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

Département d'Automatique, Télécommunications et d'Electronique

Projet de Fin d'Etudes

Pour l'obtention du diplôme de Master

Filière : Télécommunications

Spécialité : Réseaux et Télécommunications

Thème

Détection et classification des iris par des réseaux de neurones convolutifs

Préparé par :

- MESBAH A/Ouahab

Dirigé par :

M. DIBOUNE Abdelhani

Examiné par :

Mme OUALI Kahina

Mme MOKRANI Souad

Année universitaire : **2022/2023**

REMERCIMENTS

En premier lieu, je tiens à remercier mon encadrant, M. **DIBOUNE**, sans ses éclaircissements, conseils et remarques, ce travail n'aurait jamais vu le jour.

Je souhaite également exprimer ma gratitude envers les **membres du jury** pour avoir accepté d’examiner ce modeste travail.

Enfin, je tiens à remercier ma famille et mes amies qui m’ont apporté leur soutien tout au long de la réalisation de ce travail.

DEDICACE

Je dédie ce mémoire à mes parents et à mes proches, eux qui ont toujours été une source de soutien et d'encouragement tout au long de ce parcours.

A mes amies, eux qui m'ont soutenu et motivé pendant la réalisation de ce mémoire.

Et a toutes les personnes qui ont apporté leur aide, que ce soit de près ou de loin, car leur contribution été précieuse.

Chapitre II : Deep learing (apprentissage profond).

Chapitre III : Système de détection et de classification des iris.

Liste des abréviations

ADN	Deoxyribonucleic acid
AlGaAs	Aluminium gallium arsenide
CNN	Convultional Neural Network
DNN	Deep neural network
KNN	K Nearest Neighbords
LSTM	Long short-term memory
ReLu	Rectified Linear Unit
RNN	Recurrent neural network
SGD	Stochastic gradient descent
SVM	Support Vector Machine

Liste des figures

Liste des tableaux

INTRODUCTION GENERALE

La sécurité des systèmes informatiques est une préoccupation majeure dans le monde numérique d'aujourd'hui. Les méthodes traditionnelles de protection, telles que les mots de passe, peuvent être vulnérables aux attaques et aux failles de sécurité. Dans ce contexte, la biométrie émerge comme une solution prometteuse pour renforcer la sécurité des systèmes. En utilisant des caractéristiques physiologiques ou comportementales uniques, telles que l'empreinte digitale, la reconnaissance faciale ou la reconnaissance de l'iris, la biométrie offre une authentification forte et fiable, réduisant ainsi les risques liés aux identifications frauduleuses.

La biométrie de l'iris est une méthode d'identification basée sur les caractéristiques uniques de l'iris humain. L'iris, avec ses motifs complexes et sa stabilité dans le temps, offre une précision et une fiabilité élevées pour l'authentification des individus. En capturant des images de l'iris et en analysant ses caractéristiques distinctives, la biométrie de l'iris permet une reconnaissance précise et rapide. Cette technologie trouve des applications dans des domaines tels que la sécurité des systèmes informatiques, le contrôle d'accès et la gestion des identités.

Le premier chapitre se concentre sur la définition des systèmes et méthodes de reconnaissance biométrique, en mettant l'accent sur la biométrie par Iris. Il explique également les méthodes de segmentation et de prétraitement utilisées sur les images de l'iris pour créer une base de données normalisée. Ainsi que les algorithmes d’apprentissage supervisé de manière générale.

Le 2eme chapitre quant à lui sera consacré à l’apprentissage profond et au réseau de neurones profond, ou on verra comment se fait l’apprentissage de ses derniers ainsi que leur fonctionnement, tout en apportant des détails sur un type de réseau de neurones qui nous intéresse particulièrement qui est le réseau de neurones convolutifs.

Le 3eme est dernier chapitre sera consacré à la partie expérimentale, ou nous allons évaluer les performances d’un modèle CNN en le combinant avec différents classifieurs afin de voir lequel est le plus performant tout en apportant une analyse selon les résultats obtenus.

CHAPITRE I

GENERALITES SUR LA BIOMETERIE DE L'IRIS

I.1 Introduction

L'identification par la biométrie permet de reconnaître ou de vérifier l'identité des personnes avec un haut degré de fiabilité. Actuellement, l'utilisation des systèmes biométriques tels que la reconnaissance par l'iris, le visage ou par les empreintes digitales sont indispensables dans les environnements de haute sécurité.

La technologie de la reconnaissance par l'iris est une des plus fiable avec un taux d'erreur très faible ($1/10^{72}$ de trouver deux iris identiques selon Daugman [1]) et cet iris reste inchangé jusqu'à la mort de l'individus, ce qui rend cette technologie quasiment impossible à frauder. C'est le seul système pouvant être utilisé sur un grand nombre de personnes en identification complète [1].

Dans ce chapitre nous allons nous consacrer à la présentation de la biométrie de l'iris, on verra une description du système de reconnaissance d'iris en montrant l'acquisition et les différentes méthodes utilisées pour le traitement des données de la biométrie de l'iris.

I.2 La biométrie

La biométrie est une discipline déjà connue dans les temps passer. Elle désigne la reconnaissance et l'identification d'individus. En effet, elle utilise des caractéristiques physiques ou comportementales uniques d'une personne pour l'identifier de manière précise [2][3].

L'utilisation de la biométrie est dû à sa fiabilité, d'habitude on utilise des codes, mots de passe,badges, cartes à puce etc. pour identifier une personne mais cela présente quelques inconvénients tels que : il faut mémoriser ses derniers, mais aussi un risque d'utilisation par une personne non autorisée. C'est là que la biométrie a tout son intérêt, chaque individu a ses propres caractéristiques physiques qui ne peuvent être changées, perdues ou volées [4].

I.3 Les domaines d'application de la biométrie

De nos jours les techniques biométriques sont appliquées dans quasiment tous les domaines du au besoin et à la nécessité de connaître l'identité des personnes pour avoir accès à des données, services, etc. Si dessous ont peux voir ses domaines d'application :

I.3.1 Applications commerciales

Telles que l'ouverture d'un réseau informatique, la sécurité des données électroniques, l'e-commerce, l'accès Internet, les cartes de crédit, etc. [5].

I.3.2 Applications administratives

Telles que la carte d'identité nationale, le permis de conduire, la sécurité sociale, le contrôle des frontières, le contrôle des passeports, etc. [5].

I.3.3 Applications légales

Telles que l'identification de corps, la recherche criminelle, l'identification de terroristes, etc. [5].

I.4 Le système biométrique

Un système biométrique est un système de reconnaissance qui utilise les caractéristiques physiques, comportementales ou anatomiques uniques pour reconnaitre un individu. Une fois les données biométriques acquissent elles sont ensuite traitées, analysées et comparées avec une base de données contenant des données biométriques déjà stockées afin d'établir une correspondance.

I.5 Les modalités biométriques

La biométrie est basée sur les caractéristiques biométriques de l'individu, ces caractéristiques sont classées en trois grandes catégories :

I.5.1 La biométrie physique

I.5.1.1 Le visage

La reconnaissance faciale permet d'identifier et d'authentifier une personne très rapidement à partir des caractéristiques de leur visage telles que l'écartement des yeux, les arêtes du nez, les commissures des lèvres, les oreilles, menton, etc. cette technique est particulièrement efficace pour identifier une personne à partir d'une image d'un groupe d'individus, dans un lieu ou une base de données [6] [7].

I.5.1.2 L'empreinte digitale

Une empreinte digitale est constituée d'un ensemble de lignes localement parallèles formant un motif unique pour chaque individu. On distingue les stries et les sillons. De plus chaque empreinte possède un ensemble de points singuliers globaux et locaux. C'est une des techniques les plus utilisées car elle est facilement acceptée par la communauté et aussi une des moins couteuses [8].

I.5.1.3 La géométrie de la main

Cette forme de biométrie implique l'examen de 90 traits de la main, dont la longueur et la largeur des doigts ainsi que la paume, la forme des articulations ou encore le dessin des lignes de la main. Elle est habituellement employée pour le contrôle d'accès physique, ainsi que pour le pointage horaire [9].

I.5.1.4 L'iris :

Pour la technologie de la biométrie de l'iris, il est d'abord nécessaire de localiser l'iris. Ensuite, cet iris sera éclairé avec une lumière infrarouge pour réussir à la capturer. Cette méthode permet de recueillir de nombreuses caractéristiques de texture qui sont uniques à chaque individu [10] [11].

I.5.2 Biométrie comportementale

I.5.2.1 La voix

La biométrie de la voix utilise des données provenant à la fois de caractéristiques physiologiques telles que l'âge, le sexe, la tonalité et l'accent, ainsi que de facteurs comportementaux tels que la vitesse et le rythme. Néanmoins, cette technique est très facilement falsifiable [12].

I.5.2.2 La signature manuscrite

C'est une écriture personnelle d'un individu, la vérification de la signature est basée sur deux modes :

- **Mode statique :** la vérification de la signature statique se concentre sur les aspects géométriques de la signature. Généralement, dans ce mode, la signature est normalisée à une taille connue ensuite décomposée en élément simple.
- **Mode dynamique :** il exploite les propriétés dynamiques telles que l'accélération, la vitesse et les profils de trajectoire de la signature [13].

I.5.2.3 La démarche

Chaque personne a une manière particulière de marcher. Du mouvement des jambes, des bras au mouvement des articulations. Tous ses détails permettent d'identifier une personne. Malgré qu'elle soit une technique d'identification à distance, cependant elle est assez vulnérable [14].

I.5.2.4 La Frappe dynamique sur le clavier

De la pression mise sur les touches en passant par les fautes régulières et la vitesse de frappe, chacun a une méthode supposée unique de taper au clavier. Un système basé sur cette dynamique ne nécessite aucun équipement particulier, seulement un ordinateur disposant d'un clavier [15].

I.5.3 La biométrie biologique

I.5.3.1 Le réseau veineux

Les veines de la main sont des réseaux qui varient d'une personne à une autre. L'analyse de cette différence permet de maintenir des points pour différencier une personne d'une autre [16].

I.5.3.2 L'ADN

L'analyse des empreintes génétiques est une méthode extrêmement précise pour déterminer l'identité de la personne. Il est impossible de trouver deux personnes qui ont le même ADN. Cette modalité possède l'avantage d'être unique et permanente durant toute la durée de vie [16].

I.6 Les facteurs à prendre en considération lors du choix d'une caractéristiquebiométrique

On distingue sept particularités pour chaque attribut biométrique permettant de les comparer les uns aux autres et ainsi déterminer les plus appropriés selon le besoin de tel ou tel système [17] :

- **L'universalité :**

Vérifie si la modalité existe et si elle est présente de manière universelle chez tous les individus.

- **L'unicité :**

Permettre de différencier un individu par rapport à un autre.

- **La permanence :**

Détermine si la caractéristique reste permanente sur une période spécifique.

- **La mesurabilité :**

Indique le niveau de facilité pour acquérir, mesurer et exploiter cette modalité.

- **La performance :**

Décrit la robustesse, la fiabilité et la rapidité de la mesure.

- **L'acceptabilité :**

Collecter les caractéristiques d'un individu avec son accord.

- **La vulnérabilité :**

Représente la résistance du système au tentatives de contournement.

Modalité	Universalité	Unicité	Permanence	Mesurabilité	Performance	Acceptabilité	Vulnérabilité
Voix	Moyenne	Faible	Faible	Moyenne	Faible	Haute	Haute
ADN	Haute	Haute	Haute	Faible	Haute	Faible	Faible
Visage	Haute	Faible	Moyenne	Haute	Faible	Haute	Haute
Empreinte	Moyenne	Haute	Haute	Moyenne	Haute	Moyenne	Moyenne
Démarche	Moyenne	Faible	Faible	Haute	Faible	Haute	Moyenne
Géométrie de la main	Moyenne	Moyenne	Moyenne	Haute	Moyenne	Moyenne	Moyenne
Réseau veineux de la main	Moyenne	Moyenne	Moyenne	Moyenne	Moyenne	Moyenne	Faible
Iris	Haute	Haute	Haute	Moyenne	Haute	Faible	Faible
Signature	Faible	Faible	Faible	Haute	Faible	Haute	Haute

Tableau I.1 : Comparaison des modalités biométriques [18].

I.7 L'identification par l'iris

L'identification par l'iris utilise plus de paramètres que les autres méthodes d'identification et la fiabilité résultante est suffisante pour ne plus faire de l'identification mais de l'authentification.

La première étape est assez délicate, elle consiste à capturer l'image de l'iris. En effet, l'œil est un organe extrêmement sensible, et le fait qu'il soit obscurci par les cils, les paupières. Sans oublier les mouvements involontaires de la personne rend cette étape assez complexe. De plus il est nécessaire que le système de capture d'image soit à la fois rapide et précis tout en évitant l'utilisation d'une lumière qui pourrait créer des reflets sur l'œil [19].

I.8 Principe général de reconnaissance de l'iris

Le processus du système de reconnaissance de l'iris contient deux phases principales :

- Une unité de vision optique pour capturer l'image de l'iris.
- Une unité de traitement des données qui implique : segmentation de l'image de l'iris, normalisation, extraction des caractéristiques de l'iris et correspondance du modèle.

I.9 L'acquisition de l'iris

L'acquisition d'une image d'iris est considéré comme l'une des plus difficiles en biométrie. En effet, l'iris est un objet de petite taille, sombre, localisée derrière la cornée. Ces facteurs rendent son acquisition particulièrement compliquée. Tout d'abord, en raison de sa nature sombre, il est nécessaire d'éclairer l'iris, mais en même temps l'iris est très sensible à la lumière.

Ensuite, du a sa petite taille il est essentiel d'utiliser des objectifs puissants ou de rapprocher l'iris de l'objectif, mais cela doit être fait avec prudence. Enfin, la surface de la cornée est une surface réfléchissante. Cette dernière caractéristique fait que sans l'utilisation de techniques spécifiques, l'image de l'iris capturée sera souvent recouverte de reflets provenant de toutes sources lumineuses présentes dans l'environnement d'acquisition [20].

Pour ce faire, l'acquisition d'une image d'iris s'effectue avec une caméra monochrome dans le domaine du proche infrarouge (longueur d'ondes entre 650 nm et 900 nm) en utilisant une illumination artificielle réalisée par une diode AlGaAs (Aluminium Gallium Arsenic) placée à proximité de l'objectif [1].

I.10 Système de reconnaissance de l'iris

Après avoir acquis une image de l'iris, on doit passer au traitement des données obtenues par l'acquisition, ce système est composé de plusieurs phases comprenant :

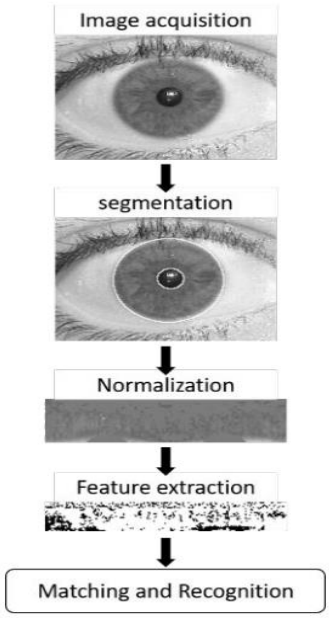


Figure I.1 : Étapes typiques de la reconnaissance de l'iris [21].

I.10.1 Segmentations de l'iris

Pour reconnaitre l'identité d'un iris, une image de l'œil est premièrement acquise. La région de l'iris doit être isolée afin d'extraire et de comparer les caractéristiques discriminantes de chaque iris avec ceux des iris enregistrés dans la base de données. La segmentation de la région de l'iris est considéré comme l'étape cruciale du système de reconnaissance de l'iris, car elle détermine directement l'efficacité de l'identification des attributs uniques de l'iris. En effet La segmentation implique de déterminer les frontières interne et externe de l'iris, d'isoler les paupières et les cils s'ils recouvrent l'iris, et enfin d'éliminer toute réflexion de lumière si présente [22].

I.10.2 Quelques méthodes de segmentation :

- **La transformée de Hough :** la transformée de Hough est un algorithme présenté par Paul Hough en 1962 pour la détection de caractéristiques d'une forme spécifique, comme des lignes ou des cercles, dans des images. Contrairement à d'autres techniques, la transformée de Hough est capable de trouver des formes même si elles

sont partiellement interrompues ou si l'image est bruitée. C'est pourquoi elle est souvent utilisée dans la reconnaissance de l'iris, car elle permet de détecter les contours de l'iris même dans des conditions difficiles [54].

- **La méthode de l'opérateur intégral-différentiel :** introduite par Daugman, cette méthode permet de segmenter l'iris en identifiant à la fois les limites internes (limbe) et externes (pupille) de la région de l'iris. L'algorithme recherche les paramètres essentiels tels que le centre et le rayon des limites circulaires en utilisant un espace paramétrique tridimensionnel. L'objectif est de maximiser les fonctions d'évaluation du modèle. Cet algorithme atteint de bonnes performances en reconnaissance de l'iris, mais il présente l'inconvénient d'une charge de calcul importante [54].
- **Algorithme de clustering flou :** Une nouvelle approche de segmentation de l'iris a été développée pour traiter des images variées et bruitées. L'algorithme utilise une méthode de clustering floue pour regrouper les pixels de l'image en fonction de leurs caractéristiques. Cela permet d'obtenir une segmentation plus précise pour la reconnaissance de l'iris, même dans des conditions difficiles. Cependant, cette méthode nécessite une recherche minutieuse pour identifier les paramètres du cercle pour les limites de la pupille et de l'iris [54].

I.10.3 La normalisation

La taille de l'iris peut changer en raison de la variation d'éclairage, même pour un iris provenant de la même personne. Afin d'obtenir des résultats de reconnaissance plus précis, il est nécessaire de corriger cette déformation élastique de la texture de l'iris. La normalisation est une technique utilisée pour préparer un iris segmenté en vue de l'extraction de ses caractéristiques. Elle permet de transformer la région circulaire de l'iris segmentée en une forme rectangulaire de taille fixe [21].

I.10.3.1 Méthode de normalisation :

- **Daugman's rubber sheet model :** Le modèle de "rubber-sheet" de Daugman est la méthode la plus utilisée pour la normalisation de l'iris. Cette méthode transforme la région circulaire de l'iris en un bloc rectangulaire de taille fixe. En utilisant ce modèle, chaque pixel de l'iris circulaire est transformé en une position équivalente sur les axes polaires (r, θ), où r est la distance radiale et θ est l'angle de rotation au rayon correspondant. La résolution radiale décrit le nombre de lignes radiales générées autour de la région de l'iris, tandis que la résolution angulaire est le nombre de points de données dans la direction radiale. La formule $I[x(r,\theta),y(r,\theta)] \rightarrow I(r,\theta)$ indique comment chaque pixel de l'image originale est transformé en un nouveau pixel correspondant dans la représentation normalisée de l'iris [55].

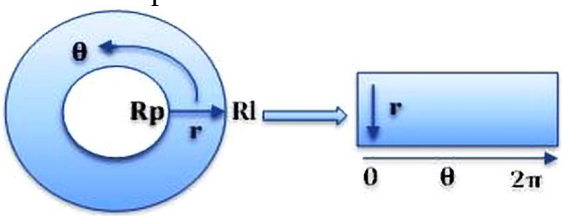


Figure I.2: Daugman's rubber sheet model [55].

I.10.4 Les difficultés de la reconnaissance par l'iris

Dans la biométrie de l'iris, on trouve beaucoup d'obstacles à surmonter afin d'arriver à définir un système de reconnaissance fiable. Des bruits occultes peuvent rendre la tâche plus complexe :

- Un des principaux facteurs à prendre en compte est la présence des paupières, car leur ouverture peut varier en fonction de l'œil, ce qui peut potentiellement masquer la texture de l'iris.
- Ces bruits peuvent également être causées par les cils en raison de leur forme irrégulière et de leur position aléatoire. De plus, le port de lentilles de contact peut rendre la reconnaissance de la texture de l'iris difficile, voire fausser complètement l'identification.
- En plus des bruits mentionnés précédemment, l'image peut également être affectée par divers facteurs indésirables. L'acquisition de l'image se fait à une certaine distance, ce qui nécessite l'utilisation d'une focale suffisamment puissante. Par conséquent, il est important de maintenir une distance appropriée lors de la capture de l'image, sinon les détails de la texture de l'iris risquent de devenir flous.

