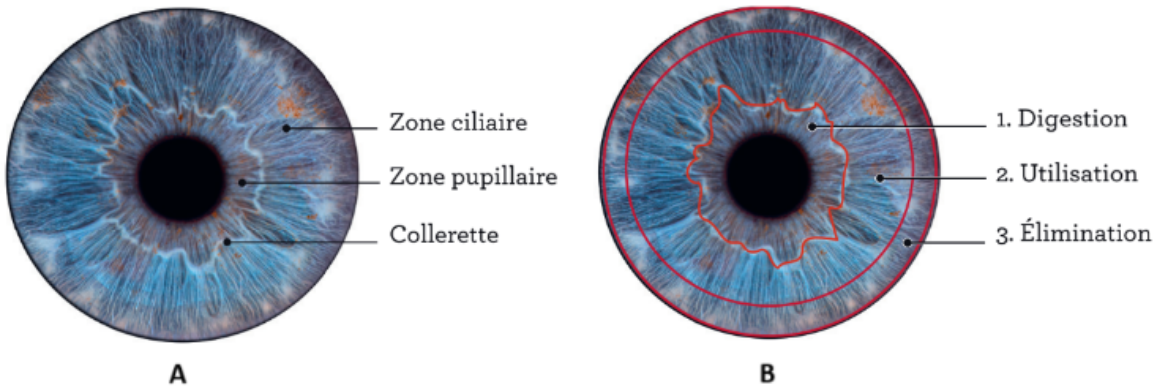

FIGURE 1.1 – Anatomie de l'iris de l'œil.

Cependant, malgré les résultats prometteurs, plusieurs défis demeurent en raison de la diversité des ensembles de données, incluant des variations dans les conditions d'éclairage, les angles de vue et les expressions faciales, ainsi que des facteurs culturels et de genre. Ces éléments compliquent la robustesse des modèles développés et soulèvent des questions quant à leur généralisation. De plus, le choix et l'optimisation des hyperparamètres des CNN sont cruciaux [4]. Un réglage inapproprié peut diminuer la précision et nuire à l'adaptabilité des modèles face à de nouvelles données. Il est donc essentiel d'ajuster soigneusement les modèles selon les particularités des données pour assurer leur fiabilité dans des contextes variés.

Ce mémoire propose une approche hybride combinant des techniques de segmentation de localisation et de classification a été proposée pour la reconnaissance de l'iris. Pour la segmentation et la localisation, un modèle U-Net amélioré a été employé, intégrant une architecture ResNet34 dans le décodeur afin d'optimiser la précision dans la délimitation des contours de l'iris et l'élimination des artéfacts. Quant à la classification, cinq architectures de réseaux de neurones convolutifs (CNN) ont été utilisées pour extraire les caractéristiques distinctives de l'iris : ResNet50, VGG16, DenseNet201, ResNet34 et InceptionV3.

Pour renforcer la qualité des images et optimiser les performances des modèles, trois techniques de prétraitement ont été mises en œuvre : l'égalisation d'histogramme pour améliorer le contraste, les filtres de Gabor pour capturer les détails texturaux, et l'extraction de motifs binaires locaux (LBP) pour révéler les caractéristiques clés de l'iris. Les expérimentations ont été effectuées sur cinq sous-ensembles de la base de données CASIA-IrisV4, fournissant un cadre rigoureux pour évaluer l'efficacité de l'approche proposée.

Le chapitre suivant présente une revue de la littérature, mettant en avant les études précédentes sur la reconnaissance de l'iris. Le chapitre 3 est consacré à la méthodologie adoptée, incluant les approches de segmentation, localisation et classification. Les résultats et leur discussion sont détaillés dans le chapitre 4, mettant en évidence les contributions et perspectives de cette recherche. Enfin, le chapitre 5 conclut le mémoire avec des perspectives pour les travaux futurs.

Chapitre 2

Revue de littérature

Les méthodes d'apprentissage profond, notamment les différentes architectures de réseaux de neurones convolutifs (CNN), ont significativement amélioré de nombreuses applications de vision par ordinateur au cours de la dernière décennie. Dans le domaine des technologies biométriques, il est logique que la reconnaissance de l'iris ait également adopté des approches purement basées sur les données à chaque étape du processus de reconnaissance, du prétraitement à la segmentation, en passant par le codage et l'appariement. Toutefois, il est intéressant de constater que l'impact de l'apprentissage automatique profond varie selon les différentes étapes du pipeline de reconnaissance de l'iris. Le tableau 2.1 résume les principales études sur la reconnaissance de l'iris.

2.1 Segmentation et localisation de l'iris

Schlett et al. [5] ont proposé une approche d'analyse multi-spectrale pour améliorer la précision de la segmentation de l'iris dans le spectre visible. Leur méthode consiste

à prétraiter les données avant la segmentation en extrayant plusieurs composantes spectrales sous forme de canaux de couleurs RGB. Bien que leur approche utilise l'apprentissage profond, ces différentes versions des données d'entrée peuvent être intégrées dans des modèles d'apprentissage profond afin d'accroître la robustesse face à des données imparfaites. Ils ont utilisé le dataset CASIA pour évaluer la précision de leur approche. Toutefois, cette étude ne fournit pas d'évaluation approfondie de la performance du modèle sur des bases de données diversifiées, limitant ainsi la généralisation des résultats à d'autres contextes.

Wu et Zhao [6] ont présenté le modèle Dense U-Net, qui intègre des couches denses au sein de l'architecture U-Net. L'objectif principal est de profiter de l'ensemble réduit de paramètres offert par le Dense U-Net tout en préservant les capacités de segmentation sémantique caractéristiques du U-Net. Le modèle proposé intègre une connectivité dense dans les chemins de contraction et d'expansion de l'architecture U-Net, ce qui permet une meilleure exploitation des informations à chaque étape du processus de segmentation. Leurs expérimentations ont été menées sur la base de données UBIRIS.v2. Par rapport aux réseaux de neurones convolutifs (CNN) traditionnels, ce modèle vise à réduire la redondance d'apprentissage et à améliorer le flux d'information entre les différentes couches, tout en maintenant un contrôle strict sur le nombre de paramètres du modèle.

Ganeva et Myasnikov [7] ont proposé une approche visant à résoudre le problème de la segmentation des images d'iris. Ils ont particulièrement étudié trois architectures de réseaux de neurones convolutionnels : U-Net, LinkNet et FC-DenseNet. Dans le cadre de leur étude, ils ont déterminé des paramètres optimaux pour l'entraînement des réseaux de neurones et utilisé des techniques d'augmentation pour améliorer la précision de la segmentation. Les expérimentations présentées dans cet article ont été réalisées sur la base de données ouverte MMU Iris Database. Les résultats obtenus montrent que les trois architectures de réseaux de neurones offrent une haute précision de segmentation, les meilleurs résultats ayant été atteints avec U-Net.

Li et al. [8] ont présenté une méthode hybride combinant des informations basées sur les contours avec des cadres d'apprentissage profond, en utilisant une architecture compacte de type Faster R-CNN pour détecter la région d'intérêt (ROI) de l'œil. La méthode IRU-Net, s'appuyant également sur des réseaux de type Faster R-CNN, permet de localiser avec précision cette région, facilitant la segmentation de l'iris dans divers environnements tout en améliorant la précision grâce à un apprentissage supervisé.Cette approche a été testée sur la base de données CASIA-IrisV4. Une fois la région d'intérêt définie, la localisation de la pupille est affinée à l'aide d'un modèle de mélange gaussien, renforçant ainsi la fiabilité de la détection. Cependant, cette approche présente une complexité computationnelle élevée liée à l'utilisation conjointe de cette approche.

Wang et al. [9] ont proposé une méthode de reconnaissance de l'iris basée sur l'apprentissage profond et le transfert de connaissances, visant à améliorer la capacité de généralisation du modèle pour une reconnaissance plus adaptable. Contrairement aux méthodes traditionnelles, qui nécessitent une forte coopération de l'utilisateur et des conditions d'imagerie strictes, cette approche permet d'entraîner un modèle initial en utilisant un ensemble de données diversifié, couvrant différentes origines, institutions et cas d'occlusion. Une fois ce modèle pré-entraîné, il est affiné sur un sous-ensemble de données spécifiques pour améliorer la précision et l'adaptabilité du modèle final aux données cibles. Testée dans le cadre du concours NIR-ISL 2021, cette méthode s'est classée parmi les trois meilleures, confirmant son efficacité et sa robustesse face aux variations des conditions d'imagerie et de diversité des données.

Viktor Varkarakis et al. [10] ont proposé une méthodologie d'augmentation de données pour générer un vaste ensemble d'images d'iris en décalage par rapport à l'axe frontal, destiné à l'entraînement d'un réseau neuronal profond de faible complexité. Malgré sa simplicité, le réseau obtenu atteint un haut niveau de précision dans la segmentation des régions d'iris sur des images d'yeux prises sous des angles difficiles. De manière intéressante, ce réseau montre également de très bonnes perfor-

mances pour la segmentation d'iris en vue frontale, se comparant avantageusement aux techniques plus complexes et sophistiquées. Grâce à sa faible complexité, ce réseau est particulièrement adapté aux applications embarquées, telles que les casques de réalité augmentée et mixte.

2.2 Reconnaissance basée sur l'apprentissage profond

Boyd et al. [11] ont évalué la performance du réseau ResNet50 pour l'extraction de caractéristiques spécifiques à l'iris. Ce modèle a montré des résultats notables dans des tâches de reconnaissance, particulièrement avec des données d'iris post-mortem, ce qui en fait une approche prometteuse pour des applications médico-légales. Ils ont utilisé la base de données ND-Iris-0405 et une base de données post-mortem collectée en interne pour leur étude. Cependant, bien que performant, l'utilisation de ResNet50 présente certains défis : le modèle nécessite une puissance de calcul considérable et peut être difficile à optimiser pour des images d'iris de qualité variable, souvent rencontrées dans des contextes médico-légaux. Ces limitations pourraient restreindre son adoption dans des environnements à ressources limitées ou dans des cas où la rapidité est cruciale.

Minaee et al. [12] ont testé le réseau VGG-Net pour la reconnaissance de l'iris et ont constaté que les caractéristiques profondes extraites par ce modèle pouvaient être efficacement transférées pour des tâches biométriques, confirmant ainsi sa polyvalence. Les résultats obtenus ont été basés sur des expérimentations menées sur la base de données UBIRIS.v2. Cependant, bien que VGG-Net montre des capacités de transfert impressionnantes, il présente aussi des limites. Sa structure profonde entraîne une complexité computationnelle élevée, rendant son déploiement difficile dans des environnements à ressources limitées. De plus, sa grande profondeur peut le rendre sensible aux problèmes de surapprentissage, en particulier lorsque les données disponibles

pour la reconnaissance de l'iris sont restreintes ou de qualité inégale, ce qui limite son potentiel d'application sans des ajustements et des optimisations supplémentaires.

Nguyen et al. [13] ont exploré des modèles CNN pré-entraînés et ont démontré que les caractéristiques génériques extraites par ces modèles sont tout aussi efficaces pour représenter les images d'iris, renforçant ainsi l'applicabilité de l'apprentissage profond dans la reconnaissance biométrique. Ils ont testé leur approche sur les bases de données CASIA-IrisV4 et IITD. Toutefois, cette approche présente certaines limitations. Les caractéristiques génériques des modèles CNN peuvent manquer de spécificité pour les particularités complexes des images d'iris, notamment dans des conditions de faible éclairage ou pour des images de qualité variable. De plus, en utilisant des modèles pré-entraînés, il peut être difficile d'adapter le réseau aux variations propres aux données biométriques d'iris sans entraîner une perte de précision, soulignant la nécessité d'un ajustement fin pour garantir des performances optimales.

Zhao et al. [14] ont proposé un modèle basé sur les réseaux de capsules pour l'extraction de caractéristiques primaires de l'iris, améliorant ainsi la robustesse du modèle face aux variations inhérentes de cette biométrie. Ils ont validé leur approche sur la base de données UBIRIS.v2. La conception de ce modèle utilise des capsules convolutives, permettant de minimiser le nombre de paramètres tout en maintenant une haute précision, ce qui est particulièrement bénéfique pour des applications nécessitant des ressources limitées. Cependant, bien que prometteuse, cette approche présente certains défis : les réseaux de capsules sont connus pour être plus difficiles à entraîner et à optimiser en comparaison avec les architectures CNN traditionnelles. De plus, leur complexité algorithmique peut rendre l'implémentation en temps réel plus difficile, surtout lorsque l'on travaille avec de grandes quantités de données ou des systèmes aux ressources limitées.

Savithiri et al. [15] ont comparé diverses techniques d'extraction de caractéristiques pour la reconnaissance de l'iris en testant un demi-modèle de l'iris, visant à réduire

la zone d'analyse tout en maintenant des performances élevées. Ils ont appliqué les méthodes GW, HOG et LBP pour extraire des caractéristiques spécifiques, évaluant leur efficacité avec un classificateur HD. Pour leurs expérimentations, ils ont utilisé 130 images de la base de données MMU, réparties en 26 classes, avec une image de chaque classe comme modèle et les quatre restantes pour les tests. Les résultats montrent des taux de reconnaissance élevés, atteignant 99,95 % pour HOG+HD, 99,94 % pour GW+HD, et 99,90 % pour LBP+HD, tout en maintenant un faible taux de fausse acceptation (FAR), démontrant ainsi l'efficacité de l'approche demi-modèle pour la reconnaissance de l'iris.

Saminathan et al. [16] ont proposé une méthode de reconnaissance biométrique de l'iris pour garantir la sécurité dans les systèmes d'authentification et d'identification. Cette approche repose sur les algorithmes de classification de type machine learning, en utilisant une machine à vecteurs de support (SVM) pour l'identification personnelle. Les motifs riches et uniques de chaque iris sont capturés et stockés sous forme de matrice contenant 4800 éléments par iris, dont 2400 éléments sont transmis sous forme de vecteurs ligne au classificateur SVM. Le SVM crée des classes distinctes pour chaque utilisateur et réalise la correspondance en se basant sur les caractéristiques spectrales uniques de l'iris. Les résultats expérimentaux montrent une performance supérieure de 98,5 %, surpassant les méthodes existantes comme la distance de Hamming, les motifs binaires locaux, et divers noyaux de SVM. Les expériences menées sur la base de données CASIA (Chinese Academy of Sciences – Institute of Automation) avec des échantillons d'iris de cinquante utilisateurs démontrent que la méthode SVM avec noyau quadratique utilisant la méthode des moindres carrés offre un taux de rejet minimal et une meilleure précision.

Zhao et al. [14] ont proposé une méthode de reconnaissance de l'iris utilisant l'architecture des réseaux de capsules, adaptée pour renforcer la robustesse du modèle face aux variations d'environnement et au bruit. Ils ont modifié la structure du réseau en introduisant un algorithme de routage dynamique entre les couches de capsules

pour améliorer la précision de la reconnaissance de l'iris. Pour optimiser l'extraction des caractéristiques, ils ont intégré trois modèles pré-entraînés – VGG16, InceptionV3 et ResNet50 – comme sous-réseaux, en remplaçant une simple couche convolutionnelle dans le réseau de capsules. Les expérimentations ont été réalisées sur trois bases de données d'iris, JluIrisV3.1, JluIrisV4 et CASIA-V4 Lamp, afin d'évaluer la performance des différentes architectures proposées. Les résultats montrent que le réseau de capsules est plus stable et performant que les réseaux conventionnels dans des environnements de lumière variable, démontrant ainsi son efficacité pour la reconnaissance de l'iris.

2.3 Détection des attaques de présentation

Sharma et Ross [17] ont utilisé DenseNet121 avec des modules d'attention canaux et spatiaux pour renforcer la précision du modèle face aux attaques de présentation par lentilles et yeux artificiels, rendant ainsi le modèle plus résistant dans des contextes variés. Ils ont utilisé la base de données Notre Dame Liveness Detection-Iris 2013 pour tester la robustesse de leur modèle. L'ajout de ces modules d'attention permet de focaliser le modèle sur des caractéristiques pertinentes de l'iris, améliorant sa capacité à distinguer les images réelles des tentatives de falsification. Cependant, cette architecture présente des défis. L'intégration de modules d'attention peut augmenter la complexité et le coût computationnel, ce qui pourrait limiter son efficacité dans des environnements à ressources limitées ou nécessitant une exécution en temps réel. De plus, la complexité accrue du modèle pourrait exiger un jeu de données étendu pour un entraînement efficace, posant un défi dans les situations où des données authentiques d'iris sont rares.

Chen et Ross [18] ont introduit un détecteur guidé par l'attention, nommé AG-PAD, basé sur DenseNet121, exploitant une attention multi-dimensionnelle pour iden-

tifier efficacement les indices de fraude. Le modèle a été testé sur les bases de données ND-Iris-0405 et LivDet-Iris 2020. Leur approche s'est montrée performante dans divers environnements, en particulier sur des bases de données aux caractéristiques variées, ce qui en fait un modèle adaptable à des conditions réelles. Cependant, bien que puissant, AG-PAD présente certains limites. L'attention multi-dimensionnelle augmente la complexité du modèle, ce qui peut alourdir les besoins en calcul et rendre l'entraînement plus exigeant en ressources. De plus, l'efficacité de cette approche dépend fortement de la qualité et de la diversité des données d'entraînement, ce qui pourrait limiter les performances du modèle dans des environnements où les données d'entrées sont moins représentatives.

Tapia et al. [19] ont conçu une architecture basée sur MobileNetv2 pour la détection de diverses attaques de présentation, notamment les lentilles texturées, les impressions, et les yeux post-mortem. Grâce à l'intégration de techniques avancées de prétraitement et de data augmentation, leur modèle a démontré une efficacité dans la détection de ces attaques, tout en restant léger et optimisé pour les dispositifs à faibles ressources. Cependant, malgré ses avantages, cette architecture pourrait rencontrer des limites dans des environnements extrêmement variés, où les variations de qualité d'image et d'éclairage sont importantes. De plus, MobileNetv2, bien qu'efficace, peut sacrifier une certaine précision par rapport aux architectures plus profondes, ce qui pourrait poser un défi pour détecter des attaques de présentation particulièrement sophistiquées.

2.4 Reconnaissance post-mortem

Trokielewicz et Czajka. [20] ont développé un modèle spécifique pour la reconnaissance de l'iris post-mortem, combinant SegNet pour la segmentation et des réseaux siamois pour l'extraction de caractéristiques. Ce modèle, optimisé exclusivement pour des échantillons d'iris post-mortem, a permis une normalisation efficace et une ex-

clusion des artefacts dus à la décomposition, rendant le processus de reconnaissance plus précis. Cependant, l'optimisation de ce modèle exclusivement pour des iris post-mortem pourrait limiter sa capacité à traiter des échantillons d'iris vivants, réduisant ainsi sa polyvalence. De plus, la complexité de cette étude nécessite un prétraitement et un ajustement des paramètres, ce qui pourrait limiter son application dans des contextes où une reconnaissance rapide et automatisée est nécessaire.

Kuehlkamp et al. [21] ont utilisé ResNet50 avec une approche de visualisation pour la classification et introduit des cartes d'activation des classes (CAM) afin d'aider les examinateurs humains à localiser les zones de décision importantes. Cette technique renforce ainsi la valeur médico-légale des images d'iris post-mortem, permettant une interprétation visuelle plus détaillée des zones contribuant aux décisions de reconnaissance. Cependant, bien que les CAM ajoutent une transparence au processus de classification, leur interprétation peut varier selon les examinateurs, introduisant ainsi une part de subjectivité. De plus, l'utilisation de ResNet50 avec des CAM augmente la complexité du modèle, ce qui peut engendrer un surcoût computationnel, potentiellement difficile à gérer dans des environnements où les ressources sont limitées.

