الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي والبحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Département : Génie électrique Et Informatique Industrielle

Mémoire de fin d'étude en vue de l'obtention du diplôme
D'Ingéniorat en Génie électrique Et Informatique Industrielle

Spécialité
Systèmes de Télécommunications et Réseaux

- Thème -

Conception et réalisation d'un système pour gestion des personnes : Application COVID 19

Réalisé par :
Djamel LAKHDARI
Abderrahim MAHI

Les membres de Jury :

Mme BOUTERFAS Malika M.C.A	Président
M. KHOUMERI El-Hadi M.C.B	Promoteur
Mme BOUTOUTA Dalila M.A.A	Examinateur
Mme HABANI Lamia M.A.A	Examinateur

Alger, le 30/06/2022

Année universitaire 2021 –2022

Conception et réalisation d'un système pour gestion des personnes : Application COVID 19

Remerciements :

Nous souhaitons adresser nos remerciements les plus sincères aux personnes qui nous ont apporté leur aide et qui ont contribué à la réalisation de ce mémoire.

Nous tenons à remercier en premier, Monsieur KHOUMERI, notre encadrant, pour sa généreuse disponibilité, son aide précieuse ainsi que ses conseils et orientations qui nous ont été très utiles.

On remercie également Les membres de jury : Mme. BOUTOUTA, Mme. BOUTERFAS et Mme. HABANI, pour L'intérêt qu'ils ont porté à mon travail.

Nous tenons à remercier, tout particulièrement, et à témoigner toute notre reconnaissance à nos professeurs du département génie électrique et informatique industrielle, pour la formation enrichissante qu'ils nous ont offerte durant ces cinq années au sein de l'Ecole Nationale Supérieure de Technologie.

Ce mémoire n'aurait jamais pu voir le jour sans le soutien actif des membres de nos familles, spécialement nos chers parents qu'ils nous ont toujours encouragés.

Table de matières :

Table de figures **1**

Liste des abréviations **3**

Introduction générale **5**

Motivation 5

Problématique : 6

Objectifs : 6

Plan du manuscrit : 6

I. Etat de l'art et Généralités **8**

I.1 Introduction : 8

I.2 Deep learning : 8

I.3 Les réseaux neuronaux : 9

I.3.1 Réseau neuronal récurrent 10

I.3.1.1 RNN un à un : 12

I.3.1.2 RNN un à plusieurs : 12

I.3.1.3 RNN plusieurs à un : 12

I.3.1.4 RNN plusieurs à plusieurs : 13

I.3.2 Réseau neuronal convolutionel 13

I.3.2.1 Les couches du CNN 15

I.3.2.2 Les étapes du fonctionnement du CNN : 19

I.4 Etat de l'art sur le comptage de personnes basé sur CNN : 20

I.4.1 Méthode MCNN : 20

I.4.2 Méthode CNNs : 20

I.4.3 Méthode Switch-CNN network architecture: 21

I.4.4 Méthode CSRNet architecture : 21

I.4.5 Méthode de structure du réseau SFCN 21

I.5 Etat de l'art sur la détection du masque : 22

I.5.1 Détecteurs à un étage : 22

I.5.2 Détecteurs à deux étages : 23

I.6 Etat de l'art sur la distanciation : 24

I.7 Conclusion : 25

II. Hardware & Software **27**

II.1 Introduction 27

II.2 Logiciel 27

II.2.1 Jupiter notebook 27
II.2.2 Visual studio code 27
II.2.3 NSIS 28
II.2.4 Python 28
II.2.5 Bibliothèques 28
II.2.5.1 OpenCV 28
II.2.5.2 Tensorflow.keras 29
II.2.5.3 Pandas 29
II.2.5.4 NumPy 29
II.2.5.5 Tqdm 30
II.2.5.6 Sklearn 30
II.2.5.7 Mathplotlib 30
II.2.5.8 Tkinter 30
II.2.5.9 Pyinstaller 30
II.2.5.10 Scipy 30
II.3 Matériel 31
II.3.1 Ordinateur 31
II.3.2 Camera 31
II.4 Conclusion 31
III. Conception et réalisation 33
III.1 Introduction 33
III.2 Détection de masque avec CNN 33
III.2.2 Préparation et prétraitement des données 36
III.2.3 Formation du modèle de détection de masque 39
III.2.3.1 Réseau de neurones convolutionnels 40
III.2.4 Implémentation du détecteur de masque 44
III.3 Détection de la distance sociale et le comptage des personnes avecYOLOv3 : 45
III.3.1 Détection de personnes : 46
III.3.2 Comptage des personnes et Distance sociale 48
III.4 Interface graphique 49
III.4.1 Fonctionnement et design de l'interface 50
III.4.1.1 Lancement de l'interface 50
III.4.1.2 Détecter à partir d'une vidéo 51
III.4.1.3 Détecter par camera 55
III.4.2 Regrouper l'application 57

III.5 Conclusion 58
IV. Tests et discussions 60
IV.1 Introduction 60
IV.2 Les tests de performances 60
IV.2.1 Tests de détection à partir des images 60
IV.2.1.1 Test 1 60
IV.2.1.2 Test 2 61
IV.2.2 Tests de détection à partir d'une vidéo: 61
IV.2.2.1 Test 1 62
IV.2.2.2 Test 2 62
IV.2.3 Tests de détection en temps réel 63
IV.3 Problèmes et limites 65
IV.4 Conclusion 65
Conclusion générale 66
Annexe 67
Bibliographie 72
Résumé 75

Table de figures

Figure 1 : Definition et relation entre AI, ML et DL. 8
Figure 2 : Représentation d'un perceptron. 9
Figure 3 : Représentation d'un réseau neuronal. 10
Figure 4 : Représentation d'un réseau neuronal récurent. 11
Figure 5 : Représentation d'un RNN un à un. 12
Figure 6 : Représentation d'un RNN un à plusieurs. 12
Figure 7 : Représentation d'un RNN plusieurs à un. 13
Figure 8 : Représentation d'un RNN plusieurs à plusieurs. 13
Figure 9 : Représentation d'un CNN qui détecte un masque à partir d'une image. 14
Figure 10 : Représentation d'un CNN qui détecte un masque à partir d'un groupe d'images. 14
Figure 11 : Représentée d'une image comme un tableau de pixels. 15
Figure 12 : Fonctionnement du filtre de convolution. 15
Figure 13 : La carte de caractéristique résultante de la couche de convolution. 16
Figure 14 : La représentation graphique de la fonction ReLU. 16
Figure 15 : La carte de caractéristique après l'application de la fonction ReLU. 17
Figure 16 : Fonctionnement de la fonction max pooling. 17
Figure 17 : Carte de caractéristique corrigée après l'application du max pooling. 18
Figure 18 : Fonctionnement du flatenning. 18
Figure 19 : La couche entièrement connectée. 19
Figure 20 : la technique de similarité triangulaire.. 24
Figure 21 : L'architecture de détection de masque. 34
Figure 22 : Echantillons d'images de la base de données. 35
Figure 23 : Histogramme de la distribution de la base de données selon le type d'image. 35
Figure 24 : La détection du visage avant l'application du seuil de confiance. 37
Figure 25 : La détection du visage après l'application du seuil de confiance. 37
Figure 26 : Rognage de l'image selon le cadre du visage. 38
Figure 27 : Illustration d'un blob 3D. 39
Figure 28 : Histogramme de la distribution de la base de données selon l'utilité. 40
Figure 29 : L'ensemble des couches du réseau CNN proposé. 40
Figure 30 : Résumé du modèle. 42
Figure 31 : Comparaison entre les courbes de perte dans l'entrainement et le test. 43
Figure 32 : Comparaison entre les courbes de précision dans l'entrainement et le test. 43
Figure 33 : Application de détection de visage 44
Figure 34 : Illustration de l'architecture du projet 45
Figure 35 : Les détections du model YOLO appliqué sur une image 46
Figure 36 : L'image après application du seuil de confiance 47
Figure 37 : Principe du fonctionnement de l'IoU 47
Figure 38 : L'image après application de la suppression non-max 48
Figure 39 : Les centres de gravité de chaque personne détectée 48
Figure 40 : Application du seuil de distance sur les personnes détéctées 49
Figure 41 : Diagramme de fonctionnement de l'interface graphique 50
Figure 42 : Fenêtre d'accueil 51
Figure 43 : Fenêtre pour l'option de détection à partir d'une vidéo 51

Liste des figures

Figure 44 : Les messages d'erreur 52
Figure 45 : L'affichage après avoir cliquer sur Sélectionner 53
Figure 46 : L'affichage après avoir cliqué sur Visualiser 53
Figure 47 : L'affichage après avoir cliqué sur Détecter 54
Figure 48 : L'affichage après la fin de la détection et du comptage par vidéo 54
Figure 49 : La fenêtre de confirmation pour quitter 55
Figure 50 : Fenêtre pour l'option de détection par caméra 55
Figure 51 : L'affichage après avoir cliqué sur ouvrir la cam 56
Figure 52 : L'affichage après la fin de la détection et du comptage par cam 56
Figure 53 : La fenêtre du logiciel NSIS 57
Figure 54 : Génération du fichier d'installation 57
Figure 55 : Installation de l'interface graphique 58
Figure 56 : Test 1 de détection à partir d'une image 60
Figure 57 : Test 2 de détection à partir d'une image 61
Figure 58 : Test 1 de détection à partir d'une vidéo 62
Figure 59 : Test 2 de détection à partir d'une vidéo_1 62
Figure 60 : Test 2 de détection à partir d'une vidéo_2 63
Figure 61 : Test de détection en temps réel – non-distanciation et masque 63
Figure 62 : Test de détection en temps réel – non-distanciation et pas de masque 64
Figure 63 : Test de détection en temps réel – distanciation et non-masque 64
Figure 64 : Test de détection en temps réel – distanciation et masque 65
Figure 65 : collection des images du Mask et sans Mask avec leurs étiquettes. 68
Figure 66 : réseau de neurones convolutionelle pour la détection du masque. 68
Figure 67 : prétraitement des données (masque, sans masque). 69
Figure 68 : la fonction de la détection du masques, personnes et la distance sociale. 70
Figure 69 : détection à partir d'une vidéo. 71
Figure 70 : détection a partir de la cam. 71

Liste des abréviations

ARN*: Acide ribonucléique*

CNN*: Convolutional neural network*

CSRNet*: Crowded scene recognition* network

CV*: Computer vision*

DL*: Deep learning*

DNN*: Deep neural network*

IA*: Artificial Intelligence*

MCNN*: Multicolonal convolutional neural network*

MERS-CoV*: CoV du syndrome respiratoire du Moyen-Orient,*

ML*: Machine learning*

nCoV*: Nouveau coronavirus*

NN*: Neural network*

NSIS*: Nullsoft Scriptable Install System*

OMS*: Organisation mondiale de la Santé*

R-CNN*Region convolutional neural networl*

ReLU*: Rectified linear unit*

R-FCN*: Region-based Fully Convolutional Network*

RNN*: Reccurent neural n*etwork, *:*

RPN *: Region Proposal Network*

SARS-CoV-2*: Severe acute respiratory syndrome coronavirus 2*

SPPNet*: Spatial pyramidal pooling network*

SSD*: Single-Shot Detector*

SVM*: Support Vector Machine*

VSCode*: Visiual studio code*

YOLO*: You Only Look Once*

Introduction générale

Introduction générale

Motivation

La COVID-19 (coronavirus) est une maladie infectieuse qui s'est révélée épidémique, selon l'Organisation mondiale de la Santé (OMS). Il a été signalé pour la première fois à Wuhan, en Chine, à la fin de 2019. Selon l'OMS, une personne peut être infectée par la COVID-19 si elle entre en contact avec d'autres personnes infectées par le virus. Jusqu'à présent, aucun médicament n'a encore été mis au point pour enrayer ce virus mortel. Par conséquent, il est nécessaire de chercher une autre mesure de contrôle pour prévenir la propagation de ce virus mortel.