Un autre type de flou est causé par les mouvements brusques de l'œil. L'ouverture et la fermeture des paupières, ainsi que la dilatation et la contraction des pupilles. Par conséquent, lorsque de tels mouvements se produisent, les détails de la texture de l'iris seront considérablement réduits, ce qui rendra la reconnaissance particulièrement difficile [20].

Figure I.3 : le bruit de prise partielle d'image de l'iris [23].

Figure I.4 : exemple de rotation de l'iris [23].

Figure I.5 : le bruit de réflexion d'éclairage [23].

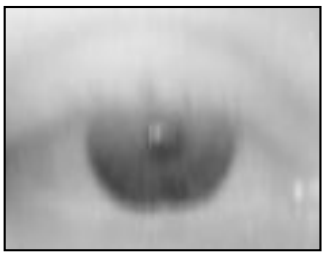


Figure I.6 : exemple d'image floue [23].

I.10.5 **Extraction des caractéristiques de l'iris**

Une fois que l'iris soit correctement segmenté et normalisé, la prochaine étape consiste à effectuer l'encodage. L'encodage implique l'extraction des caractéristiques les plus distinctives et pertinentes de l'iris, celles qui sont nécessaires et utiles pour son identification ou son authentification [23].

I.10.6 **Correspondance du modèle**

Avant de pouvoir utiliser le système de reconnaissance d'iris pour révéler l'identité d'une personne, il est nécessaire de procéder à une phase d'entraînement. Au cours de cette phase, une base de données est créée, contenant les signatures spécifiques de l'iris de chaque individu à reconnaître.

Le processus consiste à assigner à chaque personne un profil d'iris, qui sera ensuite utilisé pour comparer avec un nouveau profil d'iris d'une identité inconnue [3].

La correspondance des modèles est établie en utilisant différentes techniques. Un système de reconnaissance d'iris peut fonctionner selon deux modes :

- **Le mode authentification :**

Il comparera les informations des utilisateurs autorisés, qui sont stockées dans une base de données, avec les informations fournies. L'accès sera autorisé uniquement si les informations correspondent parfaitement [23].

- **Le mode identification :**

Permet de connaître l'identité d'une entité. Le vérificateur compare l'information fournie par cette entité avec celle de toutes les entités connues. L'objectif de l'identification est d'obtenir une correspondance unique, c'est-à-dire une seule et unique entité associée à l'information fournie [23].

I.11 Technique de reconnaissance de l'iris par apprentissages supervisé

Ces méthodes utilisent des ensembles de données étiquetés afin de développer des algorithmes capables de classifier des données ou de prédire des résultats avec précision. Au cours de l'apprentissage, le modèle ajuste ses paramètres en fonction des données d'entrée jusqu'à ce qu'il soit optimal [24].

I.11.1 Algorithmes d'apprentissage supervisé

Il existe plusieurs types d'algorithmes d'apprentissage supervisé tels que :

I.11.1.1 Machine a vecteur de support (SVM, Support Vector Machine)

Le SVM permet de résoudre les problèmes de classification ou de régression. Le principe de base consiste à trouver une fonction h qui associe à chaque vecteur d'entrée x (un échantillon, un vecteur décrivant les valeurs des caractéristiques de l'iris) une sortie $y \in \{-1, +1\}$ indiquant si l'iris appartient ou non à une personne donnée. Dans le cas linéaire on utilise la fonction h(x) = $W^T\ x + W_0$ l'objectif est d'apprendre les meilleurs poids W et le biais W_0 pour obtenir le meilleur séparateur possible. Cela se fait en utilisant les échantillons étiquetés de l'ensemble d'apprentissage :

$\{(x_1, y_1)\}, (x_2, y_2), \dots, (x_k, y_k)\} \subset R^n \times \{-1, +1\}$. $y_i\ (W^T x_i + W_0) \geq 0$. Sachant que la marge est la distance entre l'hyperplan $x \mapsto h(x) = W^T x + W_0$ et les vecteurs du support (les échantillons les plus proches, voir Figure I.7). Le but principal est de chercher l'hyperplan qui permet de maximiser cette marge [23].

$$argmax_{w,w_0} \min\{\| x - x_k \| : x \in R\ , w\ \ x + w_0 = 0\}.$$

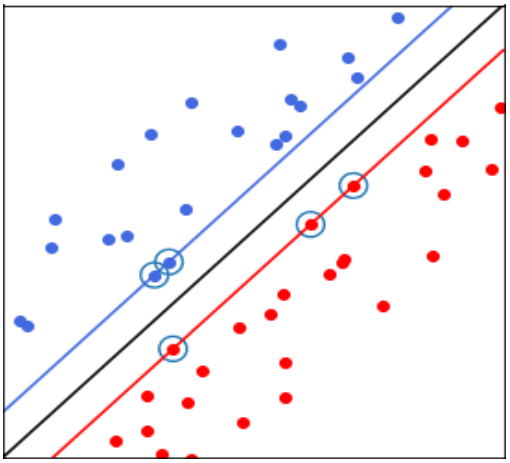


Figure I.7 : SVM [23].

I.11.1.2 Le modèle k plus proches voisins (KNN, k-nearest neighbors)

La méthode des k plus proches voisins (k-NN) est un algorithme qui part du principe que des points de données similaires peuvent être trouvés à proximité les uns des autres. Par conséquent, il cherche à calculer la distance entre les points de données, généralement par la distance euclidienne, puis il attribue une catégorie basée sur la catégorie ou la moyenne la plus fréquente [24].

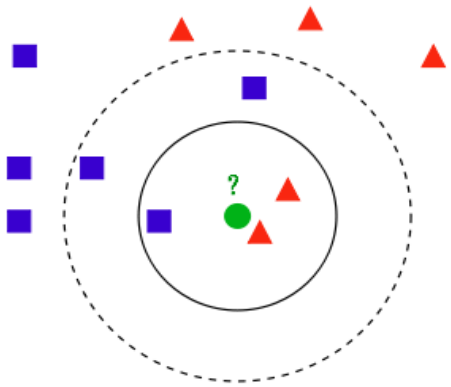


Figure I.8 : KNN [27].

Si le nombre de plus proches voisins, k, est fixé à 3, la classe du point est celle des triangles, car ces derniers sont au nombre de 2 contre un seul carré. Si k vaut 5, la classe du point est celle des carrés, au nombre de 3 contre 2 triangles [27].

I.11.1.3 Le réseau de neurones

Un neurone est une représentation abstraite d'un neurone biologique, utilisée en mathématiques et en informatique. Il calcule une valeur de sortie en fonction de ses entrées, en utilisant des coefficients ou des poids qui déterminent l'influence des connexions synaptiques. Ces coefficients sont ajustés pendant une phase d'apprentissage afin d'optimiser la capacité du neurone à représenter la fonction désirée [35].

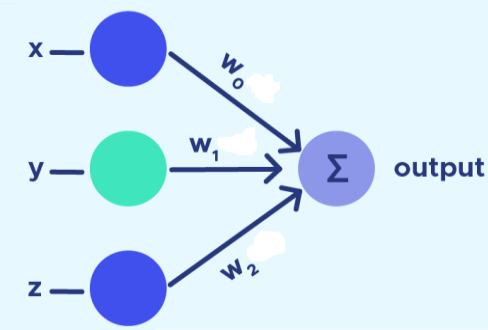


Figure I.9 : Schéma a un seul neurone [46].

Les réseaux de neurones sont constitués de plusieurs couches, notamment une couche d'entrée, une ou plusieurs couches cachées et une couche de sortie. Chacune de ces couches est composée de nœuds ou de neurones artificiels. Chaque neurone est connecté à d'autres neurones par des connexions ayant des poids et des seuils associés.

Son fonctionnement est simple, une fois qu'une couche d'entrée est déterminée, des poids sont attribuées. Ces poids permettent de déterminer l'importance d'une variable donnée, les plus

grandes contribuant de manière plus significative à la sortie par rapport aux autres entrées. Les valeurs d'entrée sont ensuite multipliées par leurs poids respectifs, puis sommées. Ensuite, la somme passe par une fonction d'activation qui détermine la sortie finale [28].

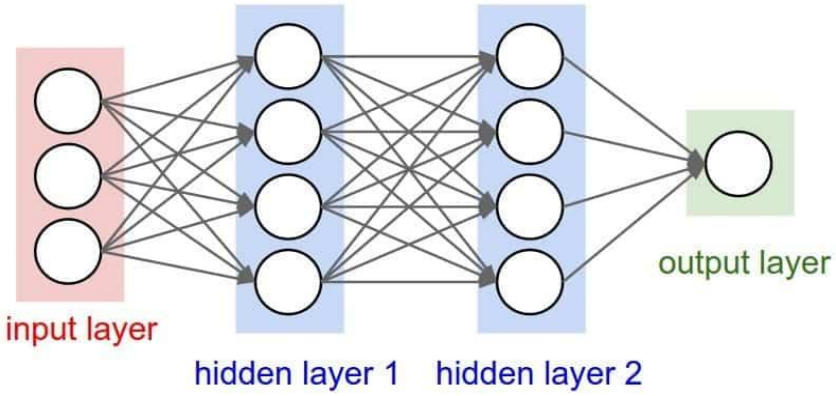


Figure I.10 : Réseau de neurones artificielles [48].

Dans le cadre de ce mémoire on s'intéresse à l'apprentissage profond.

I.12 Conclusion

En conclusion, ce chapitre met en évidence l'utilisation prometteuse de la biométrie et des techniques d'apprentissage supervisé dans le domaine de la reconnaissance et de l'authentification des individus. L'utilisation de ces approches permet d'améliorer les performances de reconnaissance en exploitant les caractéristiques uniques des individus. Cependant, des recherches supplémentaires sont nécessaires afin exploiter pleinement le potentiel de la biométrie et de l'apprentissage supervisé.

CHAPITRE II

DEEP LEARNING
(APPRENTISSAGE PROFOND)

II.1 Introduction

L'apprentissage profond est un sous-domaine de l'apprentissage automatique, lui-même faisant partie de la grande famille de l'intelligence artificielle. Il correspond à toutes les techniques de réseaux de neurones artificiels [29].

Depuis les années 90, la plupart des chercheurs avaient abandonné l'idée de l'apprentissage profond car à l'époque la formation d'un réseau neuronal profond était largement considérée comme impossible mais en 2006, Geoffrey Hinton et al. ont publié un article qui a ravivé l'intérêt de la communauté scientifique et, très vite, de nombreux nouveaux articles ont démontré que l'apprentissage profond était non seulement possible, mais qu'il permettait d'obtenir des résultats époustouflants qu'aucune autre technique d'apprentissage automatique ne pouvait espérer égaler. La force de l'apprentissage profond réside dans le fait que la machine peut extraire des caractéristiques et apprendre toute seule, indépendamment de l'intervention d'un expert, et il a été appliqué dans de nombreux domaines différents [30].

II.2 Réseaux de neurones profonds (DNN)

II.2.1 Architecture générale d'un réseau de neurone profond

Les réseaux de neurones profonds comprennent plusieurs couches cachées (N couches) dont le terme profond. La plupart des architectures profondes sont réalisées en combinant et recombinant un ensemble limité de primitives architecturales (des couches de réseaux neuronaux). Ci-dessous un bref aperçu de la structure commune d'un réseau profond [31] [32].

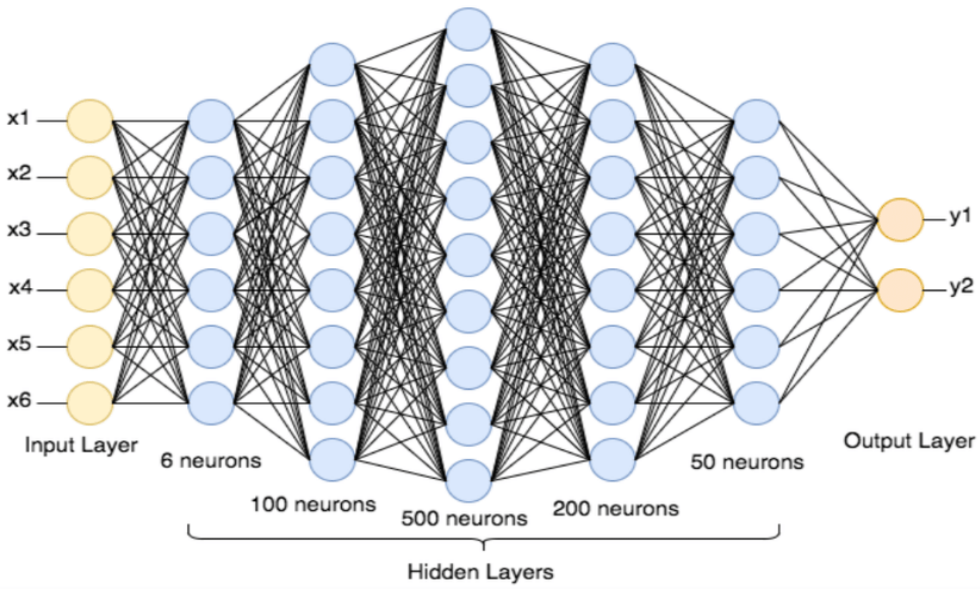


Figure II.1: architecture générale d'un DNN [49].

II.3 Apprentissage des réseaux de neurones profond

L'apprentissage des réseaux de neurones profond se fait couramment par apprentissage supervisé ou on utilise un jeu d'entraînement (aussi appelé d'apprentissage) pour apprendre aux modèles à générer les résultats souhaités. Ce jeu de données comprend des entrées et des sorties correctes, permettant ainsi aux modèles d'apprendre progressivement.

L'apprentissage supervisé peut être divisé en deux types, à savoir la classification et la régression [24] :

- La classification est une technique qui utilise des algorithmes pour classifier avec précision des données de test en différentes catégories. Elle identifie des éléments spécifiques dans l'ensemble de données et essaie de déduire comment ces éléments doivent être étiquetés ou définis.
- La régression est une méthode permettant de comprendre la relation entre les variables indépendantes et dépendantes. Elle est souvent utilisée pour effectuer des projections.

II.3.1 Test et validation du modèle

Pour éviter le surajustement aux données d'entraînement on utilise un autre ensemble de données appelé ensemble de test. Afin de faire un usage optimal des données disponibles tout en évitant d'avoir plusieurs ensembles de données indépendantes [25].

Un hyperparamètre est un paramètre dont la valeur est définie avant le début du processus d'apprentissage. Lorsque les hyperparamètres sont mal réglés le modèle peut souffrir de surajustement (overfitting) ou de sous-ajustement (underfitting), du coup ces derniers doivent souvent être réajusté pour réaliser le plein potentiel d'une méthode. C'est là l'intérêt de l'étape de validation. On évalue le modèle sur des données indépendantes (qui n'ont pas servi à l'entrainement) puis selon le résultat obtenu on réajuste ces hyperparamètres jusqu'à trouver les hyperparamètres optimaux qui permettent de maximiser la performance du modèle. On peut citer quelques méthodes de validations tell que la validation croisée k-fold, la validation croisée par permutation et la validation croisée leave-one-out [26].

II.4 Fonctionnement d'un réseau de neurones profond

Pour comprendre le fonctionnement d'un réseau de neurones profond, il est important de comprendre la 1ère étape qui est le forwarding, également connu sous le nom de propagation avant.

II.4.1 Forwarding

Pour générer une sortie, les données d'entrée doivent être acheminées vers l'avant. Les données ne doivent pas circuler dans le sens inverse pendant la génération de la sortie, sinon elles formeraient un cycle et la sortie ne pourrait jamais être générée.

À chaque neurone d'une couche cachée ou d'une couche de sortie, le traitement se fait en deux étapes :

- **La pré activation :** il s'agit d'une somme pondérée des entrées, c'est-à-dire la transformation linéaire des poids (W) en fonction des entrées disponibles. Sur la base de cette somme agrégée et de la fonction d'activation, le neurone décide de transmettre ou non cette information.
- **L'activation :** la somme pondérée calculée des entrées est transmise à la fonction d'activation. Une fonction d'activation est une fonction mathématique qui ajoute une non-linéarité au réseau. Il existe trois fonctions d'activation couramment utilisées sigmoid, ReLU, softmax [33].

II.4.1.1 La fonction d'activation

Une fois que les données sont propagées à travers le réseau de neurones dans la phase de forwarding, chaque neurone utilise une fonction d'activation pour introduire de la non-linéarité.

Son objectif est de calculer la somme pondérée des entrées et d'ajoute le biais. C'est une transformation non linéaire de la valeur d'entrée [32].

II.4.1.1.1 La fonction sigmoid

$$\sigma = \frac{1}{1+e^{-z}}$$

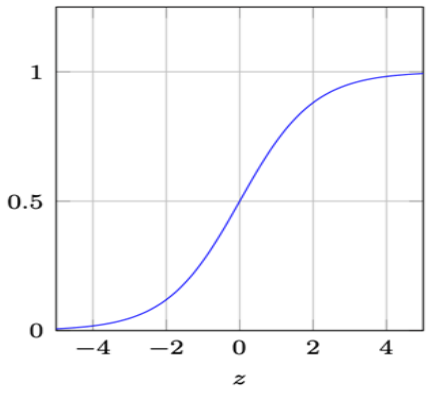


Figure II.2 : fonction sigmoid [57].

II.4.1.1.2 La fonction ReLu

$$R(z) = \max(0,z)$$

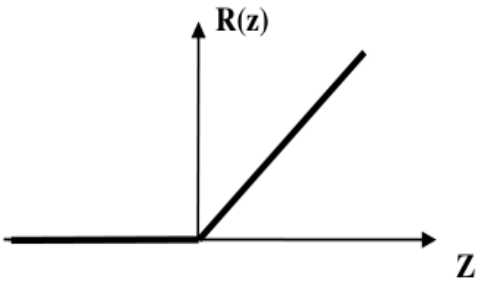


Figure II.3 : fonction ReLu [23].

II.4.1.1.3 La fonction softmax

$$\mathbf{softmax}\ (\mathbf{Z}) = \frac{e^{Zj}}{\sum_{i=1} e^{Zj}}$$

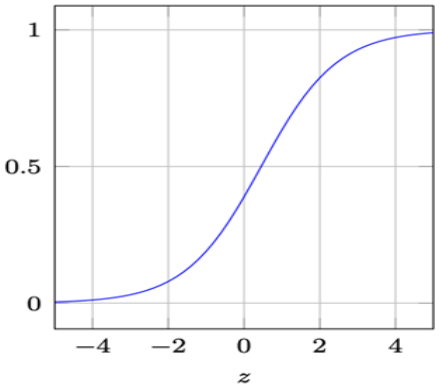


Figure II.4 : fonction softmax [57].

II.4.1.2 La fonction de cout

Une fois que les sorties des neurones ont été activées, il devient nécessaire d'évaluer la performance du modèle en utilisant une fonction de coût. Cette fonction mesure l'écart entre les prédictions du réseau neuronal et les valeurs réelles. Elle est fréquemment définie par l'erreur quadratique moyenne et se traduit par la moyenne des erreurs m, le nombre d'instances dans l'ensemble de données d'entrainement [34].

$$E = \frac{1}{2m} \sum_{i=1}^{m} \left(y_{gnd}^{(i)} - y_{tst}^{(i)} \right)^2 \qquad \textbf{(1)}$$

II.4.2 Optimisation de la fonction des couts :

L'optimisation de la fonction des couts a pour objectif de réduire l'écart entre les prédictions du réseau neuronal et les valeurs réelles, c.-à-d. de trouver les valeurs des paramètres du modèle qui minimisent cette fonction, ce qui conduit à de meilleures performances et à des prédictions plus précises.

Il existe différentes méthodes d'optimisation et l'une des méthodes les plus utiliser est le gradient descent car elle offre un bon équilibre entre efficacité et performance.

