# MISE AU POINT D'UN SYSTÈME DE DÉTECTION DU STRABISME OCULAIRE AVEC LE MODÈLE YOLOV4

Par

Vite Marylise METOUNA ONGODO

MÉMOIRE PRÉSENTÉ À L'UNIVERSITÉ DU QUÉBEC À TROISRIVIÈRES COMME EXIGENCE PARTIELLE DE LA MAITRISE EN MATHÉMATIQUE ET INFORMATIQUE APPLIQUÉES AVEC MÉMOIRE

M. Sc. A.

TROIS-RIVIÈRES, LE 21 MARS 2025

UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES

# PRÉSENTATION DU JURY

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE :

M. Jean-Sebastien Dessureault, Ph.D., directeur de mémoire
Département de Mathématique et Informatique à l'université du Québec à TroisRivières

M. Jean-François Quessy, Ph.D., ing.
Département de Mathématique et Informatique à l'université du Québec à TroisRivières

M. William Flageol, Ph.D.
Département de Mathématique et Informatique à l'université du Québec à TroisRivières

# REMERCIEMENT

Je remercie Dieu qui m’a permis de réaliser ce projet d’études en général et en particulier ce projet de mémoire.

Je remercie ma famille pour son soutien tout au long de mes études.

Je remercie l’Université du Québec à Trois-Rivières pour cette opportunité d’études offerte.

Je remercie mon directeur de recherche et l’ensemble de tous les professeurs du département de mathématiques et d'informatique.

Je remercie mes amis et toute personne m’ayant apporté un soutien durant mon cursus.

# RÉSUMÉ

L'intelligence artificielle (IA) fait l'objet de plusieurs études, car elle est considérée comme une solution probable au problème de détection précoce des pathologies oculaires. Afin de mieux implémenter ce dispositif, il est idéal de recourir à l'apprentissage profond, qui est un sous-domaine de l'IA. Plusieurs méthodes de détection existent pour l'identification des maladies de l'œil, notamment le strabisme. Elles sont soit traditionnelles, comme l'examen clinique et la photographie rétinienne, soit basée sur l'IA, telles que l'analyse d'images assistée par ordinateur ou l'apprentissage profond. Toutes ces approches ont pour objectif commun de mettre en évidence les caractéristiques qui différencient un œil normal d'un œil atteint. Grâce aux recherches et expérimentations menées, un système de détection précoce du strabisme oculaire a été mis en place. Ce dispositif peut être utilisé aussi bien dans les ménages que dans un contexte clinique, en s'appuyant sur le modèle populaire YOLOV4 (*You Only Look Once*), un algorithme de détection d'objets en temps réel, rapide et précis. Dans ce système, le sujet se place devant un écran, et les images sont capturées par une caméra. Le système détecte ensuite les yeux et effectue une prédiction, déterminant s'il s'agit d'un cas normal ou de strabisme. Ce dispositif pourrait contribuer à l'évolution de la télémédecine en ophtalmologie.

# ABSTRACT

Artificial intelligence (AI) has been the subject of numerous studies, as it is considered a probable solution to the problem of early detection of ocular pathologies. To better implement this system, it is ideal to use deep learning (DL), a subfield of AI. Several detection methods exist for identifying eye diseases, particularly strabismus. These methods are either traditional, such as clinical examination and retinal photography, or based on AI, such as computerassisted image analysis or DL. All these approaches share the common goal of highlighting the characteristics that differentiate a normal eye from an affected one. Thanks to the research and experiments conducted, an early detection system for ocular strabismus has been developed. This system can be used both in households and in clinical settings, relying on the popular YOLOV4 model, a fast and accurate real-time object detection algorithm. In this system, the subject stands in front of a screen, and images are captured by a camera. The system then detects the eyes and makes a prediction, determining whether it is a normal case or strabismus.

This device could contribute to the evolution of telemedicine in ophthalmology.

# TABLE DES MATIÈRES

# LISTE DES FIGURES

# LISTE DES TABLEAUX

# LISTE DES ABRÉVIATIONS

| | |
|---|---|
| AI | Artificial Intelligence |
| AVG | Average |
| BFLOPs | Billions of Floating-Point Operations |
| CIoU | Complete Intersection Over Union |
| CLR | Corneal Light Reflex Ratio |
| CPU | Central Processing Unit |
| CSP | Cross Stage Partial Connections |
| CSPDarknet | Cross Stage Partial Darknet |
| CSPNet | Cross Stage Partial Network |
| DL | Deep Learning |
| FLOPs | Floating Point Operations per Second |
| FPN | Feature Pyramid Network |
| FPS | Frames Per Second |
| GPU | Graphics Processing Unit |
| HSV | Hue, Saturation, Value (Color Model) |
| IOU | Intersection Over Union |
| IOU_Avg | Intersection Over Union Average |
| IA | Intelligence Artificielle |
| LSM | Least Squares Method |
| ML | Machine Learning |

| | |
|---|---|
| NMS | Non-Maximum Suppression |
| NVIDIA | GPU hardware manufacturer |
| PAN | Path Aggregation Network |
| PCA | Principal Component Analysis |
| R-CNN | Region-based Convolutional Neural Network |
| R-FCN | Region-based Fully Convolutional Network |
| RF-CNN | Random Forest Convolutional Neural Network |
| ROI | Regions of Interest |
| RPN | Region Proposal Network |
| SPP | Spatial Pyramid Pooling |
| SSD | Single Shot Detector |
| SVM | Support Vector Machine |
| WRC | Weighted Residual Connections |
| YOLO | You Only Look Once |
| YOLOV4 | You Only Look Once Version 4 |
| cuDNN | CUDA Deep Neural Network |

# CHAPITRE 1. INTRODUCTION

L'évolution rapide des technologies de l'IA a transformé de manière significative divers secteurs de la santé, ouvrant des perspectives prometteuses pour l'amélioration des diagnostics et des traitements. Cependant, malgré ces avancées, des lacunes persistent dans certains domaines, notamment dans le dépistage précoce des pathologies oculaires telles que le strabisme. Le strabisme, une anomalie courante caractérisée par le désalignement des yeux, représente un défi diagnostique majeur, particulièrement en ce qui concerne son identification précoce, qui est essentielle pour prévenir des complications visuelles irréversibles.

Actuellement, les méthodes de détection du strabisme reposent principalement sur des techniques cliniques traditionnelles, comme le test de Hirschberg et l'examen de l'acuité visuelle. Bien que ces approches aient fait leurs preuves, elles nécessitent l'intervention de spécialistes, peuvent être coûteuses et sont souvent inaccessibles dans les zones à faible densité médicale. De plus, les subtilités du strabisme peuvent échapper à ces méthodes, entraînant des retards dans le diagnostic et le traitement. Ces limitations révèlent la nécessité d'une solution innovante, plus accessible et automatisée, capable de diagnostiquer cette pathologie avec précision et rapidité.

Dans ce contexte, l'IA, et plus spécifiquement l'apprentissage automatique, offre un potentiel considérable pour surmonter ces obstacles. Avec ses capacités avancées de traitement d'images, l'IA permet de concevoir des systèmes de détection efficaces, capables de réaliser des diagnostics précis en temps réel. Parmi les modèles les plus performants dans ce domaine, YOLOv4 [56] se distingue par sa rapidité et sa précision dans la détection d'objets complexes, comme les yeux, tout en s'adaptant aux contraintes des applications médicales.

Face à ces défis, une question fondamentale se pose : dans quelle mesure le modèle de détection d'objets **YOLOv4** peut-il identifier le strabisme oculaire avec une précision égale ou supérieure à celle des méthodes cliniques traditionnelles? Nous formulons l'hypothèse que le modèle YOLOv4 est capable de détecter efficacement le strabisme avec une précision équivalente ou supérieure aux méthodes cliniques traditionnelles. En s'appuyant sur une base de données d'images médicales et un entraînement optimisé, ce modèle pourrait permettre un dépistage plus rapide et standardisé, réduisant ainsi les risques de retard dans la prise en charge des patients.

L'intégration de YOLOv4 dans le diagnostic du strabisme ouvre également des perspectives importantes pour la télémédecine. Grâce à sa rapidité et sa capacité à analyser des images en temps réel, ce modèle pourrait être utilisé dans des plateformes de dépistage à distance, réduisant ainsi la dépendance aux spécialistes et facilitant l'accès au diagnostic pour les patients

vivant dans des zones médicalement sous-dotées. Une telle avancée pourrait non seulement améliorer la prise en charge précoce du strabisme, mais aussi optimiser les ressources en santé visuelle en permettant une orientation plus efficace vers les soins spécialisés.

Ce mémoire a pour principal objectif de développer un système de détection automatique du strabisme à l'aide du modèle de vision par ordinateur YOLOv4. Ce modèle sera entraîné pour identifier avec précision les signes de strabisme dans des images médicales, afin de proposer une méthode de dépistage plus rapide, plus fiable et accessible que les techniques traditionnelles. Pour atteindre cet objectif principal, il sera nécessaire d'analyser les techniques actuelles de détection du strabisme et leurs limites, d'adapter le modèle YOLOv4 au domaine de la vision médicale, de constituer une base de données adaptée à la reconnaissance du strabisme et d'évaluer les performances du modèle en fonction de critères standards tels que la précision et le rappel. Il conviendra également d'examiner ses implications cliniques en comparaison avec les méthodes existantes.

Le choix de ce sujet s'explique par la convergence de plusieurs éléments scientifiques et pratiques. D'une part, le strabisme est une pathologie oculaire fréquente qui, en l'absence d'un dépistage précoce et d'une prise en charge adéquate, peut entraîner des conséquences graves sur la santé visuelle des patients, notamment chez les enfants. La détection précoce de cette anomalie est donc essentielle pour prévenir des complications comme l'amblyopie et les troubles de la perception binoculaire, qui peuvent affecter durablement la qualité de vie. D'autre part, les méthodes traditionnelles de dépistage du strabisme, bien qu'efficaces, reposent souvent sur des examens manuels réalisés par des spécialistes. Or, ces méthodes sont coûteuses, prennent du temps et peuvent être limitées par la disponibilité d'experts, surtout dans les régions où l'accès à des soins ophtalmologiques est restreint. Il est donc nécessaire de proposer des solutions plus accessibles et standardisées, capables de pallier ces limitations. Sur le plan scientifique, l'IA et, plus spécifiquement, les modèles de vision par ordinateur tels que YOLOv4 offrent des perspectives prometteuses pour l'automatisation de la détection de diverses anomalies médicales. Cependant, malgré leur succès dans d'autres domaines, l'application de ces modèles à la détection du strabisme reste encore peu étudiée, ce qui constitue une opportunité de recherche pertinente. Ce mémoire se propose donc de combler cette lacune en adaptant YOLOv4 à la détection de cette pathologie, avec l'ambition de proposer une solution capable d'améliorer la précision des diagnostics tout en offrant un processus potentiellement plus accessible et rapide. Enfin, la pertinence pratique de ce sujet réside dans son potentiel à transformer le dépistage du strabisme en une procédure plus rapide, standardisée et accessible,

contribuant ainsi à l'amélioration des soins oculaires et à la prévention de complications visuelles à long terme.

Ce mémoire est structuré en cinq chapitres. Le chapitre 2 débute par un rappel des fondements de l'intelligence artificielle et de la vision par ordinateur, puis présente l'anatomie de l'œil ainsi que les principaux types de strabisme et leurs conséquences cliniques. Il passe ensuite en revue les méthodes de détection du strabisme : d'abord les approches cliniques traditionnelles, telles que le test de Hirschberg, puis les techniques fondées sur le traitement d'image et l'apprentissage automatique. Enfin, il détaille les algorithmes rapides de détection d'objets (RCNN, Fast R-CNN, Faster R-CNN, SSD(*Single Shot Detector*), YOLO) et expose les raisons qui motivent la sélection du modèle YOLOv4 pour ce travail. Le chapitre 3 détaille l'architecture d'YOLOv4 et la méthodologie employée, incluant la constitution de la base de données, le traitement des données (labelling), la configuration des fichiers et le processus d'entraînement. Le chapitre 4 expose les performances du modèle à travers les résultats des tests sur divers échantillons d'images. Enfin, le chapitre 5 discute des applications cliniques potentielles de la détection du strabisme avec YOLOv4, comparant ses avantages et limites face aux méthodes classiques, avant de proposer des perspectives d'amélioration.

# CHAPITRE 2. ÉTAT DE L'ART

Ce chapitre présente une exposition des différentes méthodes utilisées pour la détection d'objets sur une image dans le cadre de la vision par ordinateur en général, et pour la détection du strabisme en particulier. Dans un premier temps, les domaines de l'IA ainsi que l'anatomie de l'œil sont introduits afin de mieux situer le contexte. Le strabisme, ses conséquences et ses différents types sont ensuite abordés. Par la suite, les diverses méthodes de détection, qu'elles soient cliniques ou traditionnelles, ainsi que celles reposant sur les techniques de l'apprentissage automatique, également appelé *Machine Learning (ML)*, de l'apprentissage profond, et de la vision par ordinateur, sont présentées. Les algorithmes de détection utilisés en vision par ordinateur pour effectuer des détections rapides sont également détaillés, en précisant celui qui sera adopté pour notre système. Enfin, une présentation approfondie du fonctionnement d'YOLOv4, le modèle choisi pour ce projet, est proposée.

## 2.1 L'IA et ses sous-domaines

Nous pouvons définir l'IA comme le concept visant à attribuer aux machines la faculté de raisonner et de prendre des décisions comme le ferait un humain face à certaines tâches, en tenant compte de leur environnement. Au cours des dernières années, l'IA a connu une adoption massive, Parmi les usages déjà opérationnels, la santé constitue l'exemple le plus documenté : l'algorithme **IDx-DR**, autorisé par la FDA en 2018, est utilisé en clinique pour dépister automatiquement la rétinopathie diabétique à partir de photos du fond d'œil, avec une sensibilité de 87 % et une spécificité de 90 % rapportées lors de son essai multicentrique [57]. Dans la vie quotidienne, elle se manifeste également à travers des fonctionnalités telles que le déverrouillage des appareils, la recommandation de contenu, ou encore la gestion de l'énergie dans des bâtiments connectés. Cela passe par un système de reconnaissance des caractéristiques comportementales et physiologiques propres à chaque individu, telles que les caractéristiques de l'œil. La Figure 1 situe les principaux sous-domaines : l'IA englobe, d'une part, des composantes « symboliques » comme le *raisonnement* (systèmes experts) et la *planification* (recherche de trajectoires optimales) et, d'autre part, le **machine learning (ML)**, qui analyse des données structurées pour produire des prédictions ; dans cette catégorie se trouve la **régression logistique**, méthode supervisée emblématique. À l'intérieur même du ML, l'**apprentissage profond (deep learning)** se distingue par l'usage de réseaux neuronaux capables de traiter des données non structurées — images, vidéos, sons — et fournit aujourd'hui les meilleures performances en reconnaissance visuelle. C'est précisément ce

champ du deep learning que nous explorerons, car il regroupe les approches les plus efficaces pour la détection d'images oculaires et, partant, pour le dépistage automatique du strabisme que nous visons dans ce travail.

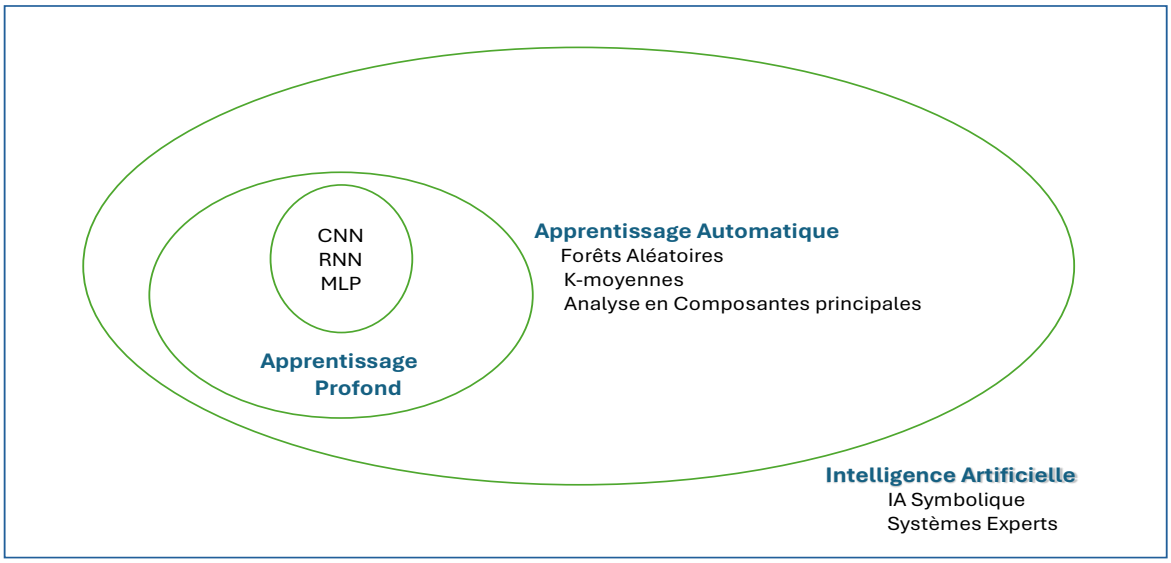


*Figure 1 l'IA et ses sous-domaines*

## 2.2 Anatomie de l'œil.

L'anatomie de l'œil humain fournit une base essentielle pour comprendre et interpréter les images utilisées par les modèles de vision artificielle. Dans le cadre de ce mémoire, le modèle YOLOV4 est appliqué pour détecter les anomalies du regard, notamment celles causées par le strabisme. Ces anomalies impliquent souvent un désalignement des axes visuels, directement lié à la position des structures comme la pupille, la cornée et l'iris. L'une des principales contributions de ce travail est de fournir une méthode robuste pour localiser et analyser les points clés anatomiques de l'œil à partir d'images frontales. Ces points servent à déterminer les angles de déviation oculaire, identifier les caractéristiques spécifiques du strabisme (exotropie, ésotropie) et faciliter une correction clinique ou une prise en charge précoce. Dans le cadre du modèle YOLOV4, une attention particulière est accordée à la localisation précise des points d'intérêt, notamment la pupille et le centre de l'iris, pour analyser les alignements oculaires. La segmentation et l'identification des caractéristiques anatomiques servent de prétraitement essentiel, permettant d'améliorer la performance du modèle sur des images complexes. La

Figure 2 illustre les principales structures anatomiques de l'œil humain, chacune jouant un rôle spécifique dans le processus de la vision.

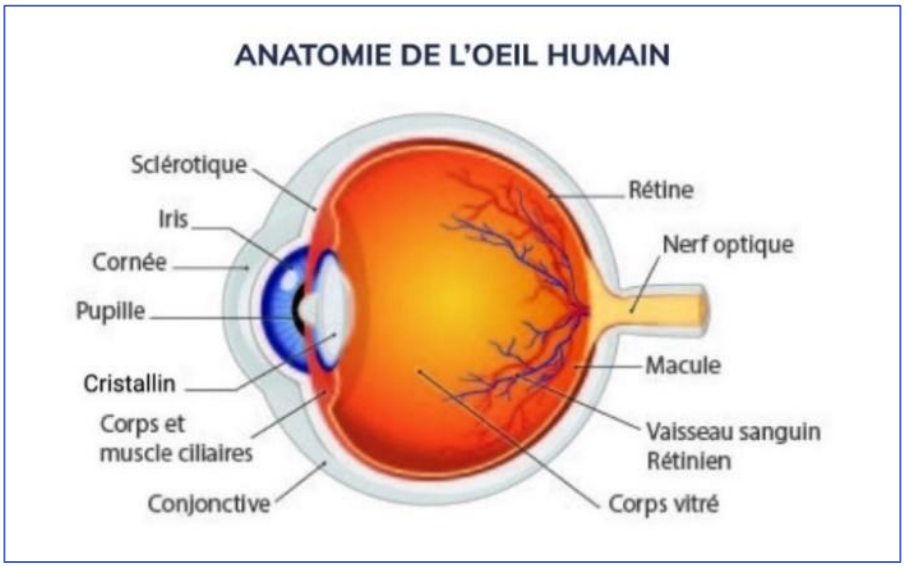


*Figure 2 Anatomie de l'œil humain [2]*

La **sclérotique** est une couche externe, blanche, résistante et fibreuse, qui protège l'œil contre les agressions mécaniques et maintient sa forme. Cette structure joue un rôle indirect dans l'apparence des images biomédicales, car lors de l'acquisition d'images, elle peut refléter une lumière parasite, introduisant des artefacts que les algorithmes doivent ignorer. À l'avant de l'œil, la **cornée**, partie transparente, participe à la réfraction de la lumière et constitue environ deux tiers du pouvoir optique total de l'œil. Sa courbure influence la géométrie des images acquises et peut impacter la détection des anomalies si elle n'est pas prise en compte dans le prétraitement. L'**iris**, partie colorée de l'œil, contrôle la quantité de lumière entrant dans l'œil en ajustant la taille de la **pupille**, qui est l'ouverture centrale jouant le rôle de diaphragme. La position et la dilatation de la pupille sont souvent des indicateurs clés dans l'analyse des mouvements oculaires et des pathologies associées, telles que le strabisme. Une détection précise de la pupille est essentielle pour les modèles de vision artificielle. Le **cristallin**, une lentille biconvexe située derrière l'iris, ajuste sa courbure (accommodation) pour focaliser les objets à différentes distances. Bien que le cristallin ne soit pas directement visible sur les images biomédicales de surface, ses anomalies, comme la cataracte, peuvent influencer les propriétés optiques de l'œil et nécessitent une segmentation adaptée dans le cadre d'une analyse. Les **corps ciliaires** et **muscles ciliaires**, qui contrôlent la courbure du cristallin en modifiant sa

tension, ne sont pas directement visibles dans les images frontales utilisées pour détecter le strabisme, mais ils influencent indirectement la dynamique oculaire. Le **corps vitré**, une substance gélatineuse transparente qui remplit l'œil et maintient sa forme interne, peut introduire des distorsions mineures lors de l'acquisition des images biomédicales. La **rétine**, couche interne sensible à la lumière, est composée de photorécepteurs (bâtonnets et cônes) qui transforment les signaux lumineux en impulsions nerveuses. Bien que la rétine ne soit pas analysée directement pour la détection du strabisme, sa configuration peut être utilisée pour évaluer la fixation oculaire. La **macule**, région centrale de la rétine responsable de la vision fine et détaillée, est cruciale dans l'analyse biomédicale pour identifier les axes de fixation visuelle, souvent impactés par des anomalies comme le strabisme. Le **nerf optique**, structure fibreuse qui transporte les signaux électriques de la rétine vers le cerveau, n'est généralement pas capturé directement dans les images biomédicales, mais des pathologies associées à ce dernier peuvent altérer la géométrie des yeux. Enfin, les **vaisseaux sanguins rétiniens**, qui nourrissent la rétine et sont visibles dans les images de fond d'œil, sont parfois utilisés comme repères dans l'analyse des images médicales pour ajuster ou valider les alignements détectés. Cette présentation globale des structures de l'œil met en lumière leur rôle direct ou indirect dans l'analyse biomédicale, notamment pour la détection et la gestion des pathologies comme le strabisme.