Cette épidémie a posé un grand défi au monde. Les techniques d'intelligence artificielle (IA) et plus précisément les algorithmes du deep learning basé sur la vision par ordinateur, peuvent aider les gens à lutter contre le virus de bien des façons. Par exemple, détecter les visages masqués, détecter les patients atteints de la COVID-19 avec l'analyse des images médicales, évaluer les risques d'infection, établir un système de suivi et des contacts, la prédiction de la structure des protéines du virus, etc. Parmi ces techniques, notre projet porte principalement sur les techniques de détection du masque et le comptage des personnes et s'assurer qu'ils appliquent de la distanciation sociale.

Comme il est bien dit, mieux vaut prévenir que guérir, l'OMS a suggéré plusieurs mesures de sécurité pour minimiser la transmission du coronavirus. La distanciation sociale s'est avérée être l'une des méthodes les plus importantes pour freiner la propagation. La distanciation sociale peut aussi être appelée « distanciation physique », ce qui signifie maintenir une distance entre une personne et les gens qui la contour, elle aide à réduire les contacts physiques ou les interactions entre des personnes possiblement infectées par la COVID-19 et des personnes en bonne santé.
Tout le monde doit garder une distance d'au moins 1 mètre, les uns des autres pour respecter la distanciation sociale. C'est un moyen important de briser la chaîne de contagion. Par conséquent, tous les pays touchés ont adopté la distanciation sociale.

L'utilisation des masques faciaux dans les lieux publics est aussi importante pour freiner la propagation du coronavirus. De plus, avec la réouverture des pays après le confinement lié au COVID-19, les gouvernements mondiaux et les organismes de santé publique recommandent le port du masque facial comme mesures essentielles pour assurer notre sécurité lorsque nous nous aventurons en public. Pour rendre obligatoire l'utilisation du masque, il devient essentiel de concevoir une technique qui oblige les personnes à appliquer un masque avant d'être exposées aux lieux publics.
La détection des masques faciaux consiste à déterminer si une personne porte un masque ou non. La détection faciale est un domaine clé dans le domaine de la vision par ordinateur et de la reconnaissance des formes. Les problèmes rencontrés avec cette approche comprennent une grande complexité dans la conception des caractéristiques et une faible précision de détection. Au cours des dernières années, des méthodes de détection en taille basées sur des réseaux de neurones à convolution profonde et le deep learning pour améliorer les performances de détection.

Problématique :

Malgré l'importance de la prévention, plusieurs personnes sont assez négligeant pour ne pas obéir aux instructions qui oblige à porter le masque, respecter la distanciation ou ne pas entrer dans des endroits limités par un nombre à ne pas dépasser mais qui est déjà atteint.

La détection et le comptage de personnes, et même la reconnaissance d'objet par le passé, ont eu un important corpus de recherches qui a contribué à la mise au point de plusieurs algorithmes basé sur les méthodes du deep learning pour former des classificateurs efficaces à la détection, à la reconnaissance et au comptage.

Ces algorithmes sont déjà conçus, mais séparément et chacun d'entre eux a un but précis hors de l'aide à la lute contre l'épidémie du COVID-19, comme la recognition du visage caché sous un masque, le comptage des gens dans les plateformes métro pour des raisons de sécurité et le calcul de la distance entre des objets et les cameras.

Comment peut-on trouver un système qui nous donne tout cela en même temps ? Un outil pas cher mais efficace. Notre solution fournis un système global pour contenir plusieurs applications dans une seule interface.

Objectifs :

Afin de s'assurer que les gens portent des masques et maintiennent une distance sociale dans les espaces publics, des règlements plus stricts doivent être mis en œuvre

Notre projet porte sur la conception et la réalisation d'une application de prévention pour lutter contre le COVID-19, en utilisant le deep learning pour détecter les violations concernant le port du masque, la non application de la distanciation social et qui fait le comptage de personnes détectés, et ceci en utilisant des méthodes existantes, en améliorant ses algorithmes, et en les mettant dans un seul code dans une interface graphique simple et facile à utiliser et interpréter par l'utilisateur, pour l'aider à s'assurer que les gens respectent les règles de sécurité liées au COVID-19.

Plan du manuscrit :

Le premier chapitre comporte diverses généralités sur la théorie du COVID-19,son contagiosité, la méthode dans laquelle il se propage, les types de vaccins existants et quelles sont les préventions qu'on doit appliquer pour limiter ses dégâts. Ce chapitre traitera aussi la théorie du deep learning et sa relation avec les réseaux neuronaux, ainsi l'état de l'art sur le comptage de personnes, la détection du masque et sur la distanciation sociale.

Dans le deuxième chapitre on présentera les différents équipements et logiciels utilisés lors de la conception et réalisation de notre application.

Le troisième est sur la conception et réalisation, qui est le cœur de notre projet, dont lequel on explique la procédure la méthodologie de la création de notre application COVID-19.

Le dernier chapitre comporte les tests sur l'application ainsi que la discussion des résultats obtenus.

Chapitre 1

Etat de l'art et généralités

I. Etat de l'art et Généralités

I.1 Introduction :

La covid-19 a posé un grand défi au monde. Les techniques d'intelligence artificielle (IA) et plus précisément les algorithmes du deep learning basés sur la vision par ordinateur, peuvent aider les gens à lutter contre le virus de bien des façons.

Dans ce chapitre on traitera la théorie du deep learning et sa relation avec les réseaux neuronaux, ainsi l'état de l'art sur le comptage de personnes, la détection du masque et sur la distanciation sociale.

I.2 Deep learning :

L'intelligence artificielle est le concept plus large des machines capables d'effectuer des tâches d'une manière qui serait considérée comme intelligente.

L'apprentissage automatique (machine learning - ML) une application actuelle de l'IA basée sur l'idée que la machine doit pouvoir accéder aux données et apprendre par elles-mêmes (1).

L'apprentissage profond (deep learning – DL) peut être considéré comme un sous-ensemble de l'apprentissage automatique, comme le montre la figure 1.

C'est un domaine qui repose sur l'apprentissage et l'amélioration en soi, en examinant les algorithmes informatiques. Alors que l'apprentissage automatique utilise des concepts plus simples, l'apprentissage profond fonctionne avec des réseaux neuronaux artificiels, qui sont conçus pour imiter la façon dont les humains pensent et apprennent.

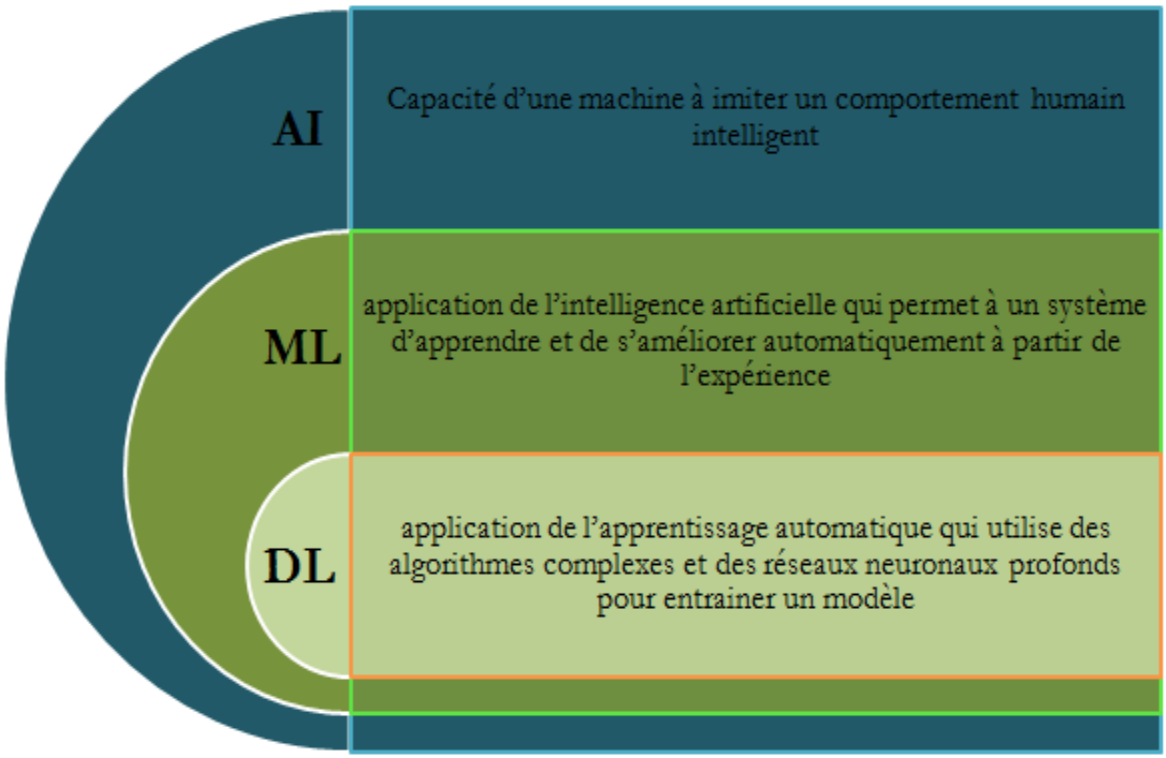


Figure 1 : Definition et relation entre AI, ML et DL.

Jusqu'à récemment, les réseaux neuronaux étaient limités par la puissance de calcul et donc par leur complexité. Cependant, les progrès réalisés en matière d'analyse des méga-données ont permis à des réseaux neuronaux plus vastes et sophistiqués, permettant aux ordinateurs d'observer, d'apprendre et de réagir à des situations complexes plus rapidement que les humains.

DL a facilité la classification des images, la traduction du langage, la reconnaissance vocale. Il peut être utilisé pour résoudre n'importe quel problème de reconnaissance de modèle et sans intervention humaine.

Considéré comme le domaine de l'apprentissage automatique qui connaît la croissance la plus rapide, l'apprentissage profond représente une technologie numérique véritablement perturbatrice, elle est utilisée par de plus en plus d'entreprises pour créer de nouveaux modèles d'affaires.

I.3 Les réseaux neuronaux :

Les réseaux neuronaux (NN) artificiels, comprenant de nombreuses couches, stimulent l'apprentissage profond. Les réseaux neuronaux profonds (DNN) sont de tels types de réseaux où chaque couche peut effectuer des opérations complexes telles que la représentation et l'abstraction qui donnent un sens aux images, au son et au texte.
Le perceptron est l'une des représentations les plus simples d'un neurone. Un perceptron qui est illustré dans la figure 2, est une unité de sortie unique, d'entrée multiple. Les données qui circulent dans le perceptron, sont de type binaire.

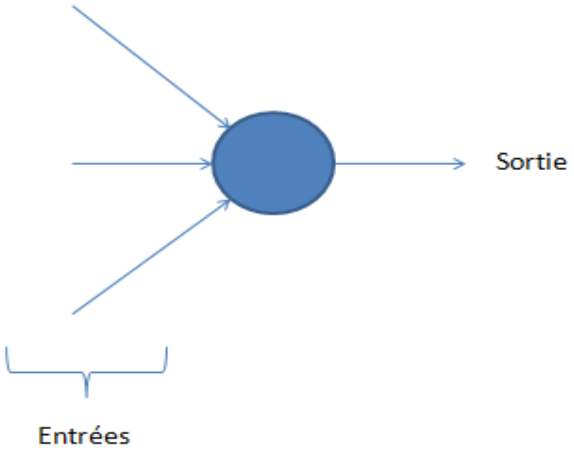


Figure 2 : Représentation d'un perceptron.

Les réseaux neuronaux sont des couches de nœuds, un peu comme le cerveau humain qui est composé de neurones, dans la figure 3, le réseau pris comme exemple contient une couche cachée principale

Les nœuds dans les couches individuelles sont connectés aux couches adjacentes. Le réseau est dit être plus profond en fonction du nombre de couches qu'il a. Un seul neurone dans le cerveau humain reçoit des milliers de signaux d'autres neurones. Dans un réseau neuronal artificiel, les signaux voyagent entre les nœuds et assignent les poids correspondants. Un nœud pondéré plus lourd exercera plus d'effet sur la couche suivante de nœuds. La couche finale compile les entrées pondérées pour produire une sortie.