II.4.2.1 Le gradient decent

Le gradient descent est un algorithme d'optimisation utilisé dans le domaine de l'apprentissage automatique, il définit le processus d'apprentissage qui optimise les poids synaptiques et réduit la fonction de cout **(1)**. Comme on le vois dans Figure II.5, suivant des dérivées partielles qui correspondent à chaque erreur donc, l'ensemble de ces erreurs et par l'intermédiaire du gradient descent, suivant un pas, nous avançons vers le point optimal [34].

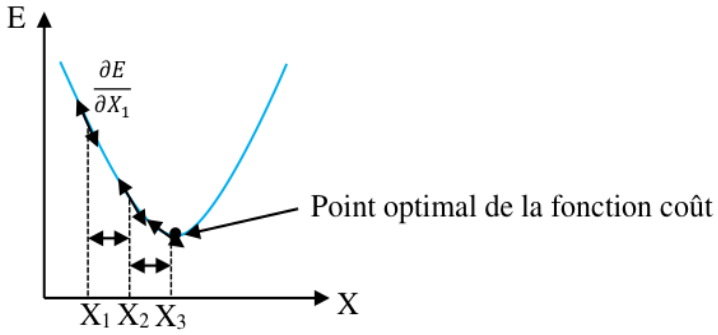


Figure II.5 : Optimisation de la fonction coût par gradient decent [34].

II.4.2.1.1 Le pas d'apprentissage

Le pas d'apprentissage represente la vitesse à laquelle la machine apprend. Cependant, il faut faire attention au sur-apprentissage car cela peut provoquer des dysfonctionnements et conduire à des prédictions erronées [34].

En plus du pas d'apprentissage, le momentum est une technique d'optimisation qui ajoute une composante d'inertie pour accélérer la convergence de gradient descent.

II.4.2.1.2 Le Momentum

Le problème avec le gradient descent est que la mise a jours des poids au moment (t) est déterminé que par le pas d'apprentissage et le gradient à ce moment-là. Ceci amené à des mises a jours des poids faibles ou inexistantes ce qui va stopper le processus d'apprentissage.

Pour résoudre ce problème imaginons qu'une balle roule depuis le point A. La balle commence à descendre et prend de l'élan sur la pente A, B. Lorsque la balle atteint le point B, elle a accumulé suffisamment d'élan pour se propulser à travers la région du plateau B et finalement suivre la pente B, C pour atterrir au niveau des minima globaux C.

C'est ce principe qui permet la mise a jours des poids pour accélérer la convergence et éviter les oscillations indésirables. Le momentum augmente pour les dimensions dont le gradient pointe vers la même direction et réduit les mises à jour dont le gradient change de direction [35] [36].

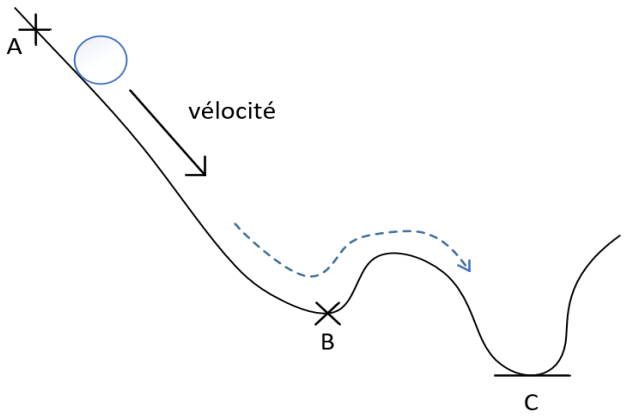


Figure II.6 : principe de momentum.

II.4.3 Les types de gradient descent

Il existe 3 types de gradient descent :

II.4.3.1 Gradient descent par batch

Le gradient descent par batch consiste à accumuler l'erreur pour chaque point d'un ensemble d'entraînement, puis à mettre à jour le modèle uniquement lorsque tous les exemples d'entrainement ont été évalués. Cependant le temps de traitement peut être long pour les grands ensembles de données d'entraînement, car il doit stocker toutes les données en mémoire jusqu'a jusqu'à ce que tous les exemples aient été évalués [43].

II.4.3.2 Stochastic gradient descent (SGD)

Le SGD met à jour les paramètres du modèle pour chaque exemple individuel, ce qui facilite le stockage en mémoire. Bien que les mises à jour fréquentes du SGD offrent plus de détails et de rapidité, elles peuvent être moins efficaces en termes de calcul. Cependant, le SGD converge plus rapidement vers un minimum global et évite les minima locaux, améliorant ainsi l'exploration des paramètres du modèle [43].

II.4.3.3 Gradient descent par mini-batch

Le gradient descent par mini-batch divise l'ensemble des données d'entraînement en mini-batchs et effectue des mises à jour sur chaque lot. Cela permet de trouver un équilibre entre l'efficacité de calcul de gradient descent par batch et la vitesse de stochastic gradient descent. En utilisant des mini-batchs, cette approche bénéficie à la fois d'une utilisation efficace des ressources et de mises à jour plus fréquentes qui peuvent conduire à une convergence plus rapide du modèle [43].

II.4.4 **Back propagation**

Le back propagation consiste à analyser comment la sortie d'un réseau neuronal varie en fonction de ses paramètres (w, b) dans chaque couche du modèle. Pour ce faire, on calcule une chaîne de gradients qui indique comment la fonction varie par rapport à la dernière couche, puis comment cette dernière couche varie par rapport à la couche précédente, et ainsi de suite jusqu'à atteindre la première couche de notre réseau de neurones. Cette méthode permet de quantifier l'impact des paramètres sur la sortie globale du réseau et facilite l'ajustement des poids et des biais pour améliorer les performances du modèle [23].

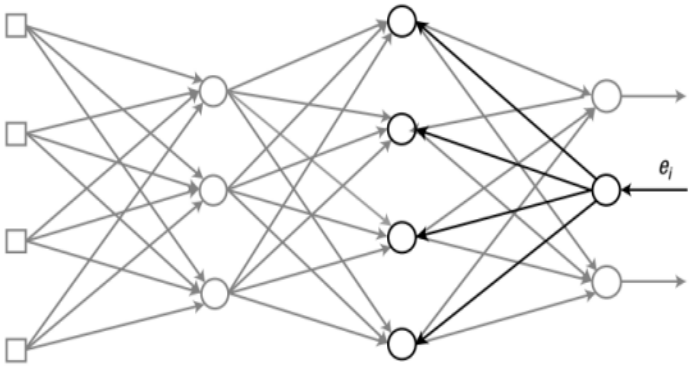


Figure II.7 : principe de back propagation [34].

II.4.5 **Batchs**

Apres avoir utilisé le back propagation nous pouvons procéder à la phase d'optimisation en utilisant des batchs de données d'entrainement. Ces batchs peuvent être divisés en mini-batchs et envoyés au réseau de neurones pour l'entraînement. Le réseau de neurones peut alors effectuer des calculs parallèles sur ces mini-batchs, ce qui peut améliorer l'efficacité de l'apprentissage en accélérant le traitement des données et la mise à jour des poids du modèle [34].

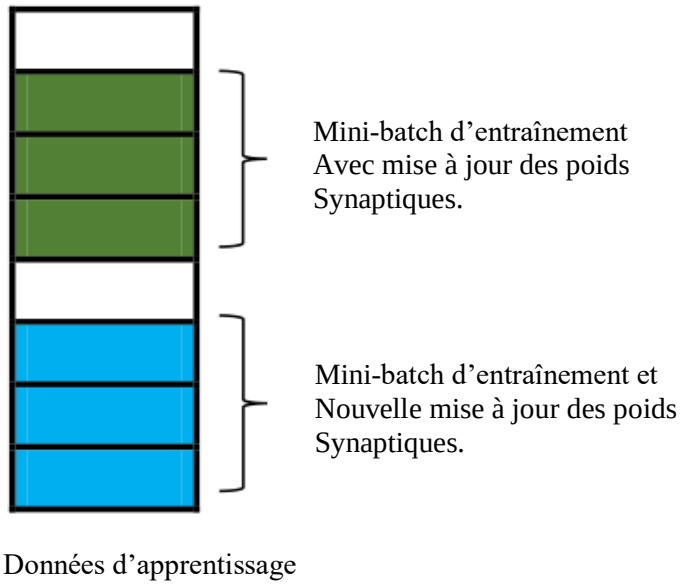


Figure II.8 : exemple de mini-batch de données d'entrainement [34].

II.4.6 **Epochs**

L'entraînement du modèle se produit généralement sur plusieurs epochs pour améliorer progressivement les performances en ajustant les poids en fonction des erreurs de prédiction.

Une epoch peut être définie comme le nombre de fois où un algorithme parcourt l'ensemble des données d'entraînement. Chaque passage correspond à un aller-retour et il permet au modèle d'ajuster ses paramètres. Le nombre d'epochs peut varier, parfois atteignant plusieurs milliers, car la procédure est répétée jusqu'à ce que le taux d'erreurs du modèle soit suffisamment réduit [37].

II.5 **Les types de réseaux de neurones profonds**

Apres avoir vu le processus d'apprentissage des réseaux de neurones profonds ainsi que leur fonctionnement, on passe maintenant à quelques types de réseau de neurones. On peut citer :

II.5.1 **Les autocodeurs**

L'autoencodeur est un type de réseau de neurones profond. Son rôle est d'encoder des données en utilisant une représentation de dimension réduite et de les décoder ensuite pour retrouver l'information d'origine. Ils se composent généralement d'une couche d'entrée (couche L1) qui représente les vecteurs de données ou de caractéristiques, d'une ou plusieurs couches cachées (couche L2) qui effectuent la transformation des caractéristiques, et d'une couche de sortie qui cherche à reproduire les données d'entrée (couche L2). Lorsque le nombre de couches cachées est supérieur à 1, l'autoencodeur est considéré comme étant profond [31].

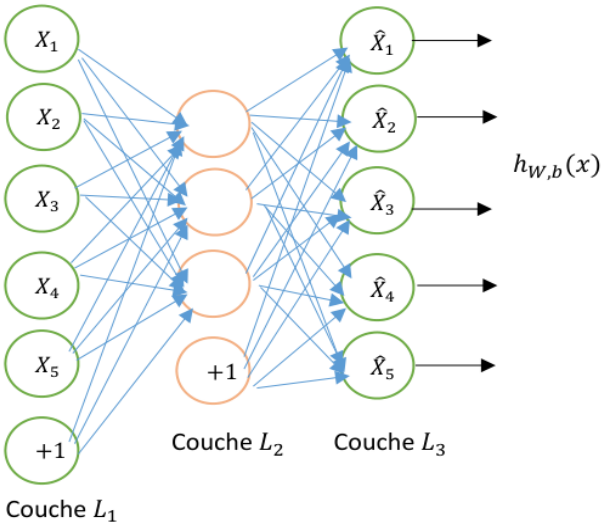


Figure II.9 : Architecture d'un autoencodeur [31].

II.5.2 **Le réseau de neurones récurrent (RNN)**

Les réseaux récurrents (RNN) sont des réseaux de neurones qui permettent la propagation de l'information dans les deux sens, y compris des couches profondes aux premières couches. Ils sont particulièrement adaptés aux applications impliquant le contexte et le traitement de séquences temporelles. Cependant, les RNN classiques ont une "mémoire à court terme" limitée et ont tendance à oublier l'information après un certain nombre d'itérations. Pour résoudre ce problème, les LSTM (Long Short Term Memory) ont été développés (une variante des RNN). Les RNN ont un potentiel théorique important, étant "Turing-complets" c.-à-d. qu'ils sont en mesure, théoriquement, de simuler n'importe quel algorithme [43].

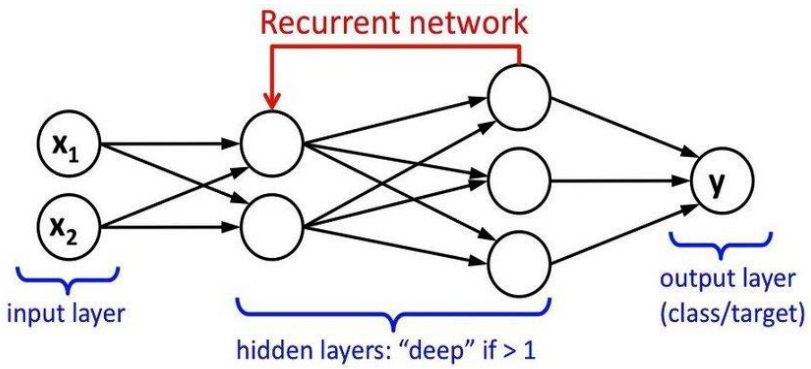


Figure II.10 : architecture d’un RNN [51].

II.5.3 **Le réseau de neurones convolutif (CNN)**

Le CNN est un type de réseau de neurones conçu pour traiter efficacement des données structurées.

On va détailler ce point par la suite car dans notre cas c’est ce type de réseau de neurones qui nous intéresse.

II.6 **Les réseaux de neurones convolutifs**

Les réseaux de neurones convolutifs sont spécifiquement adaptés pour le traitement d'images ce qui traduit leur architecture composée de deux blocs principaux.

Le premier bloc des réseaux de neurones convolutifs agit comme un extracteur de caractéristiques. Pour ce faire, il applique des opérations de filtrage par convolution. La première couche effectue une convolution de l'image avec plusieurs noyaux, produisant ainsi des "cartes de caractéristiques" (feature maps). Ces cartes de caractéristiques sont ensuite normalisées à l'aide d'une fonction d'activation et/ou redimensionnées. Ce processus peut être répété plusieurs fois. Les valeurs des dernières cartes de caractéristiques sont concaténées pour former un vecteur. Ce vecteur définit la sortie du premier bloc et devient alors l'entrée du second bloc.

Le second bloc son rôle est de transformer les valeurs du vecteur d'entrée en utilisant plusieurs combinaisons linéaires et fonctions d'activation, afin de produire un nouveau vecteur en sortie. Ce dernier vecteur a un nombre d'éléments égal au nombre de classes, où chaque élément représente la probabilité que l'image appartienne à la classe correspondante [38].

II.6.1 **Architecture d’un réseau de neurones convolutif**

L’architecture d’un CNN se compose de trois grandes couches : la couche de convolution, la couche de pooling et la couche fully-connected.

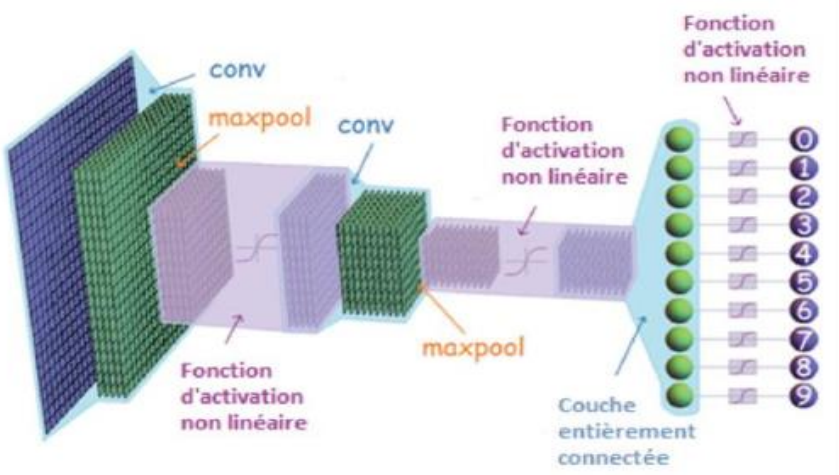


Figure II.11 : Architecture d'un CNN [23].

II.6.1.1 La couche de convolution

Le but de la convolution est d'extraire d'une images les caractéristiques qui pourraient nous intéresser. Et cela se fait en glissant une matrice par-dessus une image, et pour chaque pixel, utiliser la somme de la multiplication de ce pixel par la valeur de la matrice.

On prend l'exemple d'une image (la matrice M représente les pixels de l'image) et une autre matrice comme exemple de filtre F.

On applique le filtre sur l'mage en multipliant chaque valeur des pixels de l'image par chaque valeur correspondante du filtre. Puis on additionne toutes ces valeurs pour avoir une seule valeur. Comme on peut le voir dans Figure II.12 :

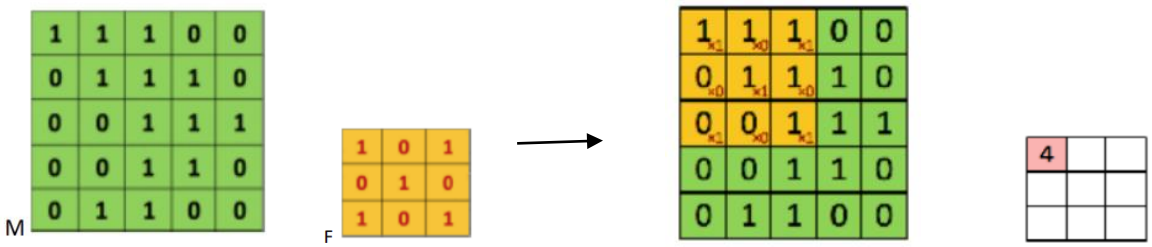


Figure II.12 : Illustration de l'application du filtre [38].

Ensuite le filtre se déplace (selon le pas) à chaque itération jusqu'à ce qu'on ait le résultat final qui est la feature map. Pendant le déplacement du filtre et afin de préserver les informations présentes aux bords de la matrice (l'image), le CNN utilise une méthode appelée "padding". Cette technique consiste à ajouter des pixels supplémentaires le long des frontières de la matrice, qui eux sont généralement définis à la valeur "0".

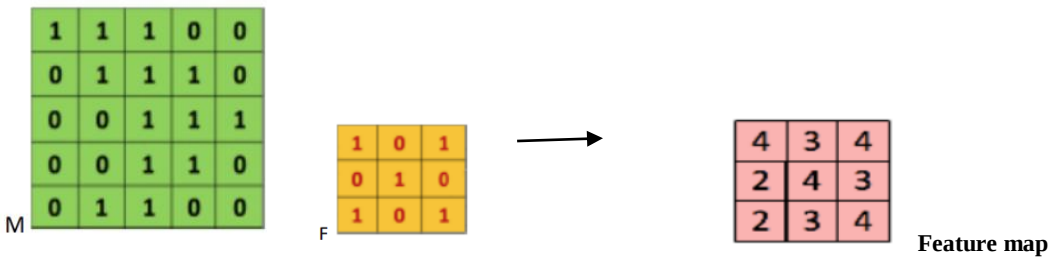


Figure II.13 : Illustration du résultat de convolution [38].

Dans ce cas on a pris un exemple avec des valeur binaires. Mais en réalité les pixels qui forment une image sont compris entre « 0 » et « 255 » pour une image à 8 bits, donc le résultat de la convolution peut comporter des valeurs supérieures à 255. Afin de normaliser ces valeurs, on leurs applique des fonctions d'activation non linéaires pour obtenir des nouvelles valeurs à partir des résultats de convolution obtenus [35][38].

Une fois le résultat de convolution obtenu on applique une fonction de correction ReLU (Rectified Linear Units) qui remplace toutes les valeurs négatives reçues en entrées par des zéros. Ce qui permet d'améliorer l'efficacité du traitement et elle joue le rôle de fonction d'activation [41].

ReLU(u)=max (0, u)

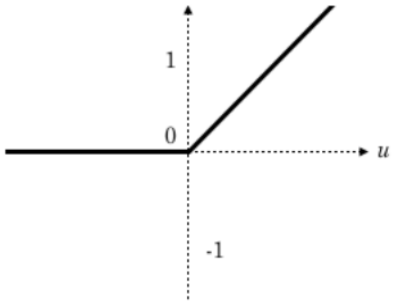


Figure II.14 : Fonction de la couche de correction ReLU [38].

II.6.1.2 La couche pooling

Ce type de couche est souvent introduite entre deux couches de convolution : elle reçoit en entrée plusieurs feature maps, et applique à chacune d'entre elles l'opération de pooling. Le but de cette couche est de réduire progressivement la taille de l'image et de grader que les informations pertinentes. Ceci amené à une réduction de la quantité de paramètre et de calcul dans le réseau, ce qui va améliorer l'efficacité du réseau et évité le sur-apprentissage.

