



République Algérienne Démocratique et Populaire  
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique  
Université des Sciences et de la Technologie Houari Boumediene



Faculté De L'informatique  
Département IA & SD

## Mémoire De Licence en Informatique

Spécialité : Informatique Académique

---

### Thème :

Détection automatique du bon ou mauvais port du masque  
dans le contexte de la COVID19.

---

Proposé et encadré par :  
**Dr. Dahmani Djamila**

Réalisé par :  
**MESMOUS Meroua**  
**METANE Youcef El Khodr**

Soutenu le :19/06/2022

Devant le jury composé de :

**Pr Baha Nadia (Présidente)**  
**Dr Bellala F. Zohra (Membre)**

Binôme n° : ACAD\_I\_058 / 2022

# Table des matières

INTRODUCTION GÉNÉRALE.....	1
CHAPITRE I : ÉTAT DE L'ART .....	2
1. INTRODUCTION.....	2
2. HISTORIQUE .....	2
3. NOTIONS DE BASE.....	3
3.1. Apprentissage automatique .....	3
3.1.1. Apprentissage supervisé .....	3
3.1.2. Apprentissage non-supervisé.....	4
3.1.3. Apprentissage par renforcement.....	4
3.2. Réseaux Neuronaux .....	4
3.3. Apprentissage Profond .....	4
3.4. La détection d'Objets.....	5
3.4.1. La detection des visages.....	5
3.5. Espaces colorimétrique pour la segmentation de la peau.....	6
4. TECHNIQUES DE DETECTION D'OBJETS .....	8
4.1. Méthode de Viola et Jones .....	8
4.2. Techniques basées sur l'apprentissage profond.....	8
4.2.1. R-CNN (Region-based Convolutional Network) .....	9
4.2.2. SSD (Single Shot Detection).....	9
4.2.3. YOLO (You Only Look Once) .....	10
5. LE RESEAU YOLO .....	10
5.1. Définition.....	10
5.2. Histoire .....	11
5.3. Comment YOLO fonctionne ?.....	11
5.4. YOLOv5 .....	12
5.5. Architecture du réseau YOLO .....	13
6. CONCLUSION.....	14
CHAPITRE II: CONCEPTION .....	15
1. INTRODUCTION.....	15
2. LA DETECTION DES VISAGES.....	15
2.1. Le détecteur SSD (Single Shot Detection) .....	16
3. COLLECTE DES DONNEES.....	16
3.1. Ensemble de données 1 .....	17
3.2. Ensemble de données 2 .....	18
4. MODELES CREES .....	19
4.1. Modèle n°1 : basé sur les CNNs et l'apprentissage par transfert.....	19
4.1.1. Transfer Learning .....	19
4.1.2. Techniques d'utilisation de l'apprentissage par transfert.....	20
4.1.3. Applications de Keras.....	21
4.1.4. MobileNetV2 .....	21
4.2. Modèle n°2 : basé sur la détection des pixels peau .....	22
4.2.1. Détection des pixels peau .....	22
4.2.2. Résumé de cette approche .....	23

5. CONCLUSION .....	25
CHAPITRE III: IMPLEMENTATION .....	26
1. INTRODUCTION.....	26
2. ENVIRONNEMENT DE TRAVAIL .....	26
2.1. Matériels.....	26
2.2. Logiciels.....	26
3. EXPERIMENTATIONS .....	29
3.1. Architecture du CNN :.....	29
3.1.1. Expérience N°1 : .....	29
3.1.2. Expérience N°2.....	30
3.1.3. Expérience N°3.....	30
3.1.4. Expérience N°4.....	31
3.2. Détection du masque en se basant sur les pixels peau .....	32
3.3. Le réseau YOLOv5 .....	33
4. VISUALISATION DES RESULTATS.....	35
4.1. La courbe ROC et la matrice de confusion .....	35
4.2. Discussion .....	37
5. CONCLUSION .....	37
CONCLUSION GENERALE .....	38
1. BIBLIOGRAPHIE.....	39
2. WEBOGRAPHIE.....	41
3. RESUME.....	42

## Table des figures

Figure 1 Fonctionnement de l'apprentissage automatique [Web 1] .....	3
Figure 2 Relation entre IA,ML,DL,NN [Web 2] .....	5
Figure 3 Comment ça marche, la détection d'objets? .....	5
Figure 4 Espace colorimétrique RGB [Web 3] .....	6
Figure 5 Espace colorimétrique HSV [Web 4] .....	7
Figure 6 Fonctionnement du R_CNN [Girshick et al 2013] .....	9
Figure 7 Détecteur SSD Multibox [Anguelov and al 2016] .....	10
Figure 8 Fonctionnement du réseau Yolo [Redmon et al] .....	12
Figure 9 Comparaison de performance des versions YOLOv5 [Web 7] .....	13
Figure 10 Diagramme de structure du réseau YOLO [Web 8] .....	14
Figure 11 Diagramme résumant les étapes du traitement .....	15
Figure 12 Architecture du réseau SSD [Angulov and al 2016] .....	16
Figure 13 Présentation de la collecte de l'ensemble de données 1 .....	17
Figure 14 Aperçu de l'ensemble de données N°2 .....	19
Figure 15 Approche traditionnelle vs. Approche de Transfert Learning [Web 10] .....	20
Figure 16 Détection des pixels peau .....	23
Figure 17 Conversion de l'image en HSV et Y'CrCb .....	23
Figure 18 Les différents masques utilisés .....	24
Figure 19 Métriques de l'évaluation sans réglage fin (Fine Tuning) .....	30
Figure 20 Métriques de l'évaluation avec le réglage fin (Fine Tuning) .....	31
Figure 21 Pseudo-algorithme de détection des masques_1 .....	31
Figure 22 Métriques de l'évaluation avec l'ensemble de données_2 .....	32
Figure 23 Pseudo-algorithme de détection des masques_2 .....	32
Figure 24 Pseudo-algorithme de seuillage .....	33
Figure 25 Tests en temps réel .....	34
Figure 26 Tests réalisés avec les trois approches .....	34
Figure 27 Courbe ROC et matrice de confusion –Modèle1 .....	35
Figure 28 Courbe ROC et matrice de confusion -Modèle2 .....	36
Figure 29 Courbe ROC et matrice de confusion -YOLOv5 .....	36

## Table des équations

Équation 1 La moyenne des pixels par canal .....	18
Équation 2 Opération faite par la couche Dense .....	22
Équation 3 Formule de la fonction Sigmoid .....	29
Équation 4 Formule de la fonction BinaryCrossentropy .....	29

# *Remerciement*

*Tous d'abord nous remercions DIEU le tout puissant de nous avoir donné la force, le courage, et la volonté nécessaire pour réaliser ce modeste travail.*

*Nos remerciements s'adressent en premier lieu à notre encadreur, Madame DAHMANI Djamila pour nous avoir proposé ce sujet, pour sa confiance, son orientation, et sa patience qu'elle nous a accordée.*

*Que Madame BAHHA Nadia trouve ici l'expressions de nos vifs remerciements pour avoir bien voulu accepter de présider le jury de ce mémoire.*

*Nous tenons à remercier également Madame BELLALA F.Zohra pour l'intérêt porté à notre travail en acceptant d'examiner et d'évaluer ce mémoire.*

*Enfin nous tenons nos sincères remerciements à tous nos proches et amis, qui nous ont toujours soutenu et encouragé.*

*Meroua & Youcef*

# *Dédicace*

*À mes chers parents Youssef et Nassira  
Qui ont sacrifié leur vie pour la réussite de leurs enfants et  
m'ont éclairé le chemin par leurs conseils judicieux.  
J'espère qu'un jour, je pourrai leurs rendre un peu de ce  
qu'ils ont fait pour moi, que dieu leur prête bonheur et  
longue vie.*

*À ma chère sœur ILham que j'aime ainsi que sa petite  
famille Jana, Iyad et Amine*

*À mes deux chers frères Hamza et Seyf El-Islam et leurs  
familles Talia, Adem, Sabah et Houaria.*

*À ma chère adorable cousine Lina*

*À la mémoire de ma grand-mère, ma tante et mes oncles*

*À toute la famille Mesmous et Hamreras*

*À mon binôme Youssef, C'était un grand plaisir de partager  
ce travail avec toi.*

*À mes très chères amies, mes camarades et tous ceux qui  
m'ont soutenu, de près ou de loin*

*Meroua*

# *Dédicace*

*Je dédie ce travail*

*À mes parents*

*Pour leur soutien et leurs sacrifices tout au long de mon  
parcours.*

*À ma sœur*

*Qui m'a toujours soutenue et encouragée.*

*Youcef*

# INTRODUCTION GÉNÉRALE

À la fin de l'année 2019, un nouveau virus nommé SARS-CoV-2 est détecté, quelques mois après, l'Organisation Mondiale de la Santé (OMS) décrète ce virus comme responsable d'une pandémie dénommée COVID-19.

La voie de transmission aérienne du SRAS-CoV-2 est une voie principale pour la propagation de ce virus. À l'heure actuelle, il n'existe pas de charge virale contaminante-seuil pour les aérosols ni de dose infectieuse minimale pour provoquer une infection. Cependant, il convient de garder à l'esprit que la bouche, le nez et les yeux sont des portes d'entrées pour la Covid-19 qui se transmet entre autres, par le biais du contact direct, la toux ou l'éternuement.

À la date du 16 avril 2022, la pandémie COVID-19 a touché plus de 467 millions de personnes dans le monde, dont 11 millions en Afrique (265 000 cas en Algérie).

L'un des défis de la lutte contre cette pandémie consiste à limiter la transmission du SRAS-CoV-2 par des personnes asymptomatiques ou pré-symptomatiques. Une revue systématique de la littérature couplée à une méta-analyse confirme que L'hygiène personnelle est la principale mesure pour empêcher la propagation du virus et le port d'un masque est considéré comme nécessaire et même obligatoire dans de nombreux pays.

Pour cela, Nous avons pensé à créer un système qui consiste à détecter le port du masque facial et identifié son état. Ce système peut être mise en œuvre dans plusieurs technologies.

Le travail réalisé et présenté dans ce mémoire est scindé en 3 chapitres :

**Le chapitre I :** Ce chapitre introductif sera dédié à la présentation des notions de base ainsi que les différentes études et recherches réalisées.

**Le chapitre II :** Ce chapitre sera dédié au prétraitement des données et la conception des différents modèles.

**Le chapitre III :** On présente dans ce chapitre les expériences faites ainsi que les résultats obtenus à la fin de l'implémentation.



# Chapitre I : État de l'art

## 1. Introduction

La détection des masques faciaux est une tâche difficile qui a eu de plus en plus d'attention à cette époque en raison de la propagation du corona virus ainsi de nombreux pays sont devenus obligés de mettre cette tâche en pratique. L'une des raisons pour laquelle elle est considérée difficile est le manque d'existence d'ensembles de données d'images annotées (datasets), qui est une condition préalable à l'entraînement des techniques d'apprentissage profond.

## 2. Historique

La détection automatique du port de masques est une tâche très importante à l'heure actuelle où les différents virus peuvent entraîner des pandémies. Un tel système peut être basé sur un apprentissage profond ou un apprentissage automatique de manière générale. Son objectif est de prévenir la propagation du COVID-19 et d'autres maladies respiratoires. Cette procédure passe par deux étapes principales : la détection du visage, et la détection du masque sur le visage.

Au cours des dernières années, de nombreuses études et recherches ont été réalisées. On peut citer :

[Shaik et al] ont utilisé l'apprentissage profond en temps réel pour classer et reconnaître les émotions, et VGG-16 a été utilisé pour catégoriser les sept visages. Cette approche prospère dans la période de confinement pour prévenir la propagation des cas de COVID-19. De plus, [Ejaz et al] a utilisé l'analyse en composantes principales pour reconnaître un visage masqué d'un visage non masqué.

Dans leurs études, [Day et al] ont proposé MobileNetMask modèle de détection des masques faciaux fondé sur l'apprentissage profond. Les procédures d'apprentissage et de tests ont été réalisées en utilisant deux ensembles de données différents avec deux classes, avec plus de 5200 images. À la suite de l'essai, qui utilise 770 échantillons de vérification, on a obtenu une précision de classification d'environ 93 %.

En utilisant l'architecture YoloV3, [Bhuiyan et al] ont élaboré une étude pour déterminer si les personnes portent un masque ou non. Une précision de classification de 96 % a été atteinte.

Visant à la détection des masques faciaux multi-échelles en temps-réel, [Addagarla et al] ont proposé deux modèles différents. Les algorithmes YOLOv3, NasNetMobile et ResNet-SSD300

ont été utilisés dans les modèles proposés. Par conséquent, les taux de rappel étaient de 98 % et de 99 % pour les deux modèles.

Dans [6], les auteurs ont mis au point une nouvelle méthode d'identification de l'état du masque facial. Ils ont pu classer trois catégories de conditions de port du masque. Les catégories sont le port correct du masque, le port incorrect du masque et l'absence de port du masque. La méthode proposée a atteint une précision de 98,70 % dans la phase de détection en taille.

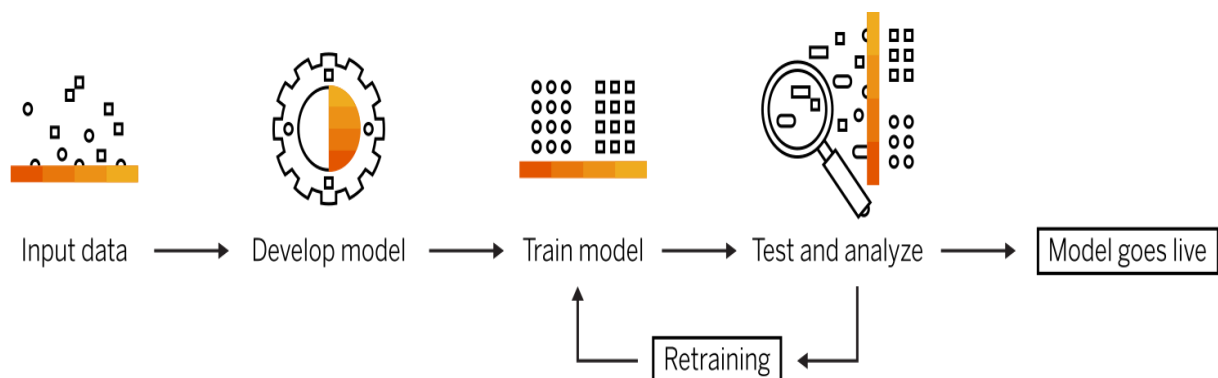
### 3. Notions de base

Dans cette section, on va décrire les différentes méthodes et algorithmes qui permettent l'identification de l'état du masque.