Les points forts de ces études soulignent des avancées significatives dans chaque étape du pipeline de reconnaissance de l'iris. En segmentation, Dense U-Net, grâce à sa connectivité dense, optimise le flux d'informations tout en maintenant un faible nombre de paramètres, permettant ainsi une segmentation sémantique précise, même avec des ressources réduites. Pour l'extraction de caractéristiques de l'iris, ResNet50 démontre des performances, particulièrement dans des applications médico-légales post-mortem, bien qu'il puisse être limité par ses besoins en calcul intensif. En ce qui concerne la détection des attaques de présentation, DenseNet121, enrichi de modules d'attention canaux et spatiaux, permet de concentrer l'attention sur des caractéristiques distinctives de l'iris, augmentant ainsi la précision face à des attaques sophistiquées telles que les lentilles texturées et les yeux artificiels.

TABLE 2.1: Reconnaissance de l'iris

Type d'étude	**Auteurs**	**Modèle utilisé**	**Base de données**
Segmentation de l'iris	Schlett et al. [5]	Multi-spectral Analysis	CASIA
	Wu et Zhao [6]	Dense U-Net	UBIRIS.v2
	Ganeva et Myasnikov [7]	U-Net, LinkNet, FC-DenseNet	MMU Iris Database
	Li et al. [8]	Faster R-CNN, IRU-Net	CASIA-IrisV4
	Varkarakis et al. [10]	Simple Neural Network	Casques AR/MR
Reconnaissance basée sur l'apprentissage profond	Boyd et al. [11]	ResNet50	ND-Iris-0405 et interne
	Minaee et al. [12]	VGG-Net	UBIRIS.v2
	Nguyen et al. [13]	Pre-trained CNNs	CASIA-IrisV4, IITD
	Zhao et al. [14]	Capsule Networks	UBIRIS.v2
	Saminathan et al. [16]	SVM	CASIA
	Zhao et al. [14]	Capsule Networks with Dynamic Routing	JluIrisV3.1, JluIrisV4, CASIA-V4 Lamp

Type d'étude	**Auteur(s)**	**Modèle utilisé**	**Base de données**
Détection des attaques de présentation	Sharma et Ross [17]	DenseNet121 with Attention Modules	Notre Dame Liveness Detection-Iris 2013
	Chen et Ross [18]	AG-PAD	ND-Iris-0405, LivDet-Iris 2020
	Tapia et al. [19]	MobileNetv2	Lentilles, impressions, yeux post-mortem
Reconnaissance post-mortem	Trokielewicz et Czajka. [20]	SegNet, Siamese Network	ND-Iris-0405 et interne
	Kuehlkamp et al. [21]	ResNet50 with CAM	ND-Iris-0405 et post-mortem

Chapitre 3

Méthodologie

Notre approche, illustrée dans la Figure3.1 , structure le processus d'identification des individus en trois étapes essentielles : préparation des données, extraction des caractéristiques et identification. Dans un premier temps, les images brutes de l'œil sont traitées pour isoler avec précision l'iris en utilisant un modèle de segmentation avancé basé sur UNet et ResNet34, couplé à un modèle de localisation qui identifie précisément les contours internes et externes de l'iris, assurant une extraction exacte de la région d'intérêt. Une fois l'iris segmenté, l'image est soumise à des processus de normalisation et d'égalisation d'histogramme pour améliorer sa qualité visuelle, optimisant ainsi l'extraction des caractéristiques en réduisant le bruit et les variations d'éclairage. La deuxième étape exploite une combinaison de modèles avancés de réseaux de neurones profonds, notamment ResNet50, ResNet34, VGG16, InceptionV3 et DenseNet201, pour capturer des attributs riches et distinctifs de l'iris, renforçant la robustesse et la précision de la reconnaissance. Enfin, l'étape d'identification utilise les caractéristiques extraites pour classifier l'individu correspondant à l'iris analysé, garantissant une reconnaissance fiable et précise des individus, essentielle pour des applications de sécurité et de biométrie nécessitant un niveau élevé de fiabilité et de précision.

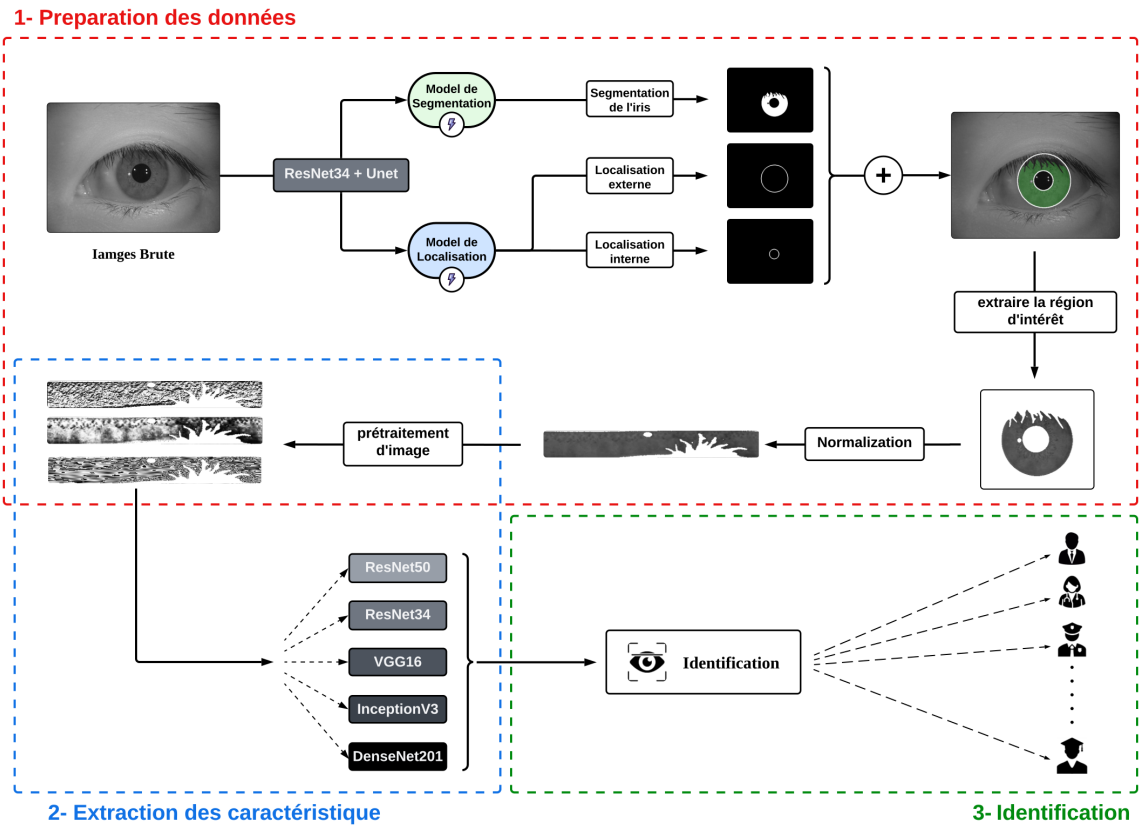

FIGURE 3.1 – Méthodologie.

3.1 Dataset

Pour notre étude, nous avons utilisé cinq ensembles de données différents, chacun apportant des spécificités uniques pour l'analyse de la reconnaissance de l'iris dans diverses conditions. Ces ensembles de données font partie de la collection SIR-Smart Iris Recognition, qui regroupe plusieurs sous-ensembles de la base de données CASIA-IrisV4. Cette collection permet d'explorer des problématiques telles que les variations d'éclairage, les angles de capture et les occlusions. La diversité des données nous a permis de tester la robustesse et la précision de notre modèle dans divers scénarios, garantissant un système de reconnaissance de l'iris fiable.Chaque sous-ensemble de SIR offre des caractéristiques uniques, rendant cette collection particulièrement utile. Le Tableau 3.1 montre le nombre d'échantillons utilisés pour chaque ensemble de

données. Pour plus de détails, consultez le lien suivant [1]. La Figure 3.2 illustre des échantillons des bases de données utilisées.

TABLE 3.1 – Résumé des ensembles de données.

Ensemble de données	**Classes**	**Total**	**Dimensions**
CASIA-Iris-Lamp	411	16 214	640 x 480
CASIA-Iris-Thousand	1000	20 000	640 x 480
CASIA-Iris-Asia	60	1395	640 x 480
CASIA-Iris-M1	200	3000	400 x 400
CASIA-Iris-Africa	370	740	1088 x 640

3.1.1 CASIA-Iris-Lamp

L'ensemble de données CASIA-Iris-Lamp a été collecté à l'aide d'un capteur d'iris portable fabriqué par OKI. Lors de la collecte, une lampe a été allumée et éteinte à proximité du sujet afin d'introduire des variations intra-classes plus marquées. Cette configuration a permis de simuler des déformations élastiques de la texture de l'iris causées par la dilatation et la contraction de la pupille sous différentes conditions d'éclairage. Ces effets sont essentiels pour étudier des défis tels que la normalisation non linéaire de l'iris et la représentation robuste des caractéristiques de l'iris. En raison de cette variabilité, CASIA-Iris-Lamp est particulièrement adapté à l'étude de la reconnaissance d'iris dans des conditions de lumière changeante. Ce jeu de données est souvent utilisé pour tester la résilience des modèles de reconnaissance face aux variations d'intensité lumineuse. Il constitue une base précieuse pour développer des solutions robustes aux changements environnementaux.

1. `https://hycasia.github.io/dataset/casia-irisv4/`

3.1.2 CASIA-Iris-Thousand

L'ensemble de données CASIA-Iris-Thousand contient 20 000 images d'iris issues de 1 000 sujets, capturées avec une caméra à double objectif IKEMB-100 produite par IrisKing. Cette caméra fournit un retour visuel pour aider les utilisateurs à ajuster leur pose, garantissant ainsi une capture d'image de haute qualité. Les variations intra-classes principales dans ce jeu de données proviennent des lunettes et des réflexions spéculaires. En tant que premier ensemble de données d'iris disponible publiquement avec un millier de sujets, CASIA-Iris-Thousand est idéal pour étudier l'unicité des caractéristiques de l'iris et pour développer de nouvelles méthodes de classification et d'indexation de l'iris. La grande diversité de sujets renforce la fiabilité des modèles pour des applications de grande échelle. De plus, il permet de valider des approches pour des bases de données massives en reconnaissance biométrique.

3.1.3 CASIA-Iris-Asia

L'ensemble de données CASIA-Iris-Asia regroupe des images en infrarouge proche (NIR) de sujets asiatiques capturées dans des environnements non coopératifs. Il inclut des images issues de CASIA-Iris-Distance, collectées avec une caméra longue portée, et de CASIA-Iris-Complex, capturées dans des environnements intérieurs complexes avec un Canon EOS 1300D modifié. Ce jeu de données comporte plusieurs sources de bruit, y compris des occlusions dues aux paupières, aux cils et aux lunettes, ainsi que des images prises sous des angles décalés. CASIA-Iris-Asia est organisé en sous-ensembles spécifiques, tels que CASIA-Iris-Complex-Occlusion (500 images iris avec occlusion) et CASIA-Iris-Complex-Off-angle (500 images hors angle). Cet ensemble est adapté pour des études sur la segmentation et la localisation de l'iris, avec des annotations manuelles fournies pour les contours de l'iris et les masques de segmentation.

3.1.4 CASIA-Iris-M1

CASIA-Iris-M1 est un ensemble de données mobile à grande échelle composé de 11 000 images provenant de 630 sujets asiatiques, répartis en trois sous-ensembles : CASIA-Iris-M1-S1 (1 400 images, 70 sujets), CASIA-Iris-M1-S2 (6 000 images, 200 sujets) et CASIA-Iris-M1-S3 (3 600 images, 360 sujets). Les images de S1 et S2 ont été capturées avec des modules d'imagerie en NIR attachés à des téléphones mobiles, tandis que celles de S3 proviennent d'un téléphone équipé de la technologie de balayage d'iris en NIR. Les principales sources de variabilité incluent un éclairage irrégulier, différentes tailles d'iris, et des occlusions dues aux lunettes. Avec des annotations précises pour la segmentation de l'iris, CASIA-Iris-M1 est conçu pour le développement de modèles de reconnaissance d'iris adaptés aux environnements mobiles.

3.1.5 CASIA-Iris-Africa

CASIA-Iris-Africa est le premier ensemble de données d'iris de grande envergure capturé en Afrique, spécifiquement au Nigéria, avec une résolution de 1088×640 pixels sous illumination en NIR. Cet ensemble introduit des facteurs de bruit, tels que des occlusions, des angles décalés, des variations d'éclairage et une diversité de tons de peau et de couleurs d'iris sombres, qui rendent la segmentation et la localisation de l'iris particulièrement complexes. Dans le cadre de la collecte, 370 images d'iris ont été sélectionnées pour l'entraînement et un autre ensemble distinct de 370 images pour les tests. Les masques de segmentation et les limites internes et externes de l'iris ont été annotés manuellement, ce qui fait de CASIA-Iris-Africa un choix pertinent pour étudier la reconnaissance de l'iris dans des conditions d'acquisition moins contraignantes.

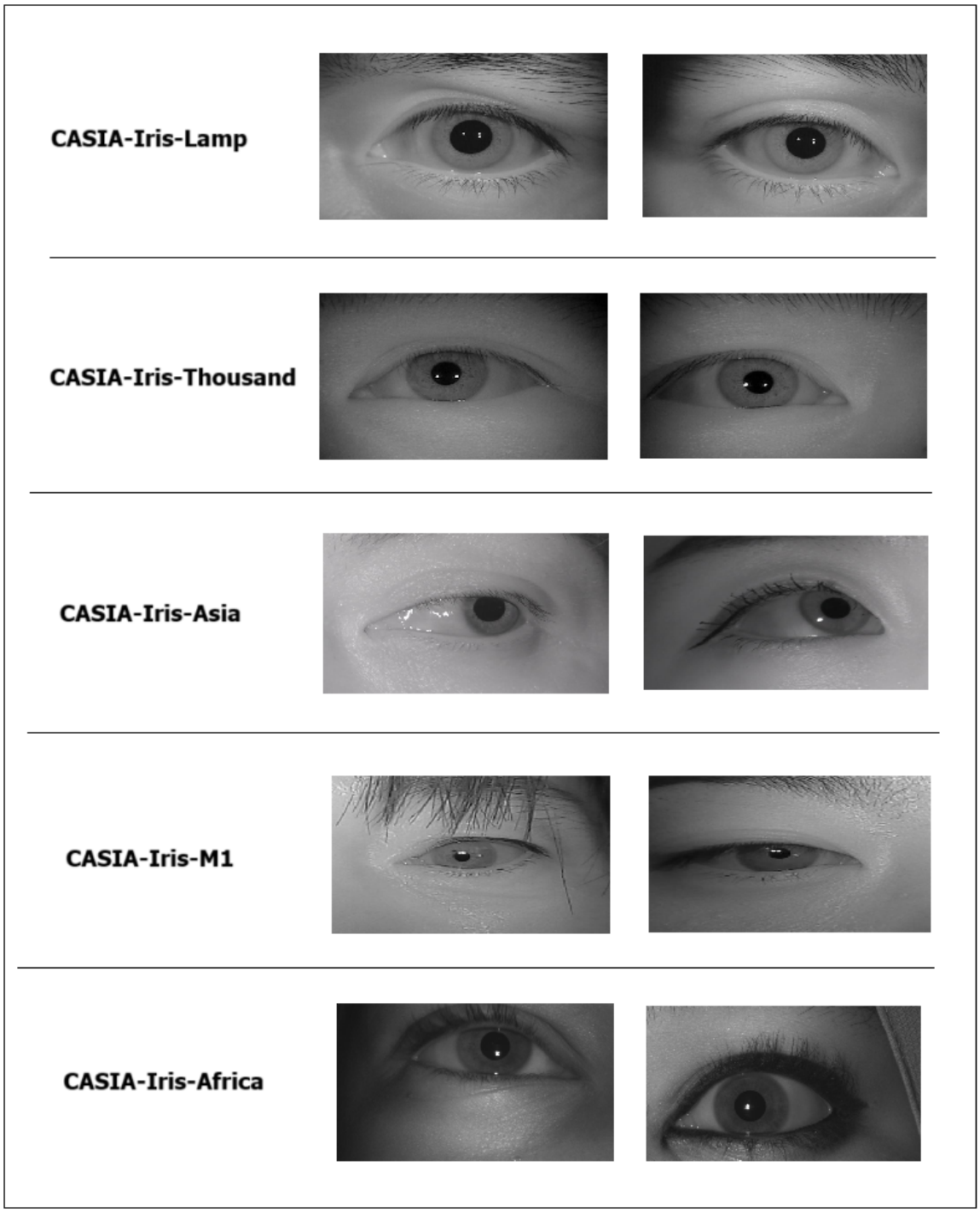

FIGURE 3.2 – Échantillons des bases de données utilisées.

3.2 Localisation et segmentation de l'iris

L'extraction de caractéristiques est essentielle pour obtenir une classification biométrique nette et précise, l'iris étant une région cruciale contenant des informations uniques pour l'identification. Dans des environnements non coopératifs, la segmentation de l'iris reste un défi majeur, impactant toutes les étapes du processus de reconnaissance. Avec l'essor de l'apprentissage profond, de nouvelles méthodes ont été introduites pour améliorer la segmentation de l'iris. En 2021, Caiyong Wang et ses collègues [9] ont organisé le **NIR Iris Challenge Evaluation (NIR-ISL 2021)** lors de la conférence IJCB, offrant une plateforme pour évaluer des méthodes de segmentation et de localisation sur des images NIR d'iris capturées dans des environnements difficiles, avec des participants asiatiques et africains. Les trois bases de données utilisées dans cette compétition sont CASIA-Iris-Asia, CASIA-Iris-Africa et CASIA-Iris-M1.

Pour le défi NIR Iris Challenge Evaluation (NIR-ISL 2021), Yiwen Zhang, Tianbao Liu et Wei Yang. [2] ont remporté la première place en utilisant une architecture basée sur le réseau UNet, avec ResNet34 comme backbone. Après traitement par ResNet34, des cartes de caractéristiques réduites à 1/32 de la taille de l'image sont obtenues, puis redimensionnées par le décodeur pour générer la segmentation. La différence entre le réseau de segmentation et celui de localisation réside dans la dernière couche : le premier produit un masque de l'iris, tandis que le second génère des masques des limites interne et externe de l'iris. La fonction sigmoid crée une carte de probabilité pour distinguer l'iris de l'arrière-plan, et les fonctions de perte, combinant dice loss et binary cross-entropy loss, améliorent la précision des résultats. Cette approche sera utilisée dans notre étude pour segmenter et localiser l'iris de l'œil, car notre choix repose sur l'efficacité démontrée de cette méthode dans des conditions non coopératives, assurant une segmentation robuste et fiable.

2. `https://github.com/xiamenwcy/NIR-ISL-2021`

UNet, développée par Ronneberger et al. [22], est un réseau de neurones convolutionnel destiné à la segmentation d'images et particulièrement adapté aux tâches nécessitant une classification précise au niveau de chaque pixel. Elle s'organise en deux branches distinctes : un chemin contractant **encodeur** et un chemin expansif **décodeur**, interconnectés par des connexions de saut (skip connections) qui permettent de préserver des informations de localisation cruciales. Le chemin contractant applique des opérations de convolution et de max-pooling pour réduire la résolution spatiale des cartes de caractéristiques tout en capturant les informations contextuelles de l'image. Le chemin expansif utilise des convolutions transposées pour restaurer la résolution spatiale d'origine, intégrant les informations spatiales par le biais des connexions de saut. Grâce à cette architecture en forme de "U", UNet parvient à combiner efficacement les informations globales et les détails fins nécessaires pour des segmentations précises.

