وزارة التعليم العالي والبحث العلمي

BADJI MOKHTAR- ANNABA UNIVERSITY

UNIVERSITE BADJI MOKHTAR ANNABA

جامعة باجي مختار-عنابة

Année : 2019

Faculté: Sciences de l'ingéniorat
Département: Electronique

MEMOIRE
Présenté en vue de l'obtention du diplôme de : MASTER

Intitulé

Reconnaissance d'iris par les réseaux de neurones convolutionnels

Domaine : Sciences et Technologie

Filière : Télécommunications

Spécialité : Systèmes des télécommunications

Par :
BELABEND Meriem

DEVANT Le JURY

Président	**: BOUTERAA N.**	**MCA UNIVERSITE BADJI MOKHTAR ANNABA**
Directeur de mémoire	**: BOUKARI K.**	**MCA UNIVERSITE BADJI MOKHTAR ANNABA**
Examinateurs	**: MESSADEG D.**	**Prof. UNIVERSITE BADJI MOKHTAR ANNABA**
	: AMARA F.	**MCB UNIVERSITE BADJI MOKHTAR ANNABA**

Remerciements

Je tiens à exprimer toute ma reconnaissance à ma directrice de mémoire, Madame BOUKARI Karima. Je la remercie de m'avoir encadré, orienté, aidé et conseillé.

J'adresse mes sincères remerciements à tous les professeurs, intervenants et toutes les personnes qui par leurs paroles, leurs écrits, leurs conseils et leurs critiques ont guidé mes réflexions et ont accepté de me rencontrer et de répondre à mes questions durant mon cursus.

Je remercie mes très chers parents, Louisa et Naser, qui ont toujours été là pour moi. Je remercie mes sœurs Yasmine et Selma, pour leurs encouragements.

Je voudrais exprimer ma reconnaissance envers mes amis et collègues qui m'ont apporté leur soutien moral et intellectuel tout au long de ma démarche.

Enfin, je remercie mon époux qui a toujours été là pour moi. Son soutien inconditionnel et ses encouragements ont été d'une grande aide.

À tous ces intervenants, je présente mes remerciements, mon respect et ma gratitude.

Résumé

La biométrie par l'iris représente une des méthodes les plus fiables et les plus performantes pour son faible taux d'erreurs malgré les contraintes lors de l'acquisition de l'iris.

En dépit des difficultés à obtenir une image de bonne qualité, dans ce travail nous allons essayer de faire une reconnaissance de l'iris, plus précisément extraire un vecteur caractéristique pour faire la classification des images de deux manières à savoir ; la classification avec un modèle de réseaux de neurones convolutionnels près-entrainé grâce à un apprentissage par transfert puis la caractérisation avec réseaux de neurones convolutionnels suivie d'une classification avec les machines à vecteurs de support.

Enfin nous allons utiliser différentes bases de données afin de tester les performances de ces réseaux.

Abstract

Iris biometry represents one of the most reliable and efficient methods for its low error rate, despite the constraints involved in acquiring the iris.
Despite the difficulties in obtaining a good quality image, in this work we will try to make an iris recognition, more precisely to extract a characteristic vector to make the classification of the images in two ways namely; classification with a near-trained convolutional neural network model through transfer learning and characterization with convolutional neural networks followed by classification with support vector machines.
Finally we will use different databases to test the performance of these networks.

ملخص

تمثل القزحية الحيوية واحدة من أكثر الطرق موثوقية وفعالة لمعدل الخطأ المنخفض ، على الرغم من القيود التي ينطوي عليها الحصول على القزحية.

على الرغم من الصعوبات في الحصول على صورة جيدة النوعية ، سنحاول في هذا العمل التعرف على قزحية العين ، وبشكل أكثر دقة لاستخراج ناقل متجه لوضع تصنيف الصور بطريقتين هما: التصنيف مع نموذج شبكة عصبية تلافيفيه مدربة تقريبًا من خلال نقل التعلم والتوصيف مع شبكات عصبية تلافيفيه متبوعًا بالتصنيف مع أجهزة ناقل الدعم.

أخيرًا ، سوف نستخدم قواعد بيانات مختلفة لاختبار أداء هذه الشبكات.

Liste des abréviations

ADN : Acide Désoxyribonucléique

CNN: convolutional neural network (Les réseaux de neurones convolutionnels)

SVM : Support Vector Machine (Les machines à vecteurs de support)

BDD : Base de données

CLAHE: contrast limited adaptive histogram equalization, (égalisation d'histogramme adaptatif avec limitation de contraste)

Liste des figures :

- **Figure 1.1 : La zone de l'iris dans l'œil**
- **Figure 1.2 : Images différentes d'Iris**
- **Figure 1.3 : Image acquise en lumière visible, conditions normales d'acquisition.**
- **Figure 1.4 : Image acquise en infrarouge**
- **Figure 2.1 : Représentation d'une image œil humain vs ordinateur**
- **Figure 2.2 : exemple d'un CNN organisé en deux parties**
- **Figure 2.3 : exemples de filtres à plusieurs niveaux du CNN**
- **Figure 2.4 : Application de la convolution sur une image**
- **Figure 2.5 : Application de ReLU à une image**
- **Figure 2.6 : Application du max pooling a une image**
- **Figure 2.7 : Mise à plat des images finales en sortie**
- **Figure 2.8 : Exemple des couches entièrement connectées**
- **Figure 2.9 : application de la couche dropout**
- **Figure 3.1 : Avant et après l'élimination des points blancs de l'iris**
- **Figure 3.2 : Exemple de l'application d'un filtre médian 3*3**
- **Figure 3.3 : Application d'un filtre médian 3*3**
- **Figure 3.4 : Détection de l'iris par la méthode de Canny**
- **Figure 3.5 : image segmentée avec reflet**
- **Figure 3.6 : image segmentée sans reflet**
- **Figure 3.7 : Détermination du rayon et du centre de la pupille**
- **Figure 3.8 : Détermination du rayon de l'iris**
- **Figure 3.9 : localisation de l'iris**
- **Figure 3.10 : Transformation en Pseudo-Polaire**
- **Figure 3.11 : Normalisation de l'iris**
- **Figure 3.12 : image d'iris améliorée par l'histogramme de CLAHE**
- **Figure 3.14 : La caméra d'iris à main utilisée pour la collecte de CASIA-Iris-Lamp**
- **Figure 3.15 : Exemples des images d'iris obtenues à partir d'une base de données CASIA-Iris-Lamp**
- **Figure3.16 : Image de la BDD1**
- **Figure3.17 : Image de la BDD2**
- **Figure 3.18 : Image de la BDD3**

- **Figure3.19 : Image de la BDD4**
- **Figure 3.20 : Image de la BDD5**
- **Figure 3.21 : exemples des images dans un cas de contrainte**
- **Figure 3.22 : réutilisation d'un réseau pré entrainé**
- **Figure 3.23 : visualisation de l'architecture du réseau AlexNet**
- **Figure 3.24 : partie architecture du réseau GoogleNet**
- **Figure 3.25 : changement des trois dernières couches du modèle AlexNet**
- **Figure 3.26 : changement des trois dernières couches du modèle GoogleNet**
- **Figure 3.27 : Exemple d'une image de la BDD brute**
- **Figure 3.28 : résultat de la BDD 5 pour une classification avec les modèles**
- **Figure 3.29 : échantillons des images de la BDD 5**

Liste des tableaux :

- **Tableau 3.1 Taux d'apprentissage pour les différentes bases de données pour une classification avec les modèles AlexNet et GoogleNet**
- **Tableau 3. 2 taux d'apprentissage pour les différentes BDD en changeant MiniBatchSize pour une classification avec le modèle AlexNet**
- **Tableau 3. 3 taux d'apprentissage pour les différentes BDD en changeant MiniBatchSize pour une classification avec les modèles GoogleNet**
- **Tableau 3. 4. taux d'apprentissage pour les différentes BDD en changeant MaxEpochs pour une classification avec le modèle AlexNet**
- **Tableau 3. 5. taux d'apprentissage pour les différentes BDD en changeant MaxEpochs pour une classification avec le modèle GoogleNet**
- **Tableau 3. 6. taux d'apprentissage pour les différentes BDD avec l'extraction des caractéristiques pour une classification avec les SVM**

Sommaire :

Sommaire 1
Introduction générale 3
CHAPITRE I : Généralités sur la biométrie 4
I.1. Introduction 5
I.2. Biométrie de l'iris 5
I.3.Conditions d'acquisition de l'iris 6
I.4. Architecture d'un système de reconnaissance d'iris 8
I.5 Conclusion 8
CHAPITRE II : Présentation des réseaux neuronaux convolutionnels 9
II.1.Introduction 10
II.2.Principe général 10
II.2.1. L'opération de convolution 11
II.2.1.1 Les filtres 11
II.2.1.2.Calcul de la convolution 12
II.2.2. La couche ReLU (rectification linéaire) 13
II.2.3.cross channel normalization 13
II.2.4. Le max pooling 14
II.2.5. Le flattening (ou mise à plat) 14
II.2.6. Les couches entièrement connectées 15
II.2.7. La couche dropout 16
II.2.8 Couche Prob « «softmax » 16
II.2.9. Mise bout à bout du réseau neuronal convolutionnel 17
II.3. Conclusion 17
CHAPITRE III : Réalisation et discussions 18
III.1. Introduction 19
III.2. Prétraitement des données 19
III.2.1. élimination des points blancs d'iris 19
III.2.2- Filtrage médian 19
III.2.3 la localisation de l'iris (segmentation) 20
III.2.4.Normalisation de la région de l'iris 23
III. 3. Post-traitement (CLAHE) 24
III.4 Bases de données 25

III.5.Apprentissage par transfert 28
III.5.1. Chargement des données 28
III.5.2. Chargement du modèle prédéfini 29
III.5.2.1.AlexNet 29
III.5.2.2.GoogleNet 29
III.5.3. Remplacement les couches finales 31
III.5.4. Entrainement du réseau 31
III.6.1 Classification avec un modèle CNN près-entrainé 32
III.6.2. Classification avec SVM 35
III.6.2.1. Extraction des caractéristiques Image 36
III.6.2.2. Ajustement du classifieur 36
III.6.2.3. Classification des Images de Test 36
III.7. Conclusion 38
IV. Conclusion générale 39
V. Références 40
VI. Annexe 42

Introduction générale:

La vision informatique évolue rapidement et occupe une place importante au quotidien. L'expansion de ces systèmes s'explique par l'accroissement permanent des applications dans différents domaines tels que : la télédétection, la médecine, l'automobile, ou encore la sécurité et notamment en biométrie.

Chaque être humain est caractérisé pas ces traits, couleur de peau, couleur de yeux, de cheveux...etc. la biométrie est la technique qui permet justement de reconnaître des personnes à partir de leurs caractéristiques physiques et comportementales. Mais aussi grâce à des parties du corps humain.
Après la publication de plusieurs travaux stipulant qu'il était possible d'utiliser les empreintes digitales pour identifier des personnes, ce procédé émergea en Asie, en Afrique du Sud et en Europe. En Inde, Edward Henry développa une méthode robuste de reconnaissance à base d'empreintes digitales. Sir Francis Galton publia des travaux détaillés sur la reconnaissance par l'empreinte digitale basés sur des caractéristiques particulières de la texture de l'empreinte.
D'autres moyens permettant une reconnaissance de l'individu existent, l'empreinte, l'iris, le visage et la forme de la main, appelés 'modalités biométriques'. On peut aussi citer l'exemple de la veine de la main et de la rétine. Pour ce qui est des modalités comportementales, on peut citer la signature (dynamique ou statique),
Dans ce mémoire, nous nous sommes intéressés uniquement aux problématiques liées à la modalité de l'iris [1]

Dans notre travail nous allons utiliser les réseaux neuronaux convolutionnels afin de faire une reconnaissance de l'iris car aujourd'hui, une grande attention est accordée aux algorithmes d'apprentissage des fonctionnalités et à ces réseaux. Dans cet algorithme, l'image est directement transmise aux réseaux de neurones de convolution, puis l'algorithme extrait les meilleures caractéristiques de cette image. Nous proposons un système de reconnaissance de l'iris dans lequel les caractéristiques sont extraites du modèle CNN pour cela nous avons choisi deux modèles près-entrainés ; AlexNet et GoogleNet, et pour la tâche de classification, nous allons faire une classification par le modèle même et par Les machines à vecteurs de support ou séparateurs à vaste marge (SVM) multi-classes.
Le développement du système de reconnaissance de l'iris proposé est présenté en trois parties: l'étape de prés-traitement, post-traitement l'étape d'extraction de caractéristiques et l'étape de classification.
L'étape de classification seras assurée par les CNN à travers deux modèles prés-définis, et on va appliquer ces modèles sur les bases de donnée CASIA-IRIS V4.

CHAPITRE I
Généralités sur la biométrie

I.1 Introduction :

Aujourd'hui, Nous avons plusieurs secteurs dans lesquels la biométrie est utilisée dont trois domaines les plus importants ; l'identification judicaire, l'administration (papiers d'identité…) et les contrôle d'accès, que ce soit dans des établissements publics ou privés.

Comme nous l'avons cité plus haut la biométrie est la technique qui permet la reconnaissance des personnes à travers des caractéristiques physiques ou comportementales,

Actuellement l'avancement des technologies a fait que la reconnaissance des personne se fait par une reconnaissance électronique, c'est-à-dire que l'authentification ne se fait pas par une carte d'identité ou un code de contrôle d'accès mais par une reconnaissance des partie du corps dont la reconnaissance de l'iris qui est une des technologies les plus fiables et les plus performantes pour identifier une personne.

Dans ce chapitre nous commencerons par la présentation de la biométrie de l'iris. Puis nous détaillerons les différentes étapes composant un système complet de reconnaissance d'iris.

I.2. Biométrie de l'iris

L'iris est un élément de l'œil qui se caractérise par un nombre incalculable de signes distinctifs. C'est pourquoi son utilisation est plus fiable que celui de l'empreinte digitale. De plus, le procédé de reconnaissance via l'iris est très simple : il suffit de placer l'œil devant un objectif pour le scan.

La reconnaissance par l'iris requiert plus de paramètres que les autres types d'identifications. Le résultat est si fiable qu'une identification de l'utilisateur n'est plus nécessaire. Avec une structure complexe, l'iris est unique pour chaque personne et il est quasi impossible que deux personnes puissent avoir le même.

Dans les années 30 la biométrie de l'IRIS a été proposée mais ça n'est que vers les années 90 que le brevet a été déposé, cette technique permet de quantifier l'utilisation des différentes caractéristiques physiques uniques de chaque personne. A la fois discrète et performante, la biométrique liée à l'œil est devenue une méthode de sécurisation de haute qualité. Ce qui fait que cette technique est fiable est que selon les estimations de Daugman il ya une chance de 1/10^72 de trouver deux iris similaires.

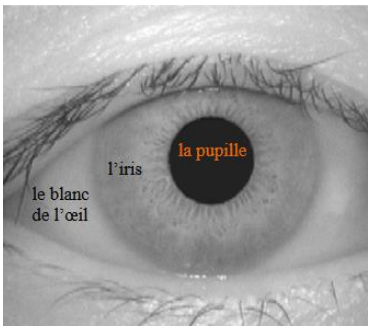


Figure 1.1.La zone de l'iris dans l'œil

Contrairement à ce que l'on croit le motif de l'iris n'est pas relié aux gènes (ADN) ; ce n'est pas en fonction des gènes des parents que le motif de l'iris est formé contrairement à la couleur des yeux. Donc deux individus, même s'ils sont parents, peuvent avoir la même couleur, mais jamais le même motif. Cette formation est chaotique, elle génère donc des motifs possédant une forte variabilité.