L'identification de l'état du masque se compose de trois catégories essentielles : Absence du masque, masque bien porté et masque mal-porté.

#### 3.1. Apprentissage automatique

L'apprentissage automatique est un sous-ensemble de l'intelligence artificielle (IA) qui peut fonctionner sur la base de deux types principaux d'approches, un apprentissage supervisé et un apprentissage non supervisé. Il vise à enseigner aux ordinateurs à apprendre à partir de données et à s'améliorer avec l'expérience - au lieu d'être explicitement programmé pour le faire. Dans l'apprentissage automatique, les algorithmes sont formés pour trouver des modèles et des corrélations dans de grands ensembles de données et pour prendre les meilleures décisions et prédictions sur la base de cette analyse.



*Figure 1 Fonctionnement de l'apprentissage automatique [Web 1]*

##### 3.1.1. Apprentissage supervisé

L'apprentissage supervisé est le paradigme d'apprentissage la plus utilisée en Apprentissage automatique et en apprentissage profond. Comme son nom l'indique, il consiste à surveiller

l'apprentissage de la machine en lui présentant des exemples annotés (ensemble de données) de ce qu'il doit effectuer.

### **3.1.2. Apprentissage non-supervisé**

À la différence de l'apprentissage supervisé, l'apprentissage non supervisé est celui où l'algorithme doit opérer à partir d'exemples non annotés et des données brutes. Dans ce cas de figure, la machine obtient ses résultats en se fondant sur la détection de similarités entre certaines de ces données.

### **3.1.3. Apprentissage par renforcement**

L'apprentissage par renforcement ou « Reinforcement Learning » en anglais est un procédé d'apprentissage automatique de plus en plus utilisé. Il consiste, pour un système autonome, à apprendre les actions à réaliser à partir d'expériences de façon à optimiser une récompense quantitative au cours du temps. Le système est plongé au sein d'un environnement, et prend ses décisions en fonction de son état courant. En retour, l'environnement procure une récompense, qui peut être positive ou négative.

Au fil des expériences, le système cherche un comportement décisionnel optimal, en ce sens qu'il maximise la somme des récompenses au cours du temps.

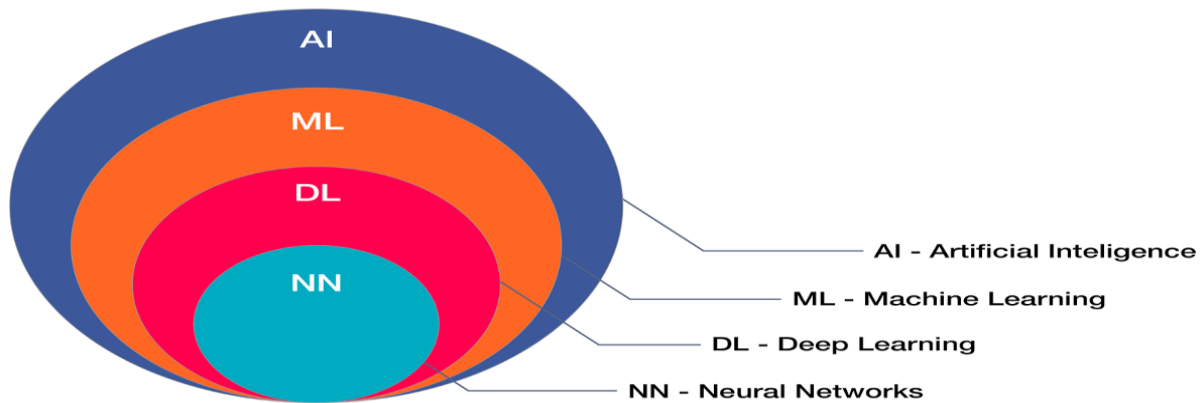
## **3.2. Réseaux Neuronaux**

Un réseau neuronal artificiel (ANN) est modelé sur les neurones d'un cerveau biologique. Les neurones artificiels sont appelés nœuds et sont regroupés en plusieurs couches, fonctionnant en parallèle. Lorsqu'un neurone artificiel reçoit un signal numérique, il le traite et le transmet aux autres neurones qui lui sont connectés. Comme dans un cerveau humain, le renforcement neuronal permet d'améliorer la reconnaissance des formes, l'expertise et l'apprentissage en général.

## **3.3. Apprentissage Profond**

Ce type d'apprentissage automatique est appelé "profond" car il comprend de nombreuses couches du réseau neuronal et des volumes massifs de données complexes et disparates. Pour réaliser l'apprentissage profond, le système s'engage avec plusieurs couches du réseau, en extrayant des résultats de plus en plus élevés. Par exemple, un système d'apprentissage profond qui traite des images de la nature et recherche des marguerites *Gloriosa* reconnaîtra, à la première couche, une plante. Au fil des couches neuronales, il identifiera ensuite une fleur, puis une marguerite, et enfin une marguerite *Gloriosa*. Parmi les exemples d'applications

d'apprentissage profond, citons la reconnaissance vocale, la classification d'images et l'analyse pharmaceutique.



*Figure 2 Relation entre IA,ML,DL,NN [Web 2]*

### 3.4. La détection d'Objets

La détection d'objets est une technologie informatique liée à la vision par ordinateur et au traitement d'images qui traite de la détection d'instances d'objets sémantique d'une certaine classe (tels que des humains, des bâtiments ou des voitures) dans des images et des vidéos numériques. Les domaines bien documentés de la détection d'objets comprennent la détection des visages et la détection des piétons. La détection d'objets a des applications dans de nombreux domaines de la vision par ordinateurs, y compris la récupération d'images et la vidéo-surveillance.



*Figure 3 Comment ça marche, la détection d'objets?*

#### 3.4.1. La détection des visages

La détection des visages humains est une technologie informatique basée sur l'IA qui permet d'identifier et de localiser la présence de visages humains dans des photos et des vidéos numériques. Elle peut être considérée comme un cas particulier de détection de classe d'objets,

où la tâche consiste à trouver les emplacements et à spécifier les tailles de tous les objets qui appartiennent à une classe donnée - dans ce cas, les visages - dans une ou plusieurs images spécifiques.

Donc elle répond simplement à deux questions : 1. Y a-t-il des visages humains dans les images ou la vidéo collectées ? 2. où se trouve-t-il ? Elle est depuis longtemps l'un des problèmes les plus fondamentaux de la vision par ordinateur et de l'interaction homme-machine. Au cours de la dernière décennie, le travail le plus influent devrait être le cadre de détection des visages proposé par Viola et Jones.

### 3.5. Espaces colorimétrique pour la segmentation de la peau

Un espace de couleur est un modèle mathématique abstrait utilisé dans le traitement des images et des signaux qui décrit comment les couleurs peuvent être représentées sous la forme d'un ensemble de nombres (par exemple, un triple en RVB ou un quadruplet en CMJN). Les modèles de couleurs peuvent généralement être décrits à l'aide d'un système de coordonnées où chaque couleur du système est représentée par un point unique dans l'espace de coordonnées.

#### RGB

L'espace colorimétrique RGB stocke des valeurs individuelles pour le rouge, le vert et le bleu. Dans un espace couleur basé sur le modèle RGB, les trois primaires sont additionnés pour créer des couleurs allant du blanc complet au noir complet.

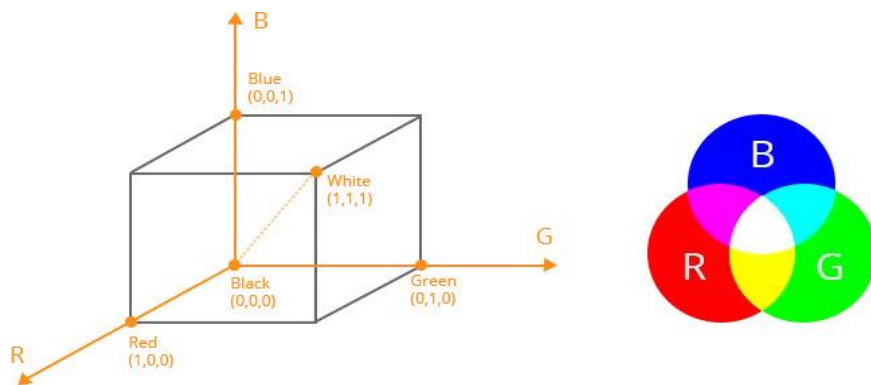


Figure 4 Espace colorimétrique RGB [Web 3]

*Matrice d'une image colorée = matrice rouge + matrice verte + matrice bleue*

## HSV

HSV (Hue=Teinte, Saturation=saturation, Value=valeur), est souvent utilisé par les artistes car il est souvent plus naturel de penser à une couleur en termes de teinte et de saturation qu'en termes de composantes de couleur additives ou soustractives.

- Teinte représente la perception de la couleur.
- La saturation décrit la pureté de la couleur, c'est-à-dire son caractère vif ou terne.
- La valeur, indiquant la quantité de lumière de la couleur, c'est-à-dire son aspect clair ou sombre.

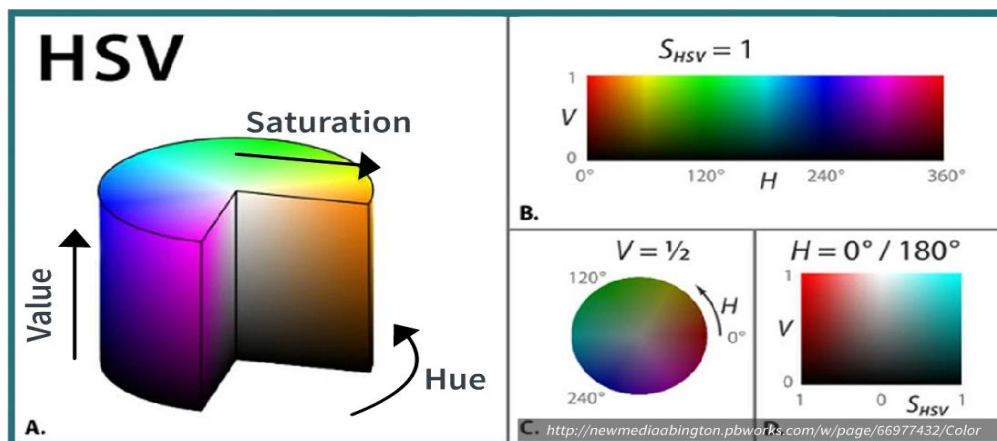


Figure 5 Espace colorimétrique HSV [Web 4]

## Y'CbCr

L'espace couleur YCrCb est un signal non-linéaire dérivé de l'espace couleur RGB et possède les trois composantes suivantes :

- Y : Luminance ou composante Luma obtenue à partir de RGB après correction gamma (c'est-à-dire l'information en noir et blanc).
- Cr = R - Y : À quelle distance de la composante Luma se trouve la composante rouge (l'information couleur).
- Cb = B - Y : À quelle distance se trouve la composante bleue du Luma (l'information couleur).

Cet espace de couleur est principalement utilisé dans la compression (des composantes Cr et Cb) dans la transmission TV.

La détection des pixels peau peut être effectué en utilisant plusieurs espaces de couleurs avec un seuillage agréé.

## 4. Techniques de détection d'objets

### 4.1. Méthode de Viola et Jones

La méthode de Viola et Jones est une méthode de détection d'objets dans une image numérique, proposée par les chercheurs Paul Viola et Michael Jones en 2001. Elle fait partie des toutes premières méthodes capables de détecter efficacement et en temps réel des objets dans une image. Cette méthode nécessite de quelques centaines à plusieurs milliers d'exemples de l'objet que l'on souhaite détecter, pour entraîner un **classifieur**. Une fois son apprentissage réalisé, ce classifieur est utilisé pour détecter la présence éventuelle de l'objet dans une image en parcourant celle-ci de manière exhaustive, à toutes les positions et dans toutes les tailles possibles. La méthode de Viola et Jones a été l'une des méthodes les plus connues et les plus utilisées, en particulier pour la détection de visages et la détection de personnes [Viola et Jones].

Il existe d'autres méthodes mais ce qui la différencie des autres est notamment :

- L'utilisation d'**images intégrales** qui permettent de calculer plus rapidement les caractéristiques
- La **sélection par boosting** des caractéristiques qui consiste à construire un classifieur « fort » à partir d'une combinaison pondérée de classifieurs « faibles » en utilisant un algorithme de boosting (en pratique une version modifiée d'AdaBoost).
- La combinaison en **cascade de classifieurs** : L'approche de la recherche exhaustive sur l'ensemble de l'image utilisée par la méthode de Viola et Jones est extrêmement coûteuse en calcul. L'idée-clé pour réduire ce coût réside dans l'utilisation séquentielle d'une cascade de classifieurs.