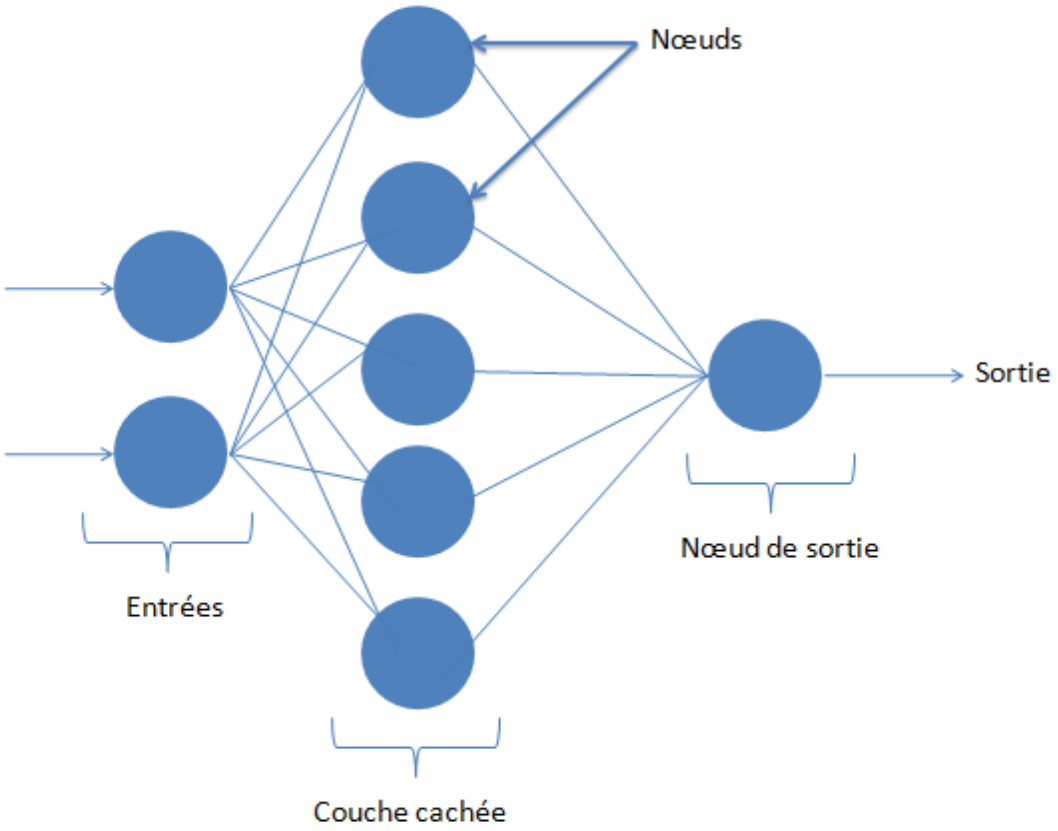


Figure 3 : Représentation d'un réseau neuronal.

Les systèmes d'apprentissage profond nécessitent de grandes quantités de données pour produire des résultats précis; par conséquent, l'information est alimentée sous forme d'énormes ensembles de données. Lors du traitement des données, les réseaux neuronaux artificiels sont en mesure de classer les données avec les réponses reçues d'une série de questions binaires vraies ou fausses impliquant des calculs mathématiques très complexes (2).

Par exemple, un programme de reconnaissance faciale fonctionne en apprenant à détecter et à reconnaître les bords et les lignes des visages, puis les parties plus importantes des visages, et, enfin, les représentations globales des visages. Au fil du temps, le programme s'entraîne et la probabilité de réponses correctes augmente. Dans ce cas, le programme de reconnaissance faciale identifiera précisément les visages avec le temps.

I.3.1 Réseau neuronal récurrent

Il fonctionne sur le principe d'enregistrement de la sortie d'une couche particulière et de la renvoyer à l'entrée afin de prédire la sortie de la couche. La principale différence entre le RNN et les autres réseaux est que chacune des cellules cachées reçoit sa propre sortie avec un délai fixe pour une ou plusieurs itérations (3).

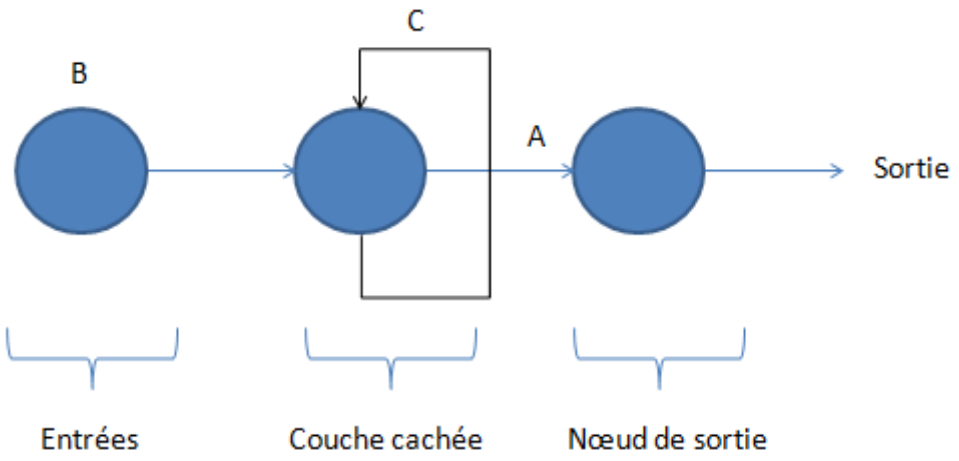


Figure 4 : Représentation d'un réseau neuronal récurent.

Les RNN sont utilisés dans une situation où la sortie dépend de l'ordre et du contexte des données transmises au modéliser. Par exemple, l'auto-complétion de texte est un cas d'utilisation parfaite pour ce type de réseaux.

Les nœuds des différentes couches du réseau neural sont compressés pour former une seule couche de réseaux neuronaux récurrents.

A, B et C sont les paramètres réseau utilisés pour améliorer la sortie du modèle. Si on nomme l'entrée 'x', à tout moment t, l'entrée courante est une combinaison d'entrée à x(t) et x(t-1). La sortie à tout moment est récupérée sur le réseau pour améliorer la sortie.

Dans les réseaux neuronaux récurrents, l'information passe par une boucle jusqu'à la couche cachée centrale.

La couche d'entrée « x » prend l'entrée dans le réseau neuronal et la traite et la transmet à la couche du milieu.

La couche cachée peut consister en plusieurs sous-couches cachées, chacune avec ses propres fonctions d'activation, poids et biais. Si vous avez un réseau neuronal où les différents paramètres des différentes couches cachées ne sont pas affectés par la couche précédente, c'est-à-dire que le réseau neuronal n'a pas de mémoire, alors vous pouvez utiliser un réseau neuronal récurrent.

Le réseau neuronal récurrent normalisera les différentes fonctions d'activation, les poids et les biais de sorte que chaque couche cachée ait les mêmes paramètres. Ensuite, au lieu de créer plusieurs couches cachées, il va en créer un et le passer en boucle autant de fois que nécessaire.

Il existe quatre types de réseaux neuronaux récurrents (4) :

I.3.1.1 RNN un à un :

Ce type de réseau neuronal est connu sous le nom de Vanilla Neural Network. Il est utilisé pour les problèmes généraux d'apprentissage automatique, qui a une seule entrée et une seule sortie.

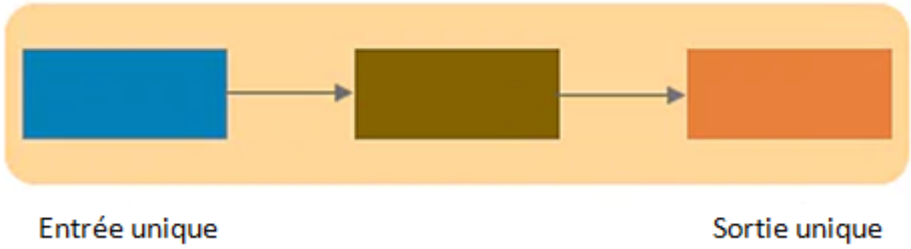


Figure 5 : Représentation d'un RNN un à un.

I.3.1.2 RNN un à plusieurs :

Ce type de réseau neuronal a une seule entrée et plusieurs sorties. Un exemple de cela est la légende de l'image.

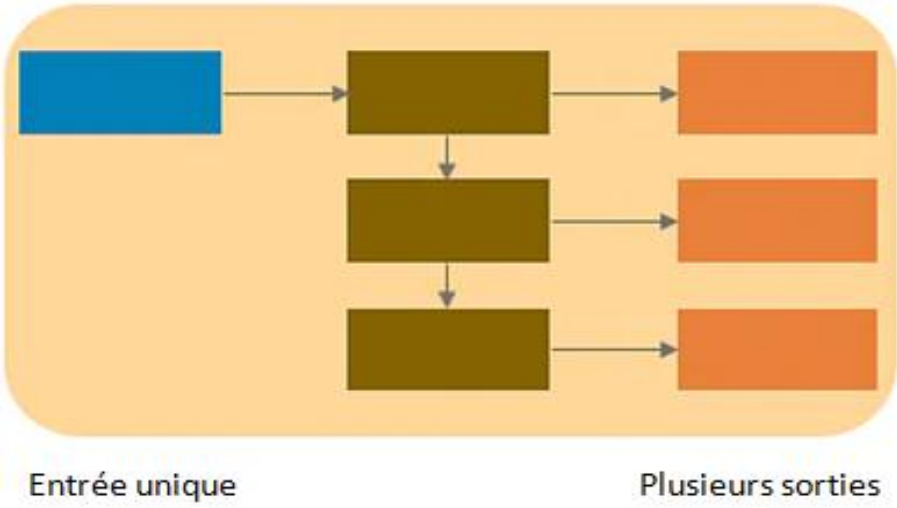


Figure 6 : Représentation d'un RNN un à plusieurs.

I.3.1.3 RNN plusieurs à un :

Ce RNN prend une séquence d'entrées et génère une seule sortie. L'analyse des sentiments est un bon exemple de ce genre de réseau où une phrase donnée peut être classée comme exprimant des sentiments positifs ou négatifs.

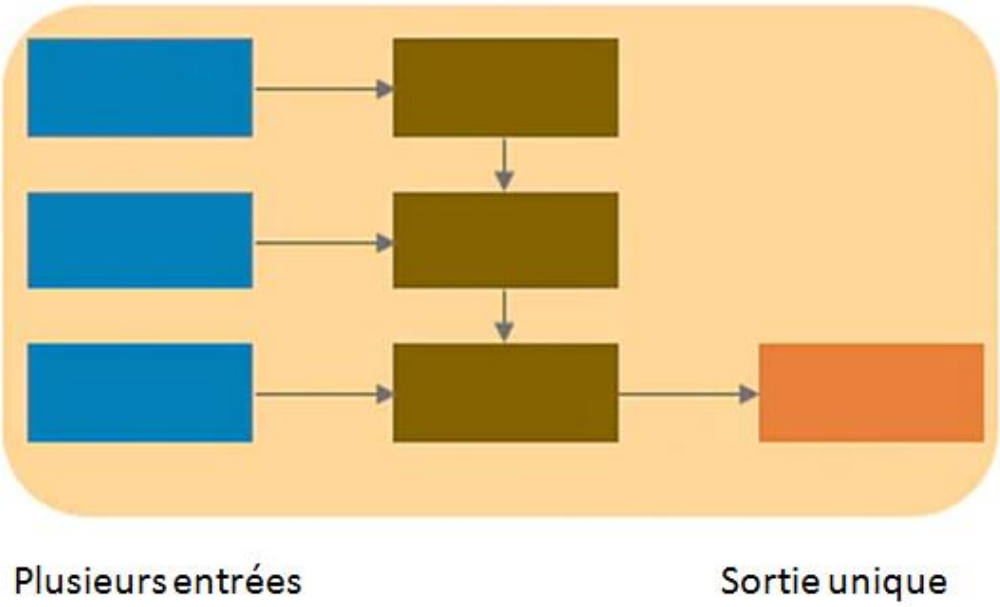


Figure 7 : Représentation d'un RNN plusieurs à un.