L'organe iridien est relativement à l'abri des lésions. S'agissant d'un tissu interne, l'iris est protégé par la cornée et l'humeur aqueuse. Étant donné que ces deux barrières sont transparentes, l'iris peut être facilement identifié à plus d'un mètre. On peut donc facilement photographier l'iris, qui n'est pas pourtant exposé à d'éventuels dommages.[2]

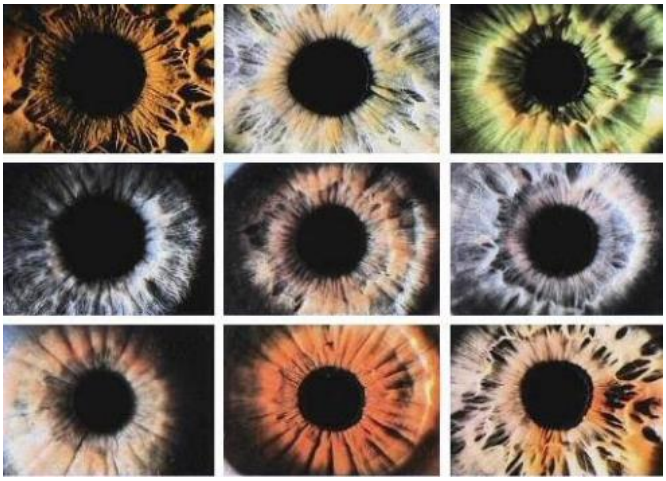


Figure 1.2:Images différentes d'Iris

I.3.Conditions d'acquisition de l'iris :

La capture de l'iris n'est pas une tâche facile car sa très petite taille qui est souvent obscurcie par les cils, les paupières, les lentilles ou des réflexions de la lumière et qu'elle varie suivant la lumière dans laquelle il se trouve ou l'état de fatigue, localisée derrière la cornée qui constitue un miroir hautement réfléchissant. Toutes ces caractéristiques en font un objet très difficile à photographier. Premièrement, l'iris est sombre, il faut donc l'éclairer mais en même temps l'iris est sensible à la lumière et de fortes illuminations peuvent engendrer des malaises chez l'utilisateur. Deuxièmement, l'iris est un objet de petite taille (environ 1cm de diamètre) il est alors impératif d'utiliser des focales très puissantes ou de rapprocher l'iris de l'objectif mais sans risque, car dans ce dernier cas, on rapprocherait l'iris de la source d'illumination ce qui pourrait nuire l'œil de l'individu ; Enfin l'iris est une surface dite Lambertienne , c'est-à dire une surface qui réfléchit la lumière dans toutes les directions car derrière la cornée se situe un miroir hautement réfléchissant. Ces deux dernières caractéristiques font que si aucune technique particulière n'est employée, l'iris photographié sera couvert par des reflets de toutes les sources lumineuses présentes dans l'environnement d'acquisition. Il est à noter que différentes longueurs d'ondes de l'illumination engendrent un pouvoir de pénétration de la cornée différent et donc un taux de réflexion sur l'iris plus grand lui aussi.

La première solution serait d'utiliser un dispositif d'illumination en lumière visible. La figure (1.3) montre un iris acquis sans condition particulière. Il est clair qu'avec ce genre d'image aucun traitement de reconnaissance n'est possible. Les réflexions proviennent de toutes les sources lumineuses présentes dans la salle d'acquisition ; tube néon, fenêtre, écran du PC...[3]

Figure I.3.Image acquise en lumière visible, conditions normales d'acquisition. (SourceDaugman)

Les chercheurs ont alors pensé à une autre méthode qui est l'utilisation d'une ou plusieurs sources infrarouge comme illuminateur puissant. La Figure 1l-4 montre le même iris que celui de la Figure I-3 mais acquis avec une lumière infrarouge. Ce pendent l'infrarouge possède deux avantages majeurs sur la lumière visible. Premièrement, la lumière est invisible, l'utilisateur ne sera pas aussi gêné qu'en lumière visible par une puissante illumination. Le deuxième avantage est que le proche infrarouge possède un pouvoir de pénétration de la cornée qui est largement plus grand que celui de la lumière visible et il est ainsi possible de récupérer une richesse de texture supérieure à celle obtenue en lumière visible surtout pour les iris sombres. Le seul inconvénient possible vis-à-vis de la lumière visible est l'impact de l'utilisation du proche infrarouge sur la sécurité de l'œil [4]

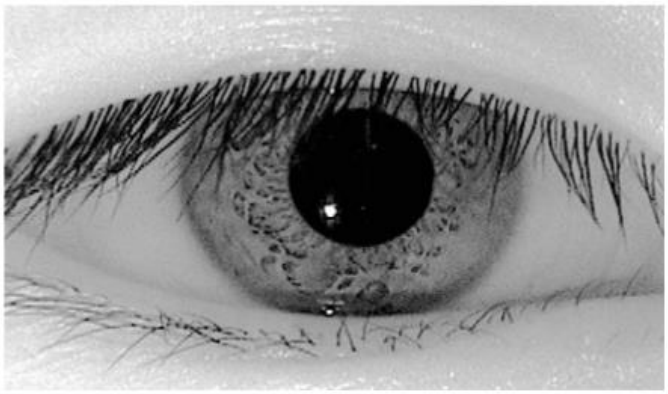


Figure 1.4.Image acquise en infrarouge (source Daugman)

L'occlusion, l'éclairage et la réflexion de la lumière de l'iris ainsi que la focalisation de l'image, sont des facteurs qui affectent la qualité de l'image car celle-ci doit avoir une résolution et une qualité minimale pour garantir l'acquisition des caractéristiques de l'iris nécessaire pour la représentation et l'identification.

En effet, l'occlusion de l'iris est provoquée par la fermeture totale ou partielle de la paupière supérieure ou inférieure. En plus les cils peuvent facilement interférer avec la région d'iris. D'autre part, un éclairage non optimisé peut résulter en une image sombre, saturée ou comportant beaucoup de réflexions.

Un autre problème que nous ajoutons qui est l'angle de l'illumination Nous pouvons trouver des images qui ne sont malheureusement pas parfaite en terme d'illumination qui peuvent

affecter la détection, Tous ces éléments sont considérés comme du bruit non significatif obtenus lors de l'acquisition. Ils limitent la région visible de l'iris, réduisent la qualité de l'image et influent indirectement sur la décision de reconnaissance de l'iris. Une segmentation efficace de ces régions est donc nécessaire. [4]

I.4. Architecture d'un système de reconnaissance d'iris

Une fois l'image de l'iris acquise, un système d'iris peut être composé de plusieurs modules comprenant:

- **La segmentation de l'iris :**
  L'image acquise doit être segmentée car elle ne comporte pas uniquement l'information de l'iris, c'est-à-dire que nous devons isoler cette information du reste de l'image et plus précisément isoler l'iris du blanc de l'œil et des paupières,
- **La normalisation de l'iris**.
  Transformer la région de l'iris caractérisée par des contours paramétrés en une bande de taille invariante, effectuer le passage d'un système cartésien à un système pseudo-polaire.
- **Extraction des caractéristiques**
  Le but de cette étape est d'extraire une représentation caractéristique et discriminante de l'iris et de produire un gabarit permettant la vérification dans l'étape de comparaison des motifs**.**
- **Comparaison des motifs** La reconnaissance de l'iris implique la mise en correspondance entre deux codes d'iris. Un score de dissimilitude est calculé pour caractériser le degré de ressemblance ou non entre deux codes d'iris [5]

I.5. Conclusion :

L'apparition des systèmes multi-biométriques a nettement amélioré la performance des systèmes d'identification. C'est seulement une question de temps avant que la biométrie puisse s'intégrer dans le tissu même de la société et s'imposer dans notre vie quotidienne, dans ce chapitre nous avons vu le développement de la biométrie en général mais surtout celle lié à l'iris, Nous avons aussi vu la raison pour laquelle cette méthode est considérée comme une des méthodes les plus fiables et des plus compétentes néanmoins nous devons choisir des critères pour l'acquisition de l'iris afin de pouvoir utiliser les images pour un traitement de reconnaissance

CHAPITRE II
Présentation des réseaux neuronaux convolutionnels

II.1. Introduction :

Lorsque l'être humain regarde dans une direction quelconque il analyse son environnement et passe par la reconnaissance des éléments ; trouver les autres personnes, identifier les voitures, les animaux…

Pour un ordinateur cette tâche étais assez difficile, l'approche de ces réseaux inspirés de notre œil (plus particulièrement du fait que certains neurones de notre aire visuelle ne réagissent qu'aux bordures verticales et d'autres aux horizontales/diagonales) a ouvert de **nombreuses applications**, que ce soit en imagerie médicale, véhicules autonomes, reconnaissance faciale, et même analyse de textes.

Dans ce chapitre nous allons parler des réseaux neuronaux convolutionnels, et voir leurs principes de fonctionnement

II.2. Principe général :

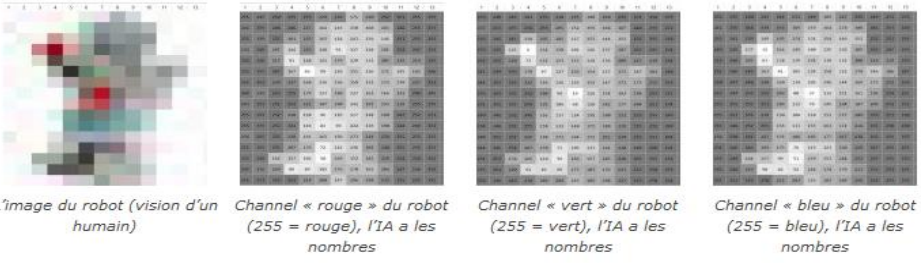
Les réseaux de neurones convolutionnels sont à ce jour les modèles les plus performants pour classer des images. Désignés par l'acronyme CNN, de l'anglais Convolutional Neural Network, le CNN compare les images fragment par fragment. Les fragments qu'il recherche sont appelés les caractéristiques. En trouvant des caractéristiques approximatives qui se ressemblent à peu près dans 2 images différentes, le CNN est bien meilleur à détecter des similitudes que par une comparaison entière image à image.

Chaque caractéristique est comme une mini-image—i.e. un petit tableau de valeurs en 2 Dimensions. Les caractéristiques rassemblent les aspects les plus communs des images.[6]

Contrairement à nous les êtres humains qui voyons les formes et les couleurs l'ordinateur ne voit que des nombres organisés comme suit :

- 16×16 pixels pour l'image du robot (ci-après), où le premier 16 est la largeur de l'image et le second est la hauteur
- **3 couches** de ces 16×16 pixels, chacune correspondante à une coloration :
    - rouge (r pour red)
    - vert (g pour green)
    - bleu (b pour blue)

Ce qui nous donne ici une **matrice 3D** de taille 16x16x3.



L'image du robot (vision d'un humain) *Channel « rouge » du robot (255 = rouge), l'IA a les nombres* *Channel « vert » du robot (255 = vert), l'IA a les nombres* *Channel « bleu » du robot (255 = bleu), l'IA a les nombres*

Figure 2.1 représentation d'une image œil humain vs ordinateur

Le réseau de neurones convolutionnels va ainsi recevoir cette image (vue comme une matrice), et appliquer un certain nombre d'opérations de deux catégories :

- **les filtres**, pour faire ressortir, par exemple, les arêtes verticales, les variations de coloration, etc…, sachant que plus les filtres interviennent tard dans l'enchaînement, plus ils permettent de détecter des formes complexes (et plus abstraites)
- **les simplifications**, pour alléger les calculs et dégager les informations principales

Ensuite, il donnera la probabilité que cette image représente un robot, un chien, un humain… grâce à un « réseau de neurones artificiels » [6]

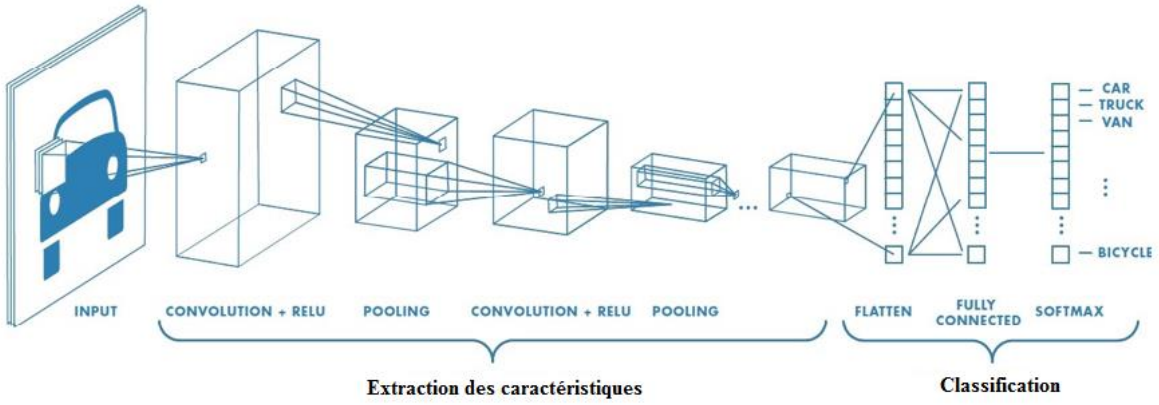


Figure 2.2 exemple d'un CNN organisé en deux parties

II.2.1. L'opération de convolution

Comme son nom l'indique la convolution est le cœur du réseau de neurones convolutionnels, à l'origine, une convolution est un outil mathématique (on parle de produit de convolution) très utilisé en retouche d'image en effet une convolution prend en entrée une image et un filtre (qui est une autre image), effectue un calcul, puis renvoie une nouvelle image (généralement plus petite). [7]

II.2.1.1 .Les filtres

Dans la pratique, les filtres d'un réseau de neurones convolutionnels ont ;

- Un kernel qui représente la taille de la matrice du filtre souvent en 3×3,
- Un stride qui représente le décalage du kernel entre chaque calcul,
- Un padding ajout des 0 autour de l'image d'entrée, pour augmenter la taille de l'image de sortie ou éviter les dépassements.

Qui sont défini à l'avance mais les valeurs des filtres sont générées aléatoirement à l'initialisation. Ensuite, lorsque le réseau apprend, les valeurs des filtres sont mises à jour pour améliorer les résultats du CNN : les valeurs des filtres font donc parties des variables (poids, biais…) que le réseau change en apprenant !

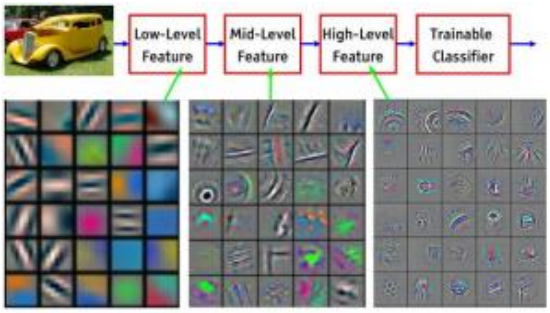


Figure 2.3 exemples de filtres à plusieurs niveaux du CNN

Le « **transfer learning** » consiste à utiliser un CNN déjà entraîné pour une tâche particulière afin de le spécialiser à un autre problème.