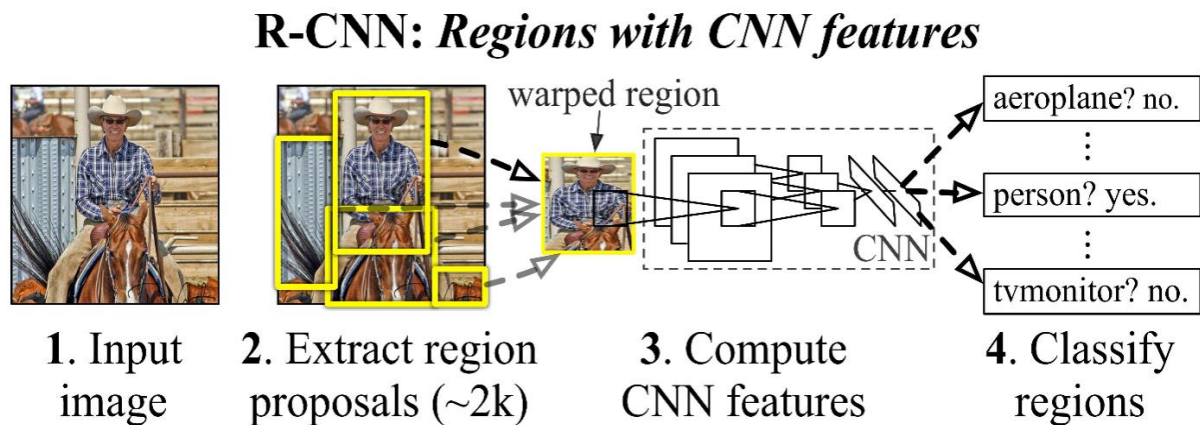
### 4.2. Techniques basées sur l'apprentissage profond

Les techniques d'apprentissage profond sont capables d'effectuer une détection d'objets de bout en bout sans définir spécifiquement les caractéristiques, et sont généralement basées sur des réseaux neuronaux convolutifs (CNN).

Un réseau neuronal convolutif (CNN, ou ConvNet) est un type particulier de réseau neuronal multicouche, conçu pour reconnaître des motifs visuels directement à partir d'images en pixels.

#### 4.2.1. R-CNN (Region-based Convolutional Network)

R\_CNN est un modèle de détection d'objets qui utilise des CNN à haute capacité pour faire des propositions de régions de bas en haut afin de localiser et de segmenter les objets. Comme les CNNs étaient trop lents et très coûteux en termes de calcul, ce modèle conçu par Ross Girshick résout ce problème en utilisant un algorithme de proposition d'objet appelé "Selective Search" qui réduit le nombre de boîtes englobantes fournies au classificateur à près de 2000 propositions de régions en regroupant les pixels adjacents par texture, couleur ou intensité.



*Figure 6 Fonctionnement du R\_CNN [Girshick et al 2013]*

Le problème est que la première implémentation de R-CNN en 2013 était vraiment lente. Cela laissait beaucoup de place à l'amélioration, et c'est ce qui a été réalisé en 2015 avec Fast R-CNN, et plus tard Faster R-CNN. Celles-ci ont remplacé l'étape sélective par le réseau de proposition de région (RPN), faisant finalement de R-CNN un détecteur d'objets de bout en bout "end-to-end" en Apprentissage profond.

#### 4.2.2. SSD (Single Shot Detection)

Les modèles précédents donnent généralement des résultats assez précis, mais ils sont assez lents, c'est pour quoi en 2016 -Wei Liu et al- ont mis au point une nouvelle architecture différente, appelée détecteurs à coup unique (SSD) qui se démarque en étant conçu spécialement pour la détection d'objets en temps réel. Ce modèle saute l'étape de proposition de région et prédit de manière dense les objets directement sur l'image en utilisant à la place un MultiBox Detector et une fonction de perte [Anguelov et al 2016].



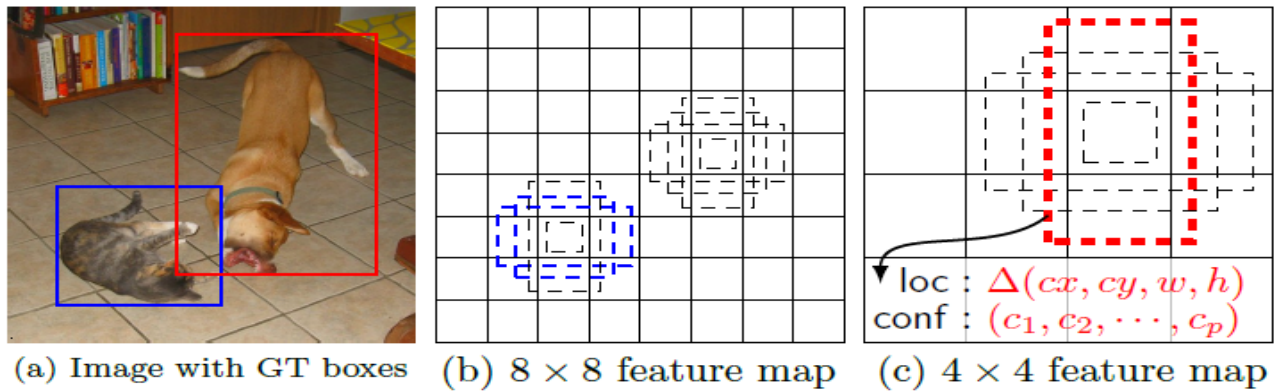


Figure 7 Détecteur SSD Multibox [Anguelov and al 2016]

#### 4.2.3. YOLO (You Only Look Once)

L'abréviation de You Only Look Once, YOLO, est un algorithme de régression qui entre dans la catégorie des méthodes de détection d'objets avec une multitude d'applications de vision par ordinateur. Au lieu de sélectionner la partie intéressante d'une image, cet algorithme prédit les boîtes englobantes « Bounding Box » et les probabilités qu'elles contiennent l'objet recherché pour l'ensemble de l'image en une seule exécution de l'algorithme. Il existe plusieurs versions de YOLO, les plus utilisées actuellement sont yolov3 [Redmon] et yolov4 [Bochkovskiy and al], mais la version la plus récente est yoloV5.

## 5. Le réseau YOLO

### 5.1. Définition

YOLO est un astucieux réseau neuronal convolutif (CNN) permettant de détecter des objets en temps réel en une seule itération (d'où son nom), ce qui va lui permettre de faire de la détection en temps réel sur des vidéos. L'algorithme applique un seul réseau neuronal à l'image complète, puis divise l'image en régions et prédit les « Bounding Box » et les probabilités pour chaque région. Cela signifie essentiellement qu'ils reconnaissent l'emplacement de l'objet, qu'ils utilisent des boîtes englobantes pour montrer où il se trouve et qu'ils utilisent la probabilité de classe pour déterminer ce qu'est l'objet.

Il s'est imposé sur le marché en raison de sa précision, de sa vitesse démontrée et de sa capacité à détecter des objets en une seule exécution, surpassant le Faster R-CNN, et le Single-Shot MultiBox Detector (SSD).

## 5.2. Histoire

Introduit à l'origine par Joseph Redmon dans Darknet, YOLO a parcouru un long chemin. Voici quelques éléments qui ont permis à la première version de YOLO de briser la concurrence sur R-CNN et DPM :

- Traitement des images en temps réel à 45 fps.
- Moins de faux positifs sur l'arrière-plan.
- Précision de détection plus élevée (mais précision moindre en matière de localisation).

L'algorithme n'a cessé d'évoluer depuis sa publication initiale en 2016. Les versions YOLOv2 et YOLOv3 ont toutes deux été écrites par Joseph Redmon. Après YOLOv3, sont arrivés de nouveaux auteurs qui ont ancré leurs propres objectifs dans chaque autre version de YOLO.

YOLOv2[Redmon\_2] : Sortie en 2017, cette version a obtenu une mention honorable à la CVPR<sup>1</sup> 2017 en raison d'améliorations significatives sur les « Anchor boxes » et d'une résolution plus élevée.

YOLOv3[Redmon\_3] : La version 2018 avait un score d'objectivité supplémentaire à la prédiction de « bounding box » et aux connexions aux couches réseau dorsales « backbone network layers ». Elle a également fourni une performance améliorée sur les objets de taille petite en raison de la possibilité d'exécuter des prédictions à trois niveaux de granularité différents.

YOLOv4[Web 5] : La version d'avril de 2020 est devenue le premier article non rédigé par Joseph Redmon. Alexey Bochkovski y a introduit de nouvelles améliorations, notamment l'optimisation pour les calculs parallèles, une meilleure agrégation des caractéristiques, etc.

YOLOv5[Web 6] : Glenn Jocher a continué à apporter de nouvelles améliorations dans sa version de juin 2020, en se concentrant sur l'architecture elle-même.

## 5.3. Comment YOLO fonctionne ?

L'algorithme YOLO utilise les trois techniques suivantes :

- Blocs résiduels : l'image est divisée en plusieurs blocs. Chaque bloc a une dimension de  $S \times S$ .
- « Bounding Box Regression » : C'est le contour qui met en évidence un objet dans une image. Il est constitué des attributs suivants :

---

<sup>1</sup>CVPR: Conference on Computer Vision and Pattern Recognition.

Largeur (bw), Hauteur (bh), Centre de la boîte englobante (bx,by),Confidence<sup>2</sup>( C ) et une classe (par exemple : voiture, bicyclette,...) .

- Intersection sur Union (IOU) : est un phénomène de détection d'objets qui décrit le chevauchement des boîtes. YOLO utilise IOU pour fournir une boîte de sortie qui entoure parfaitement les objets.

Un score de Confiance (C) est calculé de la manière suivante :  $C = \text{Pr}(\text{Object}) * \text{IoU}_{pred}^{truth}$  où  $\text{Pr}(\text{Object})$  est la probabilité de la classe, tandis que  $\text{IoU}_{pred}^{truth}$  est l'intersection sur l'union entre la boîte englobante « bounding box » prédite et la vérité du terrain.

L'image suivante montre comment les trois techniques sont appliquées pour produire les résultats finaux de la détection.

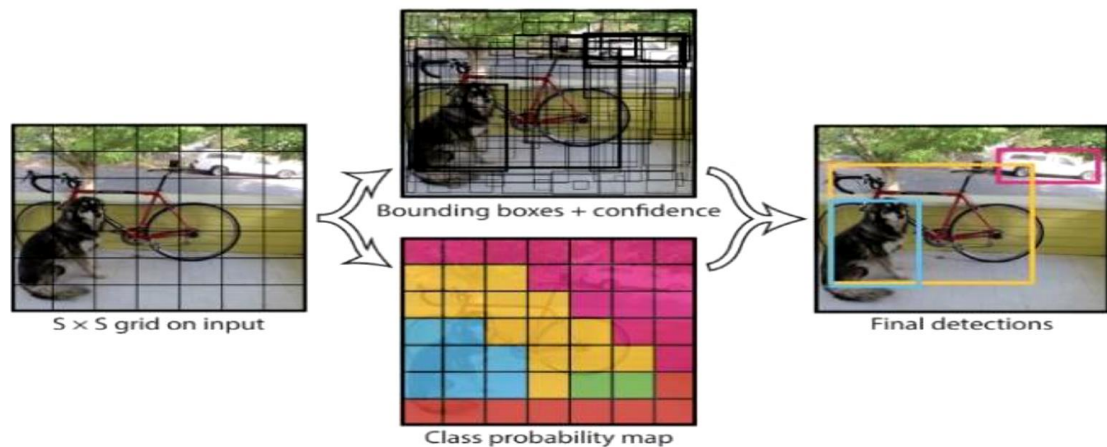


Figure 8 Fonctionnement du réseau Yolo [Redmon et al]

## 5.4. YOLOv5

YOLOv5 est la dernière version en date de YOLO, qui a été développé et publié sur GitHub par Glenn Jocher<sup>3</sup> [Web 7].

Bien qu'elle n'a pas apporté de changements architecturaux majeurs au réseau de YOLOv4, la plus grande contribution de cette version est d'être développée en Python contrairement aux autres version développée en C, cela rend son installation et intégration beaucoup plus facile notamment dans le domaine d'IoT<sup>4</sup>. Etant à présent utilisable sur « Pytorch » l'une des bibliothèques python les plus populaires pour l'apprentissage profond, cela permet une évolution plus rapide du modèle, la communauté Pytorch étant plus active que celle de DarkNet.

<sup>2</sup>Confidence : représente un score de confiance nous indiquant à quel point le détecteur est certain que la « Bounding Box » contient effectivement un objet.

<sup>3</sup> Glenn Jocher : Fondateur et CEO d'Ultralytics.

<sup>4</sup> IoT : Internet Of Things

Étant donné que YOLOv4 et YOLOv5 sont écrits dans deux langues différentes et sur deux infrastructures logicielles (frameworks) différents, il est difficile de comparer les performances de ces deux applications.

Mais après un certain temps, YOLOv5 s'est avéré plus performant que YOLOv4 dans certaines circonstances et a gagné en partie la confiance de la communauté de la vision par ordinateur grâce à la facilité de son implémentation [Do Thuan].

Il existe plusieurs versions de YOLOv5, chacune avec un modèle ayant des avantages et des inconvénients en termes de performance, rapidité d'entraînement et moyenne de précision.

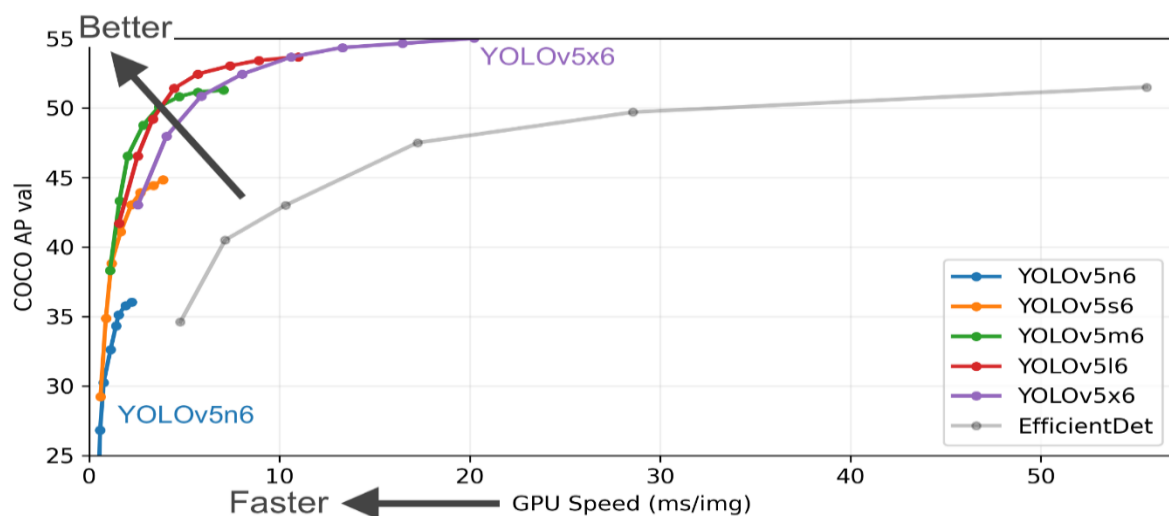


Figure 9 Comparaison de performance des versions YOLOv5 [Web 7]

## 5.5. Architecture du réseau YOLO

La famille de modèles YOLO se consiste de 3 parties principales :

### Backbone: “Focus structure, CSP network”

La dorsale « backbone » est un réseau entièrement convolutif qui aide à extraire les caractéristiques de l'image d'entrée.