Pour ce faire on découpe l'image en cellules régulières (on utilise souvent des cellules carrées de petite taille pour ne pas perdre beaucoup d'informations), puis on garde le pixel ayant la valeur maximale (max pooling) ou bien la moyennes des pixels (avg pooling) [39][40].

Comme on peut le voir dans Figure II.15 :

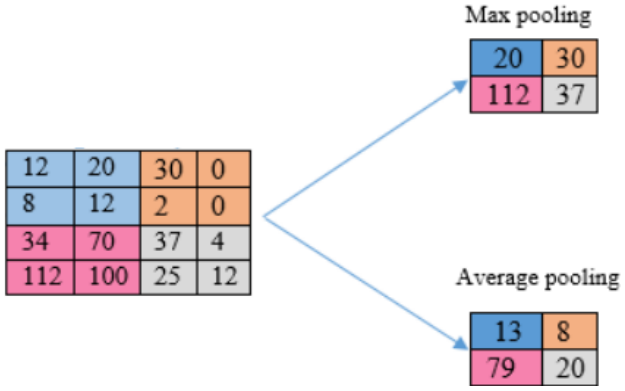


Figure II.15 : Exemple de l'étape de pooling [23].

II.6.1.3 La couche fully-connected

La couche fully-connected constitue souvent la dernière couche d'un CNN. Son objectif est de classifier l'image en entrée du réseau, et cela en combinant les features extraites par les couches précédentes. Cette couche reçoit un vecteur en entrée contenant les pixels des images filtrées, corrigées et réduites par le pooling ensuite elle applique une combinaison linéaire puis éventuellement une fonction d'activation aux valeurs reçues afin de produire un nouveau vecteur en sortie. Elle renvoie un vecteur de taille N ou chaque élément du vecteur indique la probabilité pour que l'image en entrée appartienne à une classe.

Pour calculer ses probabilités, la couche fully-connected multiplie chaque élément d'entrée par un poids spécifique, puis elle effectue la somme de ces produits, et enfin elle applique une fonction d'activation [39].

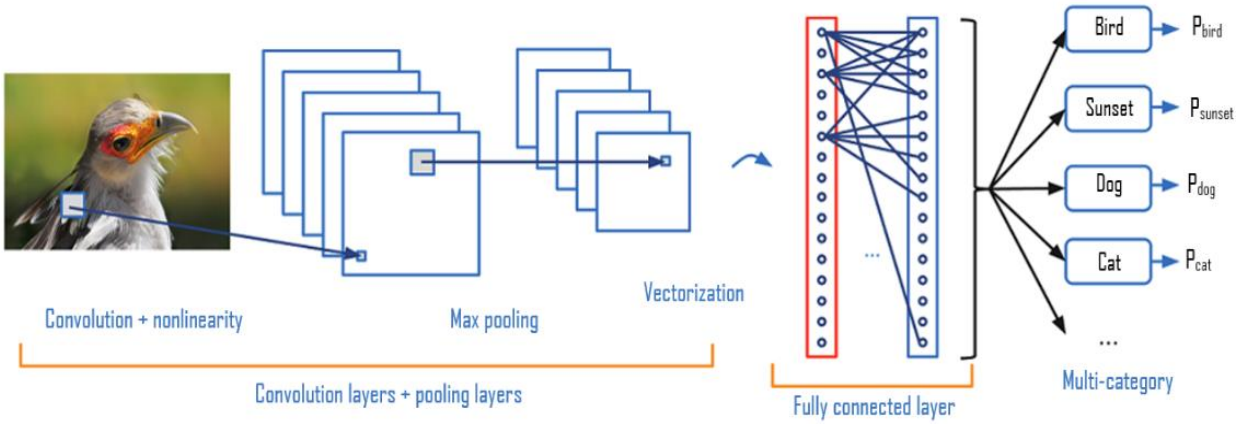


Figure II.16 : Schéma général d'un CNN [53].

II.7 Conclusion

En conclusion, l'apprentissage profond et les réseaux de neurones convolutifs ont révolutionné l'apprentissage automatique en permettant une extraction efficace des caractéristiques complexes des données. Leur capacité à apprendre des représentations hiérarchiques et à capturer des motifs complexes a ouvert de nouvelles perspectives dans de nombreux domaines de l'intelligence artificielle.

CHAPITRE III

SYSTEME DE DETECTION ET DE CLASSIFICATION DES IRIS

III.

III.1 Introduction

L'iris est une caractéristique unique de l'œil humain qui a été largement utilisée dans les systèmes de reconnaissance et de sécurité.

Ce chapitre représente une continuité d'un travail passé traitant la reconnaissance des iris par des réseaux de neurones convolutifs [23], qui consiste principalement à compléter le pipeline du système de reconnaissance de l'iris et étude de ses performances. Nous présentons une méthode de segmentation et de normalisation de l'iris, ainsi que l'évaluation des performances d'un modèle CNN en combinaison avec différents classifieurs, réalisée sur deux bases de données différentes et simulées sur Matlab.

III.2 Système de détection et classification des iris proposé

Le système de détection et de classification des iris implémenté est basé principalement sur quatre étapes :

1 - **Prétraitement des données :** cette étape consiste à localiser les régions de l'iris et à les normaliser

- **Localisation de l'iris :** cette étape utilise la méthode de Hough circulaire.
- **Normalisation :** la normalisation est réalisée en utilisant l'approche de Daugman.

2 - Extraction des caractéristiques et classification :

- Extraction des caractéristiques cachées est réalisée par les couches de convolution du réseau de neurones convolutif proposé dans [23].
- La classification des résultats de la dernière couche de convolution est ensuite réalisée en utilisant divers modèles de classifications : réseau de neurones à une couche entièrement connecté, K plus proches voisins, arbre de décision, Séparateur à large marge et Bayes naïf.

3 - **Matching :** c'est la dernière étape, elle consiste à détecter et à classer de nouvelles instances des iris.

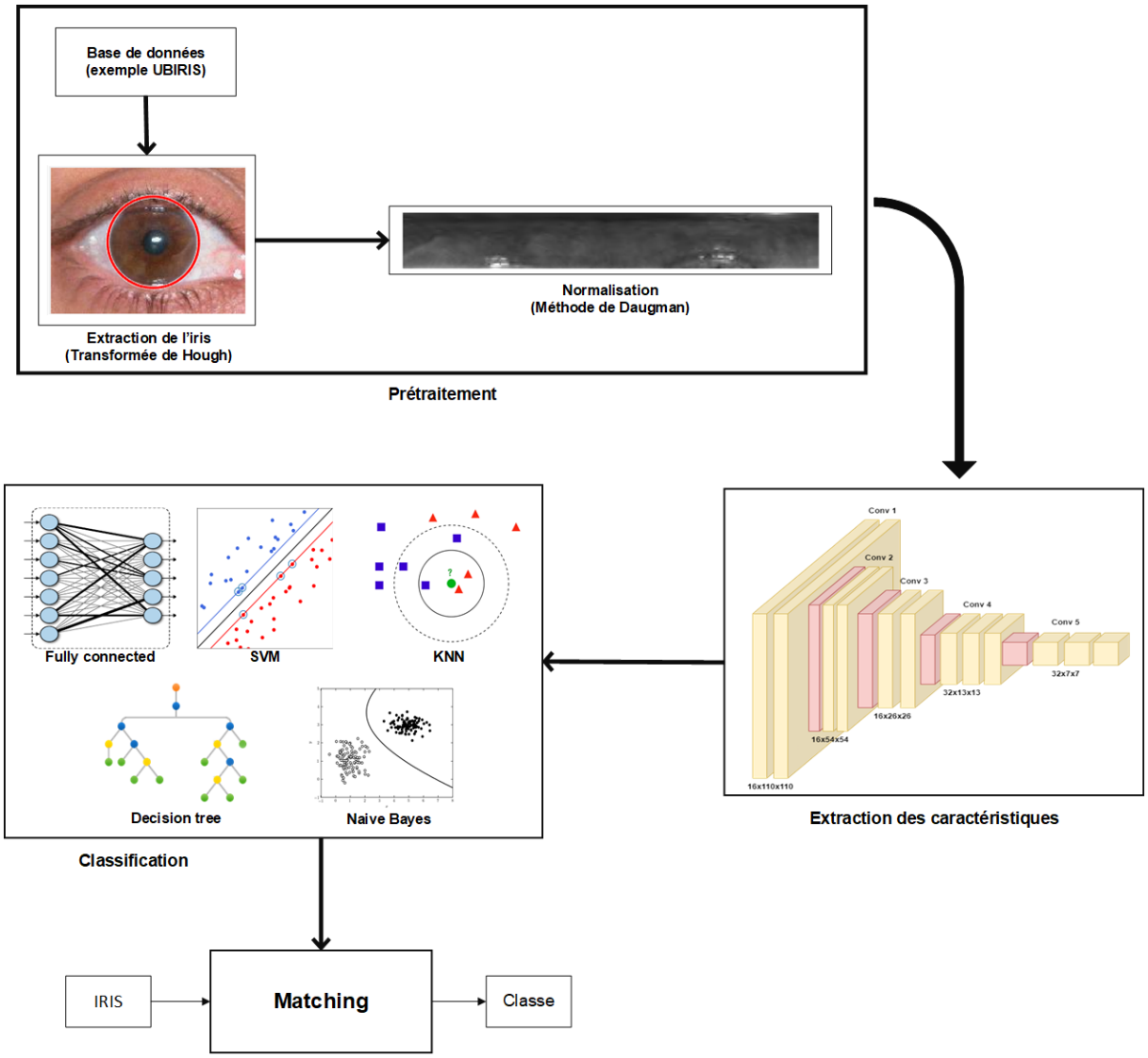


Figure III.1 : Schéma du système de détection et classification des iris.

III.3 Implémentation du système de détection et de classification des iris proposé

III.3.1 Environnement de développement :

L'environnement de développement comprend un ordinateur équipé du système d'exploitation Windows 10 x64 bits. MATLAB (R2021b) est utilisé comme logiciel principal pour la simulation. Ainsi que la base de données UBIRIS contenant des images d'iris.

Le PC utilisé pour l'exécution des codes de segmentation et de normalisation de l'iris est doté d'un processeur AMD Ryzen 5 5500 6 cœurs 12 threads avec une fréquence de 3.6 GHz jusqu'à 4.2GHz en boost et 16 Go de mémoire RAM avec une fréquence de 3200 MHz. Cela garantit une puissance de calcul suffisante pour le traitement des données et l'exécution des scripts MATLAB.

En ce qui concerne l'accélération matérielle, le PC est équipé d'une carte graphique AMD Radeon RX 580 avec 8GO de mémoire vidéo dédiée, ce qui permet d'exploiter les fonctionnalités de calcul GPU.

Pour ce qui est du stockage le PC possède un SSD de 225 Go avec une vitesse de lecture de 520 Mb/s.

III.3.2 Les base de données utilisées

III.3.2.1 Base de données UBIRIS

La base de données sur laquelle on a fait les tests est la base de données UBIRIS qui est une collection d'images d'iris utilisée dans la recherche sur la reconnaissance de l'iris. Elle offre une variété d'images acquises dans des conditions contrôlées, permettant d'évaluer la performance des algorithmes de reconnaissance dans des situations réelles. Les images de cette base de données sont prises avec la caméra Nikon E5700 et elle a été créé en septembre 2004 au sein du laboratoire Instituto de Telecomunicações au Portugal [45].

Elle comprend les images de 241 utilisateurs, toutes au format jpg (*.jpg) avec une résolution de 150 x 200 pixels.

III.3.2.2 Qualité de la base de données

parameter	Good	Average	Bad
Focus	73.83%	17.53%	8.63%
Reflections	58.87%	36.78%	4.34%
Visible Iris	36.73%	47.83%	15.44%

Tableau III.1 : Qualité de la base de données UBIRIS [45].

III.3.2.1 Base de données IITD

La base de données IIT Delhi Iris (IITD) est une collection d'images d'iris principalement obtenues auprès des étudiants et du personnel de l'IIT Delhi, situé à New Delhi, en Inde. Cette base de données a été constituée entre janvier et juillet 2007 au sein du Biometrics Research Laboratory, et les images ont été prises grâce à une caméra numérique CMOS JIRIS, modèle JPC1000. Elle comprend les images de 224 utilisateurs, toutes au format bitmap (*.bmp) avec une résolution de 240 x 420 pixels.

Les images normalisées de cette base de données ont une résolution de 48 x 432 pixels. La normalisation est réalisée en utilisant le modèle de Daugman's Rubber [56].

III.3.3 Segmentation

La méthode de segmentation principale utilisée est la transformée de Hough qui permet de détecter des cercles dans des images. Dans notre cas, ces cercles représentent les limites de l'iris.

III.3.3.1 Méthodologie

Notre approche se divise en plusieurs étapes. Tout d'abord, nous prétraitons l'images en la convertissant en niveaux de gris à l'aide de la fonction « **rgb2gray** ». Ensuite, nous appliquons la détection de contours à l'image en utilisant l'algorithme de Canny, ce qui nous permet de repérer les contours présents dans l'image.

Une fois les contours identifiés, nous utilisons la transformée de Hough circulaire **« imfindcircles** » pour détecter les limites de l'iris. Cette étape nous permet de localiser les cercles présents dans l'image qui correspondent à l'iris. Nous sélectionnons ensuite le cercle avec le plus grand rayon comme étant la limite de l'iris.

Pour isoler l'iris de l'image d'origine, nous créons un masque binaire de la région de l'iris. À l'aide des coordonnées du cercle détecté, nous calculons la distance entre chaque pixel de l'image et le centre du cercle. Si la distance est inférieure ou égale au rayon de l'iris, le pixel est inclus dans le masque binaire.

III.3.3.2 Résultats

Après avoir appliqué notre programme Matlab a certaines images de la base de données UBIRIS on a eu les résultats suivants :

La Figure III.2 montre trois images représentant le processus de segmentation de l'iris.

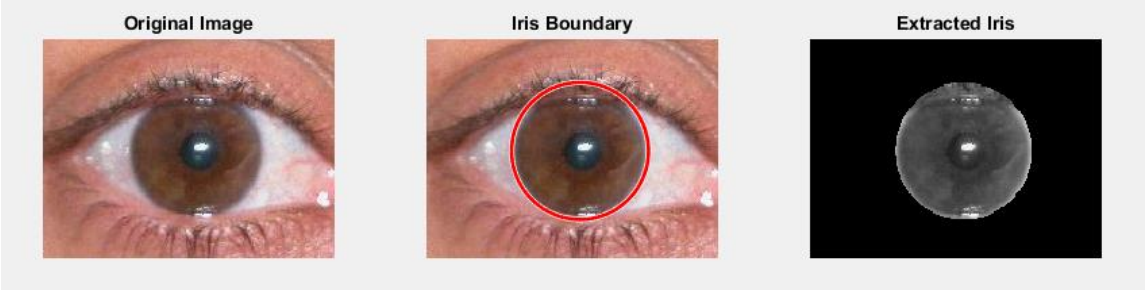


Figure III.2 : Résultats de la segmentation de l'iris.

La première image affiche l'image d'origine sur laquelle nous avons effectué la détection et l'extraction. La deuxième image présente le contour de l'iris détecté à l'aide de cercles rouges. En utilisant la méthode de Hough circulaire, nous avons réussi à identifier précisément les limites de l'iris.

Enfin, la troisième image montre l'iris extrait à partir de l'image originale en appliquant le masque binaire. Cette étape nous permet d'isoler avec précision la région de l'iris, ce qui facilite les futures analyses et traitements.

III.3.4 **Normalisation**

Pour ce qui est de la normalisation, nous avons conservé la même base de données sur laquelle nous allons appliquer un autre programme Matlab. La normalisation est réalisée en utilisant l'approche de Daugman [46].

III.3.4.1 Méthodologie

Nous commençons en convertissons l'image d'iris en niveaux de gris à l'aide de la fonction « **rgb2gray** ». Ensuite, nous appliquons la conversion en double format avec « **im2double** » pour normaliser les valeurs des pixels de l'image (les valeurs des pixels sont mises à l'échelle pour être comprises entre 0 et 1, où 0 représente le noir et 1 représente le blanc) afin de faciliter le traitement et l'analyse de l'image.

Après cela, nous procédons à la détection des contours de l'iris en utilisant une méthode basée sur la recherche des minimums locaux. Nous remplissons d'abord tous les espaces vides de l'image pour faciliter la détection. Ensuite, nous effectuons un seuillage de l'image (simplifier l'image en réduisant les détails inutiles) pour identifier les pixels correspondant aux contours de l'iris (le filtre de Gabor a été utilisé avant d'appliquer le seuillage).

Une fois les pixels des contours détectés, nous éliminons ceux qui ne correspondent pas à des minimums locaux. Cela nous permet de converger plus rapidement vers les contours réels de l'iris.

Ensuite, nous calculons les dérivées partielles pour chaque pixel des contours restants. Cela nous aide à déterminer les centres de l'iris et de la pupille, ainsi que leurs contours. Nous utilisons les fonctions « **partiales_derivative »** et « **find_center »** pour effectuer ces calculs.

Une fois que nous avons déterminé les contours, nous procédons à la normalisation de l'iris à l'aide de la fonction « **normalize_iris »**. Cette étape consiste à normaliser l'iris en taille fix.

Enfin, nous créons le code de l'iris à l'aide de la fonction « **iris_coding »**. Ce code représente les caractéristiques de l'iris normalisée, ce qui facilite les étapes de comparaison et de reconnaissance.

III.3.4.2 Résultats

Apres avoir appliquer le programme Matlab pour la normalisation a certaines images de la base de données on a eu les résultats suivants :

La Figure III.3 montre deux images représentant la normalisation et l'extraction du code de l'iris.

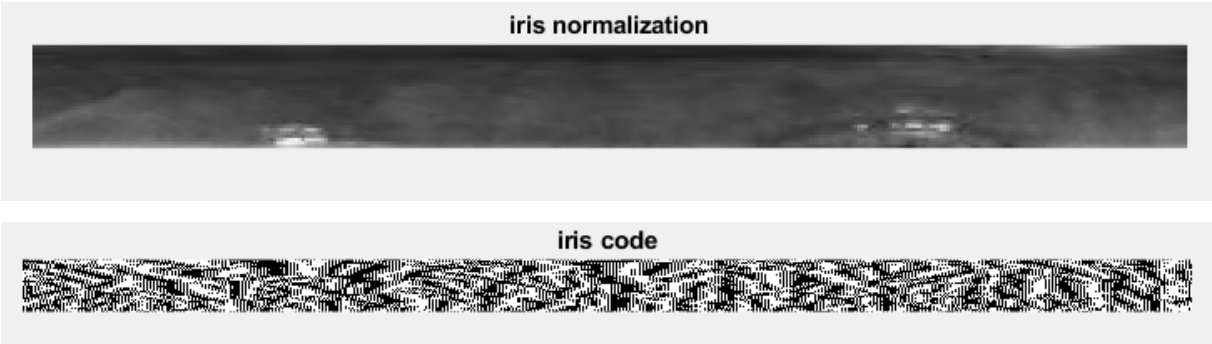


Figure III.3 : Résultats de l'étape de normalisation.

La première image correspond au résultat de la normalisation de l'iris.
La deuxième image quant à elle est la représentation du code de l'iris obtenu à partir de l'image normalisée.

III.3.5 Architecture du CNN utilisée

Le système de reconnaissance de l'iris implémenté utilise un modèle de classification basée sur un réseau de neurones convolutif dont l'architecture a été proposée dans [23].

Le système de

- **Couche d'entrée :** des images normalisées de taille 3 x 48 x 432.
- **Cinq couches de convolution et une couche entièrement connectée :** ces couches permettent de détecter les caractéristiques cachées des images de l'iris.
- **Couche entièrement connectée :** avec une seule couche cachée, cette couche permet la classification des images en fonction des caractéristiques extraites par les couches de convolution.
- Les valeurs des composantes du vecteur de sortie (prédiction) sont normalisées en utilisant l'équation softmax.
- La métrique à optimiser est la perte d'entropie croisée (cross entropy loss), définie par la formule suivante : $CEL = -\sum y_i \times log(\hat{y}_i)$ où y_i et $\hat{y}_i$sont respectivement la

valeur réelle et la valeur estimée (probabilité) de la i-ème composante du vecteur de sortie.