Les filtres se complexifient au plus la convolution intervient tard dans le CNN, au sens où ils détectent des formes de plus en plus complexes. En effet, après plusieurs applications de filtres, les images ne ressemblent plus beaucoup à l'image d'origine (il n'en reste que les informations « principales » décidées par les filtres précédents). Là où une première convolution fera ressortir les traits verticaux, la dernière convolution sera capable de trouver des structures en nid d'abeille (grâce aux transformations de l'image) : le CNN aura accumulé les détails pour en trouver l'idée globale**.** . [7]

II.2.1.2 Calcul de la convolution

Le calcul est relativement simple : pour une convolution de taille 3×3, on va sélectionner dans l'image les 3×3 premiers pixels pour en créer un nouveau dans l'image de sortie. Comme l'illustre la figure suivante

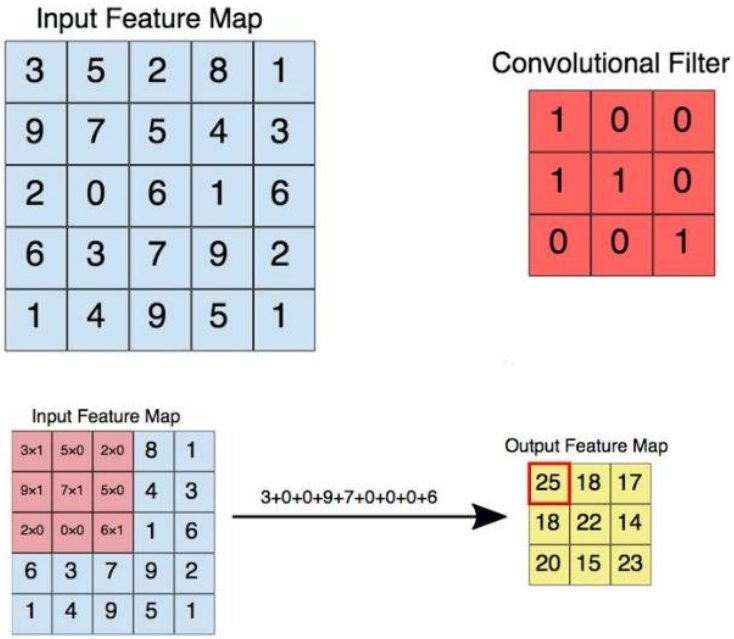


Figure 2.4 Application de la convolution sur une image

Cette taille de 3×3 est appelée le kernel (noyau en français).

Ensuite, pour calculer la deuxième valeur de l'image de sortie, on va utiliser le paramètre « stride » de la convolution. Il représente de combien de pixels dans l'image d'entrée on va se décaler pour réappliquer la convolution. [7]

II.2.2. La couche ReLU (rectification linéaire)

La fonction d'activation **ReLU** est une fonction dite « rectifier » très utilisée en apprentissage profond. Dans les réseaux de neurones convolutionnels, elle est appliquée très souvent en sortie d'une couche de convolutions pour diverses raisons :

- Comme on l'a vu, une convolution va réaliser des opérations d'additions/multiplications : les valeurs en sorties sont donc linéaires par rapport à celles en entrée.
- Or, dans une image, la linéarité n'est pas très présente ni importante (par exemple, les variations entre valeurs de pixels peuvent être importantes dans une région, l'image a des coins…).
- ReLU, de par sa définition, est une fonction qui vient **briser (une partie de) la linéarité** en supprimant une partie des valeurs (toutes celles négatives) !
- En supprimant une partie des données, ReLU permet également d'accélérer les calculs
- En ne modifiant pas les données positives, ReLU n'impacte pas les caractéristiques misent en évidence par la convolution, au contraire : elle les met davantage en évidence en creusant l'écart (valeurs négatives) « entre » deux caractéristiques (par exemple le nez et les yeux). . [7]

En résumée la couche ReLU remplace tous les nombres négatifs par la valeur 0, le noir, négatif, est supprimé, donc l'écart entre blanc et gris se creuse et il n'y a plus de transition (gradient) linéaire d'une couleur à l'autre. [7]

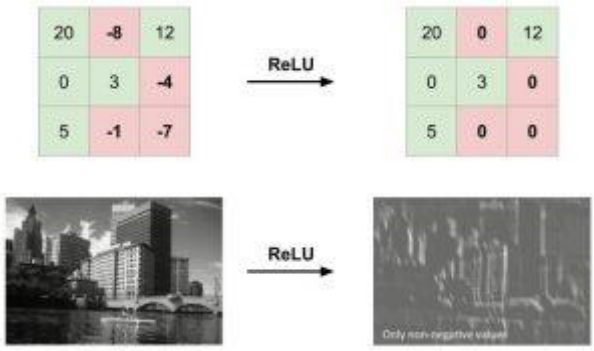


Figure 2.5 Application de ReLU à une matrice et à une image

II.2.3. cross channel normalization

Une couche cross channel normalization effectue une normalisation par canal.
Cette couche effectue une normalisation de réponse locale par canal. Il suit généralement la couche d'activation ReLU. Cette couche remplace chaque élément par une valeur normalisée obtenue à l'aide des éléments d'un certain nombre de canaux voisins (éléments dans la fenêtre de normalisation). [16]

II.2.4. Le max pooling

Le max pooling est un type de pooling qui consiste à remplacer un carré de pixels (généralement 2×2 ou 3×3) par une valeur unique. De cette manière, l'image diminue en taille et se retrouve simplifiée (lissée).

Pour appliquer le max pooling, on commence par sélectionner un carré de pixels de taille 2×2 (pour un pooling de 2×2) puis on calcule la valeur qui va venir remplacer ce carré. Ensuite, on décale ce carré vers la droite de 1 cases si le stride (= pas) vaut 1 par exemple (généralement, il vaut 1 ou 2). [7]

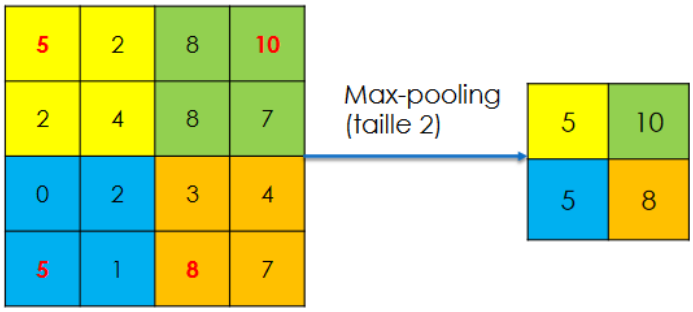


figure2.6 Application du max pooling a une image

II.2.5. Le flattening (ou mise à plat)

Dernière étape de la partie « extraction des informations », le flattening consiste simplement à **mettre bout à bout** toutes les images (matrices) que nous avons pour en faire un (long) vecteur. Les pixels (en réalité ce ne sont plus des images ou des pixels, mais des matrices de nombres, donc les pixels sont ces nombres) sont récupérés ligne par ligne et ajoutés au vecteur final. [7]

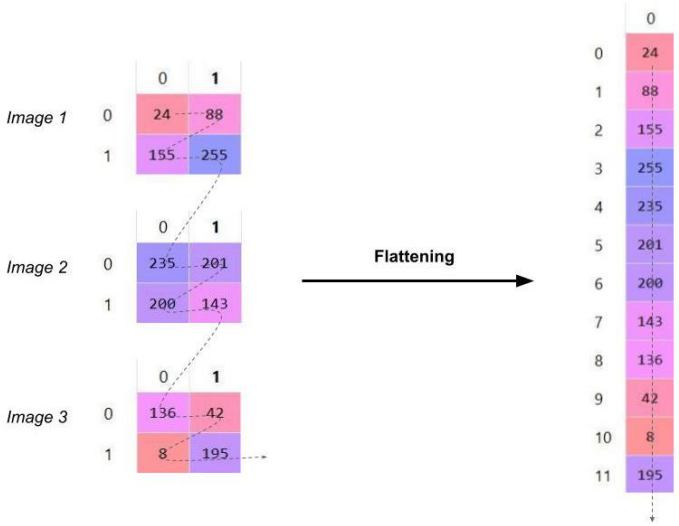


figure2.7 Mise à plat des images finales en sortie des filtres+simplificiations

En fait, le réseau de neurones (étape 5) prend simplement en entrée un vecteur (à chaque neurone d'entrée on envoie une seule valeur) !

Par conséquent, dans l'absolu, rien n'empêche d'utiliser un flattening qui lit les matrices par colonne, ou même qui mélange tous les « pixels » (sans en changer les valeurs), tant que le procédé de flattening reste toujours le même. [7]

II.2.6. les couches entièrement connectées

Les couches entièrement connectées sont les principaux blocs de construction des CNN. Au lieu de traiter l'entrée comme des tableaux de 2 Dimensions, ils sont traités en tant que liste unique et traités de manière identique. Chaque valeur à son propre vote quant à si l'image est un chat ou un chien par exemple. Cependant, le process n'est pas complètement démocratique. Certaines valeurs sont bien meilleures à détecter lorsqu'une image est un chat que d'autres, et d'autres sont bien meilleures à détecter un chien. Celles-ci ont donc davantage de pouvoir de vote que les autres. Ce vote est appelé le poids, ou la force de la connexion, entre chaque valeur et chaque catégorie.

Lorsqu'une nouvelle image est présentée au CNN, elle se répand à travers les couches inférieures jusqu'à atteindre la couche finale entièrement connectée. L'élection a ensuite lieu. Et la solution avec le plus de vote gagne et est déclarée la catégorie de l'image.

Les couches entièrement connectées, tout comme les autres couches, peuvent être ajoutées les unes à la suite des autres car leur valeur en sortie (une liste de votes) ressemble énormément à leur valeur en entrée (une liste de valeur). En pratique, plusieurs couches entièrement connectées sont souvent ajoutées les unes à la suite des autres, avec chaque couche intermédiaire votant pour des catégories "cachées" fantômes. En effet, chaque couche additionnelle laisse le réseau apprendre des combinaisons chaque fois plus complexes de caractéristiques qui aident à améliorer la prise de décision.

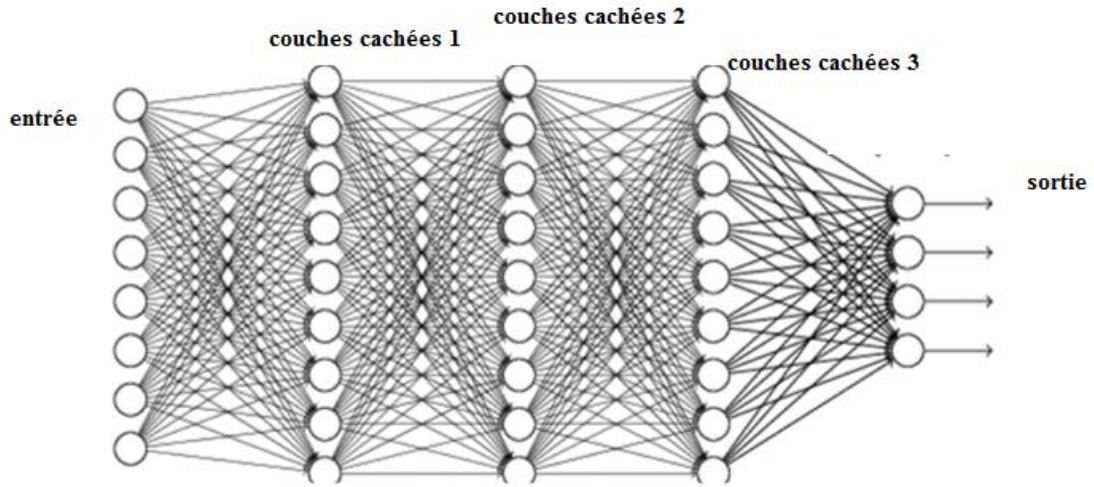


Figure 2.8 Exemple des couches entièrement connectées

- **les neurones d'entrée**, qui envoient leurs valeurs à tous les neurones de la couche suivante. Dans le cas du CNN, ce sera la valeur d'un pixel précis pour chaque neurone (tout simplement, en suivant l'ordre du vecteur)
- **les neurones cachés**, organisés en couches, qui vont envoyer la somme des signaux qu'ils reçoivent (pondérés par « l'importance » de leur liaison) aux neurones de la couche suivante
- **les neurones de sortie**, qui reçoivent la somme des signaux pondérés de la dernière couche cachée

Chaque neurone de sortie représente alors une **prédiction spécifique**. Par exemple, le premier neurone représente la prédiction « c'est un chat », le deuxième « c'est un chien », etc… Et la conclusion de notre réseau dépend de quel neurone de sortie a le signal le plus fort !

Notre partie fully connected reçoit donc un vecteur de nombres et renvoie des probabilités pour chaque classe de prédiction. On fait alors ce qu'on veut de ces probabilités, soit prendre la plus grande, soit n'en garder aucune si elles sont trop faibles, soit…

II.2.7. la couche dropout

Au moment de la prédiction, la sortie d'une couche dropout est égale à son entrée. Au moment de l'entraînement, l'opération correspond à l'abandon temporaire d'une unité choisie au hasard et de toutes ses connexions du réseau pendant l'entraînement. Ainsi, pour chaque nouvel élément d'entrée, le réseau sélectionne de manière aléatoire un sous-ensemble de neurones, formant une architecture de couche différente. Ces architectures utilisent des poids communs, mais étant donné que l'apprentissage ne dépend pas de neurones et de connexions spécifiques, la couche dropout peut aider à prévenir le sur-apprentissage ; en résumé, dropout met au hasard les éléments d'entrée à zéro avec une probabilité donnée. [16]

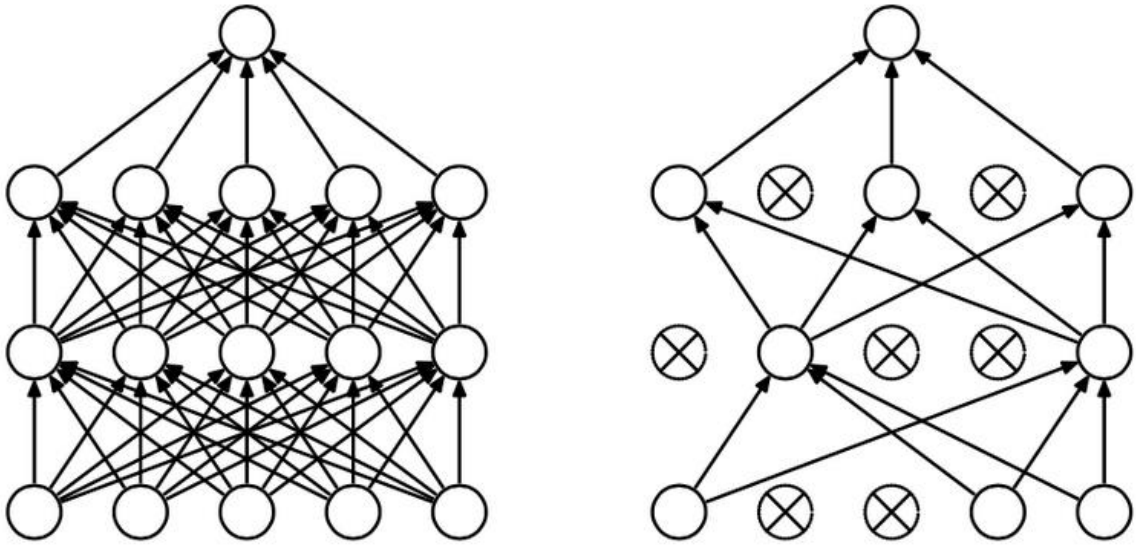


Figure 2.9 application de la couche dropout

II.2.8 Couche Prob «softmax »

La couche Softmax est mise en œuvre via une couche de réseau de neurones juste avant la couche du résultat « output ». Permet de calculer la distribution de probabilités sur les classes de sortie. La couche Softmax doit comporter le même nombre de nœuds que la couche du résultat. [23]

II.2.9. Mise bout à bout du réseau neuronal convolutionnel :

Nous avons à présent tous les outils pour comprendre l'architecture d'un réseau de neurones convolutionnels. Il en existe plusieurs dans la littérature dont l'efficacité varie en fonction de la tâche, car ils n'ont pas tous le même nombre de convolutions (ni la même structure). [7]

II.3. Conclusion :

Dans ce chapitre, nous avons présenté les réseaux de neurones convolutionnels. Qui ont la capacité d'extraire des caractéristiques d'images présentées en entrée et de classifier ces caractéristiques, ils implémentent aussi l'idée de partage des poids qui permettant de réduire les temps de calcul, l'espace mémoire nécessaire, et également d'améliorer les capacités de généralisation du réseau. Nous avons vu qu'il existée plusieurs modèles prés-entrainées sur diverses catégories d'images.