### Neck: “Le SPP block, PANet”

Celui-ci utilise le PANet « Path Aggregation Network » pour générer un réseau de pyramides de caractéristiques par des connexions latérales entre les cartes des caractéristiques « Feature maps » afin d'effectuer une agrégation sur les caractéristiques puis les transmettre à la partie Head « Output » pour la prédiction.

## Output: “YOLOv3 head using GIoU-loss”

Cette partie est la partie finale du pipeline de détection d'objets qui prédit la boîte englobante « Bounding Box » et la classification des objets. La tête de YOLOv5 n'est pas différente à celle de YOLOv3 et de YOLOv4.

En appliquant les coordonnées (centre, hauteur, largeur) des « anchors » prédites sur les caractéristiques extraites, le modèle génère des vecteurs avec des probabilités de classe, des « bounding box » et des scores d'objectalité. [Web 6]

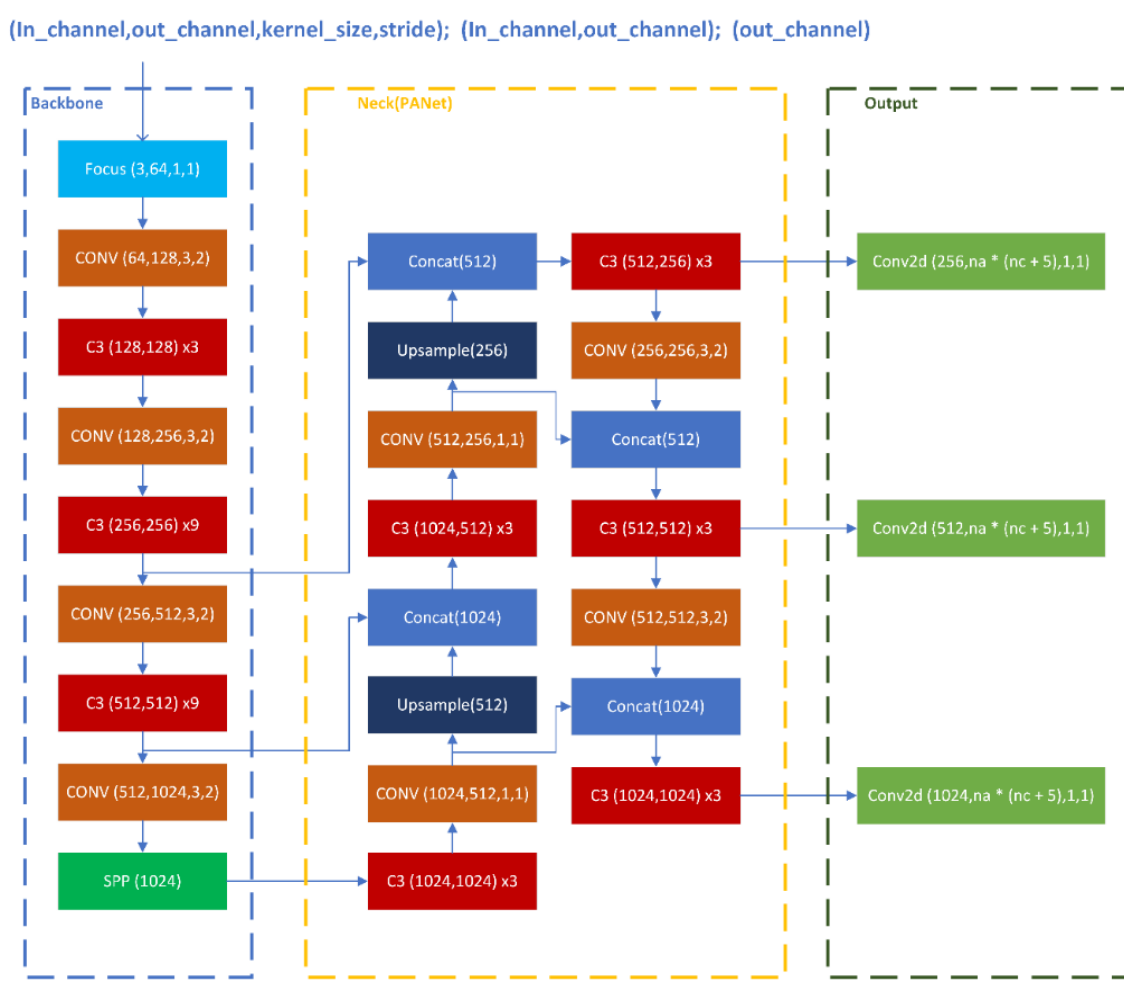


Figure 10 Diagramme de structure du réseau YOLO [Web 8]

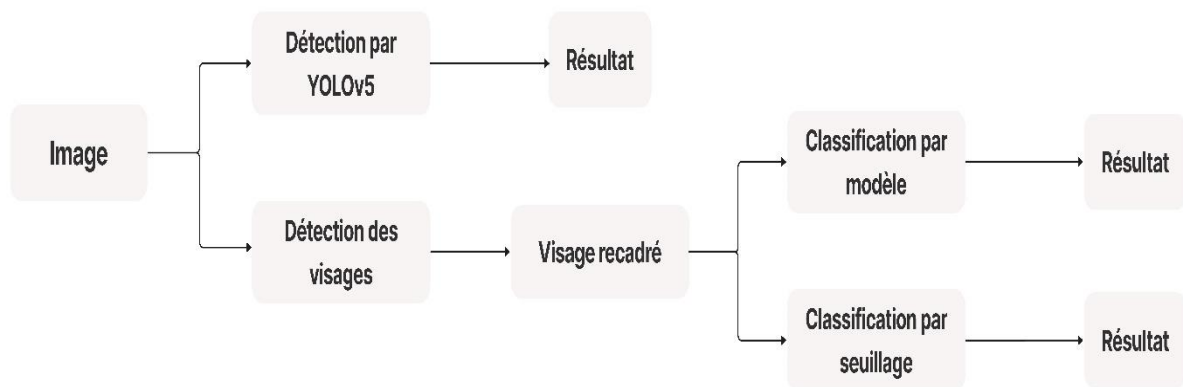
## 6. Conclusion

La crise sanitaire actuelle a contraint de nombreux gouvernements à prendre des mesures préventives strictes dont l'un d'eux est la détection des masques comme le préconisent l'Organisation mondiale de la santé (OMS, 2020) et les études scientifiques menées par (N. Leung et al, 2020), (S. Zhou et al, 2018) et (M. Sande et al, 2008), beaucoup d'études et de recherches sur cette approche ont été fait et reste encore en développement constant.

## Chapitre II: Conception

### 1. Introduction

En raison de la propagation du Corona virus, la détection des masques de visage a reçu beaucoup plus d'attention grâce à sa capacité de réduire le risque potentiel d'exposition d'une personne infectée, qu'elle présente ou non des symptômes. Alors, pour notre projet de fin d'étude, nous proposons deux systèmes capables de faciliter la tâche de la détection de la présence des masques en temps réel et nous allons les comparer par la suite.



*Figure 11 Diagramme résumant les étapes du traitement*

### 2. La détection des visages

La détection de visages dans l'image est un traitement indispensable et crucial avant la phase de détection des masques faciaux. En effet, le processus de détection des masques ne pourra jamais devenir intégralement automatique s'il n'a pas été précédé par une étape de détection efficace.

Après avoir essayé 3 détecteurs différents le premier est le détecteur de visage avec cascade de haars proposé par Viola et Jones [Viola et Jones], puis les modèles légers fournis par MediaPipe et CVzone. Nous avons fini par choisir le modèle du CVzone, qui est principalement construit au-dessus de MediaPipe, comme module de détection des visages pour son efficacité dans différentes conditions. [Web 9]

MediaPipe Face Detection est une solution de détection des visages ultra-rapide basée sur BlazeFast, un détecteur de visage léger et performant conçu pour l'inférence mobile par GPU. Ses performances en temps réel lui permettent d'être appliqué à n'importe quelle expérience de visualisation en direct qui nécessite une région d'intérêt faciale précise comme entrée pour d'autres modèles tels que la classification des caractéristiques ou d'expression faciales.

BlazeFace s'inspire de MobileNetV1/V2, un schéma d'ancrage adapté aux GPU et modifié à partir du détecteur à un seul coup (SSD).

## 2.1. Le détecteur SSD (Single Shot Detection)

Le détecteur à un seul coup est une architecture de convolution qui ne prend qu'un seul coup pour détecter plusieurs objets dans l'image. Il prédit l'objet présent dans l'image et son emplacement plus rapidement et offre une meilleure précision que la plupart des autres algorithmes de détection. L'idée clé du SSD est que, au lieu de propositions de régions, il utilise des boîtes de délimitation, puis il ajuste ces boîtes de délimitation dans le cadre de la prédiction.

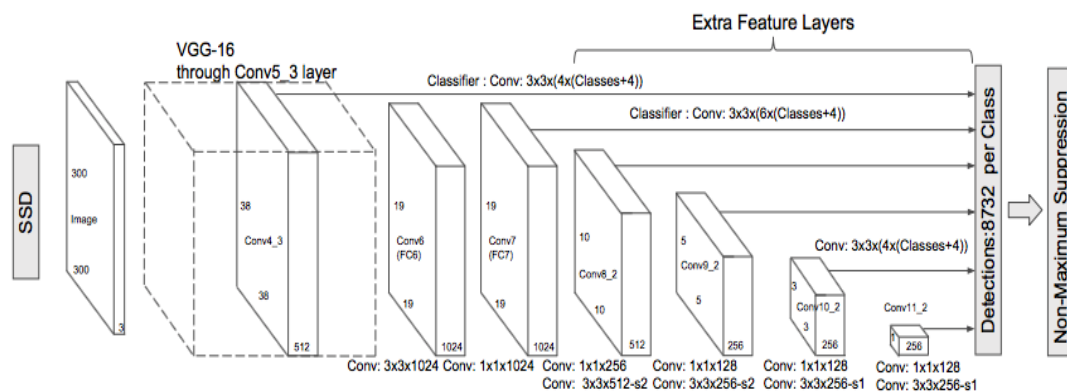


Figure 12 Architecture du réseau SSD [Angulov and al 2016]

Tout d'abord, il faut fournir une image d'entrée et les boîtes de vérité de base pour chaque objet pour l'entraînement. Ensuite, il utilise le backbone (dans le cas de cette figure -le modèle VGG16-) pour extraire la carte des caractéristiques de l'entrée, puis en utilisant les réseaux neuronaux convolutifs (CNNs) il obtient environ 8732 prédictions pour chaque classe ou pour chaque objet. Mais comme il y a beaucoup de boîtes englobantes pour chaque objet, il utilise la suppression non maximale (NMS) afin de pouvoir supprimer les prédictions en double. SSD va maintenant vérifier les valeurs de confiance pour toutes les prédictions et choisir les 200 meilleures valeurs par image.

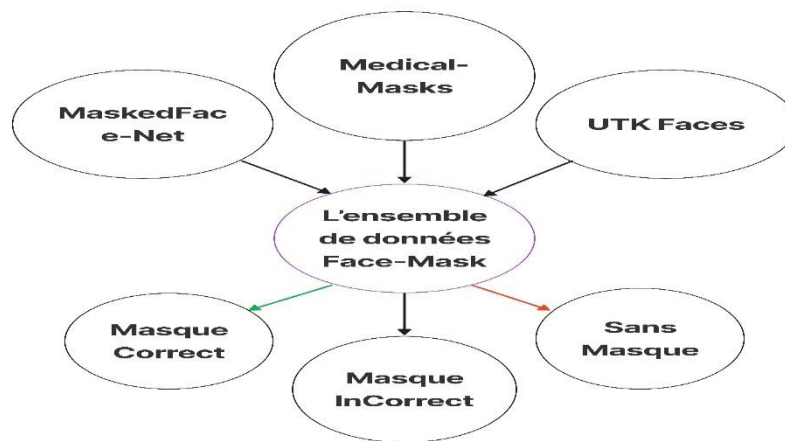
## 3. Collecte des données

L'un des problèmes les plus grand dans les projets de l'apprentissage profond est généralement l'absence de ensembles de données disponibles sur internet. Cette limitation peut affecter la performance des réseaux de neurones qui nécessitent des milliers à des millions d'exemples pour s'entraîner.

### 3.1. Ensemble de données 1

Face-Mask-Dataset est un ensemble de données qui contient +40.000 images appartenant à 3 classes 'Incorrect\_Mask -15193-', 'No\_mask -12799-', 'With\_Mask -12531-'. Les images sont collectées à partir de :

- MaskedFace-Net [Cabani].
- UTK Faces [Susan].
- Medical-masks-Part 7 [Kucev Roman].



*Figure 13 Présentation de la collecte de l'ensemble de données 1*

Pour la première phase de prétraitement des données, nous avons effectué les étapes suivantes :

- Importer les 3 ensembles de données et mettre les images dans leurs classes appropriées.
- Faire un recadrage et un redimensionnement en utilisant un script Python pour une meilleure compatibilité avec notre cas d'utilisation.
- Mettre l'ensemble de données dans GitHub pour qu'on puisse l'utiliser facilement.