### 2.3 Le strabisme : définition et conséquences

Le strabisme est un défaut de convergence ou de parallélisme des axes visuels, qui fait que les yeux ne regardent pas exactement dans la même direction en même temps. Il est plus fréquent chez les enfants, mais peut aussi apparaitre de manière tardive ce dans ce sens qu'E. Bui Quoc et M.A. Espinasse-Berrod affirment dans leur article que :
« *Le strabisme est évidemment plus fréquent dans l'enfance, mais ce qu'il faut souligner, ce sont les enjeux qui sont en cause. En premier lieu, il faut se rappeler qu'un strabisme peut révéler une pathologie potentiellement grave.* » [1]. Lorsque le strabisme est négligé, il peut créer d'énormes problèmes tels que le rétinoblastome et une éventuelle amblyopie d'où la nécessité de le détecter de manière précoce chez les enfants afin d'intervenir. Selon E. Bui Quoc et M.-A. Espinasse-Berrod, « *Un strabisme peut révéler une anomalie organique oculaire. Citons l'exceptionnel, mais redoutable rétinoblastome. Tout strabisme peut perturber la vision et entraîner une amblyopie, qui elle-même peut induire un strabisme* » [1]

## 2.4 Les cas de strabisme

Le strabisme est en grande partie détecté en fonction de l'aspect de l'œil et des paupières, il peut concerner seulement un œil (le cas monoculaire) ou les deux yeux (le cas binoculaire). « *Le strabisme est défini comme un désordre oculomoteur présentant une double composante. D'une part, une composante motrice, correspondant à une déviation d'un œil, et d'autre part, une composante sensorielle, liée à un dérèglement de la vision binoculaire* ». [2]

Le strabisme peut être divergent ou convergent. Concernant la déviation d'un seul œil, lorsque l'œil converge vers l'intérieur on parle d'estropie, lorsque l'œil part vers l'extérieur on parle de l'estropié, lorsque l'œil monte on parle d'hypertrophie et lorsque l'œil descend on parle d'hypotrophie. Concernant le cas binoculaire, on parlera du strabisme binoculaire divergent quand les deux yeux convergent vers l'intérieur et du strabisme binoculaire converge quand les deux yeux s'écartent vers l'extérieur. La Figure 3 montre différentes conditions liées au strabisme, un trouble oculaire où les yeux ne s'alignent pas correctement.

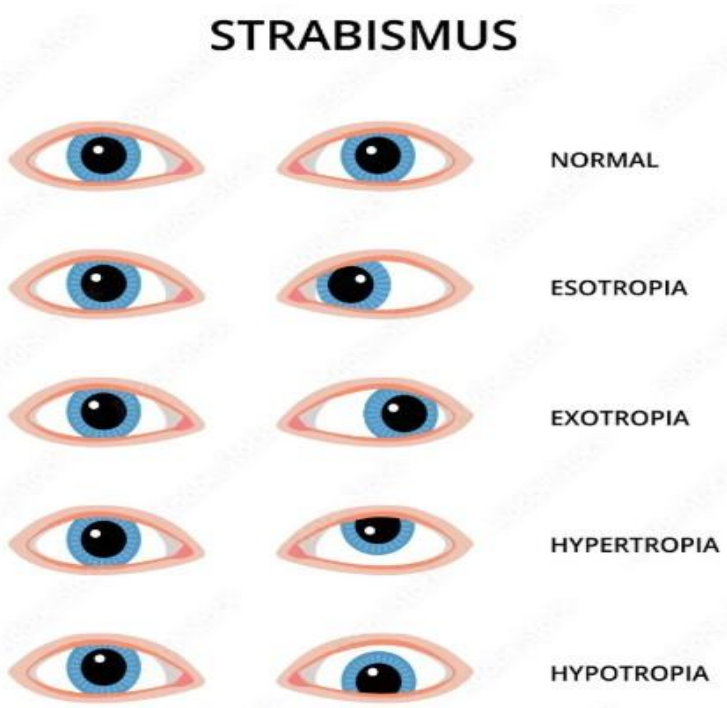


*Figure 3 différents cas de strabismes monoculaires [3]*

Le cas dit "Normal" est celui où les deux yeux sont parfaitement alignés et orientés dans la même direction. Cela correspond à une vision binoculaire normale, où les images perçues par chaque œil sont fusionnées par le cerveau. Ce cas sert de référence pour le calibrage des modèles de détection des anomalies. Dans le cas de l'**esotropie** (*esotropia*), l'œil affecté dévie

vers l'intérieur, en direction du nez. Ce type de strabisme est souvent observé chez les jeunes enfants et peut entraîner une perte de la vision binoculaire si non traitée. La détection de l'**ésotropie** repose sur l'analyse de la position relative de la pupille par rapport aux bords de l'iris. YOLOV4 doit être entraîné à distinguer cette déviation des variations normales. Pour l'**exotropie** (*exotropia*), l'œil dévie vers l'extérieur, en direction de la tempe. Ce type de strabisme peut être constant ou intermittent. L'identification de l'**exotropie** nécessite un modèle capable de détecter une asymétrie marquée dans l'alignement horizontal des yeux. L'**hypertropie** (*hypertropia*) est une condition où l'œil affecté dévie vers le haut. Cette forme est moins fréquente que l'**isotropie** ou l'**exotropie** et peut être associée à des lésions des muscles oculaires ou des nerfs. YOLOV4 doit intégrer des données d'entraînement pour analyser les variations verticales dans la position des yeux. Enfin, l'**hypotropie** est caractérisée par une déviation de l'œil vers le bas, souvent causée par une faiblesse des muscles oculaires supérieurs. La détection de ce type de strabisme est cruciale pour une analyse complète des déviations verticales.

## 2.5 Les méthodes de détection du strabisme

### 2.5.1 Méthodes cliniques et traditionnelles

Les tests cliniques habituels couramment utilisés pour la détection du strabisme incluent le test de Hirschberg, le test de l'acuité visuelle et les méthodes par vision screening.

Dans le test de Hirschberg [1, 5], une source lumineuse est placée à une distance de 50 cm, centrée sur la racine du nez. La lumière est réfléchie par la cornée. Dans les deux yeux, la distance entre ces rayons réfléchis et le centre de la pupille donne le degré de déplacement du strabisme. Si la personne ne souffre pas de strabisme, la lumière réfléchie par la cornée est visible au centre de la pupille. En d'autres termes, la lumière réfléchie par la cornée représente en réalité le centre de l'œil. « *En éclairant les pupilles avec une lumière douce à un mètre, la lueur pupillaire doit être symétrique. Le reflet pupillaire de l'œil fixateur apparaît grisâtre et l'œil non fixateur est rougeâtre ou rosé* » [2] dans le test de l'acuité visuelle, les spécialistes utilisent plusieurs techniques pour tester l'acuité visuelle. Parmi celles-ci, on trouve la méthode par occlusion, qui consiste à utiliser des cache-œils adhésifs ou un ruban chirurgical hypoallergénique de 2 pouces de large. Une autre approche repose sur l'utilisation de symboles de test adaptés. *« Les symboles HOTV et LEA, deux tests développés pour être utilisés chez les enfants d'âge préscolaire, sont actuellement considérés comme les meilleures pratiques pour tester l'acuité visuelle des enfants âgés de 36 à 72 mois* » [3]. Enfin, la méthode de vision

screening s’appuie sur un dépistage instrumental par autoréfraction ou photo réfraction. Cette dernière identifie la présence et l’ampleur de l’erreur de réfraction, mais ne fournit pas une mesure directe de l’acuité visuelle.

### 2.5.2 Quelques méthodes de détection du strabisme par L’IA et vision par ordinateur

#### 2.5.2.1 La méthode par combinaison entre la dimérisation d’Otsu et le modèle HSV (*Hue, Saturation, Value*)

La détection du strabisme à l'aide de l'IA et de la vision par ordinateur a fait l'objet de diverses approches. L'une d'elles combine la binarisation d'Otsu et le modèle de couleur HSV pour segmenter efficacement les images des yeux [4, 6]. Cette méthode est particulièrement utile pour le dépistage automatique du strabisme, notamment dans les zones reculées où l'accès aux soins médicaux est limité. La Figure 4 illustre le processus de détection du strabisme basé sur l'IA et la vision par ordinateur. Ce processus débute par l'acquisition d'une image faciale frontale, qui est ensuite traitée par un modèle de détection de visage afin d'identifier la région du visage. Une fois cette région détectée, un détecteur de points de repère faciaux est utilisé pour extraire la zone des yeux. La binarisation d'Otsu et le modèle de couleur HSV sont appliqués à cette image des yeux pour segmenter les structures oculaires pertinentes. Les résultats de ces deux méthodes sont combinés pour former une nouvelle image. Des points de pixels situés au niveau du limbe sont ensuite échantillonnés et utilisés pour estimer le centre de la pupille à l'aide de la méthode des moindres carrés (*Least Squares Method*, LSM). Enfin, la similarité de position de l'iris entre les deux yeux est calculée pour détecter un éventuel strabisme.

Cette approche permet une détection précise du strabisme en exploitant les techniques de traitement d'images et les modèles de couleur, offrant ainsi une solution efficace pour le dépistage dans des environnements à ressources limitées.

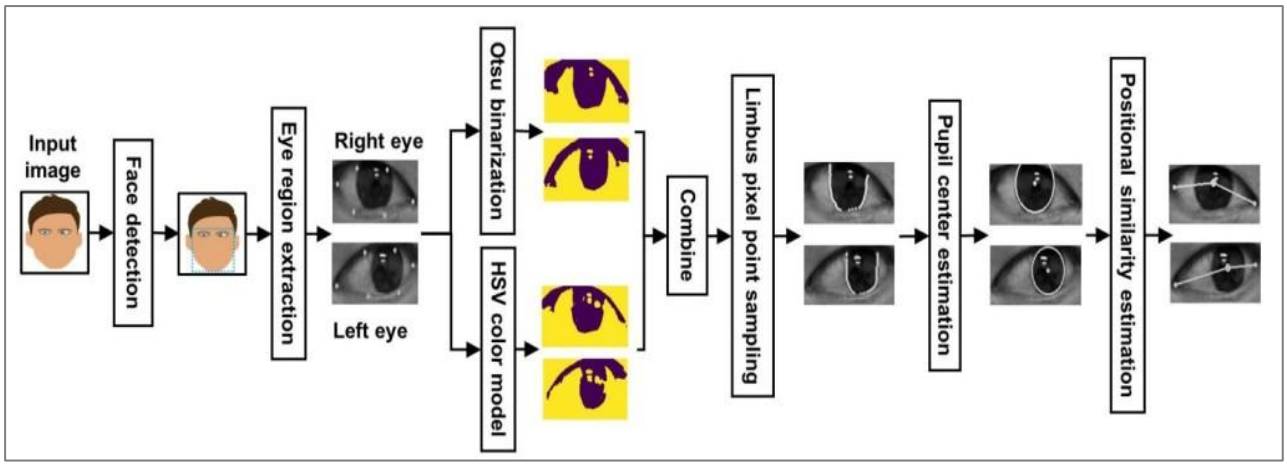


*Figure 4. Méthode de dépistage proposée par Huang et al [4]*

Par ailleurs, toujours dans la méthode de traitement d'images, La Figure 5 propose une approche basée sur l'apprentissage profond et le traitement d'images pour la détection du strabisme à l'aide du test de Hirschberg. Cette méthode vise à permettre un diagnostic plus rapide et économique grâce à une numérisation du test de Hirschberg. Elle repose sur un modèle mathématique qui établit une relation entre les degrés et les pixels, en utilisant les unités de mesure du test. La méthode exploite une caméra et un flash de téléphone portable comme outils pour capturer les images nécessaires. Le logiciel détecte la lumière réfléchie par le centre de la pupille et la cornée des deux yeux, mesure l'écart entre ces réflexions sur une image frontale du visage et fournit des informations précises à l'ophtalmologiste. Dans cette méthode, les étapes clés incluent plusieurs composantes distinctes. Tout d'abord, la détection commence par le **Face mesh**, qui utilise des techniques d'apprentissage profond pour repérer et segmenter les caractéristiques faciales importantes, telles que la position des yeux. Cela permet d'extraire les **ROI (***Region of Interest***)** correspondant aux iris des deux yeux. Ces régions sont ensuite traitées pour isoler les informations nécessaires. L'étape de **Thresh binary** fait référence à un processus de seuillage binaire, où une image est transformée en une carte binaire en distinguant les zones de forte et faible intensité lumineuse. Cela permet de simplifier l'analyse des réflexions lumineuses en isolant les régions de la pupille et de la cornée ayant des caractéristiques spécifiques. Une fois l'image binaire obtenue, les **Draw contours** servent à identifier et tracer les contours des zones détectées dans l'image binaire. Cela aide à localiser précisément les réflexions lumineuses sur la pupille et la cornée, essentielles pour la suite du traitement. La composante **Light reflected** analyse spécifiquement les points de réflexion lumineux générés par la lumière du flash sur la surface de l'œil. Cette étape est cruciale pour mesurer l'écart entre

les réflexions des deux yeux, une mesure clé dans le diagnostic du strabisme. Dans la partie inférieure de la Figure 5, le passage à l'espace **HSV de l'iris** permet de représenter l'image dans un espace colorimétrique basé sur la teinte, la saturation et la valeur, facilitant la segmentation de la pupille et de l'iris. La **masking process** applique un masque pour isoler ces zones et les rendre prêtes pour l'analyse. Enfin, la détection de la pupille (**Pupil detection**) affine la localisation des centres de réflexion sur les deux yeux. Les données extraites sont ensuite intégrées dans l'**algorithme de détection du strabisme**, qui calcule l'écart angulaire basé sur un modèle mathématique liant les pixels aux degrés mesurés par le test de Hirschberg. La méthode a été testée sur des images collectées auprès de 88 patients atteints de strabisme dans le service d'ophtalmologie de l'hôpital universitaire de Firat et validée sous la supervision d'un ophtalmologiste. Cette méthodologie a démontré son efficacité pour un diagnostic plus rapide et accessible [5].

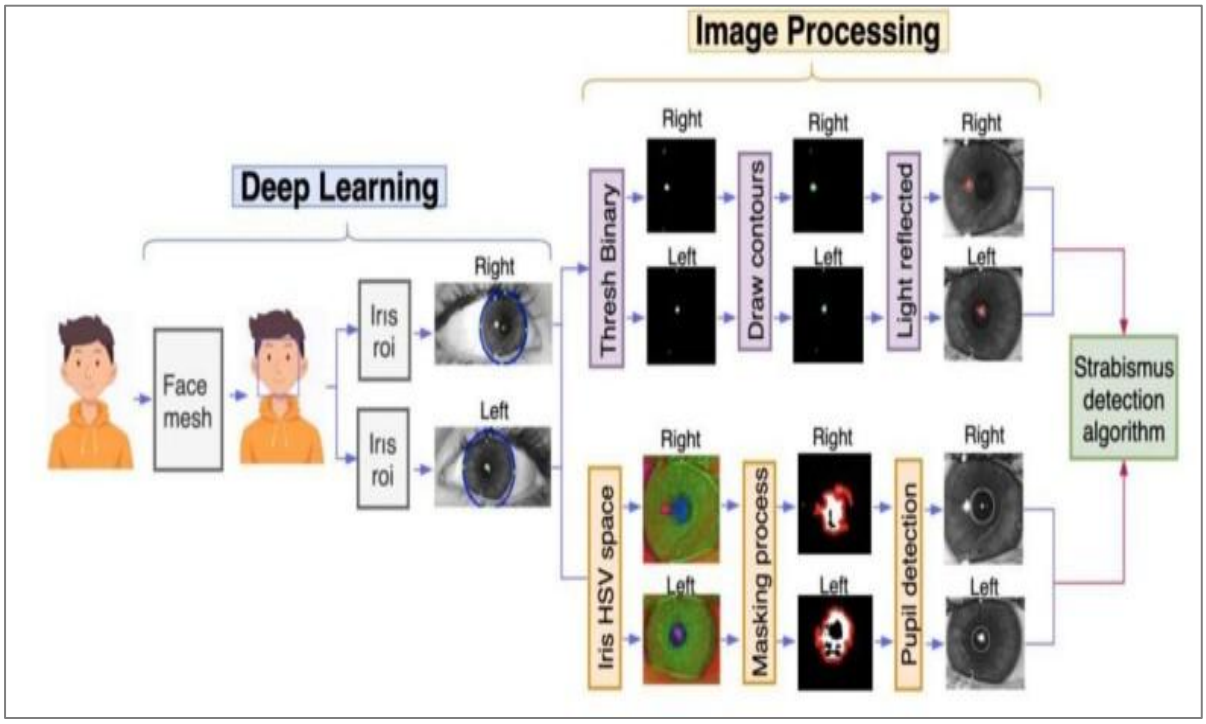


*Figure 5. Méthode basée sur l'apprentissage profond et le traitement d'images pour la détection du strabisme avec le test de Hirschberg. [5]*

La Figure 6 (a) montre la configuration expérimentale utilisée pour collecter les données, où une caméra de téléphone portable et un flash intégré servent respectivement de dispositif de capture et de source lumineuse. La distance de 50 cm entre le patient et la caméra est soigneusement respectée pour garantir une mesure cohérente. La Figure 6 (b) illustre en détail

l'image capturée sur l'écran du téléphone, où la position frontale du visage est clairement visible, permettant une analyse précise.

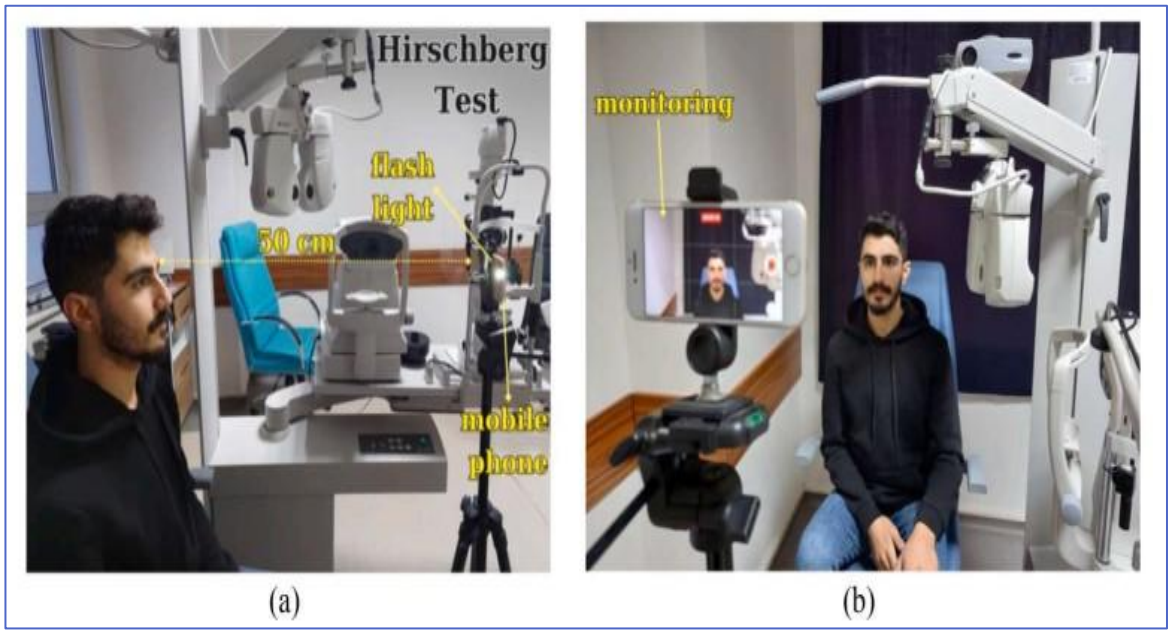


*Figure 6. Application du test Hirschberg et méthode de collecte de données [5]*

### 2.5.2.2 Méthode par combinaison de méta-apprentissage et de traitement d'images

Cette méthode avancée de dépistage du strabisme repose sur une combinaison de métaapprentissage et de traitement d'images, comme illustrée dans la Figure 7. Conçue pour pallier les limites imposées par une rareté des données, elle améliore la précision de la classification grâce à une extraction optimisée des caractéristiques visuelles.

Le processus débute par l'envoi des exemples d'entraînement (illustrés à gauche dans la Figure 7) à un réseau d'intégration préentraîné $f\emptyset$, suivi d'une Analyse en Composantes principales (*Principal Component Analysis* , PCA). Cette étape permet de transformer les caractéristiques extraites en vecteurs intégrés de faible dimension tout en conservant les informations essentielles. Simultanément, deux modules spécifiques sont appliqués aux échantillons : un module d'estimation de similarité positionnelle, chargé d'analyser les correspondances spatiales entre les points clés des yeux, tels que la position des pupilles et des réflexions lumineuses cornéennes ; et un module d'estimation des ratios de réflexe lumineux cornéen (*Corneal Light Reflex Ratio*, CLR), qui examine les alignements oculaires en fonction des réflexions détectées sur la cornée.

Les informations issues de ces modules sont ensuite combinées avec les vecteurs intégrés, produisant des vecteurs de caractéristiques enrichis, comme illustrés dans la section centrale de la Figure 7 (*Combined feature vectors*). Ces vecteurs enrichis alimentent une machine à vecteurs de support (*Support Vector Machine*, SVM), qui détermine les poids d'un hyperplan optimisé pour séparer les exemples normaux des anomalies. Cette phase garantit une généralisation robuste, même dans un contexte de données limitées.

Lors de la phase de test, les nouveaux exemples suivent un processus identique d'extraction et de combinaison des caractéristiques. Ces vecteurs sont ensuite projetés dans l'espace des vecteurs d'entraînement, permettant une classification fiable et précise des cas de strabisme.

En intégrant efficacement des informations critiques sur la région oculaire, cette approche propose une solution robuste pour le dépistage du strabisme. La Figure 7 résume l'ensemble de ce processus, mettant en évidence les étapes clés allant de l'extraction initiale des caractéristiques à la classification finale.

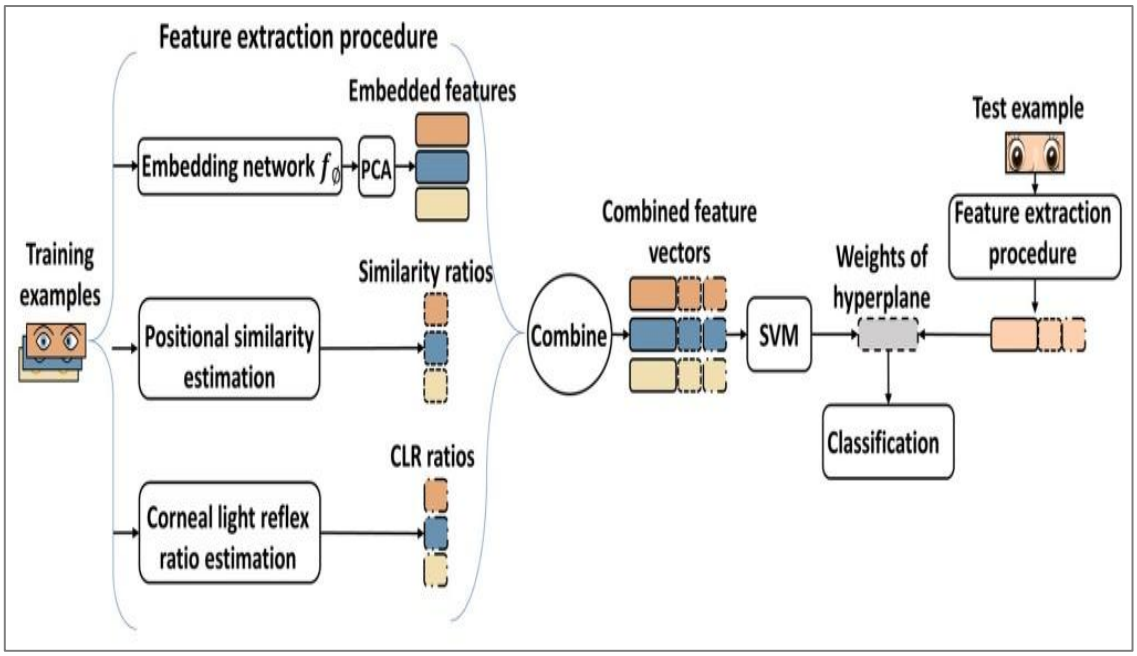


*Figure 7. Méthode par combinaison de méta-apprentissage et de traitement d'images proposée par [6]*

### 2.5.2.3 Méthode de détection par le réseau à convolution RF-CNN

Le RF-CNN (*Random Forest Convolutional Neural Network*) est un modèle d'apprentissage automatique qui combine des aspects des forêts aléatoires et des réseaux de neurones convolutionnels. Il utilise à la fois les techniques de classification des forêts aléatoires et les capacités d'apprentissage profond des réseaux de neurones convolutionnels pour effectuer des tâches de classification et de reconnaissance d'images. Certains chercheurs se sont servis de

cette méthode pour la détection automatisée du strabisme dans le cadre des applications de télémédecine [7]. Dans leurs systèmes, le RF-CNN effectue d'abord une segmentation de la région oculaire sur chaque image individuelle, puis classe ensuite les régions oculaires segmentées avec des réseaux de neurones profonds. Le schéma de leur système, ainsi que l'architecture du RF-CNN utilisée, est présenté dans la Figure 8. Elle détaille le processus de diagnostic automatisé du strabisme, structuré en plusieurs étapes. Tout d'abord, les données d'entrée, constituées d'images contenant des exemples de strabisme, alimentent le système d'analyse. Ces images sont traitées par un modèle d'apprentissage profond, le R-FCN (*RegionBased Fully Convolutional Network*), dont le rôle est de détecter et de localiser précisément les régions des yeux dans chaque image. Cette étape permet d'isoler les parties pertinentes, à savoir les zones oculaires, en fournissant les coordonnées nécessaires pour la suite de l'analyse. Les emplacements identifiés sont ensuite transmis à un réseau de neurones convolutif (CNN), qui réalise une segmentation fine des régions oculaires pour extraire les caractéristiques spécifiques associées au strabisme. En parallèle, la CNN détecte les événements pertinents en lien avec des anomalies visuelles, qui sont finalement utilisés pour établir un diagnostic précis du strabisme. Ce processus modulaire combine la détection, la segmentation et l'analyse des données visuelles pour garantir une évaluation automatisée et fiable.