CHAPITRE III

Réalisation et discussions

III.1. Introduction :

Dans ce chapitre nous allons parler du travail que nous avons réalisé c'est-à-dire le programme que nous avons fait pour la reconnaissance de l'iris, qui a était fait sous le logiciel MALAB R2018a, ainsi que des bases de données avec lesquelles nous avons travaillé et enfin discuter les résultats.

L'ordinateur utiliser est ordinateur personnel HP avec un 4GO de capacité mémoire, et un processeur Intel® coré™ i3-6006U CPU @ 2.00 GHz, avec Windows 8.1 Professionnel avec un système d'exploitation 64 bits.

III.2. Prétraitement des données :

Afin d'obtenir une bonne détection et de reconnaissance de l'iris des prétraitements sont nécessaires, c'est-à-dire qu'un système de reconnaissance biométrique par l'iris est constitué de plusieurs modules comprenant :

III.2.1- élimination des points blancs d'iris :

Grace aux opérateurs morphologiques (Binarisation, Ouverture)

Figure 3.1. Avant et après l'élimination des points blancs de l'iris

III.2.2- Filtrage médian :

Le filtrage médian est une opération non-linéaire,

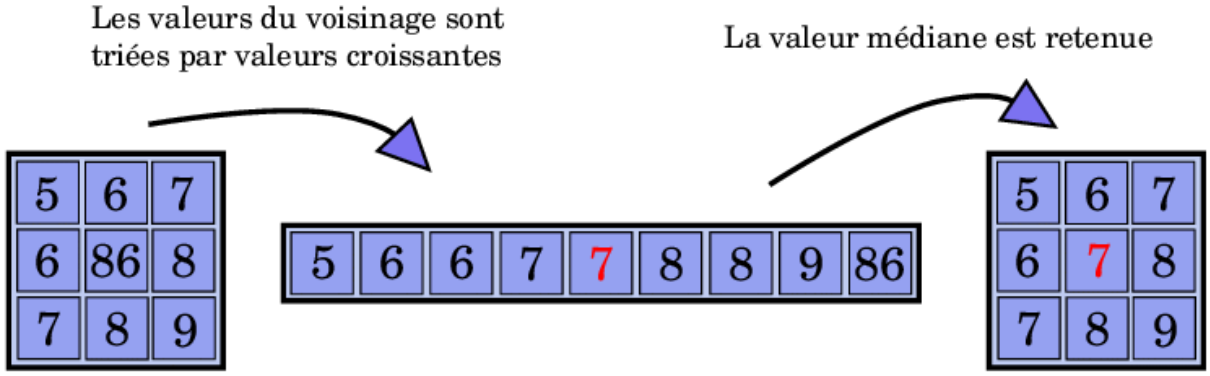


Figure 3.2. Exemple de l'application d'un filtre médian 3*3

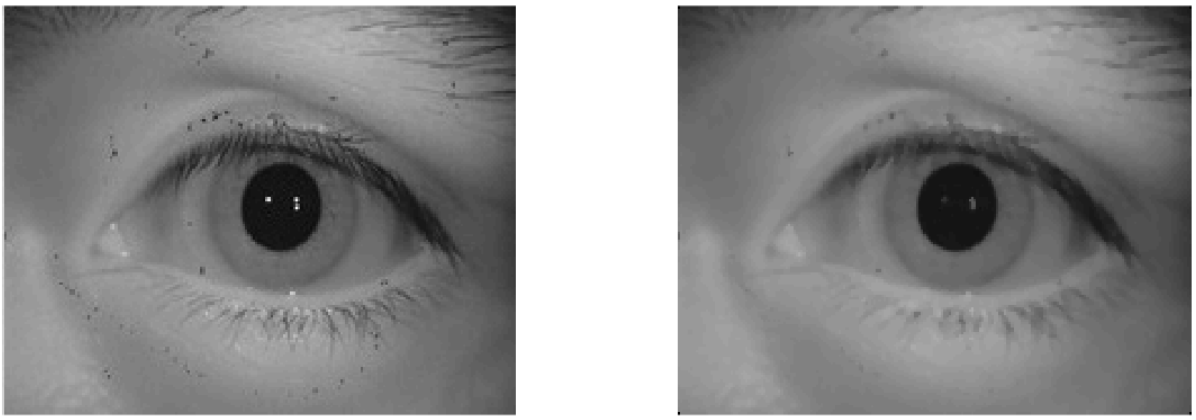


Figure 3.3. Application d'un filtre médian 3*3

Comme nous pouvons le constater le bruit sur l'image à complétement disparus, même si l'image est un peu floue

III.2.3 la localisation de l'iris (segmentation) :

Après l'acquisition de l'iris nous allons appliquer la segmentation de l'iris qui est la première étape dans le processus de reconnaissance, cette étape est très importante car nous allons isoler l'iris du reste de l'image et pour cela nous allons utiliser la transformée de Hough qui est une technique utilisée afin d'isoler des objets de formes géométriques simples dans l'image. En général, on se limite aux lignes, cercles ou ellipses présents dans l'image. L'un des grands avantages de la transformée de Hough est qu'elle est tolérante aux occlusions dans les objets recherchés et demeure relativement inaffectée par les bruits. Les objets à détecter dans l'image de l'œil (iris, pupille, paupières) sont circulaires ou ellipsoïdaux et donc se prêtent bien à une détection par la transformée de Hough. Wildes a été le premier à introduire cette méthode dans le contexte de la segmentation de l'iris .Les étapes de la transformée de Hough sont les suivantes : Une image de contours est générée par une quelconque méthode de génération de contours. Plusieurs images de contours obtenues par la méthode Canny sont montrées sur la Figure (3.5).Un processus de vote est mis en place sur l'image de contours obtenue. Chaque point de contour vote pour les cercles dont il appartient et le cercle qui obtient le plus de vote est le cercle recherché. Parfois cette méthode est lourde sur tout lorsque beaucoup de points de contours sont détectés et si nous avons une connaissance à priori sur la localisation du cercle recherché. Dans ce cas, nous comptabilisons pour chaque cercle dans la zone recherchée, les points de contours appartenant à ce cercle. Le cercle qui possède le plus de points de contours est le cercle recherché. [9]

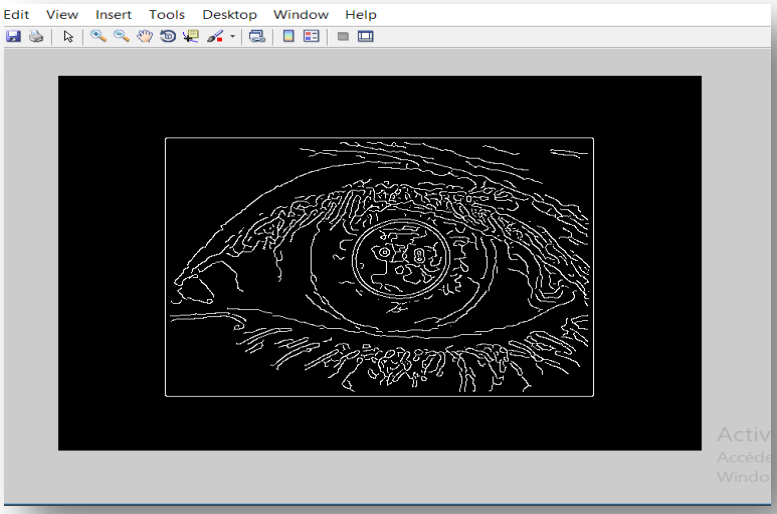


Figure 3.4 Détection de l'iris par la méthode de Canny

Quelques reflets sont localisés à l'intérieur de la pupille. Avant d'appliquer la méthode de Canny aux images d'entrées pour extraire les contours de la pupille et de l'iris, on doit d'abord éliminer ces reflets pour ne pas avoir des contours à l'intérieure de la pupille. [9]

Figure 3.5 image segmentée avec reflet

Pour éviter l'erreur à la segmentation (détection du contour d'iris), comme montré dans la figure :

Figure 3.6 image segmentée sans reflet

Les étapes à suivre pour déterminer le centre de la pupille sont donc :

- Binarisation de l'image
- Seuillage
- Détermination du rayon Rp et du centre de la pupille cp (xp, yp).

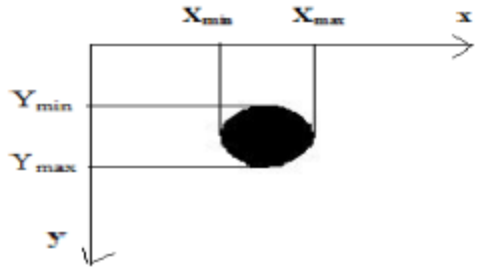


Figure 3.7 Détermination du rayon et du centre de la pupille

Le rayon et le centre de la pupille sont donnés par les formules suivantes :

$$R_p = (x_{\max} - x_{\min}) / 2$$
$$x_p = R_p + x_{\min}$$
$$y_p = R_p + x_{\min}$$

Après avoir déterminé Cp (xp, yp) on peut extraire le rayon de l'iris Ri : à partir du centre de la pupille en partant du ce point (Cp) que nous avons trouvé, nous avançons jusqu'à trouver un deuxième contour. Nous notons alors le point (Xi) où nous sommes arrivés: il appartient au bord de l'iris (voir figure.3.5) [9]

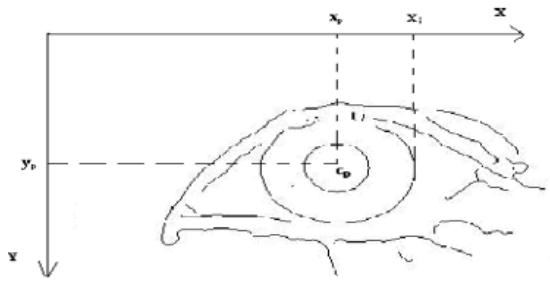


Figure 3.8 Détermination le Rayon de l'iris

Donc le rayon de l'iris est donné par l'équation suivante**:** $R_i = (x_i - x_p)$

À partir de ces équations Nous pouvons facilement isoler la région de l'iris. [9]

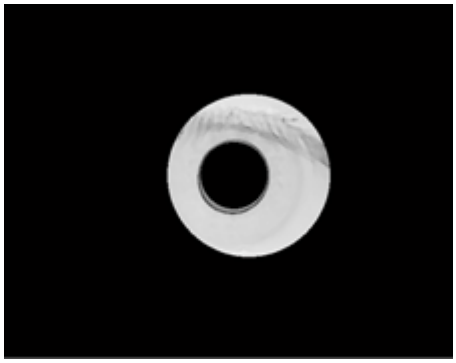


Figure 3.9 localisation de l'iris

III.2.4. Normalisation de la région de l'iris:

Après la segmentation, l'iris nécessite une opération de normalisation pour pallier le non concentricité des deux bordures et à la variation de la taille de l'iris due à la dilatation/contraction de la pupille et la distance et l'angle de capture de l'image de l'œil .Cette étape consiste à transformer l'iris en une image rectangulaire de taille fixe, afin de permettre les comparaisons. Pour choisir la taille de l'image rectangulaire adéquate qui permet une bonne identification, nous avons proposé une étude comparative en transformant l'iris segmenté en plusieurs images rectangulaires de différentes tailles,. Cette opération de normalisation de l'image de l'iris est obtenue en assignant à chaque pixel de l'iris dans le domaine cartésien, un correspondant dans le domaine pseudo polaire suivant la distance du pixel par rapport aux centres des cercles et l'angle qu'il fait avec ces centres [9]. Cette transformation de domaine peut être représentée dans l'équation suivante :

$$x(r,\theta) = (1-r)x_p + rx_s(\theta)$$
$$y(r,\theta) = (1-r)y_p + ry_s(\theta)$$

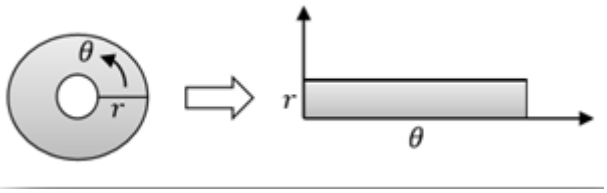


Figure 3.10 Transformation en Pseudo-Polaire

Xp(θ) : représente l'abscisse du point de la frontière détectée de la pupille dont le segment qui passe par ce point et le centre de la pupille fait un angle θ avec une direction choisie.