Ensuite, nous avons importé le nouvel ensemble de données dans un autre notebook de Google Colab pour compléter son prétraitement, et ces étapes sont présentées par la suite :

- **Fractionnement des données :** Cette étape consiste à diviser l'ensemble de données en trois parties :  
{Ensemble d'entraînement contenant 32418 images (80%), Ensemble de validation avec 6484 images (16%), Ensemble du test avec 1621 images (4%)}.  
➤ **Augmentation de l'ensemble des données :** C'est une technique permettant d'augmenter la diversité et la quantité de l'ensemble d'entraînement pour rendre le modèle plus robuste aux légères variations et améliorer ses résultats en appliquant des



transformations aléatoires (mais réalistes) sur des données déjà existantes ou des données synthétiques nouvellement créées à partir de données existantes, il existe beaucoup de couches pour le faire.

Pour cette dernière, nous avons rajouté deux couches de prétraitement de Keras :

- **RandomRotation** : Cette couche applique des rotations aléatoires à chaque image, et remplit les espaces vides si le fill\_mode est mis à vrai (True).

Avec cette couche, nous avons mis le facteur à 0.2, ce qui entraîne une rotation de la sortie d'une quantité aléatoire dans la plage  $[-20\% * 2\pi, 20\% * 2\pi]$ .

- **RandomContrast** : Cette couche ajustera de manière aléatoire le contraste d'une image ou d'images par un facteur aléatoire. Le contraste est ajusté indépendamment pour chaque canal de chaque image pendant l'entraînement.

Pour chaque canal, cette couche calcule la moyenne des pixels d'images dans le canal, puis ajuste chaque composant x de chaque pixel à :

$$x = (x - mean) * contrast\_factor + mean$$

*Équation 1 La moyenne des pixels par canal*

Avec cette couche, nous avons ajusté le facteur de contraste à 0.4

- **Normalisation de l'ensemble des données** : Pour cette étape nous avons choisi la couche de Keras :

- **Rescaling** : Cette couche redimensionne chaque valeur d'une entrée (image) en multipliant par l'échelle et en ajoutant le décalage. Cette technique permet au modèle de converger plus rapidement.

Pour cela nous avons mis le décalage « offset » à -1 et l'échelle « scale » à 1./175. Pour que les données soient comprises dans l'intervalle  $[-1,1]$ .

### 3.2. Ensemble de données 2

Notre deuxième ensemble de données est un ensemble de données qui contient +10.000 images appartenant à 2 classes 'No\_mask -5000-', 'With\_Mask -4963-'. Les images sont collectées à partir de :

- MaskedFace-Net [Cabani].
- UTKFaces dataset [Susan].
- FaceMask Dataset [Sumansid].

Pour le prétraitement de cet ensemble nous avons réalisé les étapes suivantes :

- Recadrer les images : pour chaque image, nous avons fait l'extraction de tous les visages existants pour obtenir chaque visage seul puis nous avons les redimensionné, ce qui a nous permet d'obtenir 10016 images (visages) en total.
- Fractionner les images obtenues sur deux ensembles :  
 {Ensemble d'entraînement contenant 8013 images (80%), Ensemble de validation avec 2003 images (20%)}
- Faire une augmentation de données :
  - RandomRotation avec un facteur =0.2.
- Faire une normalisation :
  - La couche Rescaling avec les paramètres suivants : (échelle=1./175., décalage=-1)

La figure suivante représente notre ensemble de données après l'application des deux premières étapes de prétraitement.



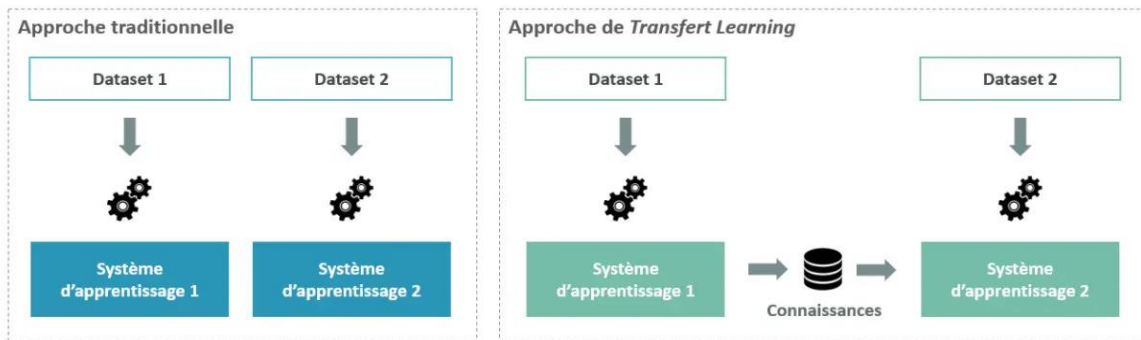
*Figure 14 Aperçu de l'ensemble de données N°2*

## 4. Modèles créés

### 4.1. Modèle n°1 : basé sur les CNNs et l'apprentissage par transfert

#### 4.1.1. Transfer Learning

Le Transfer Learning ou apprentissage par transfert en français, désigne l'ensemble des méthodes qui permettent de ré-exploiter les connaissances acquises à partir de la résolution de problèmes donnés pour traiter un autre problème. Cela est très utile car le modèle n'a pas besoin d'apprendre à partir de zéro et peut atteindre une plus grande précision en moins de temps.



*Figure 15 Approche traditionnelle vs. Approche de Transfert Learning [Web 10]*

Ce type d'apprentissage a connu une forte croissance au cours des dernières années avec l'essor d'apprentissage profond et l'apprentissage automatique et parmi les principales raisons :

- Croissance de la communauté ML et partage des connaissances.
- Les sous-problèmes communs : Cela permet aux gens d'utiliser rapidement des modèles pré-entraînés dans un domaine spécifique.
- Données d'apprentissage supervisé et ressources de formation limitées ce qui pousse les gens à commencer à apprendre à partir de modèles pré-entraînés, et donc d'utiliser les connaissances de domaines similaires.

#### **4.1.2. Techniques d'utilisation de l'apprentissage par transfert**

La technique d'apprentissage par transfert peut être utilisée de la manière suivante : Extraire des caractéristiques des couches utiles.

- Conservez les couches initiales du modèle pré-entraîné et supprimez les couches finales. Ajoutez la nouvelle couche aux couches restantes et entraînez-les pour la classification finale.
- Modifier ou ajuster les paramètres existants dans un réseau pré-entraîné, c'est-à-dire optimiser les paramètres du modèle pendant la formation pour la tâche de prédiction supervisée.

Puisque la création d'un nouveau réseau de neurones convolutif est coûteux en terme d'expertise, de matériel et de quantité de données annotées nécessaires, nous avons décidé d'utiliser un modèle pré-entraîné de Keras.

### 4.1.3. Applications de Keras

Les applications Keras sont des modèles d'apprentissage profond qui sont mis à disposition aux côtés de poids pré-entraînés. Ces modèles peuvent être utilisés pour la prédiction, l'extraction de caractéristiques et le réglage fin « fine-tuning ».

Après avoir tester deux modèles (VGG16 et MobileNetV2) nous avons finis par choisir le MobileNetV2 grâce à son précision de classification surtout en temps réel.

### 4.1.4. MobileNetV2

MobileNetV2 [Web 11] est une amélioration significative par rapport à MobileNetV1 - un modèle dont l'idée principale était de remplacer les convolutions coûteuses par des convolutions moins chères - et fait le point sur l'état de l'art de la reconnaissance visuelle mobile, notamment la classification, la détection d'objets et la segmentation sémantique. Ce modèle est pré-entraîné sur l'ensemble de données ImageNet [Web 12].

Après l'importation du modèle, il faut savoir qu'il est important de spécifier quelques paramètres :

- **Input-shape** : C'est la taille des images en entrées, ce paramètre doit être spécifié lorsque le include-top est désactivé.
- **Include-top** : Ce paramètre nous laisse décider si nous voulons inclure le classifieur du ImageNet au top ou non.
- **Layer.trainable** : Ce paramètre permet de geler les couches « layer freezing » afin d'éviter de détruire les informations qu'ils contiennent lors des prochains cycles d'entraînement.
- **Base\_model.trainable** : Permet de geler le modèle de base pour créer notre propre modèle.
- Afin de réduire la taille de l'entrée selon ses dimensions spatiales (profondeur, hauteur et largeur), nous avons utilisé AveragePooling2D qui prend la valeur moyenne sur une fenêtre d'entrée (de taille définie par pool\_size) pour chaque canal de l'entrée.
- **Dropout** : C'est la couche permettant d'éviter le surajustements « Overfitting » et elle est aléatoirement initialisée à 0. Si la valeur en entrée (d'initialisation) est différente de 0, elle va être mise à l'échelle par  $1/(1 - \text{taux})$  de sorte que la somme de toutes les entrées reste inchangée.

- Dense : La couche dense est une couche qui est connectée en profondeur, ce qui signifie que chaque neurone de la couche dense reçoit des entrées de tous les neurones de la couche précédente.

En arrière-plan, la couche dense effectue une multiplication matrice-vecteur.

$$output = activation(dot(input, kernel) + bias).$$

*Équation 2 Opération faite par la couche Dense*

- Fonction d'activation : Une fonction d'activation est une fonction utilisée dans les réseaux neuronaux artificiels qui produit une petite valeur pour de petites entrées, et une plus grande valeur si ses entrées dépassent un seuil.  
Les fonctions d'activation sont utiles car elles ajoutent des non-linéarités aux réseaux neuronaux, permettant à ces derniers d'apprendre des opérations puissantes.

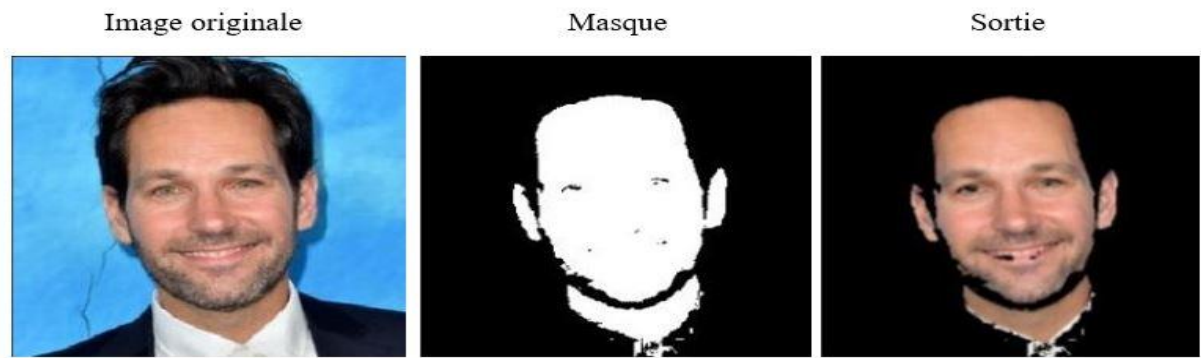
Après la spécification de tous les caractéristiques que nous regroupées toutes les caractéristiques précédentes avec la `tf.keras.Model()`, et pour le finaliser et le rendre enfin prêt pour l'entraînement :

- Fonction d'optimisation (Optimizer) : Ce sont des algorithmes ou des méthodes utilisées pour minimiser une fonction d'erreur (fonction de perte) ou pour maximiser l'efficacité de la production, ils permettent aussi de savoir comment modifier les poids et le taux d'apprentissage du réseau neuronal pour réduire les pertes.
- Fonction de perte (Loss) : L'objectif des fonctions de perte est de calculer la quantité qu'un modèle doit chercher à minimiser pendant l'apprentissage.

## **4.2. Modèle n°2 : basé sur la détection des pixels peau**

### **4.2.1. Détection des pixels peau**

Cet algorithme convertit une image colorée en un masque de peau binaire. Nous utilisons les composantes de couleur pour décider si le pixel actuel appartient à l'espace des couleurs de la peau ou non. Le résultat est une image binaire appelée masque de peau [fig 16].



*Figure 16 Détection des pixels peau*

#### 4.2.2. Résumé de cette approche

Pour chaque image il faut faire comme le suivant :

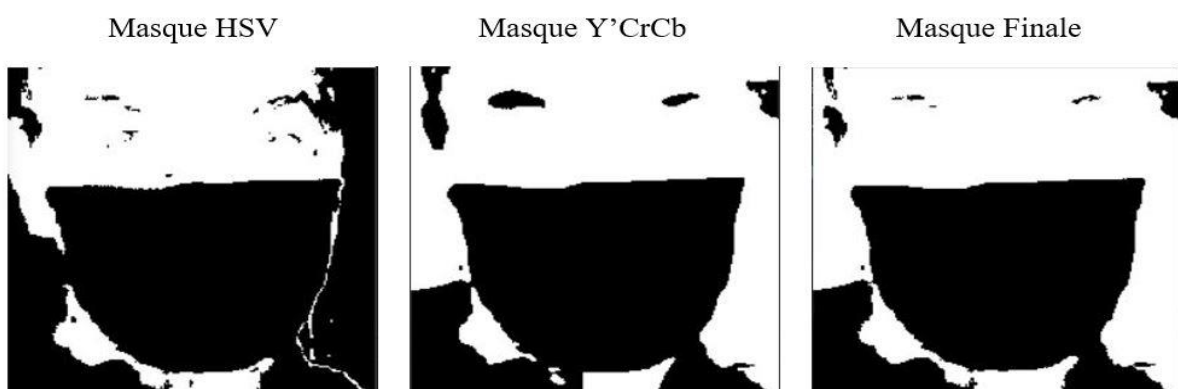
- Conversion entre les espaces de couleurs et extraction du masque :
  - Convertir l'image de BGR vers HSV (la fonction `imread` de OpenCV stocke les images en format BGR).
  - Puis après quelques essais pour trouver l'intervalle des valeurs contenant les masques, nous avons pris les pixels appartenant à l'intervalle spécifié précédemment (le masque en HSV). [fig17]
  - Convertir encore une fois l'image mais cette fois en Y'CrCb et prendre les pixels qui appartiennent au nouvel intervalle spécifié (le masque en Y'CrCb). [fig17]



*Figure 17 Conversion de l'image en HSV et Y'CrCb*

- Combiner les deux masques avec l'opérateur 'Ou' et on sauvegarde le résultat dans une nouvelle variable.
- Appliquer quelques fonctions de traitement pour avoir le masque final :

- Le flou médian « median blur » : Avec cette fonction, l'élément central de l'image sera remplacé par la médiane de tous les pixels de la zone du noyau. Elle traite les bords tout en éliminant le bruit sur chaque masque (le résultat de la combinaison des deux masques HSV et Y'CrCb).
- L'érosion « Eroding » : d'une image signifie que l'image est réduite. Si l'un des pixels d'un noyau est égal à 0, alors tous les pixels du noyau sont mis à 0. Une condition avant d'appliquer une fonction d'érosion sur une image est que l'image soit une image en niveaux de gris.
- La dilatation « Dilation » : C'est tout le contraire de l'érosion, c'est un opérateur de transformation morphologique utilisé pour augmenter la taille ou l'épaisseur de l'objet de premier plan dans une image. Ici, un élément pixel est '1' si au moins un pixel sous le noyau est '1'. Ainsi, la zone blanche de l'image ou la taille de l'objet de premier plan augmente. Dans Normalement, dans des cas comme l'élimination du bruit, l'érosion est suivie d'une dilatation. Car, l'érosion supprime les bruits blancs, mais elle rétrécit aussi notre objet. Alors on le dilate. Puisque le bruit a disparu, ils ne reviendront pas, mais notre surface d'objet augmente. Il est également utile pour joindre des parties brisées d'un objet.
- L'ouverture n'est qu'un autre nom de **l'érosion suivie d'une dilatation**. Il est utile pour supprimer le bruit, comme nous l'avons expliqué ci-dessus. Ici, nous utilisons la fonction `cv2.morphologyEx()`.