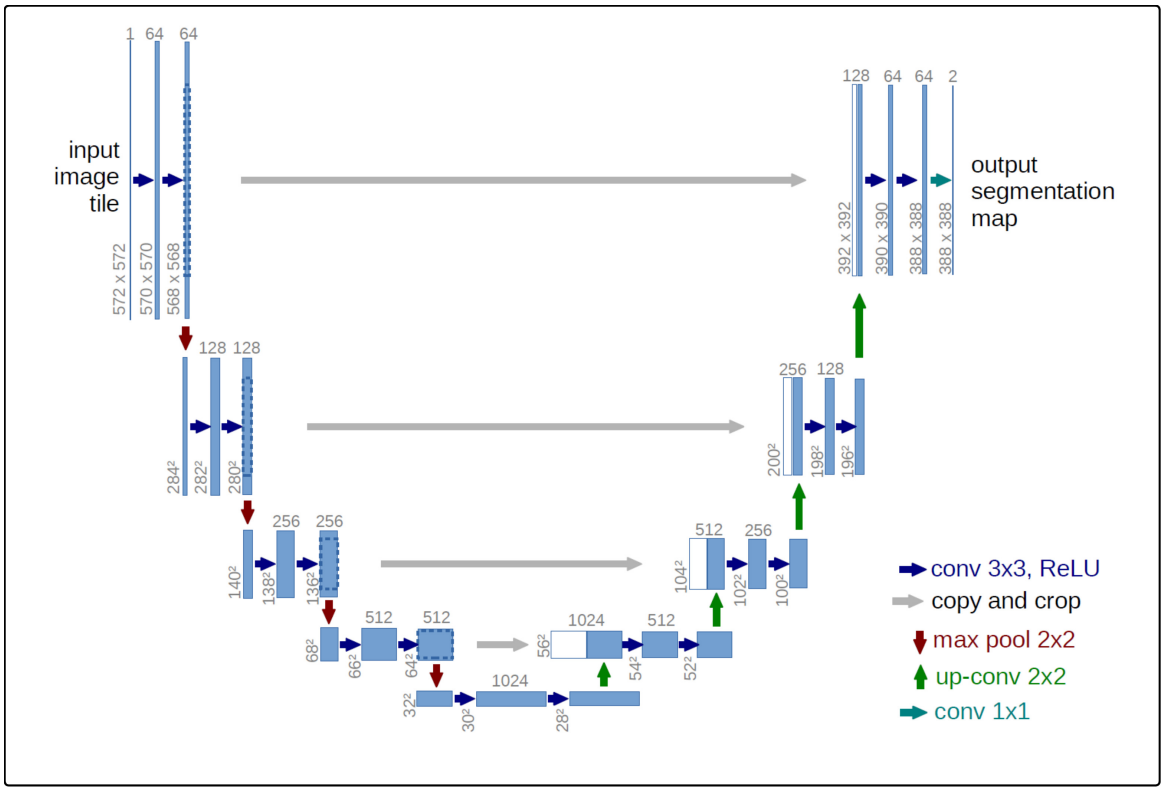

FIGURE 3.3 – Architecture UNet.

UNet utilise plusieurs formules mathématiques fondamentales pour effectuer la segmentation d'images, en particulier dans ses opérations de convolution, de max-pooling, de convolution transposée, de normalisation de batch, et d'activation ReLU. Voici les principales formules utilisées :

1. Convolution 2D :

La convolution est utilisée dans l'encodeur pour extraire des caractéristiques de l'image et dans le décodeur pour reconstruire l'image segmentée. La convolution 2D est définie par :

$$(\mathbf{f} * \mathbf{g})(\mathbf{x}, \mathbf{y}) = \sum_{i=-1}^{1} \sum_{j=-1}^{1} \mathbf{f}(i, j) \cdot \mathbf{g}(\mathbf{x} - i, \mathbf{y} - j), \tag{3.1}$$

où $\mathbf{f}$ est le noyau de convolution (ou filtre), $\mathbf{g}$ représente la carte de caractéristiques, et $(\mathbf{x}, \mathbf{y})$ sont les coordonnées de la carte de sortie.

2. Max-Pooling :

Le max-pooling est appliqué dans le chemin contractant pour réduire la résolution des cartes de caractéristiques tout en conservant les informations les plus importantes. La formule pour le max-pooling dans une fenêtre $\mathbf{p} \times \mathbf{p}$ est :

$$\mathbf{P}(\mathbf{x}, \mathbf{y}) = \max_{i,j \in \text{pool}} \mathbf{g}(\mathbf{x} + i, \mathbf{y} + j), \tag{3.2}$$

où $\mathbf{P}(\mathbf{x}, \mathbf{y})$ est la valeur maximale dans la région de pooling autour de $(\mathbf{x}, \mathbf{y})$. Cette opération permet de réduire la taille des cartes de caractéristiques, améliorant l'efficacité et la robustesse aux variations de position dans l'image.

3. Normalisation de Batch :

La normalisation de batch stabilise et accélère l'entraînement en standardisant les valeurs d'activation à travers chaque batch. La formule de normalisation de batch est :

$$\text{BN}(\mathbf{x}) = \frac{\gamma(x - \mu)}{\sqrt{\sigma^2 + \epsilon}} + \beta, \tag{3.3}$$

où $\mathbf{x}$ est la valeur de la caractéristique, μ est la moyenne des valeurs du batch, σ^2 est la variance des valeurs du batch, ϵ est une petite constante pour éviter la division par zéro, γ et β sont des paramètres appris pour ajuster l'échelle et le décalage.

4. Fonction d'Activation ReLU :

ReLU (Rectified Linear Unit) introduit la non-linéarité dans le réseau pour permettre l'apprentissage de caractéristiques complexes. La formule de la fonction d'activation ReLU est :

$$\text{ReLU}(\mathbf{x}) = \max(0, \mathbf{x}). \tag{3.4}$$

Cette fonction remplace toutes les valeurs négatives par zéro, augmentant la capacité de représentation du réseau.

Ces opérations travaillent ensemble dans UNet pour effectuer la segmentation d'images, en permettant au modèle de capturer les caractéristiques importantes tout en maintenant des détails fins, nécessaires pour une segmentation précise.

ResNet34, ou *Residual Network 34*, est un modèle de réseau de neurones profond

conçu avec 34 couches et introduit dans le cadre de la famille ResNet par les chercheurs de Microsoft [23]. Ce modèle se distingue par l'utilisation de *blocs résiduels*, une innovation qui permet de résoudre le problème de dégradation des performances observé dans les réseaux profonds en ajoutant des connexions résiduelles directes entre les couches. Ces connexions résiduelles permettent de transférer l'entrée d'une couche directement à sa sortie, ce qui facilite la circulation des gradients pendant l'entraînement. Cette architecture est particulièrement efficace pour l'apprentissage de caractéristiques profondes tout en maintenant une bonne stabilité et une convergence rapide, rendant ResNet34 très performant dans des tâches de classification et de reconnaissance d'images, comme illustré dans la Figure 3.4.

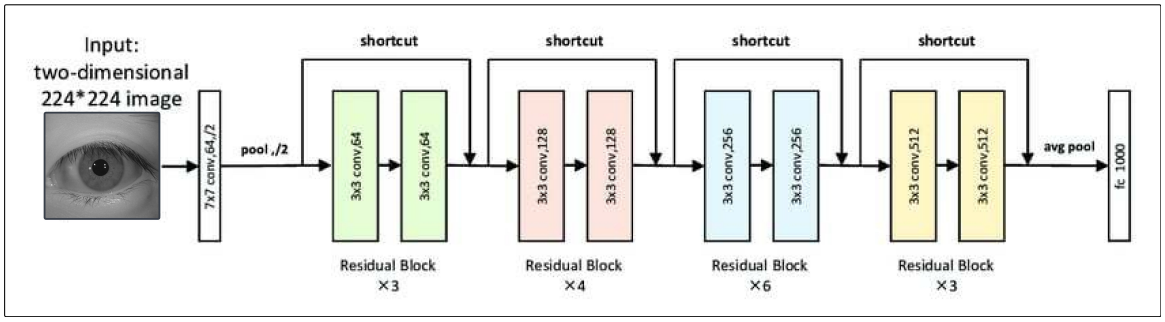

FIGURE 3.4 – Architecture ResNet34.

La fonction principale de ResNet34 repose sur une opération de convolution suivie d'une normalisation par lots (Batch Normalization) et d'une activation ReLU, le tout encapsulé dans une connexion résiduelle qui additionne directement l'entrée initiale à la sortie transformée. La normalisation par lots ajuste les données en utilisant la moyenne et la variance du batch, tout en permettant un ajustement supplémentaire grâce aux paramètres d'échelle ($\boldsymbol{\gamma}$) et de décalage ($\boldsymbol{\beta}$). Les convolutions utilisent des noyaux de 3×3 pour extraire les caractéristiques spatiales, et la fonction ReLU introduit de la non-linéarité pour enrichir l'apprentissage.

ResNet34-UNet utilise ResNet34 pour extraire des cartes de caractéristiques

avec 512 canaux, réduites à une résolution de 1/32 de l'image d'origine, et intégrées dans un cadre UNet pour une segmentation précise de l'iris. Après extraction, ces caractéristiques sont restaurées à la résolution d'origine via un décodeur, capturant les détails spatiaux essentiels. Ce réseau comporte deux têtes de sortie : la première, dédiée à la segmentation, produit un masque d'iris, tandis que la seconde, axée sur la localisation, génère des masques des contours intérieurs et extérieurs de l'iris, comme illustré dans les Figures 3.5 et 3.6. Une activation sigmoïde transforme ces sorties en cartes de probabilité, où chaque pixel est évalué : ceux supérieurs à 0,5 sont considérés comme appartenant à l'iris ou à ses contours, tandis que les autres sont classés comme arrière-plan. Pour assurer une segmentation et une localisation optimales, une combinaison de dice loss et de binary cross-entropy est appliquée, améliorant la précision.

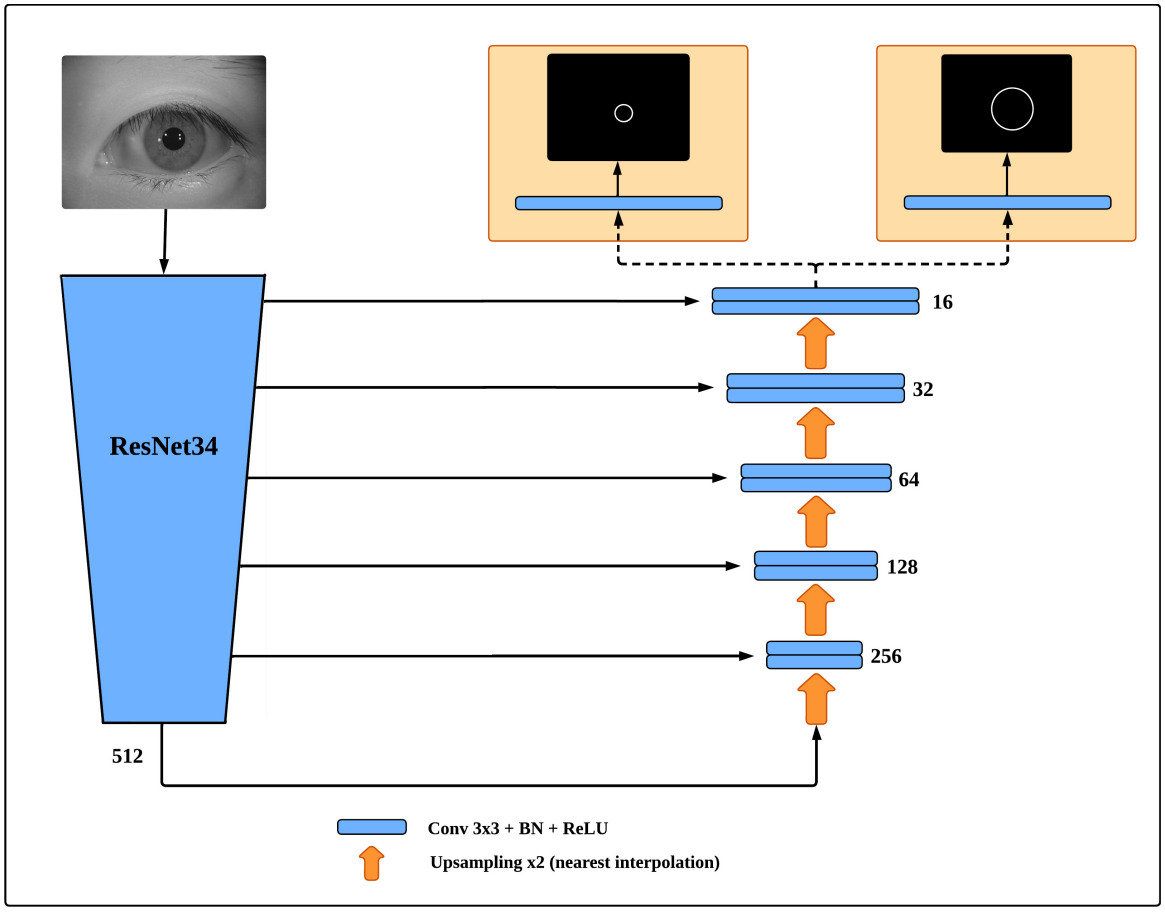

FIGURE 3.5 – Hybride ResNet34-Unet Localisation.

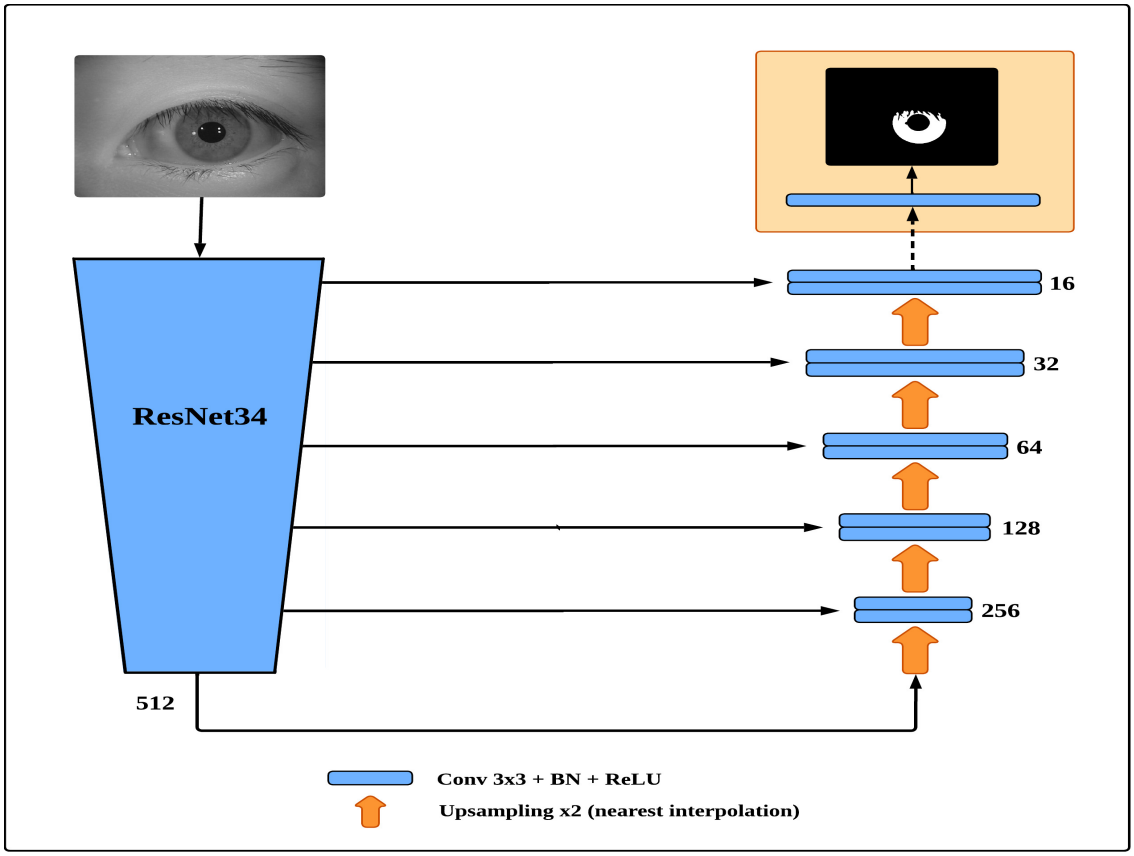

FIGURE 3.6 – Hybride ResNet34-Unet Segmentation.

3.2.1 Localisation de l'iris

Le processus de localisation commence par l'identification des limites intérieure (pupille) et extérieure (iris) de l'œil dans l'image. Comme illustré dans la Figure 3.7, pour la limite intérieure, l'image est redimensionnée et les contours de la pupille sont extraits, générant un masque qui est ensuite ajusté en forme d'ellipse pour mieux correspondre à la forme réelle de la pupille. De même, pour la limite extérieure, les contours de l'iris sont isolés, un masque est créé et ajusté en ellipse. Une fois ces limites définies, un modèle de segmentation basé sur Unet et ResNet34 est appliqué pour obtenir une segmentation précise de l'iris. En post-traitement, une technique de Test Time Augmentation (TTA) est utilisée pour renforcer la robustesse et la

précision en appliquant des transformations sur l'image d'entrée et en moyennant les prédictions, ce qui rend le modèle plus résistant aux variations.

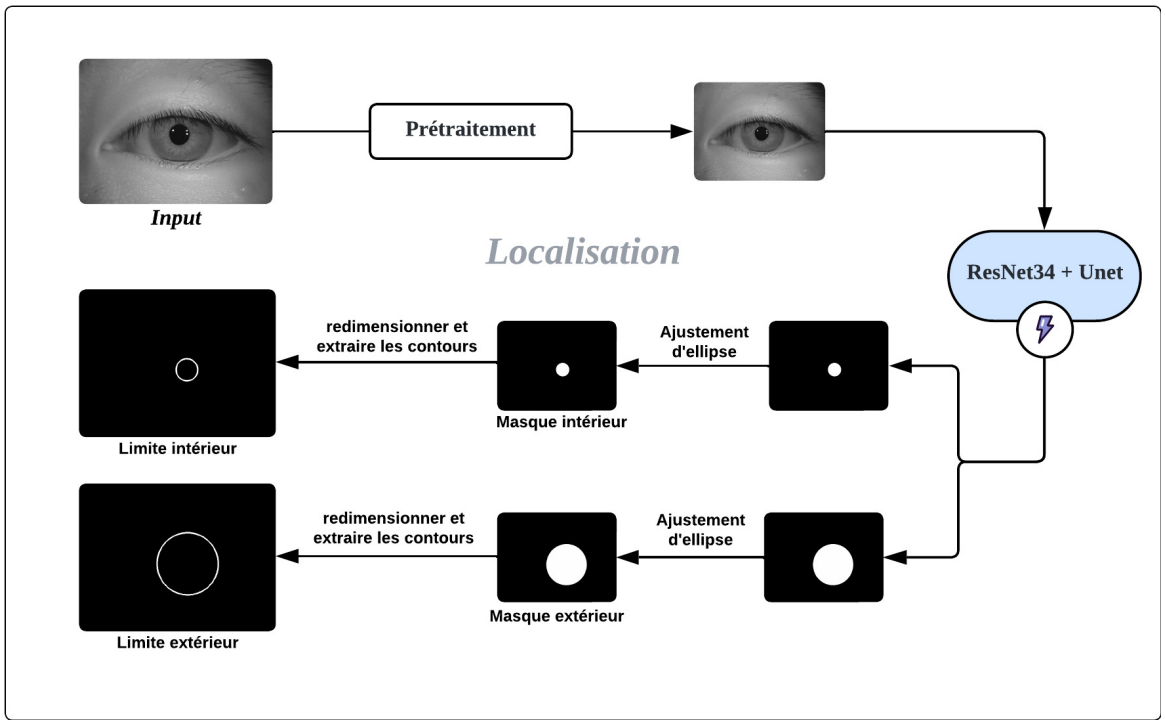

FIGURE 3.7 – Processus de Localisation.

La Figure 3.8 montre des exemples des résultats de cette approche sur des exmaples de notre dataset, illustrant l'efficacité de la localisation obtenue.