I.3.1.4 RNN plusieurs à plusieurs :

Ce RNN prend une séquence d'entrées et génère une séquence de sorties. La traduction automatique en est un exemple.

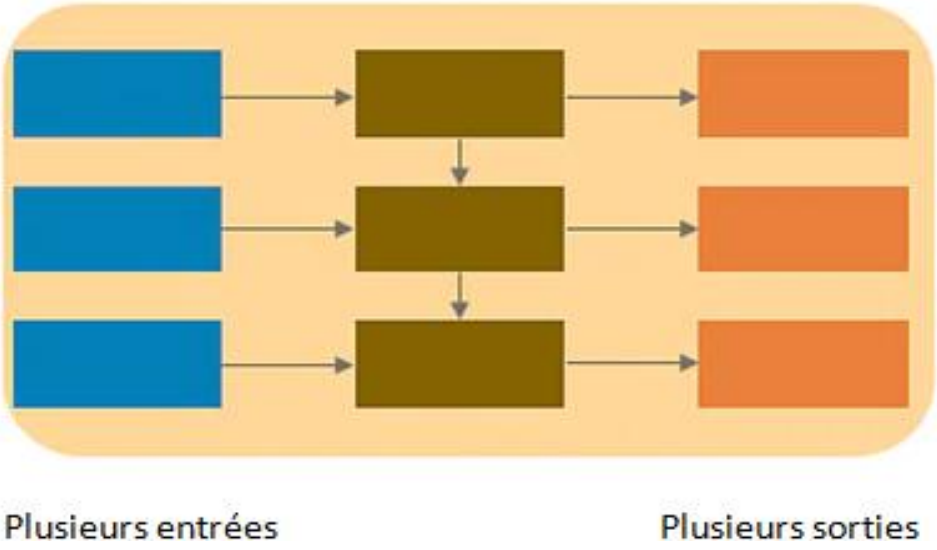


Figure 8 : Représentation d'un RNN plusieurs à plusieurs.

I.3.2 Réseau neuronal convolutionel

La reconnaissance faciale sur les médias sociaux, la détection d'objets ou la détection de maladies à l'aide d'images visuelles dans le domaine de la santé, tout cela est possible grâce aux réseaux de neurones convolutionnels (CNN) (5). Voici un exemple dans la figure 9, de réseaux de neurones convolutionnels qui illustre leur fonctionnement:

On prend une image d'une personne, et on essaie de confirmer si elle porte un masque ou non. La première chose qu'on fait c'est d'alimenter les pixels de l'image sous forme de tableaux à la couche d'entrée du réseau neuronal. Les couches cachées effectuent l'extraction

des caractéristiques en effectuant différents calculs et manipulations. Il y a plusieurs couches cachées comme le calque de convolution, le calque ReLU, et le calque de mise en commun, qui effectuent l'extraction de fonctionnalités de l'image. Enfin, il y a un calque entièrement connecté qui identifie l'objet dans l'image.

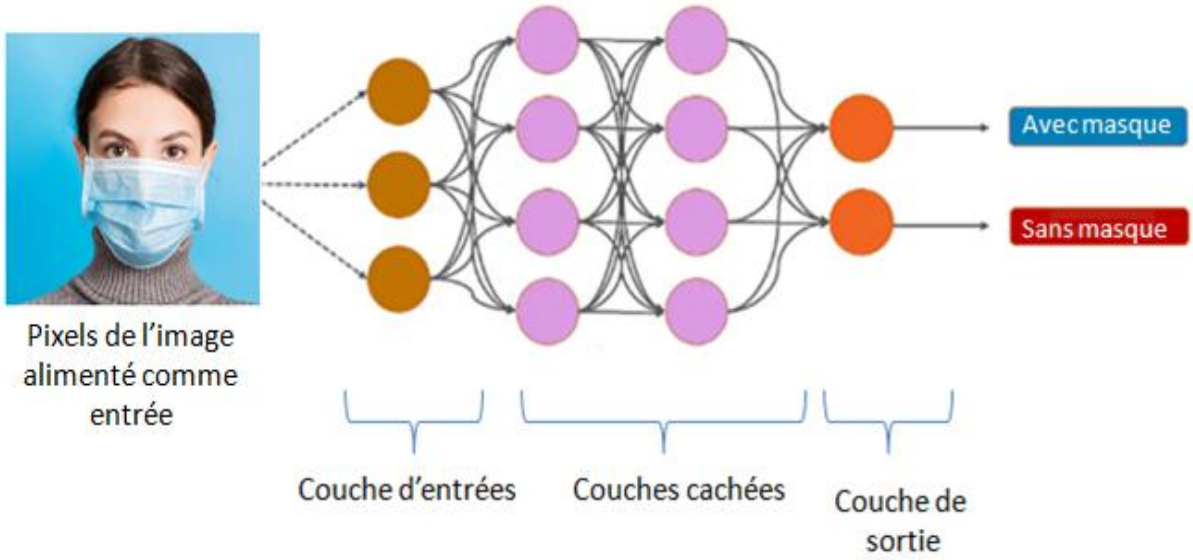


Figure 9 : Représentation d'un CNN qui détecte un masque à partir d'une image.

Un réseau de neurones convolutionnels est un réseau de neurones qui est généralement utilisé pour analyser des images visuelles en traitant des données avec une topologie semblable à une grille. Il est également connu sous le nom de ConvNet. Un réseau neuronal convolutif est utilisé pour détecter et classer les objets dans une image.

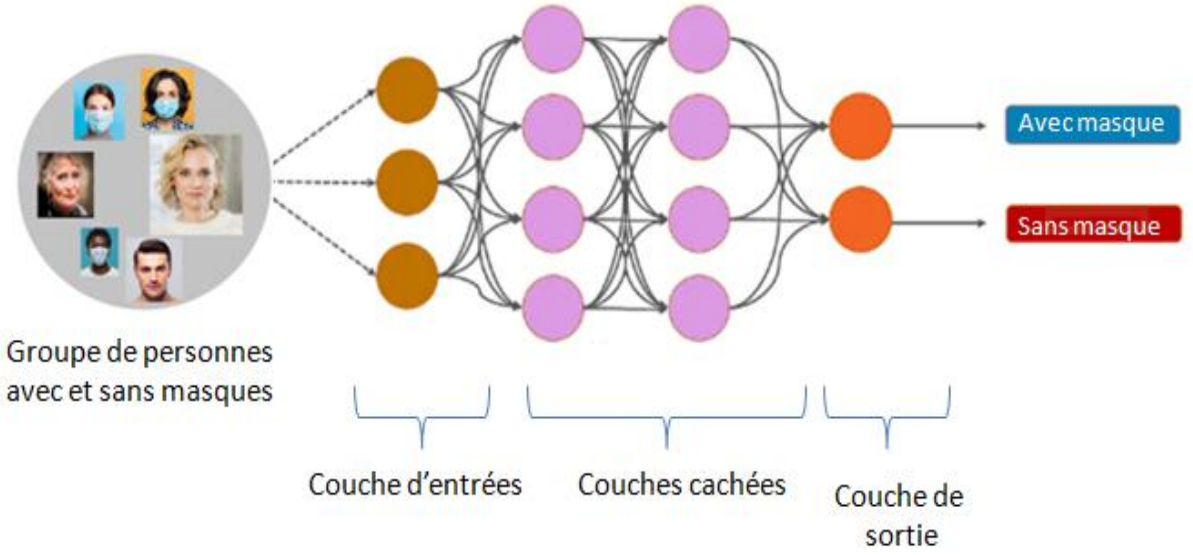


Figure 10 : Représentation d'un CNN qui détecte un masque à partir d'un groupe d'images.

Dans CNN, chaque image est représentée sous la forme d'un tableau de valeurs de pixels.



Figure 11 : Représentée d'une image comme un tableau de pixels.

I.3.2.1 Les couches du CNN

- Couche de convolution.

Il s'agit de la première étape du processus d'extraction des caractéristiques d'image. Une couche de convolution comporte plusieurs filtres qui effectuent l'opération de convolution. Chaque image est considérée comme une matrice de valeurs de pixels (6).

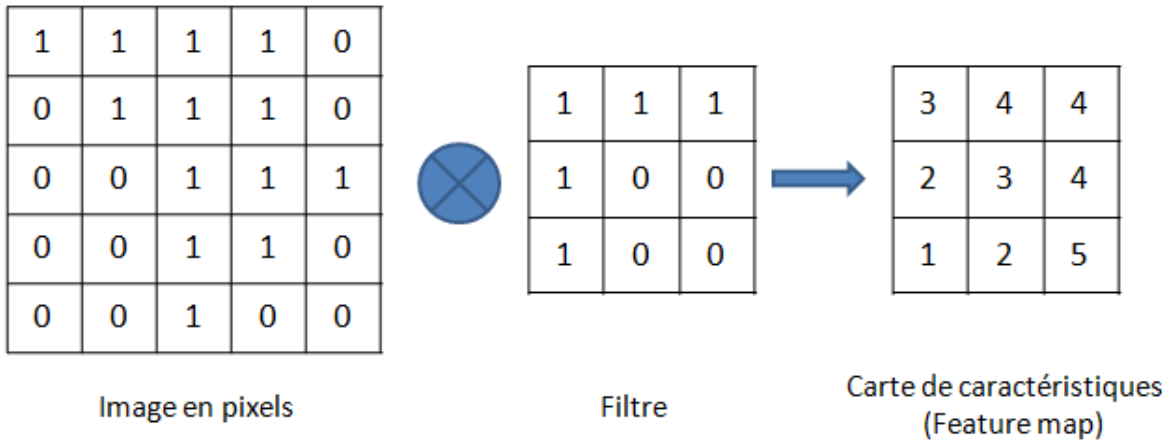


Figure 12 : Fonctionnement du filtre de convolution.

Considérant l'image ci-dessus de 5x5 dont les valeurs de pixel sont 0 ou 1. Il y a aussi une matrice de filtre avec une dimension de 3x3. En glissant la matrice de filtre sur l'image et après calcul du produit de point, on peut obtenir la matrice de la fonction de convolution.

En appliquant le filtre de convolution on obtient la carte de caractéristique (feature map) comme le montre la figure ci-dessous ;

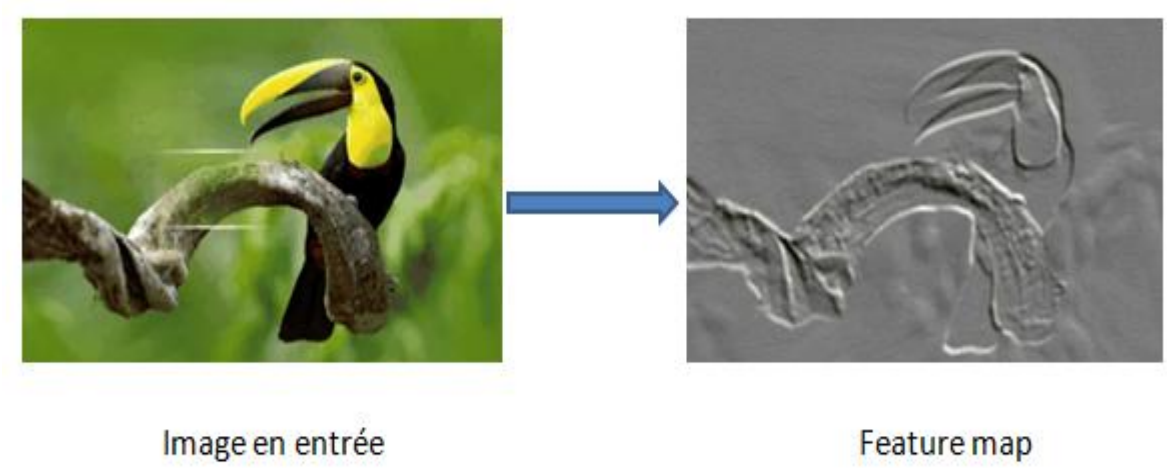


Figure 13 : La carte de caractéristique résultante de la couche de convolution.