*Figure 18 Les différents masques utilisés*

- Appliquer le masque final sur l'image pour avoir l'emplacement de la peau
- Calculer la surface de la peau et divisez-la par la surface totale pour obtenir un pourcentage, les résultats de la classification seront affichés en fonction du pourcentage obtenu.

## **5. Conclusion**

Dans ce chapitre, nous avons expliqué les différentes méthodes et approches utilisées dans la création de nos systèmes ainsi que les étapes suivies dans ce processus. Dans le chapitre suivant, nous allons enfin présenter les systèmes de détection du bon port du masque et comparer les résultats obtenus par chaque modèle.



# Chapitre III: Implémentation

## 1. Introduction

Dans les deux premiers chapitres, nous avons présenté les notions de base nécessaires ainsi que les méthodes utilisées pour la détection des masques faciaux. Ensuite nous avons présenté le processus de la construction de ces systèmes.

Enfin, nous allons présenter dans ce chapitre l'environnement de travail ainsi que les expériences faites et les résultats obtenus.

## 2. Environnement de travail

### 2.1. Matériels

#### Machine 1 :

- *Modèle* : Acer Aspire.
- *Processeur* : Intel ® Core™ i5-4210U CPU @ 1.70GHz\*4.
- *Carte graphique* : Mesa DRI Intel ® HD Graphics 4400 (HSW GT2).
- *Mémoire installée (RAM)* : 8Go.
- *Système d'exploitation* : Linux Mint.
- *Type du système* : x64 bits.

#### Machine 2 :

- *Modèle* : HP ProBook.
- *Processeur* : Intel ® Core™ i5-7200CPU @ 2.50GHz 2.71GHz.
- *Carte graphique* : Intel ® HD Graphics 620.
- *Mémoire installée (RAM)* : 4GO.
- *Edition Windows* : Windows 10 Professionnel.
- *Type du système* : x64 bits.

### 2.2. Logiciels

- **Le langage de programmation** : Python

Pour le développement de nos systèmes, nous avons opté pour le langage de programmation Python.

Python est un langage de programmation généraliste utilisé pour tout type de problème et de développement logiciel, c'est un langage dynamiquement typé, donc il offre aux développeurs

la possibilité de se concentrer sur leurs actions et de gagner plus de temps dans la conception par rapport à un autre langage compilé. En outre, Python est le langage le plus utilisé pour l'apprentissage automatique, la grande majorité des bibliothèques utilisées pour cette disciplines d'analyse de données ont des interfaces Python. Ce qui explique sa popularité en tant qu'interface de commande haut niveau pour les bibliothèques d'Apprentissage Automatique et autres algorithmes numériques.

– **Les librairies :**

**OpenCV (Open Computer Vision) :** Est une bibliothèque graphique libre initialement développement par Intel de vision par ordinateur et d'apprentissage automatique. Elle a été construite pour fournir une infrastructure pour les applications de vision par ordinateur et pour accélérer l'utilisation de la perception artificielle dans les produits commerciaux. La bibliothèque compte plus de 2500 algorithmes optimisés, ce qui inclut un ensemble complet d'algorithmes classiques et de pointe en matière de vision par ordinateur et d'apprentissage automatique. Ces algorithmes peuvent être utilisés pour détecter et reconnaître des visages, identifier des objets...

**Tensorflow :** C'est une bibliothèque libre, permettant d'exécuter des applications d'apprentissage automatique et d'apprentissage profond. Cet outil a été développé par Google, et est fortement utilisé dans le domaine de l'intelligence artificielle (IA). Elle propose une multitude de données et d'outils pour configurer et améliorer le réseau de neurones d'un logiciel ou d'un objet automatisé ainsi qu'un alignement intelligent des données et traitement intégré des données manquantes.

**Numpy :** C'est un paquetage « package » fondamental pour le calcul scientifique en python, elle introduit un objet tableau multidimensionnel, divers objets dérivés (tels que les tableaux masqués et les matrices), ainsi qu'un assortiment de routines permettant d'effectuer des opérations rapides sur les tableaux, notamment des opérations mathématiques et logiques et bien plus encore.

**Matplotlib :** C'est une bibliothèque destinée à créer des visualisations des données sous forme de graphique et des diagrammes de hautes qualités, c'est une alternative libre à MATLAB. Elle est particulièrement utile pour les personnes travaillant avec Python ou Numpy.

**OS :** Est un module fournit par python dont le but d'interagir avec les fonctionnalités du système d'exploitation, il est indépendant par rapport au système d'exploitation ce qui signifie qu'il peut fonctionner sur n'importe quel système.

De plus, le module `os` nous permet de travailler avec les fichiers et les répertoires.

**Glob :** Le module `glob` (Abréviation de Global) de python possède plusieurs fonctions qui peuvent aider à lister les fichiers sous un fichier spécifique. Nous pouvons utiliser `glob` pour rechercher un modèle de fichier spécifique, ou peut-être plus utilement, rechercher des fichiers dont le nom correspond à un certain modèle en utilisant des caractères génériques.

**Pathlib :** Le module `pathlib` est un module doté d'une interface orientée objet inclus dans python depuis la version 3.4 doté de méthodes très intuitives permettant d'interagir avec le système de fichiers d'une façon simple et conviviale. Il offre des fonctions similaires au module `OS`.

**Keras :** C'est une API de haut niveau écrite en Python, elle permet d'interagir avec les algorithmes de réseaux neurones profonds et d'apprentissage automatique, elle est interfaçable avec Tensorflow, CNTK et Theano. Elle contient de nombreuses implémentations de blocs de construction de réseaux neuronaux couramment utilisés, tels que les couches, les fonctions d'activation etc...

#### – Les environnements :

**VScode :** Un éditeur de texte multiplateforme qui fournit du support et de l'assistance pour de nombreux langages différents tels que : Python, Html, JavaScript ...

**Google Colab :** Google Colab ou Google Colaboratory est un outil d'analyse des données et d'apprentissage automatique qui nous permet de combiner du code python exécutable et du texte riche avec des graphiques, des images du LaTeX et bien d'autres choses encore dans un seul et même document sur Google Drive.

Il se connecte aux puissants moteurs d'exécution de Google Cloud Platform et vous permet de partager facilement votre travail et de collaborer avec d'autres personnes. En plus il donne accès gratuit aux GPU et TPU.

### 3. Expérimentations

#### 3.1. Architecture du CNN :

##### 3.1.1. Expérience N°1 :

Pour cette expérience, nous avons pris l'ensemble de données n°2 (+10000 images et contenant deux classes seulement « Sans masque et avec masque »), et nous avons spécifié les arguments suivant avant de commencer l'entraînement :

1. Pour pouvoir ré-entraîner le modèle avec ce qui convient avec notre cas d'étude, nous avons initialiser le `include_top` à `False` (pas de réglage fin)
2. Comme nous avons désactivé le réglage fin, il faut spécifier la taille des images en entrée, donc nous avons mis `input_shape` à `(224,224,3)` -c'est la taille d'entrée par default pour le modèle `MobileNetV2`-
3. Nous avons ensuite appliqué le `AveragePooling2D` sur notre entrée.
4. Pour le paramètre `Dense`, nous l'avons mis à 1 car nous avons voulu détecter le bon port du masque en utilisant juste deux classes puis nous avons ajouté une fonction d'activation qui est `sigmoid` notant que sa formule est :

$$S(x) = \frac{1}{1 + e^{-x}}$$

*Équation 3 Formule de la fonction Sigmoid*

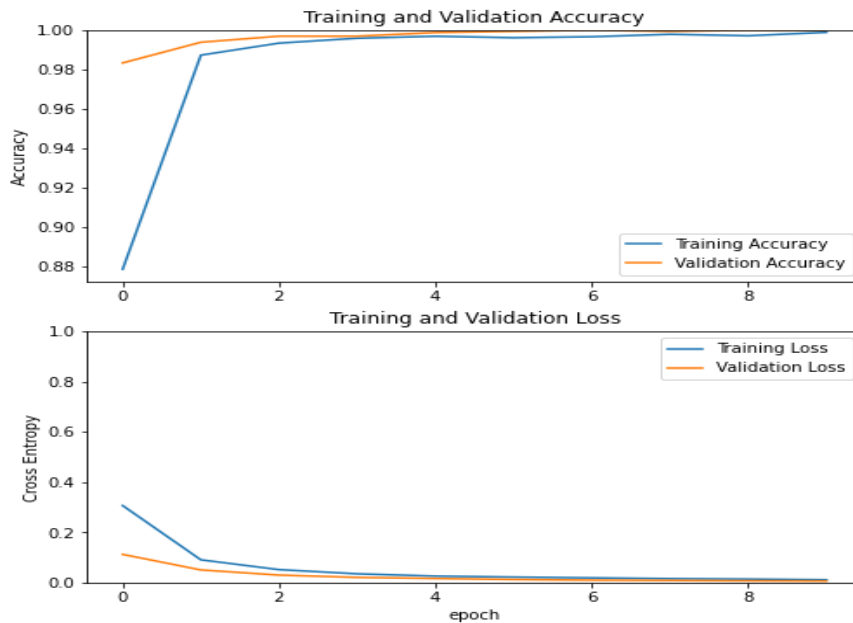
5. Comme fonction de perte, nous avons choisi le `BinaryCrossentropy` dont la fomule est la Suivante :

$$BCE = -\frac{1}{N} \sum_{i=1}^N (y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i))$$

*Équation 4 Formule de la fonction BinaryCrossentropy*

6. Et finalement nous avons choisi le `RMSprop` comme un fonction d'optimisation.

En résultat, nous avons constaté que le modèle converge rapidement et nous avons obtenu 99% de précision, mais après avoir le tester avec l'ensemble de test nous avons obtenu de très mauvais résultats et beaucoup de surajustements.



*Figure 19 Métriques de l'évaluation sans réglage fin (Fine Tuning)*

### 3.1.2. Expérience N°2

Pour la deuxième expérience nous avons gardé les mêmes paramètres spécifiés dans la première expérience mais nous avons enlevé le AveragePooling2D () et ajouté un dropout avec un taux égale à 0.2

Après 40 minutes d'entraînement, La précision du modèle a diminué de 99% (le modèle précédent) à 76% en 20 epochs et il surajuste « overfits » toujours.

### 3.1.3. Expérience N°3

Pour cette expérience, nous avons décidé d'essayer avec le réglage fin « fine tuning » alors nous avons mis le base\_model.trainable à True puis nous avons gelé les 100 premières couches et laissé 54 pour faire l'entraînement.

Cette fois ci, nous avons utilisé le AveragePooling2D () et initialisé le Dropout à 0.2.

Pour la fonction de perte, nous avons gardé le BinaryCrossentropy mais nous avons changé la fonction d'optimisation de RMSprop à Adam

Avec cette expérience, nous avons constaté qu'après 30 epochs la précision est très élevée et le modèle ne surajuste pas mais la performance est très mauvaise en temps réel.

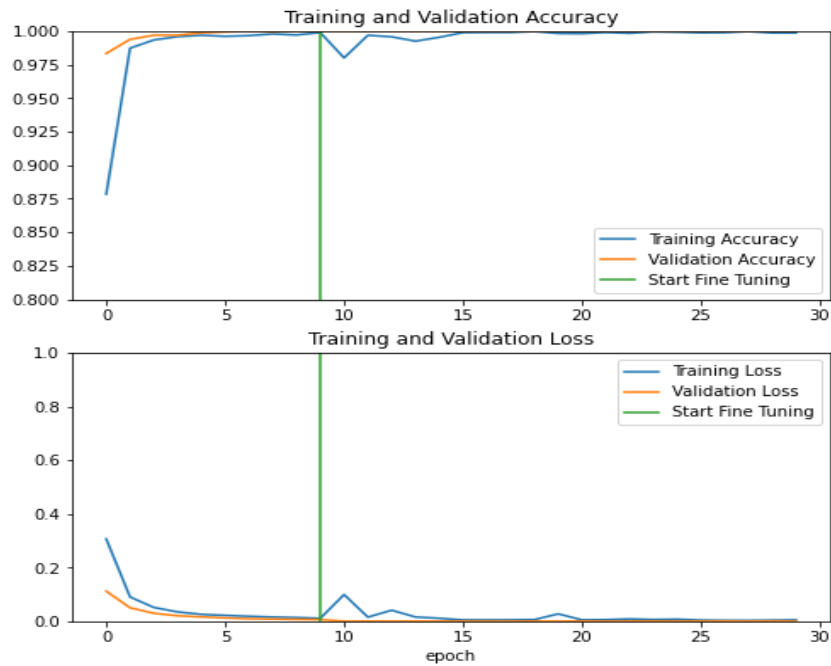


Figure 20 Métriques de l'évaluation avec le réglage fin (Fine Tuning)

### 3.1.4. Expérience N°4

Pour la dernière expérience avec l'apprentissage par transfert, nous avons décidé d'utiliser un nouvel ensemble de données (ensemble de données 1).