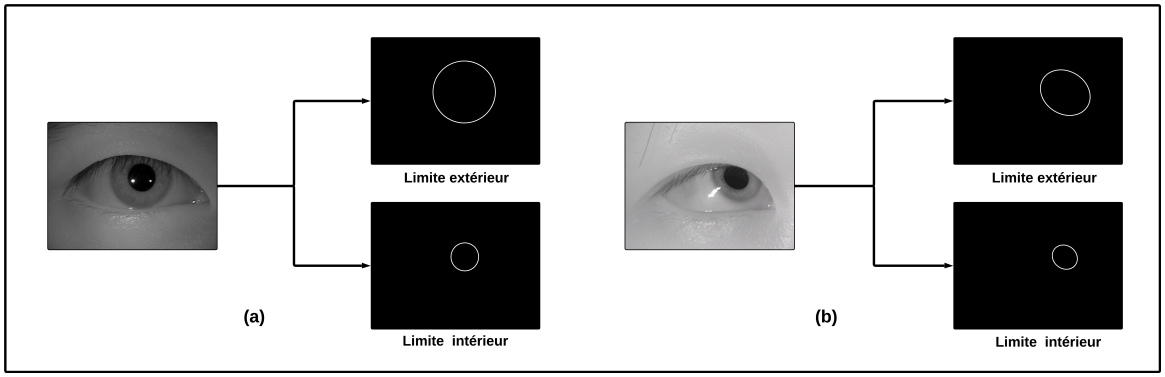

FIGURE 3.8 – Résultats de Localisation de l'Iris sur le Dataset.

3.2.2 Segmentation de l'iris

Le processus de segmentation commence avec une image d'entrée de l'œil, qui passe par une étape de prétraitement visant à améliorer la qualité en réduisant le bruit et les variations indésirables. Ensuite, l'image prétraitée est traitée par l'architecture hybride ResNet34-U-Net, conçue pour effectuer une segmentation précise et isoler la région de l'iris. Pour renforcer la robustesse de la segmentation, la technique de Test Time Augmentation (TTA) est appliquée en post-traitement : elle transforme l'image d'entrée de différentes manières, permettant au modèle de réaliser des prédictions pour chaque version transformée, avec un résultat final obtenu par la moyenne des prédictions, augmentant ainsi la précision voir la Figure 3.9. L'image segmentée de l'iris est ensuite redimensionnée pour s'adapter aux spécifications de l'étape suivante. Le résultat final est une image binaire, montrant uniquement l'iris, prête pour l'analyse ou la reconnaissance de l'iris.

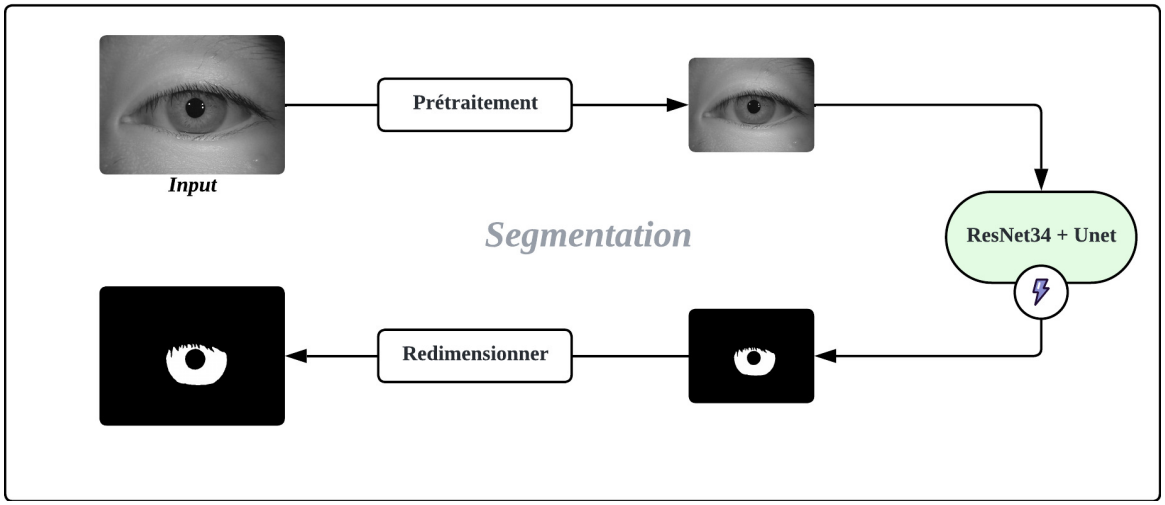

FIGURE 3.9 – Processus de Segmentation.

Les principales problématiques rencontrées dans la segmentation de l'iris incluent la réflexion de la lumière sur l'iris et l'obstruction causée par les cils. Comme illustré dans la Figure 3.10(a), la réflexion de la lumière crée des zones de haute intensité

sur l'iris qui peuvent être confondues avec des caractéristiques pertinentes, rendant la segmentation moins précise. Cette réflexion doit donc être détectée et éliminée pour éviter les erreurs d'analyse. Figure 3.10(b), la présence de cils au bord de l'iris crée des ombres et des obstructions, perturbant la détection des contours réels de l'iris. Une étape de détection et d'élimination des cils est nécessaire pour s'assurer que seul l'iris est segmenté, sans interférences. Ces étapes d'élimination sont cruciales pour améliorer la précision et la qualité de la segmentation dans les applications de reconnaissance de l'iris.

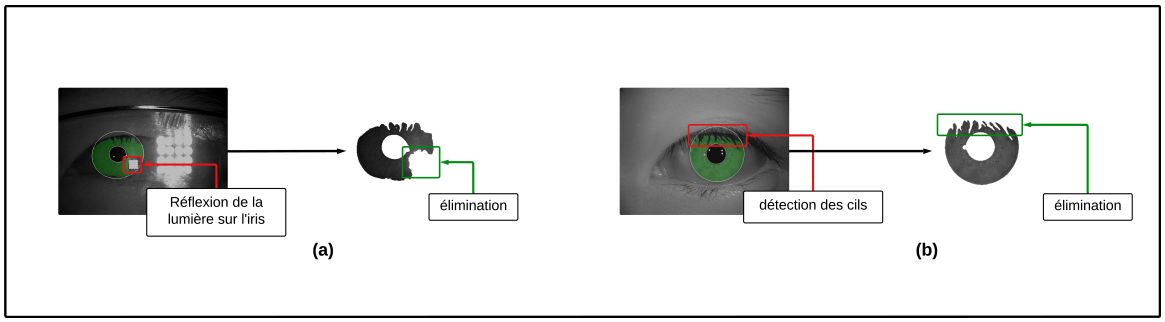

FIGURE 3.10 – Élimination des artéfacts en segmentation.

La Figure 3.11 montre les résultats de la segmentation de l'iris pour deux exemples. En (a), la segmentation isole précisément l'iris d'un œil sans obstructions. En (b), malgré l'angle d'acquisition, le masque de segmentation parvient à isoler l'iris avec quelques pertes mineures près de la zone de réflexion.

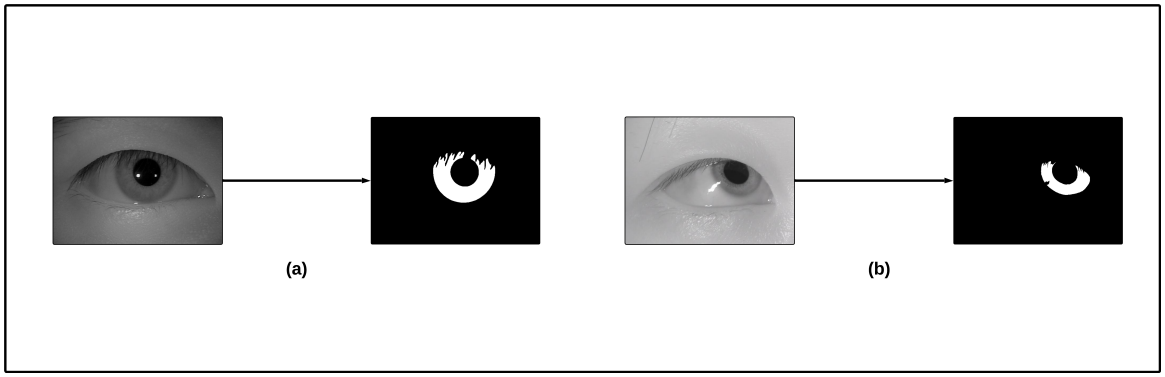

FIGURE 3.11 – Résultats de Segmentation de l'Iris sur le Dataset.

3.2.3 Validation croisée

La Figure 3.12 illustre la méthodologie détaillé pour la segmentation et la localisation de l'iris. Le dataset a été subdivisé en cinq parties égales (ou "folds") pour une validation croisée, chaque fold étant utilisé successivement comme jeu de validation, tandis que les autres servent à l'entraînement. Deux approches de modèles ont été évaluées : un modèle de base, directement entraîné sur chaque fold, et un modèle préentraîné, initialement ajusté sur les trois datasets combinés, puis affiné individuellement sur chaque fold. L'architecture réseau repose sur un framework hybride, **ResNet34+UNet**, implémenté via la bibliothèque *Segmentation Models PyTorch*. Les hyperparamètres utilisés pour l'entraînement sont résumés dans le Tableau 3.2.

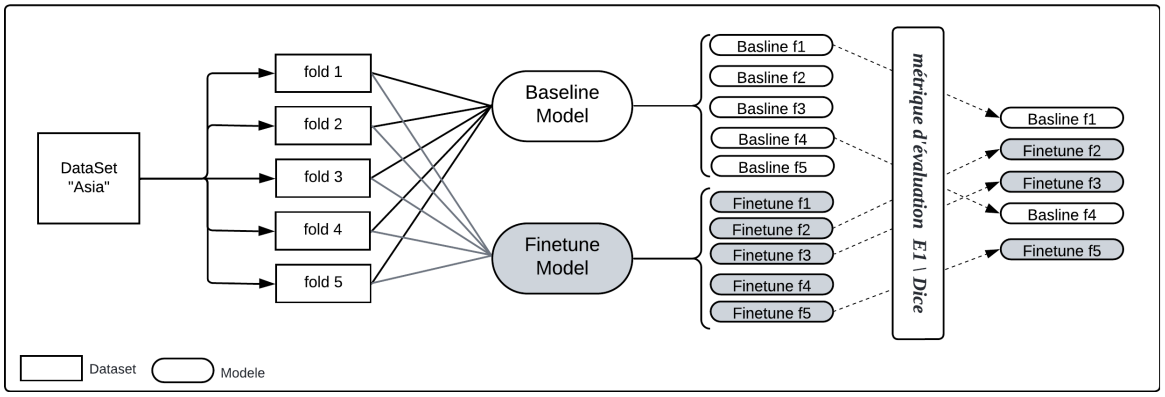

FIGURE 3.12 – Validation croisée.

Chaque modèle, après entraînement, a été évalué selon des métriques telles que **E1** pour la segmentation et **Dice** pour la localisation. Pour chaque **fold**, le modèle offrant les meilleures performances a été retenu. Ce processus permet d'identifier les modèles les plus robustes et d'améliorer la généralisation des performances sur différents sous-ensembles du dataset.

TABLE 3.2 – Hyperparamètres utilisés pour l'entraînement du modèle.

Hyperparamètre	**Valeur**
Taille d'entrée	640×480 (CASIA-Iris-Asia), 416×416 (CASIA-Iris-M1), 640×384 (CASIA-Iris-Africa)
Batch size	16 (CASIA-Iris-Asia), 28 (CASIA-Iris-M1), 16 (CASIA-Iris-Africa)
Nombre d'époques	500
Taux d'apprentissage	0,0002
Optimiseur	Adam

3.2.4 Métriques d'Évaluation

Les organisateurs de NIR-ISL 2021 [9] ont évalué la segmentation de l'iris en utilisant la métrique **E1**, qui mesure la proportion moyenne de pixels discordants entre le masque d'iris prédit et le masque de référence. Cette mesure, proposée initialement dans le cadre de la compétition NICE.I, quantifie la précision de la segmentation en calculant les différences pixel-par-pixel entre le masque binaire soumis et la vérité terrain. Elle permet ainsi de fournir une évaluation fine et objective des performances des algorithmes de segmentation, en tenant compte des variations possibles. Les modèles entraînés seront également évalués à l'aide de cette métrique pour déterminer leur efficacité. Mathématiquement, **E1** est défini comme suit :

$$\mathbf{E1} = \frac{1}{\mathbf{n} \times \mathbf{h} \times \mathbf{w}} \sum_{\mathbf{i}} \sum_{\mathbf{j}} \mathbf{M}(\mathbf{i}, \mathbf{j}) \otimes \mathbf{G}(\mathbf{i}, \mathbf{j}), \tag{3.5}$$

où $\mathbf{i}$ et $\mathbf{j}$ désignent les coordonnées des pixels dans le masque prédit $\mathbf{M}$ et le masque de référence $\mathbf{G}$, $\mathbf{h}$ et $\mathbf{w}$ représentent la hauteur et la largeur de chaque image de test, et $\mathbf{n}$ correspond au nombre total d'images. Une valeur élevée de **E1** indique une

proportion importante de pixels discordants, signalant une précision moindre dans la segmentation de l'iris.

Pour l'évaluation de la localisation de l'iris, le défi a adopté l'**indice de Dice** comme métrique pour mesurer la similarité entre le contour de l'iris prédit et le contour de référence. L'indice de Dice évalue le degré de chevauchement entre les deux contours en fonction de leurs ensembles de pixels remplis, offrant une mesure de similarité entre les frontières interne et externe des deux contours. Cet indice est défini par l'équation :

$$\text{Dice}(\mathbf{G}, \mathbf{B}) = \frac{2|\mathbf{G} \cap \mathbf{B}|}{|\mathbf{G}| + |\mathbf{B}|}, \tag{3.6}$$

où $\mathbf{G}$ et $\mathbf{B}$ désignent respectivement l'ensemble des pixels du contour de référence et de la prédiction. Le calcul est basé sur la cardinalité des ensembles, avec $|\mathbf{G} \cap \mathbf{B}|$ représentant l'intersection des pixels prédits et de référence, et $|\mathbf{G}|$ et $|\mathbf{B}|$ les nombres totaux de pixels dans chacun des contours. Un indice de Dice proche de $\mathbf{1}$ indique une forte correspondance, traduisant une localisation précise de l'iris par rapport à la vérité terrain. Nous utiliserons ces métriques pour mesurer l'efficacité de notre approche hybride, et les résultats seront présentés dans le prochain chapitre.

3.2.5 Ensemble modèles

Après la validation croisée, les modèles sélectionnés sont utilisés dans cette méthode d'ensemble, qui combine plusieurs modèles de segmentation ou localisation pour améliorer la précision de la détection de l'iris. L'image d'entrée est d'abord redimensionnée pour s'adapter aux dimensions des modèles, puis elle subit une phase d'augmentation en temps de test (Test-Time Augmentation, TTA) comme illustré dans la Figure 3.13. Cette étape génère des variantes de l'image d'origine avec des transformations simples,

renforçant ainsi la robustesse des prédictions face aux variations des données.

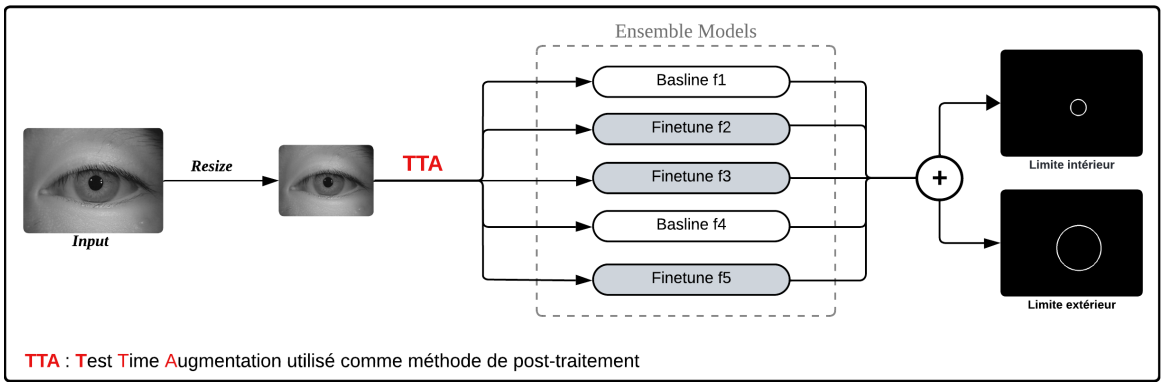

FIGURE 3.13 – Ensemble modèles pour la Localisation.

Chaque réseau de segmentation produit ensuite sa propre prédiction pour la segmentation de l'iris, comme illustré dans la Figure 3.14. Les prédictions des différents modèles sont ensuite combinées en calculant leur moyenne, ce qui aboutit à une segmentation finale plus précise et stable. En intégrant les points forts de chaque modèle, cette approche optimise la performance globale, atténue les limitations individuelles des modèles et garantit une détection plus fiable et cohérente de l'iris. Cette étape de combinaison exploite la diversité des modèles, où chaque modèle contribue à corriger les erreurs des autres.

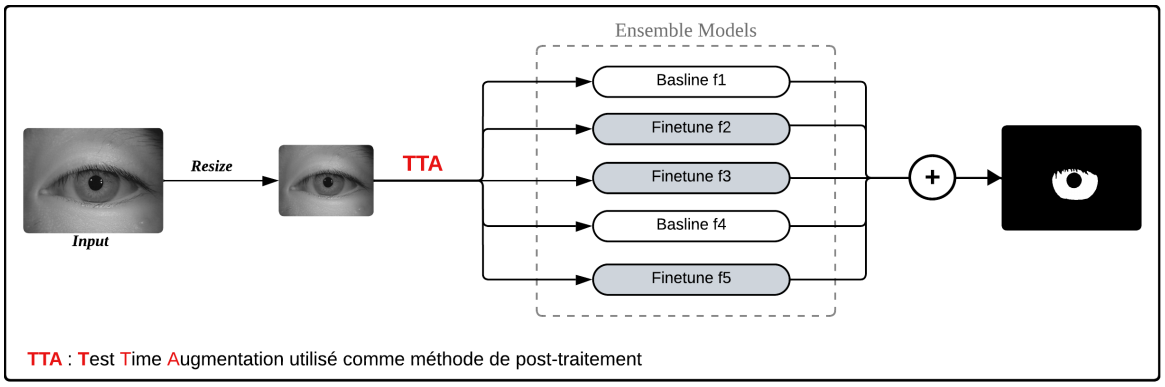

FIGURE 3.14 – Ensemble modèles Pour la Segmentation.

3.2.6 Combinaison et Extraction de l'Iris

Après avoir localisé et segmenté les contours externe et interne de l'iris, la méthode de combinaison utilise ces masques pour superposer les contours segmentés sur l'image brute, ce qui permet de mettre en évidence la région de l'iris de façon précise, comme illustré dans 3.15. L'étape de combinaison utilise ces masques pour superposer les contours segmentés sur l'image brute, mettant ainsi en évidence la région de l'iris de manière précise. Mathématiquement, cette combinaison peut être exprimée par la fonction suivante :

$$\boldsymbol{I}_{\text{combinée}}(x, y) = \boldsymbol{I}(x, y) \times (\boldsymbol{M}_{\text{ext}}(x, y) + \boldsymbol{M}_{\text{int}}(x, y)), \tag{3.7}$$

où $\boldsymbol{I}(x, y)$ est l'image brute, $\boldsymbol{M}_{\text{ext}}(x, y)$ et $\boldsymbol{M}_{\text{int}}(x, y)$ sont les masques binaires pour les contours externe et interne de l'iris respectivement, et $\boldsymbol{I}_{\text{combinée}}(x, y)$ est l'image combinée résultante montrant uniquement la région de l'iris délimitée.