- Couche ReLU

ReLU représente l'unité linéaire rectifiée. Une fois les cartes de caractéristiques extraites, l'étape suivante consiste à les déplacer vers une couche ReLU (7).

ReLU effectue une opération par élément et définit tous les pixels négatifs à 0. Il introduit la non-linéarité au réseau, et la sortie générée est une carte de caractéristiques rectifiée. Voici le graphique d'une fonction ReLU :

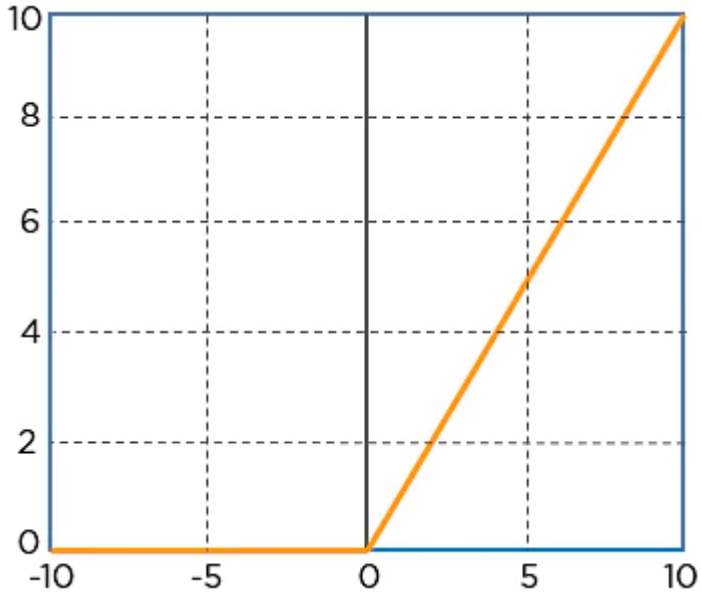


Figure 14 : La représentation graphique de la fonction ReLU.

L'image d'origine est scannée avec plusieurs convolutions et couches ReLU pour localiser les caractéristiques (features).

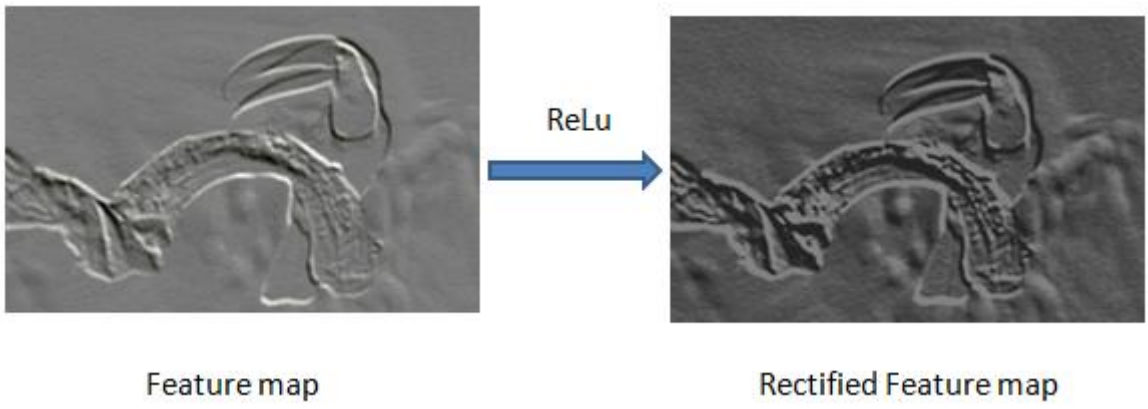


Figure 15 : La carte de caractéristique après l'application de la fonction ReLU.

- Couche de regroupement (pooling).

La couche de regroupement applique une opération d'échantillonnage descendante qui réduit la dimensionnalité de la carte des caractéristiques corrigée, pour générer une carte des caractéristiques regroupées (8).

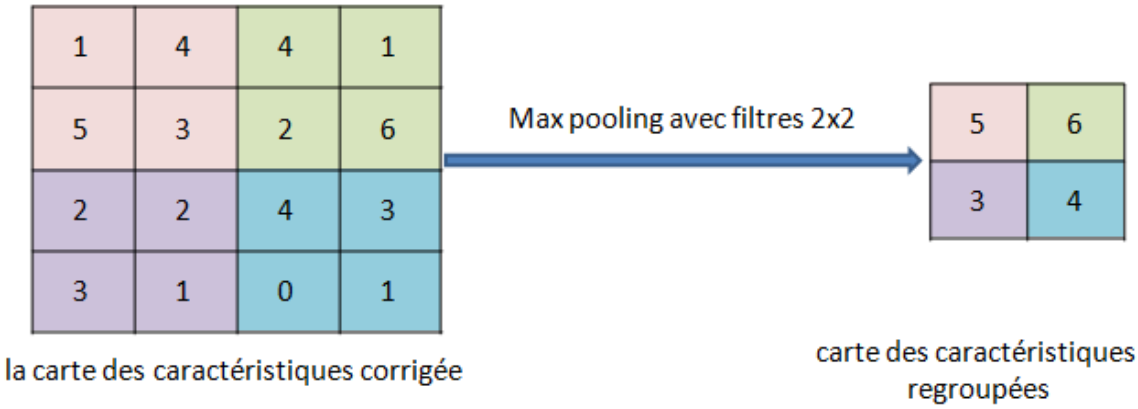


Figure 16 : Fonctionnement de la fonction max pooling.

La couche de mise en commun utilise divers filtres pour identifier différentes parties de l'image comme les bords, les coins, le corps, les plumes et les yeux.

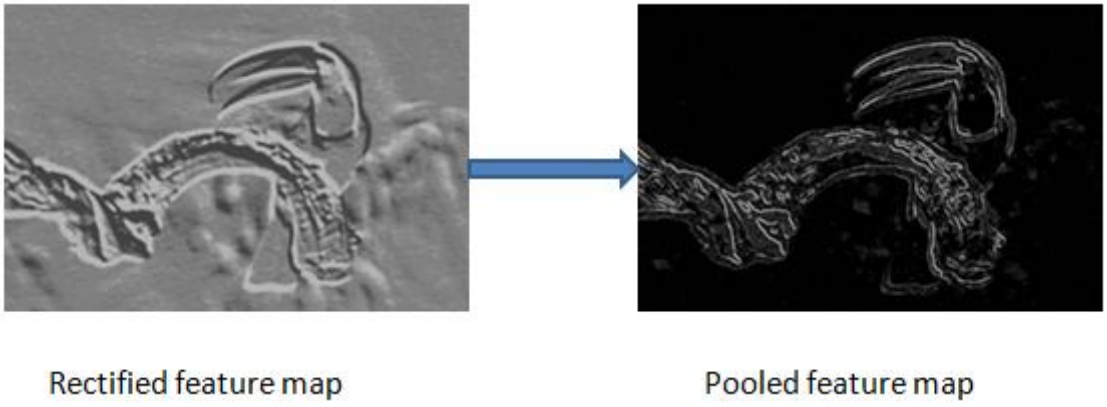


Figure 17 : Carte de caractéristique corrigée après l'application du max pooling.

- Etape aplatissement (flatenning) :

L'étape suivante du processus est appelée aplatissement. L'aplatissement est utilisé pour convertir tous les tableaux bidimensionnels résultants à partir de cartes de caractéristiques regroupées en un seul long vecteur linéaire continu (9).

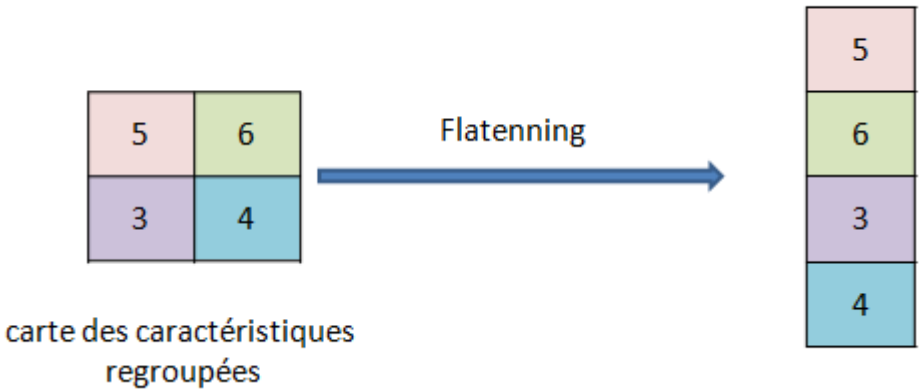


Figure 18 : Fonctionnement du flatenning.

- Couche entièrement connectée

La matrice aplatie est entrée dans la couche entièrement connecté pour classer l'image.

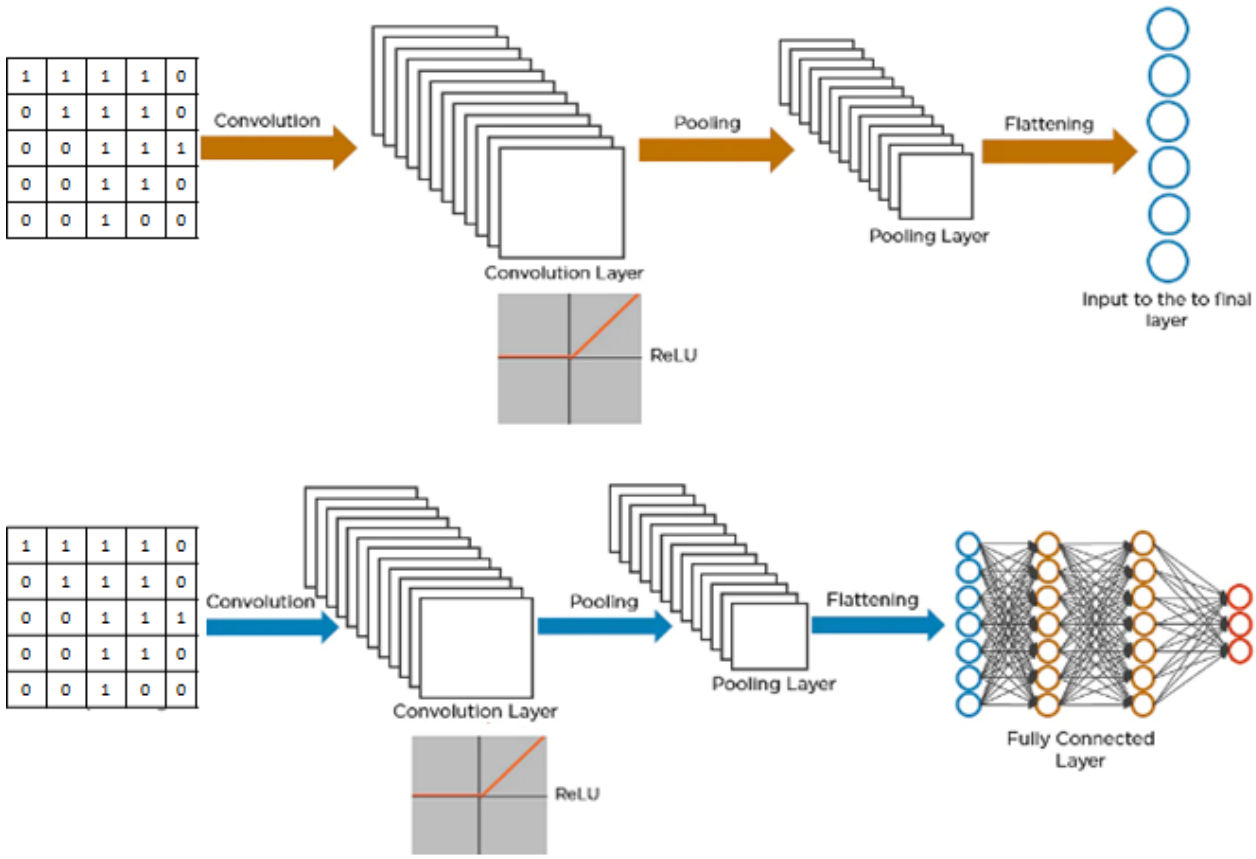


Figure 19 : La couche entièrement connectée.