	Filtres	**Kernel**	**Stride**	**Padding**	**Fonction d'activation**
Convolution 1	16	8	2	2	ReLU
Max pooling 1		4	2		
Convolution 2	16	4	1	2	ReLU
Max pooling 2		4	2		
Convolution 3	32	3	1	2	ReLU
Max pooling 3		3	2		
Convolution 4	32	3	1	2	ReLU
Max pooling 4		3	2		
Convolution 5	64	3	1	2	ReLU

Tableau III .2 : Caractéristiques du modèle CNN utilisé.

Le modèle cité dans le tableau précédent, se compose de cinq couches de convolution, quatre couches de max pooling, et une couche entièrement connectée. L'image d'entrée normalisée a une taille de 3 x 48 x 432. Tout d'abord, l'image passe par la première couche de convolution, qui est composée de 16 filtres de taille 8 x 8. Après chaque couche de convolution, nous appliquons une fonction d'activation ReLU pour créer les "feature maps". Ensuite, le max pooling est utilisé pour réduire la taille de l'image et des paramètres. À la sortie de cette couche, nous obtenons 16 "feature maps" de taille 110 x 110.

Les 16 "feature maps" obtenus sont ensuite utilisés en entrée pour la deuxième couche de convolution, qui est également composée de 16 filtres de taille 4 x 4. Après la convolution et le max pooling, nous obtenons 16 "feature maps" de taille 54 x 54.

Ce processus est répété pour les couches de convolution trois, quatre et cinq. La fonction d'activation ReLU est appliquée à chaque couche de convolution, suivie d'une couche de max pooling. À la sortie de la dernière couche, nous avons 64 "feature maps", chacune ayant une taille de 3 x 3. Le vecteur de caractéristiques résultant des convolutions a une dimension de 64 x 9 x 9.

Après ces cinq couches de convolution, nous utilisons un réseau de neurones composé d'une seule couche entièrement connectée avec 1024 neurones, suivie d'une couche softmax qui calcule la distribution de probabilité.

Figure III.4 montre l'architecture de notre modèle :

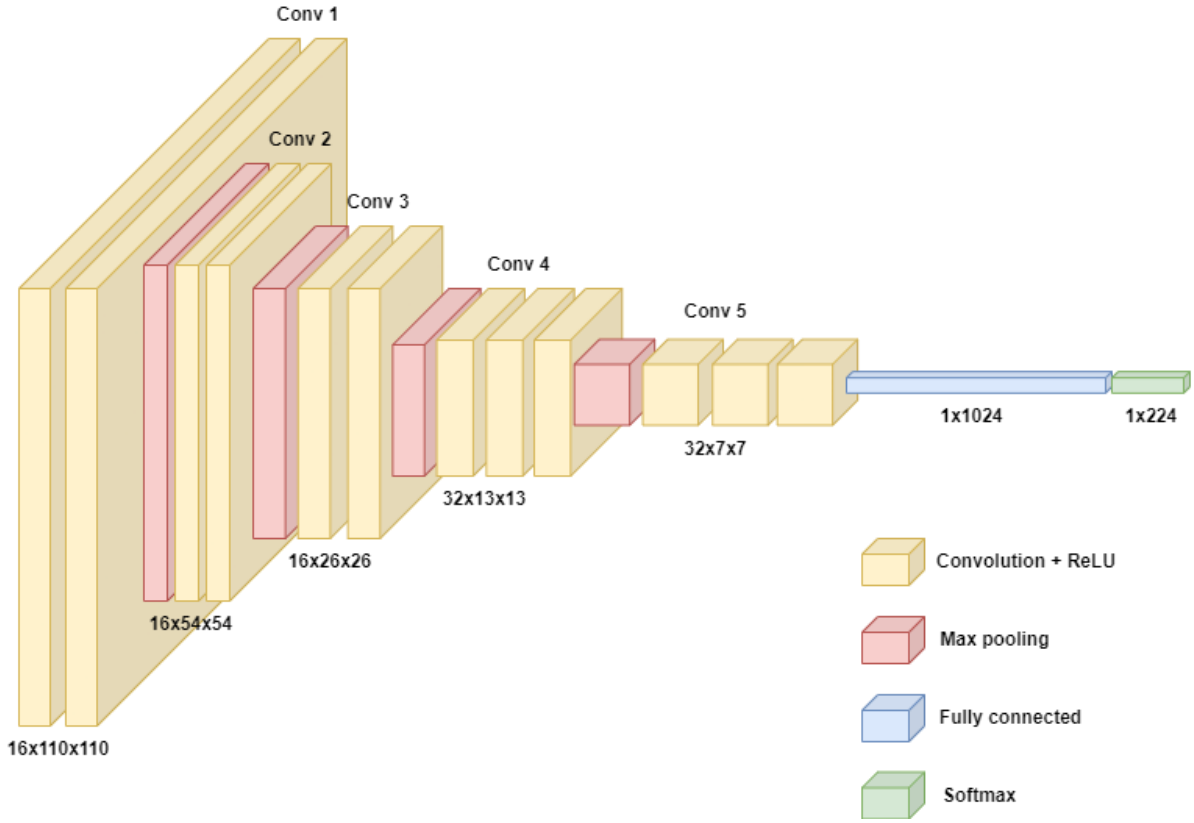


Figure III.4 : Architecture du modele CNN utilisée.

III.3.5.1 Etude des performances du CNN utilisé

On utilise la base de données UBIRIS qu'on vient de normaliser, ainsi que la base de données IITD qui contient déjà des images d'iris normalisées.

III.3.5.2 Définition de la matrice de confusion :

La matrice de confusion est un outil qui récapitule les résultats de prédiction d'un modèle de classification pour un problème spécifique. Elle permet de comparer les données réelles d'une variable cible avec les prédictions faites par le modèle. Les prédictions correctes et incorrectes sont révélées et réparties par classe, ce qui permet de les comparer avec des valeurs préétablies [23].

Dans notre cas, les matrices de confusion ont une taille de 224 x 224, où U (i, i) représente le nombre d'instances correctement classées pour la classe i, et U (i, j) (où $i \neq j$) représente le nombre d'instances de la classe i qui ont été classées par le modèle comme appartenant à la classe j.

Les quatre valeurs associées à une matrice de confusion sont les suivantes :

- Vrai positif (TP) : la prédiction et la valeur réelle sont positives.
- Vrai négatif (TN) : la prédiction et la valeur réelle sont négatives.
- Faux positif (FP) : la prédiction est positive alors que la valeur réelle est négative.
- Faux négatif (FN) : la prédiction est négative alors que la valeur réelle est positive.

III.3.5.3 Les métriques de performances

Différentes métriques peuvent être calculées à partir de la matrice de confusion pour faciliter l'interprétation et évaluer la qualité de précision du modèle. Voici quelques indicateurs couramment utilisés :

- **Précision (Accuracy) :** La précision mesure la proportion de prédictions correctes par rapport à l'ensemble des prédictions effectuées par le modèle.

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

- **Erreur (Error) :** L'erreur représente la proportion d'erreurs de prédiction par rapport à l'ensemble des prédictions effectuées par le modèle.

$$Error = \frac{(FP + FN)}{(TP + TN + FP + FN)}$$

- **Sensibilité (Sensitivity) :** La sensibilité, également appelée taux de vrais positifs, mesure la capacité du modèle à identifier correctement les cas positifs.

$$Sensitivity = \frac{TP}{(TP + FN)}$$

- **Spécificité (Specificity) :** La spécificité, également appelée taux de vrais négatifs, mesure la capacité du modèle à identifier correctement les cas négatifs.

$$Specificity = \frac{TN}{(TN + FP)}$$

- **Kappa :** Le coefficient de Kappa est une mesure qui évalue la qualité de performance d'un classificateur en prenant en compte la coïncidence entre ses performances et la qualité globale de l'exécution. Il est calculé comme suit :

$$Kappa\ (k) = \frac{(VP \times VN + FP \times FN)}{(n_1 n_2 + n_1' n_2')}$$

Où :

$$n_1 : VP + FP$$

$$n_2 : FN + VN$$

$$n_1' : VP + FN$$

$$n_2' : FP + VN$$

III.3.5.4 Relation entre le nombre de couches de convolution, la vitesse d'apprentissage et la précision

Notre modèle possède les caractéristiques suivantes :

- Batch de taille de 64 (chaque itération de l'entraînement utilise un sous-ensemble de 64 exemples pour mettre à jour les poids du modèle).
- Learning rate réduit de 20% après chaque cinq itérations.

- Gradient descent with momentum comme algorithme d'optimisation.

III.3.5.4.1 Cas 1 : Architecture a une seule couche de convolution

❖ **Evolution de la précision et la perte durant l'entrainement du modèle**

- **Base de données 1: UBIRIS**

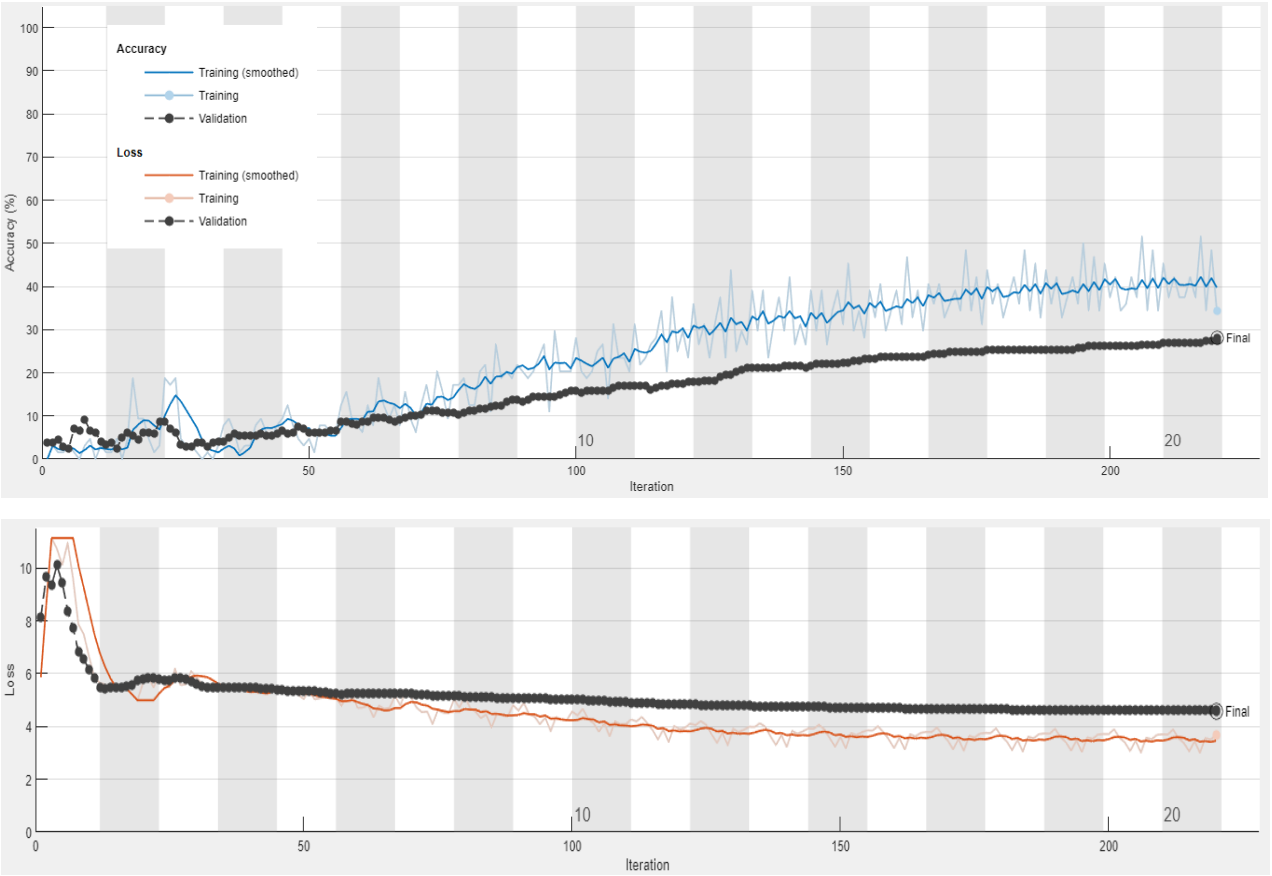
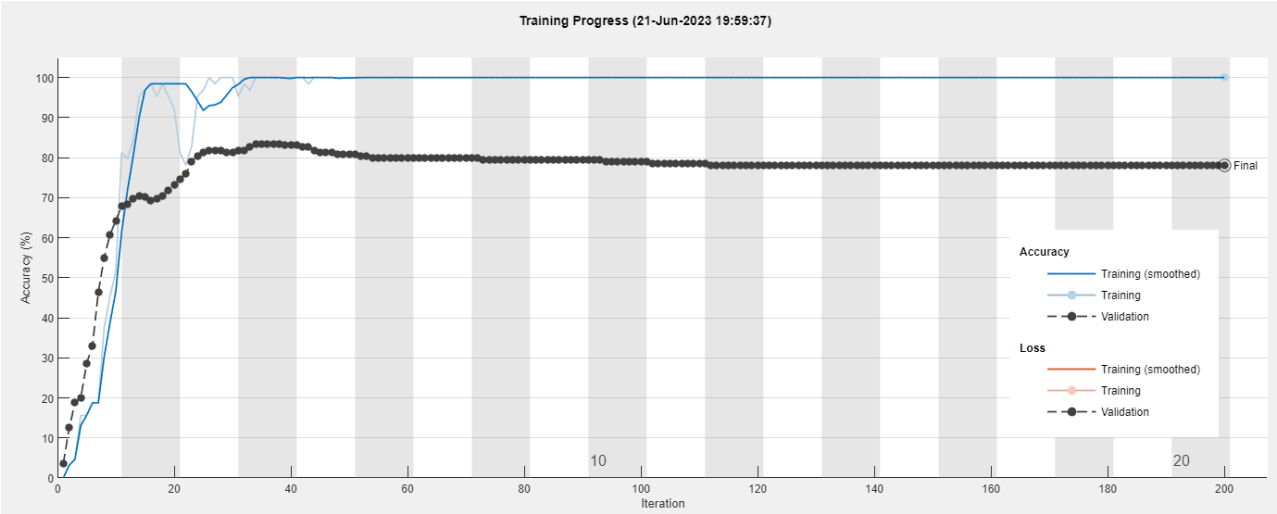


Figure III.5: Visualisation des résultats pour une architecture a une seule couche de convolution, UBIRIS.

- **Résultats:**
  - **Validation accuracy:** 28.22%, **Time:** 2 min 21 sec.
  - **Base de données 2: IITD**



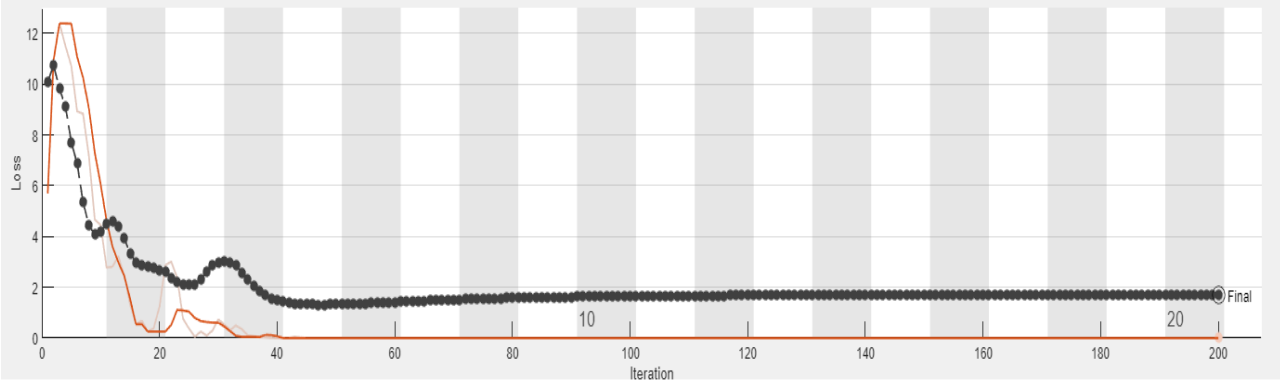


Figure III.6 : Visualisation des résultats pour une architecture a une seule couche de convolution, IITD.

- **Résultats :**
  - **Validation accuracy:** 78.12%, **Time:** 2 min 45 sec.

III.3.5.4.2 Cas 2 : Architecture a deux couches de convolution

❖ **Evolution de la précision et la perte durant l'entrainement du modèle**

- **Base de données 1: UBIRIS**

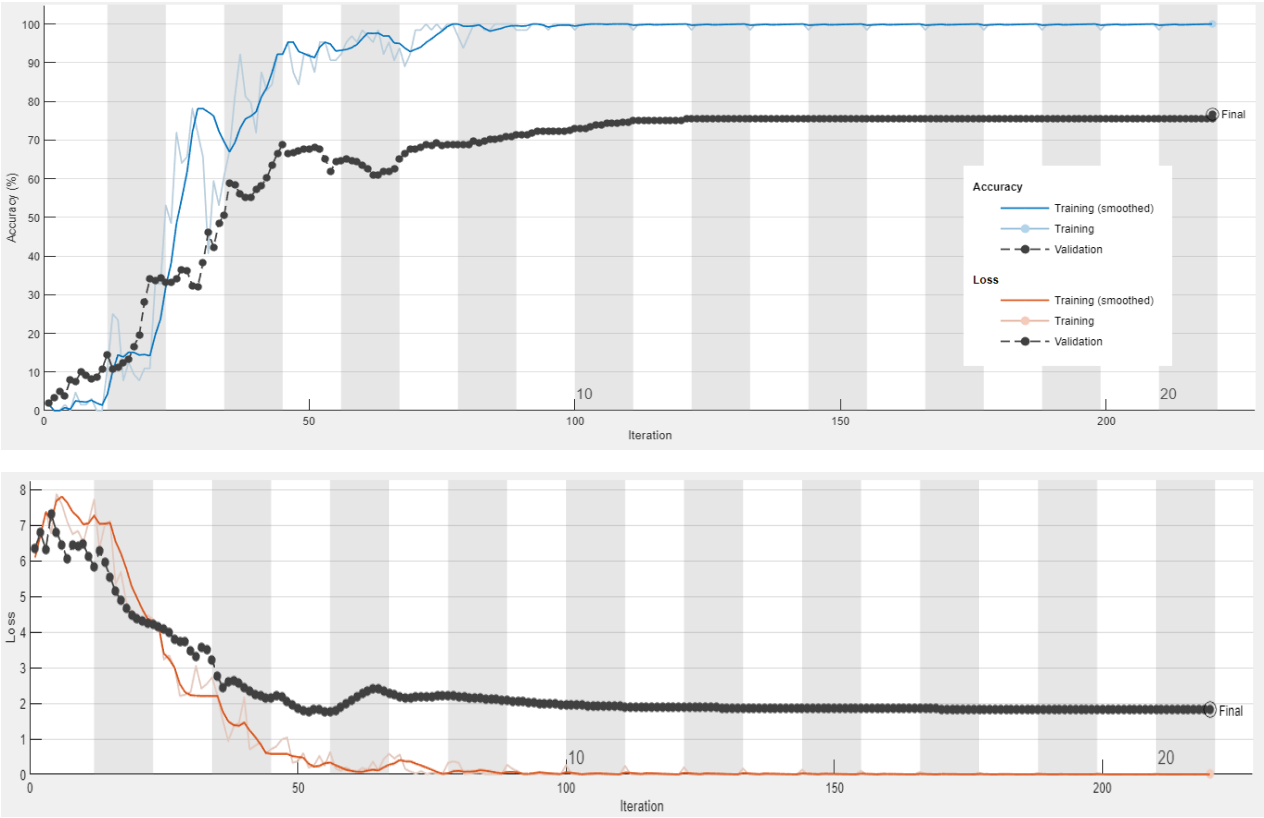


Figure III.7: Visualisation des résultats pour une architecture a une deux couches de convolution, UBIRIS.