Pour le dropout, il est toujours initialisé à 0.2, par ailleurs, nous avons utilisé le dense avec la valeur 3 mais sans spécifié la fonction d'activation (output=a(input))

Pour la fonction de perte, cette fois nous avons décidé d'utiliser le Sparse Categorical Crossentropy et Adam comme une fonction d'optimisation

```

Pseudo-algorithme De Détection du masque en utilisant MobileNetV2

ENTRÉE : image
SORTIE : prédiction écrit sur l'image
DEBUT
Charger le modèle
classes = ['Masque Incorrect', 'Sans Masque', 'Masque Correct']
POUR chaque image de source :
FAIRE
    lire(image);
    Coordonnées_visages[ ] = Détecter_Visage(image);
    POUR chaque Coor_visage de Coordonnées_visages[ ] :
        FAIRE
            image_recadrée = Recadrer l'image avec les dimensions Coor_visage;
            prédiction[] = modèle.classifier(image);
            écrire sur image(classes[index_du_max(prédiction)]);
    FAIT;
FAIT;
FIN
  
```

Figure 21 Pseudo-algorithme de détection des masques\_1

En entrainant les 54 couches avec plus de 40000 images nous avons finalement pu avoir un modèle avec une grande précision (0,99) et qui est performant en temps réel.

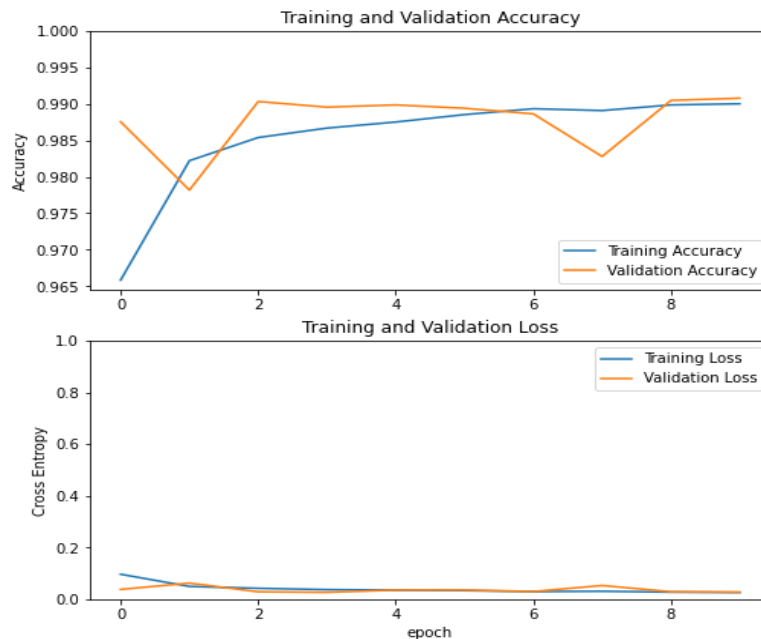


Figure 22 Métriques de l'évaluation avec l'ensemble de données\_2

### 3.2. Détection du masque en se basant sur les pixels peau

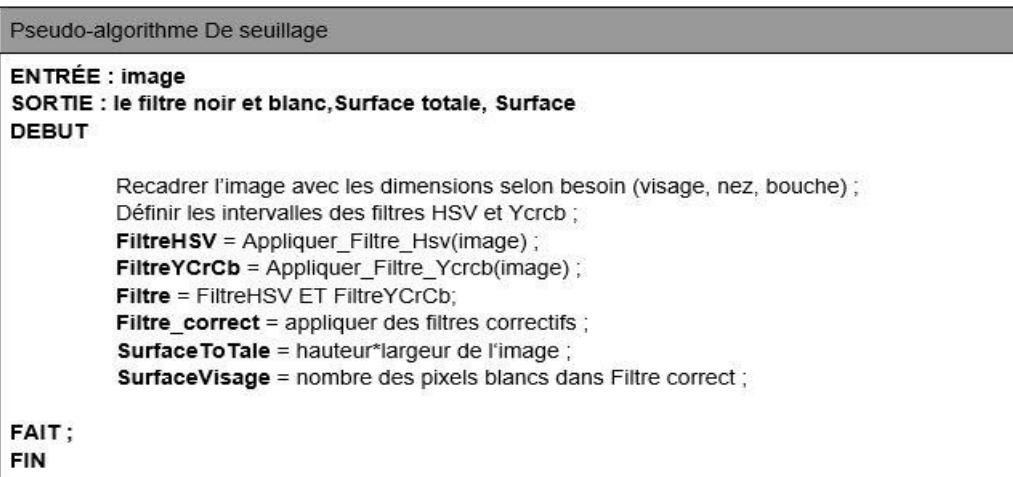
```

Pseudo-algorithme De Détection du masque en utilisant les pixel peau

DEBUT

POUR chaque image de source :
FAIRE:
    lire(image);
    Coordonnées_visages[ ] = Détecter_Visage(image);
    POUR chaque Coor_visage de Coordonnées_visages[ ] :
        FAIRE
            image_recadrée = Recadrer l'image avec les dimensions Coor_visage;
            mask, SurfaceTotal, SurfaceVisage = masque appliqué sur
            image_recadrée;
            SI SurfaceVisage/SurfaceTotal < 0.15 ALORS:
                Masque_Correct;
            SINON
                SI SurfaceVisage/SurfaceTotal > 0.65 ALORS:
                    Sans_Masque;
                SINON
                    mask, SurfaceTotal, SurfaceVisage = masque appliqué
                    sur image_recadrée sur le nez;
                    SI SurfaceVisage/SurfaceTotal < 0.3 ALORS:
                        Masque_Incorrect;
                    SINON
                        mask, SurfaceTotal, SurfaceVisage = masque
                        appliqué sur image_recadrée sur la bouche;
                        SI SurfaceVisage/SurfaceTotal > 0.3 ALORS:
                            Masque_Incorrect;
                        SINON
                            Masque_Correct;
                    FSI;
                FSI;
            FSI;
        FSI;
    FAIT;
FAIT;
FIN
    
```

Figure 23 Pseudo-algorithme de détection des masques\_2



*Figure 24 Pseudo-algorithme de seuillage*

### 3.3. Le réseau YOLOv5

En ce qui concerne ce modèle, nous avons suivi les étapes suivantes :

- Cloner le modèle YOLOv5.
- Importer l'ensemble des données
- Exécuter la commande suivante : `python3 detect.py --weights best.pt --source 0 --conf 0.6`
  - **detect.py** : effectue des inférences sur une variété de sources, télécharge automatiquement des modèles à partir de la dernière version YOLOv5 et sauvegarde les résultats dans run/detect.
  - **weights** : Le chemin vers le fichier des poids après l'entraînement
  - **source** : Le chemin vers le dossier contenant l'ensemble des données à tester (ici source 0 veut dire le Webcam).
  - **Conf** : La limite inférieure de la confiance (probabilité de la classe) pour afficher la classe
- Entraîner le modèle en exécutant la commande : `!python train.py --img 416 --batch 16 --epochs 100 --data{dataset.location}/data.yaml --cfg ./models/custom_yolov5s.yaml --weights " --name yolov5s_results --cache`
  - **train.py** : entraîne le modèle et sauvegarde les résultats dans run/train/ .
  - **img** : elle représente la taille des images en entrée.
  - **batch** : nombre de lots utilisé.
  - **epochs** : spécifie le nombre d'epochs .
  - **cfg,weights** : importer les poids pré-entraînés.



Ce qui suit sont des tests réalisés en temps réel et sur des images brutes :









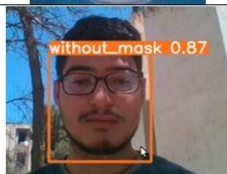
	Masque Correct	Masque Incorrect	Sans Masque
<b>MobileNetV2</b>			
<b>En détectant les pixels peau</b>			
<b>YOLOv5</b>			

Figure 25 Tests en temps réel

La figure ci-dessus montre les résultats obtenus en testant les trois modèles en temps réel, nous apercevons que le visage est déterminé par un cadre avec l'état du port du masque qui s'affiche en haut.













Variation de l'arrière-plan								
N°1		MobileNetV2 : Masque Incorrect <u>Détection Pixels</u> Peau : Masque Incorrect YOLOv5 : Masque Incorrect	N°2		MobileNetV2 : Sans Masque <u>Détection Pixels</u> Peau : Masque Correct YOLOv5 : Masque Correct			
N°3		MobileNetV2 : Masque Correct <u>Détection Pixels</u> Peau : Masque Correct YOLOv5 : Sans Masque						
Variation des masques								
N°4		MobileNetV2 : Sans Masque <u>Détection Pixels</u> Peau : Masque Incorrect YOLOv5 : Masque Correct		MobileNetV2 : Masque Correct <u>Détection Pixels</u> Peau : Sans Masque YOLOv5 : Sans Masque	N°6		MobileNetV2 : Sans Masque <u>Détection Pixels</u> Peau : Masque Incorrect YOLOv5 : Sans Masque	
Variation de peau								
N°5		MobileNetV2 : Masque Incorrect <u>Détection Pixels</u> Peau : Masque Incorrect YOLOv5 : Masque Incorrect	N°8		MobileNetV2 : Sans Masque <u>Détection Pixels</u> Peau : Masque Correct YOLOv5 : 0 Détections	N°9		MobileNetV2 : Masque Correct <u>Détection Pixels</u> Peau : Masque Incorrect YOLOv5 : Sans Masque
N°7			N°8			N°9		

Figure 26 Tests réalisés avec les trois approches

La figure ci-dessus montre les résultats obtenus en appliquant les trois approches sur des images brutes dans des différentes conditions.

## 4. Visualisation des résultats

### 4.1. La courbe ROC et la matrice de confusion

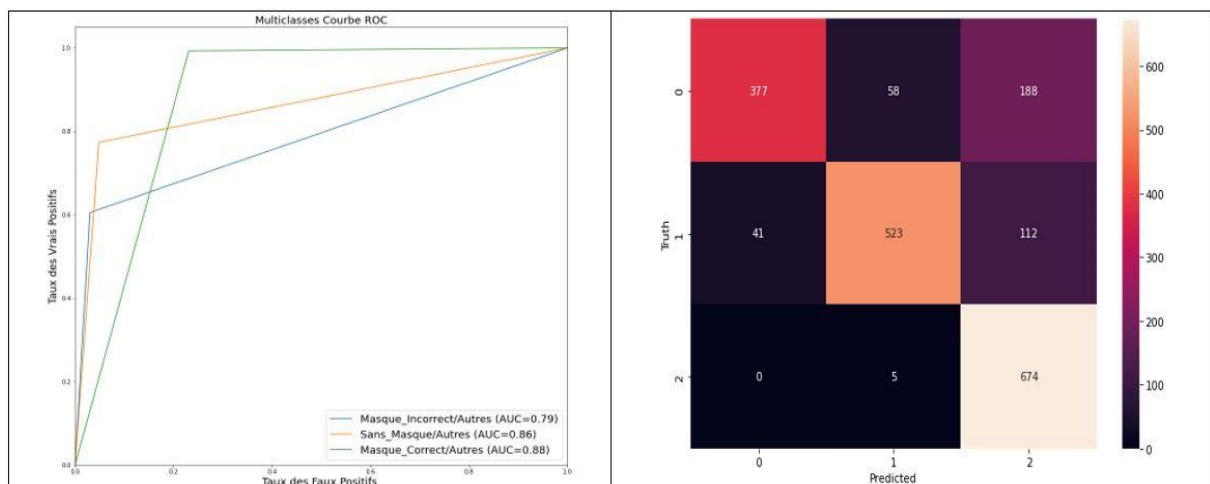
Pour cette étape, nous avons construit les courbes ROC pour trois aspects, pour les trois modèles de détections :

- Masque incorrectement porté/ (Masque correctement porté + Sans masque)
- Masque correctement porté/ (Masque porté incorrectement + Sans masque)
- Sans masque / (Masque porté correctement + Masque porté correctement)

Ensuite nous avons calculé l'indice AUC pour chaque courbe afin de déterminé la performance des systèmes.

Les paramètres utilisées pour tracer ces courbes sont les taux des vrais positifs et les taux des faux positifs, le vrai positif est le scénario où l'image est bien classée, le faux positif est le scénario où le modèle ne détecte pas la vraie classe de l'image. Ci-dessous les résultats obtenus pour chaque classe :

- **Le modèle MobileNetV2 :**



*Figure 27 Courbe ROC et matrice de confusion –Modèle1*

Comme on peut le voir, nous aurons un AUC égale à 0,88 pour les masques correctement portés et les visages non masqué car ces deux catégories sont bien présentées et un AUC égale à 0.79 pour les masques incorrects ce qui montre que le modèle est performant (93% de précision sur les images présentées dans la [fig.26] ), mais cela peut ne pas toujours fonctionner car nous avons travaillé avec un ensemble de donnée dont les images sont de piètre qualité et qui ne contient pas de diversité dans l'ethnicité humaine ( La plupart des images représentent des personnes asiatiques).

- **La méthode de détection des pixels peau :**

On constate que l'indice AUC a une valeur  $\approx 0,75$  pour les cas sans masque et masque incorrect à cause de plusieurs raisons : La non-diversité des images de l'ensemble des données (Images des Indiens) ce qui rend le modèle moins performant avec d'autres couleurs de peau le cas de l'image n°5 [fig.26], ensuite, notre modèle utilise un filtre (masque) statique avec des intervalles numériques, donc si l'image en entrée est avec un arrière-plan qui ressemble à la couleur de peau ou à une luminosité très haute le modèle ne sera pas capable de détecter la bonne classe car l'image est loin du seuil dont nous avons précisé. Le même problème figure lorsqu'une personne qui porte le masque tourne son visage, car la peau de la joue apparaît et donc le modèle trouve une difficulté à décider si cette personne porte le masque ou non.