La couche entièrement connectée est située dans la dernière partie de la couche cachée du CNN, et ne transmet que des signaux à d'autres couches entièrement connectées (10).

Les cartes de caractéristiques perdent leur topologie spatiale dans des couches entièrement connectées ; elles sont plutôt développées en vecteurs après être aplatis.

La couche convolutionelle et la couche de regroupement peuvent effectuer l'extraction de caractéristiques sur les données d'entrée. La couche entièrement connectée elle-même n'est pas censée avoir des capacités d'extraction de caractéristiques mais son rôle est d'effectuer une combinaison non linéaire des caractéristiques extraites pour obtenir la sortie et atteindre l'objectif de l'apprentissage.

I.3.2.2 Les étapes du fonctionnement du CNN :

- Les pixels de l'image sont transmis au calque convolutionnel qui effectue l'opération de convolution. Il en résulte une carte convolution
- La carte convolution est appliquée à une fonction ReLU pour générer une carte de caractéristiques rectifiée
- L'image est traitée avec plusieurs convolutions et couches ReLU pour localiser les caractéristiques
- Différentes couches de regroupement avec différents filtres sont utilisées pour identifier des parties spécifiques de l'image

- La carte des fonctionnalités regroupées est aplatie et transmise à une couche entièrement connectée pour obtenir la sortie finale

I.4 Etat de l'art sur le comptage de personnes basé sur CNN :

Les méthodes de comptage de foule visent à estimer le nombre de personnes dans des vidéos de surveillance ou une seule image. Ils peuvent être utilisés dans divers scénarios, comme des assemblées politiques, des événements sportifs et des concerts, pour assurer la sécurité publique en surveillant la densité de la foule, pour notre cas elle est très importante dans la lutte contre le COVID-19. Lorsque la densité de population dans un petit périmètre atteint un sommet élevé, le risque de propagation de la maladie augmente.

On présente ci-dessous quelque méthodes de comptage de personnes qui se basent sur les réseaux de nuerons convolutionnels.

I.4.1 Méthode MCNN :

En 2016, Zhang et al. Ont proposé une méthode de comptage des foules sur une seule image à l'aide d'un CNN multi colonne (11)

Ce modèle estime le nombre de personnes sans segmenter le premier plan. Il doit utiliser les caractéristiques de différentes échelles pour estimer avec précision le nombre de personnes dans différentes images, et le modèle peut automatiquement apprendre les caractéristiques efficaces. La raison pour utiliser MCNN est qu'en raison de la distorsion de perspective, les images contiennent souvent des têtes humaines de différentes tailles, et un seul type de noyau de convolution n'est pas suffisant. Ainsi, plusieurs colonnes de différentes convolutions sont proposées pour accommoder plusieurs tailles.

Le réseau utilise une couche convolutive pour transformer le tenseur multi-canal en un plan, remplaçant la couche d'origine entièrement connectée. De cette façon, la taille de l'image d'entrée peut être choisie arbitrairement, ce qui rend le modèle de réseau le plus largement utilisé.

I.4.2 Méthode CNNs :

En 2017, Vishwanath A. Sindagi et Vishal M. Patel ont proposé une méthode d'apprentissage multitâche en cascade basée sur l'estimation préalable et la densité avancées de CNN (12). En fusionnant un modèle de haut niveau antérieur avec le réseau (13), Ilsapprennent un modèle qui correspond à différents niveaux de densité dans l'ensemble de données. L'antériorité de haut niveau est divisée en différents groupes étiquetés en fonction du nombre de personnes sur l'image (14). Ensuite, à l'aide de marqueurs, grâce à ce précédent de haut niveau, le nombre de personnes dans l'image entière peut être estimé approximativement sans être affecté par les changements d'échelle (15). En conséquence, le réseau peut apprendre des caractéristiques globales plus discriminantes. Le réseau de haut niveau antérieur et CNN sont utilisés pour estimer ensemble la carte de densité.

Ce réseau donne une image de la taille de la tâche et produit une carte de la densité de la population. Le réseau se compose de deux parties, la première partie apprend le niveau élevé antérieur, et la seconde partie estime des cartes de densité. La première partie contient un ensemble de couches convolutionnelles, une couche de regroupement pyramidale et une

couche entièrement connectée. La seconde partie est constituée d'un ensemble de couches convolutionnelles, suivies d'une couche convolutionnelle à petit pas, qui est utilisée pour sur échantillonner la sortie de la couche précédente afin de compenser la perte de détail causée par la couche de regroupement précoce.

I.4.3 Méthode Switch-CNN network architecture:

La même année, D.B Sam et d'autres ont proposé des réseaux neuronaux à convolution commutée (16). Le réseau utilise les changements de densité de la foule dans l'image pour améliorer la précision et la localisation des comptes de foule prédits.

Tout d'abord, plusieurs CNN de différentes tailles de noyau de convolution sont utilisés comme cartes de régression pour la prédiction de carte de densité, puis un classificateur de sélection entraîné est utilisé pour sélectionner le régresseur CNN optimal pour chaque image d'entrée, et les résultats sont utilisés comme résultats finaux.

I.4.4 Méthode CSRNet architecture :

En 2018, Li et al. Ont proposé un réseau pour la reconnaissance de scènes encombrées (crowded scene recognition) appelé CSRNet. Une façon de comprendre les scénarios très encombrés est en offrant un apprentissage profond axé sur les données. Cela permet de faire des estimations de dénombrement précises et de construire des cartes de densité de grande qualité (17).

Le modèle de réseau CSRnet est principalement divisé en réseaux frontaux et dorsaux. La taille de la carte de densité de sortie est de 1/8 de l'image d'entrée originale.

Un réseau de neurones convolutionnels creux est utilisé comme réseau principal pour élargir le champ réceptif tout en maintenant la résolution pour générer une carte de distribution de foule de haute qualité.

I.4.5 Méthode de structure du réseau SFCN

En 2019, Junyu Gao et coll. ont proposé deux méthodes de rendement qui utilisent des données synthétiques pour améliorer le dénombrement de la population (18).

Tout d'abord, une stratégie de supervision est utilisée pour réduire le débordement. Plus précisément, des données synthétiques à grande échelle sont d'abord utilisées pour préparer un compteur de foule, qui est un réseau spatial entièrement convolutif (SFCN-spatial full convolutional network).

Le compteur actuel a ensuite été utilisé pour corriger le compteur. Cette méthode peut améliorer efficacement la performance des données réelles. Certaines couches des modèles traditionnels ont une initialisation aléatoire ou une distribution régulière. Par rapport à eux, la structure du réseau SFCN peut fournir des paramètres d'initialisation plus complets et meilleurs.

Deuxièmement, Junyu Gao et al. ont proposé une méthode de comptage de foule adaptative de domaine qui améliore les capacités de transfert inter-domaines. Elle peut transformer efficacement la scène de foule synthétisée en une scène réelle. Cette méthode conserve efficacement les informations de motif et de texture locales, en particulier dans les zones bondées et certains arrière-plans. Enfin, les données synthétisées sont converties en une image réaliste. Sur la base de ces données, ils ont formé un compteur de foule sans étiquettes de données réelles, ce qui fonctionne bien dans la nature.

I.5 Etat de l'art sur la détection du masque :

L'apprentissage de modèles et la reconnaissance d'objets sont les tâches inhérentes à une technique de vision par ordinateur (CV). La reconnaissance des objets englobe à la fois la classification des images et la détection des objets (19). La tâche de reconnaître le masque sur le visage dans la zone pubienne peut être accomplie en déployant un algorithme de reconnaissance d'objet efficace par des dispositifs de surveillance. Le processus de reconnaissance des objets consiste à générer les propositions régionales, puis à classer chaque proposition en catégories connexes (20). On examine l'évolution récente des techniques de proposition régionale à l'aide de détecteurs à un ou deux étages, de techniques générales pour améliorer la détection des propositions régionales et de modèles préformés basés sur ces techniques.

I.5.1 Détecteurs à un étage :

Les détecteurs à un étage traitent la détection des propositions de région comme un simple problème de régression en prenant l'image d'entrée et en apprenant les probabilités de classe et les coordonnées des boîtes limitantes. OverFeat (21) et DeepMultiBox (22)en sont les premiers exemples.

YOLO (You Only Look Once) a popularisé l'approche à une étape en démontrant des prédictions en temps réel et en réalisant une vitesse de détection remarquable, mais a souffert d'une faible précision de localisation par rapport aux détecteurs à deux étapes, en particulier lorsque de petits objets sont pris en considération (23). Fondamentalement, le réseau YOLO divise une image en une grille de taille GxG, et chaque grille génère des prédictions N pour les boîtes limitatives. Chaque boîte limitative est limitée à une seule classe pendant la prédiction, ce qui empêche le réseau de trouver des objets plus petits.

De plus, le réseau YOLO a été amélioré à YOLOv2, ce qui comprend la normalisation par lots, le classificateur de haute résolution et les boîtes d'ancrage.

En outre, le développement de YOLOv3 est construit sur YOLOv2 avec l'ajout d'un classificateur amélioré et un nouveau réseau pour l'extraction de fonctionnalités. Bien que, YOLOv3 est exécuté plus rapidement que le détecteur à prise unique (SSD-Single-Shot Detector) mais ne fonctionne pas bien en termes de précision de classification (24).

En outre, YOLOv3 nécessite une grande quantité de puissance de calcul pour l'inférence, ce qui ne convient pas pour les appareils embarqués ou mobiles. Ensuite, les réseaux SSD ont des performances supérieures à YOLO en raison de petits filtres convolutionnels, de multiples cartes de caractéristiques et de prédictions à plusieurs échelles. La principale différence entre les deux architectures est que YOLO utilise deux couches entièrement

connectées, tandis que le réseau SSD utilise des couches convolutionnelles de tailles variables.

En outre, le RetinaNet (25) proposé par Lin est également un détecteur d'objets à un étage qui utilise la pyramide d'images et la perte focale pour détecter les objets denses dans l'image à travers plusieurs couches et atteint une précision remarquable ainsi qu'une vitesse comparable aux détecteurs à deux étages

I.5.2 Détecteurs à deux étages :

Contrairement aux détecteurs à un seul étage, les détecteurs à deux étages suivent une longue ligne de raisonnement dans la vision par ordinateur pour la prédiction et la classification des propositions régionales. Ils prédisent d'abord les propositions dans une image, puis appliquent un classificateur à ces régions pour classer la détection potentielle. Divers modèles de proposition régionale en deux étapes ont déjà été proposés par les chercheurs.

Le réseau de neurones convolutionnels régional a également été abrégé en R-CNN (26) décrit en 2014 par Ross Girshick et al. Il pourrait avoir été l'une des premières applications à grande échelle de CNN au problème de localisation et de reconnaissance des objets. Le modèle a été démontré avec succès sur des ensembles de données de référence tels que VOC-2012 et ILSVRC-2013 et a produit des résultats de pointe.

Fondamentalement, R-CNN applique un algorithme de recherche sélective pour extraire un ensemble de propositions d'objets à un stade initial et applique le classificateur SVM (Support Vector Machine) pour prédire les objets et les classes connexes à un stade ultérieur. Le pooling des pyramides spatiales SPPNet (27) (modifie R-CNN avec une couche SPP) recueille les caractéristiques de diverses propositions régionales et les introduit dans une couche entièrement connectée pour la classification. La capacité de SPNN à calculer des cartes de caractéristiques de l'image entière en une seule prise a entraîné une amélioration significative de la vitesse de détection des objets d'une magnitude de près de 20 plis supérieure à R-CNN.