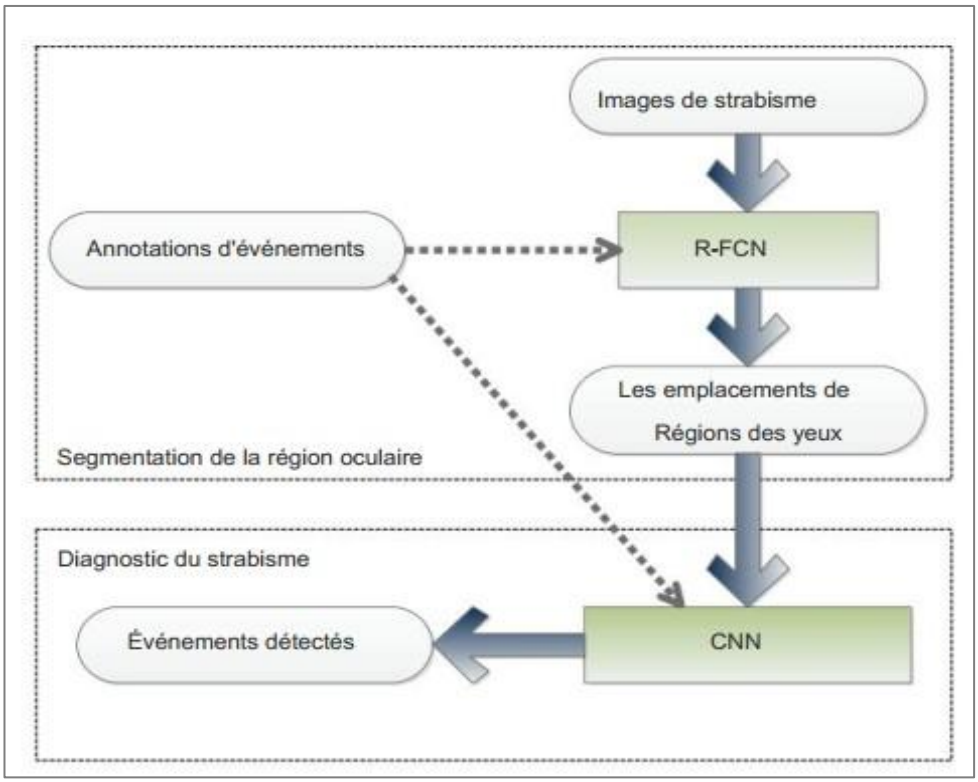

*Figure 8. Synoptique du système à base du RF-CNN. [7]*

#### 2.5.2.4 Reconnaissance du strabisme à l'aide de données de suivi oculaire et réseaux de neurones convolutifs

Parmi les méthodes de détection du strabisme, la méthode de suivi oculaire, comme celle proposée par [8], offre une solution innovante basée sur l'analyse des mouvements des yeux et l'utilisation de réseaux de neurones convolutifs (CNN). Ce système repose sur l'intégration de données de suivi oculaire obtenues grâce à un *eye tracker (*un dispositif technologique utilisé pour enregistrer et analyser les mouvements oculaires en suivant la position et le déplacement des yeux dans le temps*)*, combinées à des techniques avancées de traitement des images, pour reconnaître et classifier les cas de strabisme.

Le processus commence par l'enregistrement des mouvements oculaires d'un sujet à l'aide d'un dispositif de suivi oculaire. Ces données brutes, qui comprennent les trajectoires des regards et les points de fixation, servent à générer une image appelée GaDe (Gaze Deviation). Cette image de déviation du regard est une représentation visuelle synthétique des écarts entre les points de regard enregistrés et les cibles visuelles attendues. Elle permet de capturer des caractéristiques clés des mouvements oculaires, telles que la précision et la cohérence des points de fixation. L'image GaDe est ensuite transmise à un réseau neuronal convolutif (CNN) préalablement entraîné sur la base de données ImageNet , bien connue pour sa diversité et son volume considérable d'images.

La CNN extrait des caractéristiques profondes de l'image GaDe en exploitant les couches convolutives pour identifier des motifs spécifiques aux sujets strabiques. Les sorties des couches pleinement connectées de la CNN sont utilisées comme vecteurs de caractéristiques représentant l'image GaDe. Ces vecteurs, riches en informations discriminantes, sont ensuite transmis à un classificateur, tels qu'un SVM, chargé de séparer les cas strabiques des cas normaux.

Pour valider l'efficacité de cette méthode, un ensemble de données expérimental est constitué, comprenant des données de suivi oculaire issues de sujets strabiques et normaux. Cette base de données permet d'entraîner et de tester le système, garantissant ainsi une robustesse et une précision élevées. En combinant les avantages des données de suivi oculaire, des images GaDe et des réseaux neuronaux convolutifs, ce système propose une solution puissante et novatrice

pour le dépistage du strabisme, particulièrement utile dans des contextes cliniques et de télémédecine.

La Figure 9 synthétise les différentes étapes de ce processus, de l'enregistrement des mouvements oculaires au résultat final indiquant si le sujet est strabique ou non. Cette méthode met en évidence le potentiel des approches basées sur le suivi oculaire et l'apprentissage profond pour améliorer la reconnaissance automatisée des pathologies oculaires.

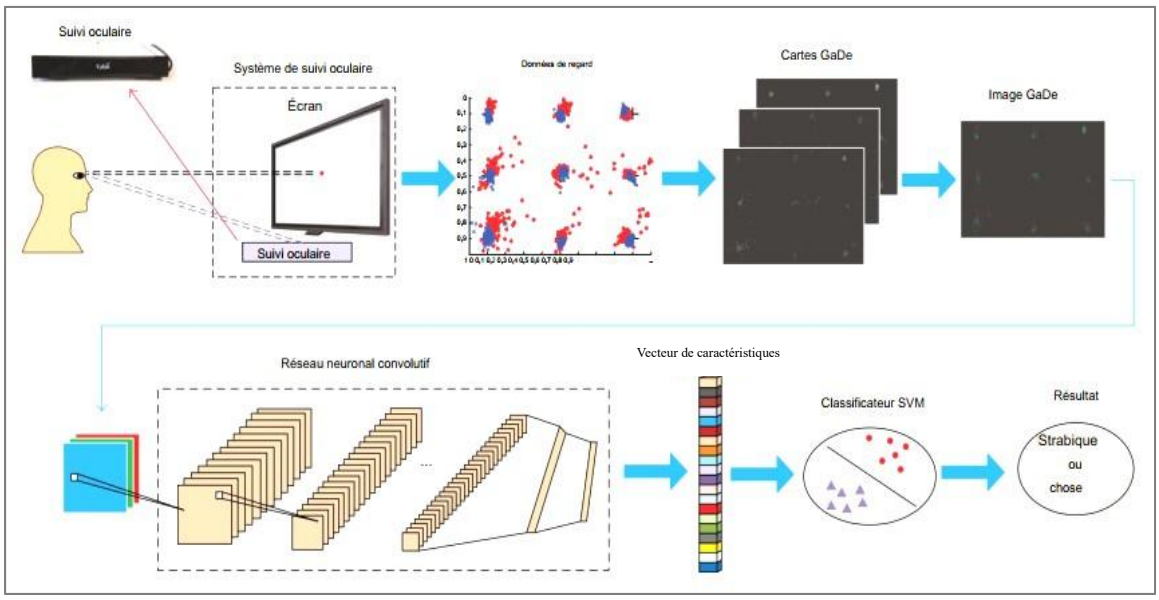


*Figure 9. Reconnaissance du strabisme à l'aide de données de suivi oculaire et Réseaux de neurones convolutifs. [8]*

Ce système utilise un algorithme bien précis pour l'acquisition des données de regard, comme illustré dans les Figures 11 et 12. Le processus débute par une interaction structurée entre le sujet et l'interface. Le sujet s'assoit devant un écran et est invité à faire suivre son regard sur neuf points spécifiques, successivement affichés à l'écran. Ces points sont stratégiquement positionnés selon des coordonnées prédéfinies, comme montré dans la Figure 11, pour couvrir différentes zones du champ visuel. Cette interface garantit une collecte systématique et standardisée des données de regard, essentielles pour l'analyse subséquente.

La procédure, détaillée dans le diagramme de la Figure 12, commence par l'ajustement du sujet et un processus d'étalonnage visant à vérifier la qualité et la fiabilité des données initiales. Si l'étalonnage est jugé acceptable, le système affiche les points de fixation à suivre un par un. Pendant cette phase, des données de regard précises sont enregistrées pour chaque point. Si des anomalies sont détectées ou si le sujet ne parvient pas à suivre correctement un point dans un délai imparti de 10 secondes, l'étalonnage peut être réitéré ou ajusté.

Les données collectées sont systématiquement enregistrées après chaque point, jusqu'à atteindre un total de 100 acquisitions ou jusqu'à ce que les neuf points soient complétés. Cette stratégie garantit une densité suffisante de données pour analyser la trajectoire du regard et détecter des schémas pouvant indiquer des anomalies, telles que celles associées au strabisme. À la fin du processus, les données de regard sont prêtes à être transmises aux étapes de traitement ultérieures, incluant la génération des images GaDe et leur analyse par des réseaux neuronaux convolutifs.

En combinant une interface visuelle intuitive et une procédure méthodique d'acquisition des données, ce système assure la collecte de données de regard de haute qualité, posant ainsi les bases pour des analyses fiables et précises dans le cadre du dépistage automatisé du strabisme.

La Figure 10 illustre une interface utilisée pour l'acquisition de données de regard en neuf points, répartis uniformément sur un écran. Chaque point est positionné selon des coordonnées normalisées (x, y) variant entre 0,1 et 0,9. Cette configuration couvre l'ensemble de l'écran, avec des points situés aux extrémités, au centre et dans des positions intermédiaires. L'objectif est de permettre une analyse complète des mouvements de regard dans toutes les directions.

Les flèches visibles dans l'interface indiquent un ordre précis de navigation entre les points. Ce parcours débute en haut à gauche, au point (0,1 ; 0,1), et suit un chemin structuré qui traverse horizontalement les points situés sur la même ligne, puis descend verticalement avant de revenir vers la gauche. Ce cheminement permet de capturer des données visuelles pour chaque position, garantissant ainsi une acquisition complète et cohérente.

Le choix du fond noir et des points rouges maximise le contraste visuel, rendant les cibles plus facilement détectables par les capteurs chargés de suivre les mouvements oculaires, selon le protocole d'acquisition décrit par Chen *et al.* [8]**.** Cette configuration permet de constituer un jeu de données précis destiné à l'apprentissage du détecteur d'objets YOLOv4, introduit par Bochkovskiy *et al.* [15], reconnu pour son excellent compromis vitesse/précision. Les données ainsi collectées jouent ensuite un rôle central dans l'identification et l'analyse des saccades et fixations, facilitant la détection d'anomalies telles que le strabisme.

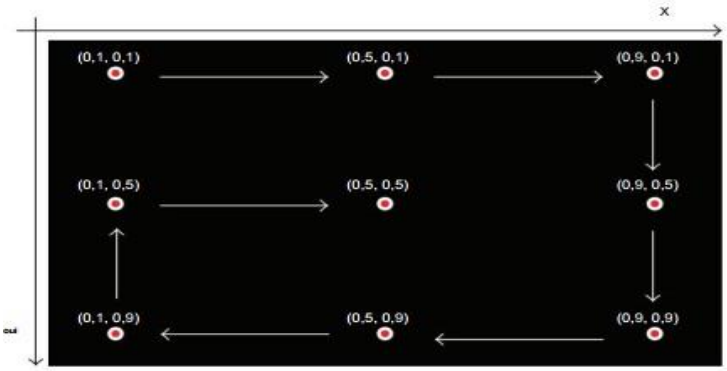


*Figure 10. L'interface d'acquisition de données de regard en neuf points. [8]*

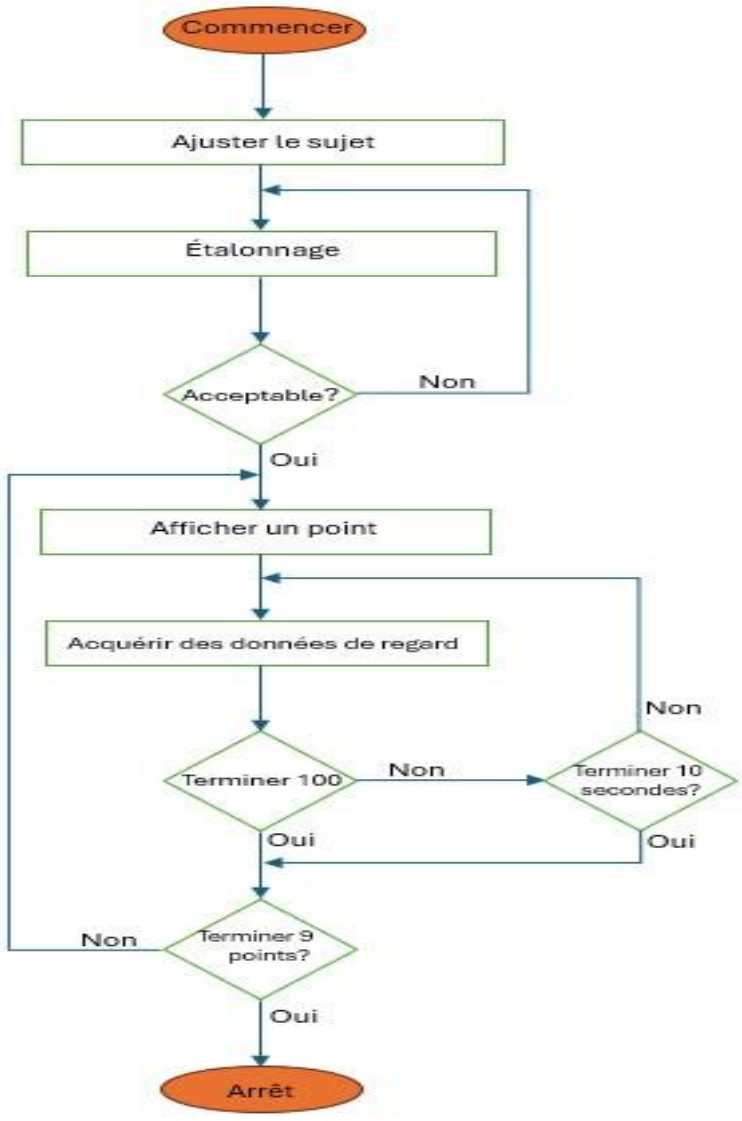


*Figure 11. La procédure d'acquisition de données de regard. [8]*

## 2.6 Quelques algorithmes de détections utilisées dans la vision par ordinateur

La vision par ordinateur est l'une des tâches principales du domaine de l'apprentissage profond. En ce qui concerne la détection d'objets, il s'agit d'une technique permettant d'identifier l'existence d'objets dans une image ou une vidéo. La détection d'objets peut être effectuée via diverses techniques, parmi lesquelles les plus populaires sont : le R-CNN (*Region-based Convolutional Neural Network*), Fast R-CNN, Faster R-CNN, SSD et les différentes versions de YOLO .

### 2.6.1 Le modèle R-CNN (Region-based Convolutional Neural Network)

Comme présenté par [9], le modèle R-CNN a été l'un des premiers à utiliser des réseaux de neurones convolutifs pour la détection d'objets. Son objectif principal est de capturer une image et d'identifier correctement l'emplacement des objets majeurs en traçant des boîtes englobantes précises autour de ces derniers. La Figure 12 illustre l'architecture du modèle RCNN. Cette approche repose sur une technique appelée "recherche sélective" (Selective Search) pour générer des propositions de régions. Cette méthode consiste à examiner l'image en appliquant des fenêtres de tailles variées. Ces fenêtres sont ensuite regroupées selon des critères tels que la texture, la couleur ou l'intensité, afin de déterminer les régions susceptibles de contenir des objets.

Une fois les régions proposées, R-CNN les transforme en des images carrées de taille standard pour permettre leur traitement par un réseau de neurones convolutifs (CNN). Ce CNN extrait les caractéristiques des régions et les transmet à un classificateur, généralement un SVM. Le rôle du SVM est de vérifier si la région correspond à un objet et, le cas échéant, d'identifier cet objet. Bien que précis, ce processus présente un inconvénient majeur : chaque proposition de région est traitée individuellement par le CNN, ce qui entraîne des temps de calcul très élevés.

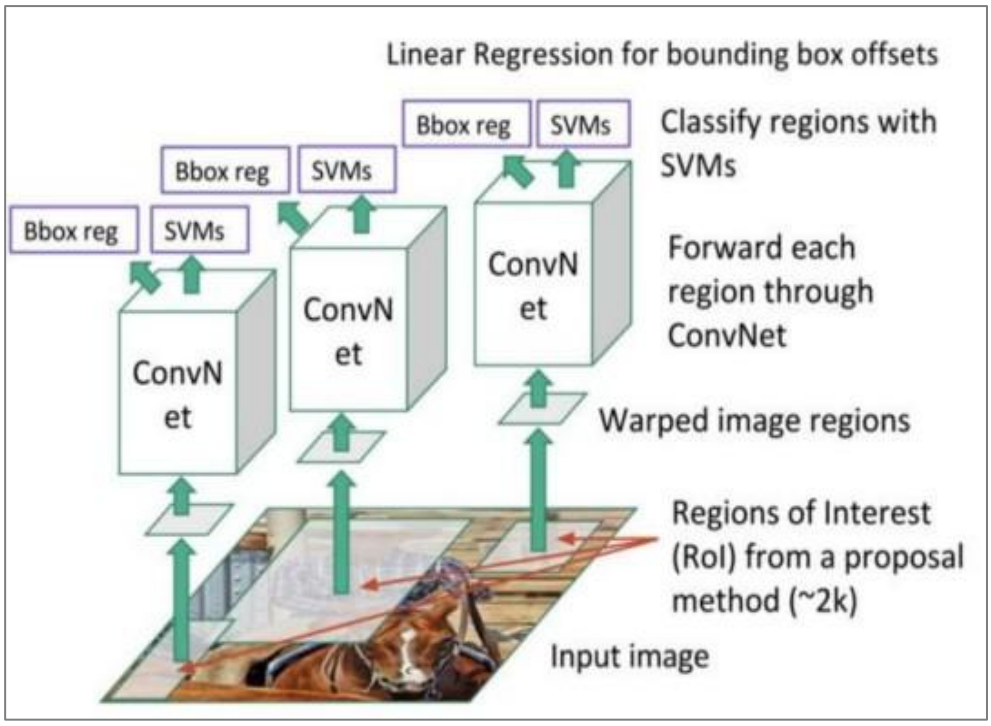


*Figure 12. Architecture du RCNN. [10]*

### 2.6.2 Le Fast R-CNN : Une amélioration optimisée de R-CNN

Le Fast R-CNN constitue une évolution significative du modèle R-CNN, visant à résoudre son principal inconvénient : la lenteur du traitement. Proposé pour réduire les temps de calcul, le Fast R-CNN introduit une méthode plus efficace en exploitant une seule convolution pour l'ensemble de l'image. Plutôt que d'appliquer un CNN séparément pour chaque proposition de région, cette approche utilise une seule convolution globale pour extraire une carte de caractéristiques de l'image entière. Ces cartes permettent de capturer les informations essentielles sur les objets potentiels présents dans l'image.

Après l'obtention des cartes de caractéristiques, la technique de recherche sélective (Selective Search) est appliquée pour identifier les ROI. Chaque région est ensuite normalisée pour une analyse approfondie via des couches entièrement connectées (Fully Connected Layers). Ces couches produisent un vecteur de caractéristiques qui alimente un classificateur Softmax pour identifier les objets. En parallèle, une régression linéaire est utilisée pour ajuster précisément les cadres de délimitation. Cette méthode, illustrée par la Figure 13, offre une précision élevée tout en réduisant considérablement le temps nécessaire pour analyser une image.

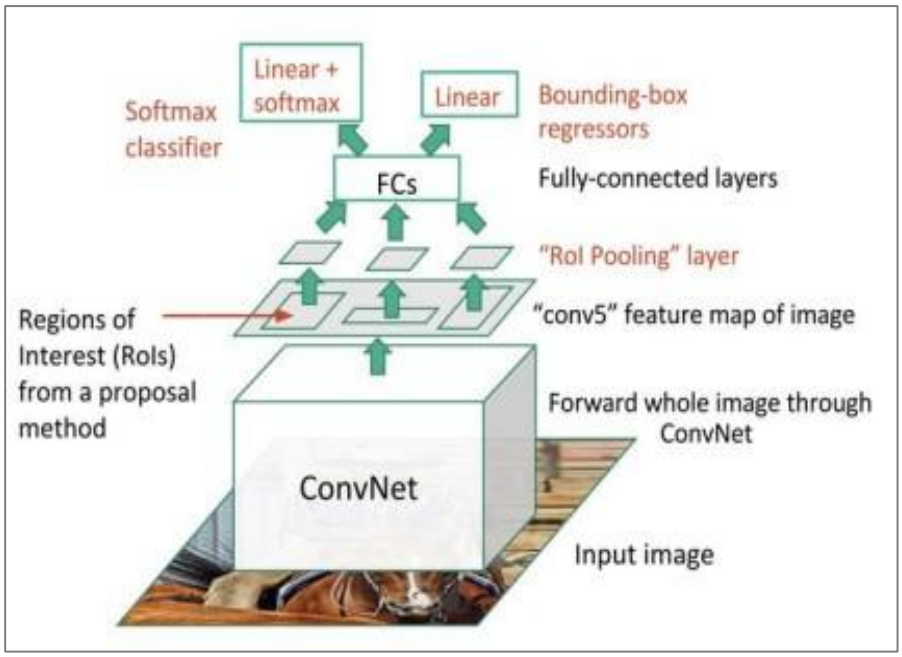


*Figure 13. Architecture du Fast R-CNN. [11]*

### 2.6.3 Le faster R-CNN

Le Faster R-CNN, dont l'architecture est illustrée à la Figure 14, constitue une amélioration majeure par rapport aux approches traditionnelles de détection d'objets en intégrant directement le processus de proposition de régions d'intérêt au sein du réseau de neurones convolutifs. Contrairement aux méthodes précédentes, qui s'appuyaient sur des algorithmes externes pour générer ROI, Faster R-CNN introduit un module spécialisé appelé *Region Proposal Network* (RPN), permettant de générer ces propositions de manière efficace et rapide. Le processus débute par l'extraction des cartes de caractéristiques (feature maps) à partir des images d'entrée à l'aide d'un réseau convolutif préentraîné, tel qu'une CNN formée sur la base ImageNet. Ces cartes contiennent des informations clés sur les motifs visuels détectés dans l'image. Le RPN analyse ces cartes pour proposer des régions susceptibles de contenir des objets. Chaque région est évaluée selon deux critères : la probabilité qu'elle contienne un objet et la qualité de la proposition de boîte englobante (bounding box). Les régions les plus prometteuses sont ensuite transmises à l'étape suivante.