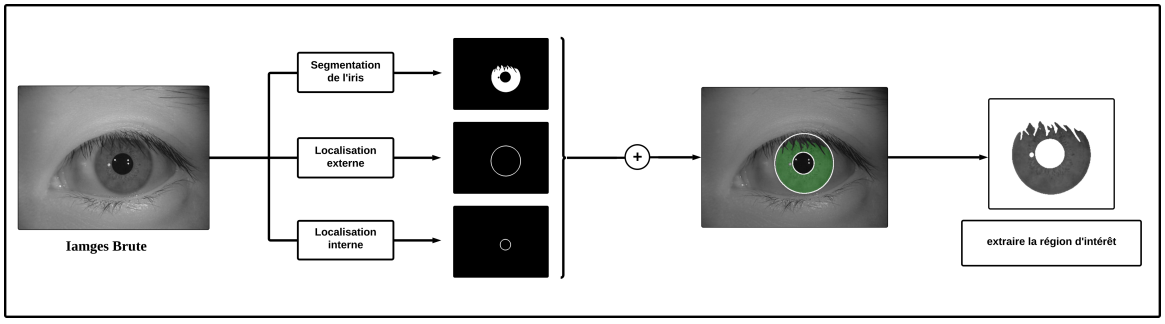

FIGURE 3.15 – Combinaison et Extraction de l'Iris.

Cette fonction multiplie pixel par pixel l'image brute avec la somme des masques, permettant une délimitation claire de la région d'intérêt (ROI). Nous appliquons ensuite des ajustements de contraste et de luminosité pour améliorer la visibilité de l'iris avant d'extraire la région d'intérêt, prête pour l'analyse et la classification. Cette

approche améliore la précision de la reconnaissance en isolant uniquement les caractéristiques pertinentes de l'iris, assurant ainsi une reconnaissance biométrique fiable même dans des conditions d'éclairage et d'angle variées.

3.2.7 Normalisation

Une fois la région circulaire de l'iris correctement segmentée à partir d'une image oculaire, une normalisation est appliquée pour transformer cette région segmentée en une forme rectangulaire de taille fixe. Ce processus garantit que la région de l'iris ait des dimensions constantes, permettant ainsi l'extraction de caractéristiques invariantes, même dans des conditions de capture différentes. Dans ce travail, le modèle de feuille de caoutchouc de Daugman [24] a été utilisé pour normaliser l'image de l'iris.

Modèle de Feuille de Caoutchouc de Daugman

Le modèle de feuille de caoutchouc de Daugman, proposé par Daugman en 1993, est la méthode de normalisation de l'iris la plus largement utilisée, voir la Figure 3.16 . Ce modèle transforme la région circulaire de l'iris en un bloc rectangulaire de taille fixe en mappant chaque pixel de la région circulaire sur un système de coordonnées polaires (r, θ), où :

— $\mathbf{r}$: Distance radiale (depuis la limite de la pupille jusqu'au limbe).
— $\boldsymbol{\theta}$: Position angulaire (angle de rotation autour de l'iris).

La transformation est exprimée comme suit :

$$\mathbf{I}[\mathbf{x}(\mathbf{r}, \boldsymbol{\theta}), \mathbf{y}(\mathbf{r}, \boldsymbol{\theta})] \rightarrow \mathbf{I}(\mathbf{r}, \boldsymbol{\theta}), \tag{3.8}$$

où :

— $\mathbf{I}(\mathbf{x}, \mathbf{y})$ représente les coordonnées cartésiennes de la région de l'iris.

— $\mathbf{I}(\mathbf{r}, \boldsymbol{\theta})$ correspond aux coordonnées polaires normalisées.

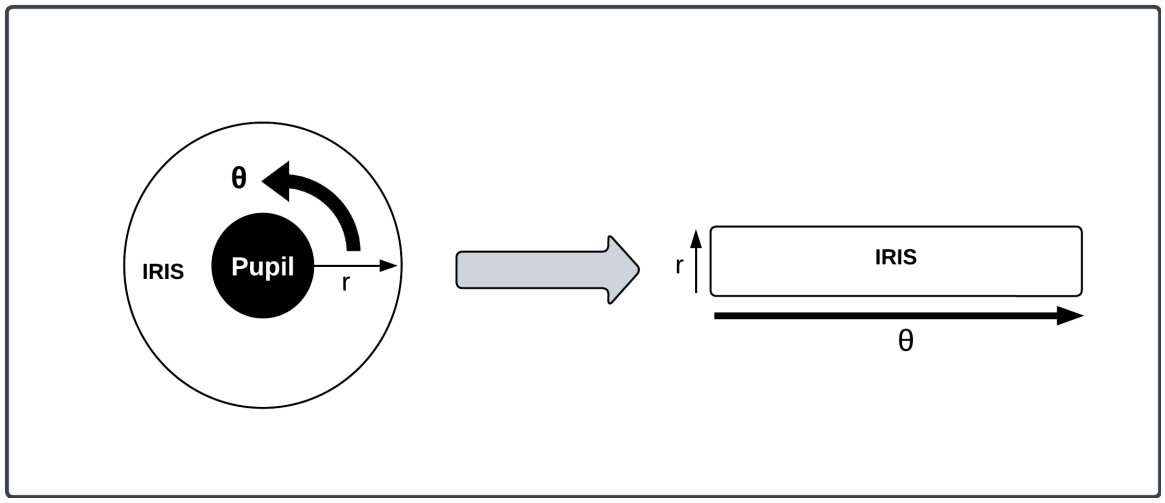

FIGURE 3.16 – Feuille de Caoutchouc de Daugman.

Les Figures 3.17.a et 3.17.b illustrent les résultats de la normalisation appliquée aux échantillons de notre base de données. Ces échantillons ont d'abord suivi le processus, expliqué précédemment, d'extraction de la région d'intérêt de l'iris.

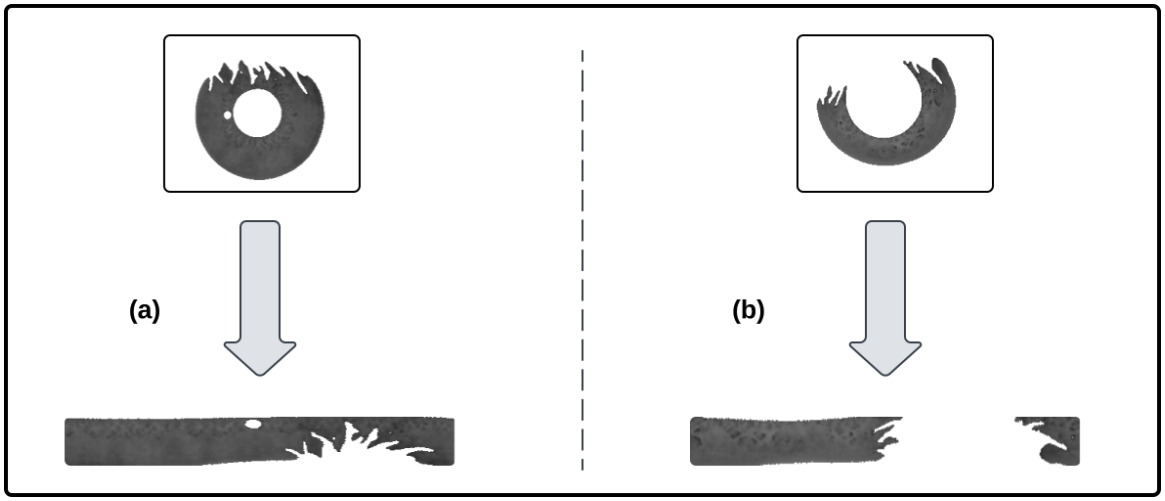

FIGURE 3.17 – Résultats de la Normalisation.

3.2.8 Prétraitement des données

Dans le domaine de la reconnaissance des individus, la qualité des données joue un rôle déterminant dans la précision et l'efficacité des systèmes d'identification. Par conséquent, le prétraitement des données constitue une étape essentielle visant à transformer les données brutes en un format structuré et exploitable. Cette étape garantit que les modèles d'apprentissage soient entraînés avec des informations pertinentes et de haute qualité, optimisant ainsi leur capacité à distinguer avec fiabilité les individus. L'impact du prétraitement se reflète directement dans les performances globales des systèmes, en renforçant à la fois leur robustesse et la précision des résultats obtenus. Un prétraitement adéquat des données améliore la capacité du modèle à généraliser, conduisant à des performances supérieures et à des décisions plus précises. Dans notre étude, trois méthodes de prétraitement ont été utilisées :

Égalisation d'histogramme : cette technique vise à améliorer le contraste des images en redistribuant les intensités des pixels de manière uniforme sur l'ensemble de la plage dynamique. En égalisant l'histogramme, les détails présents dans les zones sombres ou claires de l'image deviennent plus visibles, ce qui facilite l'extraction de caractéristiques pertinentes et améliore ainsi la performance des modèles d'apprentissage automatique dans les tâches de reconnaissance des individus. Cette approche permet également d'éviter les biais liés à une distribution non uniforme des valeurs d'intensité. Par conséquent, elle peut améliorer la qualité des données d'entrée, ce qui se traduit par des résultats plus précis et fiables. Cette approche est décrite en détail par Rafael C Gonzalez [25] dans leur ouvrage Digital Image Processing. La Figure 3.18.a. montre les résultats de cette méthode.

Local Binary Pattern (LBP) : est une méthode utilisée pour améliorer la qualité des images en analysant les relations spatiales entre un pixel central et ses voisins immédiats dans une fenêtre locale. À chaque pixel voisin, un seuil est appliqué par rapport au pixel central, générant ainsi une valeur binaire (1 ou 0) en fonction de

la comparaison d'intensité entre le pixel central et ses voisins. L'usage du LBP pour améliorer la qualité des textures a été largement validé dans la littérature, notamment par Ojala et al. [26], qui ont proposé une méthode détaillée pour le traitement des textures à l'aide de cette technique. La séquence binaire obtenue est convertie en un nombre décimal, produisant ainsi un code caractéristique pour chaque pixel du voisinage. Ces codes contribuent à l'amélioration de la qualité de l'image en créant une carte de caractéristiques (image LBP) qui reflète mieux la texture locale de l'image. Les résultats de cette amélioration sont illustrés dans la Figure 3.18.b.

Filtres de Gabor : sont utilisés pour améliorer la qualité des images en analysant les structures locales à différentes échelles et orientations. Ces filtres sont basés sur des fonctions sinusoïdales modulées par une fonction gaussienne, ce qui permet de capturer à la fois l'information fréquentielle et spatiale d'une image. L'application des filtres de Gabor consiste à convoluer l'image avec plusieurs filtres couvrant diverses orientations et échelles. Chaque filtre est défini par des paramètres tels que la fréquence, l'orientation et la largeur de la gaussienne, permettant ainsi d'extraire des réponses caractéristiques qui mettent en évidence les structures texturales présentes dans l'image. Cette méthode a été largement étudiée et prouvée efficace, notamment par Daugman [24], pour l'amélioration de la qualité des textures. Les résultats obtenus après l'application des filtres de Gabor sont illustrés dans la Figure 3.18.c.

3.2.9 Extraction des caractéristiques et classification

L'extraction des caractéristiques constitue une étape cruciale dans le cadre de votre projet de reconnaissance de l'iris, car elle permet de transformer les images normalisées en représentations discriminantes, adaptées aux étapes ultérieures d'identification. Cette phase repose sur l'utilisation de réseaux de neurones convolutifs (CNNs) pré-entraînés, tirant parti des techniques avancées de transfert d'apprentissage. Le transfert d'apprentissage permet de réutiliser les connaissances préalablement acquises

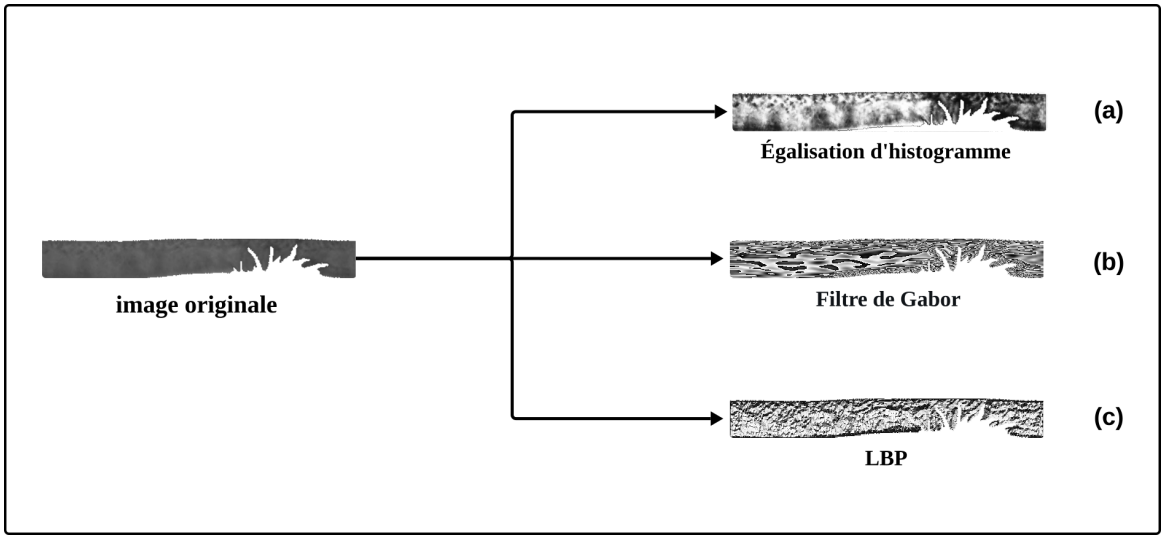

FIGURE 3.18 – Prétraitement des données.

par ces modèles sur des bases de données à large échelle, telles qu'ImageNet, en les adaptant à la tâche spécifique de la reconnaissance de l'iris. Ces CNNs, grâce à leur capacité à capturer des motifs hiérarchiques et multi-échelle, sont particulièrement efficaces pour extraire des descripteurs distinctifs reflétant les variations texturales et géométriques caractéristiques de l'iris. Dans le cadre de notre étude, nous avons mobilisé cinq réseaux de neurones convolutifs (CNN) pour l'extraction des caractéristiques, à savoir :

ResNet50, introduit par He et al. [27], est un réseau profond composé de 50 couches intégrant des *blocs résiduels*. Ces blocs résiduels permettent le passage direct des gradients via des connexions de type *skip connection*, ce qui facilite l'apprentissage dans des réseaux profonds en atténuant les problèmes de dégradation (vanishing gradients). ResNet50 utilise une combinaison de convolutions de taille fixe (3x3), suivies de couches de normalisation par lot (*batch normalization*) et de fonctions d'activation ReLU. Les blocs résiduels permettent d'apprendre des fonctions identitaires, ce qui accélère la convergence et améliore la performance. Dans notre étude, ce modèle est utilisé pour extraire des caractéristiques détaillées et complexes des motifs d'iris, souvent nécessaires pour différencier des individus présentant des iris similaires.

ResNet34 est une version simplifiée de ResNet50, composée de 34 couches. Tout comme ResNet50, il s'appuie sur des blocs résiduels, mais sa profondeur moindre en fait un modèle moins coûteux en termes de calcul tout en conservant des performances élevées. Les couches convolutives et les *skip connections* suivent une structure similaire, mais avec un nombre réduit de paramètres. Dans notre pipeline, ResNet34 offre un compromis entre précision et rapidité d'exécution, particulièrement utile dans les scénarios où les ressources sont limitées [27]. Par ailleurs, sa capacité à extraire des descripteurs discriminants en fait une architecture adaptée pour des tâches nécessitant une généralisation efficace sur des données complexes, tout en réduisant la consommation de mémoire et de temps d'entraînement.

VGG16, proposé par Simonyan et Zisserman [28], est un modèle basé sur une architecture séquentielle composée de 16 couches (13 convolutives et 3 entièrement connectées). Ce réseau utilise des convolutions fixes de taille 3x3, suivies de couches de *max pooling*. Contrairement à ResNet, VGG16 ne repose pas sur des connexions résiduelles, mais sur une profondeur uniforme et un nombre croissant de filtres convolutifs. La simplicité architecturale, combinée à une profondeur modérée, rend VGG16 particulièrement robuste pour l'extraction de caractéristiques globales et intermédiaires. Ce modèle est utilisé pour capturer des invariants globaux dans les images d'iris, ce qui le rend efficace pour gérer les variations d'échelle et de conditions d'éclairage. De plus, il permet de réduire la complexité de prétraitement des données en s'appuyant sur des caractéristiques facilement adaptables aux variations des motifs biométriques.

InceptionV3, introduit par Szegedy et al. [29], repose sur l'utilisation de *modules Inception*, qui combinent des convolutions de tailles différentes (1x1, 3x3, 5x5) dans une seule couche pour capturer des descripteurs multi-échelle. Ce modèle intègre également des optimisations telles que la factorisation des convolutions et les convolutions asymétriques, permettant une réduction significative du nombre de paramètres tout en conservant une performance élevée. L'architecture multi-échelle d'InceptionV3 est particulièrement précieuse pour capturer les variations structurelles fines et les mo-

tifs complexes des iris. De plus, son mécanisme de régularisation intégré, comme l'utilisation de couches auxiliaires, améliore la robustesse et la capacité de généralisation du modèle. Cela en fait une solution efficace pour traiter des bases de données variées présentant des conditions d'acquisition différentes.

DenseNet201, introduit par Huang et al. [30], se distingue par une connectivité dense entre les couches. Contrairement aux architectures conventionnelles, chaque couche est directement connectée à toutes les couches suivantes. Cette approche favorise la réutilisation des caractéristiques extraites et réduit la redondance dans les calculs. DenseNet201 comprend 201 couches et est particulièrement performant pour capturer des détails complexes tout en limitant la croissance exponentielle des paramètres. Dans notre étude, DenseNet201 permet d'extraire des descripteurs subtils et complexes des motifs d'iris, tout en assurant une meilleure efficacité de l'apprentissage. La propagation dense des gradients contribue également à une optimisation plus stable et rapide. Les résultats obtenus à partir de cet entraînement seront détaillés dans le prochain chapitre.

Chapitre 4

Expérimentations et discussion

Ce chapitre traite des aspects liés au matériel, aux logiciels, aux langages de programmation et aux bibliothèques utilisés dans le cadre de cette recherche. Il vise principalement à créer un environnement optimal pour la mise en œuvre de l'approche proposée et la conduite des expérimentations. Par la suite, les différentes expérimentations réalisées dans divers scénarios seront explorées. Enfin, une évaluation comparative avec les travaux existants sera présentée pour souligner les contributions et les avantages de notre approche. Une attention particulière sera accordée aux défis rencontrés lors de l'intégration des modèles dans des environnements réels. De plus, les ajustements nécessaires pour améliorer la robustesse des résultats seront discutés.

4.1 Environnement matériel

Nous avons mené nos expérimentations en utilisant une configuration matérielle haut de gamme, comprenant un processeur **Intel(R) Core (TM) i9-14900K** doté de 24 cœurs hybrides, capable d'atteindre une cadence maximale de **6,00 GHz** (et

une fréquence de base de **3,20 GHz**). Ce processeur garantit une puissance de calcul suffisante pour gérer des charges de travail intensives, telles que l'entraînement de modèles complexes. La machine est également équipée de **64 Go de RAM DDR5** fonctionnant à une vitesse de **5600 MHz**, permettant un traitement fluide et rapide des grands ensembles de données sans limitation mémoire.

Pour l'entraînement des modèles, nous avons exploité les **CUDA cores** de la carte graphique **NVIDIA GeForce RTX 4090**, équipée de **24 Go de mémoire GDDR6X**. Cette carte offre des performances exceptionnelles en calcul parallèle grâce à ses **16384 CUDA cores**, ce qui a permis une accélération significative des processus d'entraînement, en réduisant le temps nécessaire pour traiter de grandes quantités de données tout en optimisant les ressources. Cette configuration matérielle a été déterminante pour gérer efficacement les différentes étapes des expérimentations.

4.2 Environnement logiciel

Nous avons élaboré notre méthode de classification en utilisant l'environnement de développement **Anaconda** version **2.6.2**, accompagné de l'éditeur de texte **Visual Studio Code (VSCode)** version **1.95.3**. Cette approche a été mise en œuvre en utilisant **Python**, un langage de programmation largement reconnu pour son efficacité dans les domaines du développement logiciel, de la science des données et de l'apprentissage automatique. Plus précisément, nous avons utilisé la version **3.6.13** de **Python** pour mener à bien cette étude. Nous avons également exploité une **NVIDIA RTX 4090** pour l'entraînement des modèles, offrant une accélération significative des calculs nécessaires aux traitements de grande ampleur. L'utilisation combinée de ces outils nous a permis de réaliser nos expérimentations de manière **efficace** et **structurée**, tout en bénéficiant des nombreuses fonctionnalités offertes par cet environnement. De plus, nous avons exploité les bibliothèques intégrées d'Anaconda pour simplifier la

gestion des dépendances et des versions logicielles. Cela a assuré une configuration nécessaires à notre projet.