Ensuite, Fast RCNN est une extension sur R-CNN et SPPNet. Elle introduit une nouvelle couche appelée couche de regroupement de la région d'intérêt (RoI) entre les couches convolutives partagées pour affiner le modèle. De plus, il permet d'entraîner simultanément un détecteur et un régresseur sans modifier les configurations du réseau. Bien que le Fast-R-CNN intègre efficacement les avantages du R-CNN et du SPPNet, sa vitesse de détection reste inférieure à celle des détecteurs à un seul étage (28).

De plus, Faster R-CNN est un amalgame de fast R-CNN et Region Proposal Network (RPN). Il permet de réaliser des propositions de régions pratiquement gratuites en intégrant progressivement des blocs individuels (par ex.emple, détection de propositions, extraction de fonctionnalités et régression de boîtes limitatives) du système de détection d'objets en une seule étape (29). Bien que cette intégration mène à la réalisation d'une percée pour le goulot de vitesse de R-CNN rapide, mais il existe une redondance de calcul à l'étape de détection ultérieure.

Le réseau régional entièrement convolutionnel (R-FCN- Region-based Fully Convolutional Network) est le seul modèle qui permet une rétro-propagation complète pour la formation et

l'inférence (30). Feature Pyramid Networks (FPN) peut détecter des objets non uniformes, mais moins utilisés par les chercheurs en raison du coût élevé du calcul et de l'utilisation accrue de la mémoire (31).

En outre, Mask R-CNN renforce Faster R-CNN en incluant la prédiction de masques segmentés sur chaque RoI (32). Bien que deux étapes donnent une grande précision de détection d'objets, elle est limitée par une faible vitesse d'inférence en temps réel pour la surveillance vidéo (33).

I.6 Etat de l'art sur la distanciation :

Pour calculer la distance entre deux personnes d'abord la distance entre la personne et la caméra est calculée en utilisant la technique de similarité triangulaire, On calcule la distance focale perçue de la caméra, en supposant que D est la distance entre la personne et la caméra, et la hauteur réelle de la personne est de H=165cms et avec SSD de mobileNetv2, la hauteur en pixel P de la personne est identifiée à l'aide des coordonnées de la boîte de délimitation (34).

En utilisant ces valeurs, la distance focale de la caméra peut être calculée en utilisant la formule ci-dessous :

$$F = (P \times D) / H$$

Ensuite, nous utilisons la hauteur de la personne réelle H, la hauteur de pixel de la personne P, et la distance focale de la caméra F pour mesurer la distance de la personne de la caméra. La distance par rapport à la caméra peut être déterminée comme suit :

$$D1 = (H \times F) / P$$

Après avoir calculé la profondeur de la personne dans la caméra, nous calculons la distance entre deux personnes dans la vidéo. Un certain nombre de personnes peuvent être détectées dans une vidéo.

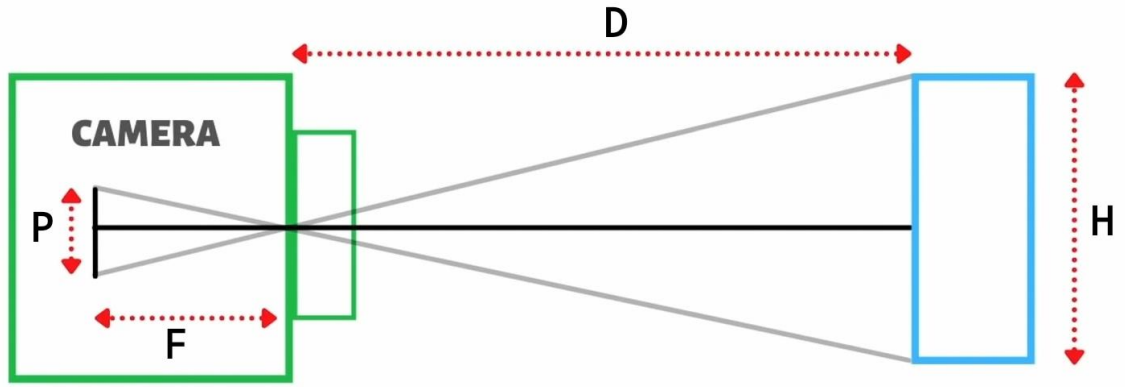


Figure 20 : la technique de similarité triangulaire..

Ainsi, la distance euclidienne est mesurée entre le point médian des boîtes limitantes de tous les individus détectés. Ce faisant, nous avons obtenu des valeurs x et y, et ces valeurs de

pixels sont convertis en centimètres. Nous avons les coordonnées x, y et z (la distance de la personne de la caméra) pour chaque personne en cm. La distance euclidienne entre chaque personne détectée est calculée à l'aide des coordonnées (x, y, z). Si la distance entre deux personnes est inférieure à 2 mètres ou 200 centimètres, une boîte de délimitation rouge est affichée autour d'eux, indiquant qu'ils ne maintiennent pas une distance sociale.

I.7 Conclusion :

Dans ce chapitre on a vu des généralités à propos du corona virus, sa méthode de propagation et les préventions qu'on doit appliquer, comme la distanciation sociale et le port du masque.

Ils existent plusieurs méthodes pour les appliquer, on a cité quelque uns dans l'état de l'art du comptage de personnes, de détection du masque et de la distanciation sociale. Ces algorithmes fonctionnent à l'aide du deep learning et les réseaux neuronaux qu'on a décrit aussi dans ce chapitre.

Chapitre 2

Hardware & Software

II. Hardware & Software

II.1 Introduction

Ce chapitre est divisé en deux parties, une est le software, où on décrit les logiciels utilisé comme jupiter notebook et vs code, ainsi que le langage de programmation utilisé python et ses différentes bibliothèques et paquets très utiles durant toute la période de conception et réalisation de notre application covid-19.

La deuxième partie de ce chapitre est consacrée pour le hardware dont on donne les spécifications de l'ordinateur qu'on utilise pour la réalisation de notre projet, et son web camera intégrée.

II.2 Logiciel

II.2.1 Jupiter notebook

Jupyter Notebook est une application web open-source qui permet de créer et partager des documents contenant du code en direct. Il permet aussi aux développeurs de partager du code et de l'exécuter dans la même interface utilisateur**.**

Ses utilisations comprennent le nettoyage et la transformation des données, la simulation numérique, la modélisation statistique, la visualisation des données, l'apprentissage automatique et bien plus encore.

Jupyter a été créé pour faciliter la présentation du travail de programmation d'un développeur et permettre aux autres de s'impliquer. Il permet de mélanger du code, des commentaires et des visualisations dans des documents interactifs appelés notebooks, qui peuvent être partagés, et réutilisés.

Le Jupyter Notebook n'est pas inclus avec Python, donc pour l'utiliser, on doit d'abord l'installer.

II.2.2 Visual studio code

Visual Studio Code (connu sous le nom de VS Code) est un éditeur de texte libre par Microsoft. Il est disponible pour Windows, Linux et macOS. Bien que l'éditeur soit relativement léger, il comprend quelques fonctionnalités puissantes qui ont fait de lui l'un des outils de développement les plus populaires ces derniers temps.

VS Code supporte un large éventail de langages de programmation de Python, C++ et java à CSS et Go. En outre, VS Code permet d'ajouter et même de créer de nouvelles extensions, y compris les débogueurs, le cloud et le développement web.

L'interface utilisateur de VS Code permet beaucoup d'interaction par rapport aux autres éditeurs de texte pour simplifier l'expérience utilisateur.

Une fois téléchargé et installé, VS Code se lance, identifie le langage de programmation utilisé et les fonctionnalités spécifiques à mettre en œuvre. Le développeur peut alors explorer ou rechercher des options pour trouver et corriger les erreurs de codage. L'édition

majeure se fait via l'option extension, tandis que le contrôle source permet l'utilisation du répertoire Git.

Les programmeurs peuvent rapidement naviguer entre plusieurs outils pour apporter les corrections nécessaires à leur code. Avec ses options de personnalisation faciles à utiliser, VS Code offre également des raccourcis clavier pour les combinaisons de touches communes et les opérations répétitives.

II.2.3 NSIS

NSIS (Nullsoft Scriptable Install System) est un système open source professionnel pour créer des installateurs Windows. Il est conçu pour être aussi petit et flexible que possible et il est donc très adapté à la distribution Internet.

II.2.4 Python

Python est un langage de programmation orienté objet, de haut niveau avec une sémantique dynamique. Ses structures de données intégrées de haut niveau, combinées à un typage dynamique et une liaison dynamique, le rendent très attirant pour le développement rapide d'applications, ainsi que pour une utilisation comme langage de script ou de collage pour connecter des composants existants. La syntaxe simple et facile à apprendre de Python met l'accent sur la lisibilité et réduit donc le coût de maintenance du programme. Python prend en charge les modules et les paquets, ce qui encourage la modularité des programmes et la réutilisation du code. L'interpréteur Python et la bibliothèque standard étendue sont disponibles sous forme source ou binaire sans frais pour toutes les plates-formes principales, et peuvent être librement distribués.
Souvent, les programmeurs préfèrent Python en raison de la productivité qu'il fournit. Comme il n'y a pas d'étape de compilation, ça le rend plus rapide. Déboguer les programmes Python est facile, le débogueur lui-même est écrit en Python, témoignant du pouvoir introspectif de Python : un bug ou une mauvaise entrée ne causera jamais de problème de segmentation. Au lieu de cela, lorsque l'interprète découvre une erreur, il soulève une exception.

L'un des plus grands avantages de Python est la liste assez vaste de bibliothèques qu'il contienne. Une bibliothèque est un ensemble de modules (constantes, variables, classes, fonctions.) qui ajoutent à Python des possibilités étendues comme la programmation réseau, le calcul numérique, génération de documents, graphisme, formatage de texte

II.2.5 Bibliothèques

II.2.5.1 OpenCV

OpenCV (Open Source Computer Vision Library) est une bibliothèque de logiciels de vision par ordinateur et d'apprentissage automatique. Elle vise à fournir une infrastructure commune pour les applications de vision par ordinateur et à accélérer l'utilisation de la perception artificielle dans les produits commerciaux.

La bibliothèque dispose de plus de 2500 algorithmes optimisés, qui comprennent un ensemble complet d'algorithmes de vision et d'apprentissage automatique. Ces algorithmes peuvent être utilisés pour détecter et reconnaître des visages, identifier des objets, classer des actions humaines dans des vidéos, suivre les mouvements des caméras, suivre des objets en

mouvement, extraire des modèles 3D d'objets, assembler les images pour produire une image haute résolution d'une scène entière, trouver des images similaires dans une base de données d'images, retirer les yeux rouges des images prises à l'aide du flash, suivre les mouvements des yeux, reconnaître les paysages et établir des marqueurs pour les superposer à la réalité augmentée, et plusieurs d'autres fonctionnalités. OpenCV a plus de 47 mille personnes de la communauté des utilisateurs et le nombre estimé de téléchargements dépasse 18 millions. Cette bibliothèque est largement utilisée dans les entreprises, les groupes de recherche et par les organismes gouvernementaux.

II.2.5.2 Tensorflow.keras

Keras est une interface de programmation d'application de haut niveau de TensorFlow. Cette dernière est une interface accessible, hautement productive pour résoudre les problèmes d'apprentissage automatique, avec un accent sur l'apprentissage profond moderne. Elle fournit des abstractions essentielles et des éléments de base pour le développement et l'expédition de solutions d'apprentissage automatique avec une vitesse d'itération élevée.

II.2.5.3 Pandas

Pandas est une bibliothèque écrite pour le langage de programmation Python pour la manipulation et l'analyse des données. Elle propose notamment des structures de données et des opérations de manipulation de tables numériques et de séries chronologiques. Il s'agit d'un logiciel libre publié dont le nom est dérivé du terme « panel data », terme économétrique pour les ensembles de données qui comprennent des observations sur plusieurs périodes pour les mêmes personnes.