Dans la phase suivante, les régions d'intérêt sélectionnées passent par une étape de *pooling*, qui adapte les tailles des régions pour les rendre compatibles avec les couches pleinement connectées du réseau. Cette étape garantit que les informations des régions sont traitées de manière uniforme, indépendamment de leur taille d'origine. Une fois cette mise à l'échelle effectuée, chaque région est classée en fonction des classes d'objets possibles. Simultanément,

la position de la boîte englobante est raffinée à l'aide d'un processus de régression, afin d'augmenter la précision de la localisation.

Le modèle Faster R-CNN présente plusieurs avantages significatifs. L'intégration du RPN au sein du réseau permet de réduire considérablement le temps de calcul par rapport aux approches précédentes, tout en maintenant une précision élevée. De plus, l'utilisation d'une architecture end-to-end facilite l'entraînement conjoint des différents modules, garantissant une cohérence entre la génération des propositions et la classification des objets.

Dans le cadre de ce mémoire, Faster R-CNN pourrait être utilisé pour analyser des images oculaires, telles que celles générées par le suivi du regard, afin de localiser les régions critiques associées aux anomalies visuelles. Sa capacité à combiner efficacement la localisation et la classification en fait une méthode robuste pour des applications comme le dépistage du strabisme. La Figure 14 illustre clairement les étapes de cette architecture, depuis l'extraction des caractéristiques jusqu'à la classification et la régression des boîtes englobantes. Cette approche démontre son efficacité dans des tâches nécessitant une précision élevée et une détection rapide.

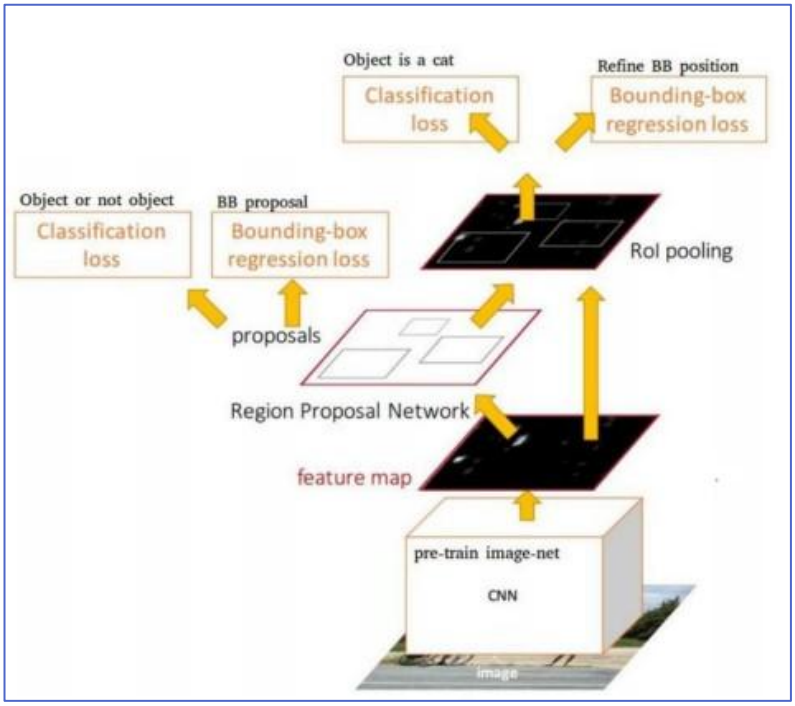


*Figure 14. Architecture du faster R-CNN [12]*

### 2.6.4 L’algorithme SSD

Le modèle SSD, dont l’architecture est illustrée à la Figure 15, est un réseau neuronal convolutionnel conçu pour la détection d'objets en temps réel. Il se distingue par sa capacité à générer des prédictions pour plusieurs classes et à localiser des objets à différentes échelles, tout en conservant des performances élevées en termes de rapidité et de précision.

Le modèle commence par une base convolutionnelle dérivée du réseau VGG-16, qui est utilisée pour extraire des cartes de caractéristiques (feature maps) à partir des images d'entrée. Ces cartes de caractéristiques capturent les informations essentielles sur les motifs visuels présents dans l’image. Les couches convolutives Conv5_3 de VGG-16 servent de point de départ pour générer les premières prédictions. À ces cartes s’ajoutent des couches supplémentaires (Extra Feature Layers), qui introduisent des résolutions plus petites et permettent ainsi de détecter des objets à différentes échelles.

Chaque couche convolutive dans l’architecture SSD est utilisée pour prédire des boîtes englobantes (bounding boxes) et des scores de confiance pour les classes d’objets. Ces prédictions sont réalisées à l’aide de filtres 3x3, qui permettent d’extraire des informations locales tout en maintenant une efficacité computationnelle. Le modèle génère jusqu'à 8732 prédictions par classe pour une image donnée, offrant une couverture complète des objets potentiels dans l’image.

L'étape finale consiste en une suppression non maximale (*Non-Maximum Suppression*, NMS), qui élimine les prédictions redondantes en ne conservant que celles avec les scores de confiance les plus élevés pour chaque objet détecté. Cela garantit une représentation concise et précise des résultats finaux. Avec une précision moyenne (mAP) de 74,3 % et une cadence de 59 images par seconde, le SSD constitue une solution performante pour les applications nécessitant une détection rapide et précises, telles que la reconnaissance des mouvements oculaires ou le dépistage des anomalies visuelles.

Dans le contexte de ce mémoire, le modèle SSD pourrait être utilisé pour analyser les images oculaires générées, telles que les images GaDe, afin de localiser et classifier automatiquement les caractéristiques associées au strabisme. Grâce à sa capacité à traiter les données visuelles à différentes échelles, le SSD s’avère particulièrement efficace pour détecter des variations subtiles dans les zones oculaires et améliorer ainsi les performances globales du système. Plus de détails sur le SSD sont présentés dans [13].

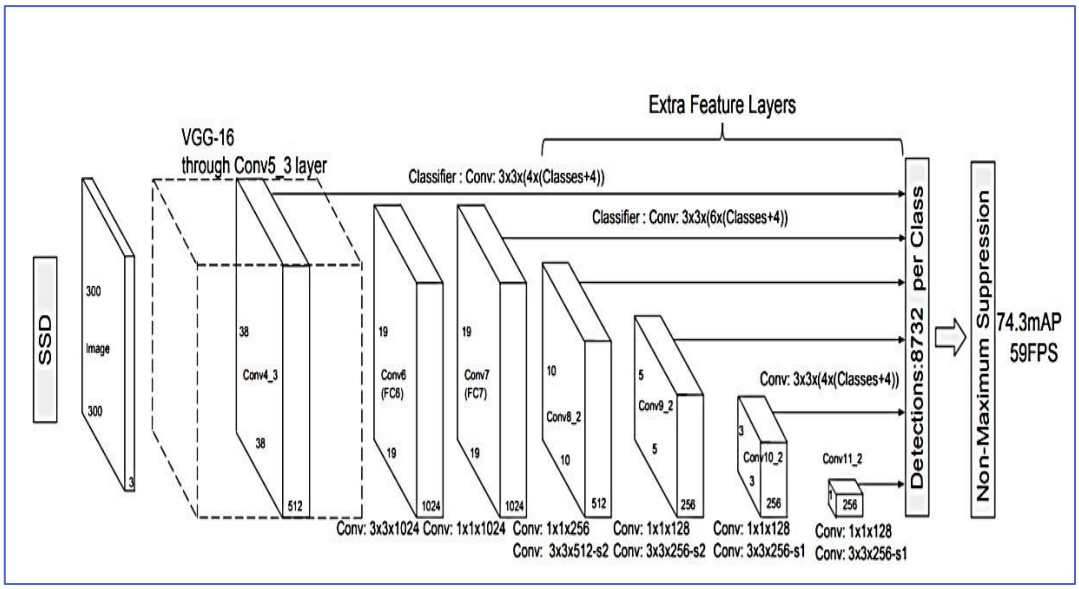


*Figure 15. L'architecture du modèle SSD [13]*

### 2.6.5 Architecture YOLO

YOLO, développé par Joseph Redmond et présenté dans [14], est un modèle d'architecture combinée et est extrêmement rapide. YOLO adopte une approche totalement différente. Il ne regarde l'image entière qu'une seule fois et passe par le réseau une fois pour détecter les objets. D'où le nom You Only Look Once. Il est très rapide, ce qui explique sa grande popularité : l'algorithme YOLO atteint environ **45 images par seconde,** soit une cadence nettement supérieure à celle de détecteurs classiques comme **Fast R-CNN** (environ 0,5) ou **SSD-300** (environ 7). Cependant, YOLOV4, que nous allons utiliser pour notre projet, est encore plus rapide que les versions précédentes de YOLO. Nous avons choisi **YOLOv4** parce qu'il offre un excellent équilibre entre rapidité et précision : son backbone allégé (CSPDarknet-53) réduit le nombre d'opérations sans sacrifier la profondeur du réseau ; un chemin d'agrégation amélioré (PANet) combine efficacement les détails fins et les informations globales ; enfin, de petites astuces d'entraînement (Mosaic, CIoU-loss) et des modules légers (Spatial Pyramid Pooling, attention SE) renforcent la justesse des prédictions sans ralentir l'inférence. Dans la pratique, ces améliorations lui permettent de conserver une cadence avoisinant la soixantaine d'images par seconde sur GPU tout en gagnant plusieurs points de précision par rapport à YOLOv3, ce qui en fait la solution la plus adaptée aux exigences temps réel de notre projet.L'étude des différentes approches de détection du strabisme, qu'elles soient cliniques ou basées sur des méthodes d'IA, a permis de mettre en évidence les avantages des modèles de vision par ordinateur, en particulier YOLOV4. Ce dernier se distingue par sa rapidité et son

efficacité dans la reconnaissance d'objets en temps réel, ce qui justifie son choix comme modèle d'apprentissage pour notre étude. Le chapitre suivant se consacre à une analyse approfondie de YOLOV4, en détaillant son architecture, son fonctionnement et ses principales composantes. Nous y décrivons également le processus de préparation des données nécessaires à l'entraînement du modèle, ainsi que les étapes méthodologiques mises en place pour optimiser ses performances.

# CHAPITRE 3. MÉTHODE DE DÉTECTION DU STRABISME AVEC L'ARCHITECTURE YOLOV4

Dans ce chapitre, nous décrirons en détail la méthodologie adoptée pour concevoir et entraîner un modèle basé sur l'architecture YOLOV4 afin de détecter le strabisme de manière efficace et précise. L'objectif principal est d'adapter le modèle préentraîné YOLOV4 à notre propre base de données, composée d'images spécifiquement collectées et annotées pour cette étude. Nous explorerons les différentes étapes du processus, depuis la préparation des données jusqu'à l'évaluation des performances du modèle. Cette méthodologie repose sur des techniques avancées de traitement d'images et d'apprentissage automatique, intégrant les spécificités de la détection des régions oculaires. Ainsi, ce chapitre vise à fournir une vue complète de la démarche suivie pour adapter YOLOV4 à un problème de détection de strabisme tout en garantissant la robustesse et la fiabilité des résultats.

## 3.1 Notions

### 3.1.1 Présentation de l'architecture de YOLOV4

L'architecture de YOLOV4, comme illustrée dans la Figure 16, est divisée en trois parties fonctionnelles : le *backbone*, le *neck* et la tête. Ce découpage permet une modularité accrue et une spécialisation des différentes étapes du traitement. Le *backbone*, souvent préentraînée sur des bases de données comme ImageNet, extrait les caractéristiques visuelles essentielles des images d'entrée. Ces caractéristiques sont ensuite transmises au *neck*, qui combine les informations issues de différentes couches pour améliorer la détection d'objets, en particulier ceux de tailles variées. Enfin, la tête produit les prédictions finales, comprenant les boîtes englobantes et la classification des objets détectés. Cette organisation repose sur des innovations techniques telles que les connexions résiduelles pondérées (*Weighted Residual Connections*, WRC), qui facilitent la transmission des gradients dans les couches profondes, et les connexions partielles par étapes croisées (*Cross Stage Partial Connections*, CSP), qui réduisent la redondance des calculs tout en maintenant une extraction efficace des caractéristiques. L'autoapprentissage inversé (SAT) optimise automatiquement les paramètres des prédictions, renforçant encore les performances du modèle.

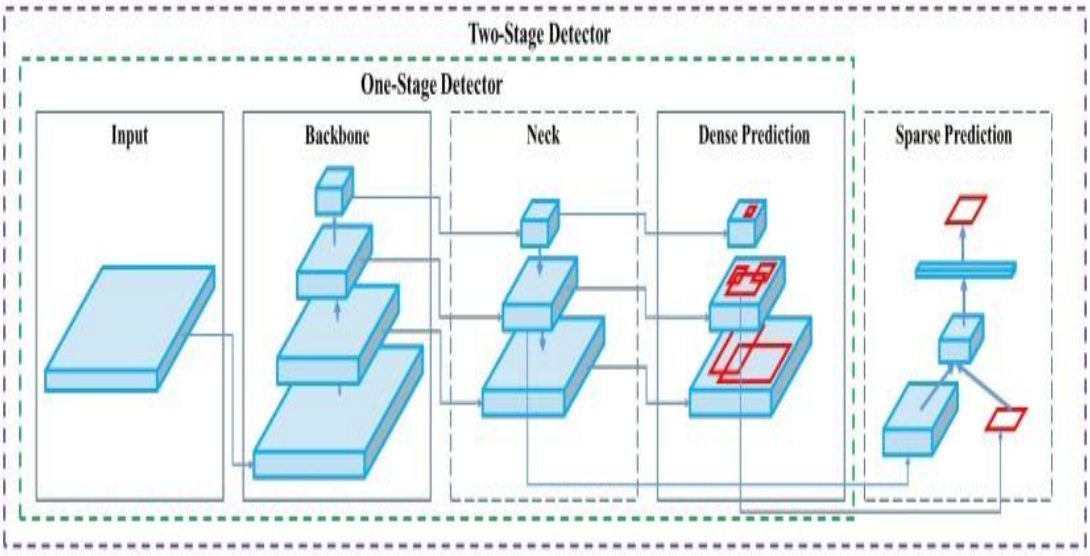


*Figure 16. Architecture de YOLOV4[14]*

La Figure 17 met en évidence les transitions fluides entre ces différentes parties et l'intégration de techniques de régularisation comme DropBlock (une technique avancée de régularisation utilisée dans les réseaux neuronaux pour améliorer leur capacité de généralisation et réduire le surapprentissage ), ainsi que des méthodes d'augmentation des données telles que la mosaïque, qui augmentent la diversité des exemples d'entraînement et améliorent la robustesse du modèle face aux variations des données d'entrée. De plus, la fonction CIoU *(Complete Intersection Over Union*) optimise les boîtes englobantes pour garantir une localisation précise des objets. Grâce à ces caractéristiques, YOLOV4 allie rapidité et précision, en s'adaptant parfaitement à des applications telles que la reconnaissance des anomalies oculaires dans des images biomédicales complexes, comme les données GaDe utilisées pour le dépistage automatisé du strabisme.

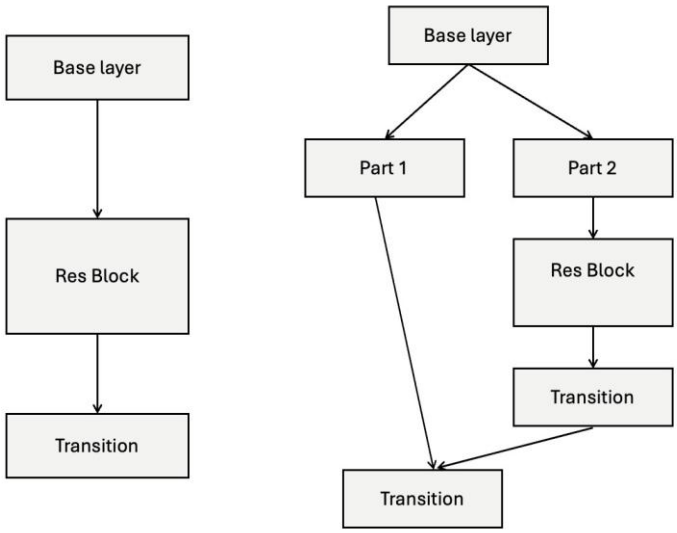


*Figure 17 architecture du CSPnet (Cross Stage Partial Network) [23]*

Le CSPDarknet (*Cross Stage Partial Darknet*) comprend 27,6 millions de paramètres et peut atteindre une vitesse de traitement de 66 images par seconde sur GPU (*Graphics Processing Unit*), ce qui en fait une meilleure architecture comparée au CSPResnext50 et à l'EfficientNetB3, comme l'indique le tableau 1:

*Tableau 1 Performances du cspDarknet53*

| Modèle Backbone | Résolution d'entrée du réseau | Taille du champ réceptif | Nombre de paramètres | Taille moyenne des sorties de couche (LxHxC) | BFLOPs (résolution 512x512) | FPS (GPU RTX 2070) |
|---|---|---|---|---|---|---|
| CSPResNext50 | 512x512 | 425x425 | 20.6 M | **1058 K** | 31 (15,5 FMA) | 62 |
| CSPDarknet53 | 512x512 | 725x725 | **27.6 M** | 950 K | 52 (26,0 FMA) | **66** |
| EfficientNet-B3 (le nôtre) | 512x512 | 1311x1311 | 12.6 M | 668 K | 11 (5,5 FMA) | 26 |

Ces architectures sont évaluées selon plusieurs critères tels que la résolution d'entrée, la taille du champ réceptif, le nombre de paramètres, la complexité en FLOPs (*Floating Point Operations per Second*), et les images par seconde (*Frames Per Second*, FPS) calculées sur un GPU RTX 2070. La résolution d'entrée fait référence à la taille des images utilisées comme données d'entrée pour entraîner le réseau. Tous les modèles présentés (CSPResNeXt50, CSPDarknet53, et EfficientNet-B3) utilisent une résolution d'entrée de 512×512 pixels, sauf EfficientNet-B3, qui est testée avec une résolution de 1311×1311 pixels. Cette différence de résolution a un impact direct sur le champ réceptif, c'est-à-dire la zone de l'image qu'une unité dans une couche donnée peut "voir". CSPDarknet53, avec un champ réceptif de 725×725, se situe dans une position intermédiaire, lui permettant de capter des informations globales et détaillées des images, ce qui est essentiel pour des tâches telles que la détection d'objets. En ce qui concerne les paramètres et la complexité computationnelle, CSPDarknet53 possède 27,6 millions de paramètres, un compromis idéal entre la taille du modèle et sa capacité à traiter des

tâches complexes. En comparaison, CSPResNeXt50 a 20,6 millions de paramètres, ce qui en fait un modèle plus léger, tandis qu'EfficientNet-B3, avec seulement 12 millions de paramètres, est conçu pour être particulièrement compact. En termes de BFLOPs (*Billions of Floating Point Operations*), CSPDarknet53 affiche 590 kFLOPs, le plaçant entre CSPResNeXt50 (1085 kFLOPs) et EfficientNet-B3 (668 kFLOPs). Cette caractéristique rend CSPDarknet53 adapté pour des environnements nécessitant une haute performance sans pour autant surcharger les ressources matérielles. Concernant la performance mesurée en FPS, CSPDarknet53 atteint un score impressionnant de 66 FPS sur un GPU RTX 2070, surpassant EfficientNet-B3 (26 FPS) et se plaçant légèrement au-dessus de CSPResNeXt50 (62 FPS). Ce résultat démontre que CSPDarknet53 est particulièrement bien optimisé pour des applications nécessitant une rapidité d'exécution, comme la reconnaissance d'images en temps réel ou la détection d'objets dans des flux vidéo. La taille moyenne des couches (W×H×C) indique la capacité du réseau à gérer les détails spatiaux et de profondeur des caractéristiques extraites. CSPDarknet53, avec une taille moyenne de 960 k, combine efficacement des caractéristiques globales et locales, équilibrant précision et performance. En conclusion, CSPDarknet53, au regard des métriques présentées dans le tableau, offre un excellent équilibre entre performance et efficacité. Avec une résolution et un champ réceptif adaptés, un nombre de paramètres optimisé, et des FPS élevés, cette architecture se positionne comme un choix idéal pour des applications nécessitant une détection rapide et précise. Comparé à EfficientNet-B3, qui mise davantage sur la compacité, et CSPResNeXt50, qui privilégie des performances légèrement plus élevées, mais à un coût computationnel supérieur, CSPDarknet53 représente une solution polyvalente, capable de répondre à une large gamme de scénarios.

### 3.1.2 Explication des parties de l'architecture YOLOV4

#### 3.1.2.1 La couche d'entrée (input)

La couche d'entrée (input) de l'architecture YOLOV4 est responsable de la préparation initiale des données avant leur traitement par le réseau. Elle reçoit une image brute en entrée, qui est redimensionnée à une taille fixe (par exemple, 416×416 pixels et 3 canaux de couleur RGB dans notre cas) et normalisée pour que les valeurs des pixels soient comprises entre 0 et 1, améliorant ainsi la stabilité de l'entraînement. Elle peut également inclure des techniques d'augmentation des données, telles que les rotations, les ajustements de luminosité et les inversions, afin d'améliorer la robustesse du modèle. En parallèle, les annotations des objets

(comme les boîtes englobantes) sont converties dans un format compatible avec YOLOV4, exprimant les coordonnées, les dimensions, et la classe de chaque objet. Enfin, cette couche transmet les données normalisées à la couche Backbone pour l'extraction des caractéristiques.

#### 3.1.2.2 Le Backbone

Le backbone est considéré comme la colonne vertébrale d'un réseau neuronal convolutif. Dans l'architecture de YOLOV4, cette composante joue un rôle crucial en permettant l'extraction des caractéristiques pertinentes à partir des images d'entrée. Contrairement aux versions précédentes de YOLO qui utilisaient le modèle **Darknet53**, YOLOV4 introduit une amélioration significative grâce à l'utilisation de **CSPDarknet53**, une version optimisée basée sur l'architecture **CSPNet**. Pour mieux comprendre le fonctionnement de cette structure, il est essentiel d'explorer les principes fondamentaux du CSPNet et son intégration dans CSPDarknet53.

Le **CSPNet** a été introduit en 2019 par Chien-Yao Wang et Hong-Yan Mark Lia, avec pour objectif d'améliorer les performances des modèles de vision par ordinateur tout en réduisant le problème de surapprentissage (overfitting). En particulier, CSPNet vise à optimiser la manière dont les caractéristiques sont extraites et combinées dans les réseaux neuronaux profonds, ce qui est essentiel pour les tâches de classification et de détection d'objets. L'innovation principale de **CSPNet** réside dans l'approche dite de **"cross stage partial"**, qui divise les caractéristiques extraites par les couches convolutionnelles en deux parties distinctes, comme illustré dans la Figure 17. Cette division commence dès la **Base Layer**, où les caractéristiques sont séparées en deux flux : **Part 1** et **Part 2**. La **Part 1** contourne directement le bloc résiduel (**Res Block**) sans subir de transformations, tandis que la **Part 2** traverse le **Res Block**, où des opérations convolutionnelles complexes sont effectuées pour extraire des caractéristiques détaillées. Après cette étape, les deux flux sont fusionnés lors de la phase de **Transition**, permettant une intégration harmonieuse des informations globales et locales. Cette architecture unique assure une meilleure gestion des informations extraites, car la bifurcation et la fusion améliorent la représentation globale des données tout en réduisant la redondance dans les calculs. Par ailleurs, cette approche optimise l'utilisation de la mémoire, car les caractéristiques sont partiellement réutilisées à différents stades du réseau. Enfin, en combinant des caractéristiques de différents niveaux de traitement, le CSPNet améliore la robustesse du

modèle face aux variations des données d'entrée, ce qui le rend particulièrement adapté pour des tâches complexes telles que la détection d'objets et la segmentation d'images.