De plus, notre travail s'est appuyé sur plusieurs bibliothèques **open source** d'apprentissage automatique, de traitement d'images et d'analyse de données, parmi lesquelles **PyTorch**, **Scikit-learn (sklearn)** et **OpenCV**. Ces bibliothèques ont été sélectionnées pour leur **flexibilité**, leur **robustesse** et leur capacité à répondre aux besoins spécifiques de notre étude. **PyTorch** a été principalement utilisé pour la conception et l'entraînement des modèles basés sur des **réseaux de neurones**, offrant une grande **modularité** et une prise en charge des calculs accélérés sur **GPU**. **Scikit-learn** a permis d'appliquer diverses techniques d'apprentissage **supervisé** et **non supervisé**, ainsi que d'évaluer les performances des modèles. Enfin, **OpenCV** a été exploité pour le **traitement d'images**, en particulier pour la gestion des données visuelles utilisées dans nos expérimentations. L'ensemble des bibliothèques utilisées, accompagnées de leurs versions respectives, est présenté dans le tableau 4.1 pour une meilleure traçabilité et reproductibilité de nos travaux.

TABLE 4.1 – Versions des bibliothèques utilisées dans le projet.

Bibliothèque	**Version**
PyTorch	1.10.2
Scikit-learn	0.24.2
Scikit-image	0.17.2
Albumentations	1.3.0
OpenCV	4.5.3
TensorboardX	2.5.1
Matplotlib	3.3.4
Numpy	1.19.2
Segmentation-models-pytorch	0.3.0

4.3 Expérimentations et discussion

Dans cette section, nous présentons les résultats obtenus à partir des différentes bases de données utilisées pour l'évaluation des performances des modèles de segmentation, de localisation et de classification d'iris. Nos expériences ont été menées en appliquant plusieurs méthodes de prétraitement (images originales, égalisation d'histogramme, filtre de Gabor et modèle binaire local LBP) et en utilisant divers modèles d'apprentissage profond, notamment **DenseNet201**, **ResNet50**, **ResNet34**, **VGG16** et **InceptionV3**. Les performances des modèles ont été comparées en tenant compte des particularités des bases de données, telles que les variations de texture, de contraste et de qualité des images. Cette analyse approfondie permet d'identifier les meilleures configurations adaptées aux différents scénarios de reconnaissance de l'iris.

4.3.1 Résultats pour la segmentation et la localisation

Résultats de la base de données CASIA-Iris-Africa

Le tableau 4.2 présente les résultats obtenus par la méthode de validation croisée pour la segmentation et la localisation appliquées à la base de données CASIA-Iris-Africa, les meilleurs scores sont sélectionnés après le calcul des métriques d'évaluation **E1** pour la segmentation et **Dice** pour la localisation. Pour la segmentation, le modèle de base montre des performances variables selon les folds, avec des scores **E1** parfois légèrement supérieurs au modèle affiné. Par exemple, pour le **fold 1** et le **fold 4**, le modèle de base obtient de meilleurs scores **E1** (indiqués en Gras dans le tableau) que le modèle affiné. Ces résultats suggèrent que, bien que l'approche de fine-tuning soit généralement plus performante, le modèle de base peut surpasser le modèle affiné dans certaines conditions de validation.

TABLE 4.2 – Tableau des performances sur la base de données CASIA-Iris-Africa.

		Métrique d'évaluation E1 (Seg) \ Dice (Loca)				
Africa	**Méthode**	**Fold1**	**Fold2**	**Fold3**	**Fold4**	**Fold5**
Segmentation	Baseline	**0,003646**	0,003841	0,003504	**0,003628**	0,003745
Segmentation	Finetune	0,003929	**0,003764**	**0,003486**	0,003638	**0,003602**
Location	Baseline	0,9408	0,9476	0,9573	0,9462	0,9481
Location	Finetune	**0,945**	**0,951**	**0,9597**	**0,9513**	**0,9491**

Dans d'autres folds, tels que les **folds 2, 3 et 5**, le modèle affiné offre une meilleure performance en termes de score **E1** pour la segmentation, démontrant l'efficacité du fine-tuning pour ajuster les paramètres du modèle de manière plus précise aux variations de données présentes dans le dataset. Cela indique que l'apprentissage par transfert (*fine-tuning*) améliore la capacité du modèle à s'adapter aux sous-ensembles de données complexes ou légèrement différents du dataset initial.

Pour la localisation, les résultats montrent une supériorité constante du modèle affiné par rapport au modèle de base dans presque tous les folds, avec des scores **Dice** plus élevés. Par exemple, pour les **folds 1, 2, 3, et 5**, le modèle affiné obtient des valeurs de **Dice** supérieures à celles du modèle de base, ce qui montre l'efficacité de l'apprentissage par transfert pour améliorer la précision des prédictions de localisation. Le fine-tuning permet ici de capter des détails subtils dans les données d'entraînement, contribuant à une meilleure délimitation des contours internes et externes de l'iris.

Résultats de la base de données CASIA-Iris-Asia

Le tableau 4.3 présente les résultats obtenus par la méthode de validation croisée pour la segmentation et la localisation appliquées à la base de données CASIA-Iris-Asia. Pour la segmentation, les résultats montrent que le modèle de base (*baseline*) a obtenu de meilleures performances pour la plupart des folds, à l'exception du **fold 5**

où le modèle affiné (*finetune*) a offert un meilleur score E1. En effet, pour les **folds 1, 2, 3 et 4**, le modèle de base affiche des scores E1 plus bas, indiquant une meilleure performance de segmentation que le modèle affiné. Ces résultats soulignent que le comportement des modèles peut varier selon les spécificités des données présentes dans chaque fold.

TABLE 4.3 – Tableau des performances sur la base de données CASIA-Iris-Asia.

		Métrique d'évaluation E1 (Seg) \ Dice (Loca)				
Asia	**Méthode**	**Fold1**	**Fold2**	**Fold3**	**Fold4**	**Fold5**
Segmentation	Baseline	**0,003564**	**0,003284**	**0,003297**	**0,003189**	0,003621
Segmentation	Finetune	0,003704	0,003419	0,003316	0,003202	**0,003442**
Location	Baseline	0,9534	0,9437	0,9566	0,9403	0,9473
Location	Finetune	**0,9623**	**0,9515**	**0,9586**	**0,9638**	**0,9604**

Cependant, dans le **fold 5**, le modèle affiné a obtenu un score E1 de **0.003442**, qui est inférieur à celui du modèle de base (**0.003621**), indiquant une meilleure performance pour cette partition spécifique. Ce résultat démontre que, bien que l'apprentissage par transfert puisse améliorer les performances de segmentation dans certains cas, il ne garantit pas systématiquement une meilleure généralisation pour cette tâche spécifique. Cela met en évidence l'importance d'adopter des approches adaptées et flexibles pour maximiser les performances globales.

Pour la localisation, le modèle affiné est clairement supérieur dans tous les folds. Par exemple, pour le **fold 1**, le modèle affiné obtient un score Dice de **0.9623**, contre **0.9534** pour le modèle de base. Cette supériorité du modèle affiné est constante dans tous les autres folds, avec des scores Dice toujours plus élevés que ceux du modèle de base. Ces observations confirment que le fine-tuning est particulièrement efficace pour capturer des détails subtils et améliorer la précision dans des tâches de localisation complexes. Cette amélioration globale reflète la capacité du modèle affiné à s'adapter aux variations structurelles présentes dans les données, augmentant ainsi la robustesse des prédictions.

Résultats de la base de données CASIA-Iris-M1

Le tableau 4.4. présente les résultats obtenus par la méthode de validation croisée pour la segmentation et la localisation appliquées à la base de données CASIA-Iris-M1. Pour la segmentation, le modèle affiné montre des performances supérieures au modèle de base dans tous les folds, avec des scores E1 plus bas. Par exemple, dans le **fold 1**, le modèle affiné obtient un score E1 de **0.003767** contre **0.005176** pour le modèle de base, ce qui indique une meilleure performance en termes de précision de segmentation. Cette tendance est constante dans les **folds 2, 3, 4 et 5**, où le modèle affiné affiche également des scores E1 inférieurs, avec **0.003649**, **0.003586**, **0.003709**, et **0.003996**, respectivement, comparativement aux scores plus élevés du modèle de base.

TABLE 4.4 – Tableau des performances sur la base de données CASIA-Iris-M1.

		Métrique d'évaluation E1 (Seg) \ Dice (Loca)				
M1	**Méthode**	**Fold1**	**Fold2**	**Fold3**	**Fold4**	**Fold5**
Segmentation	Baseline	0,005176	0,004739	0,005124	0,006511	0,005508
Segmentation	Finetune	**0,003767**	**0,003649**	**0,003586**	**0,003709**	**0,003996**
Location	Baseline	0,9216	0,9185	0,9243	0,9265	0,9144
Location	Finetune	**0,9587**	**0,9562**	**0,9604**	**0,9645**	**0,9518**

Pour la localisation, le modèle affiné est également supérieur dans tous les folds, avec des scores Dice systématiquement plus élevés. Par exemple, dans le **fold 1**, le modèle affiné atteint un score Dice de **0.9587** contre **0.9216** pour le modèle de base, montrant une amélioration significative de la précision de localisation. Cette tendance se poursuit dans les autres folds avec des scores de **0.9562**, **0.9604**, **0.9645**, et **0.9518** pour les **folds 2, 3, 4, et 5** respectivement, dépassant constamment les scores du modèle de base. Cela démontre l'efficacité du fine-tuning pour capturer les caractéristiques spécifiques à la localisation dans le dataset M1, permettant une meilleure détection des limites internes et externes de l'iris.

En somme, le fine-tuning s'avère être un outil puissant pour améliorer les performances des modèles de localisation sur toutes les bases de données, tandis que pour la segmentation, son efficacité dépend des caractéristiques spécifiques de chaque dataset. La base de données M1 profite pleinement de l'apprentissage par transfert pour les deux tâches, tandis que pour les bases de données Africa et Asia, l'efficacité du fine-tuning pour la segmentation est plus variable. Ces résultats soulignent la nécessité d'une approche flexible et adaptée pour chaque jeu de données, en combinant validation croisée et ajustement spécifique des modèles pour obtenir les meilleures performances en reconnaissance de l'iris.

Après l'entraînement des trois modèles de segmentation et de localisation sur les bases de données Asia, Africa, et M1, une analyse comparative a été réalisée pour évaluer leurs performances respectives. Parmi ces modèles, celui issu de la base de données **Asia** s'est démarqué par sa précision et sa robustesse. Ce modèle a donc été sélectionné pour la suite des travaux. Sa performance supérieure a été démontrée par sa capacité à fournir des résultats très précis pour les tâches de segmentation et de localisation de l'iris de l'œil, notamment lorsqu'il a été appliqué aux bases de données **Thousand** et **Lamp**. Ces résultats mettent en évidence l'efficacité du modèle **Asia** pour traiter des données variées tout en conservant une excellente précision.

4.3.2 Résultats pour la Classification

Dans cette sous-section, nous analysons les performances des modèles d'apprentissage profond sur les images originales de la base de données CASIA-Iris-Thousand. Sans prétraitement, les modèles doivent extraire directement les caractéristiques pertinentes des données brutes, permettant une évaluation de leur efficacité intrinsèque pour la classification des images d'iris. Ces résultats offrent une base de comparaison essentielle pour mesurer l'impact des différentes méthodes de prétraitement. Ils mettent également en évidence la capacité des modèles.

Résultats de la base de données CASIA-Iris-Thousand

Image Originale

Les résultats présentés dans le tableau 4.5 obtenus avec les images originales montrent que **DensNet201** est le modèle le plus performant, atteignant une précision de **96,98%**, ce qui en fait une référence pour cette tâche sans prétraitement. **InceptionV3** suit avec une précision de **95,83%**, tandis que **ResNet50**, **ResNet34**, et **VGG16** affichent des performances respectives de **94,93%**, **95,77%**, et **92,23%**. Ces résultats indiquent que **DensNet201** extrait des caractéristiques pertinentes plus efficacement, ce qui le rend bien adapté aux images brutes.

TABLE 4.5 – Résultats de l'image originale sur la base de données Thousand.

Modèle	**Taille d'entrée**	**Taux d'apprentissage**	**Époque**	**Erreur**	**Précision**
Resnet50	64x512	0,0001	100	5,07%	94,93%
Resnet34	64x512	0,0001	100	4,23%	95,77%
VGG16	64x512	0,0001	100	7,77%	92,23%
Densnet201	**64x512**	**0,0001**	**100**	**3,02%**	**96,98%**
InceptionV3	299x299	0,0001	100	4,17%	95,83%

Égalisation d'histogramme

L'égalisation d'histogramme améliore considérablement les performances de tous les modèles, ce qui en fait une méthode de prétraitement très efficace. **DensNet201** atteint une précision impressionnante de **99,12%**, démontrant son efficacité accrue après ce traitement. **ResNet50** et **ResNet34** obtiennent également des précisions très élevées, avec respectivement **98,73%** et **98,70%**, ce qui montre leur robustesse face à ce prétraitement. Les modèles **InceptionV3** et **VGG16** montrent également des améliorations notables, atteignant respectivement **98,58%** et **96,55%**. Ces résultats, présentés dans le tableau 4.6, confirment que l'égalisation d'histogramme permet

d'améliorer significativement la qualité des caractéristiques extraites. Cette méthode renforce la cohérence des données en équilibrant leur distribution, ce qui facilite l'apprentissage des modèles.

TABLE 4.6 – Résultats de l'égalisation d'histogramme sur la BD Thousand.

Modèle	**Taille d'entrée**	**Taux d'apprentissage**	**Époque**	**Erreur**	**Précision**
Resnet50	64x512	0,0001	100	1,27%	98,73%
Resnet34	64x512	0,0001	100	1,30%	98,70%
VGG16	64x512	0,0001	100	3,45%	96,55%
Densnet201	**64x512**	**0,0001**	**100**	**0,88%**	**99,12%**
InceptionV3	299x299	0,0001	100	1,42%	98,58%

Filtre de Gabor

Les performances avec le filtre de Gabor restent inférieures par rapport aux autres méthodes de prétraitement. **ResNet50** et **ResNet34** affichent des précisions modestes de **78,18%** et **79,33%**, respectivement, avec des erreurs relativement élevées. **VGG16** et **InceptionV3** montrent des résultats légèrement meilleurs, atteignant des précisions de **81,85%** et **85,18%**. **DensNet201** se distingue comme le modèle le plus performant, avec une précision de **88,83%** et une erreur de **12,17%**, bien que cela reste en deçà de ses performances avec d'autres prétraitements. Ces résultats, présentés dans le tableau 4.7, confirment que le filtre de Gabor n'est pas idéal pour cette tâche, bien qu'il puisse être utile dans des contextes spécifiques.

TABLE 4.7 – Résultats de Filtre de Gabor sur la base de données Thousand.

Modèle	**Taille d'entrée**	**Taux d'apprentissage**	**Époque**	**Erreur**	**Précision**
Resnet50	64x512	0,0001	100	21,82%	78,18%
Resnet34	64x512	0,0001	100	20,67%	79,33%
VGG16	64x512	0,0001	100	18,15%	81,85%
Densnet201	**64x512**	**0,0001**	**100**	**12,17%**	**88,83%**
InceptionV3	299x299	0,0001	100	14,82%	85,18%

Modèle binaire local LBP

La méthode LBP se distingue par des performances similaires à celles obtenues avec l'égalisation d'histogramme. **DensNet201** et **ResNet50** se démarquent avec des précisions respectives de **97,73%** et **97,68%**, confirmant leur efficacité dans ce contexte. **ResNet34** suit de près avec une précision de **97,47%**, tandis que **InceptionV3** obtient la meilleure précision globale à **98,38%** grâce à une erreur minimale de **1,62%**. Enfin, **VGG16** affiche une performance légèrement inférieure avec une précision de **96,12%**, mais reste compétitive. Ces résultats, présentés dans le tableau 4.8, démontrent que la méthode LBP est une approche de prétraitement très efficace, particulièrement en combinaison avec des modèles performants comme **DensNet201**, **ResNet50**, et **InceptionV3**.

TABLE 4.8 – Résultats de Modèle binaire local (LBP) sur la BD Thousand.

Modèle	**Taille d'entrée**	**Taux d'apprentissage**	**Époque**	**Erreur**	**Précision**
Resnet50	64x512	0,0001	100	2,32%	97,68%
Resnet34	64x512	0,0001	100	2,53%	97,47%
VGG16	64x512	0,0001	100	3,88%	96,12%
Densnet201	64x512	0,0001	100	2,27%	97,73%
InceptionV3	**299x299**	**0,0001**	**100**	**1,62%**	**98,38%**

Comparaison des méthodes

Le tableau récapitulatif 4.9 confirme que **DensNet201** est le modèle le plus performant de manière générale, atteignant une précision exceptionnelle de **99,12%** grâce à l'égalisation d'histogramme, ce qui en fait la meilleure combinaison modèle-méthode. **ResNet50** et **ResNet34** montrent également une grande robustesse, avec des performances élevées, notamment avec l'égalisation d'histogramme et la méthode LBP, où leurs précisions avoisinent celles de **DensNet201**. Cependant, les performances des modèles chutent considérablement avec le filtre de Gabor, révélant son inadéquation

pour cette tâche. Cette analyse souligne l'importance du choix de la méthode de prétraitement pour maximiser les performances des modèles, en mettant en avant l'efficacité de l'égalisation d'histogramme et de la méthode LBP dans le contexte de la reconnaissance de l'iris.

TABLE 4.9 – Résultats de toutes les méthodes sur la base de données Thousand.

Modèle	**Image original**	**Égalisation d'histogramme**	**Filtre de Gabor**	**LBP**
Resnet50	94,93%	98,73%	78,18%	97,68%
Resnet34	95,77%	98,70%	79,33%	97,47%
VGG16	92,23%	96,55%	81,85%	96,12%
Densnet201	96,98%	**99,12%**	88,83%	97,73%
InceptionV3	95,83%	98,58%	85,18%	98,38%

Globalement, les résultats mettent en évidence l'efficacité de l'égalisation d'histogramme et de la méthode LBP comme prétraitements dominants telque illustré dans la Figure 4.1, ainsi que la supériorité des modèles comme **DensNet201**, **ResNet50**, et **ResNet34** pour la classification des images de cette base de données.