- **Résultats:**

- **Validation accuracy:** 76.76%, **Time:** 2 min 32 sec

  - **Base de données 2: IITD**

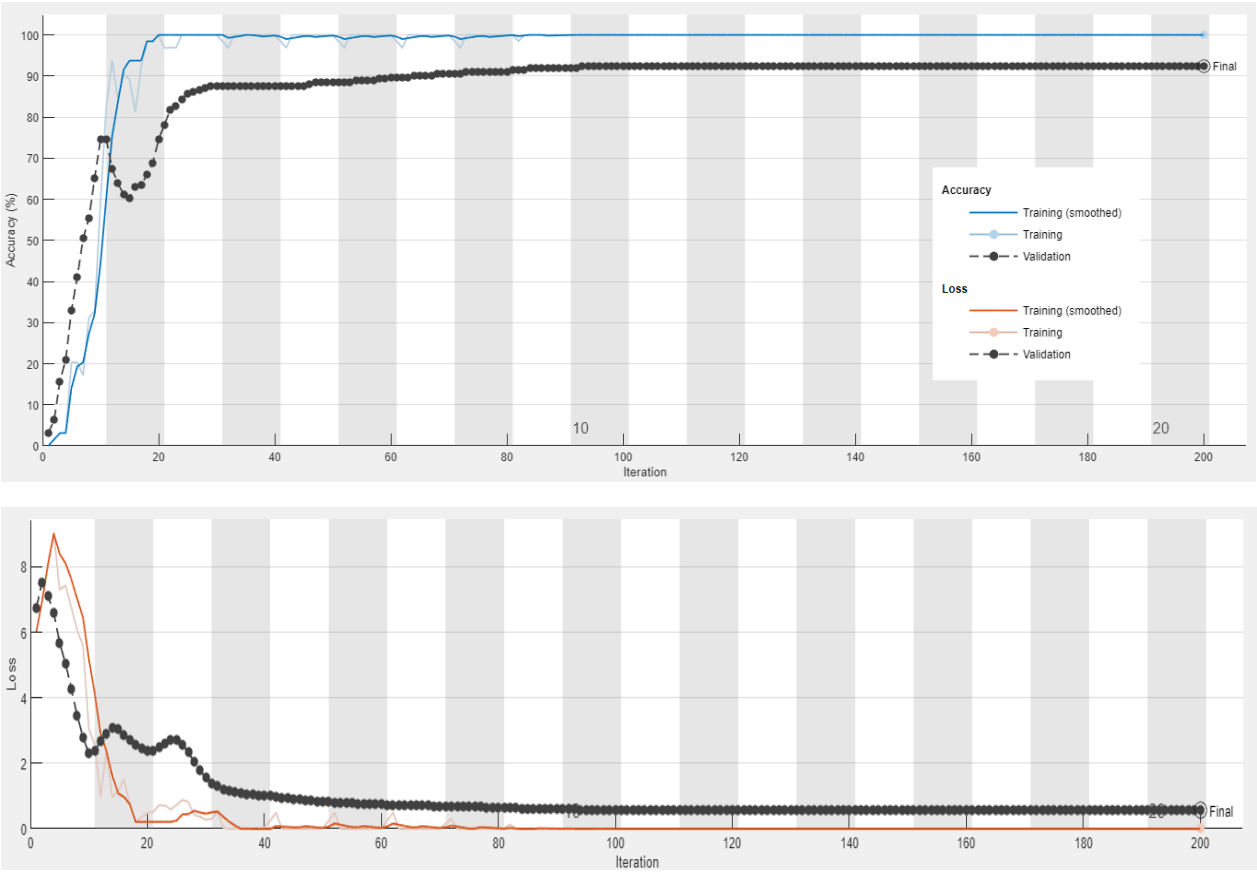


Figure III.8 : Visualisation des résultats pour une architecture a deux couches de convolution, IITD.

- **Résultats :**
  - **Validation accuracy:** 92.41%, **Time:** 3 min 15 sec.

III.3.5.4.3 Cas 3 : Architecture trois couches de convolution

❖ **Evolution de la précision et la perte durant l'entrainement du modèle**

- **Base de données 1: UBIRIS**

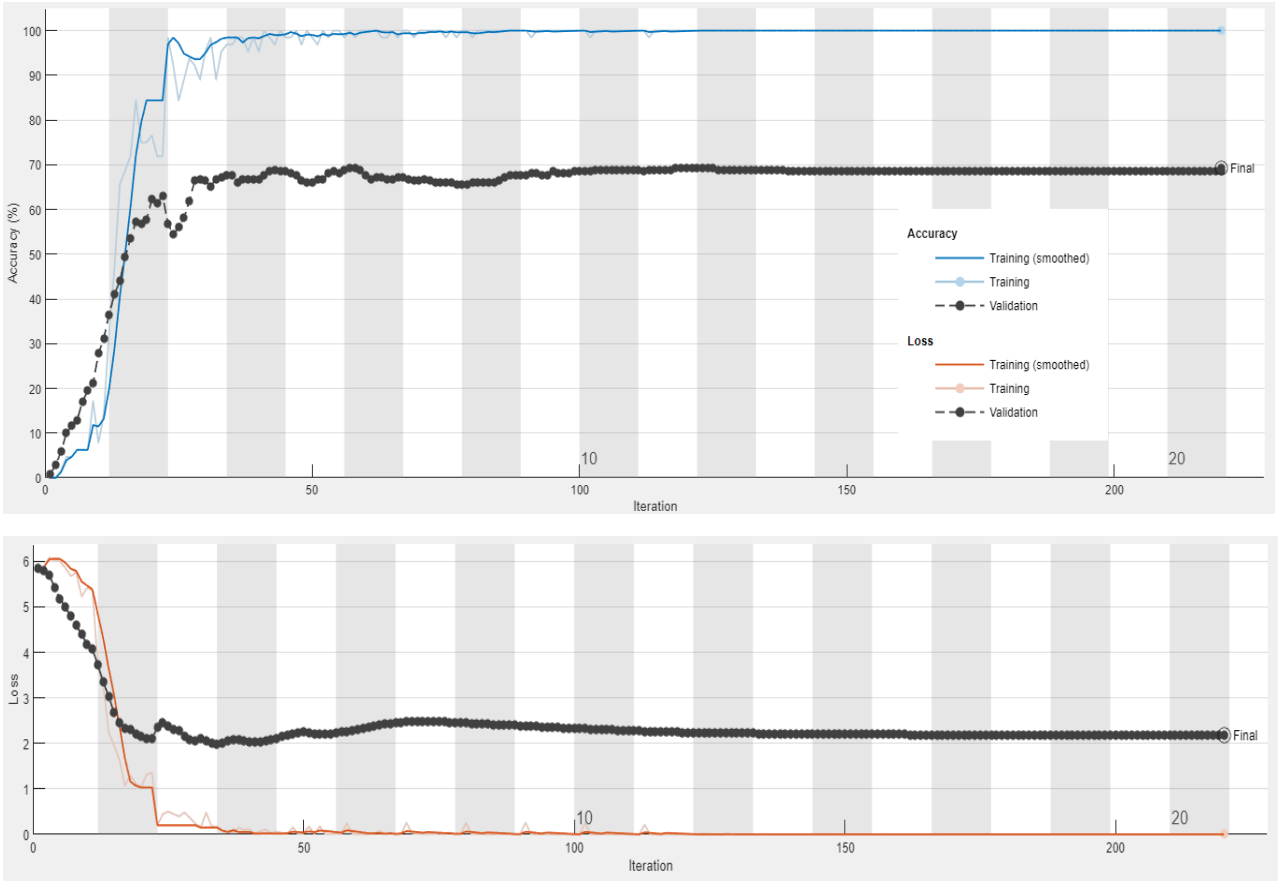
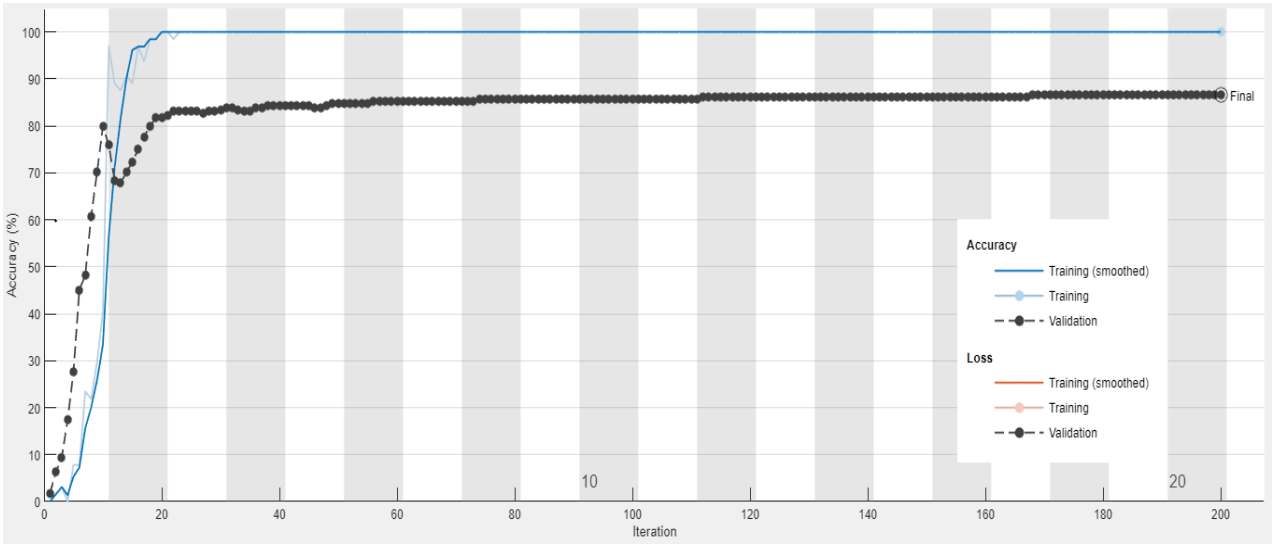


Figure III.9: Visualisation des résultats pour une architecture a trois couches de convolution, UBIRIS.

- **Résultats:**
  - **Validation accuracy:** 69.29%, **Time:** 2 min 44 sec.

- **Base de données 2: IITD**



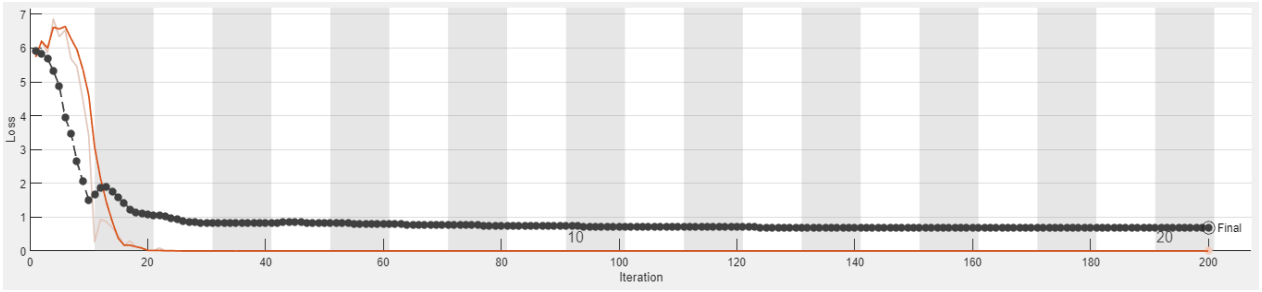


Figure III.10 : Visualisation des résultats pour une architecture a trois couches de convolution, IITD.

- **Résultats :**
  - **Validation accuracy:** 86.61%, **Time:** 3 min 9 sec.

III.3.5.4.4 Cas 4 : Architecture a quatre couches de convolution

❖ **Evolution de la précision et la perte durant l'entrainement du modèle**

- **Base de données 1: UBIRIS**

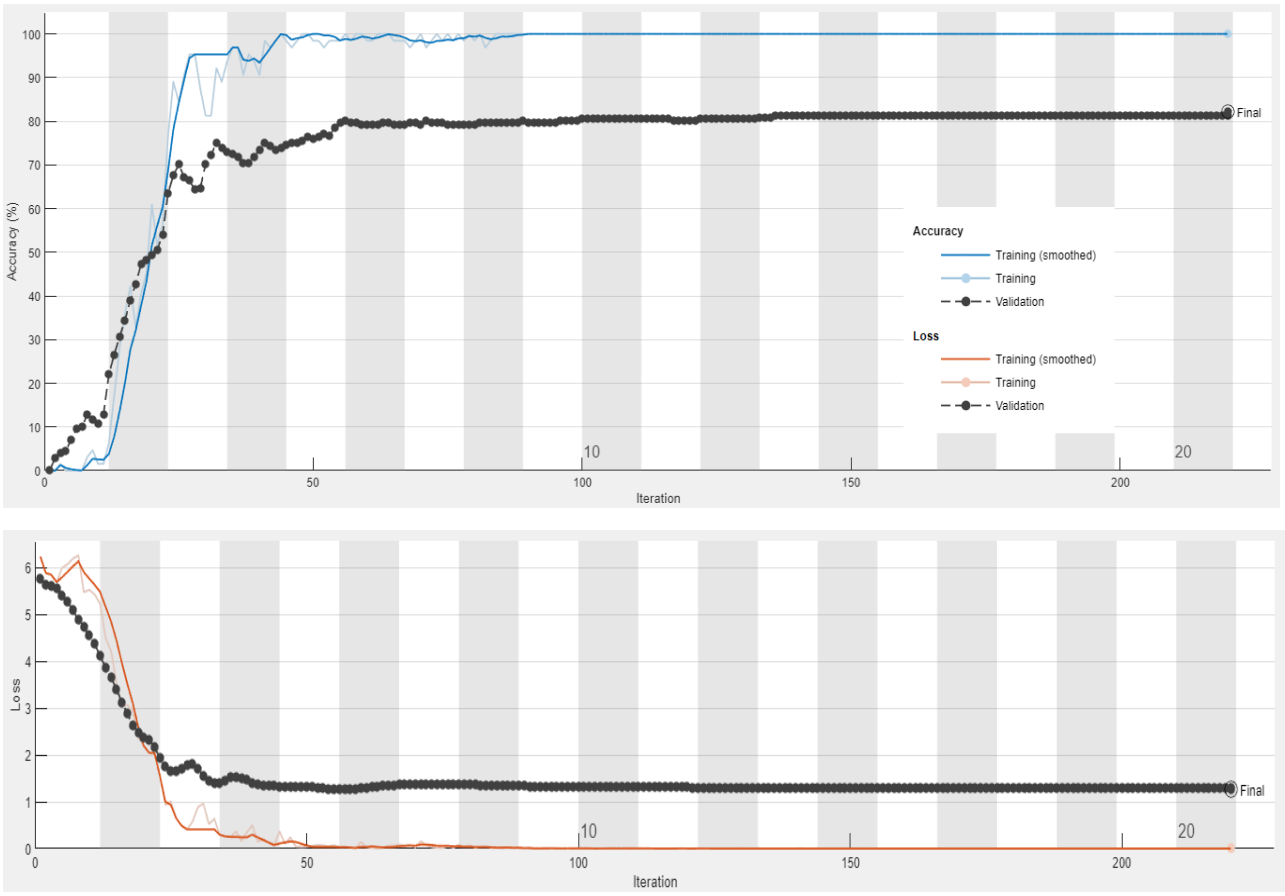


Figure III.11: Visualisation des résultats pour une architecture a quatre couches de convolution, UBIRIS.

- **Résultats:**
  - **Validation accuracy:** 82.16%, **Time:** 2 min 52 sec.

- **Base de données 2: IITD**

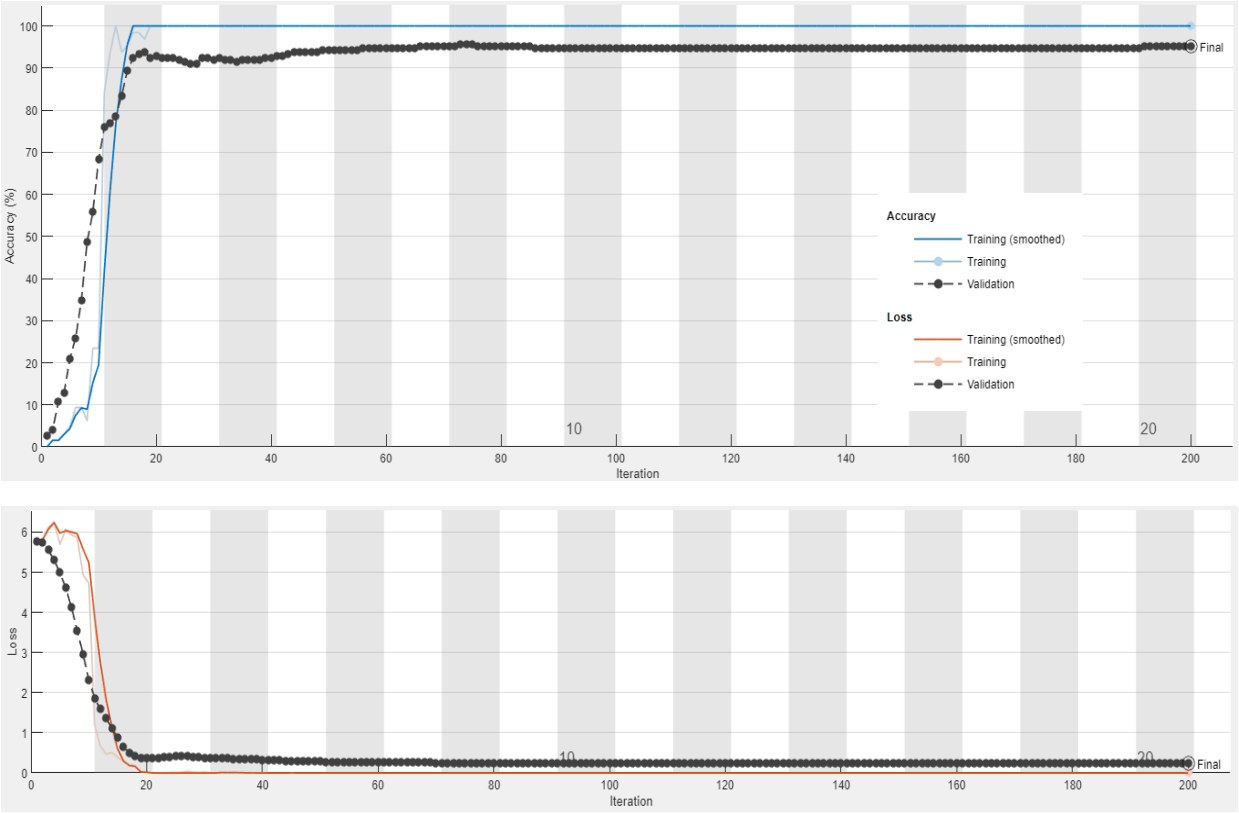


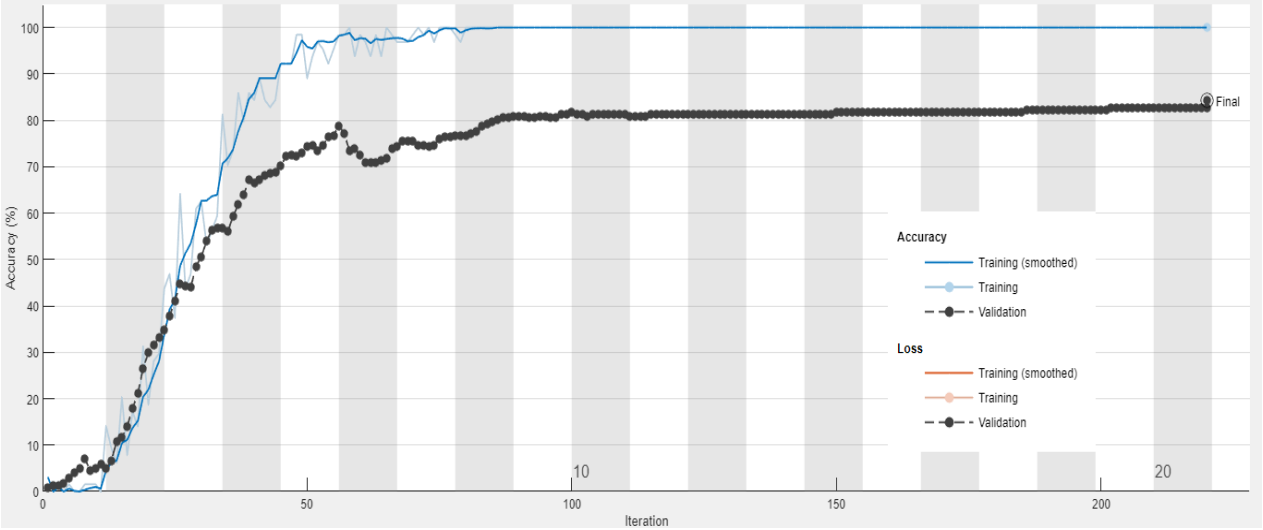
Figure III.12 : Visualisation des résultats pour une architecture a quatre couches de convolution, IITD.