Le **CSPDarknet53 est l'épine dorsale de** YOLOV4, utilisé comme backbone dans YOLOV4, s'appuie directement sur les principes de CSPNet tout en adaptant son implémentation pour répondre aux besoins spécifiques des tâches de détection en temps réel. Cette architecture conserve les blocs résiduels (ResBlocks) du Darknet53 original, mais les enrichit grâce aux connexions croisées partielles. La structure générale de CSPDarknet53 est illustrée dans la Figure 19, qui montre comment les caractéristiques sont divisées, transformées et recombinées à chaque étape du réseau. CSPDarknet53 offre plusieurs avantages par rapport à Darknet53, notamment une efficacité accrue grâce à la réduction des redondances dans les calculs, ce qui accélère le traitement et le rend idéal pour des applications en temps réel, comme la détection d'objets dans les flux vidéo. Il améliore également la représentation des caractéristiques en fusionnant des informations à différents niveaux, permettant au réseau de capturer à la fois des détails globaux et locaux. De plus, la gestion optimisée des gradients renforce sa robustesse, réduisant les problèmes de dégradation des performances dans les réseaux profonds. En intégrant CSPDarknet53 comme backbone, YOLOV4 bénéficie d'une structure d'extraction des caractéristiques hautement optimisée. Ce backbone permet d'identifier des motifs complexes dans les images, même dans des conditions difficiles telles qu'un faible éclairage ou la présence de bruit. Par exemple, dans des applications médicales comme le dépistage des anomalies oculaires, CSPDarknet53 peut extraire des informations subtiles sur la position des pupilles ou les réflexions cornéennes.

De plus, cette architecture contribue à réduire le coût computationnel global de YOLOV4 tout en augmentant sa précision. En comparaison avec d'autres architectures comme EfficientNet ou ResNet, CSPDarknet53 offre un équilibre idéal entre rapidité et précision, le rendant particulièrement adapté pour des tâches exigeant des prédictions en temps réel.

En résumé, le backbone est une composante essentielle de tout réseau neuronal convolutif, et l'adoption de CSPDarknet53 dans YOLOV4 marque une avancée significative dans l'optimisation des performances des modèles de vision par ordinateur. En tirant parti des innovations introduites par CSPNet, CSPDarknet53 améliore la capacité d'extraction des caractéristiques tout en réduisant les exigences en ressources. Cette efficacité en fait un choix privilégié pour des applications variées, allant de la reconnaissance d'objets généraux à des cas d'utilisation spécifiques comme la télémédecine ou la surveillance vidéo en temps réel.

Les Figures 18 et 19 présentent une comparaison entre le darknet 53 utilisé dans la version précédente YOLOV3 et le CSPdarknet53 utilisé dans la version d'YOLOV4

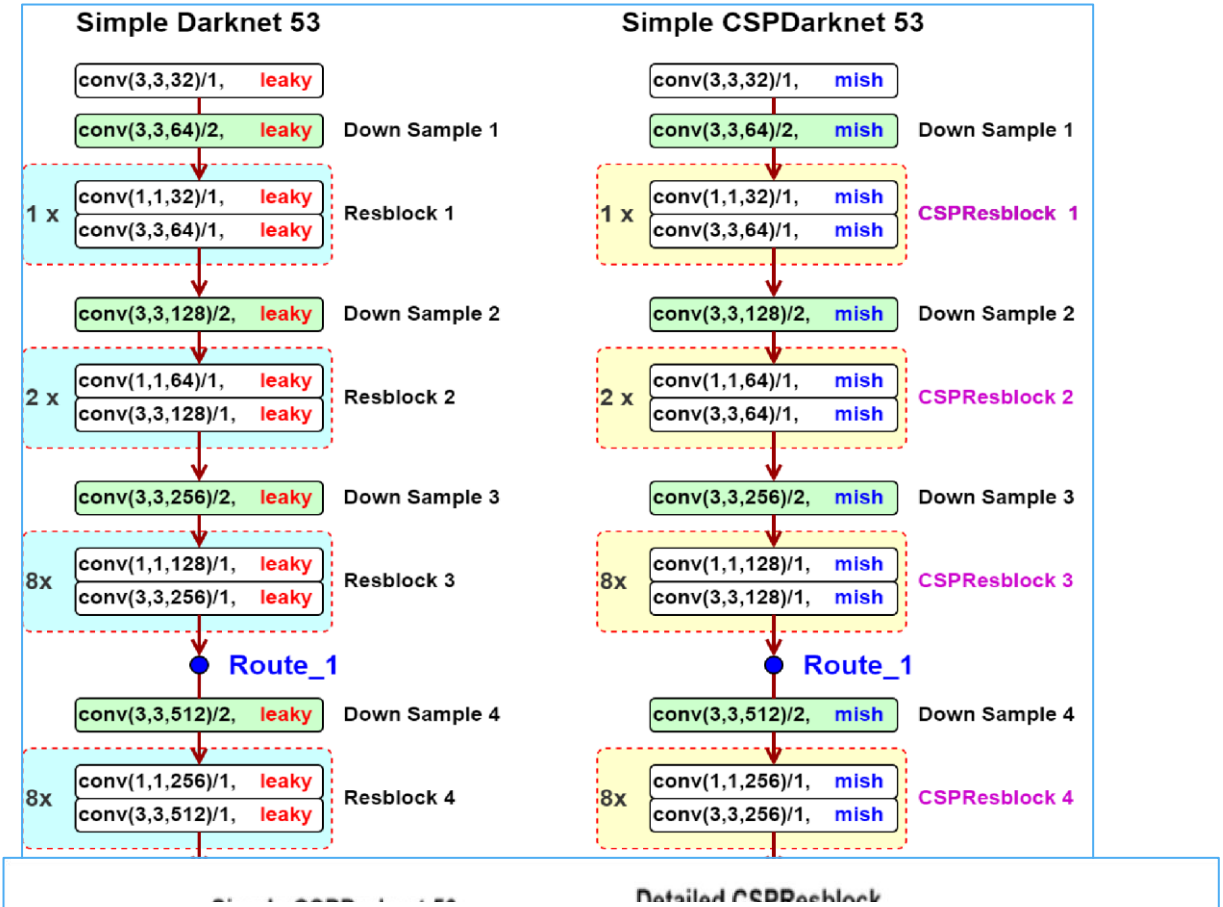


*Figure 18. Comparaison entre l'ancien Darknet 53 utiliser par YOLOV3 et le CSPDarknet 53 utilise par YOLOV4 [24]*

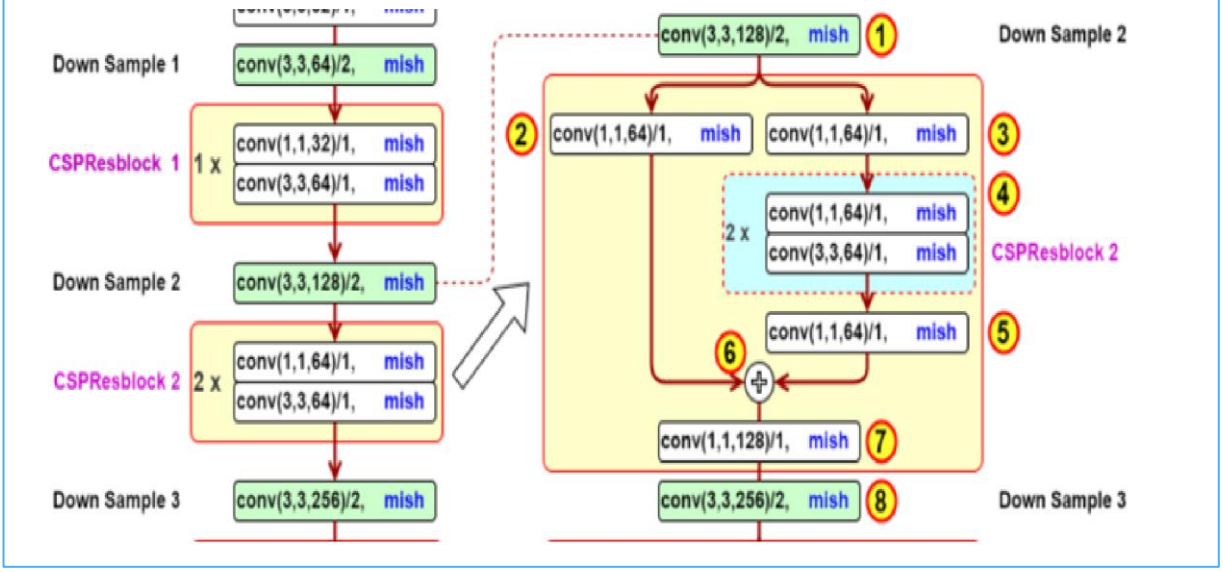


*Figure 19. Implémentation de la technique de la CSP dans les blocs résiduels de l'architecture du CSPDarknet 53 [24]*

L'architecture **CSPDarknet53** représente une évolution notable par rapport à l'ancien modèle **Darknet53**, utilisé notamment par YOLOV3. Cette nouvelle architecture, employée dans YOLOV4, améliore considérablement l'efficacité du modèle grâce à l'intégration de la technique CSP. La Figure 18 illustre une comparaison directe entre les deux versions, mettant en évidence les différences fondamentales dans leur conception. Par ailleurs, la Figure 19 détaille l'implémentation de cette technique CSP dans les blocs résiduels de CSPDarknet53. Dans cette section, nous présentons successivement les éléments clés qui composent cette architecture : les couches de convolution, la fonction d'activation mish (A Self Regularized Non-Monotonic Activation Function), et les blocs résiduels (Resblocks).

**Les couches de convolution** constituent la base de l'architecture CSPDarknet53, tout comme dans Darknet53. Chaque couche est définie par des filtres convolutifs 3×33×3, associés à des fonctions d'activation spécifiques. Ces couches permettent d'extraire les caractéristiques essentielles des images d'entrée tout en réduisant leur dimensionnalité à mesure que l'information traverse les niveaux plus profonds du réseau. Dans CSPDarknet53, ces couches sont optimisées grâce aux CSP, qui divisent les cartes de caractéristiques en deux parties. Une partie est directement transmise à l'étape suivante, tandis que l'autre traverse une série de transformations non linéaires avant d'être recombinée. Cette approche réduit la redondance des calculs tout en augmentant la diversité des caractéristiques extraites, ce qui améliore la capacité d'apprentissage et réduit les exigences en mémoire. La Figure 20 illustre l'opération de convolution, une étape clé dans les réseaux de neurones convolutifs (CNN), utilisée pour extraire des caractéristiques importantes à partir d'une image ou d'autres données en entrée.

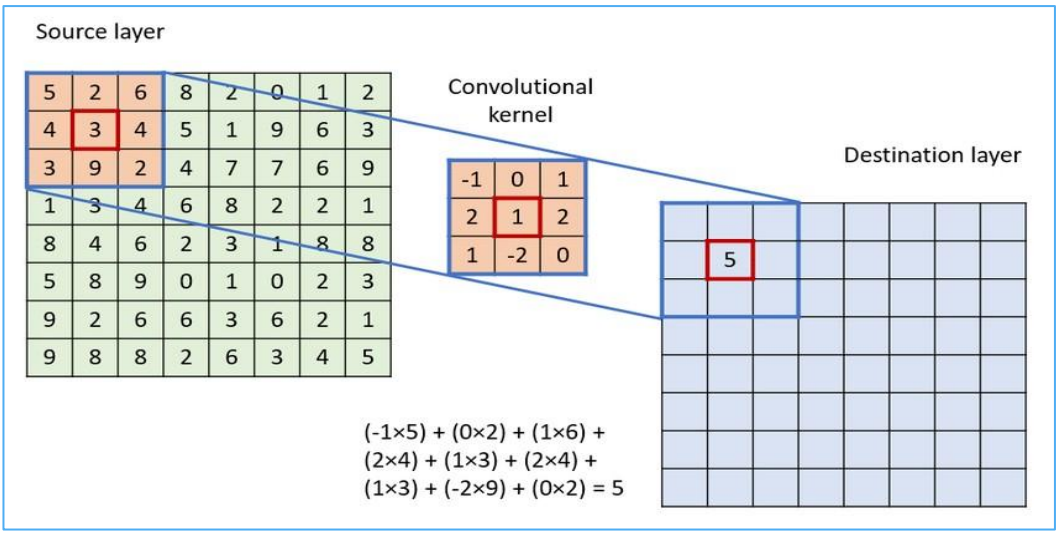


*Figure 20. Illustration de l'opération de convolution [25]*

La convolution commence par la matrice de la couche source, située à gauche, qui représente les données initiales, comme une image ou une carte de caractéristiques. Chaque élément de cette matrice contient une valeur numérique correspondant à l'intensité d'un pixel ou d'une caractéristique particulière. Au centre, le noyau convolutionnel est une petite matrice 3×3, également appelée filtre, qui agit comme une fenêtre glissant sur la matrice source. Ce noyau applique une transformation locale en multipliant ses propres valeurs par les valeurs correspondantes de la matrice source, puis en additionnant les résultats. Par exemple, dans la région illustrée, le produit des valeurs du noyau avec celles de la matrice source aboutit à une somme totale de 5, qui est ensuite placée dans la matrice de destination, représentée à droite. La matrice de destination contient donc le résultat de ces transformations pour chaque région parcourue par le noyau sur la matrice source. Ce processus se répète pour toute la matrice source, ce qui génère une nouvelle matrice, généralement de taille réduite, en fonction des dimensions du noyau et de l'application éventuelle d'un remplissage (padding). L'opération de convolution permet ainsi de détecter des motifs locaux importants dans l'image, tels que les bords, les textures, ou les motifs répétitifs, tout en réduisant les dimensions et en capturant des informations essentielles pour des tâches comme la classification ou la détection d'objets.

**La fonction d'activation mish** est une fonction non linéaire qui a été proposée comme alternative aux fonctions d'activation traditionnelles telles que ReLU, tanH et sigmoïde. Elle a été introduite par DIGANTA MISRA en 2019. Elle est caractérisée par une courbe douce qui est convexe pour les valeurs négatives et concaves pour les valeurs positives, ce qui la rend

adaptée à la rétropropagation du gradient dans les réseaux de neurones profonds. Sa formule est donnée par l'équation (1)

$$f(x) = x + \tan \quad (\ln \quad (1 \quad + e\hat{}x \quad ) \quad ) \quad (1)$$

La courbe de cette fonction, illustrée par la Figure 21, présente plusieurs avantages. Elle permet une propagation plus fluide des gradients, ce qui favorise un apprentissage plus stable et améliore la précision du modèle. De plus, mish conservent des valeurs négatives, ce qui évite une perte d'information lors des activations. Cette fonction est intégrée dans toutes les couches de CSPDarknet53, rendant le modèle plus robuste pour des tâches complexes comme la détection d'objets en temps réel.

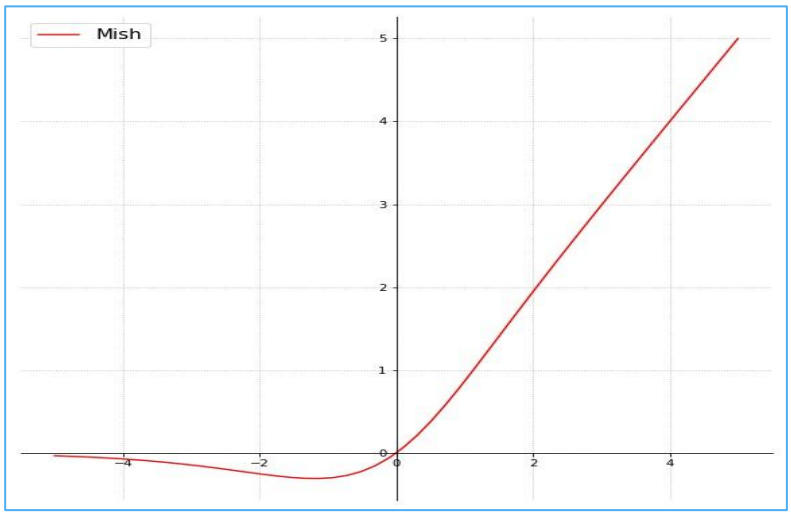


*Figure 21. Fonction d'activation mish*

Les blocs résiduels, ou **Resblocks** sont une composante essentielle de CSPDarknet53, permettant de résoudre le problème de la dégradation des gradients dans les réseaux profonds. Ces blocs introduisent des connexions directes (skip connections) entre les couches, ce qui facilite la transmission des informations importantes à travers le réseau.

Dans CSPDarknet53, les Resblocks sont adaptés en intégrant des connexions croisées partielles (CSP), comme illustrées dans la Figure 19. Ces blocs modifiés suivent un processus en trois étapes : d'abord, les cartes de caractéristiques sont divisées en deux parties distinctes, permettant une séparation des flux d'informations. Ensuite, l'une des parties subit des transformations non linéaires, comprenant des convolutions et l'application de la fonction mish, renforçant ainsi l'extraction des caractéristiques. Enfin, les deux parties sont recombinées, ce qui enrichit la diversité des informations transmises et améliore la capacité du réseau à

apprendre des représentations complexes. Cette structure améliore l'efficacité du modèle en réduisant les redondances et en augmentant la capacité de généralisation. De plus, elle réduit la consommation de mémoire et le coût computationnel, rendant CSPDarknet53 plus adapté pour des environnements contraints, comme les systèmes embarqués.

#### 3.1.2.3 Le cou du réseau (neck)

Le neck est la partie qui suit le backbone, et son rôle principal est de fusionner les caractéristiques spatiales provenant de différentes échelles dans le backbone. Cette fusion permet de capturer les informations contextuelles et de mieux localiser les objets dans l'image. Pour atteindre cet objectif, plusieurs modules sont utilisés pour améliorer la capacité du modèle à détecter les objets de tailles variées. Le **feature pyramid network (FPN)** est un module conçu pour capturer les informations à différentes échelles spatiales en utilisant une pyramide de caractéristiques spéciale allant du sommet vers la base. Cette approche permet au modèle de détecter efficacement des objets de tailles variées dans l'image. Ensuite, le ***Patch Agrégation Network*** **(PAN)** introduit un mécanisme de fusion hiérarchique qui relie les caractéristiques issues de différentes échelles spatiales. Grâce à l'utilisation de connexions résiduelles, le PAN améliore la représentation des objets dans l'image en agrégeant les caractéristiques pour une détection plus précise. Enfin, le **SPP (*spatial pyramid pooling*)** utilise une technique de pooling spatial afin d'agréger des informations provenant de différentes résolutions spatiales, enrichissant ainsi les caractéristiques utilisées par le modèle pour une meilleure localisation et classification des objets.

#### 3.1.2.4 La tête du réseau (*Head*)

La tête du réseau (Head) est une composante essentielle de l'architecture, responsable des prédictions finales, et elle se compose de deux parties principales : la dense prédiction et la sparse prédiction. La dense prédiction génère des prédictions pour un grand nombre de régions dans l'image, avec pour objectif de couvrir de manière exhaustive l'ensemble de celle-ci. Pour ce faire, le modèle dense divise l'image en une grille régulière et produit des prédictions pour chaque cellule de la grille. Ces prédictions incluent la détection d'objets, la génération de boîtes englobantes (bounding boxes), et la classification des objets détectés. En procédant ainsi, le modèle assure une analyse détaillée de l'image et garantit que les objets présents, même ceux de petite taille ou situés dans des zones complexes, sont pris en compte. Par ailleurs, la sparse prédiction vient compléter cette approche en optimisant son efficacité. Contrairement à la dense

prédiction, qui peut générer de nombreuses prédictions redondantes, la sparse prédiction applique la méthode de suppression NMS. Cette méthode élimine les prédictions superflues en ne conservant que celles qui présentent les scores de confiance les plus élevés. Cela permet de réduire considérablement la redondance et de garantir que seules les prédictions les plus fiables et pertinentes sont retenues. En combinant ces deux approches, la dense prédiction et la sparse prédiction, la tête du réseau parvient à réaliser un équilibre entre une couverture exhaustive de l'image et une efficacité optimale. Cela se traduit par une meilleure précision globale dans la détection des objets tout en réduisant les ressources nécessaires pour le traitement des prédictions multiples.

## 3.2 Processus de détection et fonction d'erreur de YOLOV4

### 3.2.1 Processus de détection avec YOLOV4

Le processus de détection des objets dans YOLOV4 repose sur une approche qui divise chaque image d'entrée en une grille de S×S cellules. Chaque cellule de la grille est chargée de détecter les objets dont le centre tombe dans son périmètre. Cela permet de simplifier le problème de détection en le décomposant en un ensemble de sous-problèmes locaux, rendant ainsi le processus plus efficace. Cette méthode est illustrée à travers la Figure 24 ci-dessous, qui montre la manière dont une image est segmentée en une grille où chaque cellule joue un rôle crucial dans la détection.

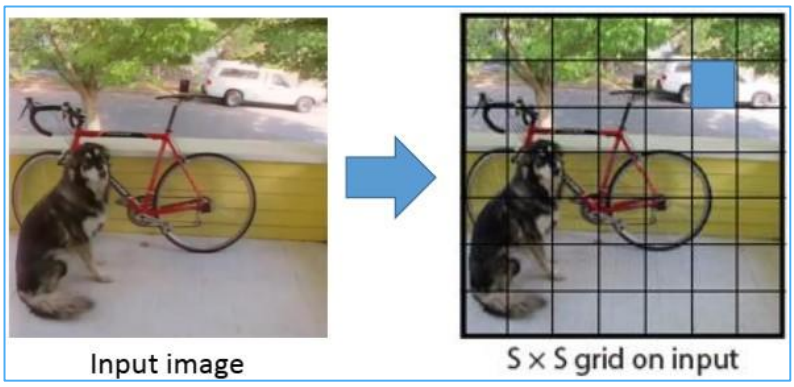


*Figure 22. Division de l'image d'entrée en une grille de S*S cellules [22]*

Chaque cellule de la grille génère un ensemble de prédictions, notamment des cadres englobants (bounding boxes) et des scores de confiance associés. Ces scores de confiance évaluent à quel point le modèle est certain qu'un objet est présent dans une boîte englobante particulière. Ils indiquent également à quel point la boîte prédite correspond avec précision à l'objet réel. Pour calculer ces scores, YOLOV4 utilise l'équation 2 suivante :

$$\Pr(Object) * IOU_{!\%"\&\%"\#\$}^{'} \quad (2)$$

Avec IOU (*Intersection Over Union*), représente le rapport entre l'intersection et l'union des zones de la boîte prédite et de la boîte de vérité terrain. Cette métrique joue un rôle clé pour évaluer la qualité des prédictions. En plus des cadres englobants et des scores de confiance, chaque cellule de la grille génère également des probabilités conditionnelles des classes, notées Pr ($Class_i|Object$). Ces probabilités sont conditionnées par la présence d'un objet dans une cellule donnée. Lors de la phase d'inférence, le modèle multiplie les probabilités de classe conditionnelles avec les scores de confiance pour produire des scores de confiance spécifiques aux classes pour chaque boîte englobante. Cette étape est exprimée par l'équation 3.

$$\Pr(\ Class_C \mid Object\ ) * \Pr(Object) * IOU_{!}^{\%"\&\%"\#\$'} = \Pr(Class_C) * IOU_{!}^{\%"\&\%"\#\$'} \qquad (3)$$

La Figure 23 illustre le processus de prédiction des boîtes englobantes et des scores de confiance par cellule, tandis que la Figure 24 montre la répartition des probabilités de classe uniques pour chaque cellule de la grille.

*Figure 23 Prédictions des boîtes englobantes et score de confiances par chaque cellule de la grille [22]*

*Figure 24. Répartition des probabilités des classes uniques par cellule de la grille [22]*

Une fois les prédictions générées, YOLOV4 applique une étape de filtrage pour éliminer les prédictions redondantes et conserver uniquement les plus pertinentes. Ce filtrage est basé sur la technique de suppression NMS, qui garantit que chaque objet détecté est représenté par une seule boîte englobante avec le score de confiance le plus élevé. En combinant les prédictions de boîtes englobantes et les probabilités de classe, le modèle parvient à produire une détection précise et exhaustive des objets présents dans l'image. La Figure 25 illustre ce processus, montrant comment les prédictions initiales sont affinées pour obtenir une détection correcte, avec des boîtes englobantes finales correspondant précisément aux objets présents dans l'image d'entrée.