Yp(θ) : représente l'ordonnée de ce même point,

Xs(θ), Ys(θ) : représentent les coordonnées des points obtenus par le même principe mais sur le contourne l'iris. L'image de la figure (3.11) montre une image normalisée obtenue par ce processus qui est rectangulaire et de taille constante, généralement la taille choisie est de 80*512 pixels

- L'image de la figure (3.12) à droite montre une image normalisée obtenue par ce processus. Comme c'est montré l'image normalisée est rectangulaire de taille constante, la taille choisie est de 460*840 pixels. La largeur de l'image représente la variation sur l'axe angulaire alors que la hauteur représente les variations sur l'axe radial. [9]

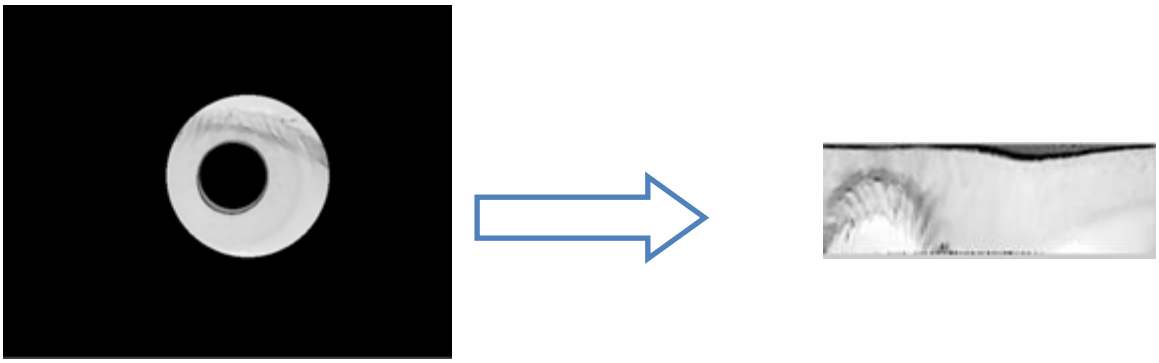


Figure3.11 Normalisation de l'iris

III. 3. Post-traitement(CLAHE):

Cette méthode, est utile pour augmenter le contraste de l'image en niveaux de gris par la transformation des valeurs de contraste limité par égalisation d'histogramme adaptatif (CLAHE). À la différence de l'égalisation normale, cette méthode fonctionne par régions de l'image, plutôt que sur l'image entière. Le contraste de chaque région est amélioré, de sorte que l'histogramme de la région de sortie correspond approximativement à l'histogramme spécifié par le paramètre 'Distribution'. Les régions voisines sont ensuite combinées en utilisant l'interpolation bilinéaire pour éliminer limites induites artificiellement. Le contraste, en particulier dans les zones homogènes, peut être limité pour éviter d'amplifier les bruits qui pourraient être présents dans l'image. [10]

Donc voici le résultat qu'on obtient après l'amélioration :

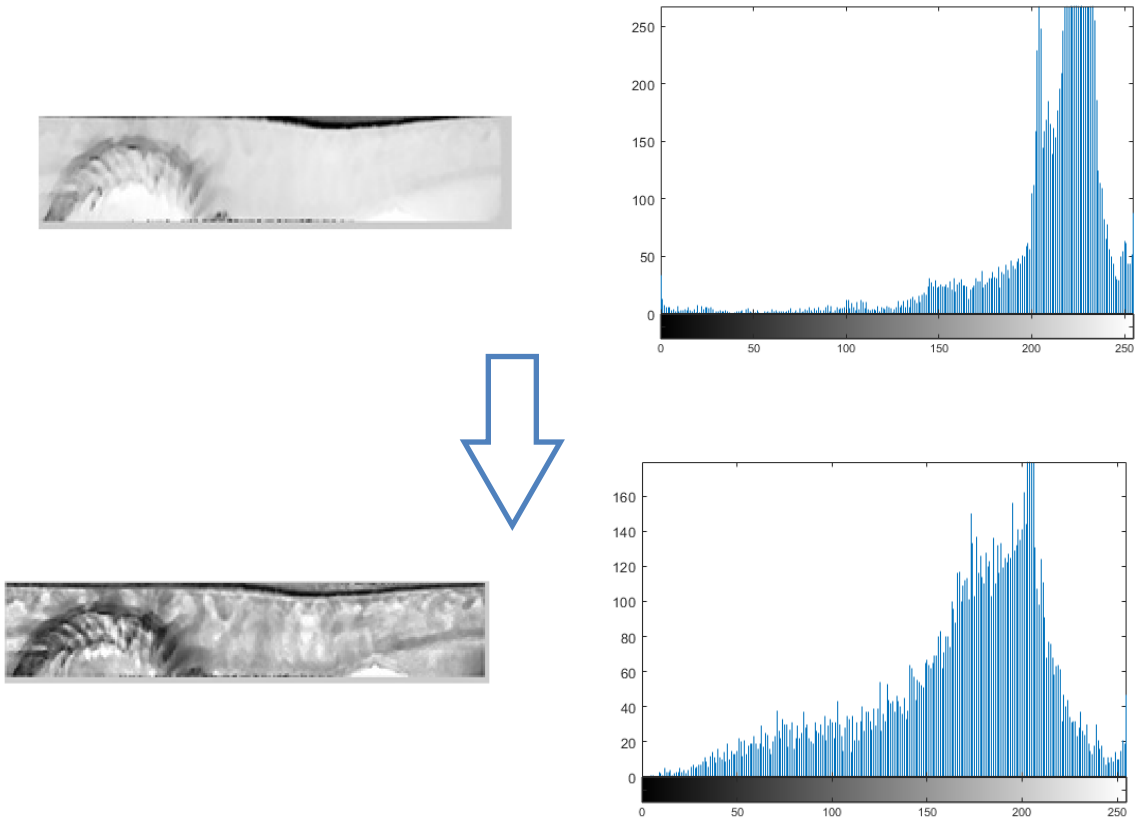


Figure 3.12 : image d'iris amélioré par l'histogramme de CLAHE

III.4. Bases de données :

CASIA-IrisV4 est une extension de CASIA-IrisV3 et contient six sous-ensembles, dont CASIA-Iris-Interval, CASIA-Iris-Lamp , CASIA-Iris-Twins, CASIA-Iris-Distance, CASIA-Iris-Thousand et CASIA-Iris-Syn.CASIA-IrisV4, Toutes les images iris sont des fichiers JPEG, collectés sous éclairage infrarouge proche ou synthétisés. Les six ensembles de données ont été collectés ou synthétisés à différents moments.

Dans ce travail on a utilisé la partie CASIA-LAMP de CASIA V4, collectée à l'aide d'un capteur de l'iris à main fabriqué par OKI (Fig.3.14). Une lampe a été allumée / éteinte près du sujet pour introduire davantage de variations intra-classe lorsque nous avons collecté CASIA-Iris-Lamp. La déformation élastique de la texture de l'iris (Fig. 3.15) due à la dilatation et à la contraction de la pupille sous différentes conditions d'éclairage est l'un des problèmes les plus courants et les plus difficiles en matière de reconnaissance de l'iris.

Figure 3.14 La caméra d'iris à main utilisée pour la collecte de CASIA-Iris-Lamp

Figure 3.15 Exemples des images d'iris obtenus à partir de base de données CASIA-Iris-Lamp

Pour notre programme nous avons choisi cinq types de base de donnée dont :

1. BDD1 : contient des images qui ont été segmentées et normalisées sans prétraitement ni poste traitement
2. BDD2 : contient des images ont été segmentées et normalisées et passée par un post traitement uniquement
3. BDD3 : contient des images en collerette ; on prend la zone de l'iris la plus proche de la pupille pour éviter la zone de bruit ; (près-traitement + segmentation + normalisation + post traitement)
4. BDD4 : contient des images d'iris localisé ; ici nous avons pris tous l'iris (près-traitement + segmentation + normalisation + post traitement)
5. BDD5 : contient des images brutes (sans aucun traitement).

NOTE : ces base de données compotent toutes la même personne c'est-à-dire que l'image de la personne 1 pour la BDD 1 est la même que pour toutes les autres bases.

Exemple d'une image de la personne 1 pour les cinq bases de données

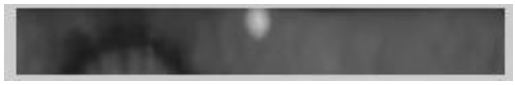


Figure 3.16 Image de la BDD1

Figure3.17 Image de la BDD2

Figure3.18 Image de la BDD3

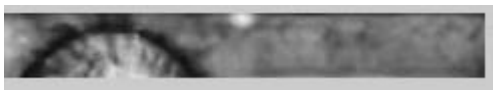


Figure3.19 Image de la BDD4

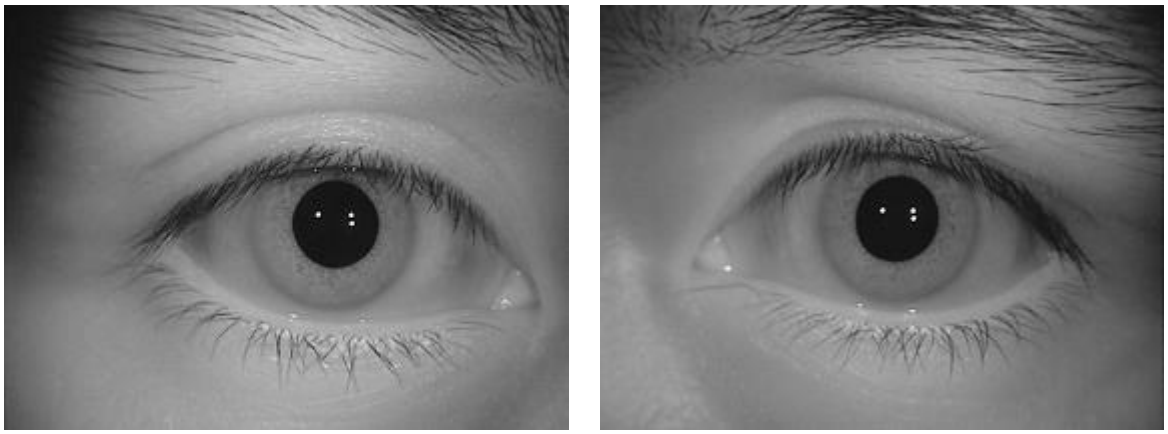


Figure 3.20 Image de la BDD5 Gauche et droite

Nous avons commencé par 10 classes seulement c'est-à-dire que nous avons pris 10 personnes et pour chaque personne nous avons pris 10 images d'iris dans des conditions d'acquisition différentes et non idéales.

Nous avons refait un deuxième test mais avec 200 personnes, pour chaque personne nous avons 10 images de l'œil gauche et 10 pour l'œil droit comme dans la figure 3.20, ce qui spécifie cette base de donnée n'est pas seulement que les images n'ont subi aucun traitement que ça soit la segmentation / normalisation ou le CLAHE mais que certaines de ces images n'ont pas été acquises dans le cas idéal, nous avons découvert que les personnes ferment légèrement les yeux, d'autres n'ont pas eu un bon angle de luminosité...etc., comme le montre la figure suivante

Figure3.21 exemple des images dans un cas de contrainte

Apres avoir choisi les bases de données et effectuer les traitements nécessaires, nous allons passer à la classification des images, pour ce faire nous avons choisi deux modèles pré-entrainées qui sont AlexNet et GoogleNet, or ces deux modèles sont entraines sur la classification des objets ils ne connaissent pas les images de l'iris donc un apprentissage par transfert est obligatoire.

III.5. Apprentissage par transfert

La plupart des applications de l'apprentissage profond utilisent la méthode d'apprentissage par transfert, ou Transfer Learning, qui consiste à mettre au point un modèle pré-entraîné. Le processus commence avec un réseau existant, tel qu'AlexNet ou GoogleNet, qu'il faut enrichir avec de nouvelles données contenant des classes auparavant inconnues du réseau. Une fois quelques ajustements effectués au réseau, nous pouvons effectuer une nouvelle tâche, telle que la reconnaissance de l'iris, plutôt que de 1 000 objets différents. Cette technique présente également l'avantage de nécessiter un volume de données beaucoup plus faible (il faut traiter des milliers d'images, plutôt que des millions), faisant du calcul une affaire d'heures ou de minutes. [18]

Nous allons voir comment ajuster un réseau de neurones convolutionnels pour effectuer la classification sur une nouvelle collection d'images.

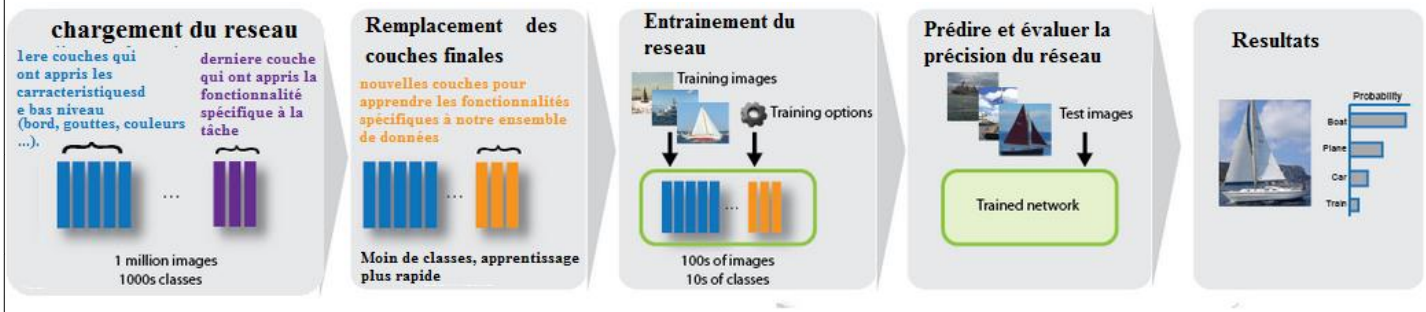


Figure 3.22 : réutilisation d'un réseau pré entrainé

III.5.1. Chargement des données :

Avant toute chose nous devons trier nos données puisque dans un premier essai nous avons choisie de travailler avec 10 personnes c'est-à-dire 10 classes nous allons donc organiser nos dossier de tels façons à ce que l'on crée 10 dossiers pour les 10 personnes et mettre les images de chaque personne dans son dossier, la commande « `ImageDatastore` » sur matlab permet justement de lire cette bases de donnée et étiquette automatiquement les images en fonction des sous dossiers.

Ensuite nous allons définir un ensemble d'images pour l'apprentissage, et un ensemble pour la validation, en utilisant la fonction « **`splitEachLabel`**», qui va prendre aléatoirement 70% des images pour l'apprentissage et les 30% restant pour la validation

III.5.2. Chargement du modèle prédéfini :

III.5.2.1.AlexNet :

Comme mentionné dans le 2ém chapitre, AlexNet a été retenu dans ILSVRC 2012. Il résout le problème de la classification des images où l'entrée est une image de l'une des 1000 classes différentes (par exemple, chats, chiens, etc.) et la sortie est un vecteur de 1000 éléments.

Le Nième élément du vecteur de sortie est interprété comme la probabilité que l'image d'entrée appartient à la Nième classe. Par conséquent, la somme de tous les éléments du vecteur de sortie est 1. [17]

AlexNet contient 5 couches convolutives et 3 couches entièrement connectées.la couche Relu est appliquée après les couches de convolution et entièrement connecté. La couche dropout est appliquée avant les première et deuxième couches entièrement connectées. Le réseau compte 62,3 millions de paramètres et nécessite 1,1 milliard d'unités de calcul dans un passage en aval. Nous pouvons également voir que les couches de convolution, qui représentent 6% de tous les paramètres, consomment 95% du calcul.