- **Résultats :**
  - **Validation accuracy:** 95.09%, **Time:** 3 min 18 sec.

III.3.5.4.5 Cas 5 : Architecture a cinq couches de convolution

❖ **Evolution de la précision et la perte durant l'entrainement du modèle**

- **Base de données 1: UBIRIS**



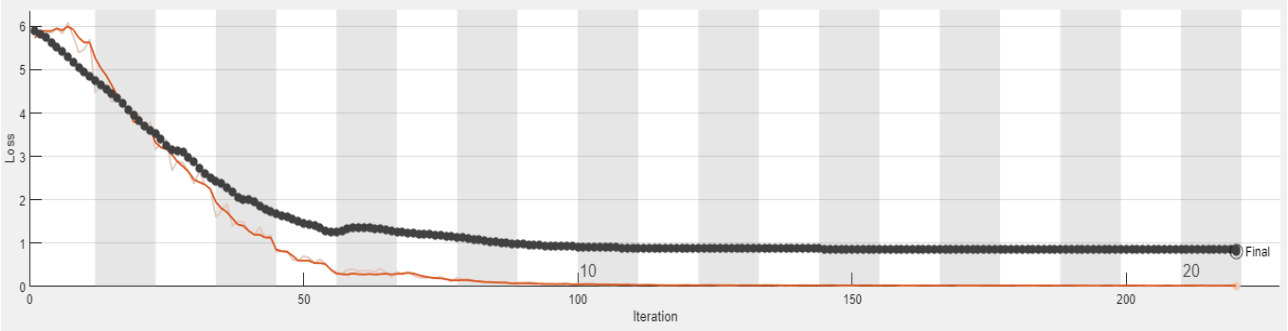


Figure III.13: Visualisation des résultats pour une architecture cinq couches de convolution, UBIRIS.

- **Résultats:**
  - **Validation accuracy:** 84.23%, **Time:** 2 min 55 sec.

- **Base de données 2: IITD**

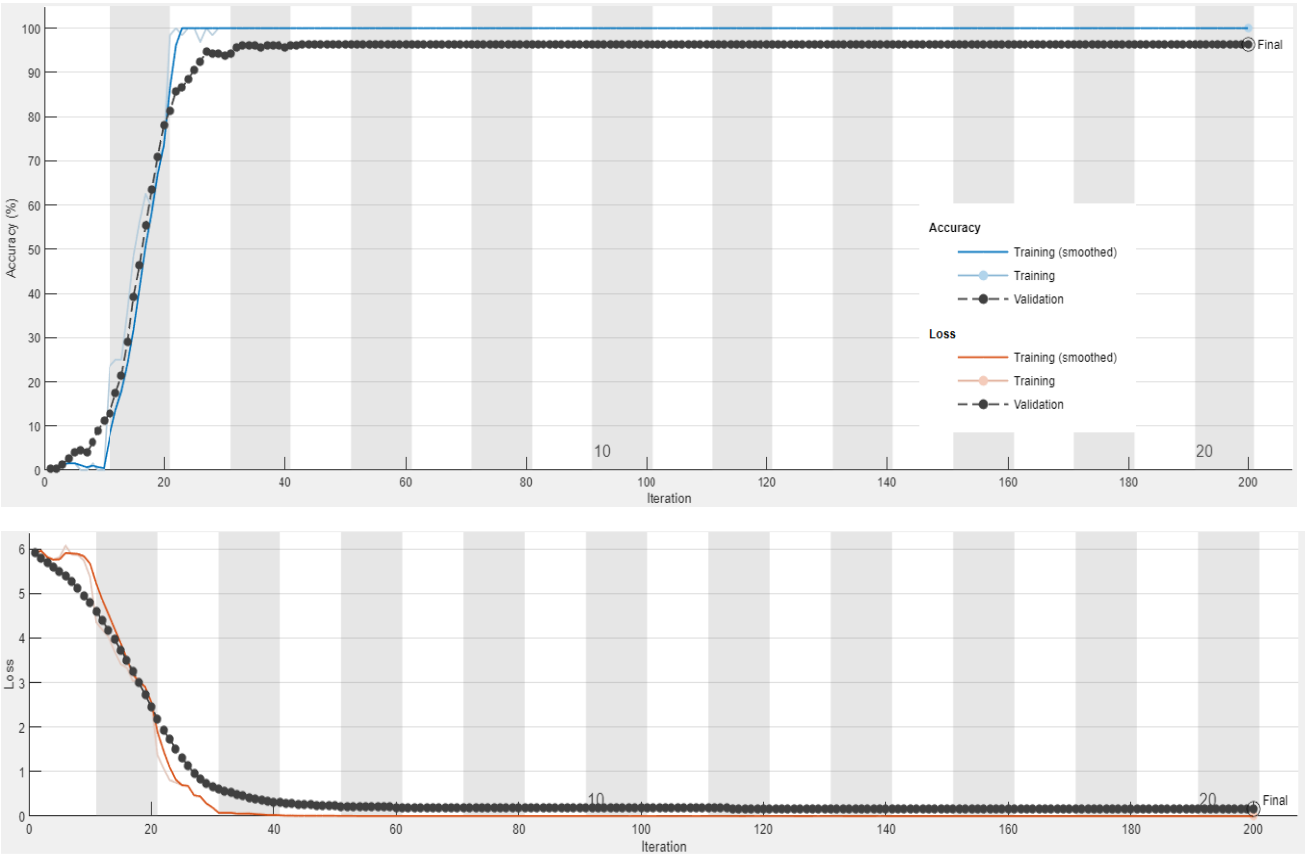


Figure III.14 : Visualisation des résultats pour une architecture a cinq couches de convolution, IITD.

- **Résultats :**
  - **Validation accuracy:** 96.43%, **Time:** 3 min 32 sec.

III.3.5.5 Analyse des résultats obtenus :

De manière globale l'augmentation du nombre de couches de convolution semble généralement améliorer la performance du modèle, avec une augmentation de la précision globale (Accuracy) et une diminution de l'erreur globale (A l'exception du résultat a trois couches de convolution qui ne suit pas vraiment cette règle). Cela suggère que l'ajout de couches de convolution permet au modèle d'extraire des caractéristiques plus complexes.

On peut voir aussi que l'ajout de couches de convolution supplémentaires entraine une convergence plus lente. Cela peut s'expliquer par le fait que l'augmentation du nombre de couches de convolution augmente la complexité du modèle, ce qui signifie qu'il y a plus de paramètres à ajuster.

En parlant de chiffre, la précision avec l'architecture a 5 couches est de 84.23% pour la base de données UBIRIS et de 96.43% pour la base de données IITD ce qui montre que de manière globale notre modèle a une bonne précision pour des temps très correctes qui sont de 2 min 52 sec et de 3 min 32 sec respectivement (plus l'architecture est complexe plus le temps augmente).

L'augmentation de la précision au fur et à mesure qu'on augmente les couches de convolution peut se justifier par le fait que chaque couche de convolution peut capturer des motifs et des structures spécifiques à différentes échelles, ce qui permet au modèle d'obtenir une représentation plus riche et discriminante des informations présentes dans les données. En augmentant le nombre de couches de convolution, le modèle devient capable de détecter des caractéristiques plus complexes ce qui se traduit par une meilleure précision.

III.3.6 **Etude des performances des différents classifieurs convolutifs**

Notre modèle utilise un réseau entièrement connecté avec une seule couche cachée pour la classification en fonction des caractéristiques extraites par cinq couches de convolution.

Nous connectons la dernière couche cachée, qui contient les caractéristiques extraites, à un classifieur automatique tel que SVM, KNN, Naive Bayes ou Decision Tree. Ces classifieurs utilise les caractéristiques issues de la dernière couche pour effectuer la classification finale.

III.3.6.1 Performance de CNN à cinq couches de convolution

Les résultats obtenus pour différentes métriques d'évaluation telles que la précision (Accuracy), l'erreur (Error), la sensibilité (Sensitivity), la spécificité (Specificity) et le coefficient Kappa sont résumés dans le tableau ci-dessous. Ces métriques fournissent une mesure de la performance du modèle pour chaque méthode de classification utilisée.

III.3.6.1.1 Métriques de performances

- **Base de données 1 : UBIRIS**

	Parametres	**Accuracy**	**Error**	**Sensitivity**	**Specificity**	**Kappa**
Naive Bayes	/	0.0788	0.9212	0.0788	0.9962	0.9910
Decision tree	MaxNumSplits = 4	0.0207	0.9793	0.0207	0.9959	0.9916
	MaxNumSplits = 8	0.0290	0.9710	0.0290	0.9960	0.9915
	MaxNumSplits = 12	0.0373	0.9627	0.0373	0.9960	0.9914
KNN	K = 3	0.5519	0.4481	0.5519	0.9981	0.9816
	K = 5	0.5768	0.4232	0.5768	0.9982	0.9805
	K = 7	0.5892	0.4108	0.5892	0.9983	0.9799
SVM	rbf	0.5975	0.4025	0.5975	0.9983	0.9795
	gaussian	**0.6598**	0.3402	0.6598	0.9986	0.9757
	linear	0.6100	0.3900	0.6100	0.9984	0.9788

	polynomial	**0.6722**	0.3278	0.6722	0.9986	0.9748

Tableau III.3 : Résultats des performances du modèle CNN en utilisant les diffèrent classifieurs, UBIRIS.

- **Base de données 2: IITD**

	Parametres	**Accuracy**	**Error**	**Sensitivity**	**Specificity**	**Kappa**
Naive Bayes	/	0.1429	0.8571	0.1429	0.9962	0.9896
Decision tree	MaxNumSplits = 4	0.0223	0.9777	0.0223	0.9956	0.9909
	MaxNumSplits = 8	0.0357	0.9643	0.0357	0.9957	0.9908
	MaxNumSplits = 12	0.0402	0.9598	0.0402	0.9957	0.9907
KNN	K= 3	0.9063	0.0938	0.9062	0.9996	0.9052
	K= 5	0.9509	0.0491	0.9509	0.9998	0.8190
	K= 7	0.8527	0.1473	0.8527	0.9993	0.9397
SVM	rbf	0.9464	0.0536	0.9464	0.9998	0.8341
	gaussian	**0.9688**	0.0312	0.9688	0.9999	0.7156
	linear	0.9509	0.0491	0.9509	0.9998	0.8190
	polynomial	0.9330	0.0670	0.9330	0.9997	0.8673

Tableau III.4 : Résultats des performances du modèle CNN en utilisant les diffèrent classifieurs, IITD.

III.3.6.1.2 Analyse des résultats obtenus :

D'après les résultats obtenus, on peut voir que le SVM (gaussian et polynomial) se démarque comme la méthode la plus performante en termes de précision, avec une valeur de 0.6722 pour la base de données 1 et de 0.9688 pour la base de données 2. Ensuite, vient le KNN avec une précision de 0.5975 (pour K=7) et de 0.9509 (pour K=5) respectivement. En revanche, le Naive Bayes et decision tree obtiennent des résultats nettement inférieurs pour les deux bases de données, indiquant une précision relativement faible.

En ce qui concerne la sensibilité, le SVM et le KNN se démarquent en ayant les valeurs les plus élevées, indiquant une bonne capacité à détecter les vrais positifs. Les autres méthodes affichent des valeurs de sensibilité nettement inférieures.

Dans l'ensemble, parmi les méthodes évaluées, le SVM présente la meilleure performance globale. Le KNN obtient également de bons résultats, tandis que le Naive Bayes et l'arbre de décision ont des performances relativement faibles.

III.3.6.2 Relation entre le nombre de couches de convolution et la précision

Dans cette étape on va procéder à l'évaluation des différents classifieurs en fonction de nombres de couches de convolution. Pour voir selon le nombre de couches lequel a la meilleure précision.

III.3.6.2.1 Métriques de performances

- **Base de données 1: UBIRIS**

	Parametres	**3 couches**	**4 couches**	**5 couches**
Naive Bayes	/	0.0871	0.0913	0.0954
Decision tree	MaxNumSplits = 4	0.0124	0.0166	0.0207
	MaxNumSplits = 8	0.0232	0.0290	0.0332
	MaxNumSplits = 12	0.0415	0.0332	0.0373
KNN	K= 3	0.7386	0.6100	0.5602
	K= 5	0.7759	0.5851	0.4689
	K= 7	0.7137	0.6307	0.5353

SVM	rbf	0.8050	0.6888	0.7054
	gaussian	0.8008	0.7220	0.6266
	linear	0.8091	0.6473	0.6183
	polynomial	0.8299	0.7344	0.6473

Tableau III.5 : Précision des classifieurs selon le nombre de couches, UBIRIS.

- **Base de données 2: IITD**

	Parametres	**3 couches**	**4 couches**	**5 couches**
Naive Bayes	/	0.1652	0.1518	0.0893
Decision tree	MaxNumSplits = 4	0.0134	0.0179	0.0223
	MaxNumSplits = 8	0.0182	0.0223	0.0312
	MaxNumSplits = 12	0.0226	0.0268	0.0491
KNN	K= 3	0.9911	0.9643	0.9018
	K= 5	0.9866	0.9732	0.9018
	K= 7	0.9955	0.9688	0.8839
SVM	rbf	0.9643	0.9777	0.9554
	gaussian	0.9732	0.9821	0.9643
	linear	0.9866	0.9732	0.9420
	polynomial	0.9821	0.9777	0.9911

Tableau III.6 : Précision des classifieurs selon le nombre de couches, IITD.

III.3.6.2.2 Analyse des résultats obtenus

Selon les résultats obtenus, on peut constater que, pour les deux bases de données, la précision de decision tree augmente en augmentant le nombre de couches de convolution. Cela indique que, pour certains classifieurs, la précision s'améliore en ajoutant des couches de convolution.

Pour le Naive Bayes, on peut voir que, dans le cas de la première base de données, la précision augmente avec le nombre de couches, tandis que pour la deuxième base de données, c'est plutôt le contraire.

En ce qui concerne le KNN, on peut observer que la précision maximale a été atteinte à la troisième couche. Cela suggère que, pour certains classifieurs, il existe un point optimal en termes de nombre de couches où la meilleure précision est obtenue.

Quant au SVM, on peut voir qu'avec la première base de données, il atteint sa précision maximale à la troisième couche, tandis que pour la deuxième base de données, il affiche des performances stables indépendamment du nombre de couches. Cela indique que sa précision peut varier selon les cas d'utilisation.

III.4 Conclusion

En conclusion, dans ce chapitre, nous avons segmenté et normalisé la base de données UBRIS, puis évalué les performances d'un modèle CNN sur les bases de données UBIRIS et IITD en combinaison avec les classifieurs SVM, KNN, Decision Tree et Naive Bayes. Les résultats ont démontré que les combinaisons avec SVM et KNN ont obtenu des résultats convaincants, offrant la meilleure précision en termes de classification. En revanche, les combinaisons avec Decision Tree et Naive Bayes ont montré des performances moins satisfaisantes. Ces conclusions fournissent des informations pertinentes pour sélectionner le classifieur optimal lors de l'utilisation d'un modèle CNN.

CONCLUSION GENERALE

En conclusion, ce mémoire contribue à l'avancement des connaissances dans le domaine de la biométrie de l'iris en mettant en évidence l'efficacité des réseaux de neurones convolutifs et en fournissant une évaluation des performances d'un modèle CNN sur deux bases de données différentes et avec différents classifieurs.

Ces résultats peuvent servir de base pour le développement de systèmes de reconnaissance de l'iris plus fiables et plus précis, ouvrant ainsi de nouvelles perspectives pour des applications telles que la sécurité, l'identification et la vérification des individus.

BIBLIOGRAPHIE

[1] K, Ghalem. (2018). « Authentification et identification de personnes par fusion d'informations provenant des images de l'iris de l'œil droit et de l'œil gauche », Thèse de doctorat, Université Mohamed Boudiaf.
[2] Introduction à la biométrie, académie de Poitiers.
[3] S, Djebbar. (2011). « Système de vérification de l'identité de personne par reconnaissance de l'iris », diplôme de Master, Syteme distribue, Université De Oum El Bouaghi.
[4] https://www.biometrie-online.net/biometrie/pourquoi-utiliser-la-biometrie, mais 2023.
[5] S, Guerfi. (2008). « Authentification d'individus par Reconnaissance de caractéristiques Biométriques liées aux visages 2D/3D », Thèse de Doctorat, Université D'Everly Val D'Essonne.
[6]https://www.thalesgroup.com/fr/europe/france/dis/gouvernement/biometrie/reconnaissance-faciale, mais 2023
[7] https://www.cnil.fr/fr/definition/reconnaissance-faciale, mais 2023
[8] N, Galy. (2010). « Etude d'un système complet de reconnaissance d'empreintes digitales pour un capteur microsystème à balayage », thèse de doctorat, Institut National Polytechnique de Grenoble.
[9] http://www.journaldunet.com/solutions/0611/061127-biometrie/4.shtml, mais 2023.
[10] F, Massicotte. (2007). « La biométrie, sa fiabilité et ses impacts sur la pratique de la démocratie libérale », mémoire de maitrise, université du Québec à Montréal.
[11] A, Benagga, L, Telib. (2016). « Reconnaissance des personnes basée sur l'empreinte de l'articulation de doigt », diplôme de master académique, université de Ouargla.
[12] S, Akrouf. (2011). « Une Approche Multimodale pour l'Identification du Locuteur », thèse de doctorat, UNIVERSITE FERHAT ABBAS.
[13] I, Doufa, A, Tadjine. (2022). « Identification biométrique des personnes par signature manuscrite », mémoire, Université 8Mai 1945.
[14] R, Mammeri, S, Laoudi. (2022). « La Reconnaissance Biométrique de la Démarche » Projet de fin d'étude pour l'obtention du diplôme de Master, Université Larbi Ben M'Hidi Oum El Bouaghui.
[15] M. Hashiyada. Development of biometric dna ink for authentication security. Tohoku Journal of Experimental Medicine, pages 109–117, 2004.
[16] A, Ghachoua, I, Kahlaoui. (2017). « Reconnaissance de personne en utilisation l'empreintes palmaires ».
[17] P.Bonazza. (2019). « Système de Sécurité Biométrique Multimodal par Imagerie,Dédié au Contrôle d'accès », Thèse de Doctorat, Informatique et Instrumentation de l'Image, Université de Bourgogne.
[18] C.L. Tisse, L. Torres, L. Martin, M. Robert. (2004). « Systèmes biométriques pour la vérification d'individu ».
[19] A, Moualdi. (2013). « Identification de personne par analyse l'image d'iris », Mémoire de Fin d'Études, Université SAAD DAHLAB.
[20] Z, Mida, S, Tliba. (2017). « La Reconnaissance Automatique Des Individus Basée Sur L'Iris En Utilisant L'algorithme De KNN », mémoire de fin d'étude, UNIVERSITE ECHAHID HAMMA LAKHDAR - EL OUED.
[21] H. Ohmaid, S. Eddarouich, A. Bourouhou, M. Timouyas. (2020). « Comparison between SVM and KNN classifiers for iris recognition using a new unsupervised neural approach in segmentation », 1National School of Computer Science and Systems Analysis (ENSIAS).