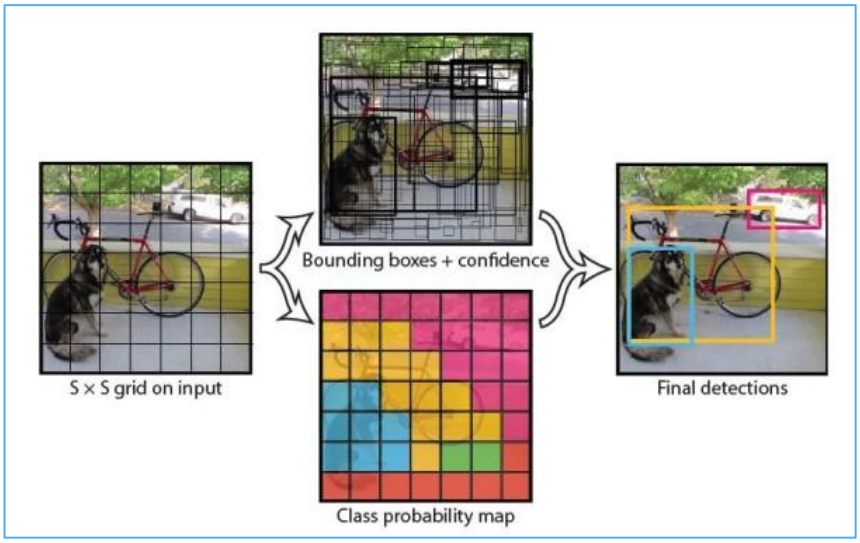


*Figure 25. Détection d'objet à l'aide de YOLO [22]*

Le processus de détection dans YOLOV4 repose sur une division méthodique de l'image en une grille et l'attribution de tâches de détection à chaque cellule. En combinant les prédictions locales des cellules avec des scores de confiance et des probabilités conditionnelles, le modèle parvient à détecter efficacement les objets tout en réduisant les erreurs grâce au filtrage NMS. Cette approche garantit une performance optimale, tant en termes de précision que de rapidité, ce qui en fait une solution de choix pour les applications en temps réel.

### 3.2.2 Fonction d'erreur (loss function) de YOLOV4

La fonction d'erreur (ou fonction de coût) de YOLOV4 est conçue pour mesurer l'écart entre les prédictions du modèle et les valeurs réelles des données d'entraînement. Cette fonction est

composée de plusieurs termes spécifiques, chacun correspondant à un aspect particulier du processus de détection : localisation des objets, prédiction des boîtes englobantes, scores de confiance et classification. La fonction d'erreur de YOLOV4 est exprimée la somme des équation 3, 4, 5 et 6 qui prend en compte tous les aspects essentiels de la détection d'objets (localisation, dimensions, confiance et classification), permettant un entraînement efficace et une convergence rapide du modèle.

### 3.2.2.1 Erreur de localisation

La localisation des objets est un élément central de la fonction d'erreur de YOLOV4. Le terme associé est représenté par :

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{obj}} \left[ (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \qquad (4)$$

**$S^2$** représente la taille de la grille utilisée pour diviser l'image en cellules, par exemple une grille 7×7, où chaque cellule est responsable de la détection d'objets dans sa région respective. **B** indique le nombre de boîtes englobantes prédites par chaque cellule, chacune décrivant un objet potentiel dans cette région. Les coordonnées **Xi, Yi** désignent le centre de la boîte réelle (ground truth), qui correspond à la position exacte de l'objet dans l'image, tandis que x̂ i, ŷiindiquent les coordonnées du centre de la boîte prédite par le modèle. L'indicateur binaire $\mathbf{1}^{obj}_{ij}$ est utilisé pour signaler si un objet est présent ou non dans une cellule donnée ; il prend la valeur 1 si un objet est détecté dans la cellule **i**, et 0 sinon. Ce mécanisme permet de concentrer le calcul des pertes uniquement sur les cellules contenant des objets, évitant ainsi les erreurs inutiles dans les zones vides. Enfin, **λcoord** est un coefficient de pondération qui ajuste l'importance de l'erreur de localisation dans la fonction de perte. Cela garantit que la précision des prédictions de position des boîtes englobantes est priorisée par rapport aux autres composantes de l'erreur, comme la classification ou la confiance. Cette structure garantit une détection robuste et précise en équilibrant localisation, classification, et confiance.

### 3.2.2.2 Erreur des dimensions des boîtes

Ce terme mesure l'écart entre les dimensions (largeur www et hauteur hhh) des boîtes prédites et celles de la vérité terrain. Il est défini comme suit :

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{obj}} \left[ (\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2 \right] \quad (5)$$

Les paramètres **wi** et **hi** représentent respectivement la largeur et la hauteur réelles des boîtes englobantes, connues sous le nom de "ground truth", qui décrivent la position exacte des objets dans l'image. Ces dimensions servent de référence pour évaluer la qualité des prédictions effectuées par le modèle. En revanche, **ŵi** et **ĥi** désignent les largeurs et hauteurs prédites des boîtes par le réseau neuronal. Ces prédictions sont obtenues après le traitement de l'image et sont comparées aux dimensions réelles à l'aide de fonctions de perte. Cette comparaison, souvent basée sur la différence entre les racines carrées des dimensions réelles et prédites, permet de calculer une erreur qui guide l'optimisation du modèle. L'utilisation des racines carrées ($\sqrt{\text{w}}$ $et$ $\sqrt{\text{h}}$) au lieu des dimensions brutes permet de mieux gérer les différences relatives entre les petites et grandes boîtes. Cela évite que de grandes erreurs sur des boîtes de grande taille dominent la fonction de coût.

#### 3.2.2.3 Erreur de confiance (confidence loss)

La confiance est une mesure de la probabilité qu'une boîte englobante contienne un objet. La fonction d'erreur intègre deux termes distincts pour gérer les boîtes contenant des objets et celles qui n'en contiennent pas :

$$\sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{obj}} (C_i - \hat{C}_i)^2 + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{noobj}} (C_i - \hat{C}_i)^2 \quad (6)$$

La pondération λnoobj est utilisée pour réduire l'impact des fausses prédictions sur les cellules sans objets, évitant ainsi que le modèle se concentre trop sur les cellules vides.

Les paramètres **Ci** et **Ĉi** représentent respectivement le score de confiance réel et le score de confiance prédit. Le score de confiance réel, **Ci**, indique la probabilité qu'une boîte englobante contienne effectivement un objet, tandis que le score prédit, **Ĉi**, est calculé par le modèle pour estimer cette même probabilité. L'indicateur binaire $\mathbf{1}^{obj}_{ij}$ est égal à 1 si une boîte contient un objet dans une cellule donnée et à 0 sinon, permettant de limiter les calculs uniquement aux zones pertinentes. Par ailleurs, la pondération **λnoobj** joue un rôle essentiel en réduisant l'impact des fausses prédictions sur les cellules qui ne contiennent pas d'objets. Cela évite que le modèle se concentre excessivement sur les cellules vides et améliore ainsi sa capacité à se focaliser sur les

régions de l'image où des objets sont effectivement présents, augmentant la précision globale des prédictions.

#### 3.2.2.4 Erreur de classification

La classification concerne l'identification correcte des classes des objets détectés. Ce terme est exprimé par :

$$\sum_{i=0}^{S^2} 1_i^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \tag{7}$$

Les paramètres **pi(c)** et **$\hat{P}$ i(c)** représentent respectivement la probabilité réelle et la probabilité prédite pour une classe donnée **c**. La probabilité réelle, **pi(c)**, correspond à la vérité terrain (ground truth), indiquant si une cellule contient un objet appartenant à une classe spécifique. En revanche, **$\hat{P}$ i(c)** est la probabilité estimée par le modèle pour cette même classe. Pour chaque cellule contenant un objet, la fonction de perte compare ces deux probabilités afin d'évaluer la différence entre la vérité terrain et les prédictions du modèle. Cette comparaison est effectuée pour toutes les classes possibles, ce qui permet de minimiser les erreurs de classification. En ajustant les paramètres du modèle pour réduire cette erreur, la précision globale des prédictions de classe est améliorée, garantissant une meilleure identification des objets présents dans l'image.

### 3.3 Préparation de la Base de Données pour l'Apprentissage

La première étape de notre projet consiste en la constitution et le traitement de la base de données, une phase essentielle, car la qualité de détection et la précision du modèle dépendent étroitement de la qualité des images utilisées. Pour cette expérimentation, nous avons collecté un total de 325 images réparties en deux classes distinctes : la première classe, représentant des cas normaux, comprend 156 images, tandis que la seconde classe, correspondant à des cas de strabisme, contient 169 images. Ces données ont été obtenues principalement à partir de Google Images et Bing, en utilisant la bibliothèque Python bing-image-downloader pour automatiser le processus de téléchargement. Les images, toutes issues au départ du format JPEG pour garantir l'uniformité du jeu de données, présentent des résolutions variables ; certaines contiennent même plusieurs cas de strabisme, enrichissant ainsi la diversité et favorisant la généralisation du modèle. À terme, l'emploi d'un format sans perte comme le PNG pourrait toutefois préserver davantage les micro-détails visuels et constituer une amélioration pertinente

pour accroître la précision. Une fois la collecte terminée, nous avons traité l'ensemble des images à l'aide d'un logiciel de marquage, permettant d'encadrer les zones d'intérêt avec des boîtes englobantes (bounding boxes) pour indiquer précisément les objets à détecter et classifier. Ce marquage a permis de générer des fichiers au format YOLO, contenant les classes des objets (normal ou strabisme) et les coordonnées des boîtes englobantes associées. De plus, un fichier nommé classes.txt a été créé pour répertorier les noms des différentes classes présentes dans les images, simplifiant ainsi le processus de classification pour le modèle YOLO. La Figure 26 illustre l'interface de marquage utilisée durant la phase de traitement des données visuelles. Cette interface permet de délimiter manuellement la région d'intérêt (ROI) correspondant à la zone oculaire du sujet, à l'aide d'un rectangle ajustable. L'objectif de ce marquage est de centrer l'analyse sur les deux yeux afin de préparer les images pour les étapes ultérieures du traitement, telles que l'entraînement ou l'évaluation d'un modèle de détection. Cette étape est essentielle pour garantir que les données fournies à l'algorithme sont précises et cohérentes, notamment dans le contexte d'une détection de strabisme ou d'une analyse du regard.

*Figure 26. Interface de marquage*

Les données collectées pour l'entraînement du modèle incluent la classe de l'objet ainsi que ses coordonnées et dimensions normalisées sur l'image, représentées visuellement par une zone bleutée, comme illustré dans la Figure 26. Cette zone correspond à une boîte englobante (bounding box) définie autour de la région d'intérêt, ici les yeux. Plus précisément, les coordonnées **X** et **Y** de cette boîte représentent la position centrale de l'objet à détecter, tandis

que la largeur et la hauteur indiquent ses dimensions relatives à la taille totale de l'image. Par exemple, une instance de la classe 0 possède une coordonnée **X** de 0,517, une coordonnée **Y** de 0,444, une largeur de 0,728 et une hauteur de 0,503. Ces valeurs, exprimées sous forme normalisée, permettent de standardiser les données et sont cruciales pour guider le modèle dans l'apprentissage de la localisation et de la détection des objets. La zone bleutée facilite ainsi la visualisation des régions ciblées pour entraîner le modèle à reconnaître et classifier les objets avec précision.

## 3.4 Configuration de l'architecture de YOLOV4

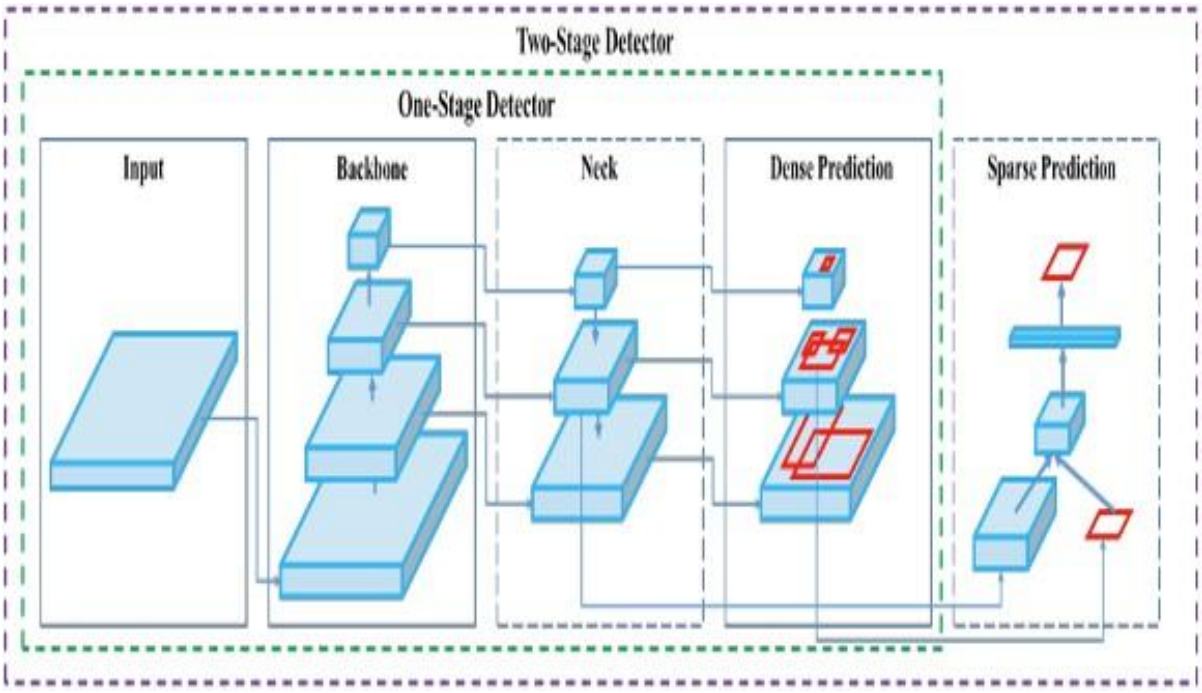


*Figure 27 Rappel de l'architecture de YOLOV4 [14]*

Le framework open source Darknet a été utilisé pour entraîner le détecteur, en le clonant depuis le dépôt officiel de Darknet sur le GitHub d'AlexeyAB. Afin d'adapter le modèle aux besoins spécifiques, plusieurs modifications ont été apportées au fichier de configuration. Tout d'abord, la ligne dédiée à l'entraînement a été décommentée tandis que la ligne de test liée au batch subdivision a été commentée. Le paramètre batch a ensuite été défini à **batch=64** pour optimiser le traitement des lots d'images, et le paramètre subdivisions ajusté à **subdivisions=16** pour mieux gérer les ressources durant l'entraînement. La taille du réseau d'entrée a été configurée avec **width = 416** et **height=416**, ou toute autre valeur multiple de 32, afin d'assurer la compatibilité avec YOLOV4. De plus, la ligne **max_batches** a été paramétrée en suivant la

formule (nombre_de_classes x 2000), tout en garantissant qu'elle ne soit pas inférieure à 6000 et supérieure au nombre d'images d'entraînement ; dans ce cas, elle a été fixée à **6000**. Enfin, les lignes **filters = 32** situées après les trois dernières couches [yolo] ont été modifiées selon la formule (nombre_de_classes+5) ×3, ce qui a donné une valeur de **filters=21**. Ces ajustements ont permis d'assurer une configuration optimale pour l'entraînement du modèle en fonction des spécificités des données et des classes à détecter.

## 3.5 Entraînement du Modèle

L'entraînement du modèle constitue une étape centrale et stratégique dans ce projet, car il détermine directement la capacité du détecteur d'objets à reconnaître et à classer les éléments visés avec précision. Ce processus repose sur l'utilisation de données annotées qui permettent au modèle YOLOV4 d'apprendre à différencier les caractéristiques pertinentes des objets à détecter.

L'objectif principal est d'obtenir un modèle robuste, capable de générer des prédictions fiables dans divers contextes, y compris ceux où les conditions d'éclairage, les angles de vue ou les caractéristiques des images peuvent varier. Pour y parvenir, il est crucial de calibrer les paramètres d'entraînement de manière optimale, notamment en ajustant des éléments tels que le taux d'apprentissage, le nombre d'itérations, et les techniques de régularisation utilisées pour éviter le surapprentissage.

Cette section détaille les étapes essentielles du processus d'entraînement, en commençant par la préparation des données, le choix des hyperparamètres, et l'utilisation des algorithmes d'optimisation, jusqu'à l'évaluation intermédiaire des performances du modèle. Chaque décision prise lors de cette phase vise à maximiser la précision du détecteur, tout en maintenant un temps d'inférence rapide, condition indispensable pour les applications en temps réel.

### 3.5.1 Configuration du Framework Darknet

Avant de lancer l'entraînement, plusieurs configurations ont été nécessaires pour maximiser l'utilisation des ressources matérielles, notamment le GPU. Le paramètre **opencv=1** a été activé, permettant l'utilisation de la bibliothèque OpenCV, qui offre des fonctionnalités avancées pour le traitement d'images et de vidéos. Cela inclut le redimensionnement d'images, la conversion de formats, et divers traitements préliminaires sur les données d'entrée, de

manière optimisée. Ensuite, le réglage **gpu=1** a activé l'utilisation du GPU pour effectuer les calculs. Contrairement au CPU (*Central Processing Unit*), le GPU est capable de gérer un grand nombre de calculs en parallèle, ce qui est idéal pour les modèles d'apprentissage profond exigeants. Ce paramètre améliore considérablement les performances d'entraînement et d'inférence, notamment pour les algorithmes traitant des matrices de grande taille. Enfin, le paramètre **cudnn=1** a permis d'exploiter cuDNN (*CUDA Deep Neural Network*), une bibliothèque GPU développée par NVIDIA (*GPU hardware manufacturer*) et spécifiquement conçue pour accélérer les opérations des réseaux de neurones profonds, telles que la convolution et les couches de pooling. L'activation de cuDNN a encore renforcé les performances du GPU, réduisant le temps d'entraînement et améliorant la vitesse d'exécution lors de l'inférence. Ces trois réglages combinés ont permis d'exploiter pleinement les capacités matérielles pour accélérer les processus de traitement d'images et d'apprentissage profond, rendant ainsi l'ensemble du pipeline beaucoup plus efficace.

### 3.5.2 Préparation des fichiers d'entraînement

Dans le cadre de l'entraînement, nous avons créé deux fichiers essentiels pour structurer les données. Le fichier **classe.names** permet de définir les différentes classes d'objets à détecter(strabisme ou normal), facilitant ainsi leur identification par le modèle. Le fichier **labelled.data** contient des informations détaillées sur la configuration des données, telles que les chemins d'accès aux images, le nombre de classes, ainsi que la répartition des images entre les ensembles d'entraînement et de tests. Cette structuration méthodique des fichiers est indispensable pour assurer le bon déroulement et la fiabilité du processus d'entraînement.

### 3.5.3 Division des Données

La base de données a été divisée en deux ensembles : un **ensemble d'entraînement de 325 images (83,76%)**, destiné à l'apprentissage du modèle, et un **ensemble de tests de 63 images (16,24 %)** utilisés pour évaluer ses performances sur des données inédites. Cette division est essentielle pour vérifier que le modèle n'est pas sujet au surapprentissage et qu'il est capable de généraliser efficacement à de nouvelles images. Pour automatiser la création des fichiers nécessaires à cette répartition, deux scripts Python ont été développés, garantissant ainsi une préparation méthodique et reproductible des données.

### 3.5.4 Lancement de l'entraînement

Une fois les données préparées et les fichiers nécessaires créés, le modèle est prêt pour l'entraînement. Cette phase repose sur des hyperparamètres soigneusement définis, qui jouent un rôle clé dans l'optimisation des performances du modèle. **Le Batch size=64** est paramètre qui indique le nombre d'exemples de données traités par le modèle à chaque itération avant la mise à jour des poids. Un batch size de **64** signifie que 64 images seront utilisées simultanément pour calculer les gradients. Ce choix équilibre la consommation de mémoire et la vitesse d'entraînement : des lots plus petits consomment moins de mémoire, mais peuvent rendre l'apprentissage moins stable, tandis que des lots plus grands nécessitent plus de ressources, mais permettent des calculs plus précis des gradients. **Le *Learning Rate*** est un hyperparamètre crucial qui contrôle la taille des étapes lors de la mise à jour des poids du modèle au cours de l'apprentissage. Un taux d'apprentissage bien ajusté permet au modèle de converger efficacement vers le minimum global de la fonction de coût. S'il est trop élevé, le modèle risque de diverger ou d'osciller autour de la solution optimale. À l'inverse, s'il est trop faible, la convergence sera trop lente. Dans ce contexte, le learning rate a été ajusté dynamiquement pour optimiser la vitesse et la stabilité de la convergence. Ces paramètres sont fondamentaux pour garantir un entraînement efficace, équilibrant la précision, la rapidité et la consommation des ressources. Leur ajustement contribue directement à la qualité des résultats obtenus par le modèle. L'entraînement s'effectue en plusieurs itérations où le modèle ajuste ses poids pour minimiser la fonction de perte **CIoU** , qui évalue la précision des boîtes englobantes. À chaque itération, les performances du modèle sont évaluées sur l'ensemble de tests, permettant des ajustements d'hyperparamètres si nécessaire.

### 3.5.5 Amélioration des Performances et Fin de l'Entraînement

Les performances du modèle s'améliorent au fur et à mesure de l'entraînement. Le processus se poursuit jusqu'à ce que la fonction de perte atteigne un seuil minimal ou que le nombre maximal d'itérations (**max_batches**) soit atteint. Ainsi, ce processus rigoureux garantit que notre modèle YOLOV4 est performant, capable de détecter et de classifier les objets rapidement et avec précision, prêt pour une utilisation en conditions réelles. L'exploration de l'architecture de YOLOV4 et la mise en place de la méthodologie d'entraînement ont permis de construire un modèle de détection du strabisme optimisé. Après avoir détaillé les différentes étapes de

préparation des données, de configuration et d'apprentissage du réseau neuronal, il est essentiel d'évaluer la performance du modèle obtenu.

Le chapitre suivant est consacré à l'analyse des résultats de l'entraînement et aux tests effectués sur divers échantillons d'images. Nous y examinerons les performances du modèle en termes de précision, de taux de détection et d'erreurs éventuelles, en nous appuyant sur des indicateurs clés tels que le pourcentage de réussite et les graphiques d'évaluation. Cette analyse nous permettra de discuter des forces et des limites de notre approche, ouvrant ainsi la voie à d'éventuelles améliorations.

# CHAPITRE 4. RÉSULTATS ET DISCUSSION

Dans cette partie nous présenterons les résultats de l'entraînement de notre modèle mettant en exergue ses performances à travers la courbe d'entraînement ainsi que des prédictions effectuées par notre modèle sur quelques échantillons d'images en guise de test.

## 4.1 Analyse des Résultats d'Entraînement de YOLO

L'entraînement du modèle a été lancé sur Google Colab en utilisant la version professionnelle, ce qui a permis de bénéficier d'un GPU V100 performant doté d'une mémoire RAM de 16 Go, optimisant ainsi les performances et la rapidité du processus. Avec une configuration d'entraînement fixée à 6000 itérations, comme mentionné précédemment, l'entraînement a généré trois fichiers distincts, représentant les étapes clés du processus d'apprentissage. Le fichier **yolov4-tiny_custom_6000.weights** contient les poids enregistrés après 6000 itérations, correspondant à un modèle ayant appris un ensemble initial de caractéristiques, bien qu'encore en phase d'optimisation. Le fichier **yolov4-tiny_custom_last.weights** capture les poids du réseau dans les dernières itérations, reflétant un modèle ayant atteint un niveau élevé d'optimisation, mais pas encore complètement finalisé. Enfin, le fichier **yolov4tiny_custom_final.weights** représente les poids finaux obtenus à la fin de l'entraînement, symbolisant un réseau entièrement optimisé et prêt à être utilisé pour la tâche spécifique. Cette méthodologie, combinée aux ressources matérielles performantes, garantit une précision et une efficacité optimales du modèle. L'entraînement a duré au total 1 heure et 10 minutes, pendant lesquelles le modèle a ajusté ses poids au fil des itérations. L'erreur de classification des différentes classes est class_loss = 0.023 et l'erreur de prédiction des boîtes englobantes obtenue est iou_loss = 0.602 ce qui donne une erreur moyenne de AVG loss de 0,059. L'évolution de l'entraînement et la précision du modèle sont présentées au Tableau 2 qui est le seul graphe de performance généré par YOLO après l'entraînement pour évaluer le modèle.