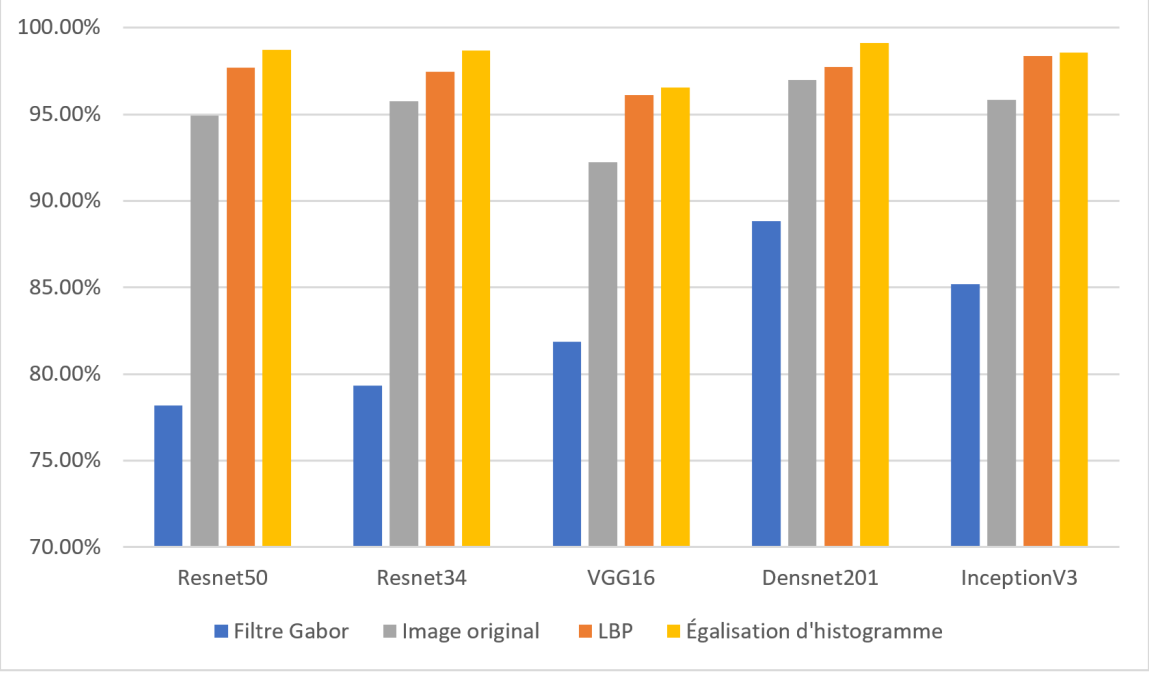

FIGURE 4.1 – Comparaison entre les méthodes sur la base de données Thousand.

Résultats de la base de données CASIA-Iris-Lamp

Image Originale

Pour les images originales, **DensNet201** est le modèle le plus performant, atteignant une précision de **99,82%** avec une erreur minimale de **0,18%**, démontrant sa capacité à extraire efficacement des caractéristiques pertinentes. **ResNet50** et **ResNet34** suivent de près avec des précisions de **99,79%** et une erreur de **0,21%**. **InceptionV3** atteint une précision de **99,65%**, tandis que **VGG16** affiche une précision légèrement inférieure de **99,26%**. Ces résultats montrent une excellente performance globale des modèles même sans prétraitement et sont présentés dans le tableau 4.10.

TABLE 4.10 – Résultats de l'image originale sur la base de données Lamp.

Modèle	**Taille d'entrée**	**Taux d'apprentissage**	**Époque**	**Erreur**	**Précision**
Resnet50	64x512	0,0001	100	0,21%	99,79%
Resnet34	64x512	0,0001	100	0,21%	99,79%
VGG16	64x512	0,0001	100	0,74%	99,26%
Densnet201	**64x512**	**0,0001**	**100**	**0,18%**	**99,82%**
InceptionV3	299x299	0,0001	100	0,35%	99,65%

Égalisation d'histogramme

L'égalisation d'histogramme améliore légèrement les performances de presque tous les modèles. Le tableau 4.11 montre que **DensNet201** se distingue à nouveau avec une précision maximale de **99,88%** et une erreur réduite à **0,12%**, consolidant sa position comme le modèle le plus performant. **ResNet34** suit avec une précision de **99,86%** et une erreur de **0,14%**, dépassant légèrement **ResNet50** à **99,84%**. **InceptionV3** atteint une précision de **99,67%**, tandis que **VGG16** reste stable à **99,22%**. Ces résultats confirment l'efficacité de l'égalisation d'histogramme comme méthode de prétraitement.

TABLE 4.11 – Résultats de l'égalisation d'histogramme sur la base de données Lamp.

Modèle	**Taille d'entrée**	**Taux d'apprentissage**	**Époque**	**Erreur**	**Précision**
Resnet50	64x512	0,0001	100	0,16%	99,84%
Resnet34	64x512	0,0001	100	0,14%	99,86%
VGG16	64x512	0,0001	100	0,78%	99,22%
Densnet201	**64x512**	**0,0001**	**100**	**0,12%**	**99,88%**
InceptionV3	299x299	0,0001	100	0,33%	99,67%

Filtre de Gabor

Le tableau 4.12 montre les Résultats Avec le filtre de Gabor, les performances des modèles chutent globalement. **DensNet201** reste le plus performant avec une précision de **97,91%** et une erreur de **12,17%**, suivi par **InceptionV3** à **97,74%**. Les modèles **ResNet50** et **ResNet34** affichent des précisions de **96,41%** et **97,33%**, respectivement, tandis que **VGG16** atteint **96,92%**. Ces résultats montrent que le filtre de Gabor est moins adapté pour ce type de données.

TABLE 4.12 – Résultats de Filtre de Gabor sur la base de données Lamp.

Modèle	**Taille d'entrée**	**Taux d'apprentissage**	**Époque**	**Erreur**	**Précision**
Resnet50	64x512	0,0001	100	3,59%	96,41%
Resnet34	64x512	0,0001	100	2,67%	97,33%
VGG16	64x512	0,0001	100	3,08%	96,92%
Densnet201	**64x512**	**0,0001**	**100**	**2,09%**	**97,91%**
InceptionV3	299x299	0,0001	100	2,26%	97,74%

Modèle binaire local LBP

La méthode LBP produit des performances très élevées, rivalisant avec celles obtenues avec l'égalisation d'histogramme. Le tableau 4.13 montre que le modele **InceptionV3** se distingue cette fois avec une précision de **99,30%** et une erreur de **1,62%**, tandis que **DensNet201** suit avec **99,61%**. **ResNet34** et **ResNet50** affichent des

précisions respectives de **99,65%** et **99,57%**, confirmant leur robustesse. **VGG16**, bien qu'en deçà des autres modèles, reste performant avec une précision de **98,50%**.

TABLE 4.13 – Résultats de Modèle binaire local (LBP) sur la base de données Lamp.

Modèle	**Taille d'entrée**	**Taux d'apprentissage**	**Époque**	**Erreur**	**Précision**
Resnet50	64x512	0,0001	100	0,43%	99,57%
Resnet34	**64x512**	**0,0001**	**100**	**0,35%**	**99,65%**
VGG16	64x512	0,0001	100	1,50%	98,50%
Densnet201	64x512	0,0001	100	0,39%	99,61%
InceptionV3	299x299	0,0001	100	0,70%	99,30%

Comparaison des méthodes

Le tableau récapitulatif 4.14 met en évidence que **DensNet201**, combiné à **l'égalisation d'histogramme**, offre les meilleures performances globales avec une précision maximale de **99,88%**. **InceptionV3** et **ResNet34** montrent également d'excellentes capacités sur plusieurs méthodes. En revanche, le filtre de Gabor se révèle moins performant, montrant des baisses significatives de précision pour tous les modèles. Globalement, l'égalisation d'histogramme et la méthode LBP sont les prétraitements les plus efficaces, et **DensNet201** reste le modèle de référence pour la classification des images dans cette base de données.

TABLE 4.14 – Résultats de toutes les méthodes sur la base de données Lamp.

Modèle	**Image original**	**Égalisation d'histogramme**	**Filtre de Gabor**	**LBP**
Resnet50	99,79%	99,84%	96,41%	99,57%
Resnet34	99,79%	99,86%	97,33%	99,65%
VGG16	99,26%	99,22%	96,92%	98,50%
Densnet201	99,82%	**99,88%**	97,91%	99,61%
InceptionV3	99,65%	99,67%	97,74%	99,30%

La Figure 4.2 illustre les performances de différents modèles de deep learning (*ResNet50*, *ResNet34*, *VGG16*, *DenseNet201* et *InceptionV3*) sur des images d'iris

prétraitées selon quatre techniques : filtre de Gabor, images originales, *LBP*, et égalisation d'histogramme. Les modèles montrent une meilleure précision pour les images prétraitées avec *LBP* et l'égalisation d'histogramme, atteignant presque 100 %, tandis que les images filtrées par Gabor offrent des performances plus faibles. *DenseNet201* et *InceptionV3* semblent particulièrement robustes indépendamment de la méthode de prétraitement. Ces résultats mettent en évidence l'importance du choix du prétraitement pour optimiser la performance des modèles.

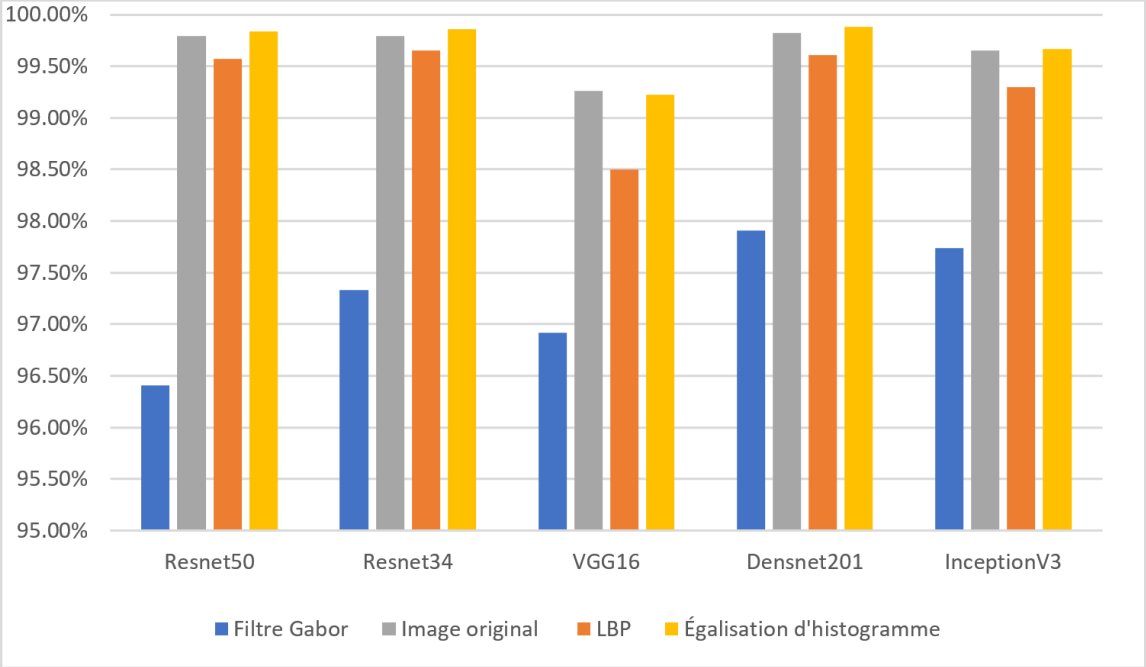

FIGURE 4.2 – Comparaison entre les méthodes sur la base de données Lamp.

4.3.3 Étude comparative sur la reconnaissance de l'iris

Notre méthode, basée sur un modèle hybride combinant **ResNet34-UNet** pour la segmentation,la localisation et **DenseNet201** avec l'égalisation d'histogramme pour la classification, atteint une précision de **99,12%** sur la base CASIA Thousand, surpassant ou égalant toutes les approches précédentes. Par exemple, l'approche de *Liu*

& Mei [31] atteint une précision de **98,56%**, soit un écart de **0,56%**, tandis que *Salve & Narote* [32] obtiennent **92,50%** avec **ANN** et **95,90%** avec **SVM**, présentant des écarts respectifs de **6,62%** et **3,22%**. Ces résultats reflètent les limites des modèles traditionnels face à des bases de données complexes. De même, l'approche de *Jie, Zhe-Ming & Zhou* [33] avec **KNN** atteint **96,30%**, soit un écart de **2,82%**, et celle de *Ahmed Sahran* [34] avec **DCT + ANN** affiche une précision de **96,00%**, inférieure de **3,12%** à notre méthode.

L'approche de *Kumar et al.* [35], combinant **PCA + DWT + ANN**, se rapproche de notre résultat avec une précision de **99,07%**, mais reste légèrement en dessous avec un écart de **0,05%**. Ces différences démontrent l'efficacité des technologies modernes, notamment des réseaux de neurones profonds, qui surpassent les techniques classiques comme PCA, DWT ou LDA. De plus, notre méthode bénéficie d'une base de données plus grande et diversifiée (1000 classes et 20 000 images), offrant une meilleure généralisation. Ainsi, notre méthode se distingue par sa capacité à exploiter des modèles avancés pour atteindre une précision optimale, confirmant la supériorité des approches modernes de deep learning dans la reconnaissance de l'iris. Voir le tableau 4.15

TABLE 4.15 – Étude comparative sur la reconnaissance de l'iris

Chercheurs	**Méthode**	**Base de données**	**Classes**	**Images totales**	**Précision**
Chengqiang & Mei [31]	DLDA + ED	CASIA V2	60	1200	98,56%
Salve & Narote [32]	1D Log GW + ANN	CASIA V4	1000	20 000	92,50%
Salve & Narote [32]	1D Log GW + SVM	CASIA V4	1000	20 000	95,90%
Jie, Zhe-Ming & Zhou [33]	(CT + PCA + LDA) + KNN	CASIA V4	27	270	96,30%
Ahmed Sahran [34]	DCT + ANN	CASIA V2	60	600	96,00%
Kumar et al. [35]	PCA + DWT + ANN	CASIA V1	108	756	99,07%
Notre méthode	**ResNet34-UNet + DenseNet201**	**CASIA V4**	**1000**	**20 000**	**99,12%**

Chapitre 5

Conclusion et perspectives

Ce mémoire a exploré une approche hybride combinant segmentation et classification pour améliorer la reconnaissance biométrique de l'iris, en relevant les défis posés par les conditions d'acquisition non coopératives. Grâce à l'utilisation d'un modèle U-Net modifié pour la segmentation, intégrant une architecture ResNet34, et de cinq architectures CNN pour la classification (ResNet34, ResNet50, VGG16, InceptionV3 et DenseNet201), des performances significatives ont été obtenues. Les techniques de prétraitement appliquées – égalisation d'histogramme, filtres de Gabor et motifs binaires locaux (LBP) – ont également permis d'optimiser la qualité des données et d'améliorer la précision des modèles.

Les expérimentations menées sur cinq sous-ensembles de la base de données CASIA-IrisV4 ont démontré une amélioration notable des performances, en termes de précision et de stabilité. Ces résultats confirment la robustesse et l'efficacité de l'approche proposée, mettant en évidence son potentiel pour des applications pratiques, notamment dans des contextes biométriques exigeants.

Enfin, cette étude ouvre la voie à plusieurs perspectives. Parmi celles-ci, l'intégration

de techniques d'apprentissage auto-supervisé pour réduire la dépendance aux données annotées, ou encore l'exploration de modèles d'ensemble pour combiner les forces des architectures CNN utilisées. De plus, l'application de cette approche à d'autres bases de données, incluant des images capturées dans des environnements encore plus contraignants, pourrait enrichir davantage les contributions de ce travail. Ainsi, cette recherche constitue une avancée significative pour le domaine de la reconnaissance de l'iris et ses nombreuses applications.

Bibliographie

[1] R. Alrawili, A. A. S. AlQahtani et M. K. Khan, « Comprehensive survey : Biometric user authentication application, evaluation, and discussion », *Computers and Electrical Engineering*, vol. 119, p. 109485, 2024.

[2] A. Bertillon, *La couleur de l'iris*. Masson, 1886.

[3] A. Budiman, R. A. Yaputera, S. Achmad, A. Kurniawan *et al.*, « Student attendance with face recognition (lbph or cnn) : Systematic literature review », *Procedia Computer Science*, vol. 216, p. 31–38, 2023.

[4] S. Wang, X. He, Z. Jian, J. Li, C. Xu, Y. Chen, Y. Liu, H. Chen, C. Huang, J. Hu *et al.*, « Advances and prospects of multi-modal ophthalmic artificial intelligence based on deep learning : a review », *Eye and Vision*, vol. 11, no. 1, p. 38, 2024.

[5] T. Schlett, C. Rathgeb et C. Busch, « Multi-spectral iris segmentation in visible wavelengths », *in 2018 International Conference on Biometrics (ICB)*, p. 190–194, IEEE, 2018.

[6] X. Wu et L. Zhao, « Study on iris segmentation algorithm based on dense u-net », *IEEE Access*, vol. 7, p. 123959–123968, 2019.

[7] Y. Ganeeva et E. Myasnikov, « Using convolutional neural networks for segmentation of iris images », *in 2020 International Multi-Conference on Industrial Engineering and Modern Technologies (FarEastCon)*, p. 1–4, IEEE, 2020.

[8] Y.-H. Li, P.-J. Huang et Y. Juan, « An efficient and robust iris segmentation algorithm using deep learning », *Mobile Information Systems*, vol. 2019, no. 1,

p. 4568929, 2019.

[9] C. Wang, Y. Wang, K. Zhang, J. Muhammad, T. Lu, Q. Zhang, Q. Tian, Z. He, Z. Sun, Y. Zhang *et al.*, « Nir iris challenge evaluation in non-cooperative environments : Segmentation and localization », *in 2021 IEEE International joint conference on biometrics (IJCB)*, p. 1–10, IEEE, 2021.

[10] V. Varkarakis, S. Bazrafkan et P. Corcoran, « Deep neural network and data augmentation methodology for off-axis iris segmentation in wearable headsets », *Neural Networks*, vol. 121, p. 101–121, 2020.

[11] A. Boyd, A. Czajka et K. Bowyer, « Deep learning-based feature extraction in iris recognition : Use existing models, fine-tune or train from scratch ? », *in 2019 IEEE 10th International Conference on Biometrics Theory, Applications and Systems (BTAS)*, p. 1–9, IEEE, 2019.

[12] S. Minaee, A. Abdolrashidiy et Y. Wang, « An experimental study of deep convolutional features for iris recognition », *in 2016 IEEE signal processing in medicine and biology symposium (SPMB)*, p. 1–6, IEEE, 2016.

[13] K. Nguyen, C. Fookes, A. Ross et S. Sridharan, « Iris recognition with off-the-shelf cnn features : A deep learning perspective », *Ieee Access*, vol. 6, p. 18848–18855, 2017.

[14] T. Zhao, Y. Liu, G. Huo et X. Zhu, « A deep learning iris recognition method based on capsule network architecture », *IEEE Access*, vol. 7, p. 49691–49701, 2019.

[15] G. Savithiri et A. Murugan, « Performance analysis on half iris feature extraction using gw, lbp and hog », *International journal of computer applications*, vol. 22, no. 2, p. 27–32, 2011.

[16] K. Saminathan, T. Chakravarthy et M. C. Devi, « Iris recognition based on kernels of support vector machine », *ICTACT J Soft Comput*, vol. 5, no. 2, p. 889–895, 2015.

[17] R. Sharma et A. Ross, « D-netpad : An explainable and interpretable iris presentation attack detector », *in 2020 IEEE international joint conference on biometrics (IJCB)*, p. 1–10, IEEE, 2020.

[18] C. Chen et A. Ross, « An explainable attention-guided iris presentation attack detector », *in Proceedings of the IEEE/CVF winter conference on applications of computer vision*, p. 97–106, 2021.

[19] J. E. Tapia, S. Gonzalez et C. Busch, « Iris liveness detection using a cascade of dedicated deep learning networks », *IEEE Transactions on Information Forensics and Security*, vol. 17, p. 42–52, 2021.

[20] M. Trokielewicz et A. Czajka, « Data-driven segmentation of post-mortem iris images », *in 2018 International Workshop on Biometrics and Forensics (IWBF)*, p. 1–7, IEEE, 2018.

[21] A. Kuehlkamp, A. Boyd, A. Czajka, K. Bowyer, P. Flynn, D. Chute et E. Benjamin, « Interpretable deep learning-based forensic iris segmentation and recognition », *in Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, p. 359–368, 2022.

[22] O. Ronneberger, P. Fischer et T. Brox, « U-net : Convolutional networks for biomedical image segmentation », *Springer*, p. 234–241, 2015.

[23] K. He, X. Zhang, S. Ren et J. Sun, « Deep residual learning for image recognition », *Proceedings of the IEEE conference on computer vision and pattern recognition*, p. 770–778, 2016.

[24] J. Daugman, « High confidence visual recognition of persons by a test of statistical independence », *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 11, p. 1148–1161, 1993.

[25] R. C. Gonzalez, *Digital image processing*. Pearson education india, 2009.