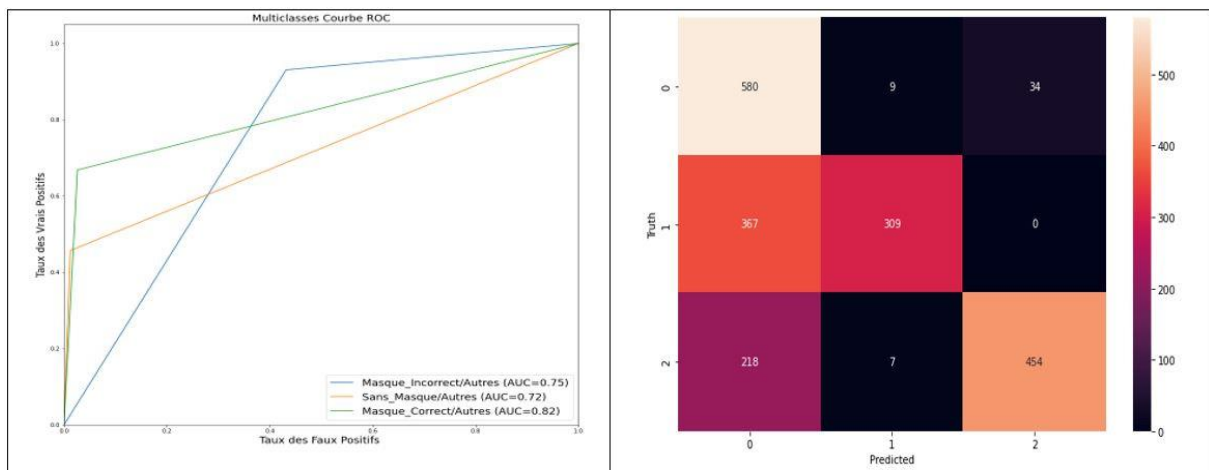


Figure 28 Courbe ROC et matrice de confusion -Modèle2

- **Le modèle YOLOv5 :**

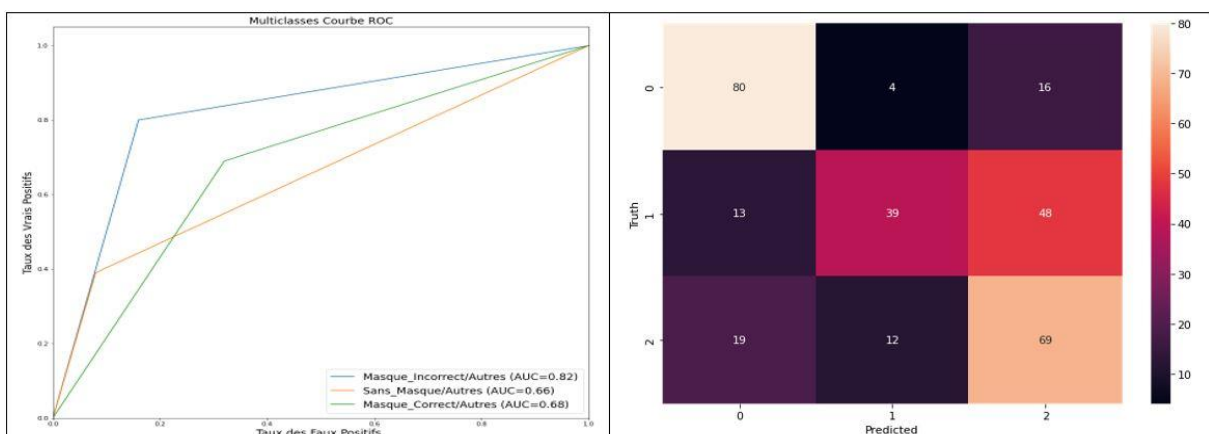


Figure 29 Courbe ROC et matrice de confusion -YOLOv5

En utilisant le modèle YOLO, nous aurons un AUC élevé pour la détection des masques incorrects égale à 0,82 car cette classe est bien présentée dans notre ensemble de données et

une valeur  $\approx 0,67$  pour les masques corrects et les visages sans masque étant donné que ces deux classes ne contenant pas beaucoup d'images et comme l'ensemble de données utilisé est si petit, ce modèle n'est pas capable de détecter beaucoup de cas comme vous le voyez dans l'image n°4 [fig.26] . En regardant la matrice de confusion, on constate que les vrais positifs sont élevés pour les trois catégories.

## **4.2. Discussion**

La tâche de détection du bon port masques faciaux a été réalisée avec images variées. Dans tous ceux qui précèdent, nous avons constaté que les résultats se changent d'un modèle à un autre.

D'après la courbe du ROC et la matrice de confusion, on observe que le modèle MobileNetV2 donne des bonne résultats ( un AUC  $\approx 0,8-0,9$ ) dans tous les cas du masque ce qui est bien visible dans les résultats de la [fig.26]. D'autre part, le modèle basé sur la détection des pixels peau ne semble pas qu'il est très performant et on peut constater ce problème dans la [fig.26] où 26% des résultats obtenus sont justes. En revanche, avec le modèle YOLOv5, il est clair que ce modèle performe avec les visages incorrectement maqués qu'avec les autres cas, mais en regardant la [fig.26] on peut dire que sa performance ne dépasse pas 15%.

En interprétant ces résultats, on conclut que les performances des trois modèles sont bonnes mais en termes d'efficacité et de bonne détection, notre modèle MobileNetV2 est meilleur que le modèle YOLOv5 et le modèle basé sur la détection des pixels peau.

## **5. Conclusion**

Dans ce chapitre, nous avons d'abord présenter les outils nécessaires pour la construction de notre modèle. Ensuite, nous avons présenté les expériences faites lors de la réalisation. Enfin nous avons mis en œuvre un modèle capable de détecter la présence des masques de visage et leurs états du port, les résultats obtenus sont très prometteurs et ont été validés par de l'expérimentation.

En se basant sur les résultats obtenus dans ce chapitre, nous pouvons finalement constater que notre système -MobileNetV2- basé sur les réseaux neuronaux est prêt pour la production pour des cas d'utilisation spécifiques, et ce parce que sa précision est en concurrence avec d'autres modèles comme les modèles de test que nous avons vus tout au long du projet.

Ces systèmes peuvent être particulièrement utiles à des fins de sécurité pour vérifier si la transmission des maladies est maîtrisée, notamment pour les enfants et les personnes âgées.

## Conclusion générale

Dans le cadre de ce travail, nous nous sommes intéressés au développement d'un système de détection du bon port des masques faciaux.

Nous avons d'abord présenté une étude de l'état de l'art où nous avons analysé les différents travaux et recherches réalisées. Ensuite, nous avons présenté les différentes approches de détection d'objets en utilisant l'apprentissage profond. Puis en se basant sur cette étude, nous avons essayées des approches différentes en lui apportant des modifications afin qu'elles soient adaptées à nos besoins et pour avoir nos propres détecteurs des masques.

En effet, nos détecteurs sont capables de localiser les visages humains puis détecter l'état du port des masques, si le masque existe ou non et est-ce qu'il est bien porté ou non.

Vu qu'un système parfait n'existe pas lorsqu'il s'agit d'un domaine de recherche, plusieurs améliorations et perspectives sont envisageables malgré l'efficacité que fournit nos systèmes et la satisfaction de nos résultats obtenus.

Nos systèmes ont été réalisés sur un ensemble de donnée avec +40000 images, ce qui représente une petite catégorie de la population et donc beaucoup de variétés n'ont été pas prises en considérations. Aussi pour des prochaines recherches, il serait avantageux de s'intéresser à une plus large catégorie de personnes avec des masques bien ou mal portées et même des personnes sans masque dans des conditions différentes pour avoir des meilleurs résultats.

## 1. Bibliographie

[Shaik et Ahlam] : S. A. Hussain, A.S.A.A. Balushi, A real time face emotion classification and recognition using deep learning model, J. Phys.: Conf. Ser. 1432 (2020) 012087.

[Ejaz et al] : Ejaz MS, Islam MR, Sifatullah M, Sarker A (2019) Implementation of principal component analysis on masked and non-masked face recognition. In: 2019 1St international conference on advances in science, engineering and robotics technology (ICASERT). IEEE, pp 1–5.

[Day et al] : S.K. Dey, A. Howlader, C. Deb, MobileNet Mask: A Multi-phase Face Mask Detection Model to Prevent Person-To-Person Transmission of SARS-CoV-2, in: Proceedings of International Conference on Trends in Computational and Cognitive Engineering. 2021. Springer

[Bhuiyan et al] : M.R. Bhuiyan, S.A. Khushbu, M.S. Islam, A Deep Learning Based Assistive System to Classify COVID-19 Face Mask for Human Safety with YOLOv3, in: 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT). 2020. IEEE.

[Addagarla et al] : S.K. Addagarla, G.K. Chakravarthi, P. Anitha, Real Time Multi-Scale Facial Mask Detection and Classification Using Deep Transfer Learning Techniques, Int. J. Adv. Trends Comput. Sci. Eng. 9(4) (2020).

[Qin et al]: B. QIN and D. Li, Identifying facemask-wearing condition using image superresolution with classification network to prevent COVID-19, May 2020.

[Viola et Jones] : Paul Viola and Michael Jones : Boosting: Foundations and Algorithms. Rapid object detection using a boosted cascade of simple features. In Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001, volume 1, pages I{I. IEEE, 2001.

[Girshick et al 2013] : R\_CNN: Rich feature hierarchies for accurate object detection and semantic segmentation Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik 22 Oct 2014

[Angulov et al. 2016] SSD : Single Shot MultiBox Detector Wei Liu<sup>1</sup> , Dragomir Anguelov<sup>2</sup> , Dumitru Erhan<sup>3</sup> , Christian Szegedy<sup>3</sup> , Scott Reed<sup>4</sup> , Cheng-Yang Fu<sup>1</sup> , Alexander

C. Berg<sup>1</sup> <sup>1</sup>UNC Chapel Hill <sup>2</sup>Zoox Inc. <sup>3</sup>Google Inc. <sup>4</sup>University of Michigan, Ann-Arbor 29  
40 Decembre 2016.

[Redmon] YOLO9000: Better, Faster, Stronger Joseph Redmon Ali Farhadi 25 Decembre 2016

[Redmon\_3] YOLOv3 : An Incremental Improvement Joseph Redmon Ali Farhadi 8 Avril  
2018.

[Bochkovski and al] YOLOv4: Optimal Speed and Accuracy of Object Detection Alexey  
Bochkovski Chien-Yao Wang Hong-Yuan Mark Liao 23 Avril 2020.

[Redmon and al] You Only Look Once: Unified, Real-Time Object Detection Joseph Redmon<sup>1</sup>,  
Santosh Divvala<sup>2</sup>, Ross Girshick, Ali Farhadi<sup>3</sup> University of Washington, Allen Institute for  
AIy, Facebook AI Research.

[Do Thuan] Evolution of YOLO Algorithm and YOLOv5: The state-of-the-Art Object  
Detection Algorithm. Do Thuan. Oulu University of Applied Sciences.

[Cabani] MaskedFace-Net Dataset: <https://github.com/cabani/MaskedFace-Net>

[Susan] UTK Faces Dataset : <https://susanqq.github.io/UTKFace/>

[Kucev Roman] Medical-masks part7 Dataset:

<https://www.kaggle.com/datasets/tapakah68/medical-masks-part7>

[Sumansid] FaceMask Dataset : <https://www.kaggle.com/datasets/sumansid/facemask-dataset>

## 2. Webographie

- [Web1] : <https://www.sap.com/insights/what-is-machine-learning.html>
- [Web 2] : <https://intellipaat.com/community/9868/what-is-the-difference-between-deep-learning-and-traditional-artificial-neural-network-machine-learning>
- [ Web 3] RGB : <https://www.dynamsoft.com/blog/insights/image-processing/image-processing-101-color-models/>
- [Web 4] HSV : <https://www.parthshandilya.com/color-in-information-visualization/>
- [Web 5] YOLOv4 : <https://www.v7labs.com/blog/yolo-object-detection>
- [Web 6] YOLOv5 : <https://pyimagesearch.com/2022/04/04/introduction-to-the-yolo-family/>
- [Web 7] YOLOv5: <https://github.com/ultralytics/yolov5>
- [Web 8] <https://blog.csdn.net/Q1u1NG/article/details/107511465>
- [Web 9] Mediapipe: <https://google.github.io/mediapipe/>
- [Web 10] Apprentissage par transfert: <https://datascientest.com/transfer-learning>
- [Web 11] MobileNetV2: <https://keras.io/api/applications/mobilenet/>
- [Web 12] ImageNet: <https://www.image-net.org/>
- [Web 13] Python : <https://www.python.org/>
- [Web 14] OpenCV : <https://opencv.org/>
- [Web 16] TensorFlow : <https://www.tensorflow.org/>
- [Web 17] Numpy : <https://numpy.org/>
- [Web 18] Matplotlib : <https://matplotlib.org/>
- [Web 19] OS : <https://docs.python.org/3/library/os.html>
- [Web 20] Glob : <https://docs.python.org/3/library/glob.html>
- [Web 21] Pathlib : <https://docs.python.org/3.9/library/pathlib.html>
- [Web 22] Keras : <https://keras.io/>
- [Web 23] VScode : <https://code.visualstudio.com/>
- [Web 24] Google Colab : <https://colab.research.google.com/?hl=fr>



### **3. Résumé**

Le présent projet de licence est une continuité d'un projet de licence soutenu l'année passée qui a proposé la détection du bon port du masque dans une séquence vidéo.

Le but de notre projet de fin d'étude est de réaliser un système de détection du bon port du masque en temps réel en utilisant une approche basée sur la détection des pixels peau puis le comparer avec deux autres systèmes : le premier est basé sur les réseaux neuronaux et l'autre il se base sur le réseau YOLOv5.

Mots-clés : Détection d'objets, Détection des masques faciaux, CNN, YOLOv5, Pixels Peau, Apprentissage automatique, Apprentissage profond.