*Tableau 2 Résultats d'entraînement*

| Iterations | Total_bbox | IOU_Avg | Count | Class_Loss | IOU_Loss |
|---|---|---|---|---|---|
| 6000 | 594590 | 0.849 | 11 | 0.035 | 0.628 |
| 6000 | 594603 | 0.789 | 2 | 0.046 | 0.451 |
| 6000 | 594613 | 0.848 | 13 | 0.076 | 1.596 |
| 6000 | 594624 | 0.869 | 10 | 0.004 | 1.139 |
| 6000 | 594624 | 0.858 | 11 | 0.003 | 0.476 |
| 6000 | 594632 | 0.840 | 7 | 0.043 | 0.914 |

| | | | | | |
|---|---|---|---|---|---|
| 6000 | 594632 | 0.818 | 1 | 0.023 | 0.602 |

Lors de l'entraînement du modèle **YOLO**, plusieurs métriques clés sont utilisées pour évaluer ses performances et identifier les axes d'amélioration. Ces résultats, présentés dans le tableau, offrent un aperçu précis de l'évolution du modèle au fil des itérations et permettent de comprendre ses points forts ainsi que les domaines nécessitant des ajustements. L'une des premières métriques importantes concerne la version du modèle utilisé, en l'occurrence **YOLO v3**. Cette version intègre des optimisations significatives par rapport aux précédentes, notamment une gestion améliorée des images et des objets à détecter, ce qui se traduit par des performances globales accrues, tant en précision qu'en efficacité. Le **Total_bbox** représente le nombre total de boîtes englobantes analysées par le modèle. Ces boîtes, générées autour des objets détectés dans les images, augmentent progressivement à mesure que l'entraînement avance. Cette augmentation reflète une exposition croissante du modèle à des exemples d'objets, favorisant ainsi une meilleure généralisation des prédictions sur de nouvelles données. L'**IOU_Avg** (*Intersection Over Union Average*) est une métrique essentielle pour évaluer la précision des boîtes englobantes. Elle compare les boîtes prédites par le modèle aux véritables boîtes réelles (*ground truth*). Une valeur proche de 1 indique des prédictions très précises. Dans les résultats obtenus, les valeurs moyennes de l'IOU varient entre **0,789** et **0,885**, ce qui témoigne de performances globalement satisfaisantes, bien qu'il reste une marge d'amélioration.

La **Class_Loss**, ou perte de classification, mesure la capacité du modèle à attribuer correctement une classe aux objets détectés. Elle est calculée en évaluant l'écart entre la classe prédite et la classe réelle. Dans les résultats, cette perte reste faible, oscillante entre **0,004** et **0,076**, ce qui montre une bonne précision dans l'attribution des classes aux objets.

Le **IOU_Loss**, ou perte liée à l'IOU, mesure l'écart entre les boîtes englobantes prédites et celles réelles. Des valeurs élevées de cette métrique, comme celles observées (entre **0,069** et **1,596**), indiquent que certaines boîtes nécessitent des ajustements pour mieux correspondre aux objets détectés. Cette métrique reste essentielle pour affiner les prédictions spatiales.

Enfin, le **Count** représente le nombre d'objets détectés par le modèle dans chaque image à une étape donnée de l'entraînement. Ce nombre varie entre **1** et **13** objets dans les résultats obtenus, illustrant la capacité du modèle à identifier simultanément plusieurs objets, même dans des

images complexes. Les différentes lignes du tableau mettent en évidence les performances du modèle à des étapes spécifiques de l'entraînement. Par exemple, sur la première ligne, où **Total_bbox = 594590**, le modèle atteint une précision élevée avec une **IOU_Avg** de **0,849** et une **Class_Loss** relativement faible de **0,035**, indiquant des classifications précises et des boîtes bien ajustées (**IOU_Loss = 0,628**). En revanche, sur la ligne où **Total_bbox = 594613**, bien que la **IOU_Avg** reste stable à **0,848**, l'**IOU_Loss** atteint **1,59**, signalant des difficultés dans l'ajustement des boîtes englobantes à cette étape. Enfin, sur la dernière ligne, avec **Total_bbox = 594632**, on observe une nette amélioration globale, avec une **Class_Loss** réduite à **0,023** et une **IOU_Loss** diminuée à **0,602**, ce qui suggère que le modèle a atteint une optimisation notable à la fin de l'entraînement. Ces exemples démontrent comment les performances du modèle évoluent tout au long de l'entraînement. Les ajustements constants réalisés sur les paramètres du modèle, notamment les poids des connexions, permettent d'améliorer progressivement la précision des prédictions et la capacité du modèle à généraliser efficacement sur des données nouvelles. La Figure 28 illustre l'évolution de la perte moyenne (**AVG Loss**) au cours de l'entraînement du modèle YOLO. Comme on peut l'observer, la courbe commence avec une perte élevée, qui diminue rapidement lors des premières itérations. Cette baisse reflète l'ajustement initial des poids du modèle pour s'aligner avec les données d'entraînement. À mesure que l'entraînement progresse, la courbe se stabilise et converge vers une valeur basse, autour de **0,059** à l'**itération 6000**, indiquant que le modèle a atteint un bon niveau d'optimisation. La Figure 29 confirme la stabilité du processus d'entraînement, avec une convergence progressive vers une erreur minimale, ce qui témoigne de la capacité du modèle à s'améliorer au fil des itérations. Elle joue un rôle clé dans l'évaluation des performances, en permettant de vérifier que l'entraînement est efficace et que le modèle n'est pas en surapprentissage. En complément du tableau précédent, cette visualisation apporte une validation graphique des résultats obtenus et met en évidence l'importance des ajustements réalisés pour atteindre une perte minimale.

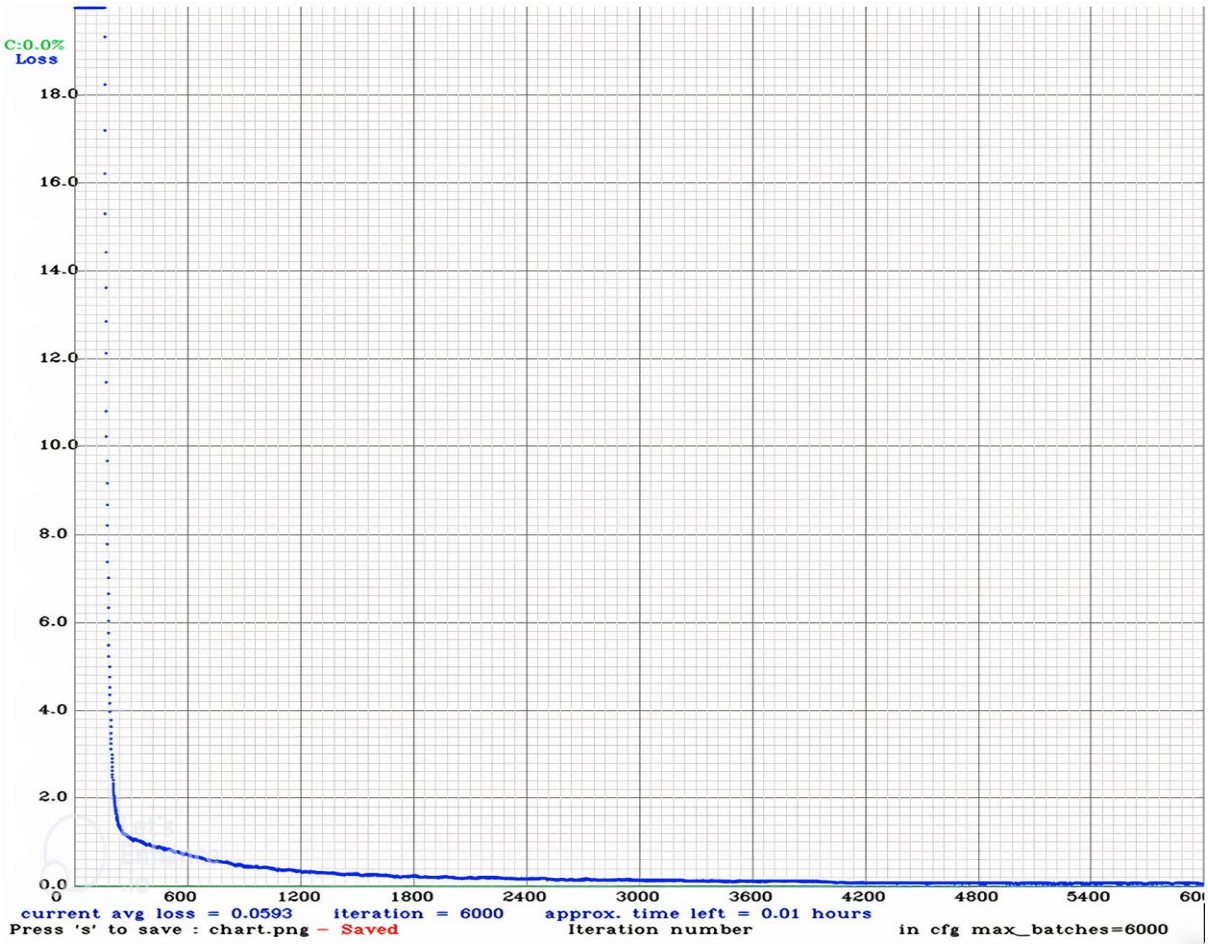


*Figure 28. Graphe de performance obtenu après entraînement*

## 4.2 Test du modèle obtenu

Nous avons évalué notre modèle sur 63 images représentatives récupérées via une recherche Google Images, afin d'illustrer concrètement son fonctionnement et d'en mettre en évidence les limites éventuelles. Les cas des strabismes montrent que les probabilités sont bonnes et pour ces échantillons, elles varient entre 0,95 et 1. Le modèle a correctement détecté les yeux et assigné des scores de confiance élevés, allant de **0,94** à **0,99**. Ces scores témoignent de la fiabilité du système à différencier les cas normaux des cas anormaux avec une grande précision. Les Figures 29 et 30 illustrent non seulement la robustesse du modèle dans la détection des cas normaux, mais également sa capacité à fournir des prédictions cohérentes et précises sur des données tests, validant ainsi l'efficacité de l'approche d'entraînement utilisée

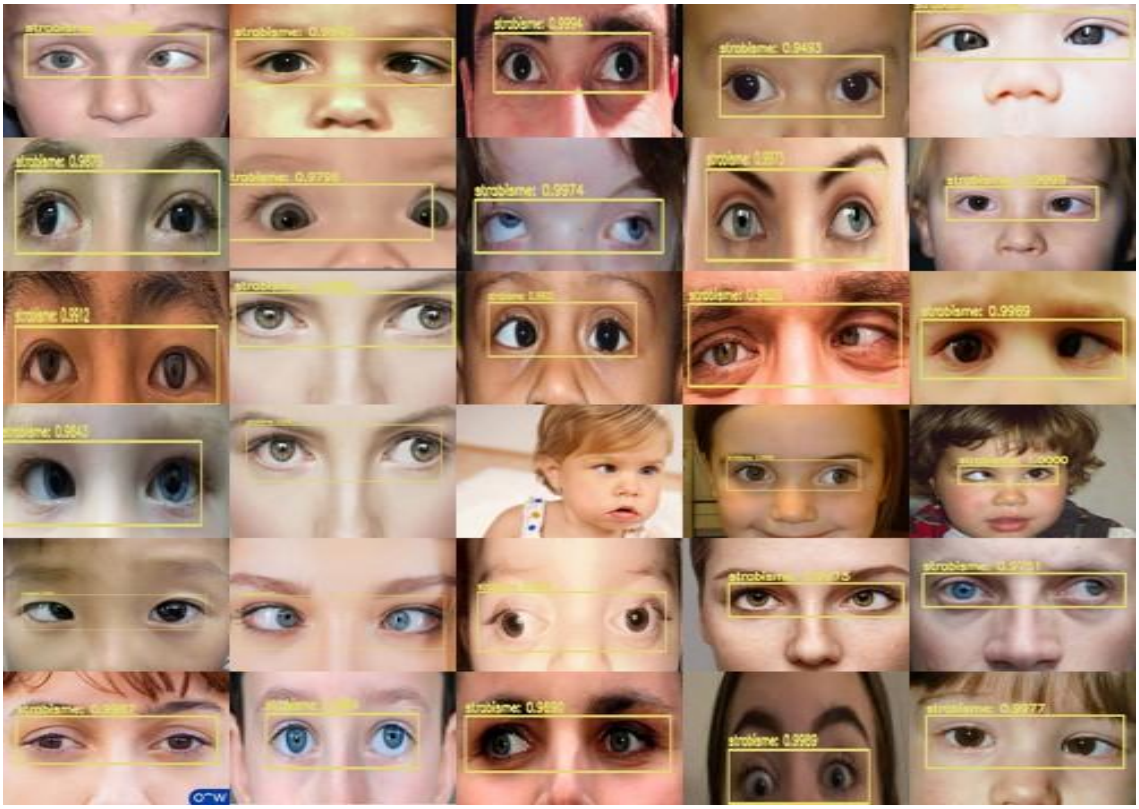

*Figure 29. Images illustrant la détection du strabisme par notre modèle*

*Figure 30. Images illustrant les cas normaux détectés par notre système*

Pendant les tests nous avons constaté certaines imperfections comme le fait que le système ne détecte pas certains cas ou se trompe sur d'autres ce qui montre que le système a présenté quelques défauts pouvant être améliorés. La Figure 31 met en évidence certains cas d'imperfections relevés lors des tests effectués sur le modèle. Ces images illustrent des situations où le système a présenté des difficultés, telles que l'incapacité à détecter certains cas ou des erreurs dans la classification des images. Ces imperfections soulignent les limites actuelles du modèle et mettent en évidence des axes d'amélioration potentiels. Le système classe un cas normal ou faiblement dévié comme étant un cas de strabisme ou inversement. De telles erreurs peuvent être liées à la qualité des données d'entraînement, à des variations dans les caractéristiques visuelles des images (lumière, positionnement, etc.), ou à des limites inhérentes à l'architecture utilisée.


*Figure 31 Images illustrant le défaut de détections sur certaines images*

Ces observations soulignent que, bien que le modèle affiche des performances satisfaisantes dans la majorité des cas, il reste sensible à certaines variations contextuelles ou à des cas ambigus. Cela indique qu'un renforcement des données d'entraînement, ainsi que des ajustements supplémentaires des hyperparamètres ou des techniques de prétraitement, pourrait améliorer sa robustesse et réduire les erreurs de classification.

### 4.2.1 Analyse des Performances du Modèle de Détection du Strabisme

Le modèle YOLOV4 a été testé sur un ensemble de données d'images, y compris des cas normaux, des cas de strabisme, et des erreurs dans la détection. Ce test permet d'évaluer la performance du modèle dans des conditions réelles. Grâce aux données de tests (les nouvelles images analysées par le modèle), on va mesurer le taux de réussite du modèle. L'objectif est de

déterminer si le modèle réussit à détecter correctement le strabisme et les cas normaux, et d'analyser ses erreurs.

#### 4.2.1.1 Résultats des Tests

Le tableau 3 résume les performances d'un modèle de classification appliqué à un ensemble de données composé de 63 cas, répartis en deux catégories principales : les cas normaux et les cas de strabisme. Chaque ligne du tableau représente un cas spécifique, avec des informations détaillées sur la classe réelle de l'exemple, la classe prédite par le modèle, la probabilité associée à la prédiction et un commentaire sur la qualité de la classification

*Tableau 3 récapitulatif des performances*

| Cas | Classe Réelle | Classe Prédite | Probabilité | Commentaire |
|---|---|---|---|---|
| 1 | Normal | Normal | 0.9609 | Correctement classifié |
| 2 | Normal | Normal | 0.9736 | Correctement classifié |
| 3 | Normal | Normal | 0.9921 | Correctement classifié |
| 4 | Normal | Normal | 0.9793 | Correctement classifié |
| 5 | Normal | Normal | 0.9927 | Correctement classifié |
| 6 | Normal | Normal | 0.9986 | Correctement classifié |
| 7 | Normal | Normal | 0.9228 | Correctement classifié |
| 8 | Normal | Normal | 0.9988 | Correctement classifié |
| 9 | Normal | Normal | 0.9990 | Correctement classifié |
| 10 | Normal | Normal | 0.9978 | Correctement classifié |
| 11 | Normal | Normal | 0.9862 | Correctement classifié |
| 12 | Normal | Normal | 0.9671 | Correctement classifié |
| 13 | Normal | Normal | 0.9949 | Correctement classifié |
| 14 | Normal | Normal | 0.9977 | Correctement classifié |
| 15 | Normal | Normal | 0.9981 | Correctement classifié |
| 16 | Normal | Normal | 0.9999 | Correctement classifié |
| 17 | Normal | Normal | 0.9976 | Correctement classifié |
| 18 | Normal | Normal | 0.9943 | Correctement classifié |
| 19 | Normal | Normal | 0.9486 | Correctement classifié |
| 20 | Normal | Normal | 0.9992 | Correctement classifié |
| 21 | Normal | Normal | 0.9922 | Correctement classifié |
| 22 | Normal | Normal | 0.9977 | Correctement classifié |
| 23 | Normal | Normal | 0.9994 | Correctement classifié |
| 24 | Normal | Normal | 0.9921 | Correctement classifié |
| 25 | Normal | Normal | 0.9913 | Correctement classifié |
| 26 | Normal | Normal | 0.9965 | Correctement classifié |
| 27 | Normal | Normal | 0.9993 | Correctement classifié |
| 28 | Normal | Normal | 0.9974 | Correctement classifié |
| 29 | Normal | Normal | 0.9936 | Correctement classifié |
| 30 | Normal | Normal | 0.9744 | Correctement classifié |
| 31 | Normal | Strabisme | 0.5210 | Normal mal classifié |
| 32 | Normal | Strabisme | 0.9834 | Normal mal classifié |
| 33 | Strabisme | Strabisme | 0.9908 | Correctement classifié |

| | | | | |
|---|---|---|---|---|
| 34 | Strabisme | Strabisme | 0.9945 | Correctement classifié |
| 35 | Strabisme | Strabisme | 0.9994 | Correctement classifié |
| 36 | Strabisme | Strabisme | 0.9493 | Correctement classifié |
| 37 | Strabisme | Strabisme | 0.9640 | Correctement classifié |
| 38 | Strabisme | Strabisme | 0.9879 | Correctement classifié |
| 39 | Strabisme | Strabisme | 0.9796 | Correctement classifié |
| 40 | Strabisme | Strabisme | 0.9974 | Correctement classifié |
| 41 | Strabisme | Strabisme | 0.9973 | Correctement classifié |
| 42 | Strabisme | Strabisme | 0.9999 | Correctement classifié |
| 43 | Strabisme | Strabisme | 0.9912 | Correctement classifié |
| 44 | Strabisme | Strabisme | 0.9882 | Correctement classifié |
| 45 | Strabisme | Strabisme | 0.9900 | Correctement classifié |
| 46 | Strabisme | Strabisme | 0.9628 | Correctement classifié |
| 47 | Strabisme | Strabisme | 0.9969 | Correctement classifié |
| 48 | Strabisme | Strabisme | 0.9843 | Correctement classifié |
| 49 | Strabisme | Strabisme | 0.9987 | Correctement classifié |
| 50 | Strabisme | Strabisme | 0.9995 | Correctement classifié |
| 51 | Strabisme | Strabisme | 0.9998 | Correctement classifié |
| 52 | Strabisme | Strabisme | 1.0000 | Correctement classifié |
| 53 | Strabisme | Strabisme | 0.9820 | Correctement classifié |
| 54 | Strabisme | Strabisme | 0.9977 | Correctement classifié |
| 55 | Strabisme | Strabisme | 0.9906 | Correctement classifié |
| 56 | Strabisme | Strabisme | 0.9975 | Correctement classifié |
| 57 | Strabisme | Strabisme | 0.9751 | Correctement classifié |
| 58 | Strabisme | Strabisme | 0.9987 | Correctement classifié |
| 59 | Strabisme | Strabisme | 0.9984 | Correctement classifié |
| 60 | Strabisme | Strabisme | 0.9690 | Correctement classifié |
| 61 | Strabisme | Strabisme | 0.9989 | Correctement classifié |
| 62 | Strabisme | Strabisme | 0.9977 | Correctement classifié |
| 63 | Strabisme | Normal | 0.6142 | Strabisme mal classifié |

Sur les 63 cas analysés, 30 sont des cas normaux et 30 sont des cas de strabisme, tandis que 3 cas représentent des erreurs de classification. Pour les cas normaux, la plupart des prédictions sont correctes avec des probabilités élevées (généralement supérieures à 0,9), ce qui témoigne de la fiabilité du modèle dans cette catégorie. Cependant, deux cas normaux (les cas 31 et 32) ont été mal classifiés comme strabisme, avec des probabilités respectives de 0.5210 et 0.9834. Ces erreurs montrent que le modèle peut être induit en erreur par des caractéristiques ambiguës ou atypiques dans certains cas normaux. De même, pour les cas de strabisme, la performance globale est également très élevée, avec des probabilités de classification correctes proches de 1 dans la majorité des exemples. Toutefois, une erreur significative a été relevée pour le cas 63, où un strabisme a été mal classifié comme normal, avec une probabilité relativement faible de 0.6142. Cette erreur souligne un défi potentiel dans la différenciation des cas limites où les

caractéristiques du strabisme pourraient ne pas être suffisamment prononcées. Le tableau met également en évidence des commentaires spécifiques pour chaque cas, permettant de distinguer les prédictions correctes des erreurs. Les expressions comme Correctement classifié indiquent que le modèle a attribué une probabilité élevée à la classe réelle, tandis que des mentions comme Normal mal classifié ou Strabisme mal classifié soulignent les erreurs de prédiction, accompagnées des probabilités correspondantes. Ces informations permettent une analyse approfondie des performances du modèle et offrent des pistes pour son amélioration

#### 4.2.1.2 Graphique des performances

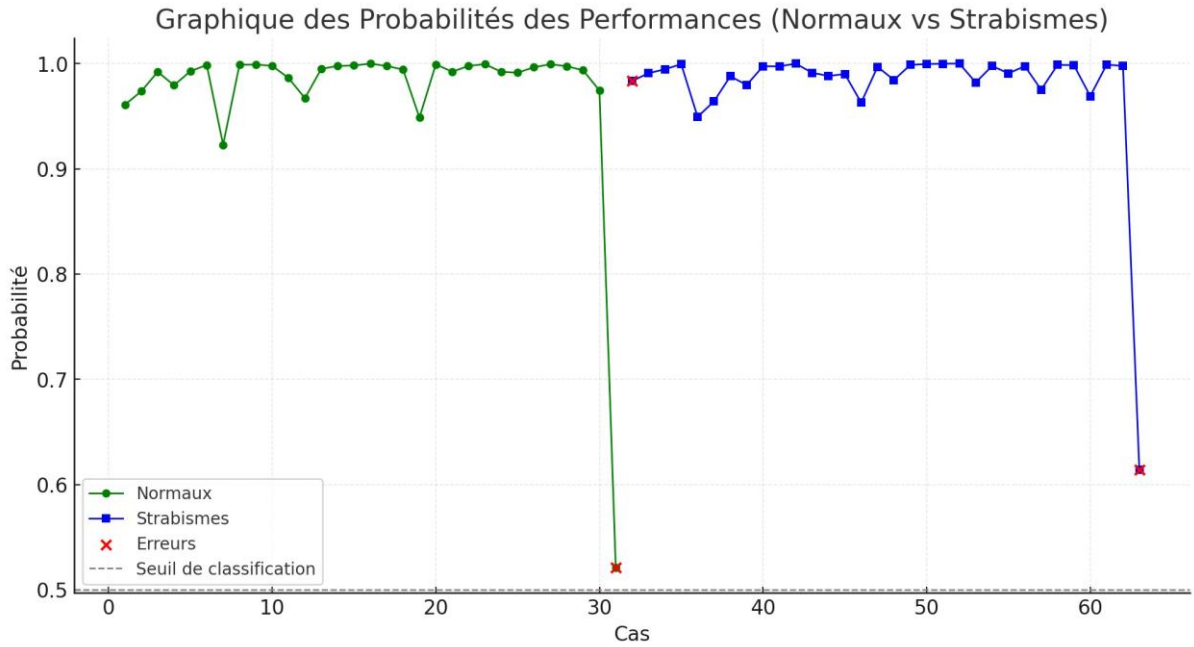


*Figure 32 Graphe récapitulatif des cas normaux, de strabismes et des erreurs*

La Figure 32 illustre les probabilités de classification des cas normaux et des cas de strabisme par le modèle, en distinguant visuellement les deux classes. Les cas normaux sont représentés par une courbe verte, tandis que les cas de strabisme sont représentés par une courbe bleue. Les points rouges indiquent les erreurs de classification. On observe que la majorité des probabilités sont proches de 1, ce qui reflète une performance globale élevée du modèle dans la reconnaissance des deux classes. Cependant, trois erreurs de classification sont identifiées : une pour un cas de strabisme mal classifié comme normal (à une probabilité de 0.6142) et deux pour des cas normaux mal classifiés comme strabisme (avec des probabilités de 0.5210 et 0.9834). Le seuil de classification, indiqué par une ligne grise pointillée à 0.5, sert de référence

pour distinguer les décisions du modèle. Ce graphique met en évidence la robustesse du modèle pour la plupart des cas tout en suggérant des améliorations pour mieux gérer les exemples ambigus ou atypiques.