[26] T. Ojala, T. Maenpaa, M. Pietikainen, J. Viertola, J. Kyllonen et S. Huovinen, « Outex-new framework for empirical evaluation of texture analysis algorithms », *IEEE International Conference Mechatronics and Automation*, vol. 1, p. 701–706, 2002.

[27] K. He, X. Zhang, S. Ren et J. Sun, « Deep residual learning for image recognition », *in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, p. 770–778, 2016.

[28] K. Simonyan et A. Zisserman, « Very deep convolutional networks for large-scale image recognition », *arXiv preprint arXiv :1409.1556*, 2014.

[29] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens et Z. Wojna, « Rethinking the inception architecture for computer vision », *in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, p. 2818–2826, 2016.

[30] G. Huang, Z. Liu, L. Van Der Maaten et K. Q. Weinberger, « Densely connected convolutional networks », *in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, p. 4700–4708, 2017.

[31] C. Liu et X. Mei, « Iris recognition based on dlda », *in 18th International Conference on Pattern Recognition (ICPR'06)*, vol. 4, p. 489–492, IEEE, 2006.

[32] P. Salve et S. Narote, « Iris recognition using svm and ann », *2016 International Conference on Computing, Analytics and Security Trends (CAST)*, p. 289–293, 2016.

[33] W. Jie, L. Zhe-Ming et J. Zhou, « Iris recognition based on contourlet transform and knn », *2010 International Conference on Machine Learning and Cybernetics*, vol. 6, p. 3130–3134, 2010.

[34] A. Sahran, « Iris recognition using discrete cosine transform and artificial neural networks », *International Journal of Computer Applications*, vol. 11, no. 12, p. 1–5, 2010.

[35] M. Kumar, S. Chootaray, K. B. Raja et S. Pattnaik, « Pca, dwt and ann based approach for iris recognition », *International Journal of Computer Science and Information Security*, vol. 7, no. 2, p. 305–312, 2010.

Annexe A

Scripts utilisés

```
###################### ResNet34 + Unet ############################

    lr_max = 0.0002
    L2 = 0.0001

    save_name = 'bs{}_epoch{}_fold{}'.format(args.bs, args.epoch, args.fold)
    save_dir = os.path.join('trained_models/{}/Seg/{}_{}'.format(description, args.name, args.net), save_name)
    if os.path.exists(save_dir):
        shutil.rmtree(save_dir)
    os.makedirs(save_dir, exist_ok=True)
    train_writer = SummaryWriter(os.path.join(save_dir, 'log/train'), flush_secs=2)
    val_writer = SummaryWriter(os.path.join(save_dir, 'log/val'), flush_secs=2)
    print(os.path.join(save_dir))

    print('data loading')
    train_data = Dataset_train(data_root=data_root, split_file=split_file, size=input_size, fold=args.fold, resize=resize)
    train_dataloader = DataLoader(dataset=train_data, batch_size=args.bs, shuffle=True, num_workers=8, pin_memory=True, persistent_workers=True)
    val_data = Dataset_val(data_root=data_root, split_file=split_file, size=input_size, fold=args.fold, resize=resize)
    val_dataloader = DataLoader(dataset=val_data, batch_size=args.bs, shuffle=False, num_workers=8, pin_memory=True, persistent_workers=True)

    print('model loading')
    if args.net.lower() == 'unet_resnet34':
        net = smp.Unet('resnet34', in_channels=1, classes=1, activation=None).cuda()
    if args.net.lower() == 'unet_resnet101':
        net = smp.Unet('resnet101', in_channels=1, classes=1, activation=None).cuda()

    train_data_len = train_data.len
    val_data_len = val_data.len
    print('train_lenth: %i  val_lenth: %i' % (train_data_len, val_data_len))

    criterion = dice_BCE_loss(0.5, 0.5)
    optimizer = optim.Adam(net.parameters(), lr=lr_max, weight_decay=L2)
    scheduler = MultiStepLR(optimizer, milestones=[int((5 / 10) * args.epoch),
                                                   int((8 / 10) * args.epoch)], gamma=0.1, last_epoch=-1)
    best_acc = 0
    IMAGE = []
    SEG = []

    print('training')
    for epoch in range(args.epoch):
        net.train()
        for param_group in optimizer.param_groups:
            lr = param_group['lr']
            break
        print('lr for this epoch:', lr)
        epoch_train_loss = []
        epoch_train_acc = []
        for i, (inputs, segs) in enumerate(train_dataloader):
            inputs, segs = inputs.float().cuda(), segs.float().cuda()
            optimizer.zero_grad()
            result = net(inputs)
            result = torch.sigmoid(result)
            train_loss = criterion(result, segs)
```

```python
# Define model paths based on args.model
if args.model.lower() == 'asia':
    seg_model_path = [
        'trained_models/Asia/Seg/baseline_UNet_ResNet34/bs16_epoch500_fold0/best_acc.pth',
        'trained_models/Asia/Seg/baseline_UNet_ResNet34/bs16_epoch500_fold1/best_acc.pth',
        'trained_models/Asia/Seg/baseline_UNet_ResNet34/bs16_epoch500_fold2/best_acc.pth',
        'trained_models/Asia/Seg/baseline_UNet_ResNet34/bs16_epoch500_fold3/best_acc.pth',
        'trained_models/Asia/Seg/finetune_UNet_ResNet34/bs16_epoch200_fold4/best_acc.pth'
    ]
    local_model_path = [
        'trained_models/Asia/Local_inner_outer/finetune_UNet_ResNet34/bs16_epoch200_fold0/best_model.pth',
        'trained_models/Asia/Local_inner_outer/finetune_UNet_ResNet34/bs16_epoch200_fold1/best_model.pth',
        'trained_models/Asia/Local_inner_outer/finetune_UNet_ResNet34/bs16_epoch200_fold2/best_model.pth',
        'trained_models/Asia/Local_inner_outer/baseline_UNet_ResNet34/bs16_epoch500_fold3/best_model.pth',
        'trained_models/Asia/Local_inner_outer/baseline_UNet_ResNet34/bs16_epoch500_fold4/best_model.pth'
    ]
    input_size = (480, 640)
elif args.model.lower() == 'africa':
    seg_model_path = [
        'trained_models/Africa/Seg/baseline_UNet_ResNet34/bs16_epoch500_fold0/best_acc.pth',
        'trained_models/Africa/Seg/finetune_UNet_ResNet34/bs16_epoch200_fold1/best_acc.pth',
        'trained_models/Africa/Seg/finetune_UNet_ResNet34/bs16_epoch200_fold2/best_acc.pth',
        'trained_models/Africa/Seg/baseline_UNet_ResNet34/bs16_epoch500_fold3/best_acc.pth',
        'trained_models/Africa/Seg/finetune_UNet_ResNet34/bs16_epoch200_fold4/best_acc.pth'
    ]
    local_model_path = [
        'trained_models/Africa/Local_inner_outer/finetune_UNet_ResNet34/bs16_epoch200_fold0/best_model.pth',
        'trained_models/Africa/Local_inner_outer/finetune_UNet_ResNet34/bs16_epoch200_fold1/best_model.pth',
        'trained_models/Africa/Local_inner_outer/finetune_UNet_ResNet34/bs16_epoch200_fold2/best_model.pth',
        'trained_models/Africa/Local_inner_outer/finetune_UNet_ResNet34/bs16_epoch200_fold3/best_model.pth',
        'trained_models/Africa/Local_inner_outer/finetune_UNet_ResNet34/bs16_epoch200_fold4/best_model.pth'
    ]
    input_size = (384, 640)
elif args.model.lower() == 'm1':
    seg_model_path = [
        'trained_models/M1/Seg/finetune_UNet_ResNet34/bs28_epoch100_fold0/best_acc.pth',
        'trained_models/M1/Seg/finetune_UNet_ResNet34/bs28_epoch100_fold1/best_acc.pth',
        'trained_models/M1/Seg/finetune_UNet_ResNet34/bs28_epoch100_fold2/best_acc.pth',
        'trained_models/M1/Seg/finetune_UNet_ResNet34/bs28_epoch100_fold3/best_acc.pth',
        'trained_models/M1/Seg/finetune_UNet_ResNet34/bs28_epoch100_fold4/best_acc.pth'
    ]
    local_model_path = [
        'trained_models/M1/Local_inner_outer/finetune_UNet_ResNet34/bs24_epoch150_fold0/best_model.pth',
        'trained_models/M1/Local_inner_outer/finetune_UNet_ResNet34/bs24_epoch150_fold1/best_model.pth',
        'trained_models/M1/Local_inner_outer/finetune_UNet_ResNet34/bs24_epoch150_fold2/best_model.pth',
        'trained_models/M1/Local_inner_outer/finetune_UNet_ResNet34/bs24_epoch150_fold3/best_model.pth',
        'trained_models/M1/Local_inner_outer/finetune_UNet_ResNet34/bs24_epoch150_fold4/best_model.pth'
    ]
    input_size = (416, 416)
```

```
##########################################################################################################################

# Unwrapping function (add this part in your script)
def unwrap_iris(masked_img_path, inner_boundary_path, outer_boundary_path):
    # Load the RGBA image (no_bg) with transparency
    no_bg_image = cv2.imread(masked_img_path, cv2.IMREAD_UNCHANGED)

    # Load the boundary images (inner and outer)
    inner_boundary = io.imread(inner_boundary_path)
    outer_boundary = io.imread(outer_boundary_path)

    if no_bg_image is None or inner_boundary is None or outer_boundary is None:
        print(f"Error: Unable to load one of the required images.")
        return

    # Get the center of the pupil and the iris from the inner and outer boundaries
    inner_contours, _ = cv2.findContours(inner_boundary, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
    outer_contours, _ = cv2.findContours(outer_boundary, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

    inner_contour = max(inner_contours, key=cv2.contourArea)
    outer_contour = max(outer_contours, key=cv2.contourArea)

    # Fit circles to the contours to get the centers and radii
    (center_x, center_y), radius_pupil = cv2.minEnclosingCircle(inner_contour)
    (outer_center_x, outer_center_y), outer_radius = cv2.minEnclosingCircle(outer_contour)

    center_x, center_y, radius_pupil, outer_radius = int(center_x), int(center_y), int(radius_pupil), int(outer_radius)

    # Define the iris radius (distance between pupil and iris outer boundary)
    iris_radius = outer_radius - radius_pupil

    # Unwrap the iris using polar coordinates
    nsamples = 360
    samples = np.linspace(0, 2 * np.pi, nsamples)[:-1]
    polar = np.zeros((iris_radius, nsamples, 4), dtype=np.uint8)  # RGBA output

    for r in range(iris_radius):
        for theta in samples:
            x = int((r + radius_pupil) * np.cos(theta) + center_x)
            y = int((r + radius_pupil) * np.sin(theta) + center_y)
            try:
                polar[r, int((theta * nsamples) / (2 * np.pi))] = no_bg_image[y, x]
            except IndexError:
                pass

    # Resize to standard output size
    unwrapped = cv2.resize(polar, (512, 64))
    return unwrapped

##########################################################################################################################
```

```
############### DATA Loader ##################

class IrisDataset(Dataset):
    def __init__(self, root_dir, transform=None):
        self.root_dir = root_dir
        self.transform = transform
        self.image_paths = []
        self.labels = []

        # Loop through 411 folders (001 to 411)
        for folder_num in range(1, 412): # (1, 412)   (1000)
            folder_path = os.path.join(root_dir, f"{folder_num:03d}")
            left_path = os.path.join(folder_path, "L")
            right_path = os.path.join(folder_path, "R")

            # Label is based on the folder number
            label = folder_num   # Labels start from 0

            # Add images from L and R folders
            for side in ["L", "R"]:
                side_path = os.path.join(folder_path, side)
                for img_name in os.listdir(side_path):
                    if img_name.endswith("_lbp.png"): #  _unwrapped   _enhanced   _Gabor  _lbp
                        img_path = os.path.join(side_path, img_name)
                        self.image_paths.append(img_path)
                        self.labels.append(label-1)  # remove -1 with casia v4 thousand

    def __len__(self):
        return len(self.image_paths)

    def __getitem__(self, idx):
        image = Image.open(self.image_paths[idx]).convert("RGB")
        label = self.labels[idx]
        if self.transform:
            image = self.transform(image)
        return image, label

# Define transformations (resize to 64x512 and normalize)
transform = transforms.Compose([
    transforms.Resize((299, 299)), # 64x512 with resnet  299x299 with inceptionV3
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])

# Create dataset and dataloader
train_dir = r'C:\Users\Mouad\Desktop\Final Project\DATABASE\CASIA-IrisV4\CASIA-Iris-Lamp-normalized\train'
train_dataset = IrisDataset(root_dir=train_dir, transform=transform)
validation_dir = r'C:\Users\Mouad\Desktop\Final Project\DATABASE\CASIA-IrisV4\CASIA-Iris-Lamp-normalized\validation'
val_dataset = IrisDataset(root_dir=validation_dir, transform=transform)
```

```
############  ResNet50 ######################

# Define the model, optimizer, and loss function
import torch.nn as nn
import torch.optim as optim
from torchvision import models

# Load pre-trained ResNet-50 model and modify for 411 classes
model = models.resnet50(pretrained=True)
model.fc = nn.Linear(model.fc.in_features, 411)  # Adjust to 411 in caisa Lamp and 1000 in casia thousand

# Move model to device (GPU if available)
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = model.to(device)

# Define optimizer and loss function
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.0001)


# Create DataLoaders for training and validation sets
train_loader = DataLoader(train_dataset, batch_size=64, shuffle=True)
val_loader = DataLoader(val_dataset, batch_size=64, shuffle=False)

def train_model(model, train_loader, val_loader, criterion, optimizer, num_epochs):

    best_val_accuracy = 0.0  # Track the best validation accuracy
    best_model_weights = None  # Store the weights of the best model

    for epoch in range(num_epochs):
        # Training phase
        model.train()
        running_loss, correct_train, total_train = 0.0, 0, 0
        for images, labels in train_loader:
            images, labels = images.to(device), labels.to(device)
            optimizer.zero_grad()
            outputs = model(images)
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()

            running_loss += loss.item()
            _, predicted = torch.max(outputs, 1)
            correct_train += (predicted == labels).sum().item()
            total_train += labels.size(0)

        train_loss = running_loss / len(train_loader)
        train_accuracy = correct_train / total_train * 100
```

```
############  ResNet34 ######################

# Define the model, optimizer, and loss function
import torch.nn as nn
import torch.optim as optim
from torchvision import models

# Load pre-trained ResNet-34 model and modify for 411 classes
model = models.resnet34(pretrained=True)
model.fc = nn.Linear(model.fc.in_features, 411)  # Adjust to 411 in caisa Lamp and 1000 in casia thousand

# Move model to device (GPU if available)
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = model.to(device)

# Define optimizer and loss function
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.0001)

from torch.utils.data import random_split


# Create DataLoaders for training and validation sets
train_loader = DataLoader(train_dataset, batch_size=64, shuffle=True)
val_loader = DataLoader(val_dataset, batch_size=64, shuffle=False)

def train_model(model, train_loader, val_loader, criterion, optimizer, num_epochs):

    best_val_accuracy = 0.0  # Track the best validation accuracy
    best_model_weights = None  # Store the weights of the best model

    for epoch in range(num_epochs):
        # Training phase
        model.train()
        running_loss, correct_train, total_train = 0.0, 0, 0
        for images, labels in train_loader:
            images, labels = images.to(device), labels.to(device)
            optimizer.zero_grad()
            outputs = model(images)
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()

            running_loss += loss.item()
            _, predicted = torch.max(outputs, 1)
            correct_train += (predicted == labels).sum().item()
            total_train += labels.size(0)

        train_loss = running_loss / len(train_loader)
        train_accuracy = correct_train / total_train * 100
```

```
################## VGG16 ###############

import torch
import torch.nn as nn
import torch.optim as optim
from torchvision import models
from torch.utils.data import DataLoader, random_split

# Load pre-trained VGG16 model and modify for 411 classes
model = models.vgg16(pretrained=True)
model.classifier[6] = nn.Linear(model.classifier[6].in_features, 411)  # Adjust to 411 in caisa Lamp and 1000 in casia thousand
# Move model to device (GPU if available)
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = model.to(device)

# Define optimizer and loss function
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.0001)



# Create DataLoaders for training and validation sets
train_loader = DataLoader(train_dataset, batch_size=64, shuffle=True)
val_loader = DataLoader(val_dataset, batch_size=64, shuffle=False)

def train_model(model, train_loader, val_loader, criterion, optimizer, num_epochs):
    best_val_accuracy = 0.0  # Track the best validation accuracy
    best_model_weights = None  # Store the weights of the best model

    for epoch in range(num_epochs):
        # Training phase
        model.train()
        running_loss, correct_train, total_train = 0.0, 0, 0
        for images, labels in train_loader:
            images, labels = images.to(device), labels.to(device)
            optimizer.zero_grad()
            outputs = model(images)
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()

            running_loss += loss.item()
            _, predicted = torch.max(outputs, 1)
            correct_train += (predicted == labels).sum().item()
            total_train += labels.size(0)

        train_loss = running_loss / len(train_loader)
        train_accuracy = correct_train / total_train * 100
```

```
################## DenseNet201 ###############

import torch
import torch.nn as nn
import torch.optim as optim
from torchvision import models
from torch.utils.data import DataLoader, random_split

# Load pre-trained DenseNet201 model and modify for 1000 classes
model = models.densenet201(pretrained=True)
model.classifier = nn.Linear(model.classifier.in_features, 411)  # Adjust to 411 in CASIA Lamp or 1000 in CASIA Thousand

# Move model to device (GPU if available)
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = model.to(device)

# Define optimizer and loss function
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.0001)



# Create DataLoaders for training and validation sets
train_loader = DataLoader(train_dataset, batch_size=64, shuffle=True)
val_loader = DataLoader(val_dataset, batch_size=64, shuffle=False)

def train_model(model, train_loader, val_loader, criterion, optimizer, num_epochs):
    best_val_accuracy = 0.0  # Track the best validation accuracy
    best_model_weights = None  # Store the weights of the best model

    for epoch in range(num_epochs):
        # Training phase
        model.train()
        running_loss, correct_train, total_train = 0.0, 0, 0
        for images, labels in train_loader:
            images, labels = images.to(device), labels.to(device)
            optimizer.zero_grad()
            outputs = model(images)
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()

            running_loss += loss.item()
            _, predicted = torch.max(outputs, 1)
            correct_train += (predicted == labels).sum().item()
            total_train += labels.size(0)

        train_loss = running_loss / len(train_loader)
        train_accuracy = correct_train / total_train * 100
```

```python
################ Inception V3 ################
import torch
import torch.nn as nn
import torch.optim as optim
from torchvision import models, transforms
from torch.utils.data import random_split, DataLoader

# Load pre-trained Inception v3 model and modify for 411 classes
model = models.inception_v3(pretrained=True)
model.aux_logits = False  # Disable auxiliary outputs during inference
model.fc = nn.Linear(model.fc.in_features, 411)  # Adjust to 411 classes

# Move model to device (GPU if available)
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = model.to(device)

# Define optimizer and loss function
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.0001)

# Create DataLoaders for training and validation sets
train_loader = DataLoader(train_dataset, batch_size=64, shuffle=True)
val_loader = DataLoader(val_dataset, batch_size=64, shuffle=False)

def train_model(model, train_loader, val_loader, criterion, optimizer, num_epochs):
    best_val_accuracy = 0.0  # Track the best validation accuracy
    best_model_weights = None  # Store the weights of the best model

    for epoch in range(num_epochs):
        # Training phase
        model.train()
        running_loss, correct_train, total_train = 0.0, 0, 0
        for images, labels in train_loader:
            images, labels = images.to(device), labels.to(device)
            optimizer.zero_grad()


            outputs = model(images)
            if isinstance(outputs, tuple):
                outputs = outputs[0]  # Get the main output

            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()

            running_loss += loss.item()
            _, predicted = torch.max(outputs, 1)
            correct_train += (predicted == labels).sum().item()
            total_train += labels.size(0)

        train_loss = running_loss / len(train_loader)
        train_accuracy = correct_train / total_train * 100
```