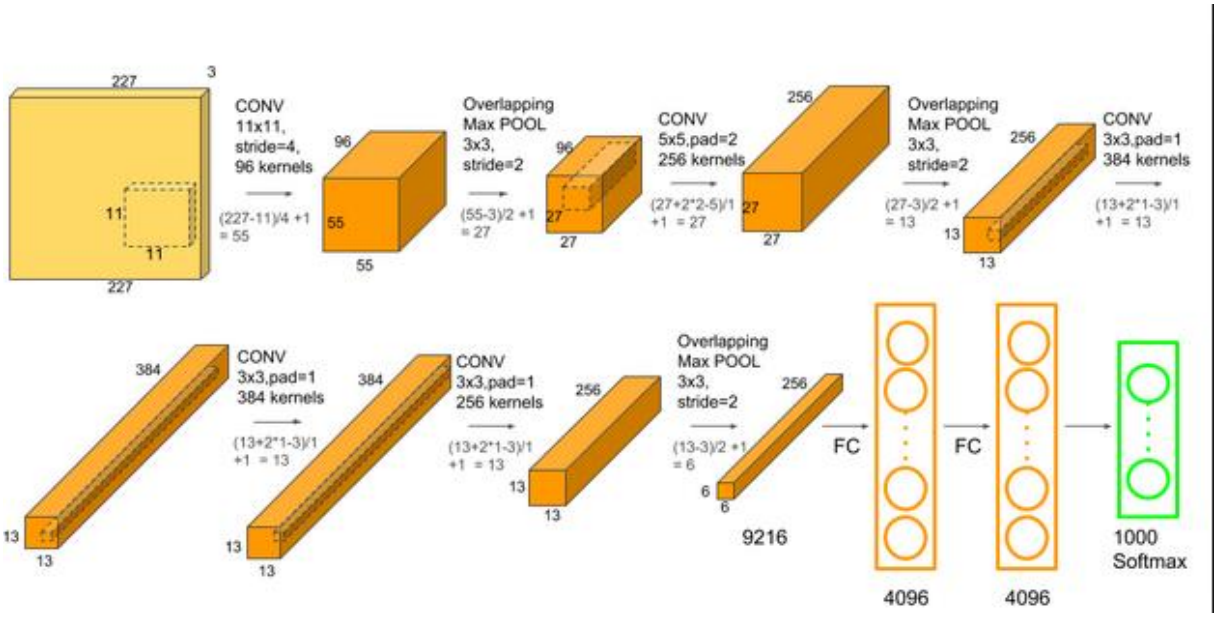


Figure 3.23 visualisation de l'architecture du réseau AlexNet

Pour afficher une visualisation interactive de l'architecture du réseau et des informations détaillées sur les couches réseau nous utiliserons la commande **`analyzeNetwork [ANNEXE 1]`**

III.5.2.2.GoogleNet :

GoogleNet est le lauréat de l'ILSVRC 2014, avec un taux d'erreur de 6,67% dans le top 5, le réseau utilise un réseau CNN inspiré par LeNet. Son architecture contient une convolution 1 × 1 au centre du réseau et la mutualisation moyenne globale est utilisée à la fin du réseau au lieu d'utiliser des couches entièrement connectées. Il utilise également le module Inception, qui consiste à avoir différentes tailles / types de convolutions pour la même entrée et l'empilement de toutes les sorties. Il a également utilisé la normalisation par lots, les distorsions d'image et RMSprop. Dans GoogLenet 1 × 1, la convolution est utilisée comme

module de réduction des dimensions pour réduire le calcul. En réduisant le goulot d'étranglement du calcul, il est possible d'augmenter la profondeur et la largeur. L'architecture de GoogleNet consistait en un réseau CNN profond de 22 couches mais réduisait le nombre de paramètres de 60 millions (AlexNet) à 4 millions. [17]

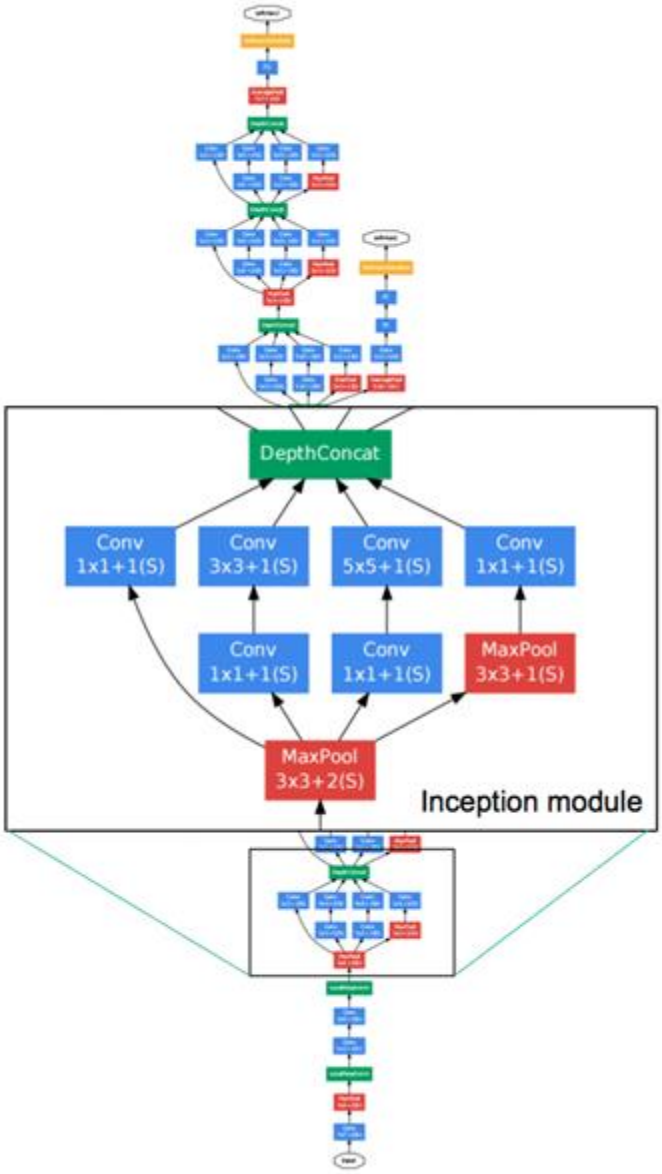


Figure 3.24 .partie architecture du réseau GoogleNet

L'idée de base : avoir des filtres en parallèle avec des champs récepteurs de tailles multiples (pyramide spatiale), la couche suivante a donc accès à des caractéristiques à plusieurs échelles spatiales. [17]

NB : La première couche qui est la couche d'entrée d'image, nécessite des images d'entrée d'une taille spécifique sans ça le réseau ne les acceptera pas,

AlexNet ➔ `227 x 227 x 3` GoogleNet ➔ `224 x 224 x 3`

III.5.3. Remplacement les couches finales

Les trois dernières couches du réseau pré-entraîné sont configurées pour 1000 classes. Ces trois couches doivent être ajustées pour le nouveau problème de classification. En remplaçant les trois derniers couches ; par une couche entièrement connecté « fc », une couche « softmax » et une couche « classification output ». Comme dans la figure ci-dessous :

- **Pour le modèle AlexNet :**

Figure 3.25 changement des trois dernières couches du modèle AlexNet

- **Pour le modèle GoogleNet :**

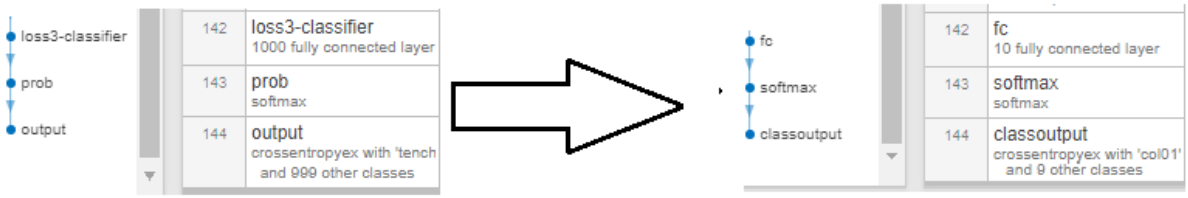


Figure 3.26 changement des trois dernières couches du modèle GoogleNet

Comme nous pouvons le voir sur les figure les 3 dernières couches ont été remplacées.

En fonction des nouvelles données nous allons spécifier les options de la nouvelle couche entièrement connectée « fc », car elle doit avoir la même taille que le nombre de classes pour les nouvelles données. [15]

III.5.4. Entrainement du réseau

Le réseau nécessite des images d'entrée de taille 227 x 227 x 3 pour l'AlexNet et 224 x 224 x 3 pour GoogleNet

, mais les images utilisées ont des tailles différentes.

Nous devons donc redimensionner les images de l'apprentissage et spécifier les opérations d'augmentation supplémentaires à effectuer sur les images d'apprentissage: faire pivoter les images d'apprentissage de façon aléatoire le long de l'axe vertical et effectuez leur translation de manière aléatoire, jusqu'à 30 pixels horizontalement et verticalement.

Pour la partie apprentissage « training » il est nécessaire de faire une augmentation de données pour enrichir la base de données et, améliorer le résultat,

En d'autres termes effectués des transformations aléatoires sur les images d'entrée. Cela permet d'empêcher le réseau de mémoriser la position et l'orientation exactes des objets, ou

encore en générant des échantillons artificiels pour une ou plusieurs classes du jeu de données en **rotation, zoom, mise en miroir, flou** sur l'ensemble original d'images.[16]

III.6.1 la classification avec un modèle CNN près-entrainés

La classification des images consiste à répartir systématiquement des images selon des classes établies au préalable, classer une image lui fait correspondre une classe, marquant ainsi sa parenté avec d'autres images.

L'objectif de la classification d'images est de mettre en place un système capable d'attribuer une classe automatiquement à une image. Notre programme va donc classer les images les images de validation utilisant le réseau, et enfin calculer le taux d'apprentissage « Accuracy » c'est-à-dire l'efficacité de la méthode ou l'exactitude de la classification. La précision est la fraction des étiquettes que le réseau prédit correctement.

L'utilisation des réseaux de neurones à convolution nous simplifie l'opération de classification car nous n'allons pas effectuer une extraction de caractéristiques manuelle, ni de passée par la tâche d'identification des caractéristiques utilisées.
En effet le réseau extrait des caractéristiques directement depuis des images. Les caractéristiques utilisées ne sont pas pré-entraînées : le réseau les apprend lui-même en s'entraînant sur des images. Cette extraction automatisée des caractéristiques permet aux modèles de Deep Learning d'atteindre un taux de précision particulièrement élevé pour les tâches de vision par ordinateur telles que la classification d'objets. [17]

Voici donc un tableau qui va décrire les résultats obtenus avec les différentes bases de données pour les modèles AlexNet et GoogleNet

	Taux d'apprentissage	
BDD	**AlexNet**	**GoogleNet**
BDD1	**40,00%**	**50,00%**
BDD2	**76,67%**	**66,67%**
BDD3	**66,67**	**46,67%**
BDD4	**83,33%**	**76,67%**
BDD5	**100%**	**96,76%**

Tableau 3.1 Taux d'apprentissage pour les différentes bases de données avec une classification avec les modèles AlexNet et GoogleNet

Nous cherchons à améliorer les résultats donc nous allons changer les options du training qui sont : `MiniBatchSize`, `MaxEpochs`

- Le BatchSize : est un hyper-paramètre de descente de gradient qui contrôle le nombre d'échantillons d'entraînement à traiter avant la mise à jour des paramètres internes du modèle. Un mini- BatchSize est un sous-ensemble de l'ensemble d'apprentissage utilisé pour évaluer le gradient de la fonction de perte et mettre à jour les poids.

- Le nombre d'époques est un hyper paramètre de descente de gradient qui contrôle le nombre de passages complets dans l'ensemble de données d'apprentissage. 'MaxEpochs' c'est le nombre maximal d'époques à utiliser pour l'apprentissage, spécifié. [12]

Voici les résultats obtenu après avoir modifier les valeurs de ces options :

	taux d'apprentissage		
BDD	**MiniBatchSize: 10, MaxEpochs : 6**	**MiniBatchSize: 20, MaxEpochs : 6**	**MiniBatchSize: 25, MaxEpochs : 6**
BDD2	**56,67%**	**46.67%**	**40%**
BDD3	**66,67**	**36,67**	**40%**
BDD4	**83,33%**	**83,33%**	**66,67%**
BDD5	**100%**	**100%**	**100%**

Tableau 3. 2 Taux d'apprentissage pour les différentes BDD en changeant MiniBatchSize pour une classification avec le modèle l'AlexNet

	taux d'apprentissage		
BDD	**MiniBatchSize: 10, MaxEpochs : 6**	**MiniBatchSize: 20, MaxEpochs : 6**	**MiniBatchSize: 25, MaxEpochs : 6**
BDD2	**63,33%**	**53,33**	**40%**
BDD3	**46,67%**	**46,67**	**26,67%**
BDD4	**76,67%**	**53,33%**	**40,00%**
BDD5	**96,76**	**100%**	**93,33%**

Tableau 3. 3 Taux d'apprentissage pour les différentes BDD en changeant MiniBatchSize pour une classification avec le modèle GoogleNet

Remarque : La BDD1 a été écartée car son taux reste toujours faible malgré les changements.

Nous avons commencé par augmenter les valeurs du MiniBatchSize mais nous pouvons remarquer que pour les trois première BDD le taux diminue car pour la mise à jour du modèle elle sera retardé, (au lieu de faire la mise à jour avec un sous ensemble de 10 il le fera avec 20), et ça nous arrange pas, par contre pour les images de la dernière base ça a donnée directement un résultat de 100% Pour le modèle AlexNet mais pour GoogleNet c'est assez bon

Nous allons donc changer la valeur de MaxEpochs

	taux d'apprentissage			
BDD	**MiniBatchSize: 10, MaxEpochs : 18**	**MiniBatchSize: 10, MaxEpochs : 25**	**MiniBatchSize: 10, MaxEpochs : 33**	**MiniBatchSize: 10, MaxEpochs : 36**
BDD2	**56,76%**	**83,33%**	**90%**	**86%**
BDD3	**73,33%**	**97,67%**	**90%**	**90%**
BDD4	**83,33%**	**90%**	**93,33**	**93,33%**
BDD5	**100%**	**100%**	**100%**	**100%**

Tableau 3. 4. Taux d'apprentissage pour les différentes BDD en changeant MaxEpochs pour une classification avec le modèle AlexNet

	taux d'apprentissage			
BDD	**MiniBatchSize: 10, MaxEpochs : 18**	**MiniBatchSize: 10, MaxEpochs : 25**	**MiniBatchSize: 10, MaxEpochs : 33**	**MiniBatchSize: 10, MaxEpochs : 36**
BDD2	**76,67%**	**80%**	**80%**	**93,33%**
BDD3	**70%**	**70%**	**73%**	**76,67%**
BDD4	**70%**	**76,67%**	**86,67**	**93,33**
BDD5	**96,77%**	**98%**	**96,77%**	**63,76%**

Tableau 3. 5. Taux d'apprentissage pour les différentes BDD en changeant MaxEpochs pour une classification avec le modèle GoogleNet

La précision d'apprentissage augmente avec le nombre d'époque ceci reflète qu'après chaque époque le modèle apprend plus d'informations.

Nous pouvons voir que le résultat s'améliore pour le modèle AlexNet mais le problème reste le temps d'exécution, plus Nous augmentant le maxEpochs plus le programme prend du temps à s'exécuter, nous constatant que la BDD des images brutes qui n'ont subis aucun traitement a donné un taux d'apprentissage de 100% dans les quatre cas. Le résultat pour GoogleNet et moins bon que pour le premier modèle malgré l'augmentation des paramètres de l'apprentissage.
Notre but est la reconnaissance de l'iris par CNN, or 10 classes c'est petit, nous allons essayer la classification avec l'AlexNet vue qu'il a donné de meilleur résultat, pour 200 classes et pour chaque classe nous allons mettre les images de l'œil de la personne en question en laissant les options par défauts.