#### 4.2.1.3 Calcul du pourcentage de réussite

Le calcul du pourcentage de réussite repose sur l'évaluation des performances du modèle à travers un ensemble de 63 cas. Ces cas incluent 30 exemples de la classe « Normaux » et 30 exemples de la classe « Strabismes », ainsi que 3 cas d'erreurs de classification identifiées. Chaque cas a été évalué sur la base de la probabilité attribuée par le modèle à la classe prédite, permettant de déterminer si la classification était correcte ou non. Ces calculs intègrent également des métriques clés comme l'exactitude, la précision, le rappel et le F1-score (moyenne harmonique entre la précision et le rappel). Pour débuter, l'exactitude globale du modèle est calculée comme la proportion des classifications correctes par rapport au nombre total de cas analysés. Le modèle a correctement classifié 30 cas normaux et 30 cas de strabisme, soit un total de 60 prédictions correctes sur 63 cas. L'exactitude est donc calculée comme suit :

$$Exactitude = \frac{\text{-./0"\#}}{\text{-./0"\# \%.\%43 \$\# 245}} \text{./0"\#} \quad \$\#classifications \quad 2.""\#2\%\#5\ x\ 100 \quad (8)$$

$$Exactitude = \frac{60}{63} x\ 100 \approx 95{,}2 \quad \% \quad (9)$$

Ensuite, la précision pour chaque classe est évaluée. La précision de la classe « Normaux » est obtenue en calculant le rapport entre les vrais positifs (cas normaux correctement classifiés) et l'ensemble des prédictions positives pour cette classe. Sur les 32 prédictions positives effectuées par le modèle pour la classe « Normaux », deux cas étaient incorrects (cas 31 et 32). Ainsi, la précision pour cette classe est d'environ 93.75%.

$$\textbf{Précision} \quad (\textbf{Cas}) = \frac{\textbf{Vrais positifs}}{(\textbf{Vrais positifs E Faux positifs})} \text{ x } 100 \quad (10)$$

$$\textbf{Précision} \quad (\textbf{Normaux}) = \text{—} x\ 100 \approx 93{,}75^{30} \quad \% \quad (11)$$

32

Pour la classe « Strabismes », une seule erreur a été relevée (cas 63), sur un total de 31 prédictions positives. Par conséquent, la précision pour cette classe est d'environ 96.77.

$$\boldsymbol{Précision}\ (\boldsymbol{Strabismes}) = \frac{30}{31} x\ 100 \approx 96{,}77\ \% \quad (12)$$

Le rappel évalue la capacité du modèle à identifier correctement les cas pertinents dans chaque classe. Pour les classe « Normaux » et « Strabismes », le rappel est égal à :

$$\text{Rappel (Normaux)} = \frac{\text{NOPQR}}{\text{STRQUQVR}} \text{ x } 100 \quad (13)$$

(NOPQR STRQUQVR E WPXY Zé\PUQVR)

$$\boldsymbol{Rappel}\ (\boldsymbol{Normaux}) = \frac{30}{30+0} x\ 100 \approx 100\% \quad (14)$$

Enfin, le F1-score (moyenne harmonique entre la précision et le rappel), qui combine précision et rappel, est calculé pour fournir une vue d'ensemble équilibrée des performances. Le F1score pour chaque classe est donné par la formule :

$$\boldsymbol{F1-score} = 2 x \frac{\boldsymbol{Précision}\ x\ \boldsymbol{Rappel}}{(\boldsymbol{Précision}\ \text{E}\ \boldsymbol{Rappel})} \quad (15)$$

Pour les cas normaux, le F1-score est d'environ :

$$\boldsymbol{F1-score} = 2 x \frac{\text{h:,jk} \times \text{m99}}{(\text{h:,jk E m99})} \approx 96{,}77\ \% \quad (16)$$

Pour les cas de strabisme, il est de :

$$\boldsymbol{F1-score} = 2 x \frac{\text{h8,jj} \times \text{m99}}{(\text{h8,jj E m99})} \approx 98{,}36\ \% \quad (17)$$

Ces calculs montrent que le modèle offre des performances équilibrées pour les deux classes, avec une exactitude globale de 95,2 % et des scores F1 élevés pour les deux catégories. Ces résultats soulignent la robustesse du modèle pour la classification des cas normaux et de

strabisme tout en mettant en évidence quelques marges d'amélioration pour gérer les cas atypiques ou limites.

#### 4.2.1.4 Analyse du pourcentage de réussite

L'analyse du pourcentage de réussite obtenu par le modèle YOLOV4 met en lumière une performance globale satisfaisante tout en soulignant certains points faibles qui méritent une attention particulière. Avec un taux de réussite global de 95,24 %, le modèle démontre une capacité remarquable à différencier les cas normaux des cas de strabisme. Ce résultat reflète la qualité de l'entraînement sur les données disponibles et l'efficacité générale de l'algorithme dans des scénarios standards. Cependant, les erreurs identifiées, bien que peu nombreuses, révèlent des limites spécifiques dans la reconnaissance de certains cas.
Les trois erreurs de classification observées dans les 63 cas analysés se répartissent de manière inégale entre les deux catégories principales. Deux cas normaux (cas 31 et 32) ont été mal classifiés comme strabisme, avec des probabilités respectives de 0.5210 et 0.9834. Ces valeurs indiquent que le modèle était relativement incertain dans ces cas, probablement en raison de caractéristiques ambiguës ou partagées entre les deux classes. De manière similaire, un cas de strabisme (cas 63) a été mal classifié comme normal, avec une probabilité de 0.6142. Cette erreur suggère une faiblesse dans la détection des strabismes aux caractéristiques moins marquées ou atypiques.

Ce pourcentage de réussite reflète que le modèle a été bien entraîné sur des cas courants, notamment des yeux normaux et des formes fréquentes de strabisme. Cependant, des variations spécifiques, comme les strabismes convergents et divergents, semblent moins bien détectées, ce qui pourrait être dû à un manque de diversité dans les données d'entraînement ou à une sousreprésentation de ces cas spécifiques dans le dataset. Cette observation souligne un besoin d'amélioration dans la gestion des cas atypiques et limites.

Pour améliorer ces performances, plusieurs pistes peuvent être explorées. Premièrement, l'augmentation du volume des données d'entraînement, en incluant davantage de cas de strabisme convergent et divergent, pourrait améliorer la capacité du modèle à reconnaître ces configurations spécifiques. Des techniques d'augmentation des données, comme la rotation des images, la mise à l'échelle ou des ajustements de luminosité, pourraient également contribuer à diversifier les exemples présentés au modèle, renforçant ainsi sa robustesse. Deuxièmement,

un ajustement des hyperparamètres, tels que le taux d'apprentissage ou la taille des lots, pourrait optimiser davantage la capacité de généralisation du modèle. Enfin, l'exploration d'architectures plus récentes, comme YOLOV5 ou YOLOV8, pourrait offrir des performances supérieures en termes de précision et de robustesse dans les détections.

En conclusion, avec un taux de réussite global de 95,24 %, le modèle affiche des performances respectables dans la classification des cas normaux et des cas de strabisme. Toutefois, les erreurs identifiées montrent qu'il reste des marges d'amélioration, notamment dans les scénarios complexes ou atypiques. Des efforts ciblés sur l'enrichissement des données et l'ajustement des paramètres d'entraînement pourraient permettre d'améliorer significativement les performances, rendant le modèle encore plus fiable et performant dans des conditions variées. L'analyse des résultats obtenus a permis d'évaluer les performances de notre modèle de détection du strabisme basé sur YOLOV4. Les tests menés ont mis en évidence ses atouts en termes de précision et de rapidité, tout en soulignant certaines limites, notamment en ce qui concerne les erreurs de classification ou les cas particuliers de détection. Dans le chapitre suivant, nous allons approfondir ces observations en discutant des implications cliniques potentielles de notre approche. Nous proposerons également des pistes d'amélioration afin d'optimiser davantage la robustesse et la fiabilité du modèle. Enfin, nous conclurons notre étude en récapitulant les principales contributions de ce travail et en ouvrant la réflexion sur les futures évolutions possibles dans le domaine de la détection du strabisme par IA.

# CHAPITRE 5. PERSPECTIVES ET CONCLUSION

Ce mémoire avait pour objectif d'évaluer dans quelle mesure le modèle de détection d'objets YOLOv4 peut identifier le strabisme oculaire avec une précision égale ou supérieure à celle des méthodes cliniques traditionnelles, afin de contribuer à un dépistage précoce aussi bien à domicile qu'en milieu médical. Le projet visait à concevoir un système basé sur la vision par ordinateur, capable de surpasser les méthodes de dépistage traditionnelles. Pour ce faire, l'approche adoptée s'appuyait sur le modèle YOLOV4, réputé pour ses performances dans la détection d'objets, afin de relever ce défi médical.

Les résultats obtenus lors de l'entraînement et des tests du modèle YOLOV4 ont démontré des performances globalement satisfaisantes, avec un taux de réussite global de **95,24 %** sur les 63 cas analysés. Le modèle a montré une grande précision dans la classification des cas normaux, avec un taux de réussite de **93,33 %** (28 cas normaux correctement classifiés sur 30) et des probabilités souvent proches de **1**, illustrant sa fiabilité dans cette catégorie. De plus, la majorité des images de strabisme ont été correctement détectées, avec un taux de réussite de **96,67 %** (29 cas correctement classifiés sur 30). Cependant, certaines limitations subsistent, notamment en ce qui concerne la détection des strabismes divergents et l'ajustement des boîtes englobantes. Ces difficultés ont conduit à des probabilités plus faibles pour certains cas spécifiques, notamment une probabilité de **0,5210** pour un cas normal mal classifié comme strabisme, et une probabilité de **0,6142** pour un cas de strabisme mal classifié comme normal.

Bien que l'IA, et en particulier YOLOv4, se soit révélée capable de détecter le strabisme avec une précision élevée (environ 95 %), certaines limitations subsistent dans la gestion des variations de strabisme plus complexes, telles que les cas divergents et convergents. Des améliorations, comme un ajustement des hyperparamètres ou un entraînement prolongé avec des images cliniques plus variées, pourraient significativement améliorer les performances globales.

L'une des contributions majeures de ce mémoire réside dans l'application de l'IA, et plus précisément de la vision par ordinateur, à la détection médicale. D'un point de vue pratique, un système tel que celui proposé pourrait permettre aux familles de détecter le strabisme à domicile, réduisant ainsi les délais de consultation. Dans un contexte clinique, ce modèle pourrait également faciliter le travail des spécialistes, notamment dans les régions où l'accès aux soins ophtalmologiques est limité. De plus, l'intégration de ce système dans des solutions

de télémédecine pourrait améliorer l'accès au diagnostic à distance, rendant le dépistage plus accessible et plus rapide.

Bien que les résultats obtenus soient prometteurs, des ajustements sont nécessaires pour améliorer la précision du modèle. Une piste à explorer serait d'élargir et de diversifier la base de données d'entraînement, en intégrant davantage d'images cliniques et de meilleure qualité. Un ajustement des hyperparamètres, tels que la taille des lots et le taux d'apprentissage, pourrait également contribuer à réduire les pertes et à optimiser les performances globales du modèle. Il serait également intéressant de comparer les performances de YOLOV4 à celles d'autres modèles de vision par ordinateur pour déterminer le meilleur algorithme pour cette tâche.

En conclusion, ce travail a démontré la faisabilité d'un système de détection automatisé du strabisme basé sur l'IA, avec des résultats encourageants malgré des marges d'amélioration. Ce projet ouvre des perspectives importantes pour améliorer la qualité des soins oculaires, en rendant le dépistage du strabisme plus accessible et plus fiable. L'originalité de ce travail repose sur l'utilisation de YOLOV4 dans un domaine médical, constituant ainsi une avancée notable dans l'application de l'IA à la santé visuelle. Toutefois, la littérature 2023-2025 signale que des architectures plus récentes – telles que YOLOv5, YOLOv8 ou EfficientDet-D0 – peuvent, selon le contexte, offrir un meilleur compromis entre sensibilité, vitesse d'inférence et empreinte mémoire ; une comparaison approfondie avec ces modèles représenterait donc une étape naturelle pour optimiser davantage le dépistage automatisé du strabisme et préparer son déploiement clinique à grande échelle [51-55].

# Bibliographie


[1] E. B. Quoc and M.-A. Espinasse-Berrod, “Strabisme chez l’enfant,” *EMC – Pédiatrie*, vol. 1, no. 4, pp. 397–409, 2004.

[2] J. M. Pedespan and S. Cabasson, “6Strabisme,” 2011.

[3] S. A. Cotter, C. L. Cyert, L. A. Miller, M. J. Quinn, *et al.*, “Vision screening for children 36 to < 72 months: recommended practices,” *Optometry and Vision Science*, vol. 92, no. 1, p. 6, 2015.

[4] X. Huang, S. J. Lee, C. Z. Kim, *et al.*, “An automatic screening method for strabismus detection based on image processing,” *PLOS ONE*, vol. 16, no. 8, pp. 1–14, 2021.

[5] . K. Karaaslan, S. Güngör, and G. M. Karaaslan, “A new method based on deep learning and image processing for detection of strabismus with the Hirschberg test,” *Photodiagnosis and Photodynamic Therapy*, vol. 44, 2023.

[6] . Huang, S. J. Lee, C. Z. Kim, *et al.*, “An improved strabismus screening method with combination of meta-learning and image processing under data scarcity,” *PLOS ONE*, vol. 17, no. 8, pp. 4–13, 2022.

[7] J. F. Lu, Z. Zhang, C. Zhou, *et al.*, “Automated strabismus detection for telemedicine applications,” *arXiv preprint*, arXiv:1809.02940, pp. 2–3, 2018.

[8] Z. Chen, F. Huang, H. Liu, W.-L. Li, *et al.*, “Strabismus recognition using eyetracking data and convolutional neural networks,” *Journal of Healthcare Engineering*, vol. 2018, 2018.

[9] B. U. Yadav and N. Yadav, “Comparative study of object detection algorithms,” *International Research Journal of Engineering and Technology (IRJET)*, vol. 4, no. 11, pp. 586–591, 2017.

[10] M. S. Waheed, S. R. Shama, N. M. Rafique, R. M. Sajid, and M. S. Malik, “Deep learning algorithms-based object detection and localization revisited,” *Journal of Physics: Conference Series*, IOP Publishing, 2021.

[11] R. Girshick, “Fast R-CNN,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1440–1448.

[12] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” in *Advances in Neural Information Processing Systems*, vol. 28, 2015.

[13] W. Liu, A. Rabinovich, E. Ding, *et al.*, "SSD: Single shot multibox detector," in *Proceedings of the 14th European Conference on Computer Vision (ECCV)*, 2016.

[14] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.

[15] T.-Y. Lin, P. Goyal, R. Girshick, *et al.*, "Focal loss for dense object detection," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2980–2988.

[16] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," *arXiv preprint*, arXiv:2004.10934, 2020.

[17] H. Cai, W. Qin, Q. Chen, T. Liu, *et al.*, "The cross-depiction problem: Computer vision algorithms for recognizing objects in artwork and in photographs," *arXiv preprint*, arXiv:1505.00110, 2015.

[18] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 2005, pp. 886–893.

[19] T. Dean, M. A. Ruzon, M. Segal, J. Shlens, S. Vijayanarasimhan, D. Anguelov, and J. Yagnik, "Fast, accurate detection of 100,000 object classes on a single machine," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 1814–1821.

[20] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686–707, 2019. [Online]. Available: https://doi.org/10.1016/j.jcp.2018.10.045

[21] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," *arXiv preprint*, arXiv:2004.10934, 2020. [Online]. Available: https://arxiv.org/abs/2004.10934

[22] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788. [Online]. Available: https://doi.org/10.1109/CVPR.2016.91

[23] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Scaled-YOLOv4: Scaling cross stage partial network," *arXiv preprint*, arXiv:2011.08036, 2020. [Online]. Available: https://arxiv.org/abs/2011.08036

[24] C.-Y. Wang, H.-Y. M. Liao, I.-H. Yeh, Y.-H. Wu, P.-Y. Chen, and J.-W. Hsieh, “CSPNet: A new backbone that can enhance learning capability of CNN,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2020, pp. 390–391.

[25] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org/

[26] J. Redmon and A. Farhadi, “YOLO9000: Better, faster, stronger,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6517–6525.

[27] S. J. Lee, C. Z. Kim, J.-H. Cho, *et al.*, “Deep CNN-based corneal reflex analysis for accurate strabismus screening,” *PLOS ONE*, vol. 17, no. 10, p. e0274632, 2022.

[28] Y. Li, M. Feng, and L. Wu, “Automated angle measurement for strabismus using enhanced eye-tracking and image processing,” *Biomedical Signal Processing and Control*, vol. 70, p. 103015, 2021.

[29] S. Rahimi, H.-Y. Jung, and S. J. Lee, “Strabismus detection in children through smartphone-based videorefraction,” *Sensors*, vol. 21, no. 22, p. 7645, 2021.

[30] P. Wang, X. Ji, and L. Zhou, “Hybrid Hirschberg and Purkinje image technique for robust strabismus diagnosis,” *IEEE Access*, vol. 10, pp. 50500–50509, 2022.

[31] A. Brown, X. Chen, and T. Zhang, “Augmenting limited pediatric strabismus data with GANs for improved classifier performance,” *Computer Methods and Programs in Biomedicine*, vol. 225, p. 107042, 2022.

[32] L. Cortez, X. Huang, and C. Z. Kim, “Evaluation of deep learning-based eye alignment detection for esotropia and exotropia,” *Journal of Telemedicine and Telecare*, vol. 29, no. 1, pp. 56–64, 2023.

[33] U. Ahmad, M. Ibrahim, and M. Saeed, “Low-cost strabismus screening using smartphone imaging and CNN analysis,” *IEEE Journal of Biomedical and Health Informatics*, vol. 26, no. 12, pp. 6105–6113, 2022.

[34] M. Yu, L. Fang, and Z. Chen, “Computer-aided strabismus detection: A comparative study of classic and deep feature extraction methods,” *Journal of Healthcare Engineering*, vol. 2022, Art. ID 9492367, 2022.

[35] M. Tan and Q. V. Le, “EfficientDet: Scalable and efficient object detection,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 10781–10790.

[36] Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, “Object detection with deep learning: A review,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 11, pp. 3212–3232, 2019.

[37] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, “CenterNet: Keypoint triplets for object detection,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 6569–6578.

[38] Z. Zheng, P. Wang, W. Liu, *et al.*, “Distance-IoU loss: Faster and better learning for bounding box regression,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 7, pp. 12993–13000, 2020.

[39] H. Wu, J. Xu, P. Dai, and G. Gu, “Comparative analysis of YOLOv3, YOLOv4, and Faster R-CNN for real-time object detection,” *International Research Journal of Engineering and Technology (IRJET)*, vol. 8, no. 9, pp. 453–460, 2021.

[40] L. Liu, W. Ouyang, X. Wang, *et al.*, “Deep learning for generic object detection: A survey,” *International Journal of Computer Vision*, vol. 128, no. 2, pp. 261–318, 2020.

[41] M. M. Rahman, X. Chen, J. Qi, *et al.*, “A systematic review on region proposal methods for object detection,” *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 5, pp. 5435–5450, 2021.

[42] Z. Ge, S. Liu, F. Wang, and Z. Li, “YOLOX: Exceeding YOLO series in 2021,” *arXiv preprint*, arXiv:2107.08430, 2021.

[43] A. D. Jagtap, K. Kawaguchi, and G. E. Karniadakis, “Adaptive activation functions accelerate convergence in deep and physics-informed neural networks,” *Journal of Computational Physics*, vol. 404, p. 109136, 2020.

[44] G. E. Karniadakis, R. M. Kirby II, P. Perdikaris, *et al.*, “Numerical analysis and deep learning for partial differential equations,” in *Handbook of Numerical Analysis*, vol. 23, pp. 269–314, 2021.

[45] M. Raissi, P. Perdikaris, and G. E. Karniadakis, “Physics-informed neural networks for quantum mechanics,” *SIAM Journal on Scientific Computing*, vol. 41, no. 4, pp. B425–B446, 2019.

[46] J. Howard and S. Gugger, *Deep Learning for Coders with fastai and PyTorch*. O’Reilly Media, 2020.

[47] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, “YOLOv4-tiny: Real-time object detection on edge devices,” *arXiv preprint*, arXiv:2011.04244, 2021.

[48] N. H. Abdulkareem, B. Usman, and N. Salim, “Implementation of YOLOv5 for real-time object detection,” *Journal of Physics: Conference Series*, vol. 1962, no. 1, p. 012006, 2021.

[49] F. S. Saleh, S. N. H. Abdullah, and M. S. Hossain, “Lightweight real-time object detection based on MobileNet and YOLO,” *IEEE Access*, vol. 9, pp. 59346–59359, 2021.

[50] A. Paszke, S. Gross, F. Massa, *et al.*, “PyTorch: An imperative style, highperformance deep learning library,” in *Advances in Neural Information Processing Systems*, vol. 32, pp. 8024–8035, 2019.

[51] Bachir N., Tamine L., Chiblak S. *Benchmarking YOLOv5 Models for Improved Human Detection in Search-and-Rescue Missions.* **Pattern Recognition Letters**, 2024, 178: 1-10. DOI: 10.1016/j.patrec.2024.01.019.

[52] Khanam R., Asghar T., Hussain M. *Comparative Performance Evaluation of YOLOv5, YOLOv8 and YOLOv11 for Solar Panel Defect Detection.* **Sensors**, 2025, 25(1): 6. DOI: 10.3390/s25010006.

[53] Li J., Chen W., Zhang Y. *YOLOv5 vs. YOLOv8: Performance Benchmarking in Wildfire and Smoke Detection.* **Journal of Image and Graphics**, 2024, 12(2): 127-135.

[54] Ibrahim M., Al-Fuqaha A., Bensaali F. *Target Detection and Classification via EfficientDet and CNN over Aerial Image Sequences.* **Frontiers in Neurorobotics**, 2024, 18: 1448538. DOI: 10.3389/fnbot.2024.1448538.

[55] Zhang H., Li X., Wang R. *Deep Learning-Based Object Detection Algorithms in Medical Imaging: A Review.* **Heliyon**, 2024, 10(4): e17618. DOI: 10.1016/j.heliyon.2024.e17618.

[56] Bochkovskiy A., Wang C.-Y., Liao H.-Y.M. (2020). *YOLOv4: Optimal Speed and Accuracy of Object Detection*. arXiv preprint arXiv:2004.10934.

[57] M. Abràmoff, Y. Lavin, M. Birch, N. Shah, et al., “Pivotal trial of an autonomous AI-based diagnostic system for detection of diabetic

retinopathy in primary care offices," *NPJ Digital Medicine*, vol. 1, no. 1, 2018.