[22] A. HILAL, « Système d'identification à partir de l'image d'iris et détermination de la localisation des informations », Thèse de doctorat de l'UTT, Université Libanaise – Beyrouth, 2013.
[23] D, Lahouazi, L. Bouhzila. (2022). « Reconnaissance des iris par des réseaux de Neurone convolutifs », Mémoire de fin d'études, Université A. MIRA-Bejaia.
[24] https://www.ibm.com/en-en/topics/supervised-learning, mais 2023.
[25] J, A, Lopez Gonsalez. (2016). « Exploration des arbres de décision et des support vector machines en vue d'applications dans l'analyse de texte », Maîtrise en mathématiques et informa tique appliquées, Université du Québec à Trois-Rivières.
[26] F, Belarbi. (2019). « Optimisation des hyperparamètres pour la classification des virus », mémoire maitrise en informatique, université du Québec à Montréal.
[27] https://dataanalyticspost.com/Lexique/k-nearest-neighbours/
[28]https://www.ibm.com/topics/neuralnetworks#:~:text=Les%20r%C3%A9seaux%20de%20neurones%20refl%C3%A8tent,de%20l'apprentissage%20en%20profondeur, mais 2023.
[29]https://www.sage.com/fr-fr/blog/glossaire/deep-learning-definition/#:~:text=D%C3%A9finition-,Deep%20Learning%20%3A%20d%C3%A9finition,de%20r%C3%A9seaux%20de%20neurones%20artificiels.
[30] H, Bellahmer. (2020). « Implémentation et évaluation d'un modèle d'apprentissage automatique pour l'estimation de la valeur marchande de propriétés immobilières », Mémoire de fin d'études, Université MOULOUD MAMMERI de Tizi-Ouzou.
[31] A, Labiad. (2017). « Sélection des mots clés basée sur la classification et l'extraction des règles d'association », Maîtrise en mathématiques et informa tique appliquées, Université du Québec à Trois-Rivières.
[32] F, TEGHLIL, A, LAOUIRA. (2021). « Scanner à flux de fuite magnétique pour l'inspection des tôles ferromagnétiques », Mémoire de fin d'études, université Jijel.
[33] https://towardsdatascience.com/forward-propagation-in-neural-networks-simplified-math-and-code-version-bbcfef6f9250, mais 2023.
[34] S, Terai. (2021). « Mémoire Algorithme Hybride de Débruitage Image/Vidéo à base de Réseaux de Neurones Convolutionnels (CNN) », Mémoire de fin d'étude, Université B.B.A.
[35] K, Bouaichi, A, Bennai. (2022). « Détection d'intrusions et classification des attaques réseaux par les réseaux de neurones. », Projet de Fin d'Etudes, Université Abderrahmane Mira.
[36] https://towardsdatascience.com/gradient-descent-with-momentum-59420f626c8f, mais 2023.
[37] https://datascientest.com/qu-est-ce-qu-un-epoch-en-machine learning#:~:text=En%20r%C3%A9sum%C3%A9%2C%20l'epoch%20est,domaine%2C%20vous%20pouvez%20choisir%20DataScientest, mais 2023.
[38] Dr. Merzougui. Deep learning [diapositives]. http://staff.univ-batna2.dz/sites/default/files/merzougui_ghalia/files/support_de_cours_-deep_learning-chapitre3-cnn.pdf.
[39] https://openclassrooms.com/fr/courses/4470531-classez-et-segmentez-des-donnees-visuelles/5083336-decouvrez-les-differentes-couches-dun-cnn, mais 2023.
[40] https://yannicksergeobam.medium.com/comprendre-les-r%C3%A9seaux-de-neurones-convolutifs-cnn-d5f14d963714, mais 2023.
[41] https://www.datasciencetoday.net/index.php/fr/deep-learning/173-les-reseaux-de-neurones-convolutifs, mais 2023.
[42] https://www.researchgate.net/figure/Deep-Neural-Network-architecture_fig1_330120030, juin 2023.
[43] https://www.ibm.com/topics/gradient-descent, juin 2023.

[44] https://dataanalyticspost.com/Lexique/reseaux-de-neurones-recurrents/, juin 2023.
[45] http://iris.di.ubi.pt/ubiris1.html, juillet 2023.
[46] https://github.com/gugarosa/iris_recognition, juillet 2023.
[47] https://datascientest.com/perceptron, juillet 2023.
[48] https://www.lebigdata.fr/reseau-de-neurones-artificiels-definition, juillet 2023.
[49] https://www.researchgate.net/figure/Deep-Neural-Network-architecture_fig1_330120030, juillet 2023.
[50] https://www.freepng.fr/png-0glog8/, juillet 2023.
[51] https://towardsdatascience.com/implementation-of-rnn-lstm-and-gru-a4250bf6c090, juillet 2023.
[52] https://adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/, juillet 2023.
[54] S. Jayalakshmi, M. Sundaresan. (2023). « A surver on iris segmentation methods ».

[55] K. Humayan, R. Akthar, S. Shafiul, M. Moni. (2019). « A fast iris recognition system through optimum feature extraction ».

[56] https://www4.comp.polyu.edu.hk/~csajaykr/IITD/Database_Iris.htm, juillet 2023.

[57] https://www.nomidl.com/deep-learning/what-is-the-difference-between-sigmoid-and-softmax-activation-function/, septembre 2023.

Annexe

Code segmentation:

```
clc
clear all;
close all;
n_people = 1;
n_session = 1;
n_samples = 1;

for p = 1:n_people
    for q = 1:n_session
        for r = 1:n_samples
        % Reading input image
        s = strcat('UBIRIS/', num2str(p), '/', num2str(p), '_', num2str(q), '_',
num2str(r), '.jpg');
        I = imread(s);

% Convert the eye image to grayscale
grayImage = rgb2gray(I);

% Apply edge detection to the grayscale image
edgeImage = edge(grayImage, 'canny');

% Perform circular Hough transform to detect iris boundaries
[centers, radii, ~] = imfindcircles(edgeImage, [30 60], 'Sensitivity', 0.92);

% Select the circle with the largest radius as the iris boundary
[~, index] = max(radii);
irisCenter = centers(index, :);
irisRadius = radii(index);

% Create a binary mask of the iris region
[rows, cols] = size(grayImage);
[x, y] = meshgrid(1:cols, 1:rows);
mask = hypot(x - irisCenter(1), y - irisCenter(2)) <= irisRadius;

% Apply the binary mask to the grayscale image
irisImage = grayImage .* uint8(mask);

% Display the original image, iris boundary, and extracted iris
figure;
subplot(1, 3, 1);
imshow(I);
title('Original Image');

subplot(1, 3, 2);
imshow(I);
viscircles(irisCenter, irisRadius, 'EdgeColor', 'r');
title('Iris Boundary');

subplot(1, 3, 3);
imshow(I);
title('Extracted Iris');
             end
```

```
    end
end
```

Code normlisation:

```
clc
clear all
close all

n_people = 241;
n_session = 1;
n_samples = 5;

% Création d'un dossier pour sauvegarder les figures
mkdir('figures');

counter = 1; % Compteur pour suivre le nombre d'échantillons

for p = 1:n_people
    % Création du dossier pour chaque personne
    person_folder_name = sprintf('%03d', p);
    person_folder_path = fullfile('Figures', person_folder_name);
    mkdir(person_folder_path);

    for r = 1:n_samples
        % Lecture de l'image d'entrée
        s = strcat('UBIRIS/', num2str(p), '/', num2str(p), '_',
num2str(n_session), '_', num2str(r), '.jpg');
        I = imread(s);

        % Conversion en niveaux de gris et format double
        I = rgb2gray(I);
        I = im2double(I);
        original_I = I;

        % Variables d'entrée (rayon minimum et maximum)
        min_radius = 25;
        max_radius = 50;

        % Remplissage de tous les espaces vides pour permettre une détection
facile
        I = imcomplement(imfill(imcomplement(I), 'holes'));

        % Seuillage des éléments de l'image
        [radius, c] = size(I);
        [x, y] = find(I < 0.5);

        % Suppression de tous les pixels non minimaux locaux pour converger plus
rapidement
        for k = 1:size(x, 1)
            if (x(k) > min_radius) && (y(k) > max_radius) && (x(k) <= (radius -
min_radius)) && (y(k) < (c - min_radius))
                % Vérification du voisinage du pixel courant
                V = I((x(k) - 1):(x(k) + 1), (y(k) - 1):(y(k) + 1));
                min_pixel = min(min(V));
                if I(x(k), y(k)) ~= min_pixel
                    x(k) = 0;
                    y(k) = 0;
```

```
                end
            end
        end

        % Suppression de tous les pixels de bordure et redimensionnement de la
matrice des éléments
        k = find((x <= min_radius) | (y <= min_radius) | (x > (radius -
min_radius)) | (y > (c - min_radius)));
        x(k) = [];
        y(k) = [];
        n = size(x, 1);

        % Recherche de toutes les dérivées partielles
        for k = 1:n
            [b, radius, blur] = partial_derivative(I, [x(k), y(k)], min_radius,
max_radius, 'iris');
            max_b(x(k), y(k)) = b;
        end
        [x, y] = find(max_b == max(max(max_b)));

        % Recherche du centre de l'iris et de son rayon
        center_iris = find_center(I, min_radius, max_radius, x, y, 'iris');

        % Recherche du centre de la pupille et de son rayon
        center_pupil = find_center(I, round(0.1 * radius), round(0.8 * radius),
center_iris(1), center_iris(2), 'pupil');

        % Normalisation de la région de l'iris
        norm_I = normalize_iris(original_I, center_pupil, center_iris, 31, 180);

        % Création du code de l'iris
%         iris_code = iris_coding(norm_I, 0.5);

        % Sauvegarde de la figure dans le dossier correspondant
        figure_name = sprintf('%s_%d.bmp', person_folder_name, r);
        figure_path = fullfile(person_folder_path, figure_name);
        imwrite(norm_I, figure_path);

        counter = counter + 1; % Incrémentation du compteur

        % Clearing unnecessary data
        clear I original_I V max_b x y n
    end

    % Clearing unnecessary data
    clear person_folder_name person_folder_path
end
```

Code CNN, SVM:

```
close all
clear all
clc

NMCLS = 224
PTH = './IITD'
Sz = imsize('.\IITD\001\001_1.BMP')
```

```
% Chargement du dataset

disp('chargement du dataset')

imds = imageDatastore(PTH, 'LabelSource', 'foldernames',
'IncludeSubfolders',true);
    [trainingSet,validationSet, testSet] = splitEachLabel(imds,0.6, 0.2, 0.2
,'randomized');


% Affichage du nombre d'instances dans chaque catégorie

 layers = [

    imageInputLayer(Sz)
    convolution2dLayer(8,16,"Padding",1,"Stride",2)
    batchNormalizationLayer
    reluLayer('Name','relu_1')

    maxPooling2dLayer(4,"Stride",2)
    convolution2dLayer(4,16,"Padding",2)
    batchNormalizationLayer
    reluLayer('Name','relu_2')

    maxPooling2dLayer(4,"Stride",2)
    convolution2dLayer(3,32,"Padding",2)
    batchNormalizationLayer
    reluLayer('Name','relu_3')

    maxPooling2dLayer(3,"Stride",2)
    convolution2dLayer(3,32,"Padding",2)
    batchNormalizationLayer
    reluLayer('Name','relu_4')

    maxPooling2dLayer(3,"Stride",2)
    convolution2dLayer(3,64,"Padding",2)
    batchNormalizationLayer
     reluLayer('Name','relu_5')

    fullyConnectedLayer(NMCLS)
    softmaxLayer('Name','softmax_1')
    classificationLayer];

 disp ('break point 1')

 options = trainingOptions("sgdm", 'LearnRateSchedule', "piecewise", ...
    'LearnRateDropFactor', 0.2, ...
    'LearnRateDropPeriod', 5, ...
    'MaxEpochs',20, ...
    'ValidationData',validationSet, ...
    'ValidationFrequency',4, ...
    'MiniBatchSize', 64, ...
    'Plots', "training-progress")

disp ('break point 2 ')

convnet = trainNetwork(trainingSet,layers,options);

disp ('break point 3')
```

```
disp('inspecter la premiere couche')

convnet.Layers(1)

disp('inspecter la derniere couche')

convnet.Layers(end)

% Display conv layer

w1 = convnet.Layers(2).Weights;
w1 = mat2gray(w1);
w1 = imresize(w1,5);
figure
montage(w1)
title('Premiere couche de convolution')

featureLayer = 'softmax_1';
trainingFeatures = activations(convnet, trainingSet, featureLayer, ...
    'MiniBatchSize', 64, 'OutputAs', 'columns');

% Get training labels from the trainingSet
trainingLabels = trainingSet.Labels;

% Train multiclass SVM classifier
params = templateSVM('KernelFunction','polynomial',
'KernelScale',6,'BoxConstraint',1);

classifier = fitcecoc(trainingFeatures, trainingLabels, ...
    'Learners', params, 'Coding', 'onevsall', 'ObservationsIn', 'columns');

% Extract test features using the CNN
testFeatures = activations(convnet, testSet, featureLayer, ...
    'MiniBatchSize', 32, 'OutputAs', 'columns');

% Pass CNN image features to trained classifier
predictedLabels = predict(classifier, testFeatures, 'ObservationsIn', 'columns');

% Get the known labels
testLabels = testSet.Labels;

% Confusion Matix
confMat = confusionmat(testLabels, predictedLabels);

[Result,RefereceResult]=confusion.getValues(confMat);

Result
```

Code CNN, decision tree:

```
close all
clear all
clc

NMCLS = 241
PTH = './UBIRIS_1'
```

```
Sz = imsize('.\UBIRIS_1\001\001_1.BMP')

% Chargement du dataset

disp('chargement du dataset')

imds = imageDatastore(PTH, 'LabelSource', 'foldernames',
'IncludeSubfolders',true);
    [trainingSet,validationSet, testSet] = splitEachLabel(imds,0.6, 0.2, 0.2 ...
        ...%399,49,49 ...
         ...%100,10,10 ...
        ,'randomized');

 layers = [

    imageInputLayer(Sz)
    convolution2dLayer(8,16,"Padding",1,"Stride",2)
    batchNormalizationLayer
    reluLayer('Name','relu_1')

    maxPooling2dLayer(4,"Stride",2)
    convolution2dLayer(4,16,"Padding",2)
    batchNormalizationLayer
    reluLayer('Name','relu_2')

    maxPooling2dLayer(4,"Stride",2)
    convolution2dLayer(3,32,"Padding",2)
    batchNormalizationLayer
    reluLayer('Name','relu_3')

    maxPooling2dLayer(3,"Stride",2)
    convolution2dLayer(3,32,"Padding",2)
    batchNormalizationLayer
    reluLayer('Name','relu_4')

    maxPooling2dLayer(3,"Stride",2)
    convolution2dLayer(3,64,"Padding",2)
    batchNormalizationLayer
    reluLayer('Name','relu_5')

    fullyConnectedLayer(NMCLS)
    softmaxLayer('Name','softmax_1')
    classificationLayer
];

 disp ('break point 1')

 options = trainingOptions("sgdm", 'LearnRateSchedule', "piecewise", ...
    'LearnRateDropFactor', 0.2, ...
    'LearnRateDropPeriod', 5, ...
    'MaxEpochs',10, ...
    'ValidationData',validationSet, ...
    'ValidationFrequency',4, ...
    'MiniBatchSize', 64, ...
    'Plots', "training-progress")

disp ('break point 2 ')
convnet = trainNetwork(trainingSet,layers,options);
```

```
disp ('break point 3')
disp('inspecter la premiere couche')
convnet.Layers(1)
disp('inspecter la derniere couche')

featureLayer = 'softmax_1';
trainingFeatures = activations(convnet, trainingSet, featureLayer, ...
    'MiniBatchSize', 64, 'OutputAs', 'columns');

% Get training labels from the trainingSet
trainingLabels = trainingSet.Labels;

% Train multiclass BT
classifier = fitctree(trainingFeatures', trainingLabels, 'MaxNumSplits', 12)

% Extract test features using the CNN
testFeatures = activations(convnet, testSet, featureLayer, ...
    'MiniBatchSize', 32, 'OutputAs', 'columns');

% Pass CNN image features to trained classifier
predictedLabels = predict(classifier, testFeatures);

% Get the known labels
testLabels = testSet.Labels;

% Tabulate the results using a confusion matrix.
confMat = confusionmat(testLabels, predictedLabels);
[Result,RefereceResult]=confusion.getValues(confMat);
Result
```

Code CNN, Naive Bayes:

```
close all
clear all
clc

NMCLS = 224
PTH = './IITD'
Sz = imsize('.\IITD\001\001_1.BMP')

% Chargement du dataset

disp('chargement du dataset')

imds = imageDatastore(PTH, 'LabelSource', 'foldernames',
'IncludeSubfolders',true);
    [trainingSet,validationSet, testSet] = splitEachLabel(imds,0.6, 0.2, 0.2 ...
        ,'randomized');

 layers = [

    imageInputLayer(Sz)
    convolution2dLayer(8,16,"Padding",1,"Stride",2)
    batchNormalizationLayer
    reluLayer('Name','relu_1')

    maxPooling2dLayer(4,"Stride",2)
    convolution2dLayer(4,16,"Padding",2)
```

```
    batchNormalizationLayer
    reluLayer('Name','relu_2')

    maxPooling2dLayer(4,"Stride",2)
    convolution2dLayer(3,32,"Padding",2)
    batchNormalizationLayer
    reluLayer('Name','relu_3')

    maxPooling2dLayer(3,"Stride",2)
    convolution2dLayer(3,32,"Padding",2)
    batchNormalizationLayer
    reluLayer('Name','relu_4')

    maxPooling2dLayer(3,"Stride",2)
    convolution2dLayer(3,64,"Padding",2)
    batchNormalizationLayer
    reluLayer('Name','relu_5')

    fullyConnectedLayer(NMCLS)
    softmaxLayer('Name','softmax_1')
    classificationLayer];

 disp ('break point 1')

 options = trainingOptions("sgdm", 'LearnRateSchedule', "piecewise", ...
    'LearnRateDropFactor', 0.2, ...
    'LearnRateDropPeriod', 5, ...
    'MaxEpochs',10, ...
    'ValidationData',validationSet, ...
    'ValidationFrequency',4, ...
    'MiniBatchSize', 64, ...
    'Plots', "training-progress")

disp ('break point 2 ')

convnet = trainNetwork(trainingSet,layers,options);
disp ('break point 3')
disp('inspecter la premiere couche')
convnet.Layers(1)
disp('inspecter la derniere couche')
convnet.Layers(end)

% Display conv layers

w1 = convnet.Layers(2).Weights;
w1 = mat2gray(w1);
w1 = imresize(w1,5);
figure
montage(w1)
title('Premiere couche de convolution')


featureLayer = 'softmax_1';
trainingFeatures = activations(convnet, trainingSet, featureLayer, ...
    'MiniBatchSize', 64, 'OutputAs', 'columns');

% Get training labels from the trainingSet
trainingLabels = trainingSet.Labels;
```

```
% Train NB
classifier = fitcnb(trainingFeatures', trainingLabels)

% Extract test features using the CNN
testFeatures = activations(convnet, testSet, featureLayer, ...
    'MiniBatchSize', 32, 'OutputAs', 'columns');

% Pass CNN image features to trained classifier
predictedLabels = predict(classifier, testFeatures);

% Get the known labels
testLabels = testSet.Labels;

% Tabulate the results using a confusion matrix.
confMat = confusionmat(testLabels, predictedLabels);
[Result,RefereceResult]=confusion.getValues(confMat);
Result
```