Figure 3.27. Exemple d'une image de la BDD5

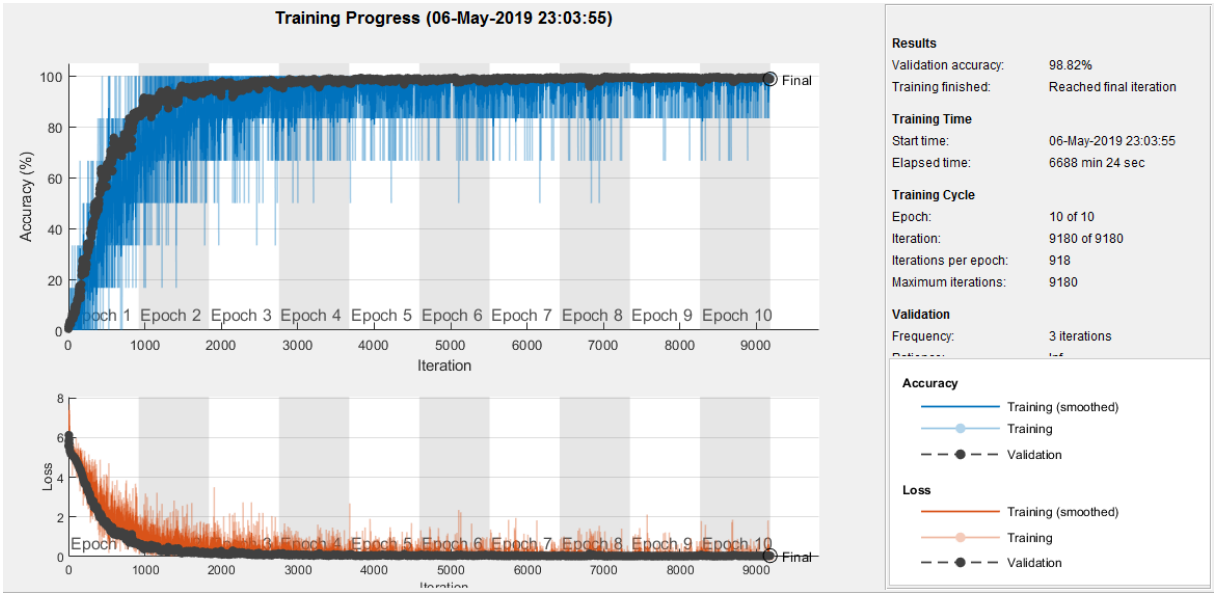


Figure 3.28 résultat de la BDD 5 pour une classification avec le modèle AlexNet

Comme nous pouvons le constater AlexNet à donner un taux d'apprentissage de 98.82% Nous pouvons constater qu'il s'est stabilisé à l'époque 4, malgré le volume de la base de données et les qualités des images, le seul inconvénient reste temps d'exécution qui est trop long, et ceci à cause de la faiblesse de la machine.

Nous allons essayer une autre approche qui est connue sous le nom de « feature extraction » en français extraction de caractéristiques qui est un moyen facile et rapide à utiliser, la puissance de l'apprentissage profond sans investir temps et effort dans un réseau complet de formation. Parce qu'il exige seulement un seul passage sur les images de la formation, nous allons extraire les fonctionnalités d'image apprises à l'aide d'un réseau pré-entraîné, puis utiliser ces fonctionnalités pour former un classificateur, tel que des machines à vecteurs de support (SVM). [13]

III.6.2. Classification avec SVM:

Une machine à vecteurs de support (SVM) est un algorithme d'apprentissage supervisé qui peut être utilisé pour la classification ou la régression binaire. Qui construit un hyperplan optimal en tant que surface de décision de manière à maximiser la marge de séparation entre les deux classes des données. Les vecteurs de support font référence à un petit sous-ensemble des observations d'entraînement utilisées comme support pour l'emplacement optimal de la surface de décision. [19]

III.6.1. Extraction des caractéristiques Image

Puisque toutes les couches sont chargées d'apprendre certaines caractéristiques à partir des images, nous pouvons récupérer ces caractéristiques depuis le réseau à n'importe quel moment du processus d'entraînement. Ensuite utiliser ces caractéristiques en tant que données d'entrée pour un modèle de classification avec les machines à vecteurs de support (SVM). [19]

Comme pour la méthode précédente nous allons charger les bases de données redimensionner les images et les deviser et deux partie dont 70% pour l'apprentissage et 30% pour le test

Ensuite nous allons extraire les étiquettes de classe de l'apprentissage et les données de test. [14]

III.6.2. Ajustement du classifieur

Nous utilisons un réseau pré-entraîné comme extracteur de caractéristiques en utilisant les couches d'activations comme caractéristiques. Et utiliser les fonctions extraites à partir des images de l'apprentissage comme variables prédictives et monter une machines à vecteurs de support (SVM) à l'aide de `fitcecoc` (statistiques et Machine Learning Toolbox).

III.6.3. Classification des Images de Test :

Support Vector Machine est un algorithme d'apprentissage automatique supervisé qui est principalement utilisé dans les problèmes de classification. Dans cet algorithme, nous plaçons chaque donnée sous forme de point dans un espace à n dimensions (où n est le nombre de caractéristiques que vous avez), la valeur de chaque caractéristique étant la valeur d'une coordonnée particulière. Ensuite, nous effectuons la classification en recherchant l'hyperplan qui différencie très bien les deux classes.

Nous allons procéder a une classification des images de test en utilisant le modèle SVM formé les caractéristiques extraites à partir des images de test.et enfin afficher des images de test échantillon avec leurs étiquettes prédites. Sans oublier de calculer le taux d'apprentissage (accuracy) [14]

Le tableau ci-dessus affiche les résultats avec nos 4 bases de données

BDD	taux d'apprentissage
BDD2	**86,67%**
BDD3	**93,33%**
BDD4	**93,33%**
BDD5	**100%**

Tableau 3. 6. taux d'apprentissage pour les différentes BDD avec l'extraction des caractéristiques pour une classification avec les SVM

Comme pour la 1er méthode nous allons exécuter le programme pour la 5ém base de données qui comporte 200 classes pour chaque classe nous avons 20 images de l'œil gauche et droit sans aucun traitement

Nous présentons des résultats qui montrent qu'il est désormais possible d'utiliser les SVM pour résoudre des problèmes avec des millions d'exemples.

Figure 3.29. Échantillons des images de la BDD 5

Nous avons obtenu un taux d'apprentissage de 98.84 % comparé à la méthode de la classification avec un modèle prés-entrainer, en utilisant la classification avec SVM on obtient de bon résultat mais surtout plus rapidement.

III.7. Conclusion :

Dans ce chapitre, nous avons décrit la démarche à suivre afin d'obtenir le taux le plus précis possible. Nous avons vu le fonctionnement de deux modèles pré-entrainé pour la classification des images et comment les ajuster afin de faire une reconnaissance de l'iris, Nous avons fait changer les paramètres un à un afin de comprendre l'impact de chacun sur la performance finale. En utilisant l'apprentissage par transfert, aussi nous avons travaillé avec deux méthodes ; la classification par le modèle pré-entrainé et la classification avec SVM pour que l'on puisse avoir un bon résultat, nous avons conclue qu'avec l'AlexNet nous avons un bien meilleur résultat par rapport à GoogleNet

IV Conclusion Générale :

Dans ce travail nous nous sommes consacrés et la reconnaissance de l'iris par les CNN, nous avons vu qu'il exister plusieurs caractéristiques physiques qui peuvent être utilisées pour identifier les individus, mais la plus fiable était l'iris, nous avons aussi vu le fonctionnement d'un CNN, Les progrès récents en apprentissage profond et en vision par ordinateur indiquent que les descripteurs génériques extraits à l'aide de réseaux de neurones à convolution (CNN) sont capables de représenter des caractéristiques d'image complexes. Étant donné les performances supérieures des CNN dans le cadre du défi de reconnaissance visuelle à grande échelle. Nous avons utilisé deux modèles prés-entrainés sur plusieurs bases de données en faisant une classification des images de deux manières et nous avons eu des résultats satisfaisant voir bons pour les images sans aucun près/post traitement ce qui n'est pas le cas pour un classifier classique où on est obligé d'isoler l'iris, mais il consomme beaucoup de temps en apprentissage.

Les paramètres qui vont jouer sur l'amélioration du taux d'apprentissage ;

Nombre d'époque, la taille du minibatch, la taille de la base d'apprentissage et la nature du modèle,

En perspective nous proposons de:

- Augmenter la base de données
- Tester d'autres modèles CNN prédéfinis.
- Proposer un nouveau modèle en jouant sur la profondeur du modèle

Pour résoudre le problème de temps d'apprentissage qui est trop grand, il est conseillé d'utiliser les GPU qui sont les processeurs adaptés pour entraîner des réseaux de neurones convolutionnels. Initialement développés pour les opérations graphiques, NVIDIA a lancé en 2007 des GPU destinés à des calculs plus génériques. Ces GPU accélèrent les portions de code intensives en calculs parallélisables, les portions séquentielles restantes étant traitées par les CPU. Les réseaux de neurones nécessitent de nombreuses opérations matricielles, qui sont accélérées d'un facteur de dix lorsqu'elles sont distribuées sur les milliers de cœurs de GPU.

V. Références :

[1] : https://labiometrie.wordpress.com/2017/02/11/reconnaissance-de-liris/ consultée le 10/04/2019

[2] : http://www.univ-usto.dz/theses_en_ligne/doc_num.php?explnum_id=410 consulté le 11/04/2019

[3] : J. R. Matey, O. Naroditsky, K. Hanna, R. Kolczynski, D. J. LoIacono, S. Mangru, M. Tinker, T. M. Zappia, andW. Y. Zhao, ―Iris on the move: Acquisition of images for iris recognition in less constrained environments,‖ Proceedings of the IEEE, vol. 94, pp.1936–1947, November 2006.

[4] : http://biblio.univ-annaba.dz/ingeniorat/wp-content/uploads/2017/11/lekmiti-soumia-m%C3%A9moire.pdf consulté le 13/04/2019

[5] : https://tel.archives-ouvertes.fr/tel-01591556/document#subsection.1.3.2

[6] : https://medium.com/@CharlesCrouspeyre/comment-les-r%C3%A9seaux-de-neurones-%C3%A0-convolution-fonctionnent-b288519dbcf8 consulté le 13/04/2019

[7] : http://penseeartificielle.fr/focus-reseau-neurones-convolutifs/ consulté le 15/04/2019

[8] : https://indoml.com/2018/03/07/student-notes-convolutional-neural-networks-cnn-introduction consulté le 15/04/2019

[9] : R.DonidaLabati,V.Piuri,F.Scotti,Agent-based image iris segmentation and multiple views boundary refining, in:Proceeding sof the IEEE Third Interna- tional Conferenceon Biometrics :Theory,ApplicationsandSystems,November 20, 2009

[10] :https://riunet.upv.es/bitstream/handle/10251/35155/PachecoLloret_ManuelaBego%C3%B1a.pdf?sequence=1 consultée le 20/04/2019

[11] :https://fr.mathworks.com/help/deeplearning/ref/alexnet.html;jsessionid=e8d6a713f38e13d2f4b505370a98 consultée le 09/05/2019

[12] : https://fr.mathworks.com/help/deeplearning/ref/trainingoptions.html consultée le 25/04/2019

[13] : https://fr.mathworks.com/help/deeplearning/ug/pretrained-convolutional-neural-networks.html consultée le 27/04/2019 consultée le 12/05/2019

[14] :https://fr.mathworks.com/help/deeplearning/ref/alexnet.html?searchHighlight=alexNet%20feature%20extraction&s_tid=doc_srchtitle consultée le 12/05/2019

[15] : https://fr.mathworks.com/help/deeplearning/examples/transfer-learning-using-alexnet.html?searchHighlight=alexNet&s_tid=doc_srchtitle consultée le 12/05/2019

[16] : https://www.academia.edu/34938587/Neural_Network_Toolbox_Users_Guide .consulté le 20/05/2019

[17] : https://www.learnopencv.com/understanding-alexnet/ Consulté le 22/05/2019

[18] : http://dspace.univ-tlemcen.dz/bitstream/112/12235/1/Classification-des-images-avec-les-reseaux-de-neurones.pdf Consulté le 23/05/2019

[19] : https://fr.mathworks.com/discovery/support-vector-machine.html Consulté le 23/05/2019

[20] : http://www.cbsr.ia.ac.cn/english/IrisDatabase.asp consulté le 01/06/2019

[22] : https://blog.octo.com/classification-dimages-les-reseaux-de-neurones-convolutifs-en-toute-simplicite/ consulté le 04/06/2019

[23] : https://developers.google.com/machine-learning/crash-course/multi-class-neural-networks/softmax?hl=fr : consulté le 06/06/2019

VI. Annexe :

ANALYSIS RESULT

↑	NAME	TYPE	ACTIVATIONS	LEARNABLES
1	data 227x227x3 images with 'zerocenter' normalization	Image Input	227×227×3	-
2	conv1 96 11x11x3 convolutions with stride [4 4] and padding [0 0 0 0]	Convolution	55×55×96	Weights 11×11×3×96 Bias 1×1×96
3	relu1 ReLU	ReLU	55×55×96	-
4	norm1 cross channel normalization with 5 channels per element	Cross Channel Nor…	55×55×96	-
5	pool1 3x3 max pooling with stride [2 2] and padding [0 0 0 0]	Max Pooling	27×27×96	-
6	conv2 256 5x5x48 convolutions with stride [1 1] and padding [2 2 2 2]	Convolution	27×27×256	Weights 5×5×48×256 Bias 1×1×256
7	relu2 ReLU	ReLU	27×27×256	-
8	norm2 cross channel normalization with 5 channels per element	Cross Channel Nor…	27×27×256	-
9	pool2 3x3 max pooling with stride [2 2] and padding [0 0 0 0]	Max Pooling	13×13×256	-
10	conv3 384 3x3x256 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	13×13×384	Weights 3×3×256×38 Bias 1×1×384
11	relu3 ReLU	ReLU	13×13×384	-
12	conv4 384 3x3x192 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	13×13×384	Weights 3×3×192×38 Bias 1×1×384
13	relu4 ReLU	ReLU	13×13×384	-
14	conv5 256 3x3x192 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	13×13×256	Weights 3×3×192×25 Bias 1×1×256
15	relu5 ReLU	ReLU	13×13×256	-
16	pool5 3x3 max pooling with stride [2 2] and padding [0 0 0 0]	Max Pooling	6×6×256	-
17	fc6 4096 fully connected layer	Fully Connected	1×1×4096	Weights 4096×9216 Bias 4096×1
18	relu6 ReLU	ReLU	1×1×4096	-
19	drop6 50% dropout	Dropout	1×1×4096	-
20	fc7 4096 fully connected layer	Fully Connected	1×1×4096	Weights 4096×4096 Bias 4096×1
21	relu7 ReLU	ReLU	1×1×4096	-
22	drop7 50% dropout	Dropout	1×1×4096	-
23	fc8 1000 fully connected layer	Fully Connected	1×1×1000	Weights 1000×4096 Bias 1000×1
24	prob softmax	Softmax	1×1×1000	-
25	output crossentropyex with 'tench' and 999 other classes	Classification Output	-	-

Annexe 1: Architecture et détail du modèle AlexNet

Annexe 2 : Architecture et détails du modèle GoogleNet

	NAME	TYPE	ACTIVATIONS	LEARNABLES
1	data 224x224x3 images with 'zerocenter' normalization	Image Input	224×224×3	-
2	conv1-7x7_s2 64 7x7x3 convolutions with stride [2 2] and padding [3 3 3 3]	Convolution	112×112×64	Weights 7×7×3×64 Bias 1×1×64
3	conv1-relu_7x7 ReLU	ReLU	112×112×64	-
4	pool1-3x3_s2 3x3 max pooling with stride [2 2] and padding [0 1 0 1]	Max Pooling	56×56×64	-
5	pool1-norm1 cross channel normalization with 5 channels per element	Cross Channel Nor…	56×56×64	-
6	conv2-3x3_reduce 64 1x1x64 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	56×56×64	Weights 1×1×64×64 Bias 1×1×64
7	conv2-relu_3x3_reduce ReLU	ReLU	56×56×64	-
8	conv2-3x3 192 3x3x64 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	56×56×192	Weights 3×3×64×192 Bias 1×1×192
9	conv2-relu_3x3 ReLU	ReLU	56×56×192	-
10	conv2-norm2 cross channel normalization with 5 channels per element	Cross Channel Nor…	56×56×192	-
11	pool2-3x3_s2 3x3 max pooling with stride [2 2] and padding [0 1 0 1]	Max Pooling	28×28×192	-
12	inception_3a-1x1 64 1x1x192 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	28×28×64	Weights 1×1×192×64 Bias 1×1×64
13	inception_3a-relu_1x1 ReLU	ReLU	28×28×64	-
14	inception_3a-3x3_reduce 96 1x1x192 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	28×28×96	Weights 1×1×192×96 Bias 1×1×96
15	inception_3a-relu_3x3_reduce ReLU	ReLU	28×28×96	-
16	inception_3a-3x3 128 3x3x96 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	28×28×128	Weights 3×3×96×128 Bias 1×1×128
17	inception_3a-relu_3x3 ReLU	ReLU	28×28×128	-
18	inception_3a-5x5_reduce 16 1x1x192 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	28×28×16	Weights 1×1×192×16 Bias 1×1×16
19	inception_3a-relu_5x5_reduce ReLU	ReLU	28×28×16	-
20	inception_3a-5x5 32 5x5x16 convolutions with stride [1 1] and padding [2 2 2 2]	Convolution	28×28×32	Weights 5×5×16×32 Bias 1×1×32
21	inception_3a-relu_5x5 ReLU	ReLU	28×28×32	-
22	inception_3a-pool 3x3 max pooling with stride [1 1] and padding [1 1 1 1]	Max Pooling	28×28×192	-
23	inception_3a-pool_proj 32 1x1x192 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	28×28×32	Weights 1×1×192×32 Bias 1×1×32
24	inception_3a-relu_pool_proj ReLU	ReLU	28×28×32	-
25	inception_3a-output Depth concatenation of 4 inputs	Depth concatenation	28×28×256	-
26	inception_3b-1x1 128 1x1x256 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	28×28×128	Weights 1×1×256×128 Bias 1×1×128
27	inception_3b-relu_1x1 ReLU	ReLU	28×28×128	-
28	inception_3b-3x3_reduce 128 1x1x256 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	28×28×128	Weights 1×1×256×128 Bias 1×1×128
29	inception_3b-relu_3x3_reduce ReLU	ReLU	28×28×128	-
30	inception_3b-3x3 192 3x3x128 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	28×28×192	Weights 3×3×128×192 Bias 1×1×192
31	inception_3b-relu_3x3 ReLU	ReLU	28×28×192	-
32	inception_3b-5x5_reduce 32 1x1x256 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	28×28×32	Weights 1×1×256×32 Bias 1×1×32
33	inception_3b-relu_5x5_reduce ReLU	ReLU	28×28×32	-
34	inception_3b-5x5 96 5x5x32 convolutions with stride [1 1] and padding [2 2 2 2]	Convolution	28×28×96	Weights 5×5×32×96 Bias 1×1×96
35	inception_3b-relu_5x5 ReLU	ReLU	28×28×96	-
36	inception_3b-pool 3x3 max pooling with stride [1 1] and padding [1 1 1 1]	Max Pooling	28×28×256	-
37	inception_3b-pool_proj 64 1x1x256 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	28×28×64	Weights 1×1×256×64 Bias 1×1×64
38	inception_3b-relu_pool_proj ReLU	ReLU	28×28×64	-
39	inception_3b-output Depth concatenation of 4 inputs	Depth concatenation	28×28×480	-
40	pool3-3x3_s2 3x3 max pooling with stride [2 2] and padding [0 1 0 1]	Max Pooling	14×14×480	-
41	inception_4a-1x1 192 1x1x480 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	14×14×192	Weights 1×1×480×192 Bias 1×1×192
42	inception_4a-relu_1x1 ReLU	ReLU	14×14×192	-
43	inception_4a-3x3_reduce 96 1x1x480 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	14×14×96	Weights 1×1×480×96 Bias 1×1×96
44	inception_4a-relu_3x3_reduce ReLU	ReLU	14×14×96	-
45	inception_4a-3x3 208 3x3x96 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	14×14×208	Weights 3×3×96×208 Bias 1×1×208
46	inception_4a-relu_3x3 ReLU	ReLU	14×14×208	-
47	inception_4a-5x5_reduce 16 1x1x480 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	14×14×16	Weights 1×1×480×16 Bias 1×1×16
48	inception_4a-relu_5x5_reduce ReLU	ReLU	14×14×16	-
49	inception_4a-5x5 48 5x5x16 convolutions with stride [1 1] and padding [2 2 2 2]	Convolution	14×14×48	Weights 5×5×16×48 Bias 1×1×48
50	inception_4a-relu_5x5 ReLU	ReLU	14×14×48	-
51	inception_4a-pool 3x3 max pooling with stride [1 1] and padding [1 1 1 1]	Max Pooling	14×14×480	-
52	inception_4a-pool_proj 64 1x1x480 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	14×14×64	Weights 1×1×480×64 Bias 1×1×64
53	inception_4a-relu_pool_proj ReLU	ReLU	14×14×64	-
54	inception_4a-output Depth concatenation of 4 inputs	Depth concatenation	14×14×512	-
55	inception_4b-1x1 160 1x1x512 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	14×14×160	Weights 1×1×512×160 Bias 1×1×160

Annexe 2 : Architecture et détails du modèle GoogleNet (suite)

55	inception_4b-1x1 160 1x1x512 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	14×14×160	Weights 1×1×512×160 Bias 1×1×160
56	inception_4b-relu_1x1 ReLU	ReLU	14×14×160	-
57	inception_4b-3x3_reduce 112 1x1x512 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	14×14×112	Weights 1×1×512×112 Bias 1×1×112
58	inception_4b-relu_3x3_reduce ReLU	ReLU	14×14×112	-
59	inception_4b-3x3 224 3x3x112 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	14×14×224	Weights 3×3×112×224 Bias 1×1×224
60	inception_4b-relu_3x3 ReLU	ReLU	14×14×224	-
61	inception_4b-5x5_reduce 24 1x1x512 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	14×14×24	Weights 1×1×512×24 Bias 1×1×24
62	inception_4b-relu_5x5_reduce ReLU	ReLU	14×14×24	-
63	inception_4b-5x5 64 5x5x24 convolutions with stride [1 1] and padding [2 2 2 2]	Convolution	14×14×64	Weights 5×5×24×64 Bias 1×1×64
64	inception_4b-relu_5x5 ReLU	ReLU	14×14×64	-
65	inception_4b-pool 3x3 max pooling with stride [1 1] and padding [1 1 1 1]	Max Pooling	14×14×512	-
66	inception_4b-pool_proj 64 1x1x512 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	14×14×64	Weights 1×1×512×64 Bias 1×1×64
67	inception_4b-relu_pool_proj ReLU	ReLU	14×14×64	-
68	inception_4b-output Depth concatenation of 4 inputs	Depth concatenation	14×14×512	-
69	inception_4c-1x1 128 1x1x512 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	14×14×128	Weights 1×1×512×128 Bias 1×1×128
70	inception_4c-relu_1x1 ReLU	ReLU	14×14×128	-
71	inception_4c-3x3_reduce 128 1x1x512 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	14×14×128	Weights 1×1×512×128 Bias 1×1×128
72	inception_4c-relu_3x3_reduce ReLU	ReLU	14×14×128	-
73	inception_4c-3x3 256 3x3x128 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	14×14×256	Weights 3×3×128×256 Bias 1×1×256
74	inception_4c-relu_3x3 ReLU	ReLU	14×14×256	-
75	inception_4c-5x5_reduce 24 1x1x512 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	14×14×24	Weights 1×1×512×24 Bias 1×1×24
76	inception_4c-relu_5x5_reduce ReLU	ReLU	14×14×24	-
77	inception_4c-5x5 64 5x5x24 convolutions with stride [1 1] and padding [2 2 2 2]	Convolution	14×14×64	Weights 5×5×24×64 Bias 1×1×64
78	inception_4c-relu_5x5 ReLU	ReLU	14×14×64	-
79	inception_4c-pool 3x3 max pooling with stride [1 1] and padding [1 1 1 1]	Max Pooling	14×14×512	-
80	inception_4c-pool_proj 64 1x1x512 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	14×14×64	Weights 1×1×512×64 Bias 1×1×64
81	inception_4c-relu_pool_proj ReLU	ReLU	14×14×64	-
82	inception_4c-output Depth concatenation of 4 inputs	Depth concatenation	14×14×512	-
83	inception_4d-1x1 112 1x1x512 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	14×14×112	Weights 1×1×512×112 Bias 1×1×112
84	inception_4d-relu_1x1 ReLU	ReLU	14×14×112	-
85	inception_4d-3x3_reduce 144 1x1x512 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	14×14×144	Weights 1×1×512×144 Bias 1×1×144
86	inception_4d-relu_3x3_reduce ReLU	ReLU	14×14×144	-
87	inception_4d-3x3 288 3x3x144 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	14×14×288	Weights 3×3×144×288 Bias 1×1×288
88	inception_4d-relu_3x3 ReLU	ReLU	14×14×288	-
89	inception_4d-5x5_reduce 32 1x1x512 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	14×14×32	Weights 1×1×512×32 Bias 1×1×32
90	inception_4d-relu_5x5_reduce ReLU	ReLU	14×14×32	-
91	inception_4d-5x5 64 5x5x32 convolutions with stride [1 1] and padding [2 2 2 2]	Convolution	14×14×64	Weights 5×5×32×64 Bias 1×1×64
92	inception_4d-relu_5x5 ReLU	ReLU	14×14×64	-
93	inception_4d-pool 3x3 max pooling with stride [1 1] and padding [1 1 1 1]	Max Pooling	14×14×512	-
94	inception_4d-pool_proj 64 1x1x512 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	14×14×64	Weights 1×1×512×64 Bias 1×1×64
95	inception_4d-relu_pool_proj ReLU	ReLU	14×14×64	-
96	inception_4d-output Depth concatenation of 4 inputs	Depth concatenation	14×14×528	-
97	inception_4e-1x1 256 1x1x528 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	14×14×256	Weights 1×1×528×256 Bias 1×1×256
98	inception_4e-relu_1x1 ReLU	ReLU	14×14×256	-
99	inception_4e-3x3_reduce 160 1x1x528 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	14×14×160	Weights 1×1×528×160 Bias 1×1×160
100	inception_4e-relu_3x3_reduce ReLU	ReLU	14×14×160	-
101	inception_4e-3x3 320 3x3x160 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	14×14×320	Weights 3×3×160×320 Bias 1×1×320
102	inception_4e-relu_3x3 ReLU	ReLU	14×14×320	-
103	inception_4e-5x5_reduce 32 1x1x528 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	14×14×32	Weights 1×1×528×32 Bias 1×1×32
104	inception_4e-relu_5x5_reduce ReLU	ReLU	14×14×32	-
105	inception_4e-5x5 128 5x5x32 convolutions with stride [1 1] and padding [2 2 2 2]	Convolution	14×14×128	Weights 5×5×32×128 Bias 1×1×128
106	inception_4e-relu_5x5 ReLU	ReLU	14×14×128	-
107	inception_4e-pool 3x3 max pooling with stride [1 1] and padding [1 1 1 1]	Max Pooling	14×14×528	-
108	inception_4e-pool_proj 128 1x1x528 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	14×14×128	Weights 1×1×528×128 Bias 1×1×128
109	inception_4e-relu_pool_proj ReLU	ReLU	14×14×128	-
110	inception_4e-output Depth concatenation of 4 inputs	Depth concatenation	14×14×832	-

Annexe 2 : Architecture et détails du modèle GoogleNet (suite)

111	pool4-3x3_s2 3x3 max pooling with stride [2 2] and padding [0 1 0 1]	Max Pooling	7×7×832	-
112	inception_5a-1x1 256 1x1x832 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	7×7×256	Weights 1×1×832×256 Bias 1×1×256
113	inception_5a-relu_1x1 ReLU	ReLU	7×7×256	-
114	inception_5a-3x3_reduce 160 1x1x832 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	7×7×160	Weights 1×1×832×160 Bias 1×1×160
115	inception_5a-relu_3x3_reduce ReLU	ReLU	7×7×160	-
116	inception_5a-3x3 320 3x3x160 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	7×7×320	Weights 3×3×160×320 Bias 1×1×320
117	inception_5a-relu_3x3 ReLU	ReLU	7×7×320	-
118	inception_5a-5x5_reduce 32 1x1x832 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	7×7×32	Weights 1×1×832×32 Bias 1×1×32
119	inception_5a-relu_5x5_reduce ReLU	ReLU	7×7×32	-
120	inception_5a-5x5 128 5x5x32 convolutions with stride [1 1] and padding [2 2 2 2]	Convolution	7×7×128	Weights 5×5×32×128 Bias 1×1×128
121	inception_5a-relu_5x5 ReLU	ReLU	7×7×128	-
122	inception_5a-pool 3x3 max pooling with stride [1 1] and padding [1 1 1 1]	Max Pooling	7×7×832	-
123	inception_5a-pool_proj 128 1x1x832 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	7×7×128	Weights 1×1×832×128 Bias 1×1×128
124	inception_5a-relu_pool_proj ReLU	ReLU	7×7×128	-
125	inception_5a-output Depth concatenation of 4 inputs	Depth concatenation	7×7×832	-
126	inception_5b-1x1 384 1x1x832 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	7×7×384	Weights 1×1×832×384 Bias 1×1×384
127	inception_5b-relu_1x1 ReLU	ReLU	7×7×384	-
128	inception_5b-3x3_reduce 192 1x1x832 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	7×7×192	Weights 1×1×832×192 Bias 1×1×192
129	inception_5b-relu_3x3_reduce ReLU	ReLU	7×7×192	-
130	inception_5b-3x3 384 3x3x192 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	7×7×384	Weights 3×3×192×384 Bias 1×1×384
131	inception_5b-relu_3x3 ReLU	ReLU	7×7×384	-
132	inception_5b-5x5_reduce 48 1x1x832 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	7×7×48	Weights 1×1×832×48 Bias 1×1×48
133	inception_5b-relu_5x5_reduce ReLU	ReLU	7×7×48	-
134	inception_5b-5x5 128 5x5x48 convolutions with stride [1 1] and padding [2 2 2 2]	Convolution	7×7×128	Weights 5×5×48×128 Bias 1×1×128
135	inception_5b-relu_5x5 ReLU	ReLU	7×7×128	-
136	inception_5b-pool 3x3 max pooling with stride [1 1] and padding [1 1 1 1]	Max Pooling	7×7×832	-
137	inception_5b-pool_proj 128 1x1x832 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	7×7×128	Weights 1×1×832×128 Bias 1×1×128
138	inception_5b-relu_pool_proj ReLU	ReLU	7×7×128	-
139	inception_5b-output Depth concatenation of 4 inputs	Depth concatenation	7×7×1024	-
140	pool5-7x7_s1 7x7 average pooling with stride [1 1] and padding [0 0 0 0]	Average Pooling	1×1×1024	-
141	pool5-drop_7x7_s1 40% dropout	Dropout	1×1×1024	-
142	fc 10 fully connected layer	Fully Connected	1×1×10	Weights 10×1024 Bias 10×1
143	softmax softmax	Softmax	1×1×10	-
144	classoutput crossentropyex with '001' and 9 other classes	Classification Output	-	-

Annexe 2 : Architecture et détails du modèle GoogleNet (suite)