Pandas est principalement utilisé pour l'analyse des données et la manipulation des données tabulaires dans les bases de données. Il permet d'importer des données à partir de différents formats de fichiers tels que JSON, Parquet, tables ou requêtes de base de données SQL, et Microsoft Excel. Pandas permet diverses opérations de manipulation de données telles que la fusion, le remodelage, la sélection, ainsi que le nettoyage des données.

Le développement de pandas introduit dans Python de nombreuses fonctionnalités de travail avec des data frames. Cette bibliothèque pandas est construite sur une autre bibliothèque NumPy, qui est orientée pour travailler efficacement avec des tableaux au lieu des fonctionnalités de travail sur des Data frames.

II.2.5.4 NumPy

NumPy est le paquet fondamental pour le calcul scientifique en Python. Il s'agit d'une bibliothèque Python qui fournit un objet de tableau multidimensionnel, divers objets dérivés (tels que les tableaux masqués et matrices), fait des opérations rapides sur les tableaux, y compris mathématique, logique, manipulation de la forme, tri, sélection, transformations discrètes de Fourier, algèbre linéaire de base, opérations statistiques de base, simulation aléatoire et bien plus encore.

En Python, ils existent des listes qui sont utile pour les tableaux, mais elles sont lentes à traiter. Les tableaux sont très fréquemment utilisés en science des données, où la rapidité et les ressources sont très importantes. NumPy vise à fournir un objet tableau jusqu'à 50x plus rapide que les listes Python traditionnelles.

II.2.5.5 Tqdm

C'est une bibliothèque en Python qui est utilisée pour créer des Progrémetre ou des barres de progression. Tqdm tire son nom du nom arabe taqaddum, qui signifie « progrès ».

Implémenter tqdm peut être fait sans effort dans les boucles, fonctions ou même Pandas. Les barres de progression sont assez utiles en Python parce que grâce à eux on peut voir si le Kernal fonctionne toujours, il donne aussi le temps d'exécution du code et le temps estimé pour le code à compléter qui aiderait tout en travaillant sur d'énormes ensembles de données

II.2.5.6 Sklearn

Scikit-learn est une bibliothèque clé pour le langage de programmation Python qui est généralement utilisé dans les projets d'apprentissage automatique. Elle est axée sur des outils, y compris les algorithmes d'usage général, mathématiques, et statistiques qui forment la base de nombreuses technologies d'apprentissage automatique. En tant qu'outil gratuit, Scikit-learn est extrêmement important dans de nombreux types de développement d'algorithmes pour l'apprentissage automatique et les technologies connexes.

II.2.5.7 Mathplotlib

Matplotlib est l'une des bibliothèques présente dans python les plus populaires et les plus utilisées, qui fournissent divers outils pour la visualisation des données en Python.

C'est l'une des bibliothèques de traçage les plus puissantes en Python. Il s'agit d'une bibliothèque multiplateforme qui fournit divers outils pour créer des tracés 2D à partir des données dans des listes ou des tableaux en python.

Il permet de visualiser les données en utilisant une variété de différents types de tracés pour rendre les données compréhensibles. On peut utiliser ces différents types de graphiques (histogrammes, graphiques à barres, boxplots, et plusieurs d'autres.) en écrivant quelques lignes de code en python.

II.2.5.8 Tkinter

Tkinter est l'abréviation de 'Tool kit interface' est une bibliothèque python qui permet de crée des interfaces graphiques.

II.2.5.9 Pyinstaller

PyInstaller est un paquet en Python qui regroupe toutes les dépendances des applications Python dans un seul paquet, tel qu'on n'a pas besoin d'installer différents paquets ou modules pour différentes applications.

PyInstaller lit et analyse le code python, puis découvre les modules dont le programme a besoin pour s'exécuter. Il les regroupe ensuite dans un seul dossier ou un seul fichier exécutable avec l'extension .exe pour windows.

II.2.5.10 Scipy

SciPy est une bibliothèque scientifique pour python, elle contient un ensemble d'algorithmes mathématiques et de fonctions reposant sur l'extension NumPy de Python. Elle ajoute une puissance significative à la session Python interactive en fournissant à l'utilisateur des commandes de haut niveau et des classes pour manipuler et visualiser les données. Avec

SciPy, une session Python interactive devient un environnement de traitement de données rivalisant avec des systèmes tels que MATLAB, IDL et SciLab.

II.3 Matériel

II.3.1 Ordinateur

L'appareil utilisé : DESKTOP-02P8MI6
Processeur : Intel Core(TM) i7-6600U CPU @ 2.60GHz 2.80 GHz
Mémoire RAM installée : 8,00 Go (7,68 Go utilisable)
Type du système : Système d'exploitation 64 bits

II.3.2 Camera

La caméra utilisée est intégré à l'ordinateur, en utilisant une application web cam, pour l'exploiter nommé : ECap de la version : 1.0.0.9 .

II.4 Conclusion

Ce chapitre a permit de décrire les différents logiciels et bibliothèques du langage de programmation python qui ont été utilisées au cours de la réalisation de notre projet.

On a aussi mis en évidence les spécifications de l'ordinateur et la camera utilisés.

Chapitre 3

Conception et réalisation

III. Conception et réalisation

III.1 Introduction

Dans ce chapitre, qui est divisé en trois parties majeures, on a résumé les étapes essentielles de la conception et réalisation de notre application.

La première partie concerne la détection de masque avec CNN, dont laquelle on s'est focalisé sur la base de données utilisée, sa préparation, prétraitement et formation du modèle de détection de masque, puis son implémentation.

La seconde est consacrée pour l'explication de la méthode de détection et comptage de personnes et la distance qui leurs séparent, en utilisant le modèle YOLOv3 de la bibliothèque de python nommé OpenCv basée sur le DL ;

Dans la troisième partie on a élucidé le fonctionnement de notre application, avec toutes ses options, que ce soit via une utilisation de camera en directe, ou en sélectionnant une vidéo à partir du stockage interne

Cette interface a permis de faciliter la détection de violations, concernant le port du masque, la non-application de la distanciation sociale et le non-respect de la limite autorisé de nombre de personnes.

III.2 Détection de masque avec CNN

Afin de prédire est-ce qu'une personne porte un masque ou pas, le modèle utilisé nécessite d'abord d'apprendre d'un ensemble de données bien structuré.

Ce modèle est basé sur un réseau neuronal convolutif (CNN) comme architecture de base , qui est divisée en plusieurs couches. Ce réseau fonctionne à l'aide de plusieurs bibliothèques comme Open CV, Keras et numpy.

Le modèle proposé est conçu en trois phases : prétraitement des données, formation du modèle et application de détecteur de masque.

Le prétraitement se fait d'abord en collectant les images, ensuite on charge le modèle de Caffe, et on commence le traitement et en fin on fait stocker les images dans un vecteur.

La formation du modèle CNN, est faite après chargement des données, en les faisant passer par les différentes couches du réseaux neuronal convolutionnel, comme la couche du max-pooling, ReLU, et l'optimiseur ADAM, puis on sauvegarde le modèle.

La dernière étape qui est l'application du détecteur, commence en chargeant le modèle formé précédemment et lui insérer les visages détectés comme entrées à fin de détecter la présence du masque.

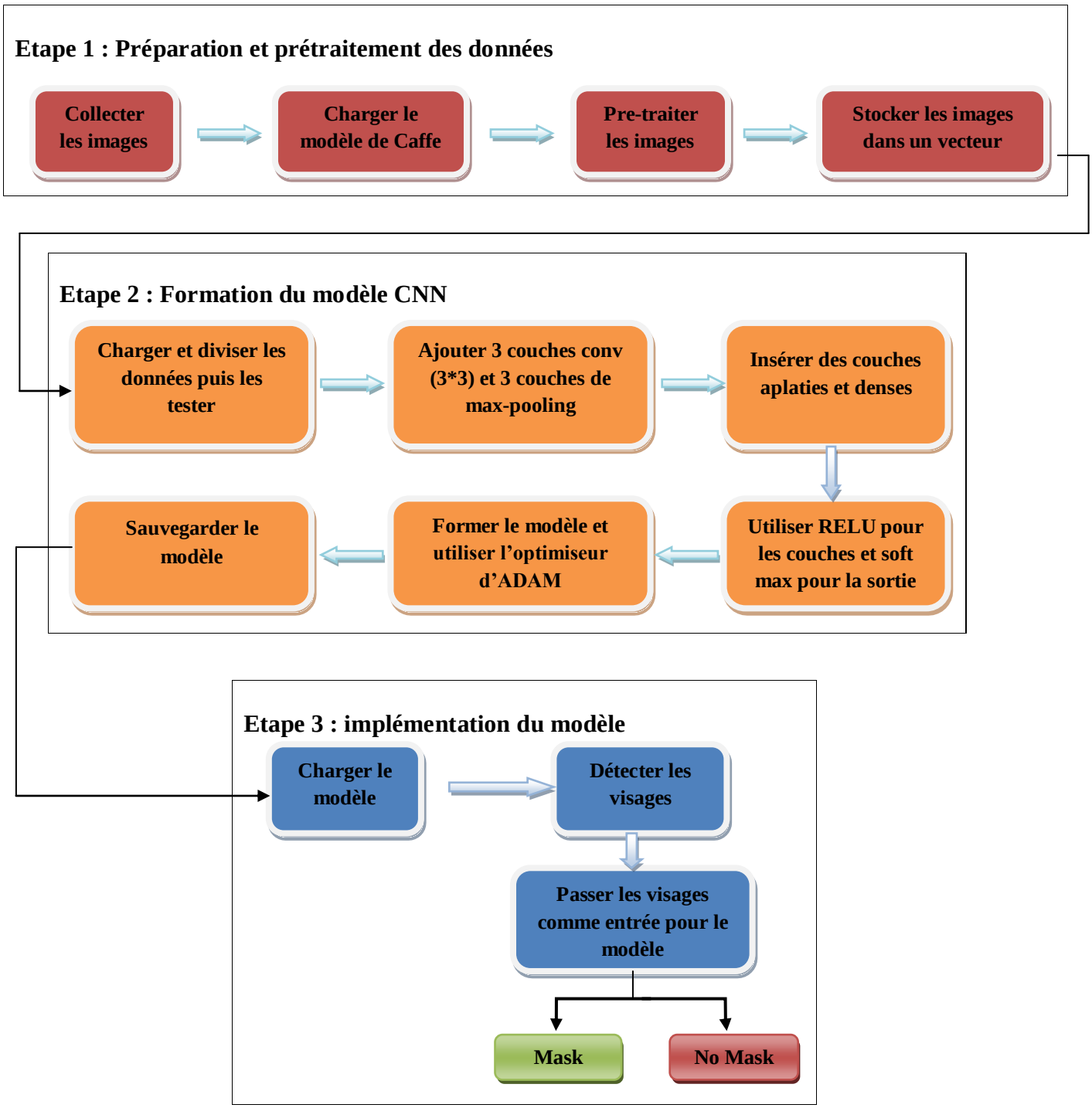


Figure 21 : L'architecture de détection de masque.

III.2.1 Base de données

L'ensemble de données utilisées dans ce projet a été collecté sous plusieurs formats d'image tels que JPG, PNG et d'autres. La figure 22 présente l'échantillon de l'ensemble de données. C'est un mélange de différents ensembles de données et d'images open-source, y compris l'ensemble de données pour la détection de masque crée par Umansid (35) sur Kaggle, qui est une plateforme web qui accueille la plus grande communauté de Data Science au monde,et aussi l'ensemble des données de MaskedFace-Net (36).

Par conséquence, il y avait un ensemble d'images variées avec déférentes taille et résolution. Toutes les photos provenaient de ressources simulant des scénarios réels, et d'autres ont été créées artificiellement en mettant un masque sur le visage.

L'ensemble de données de Masked Face-Net a rassemblé des images essentielles des visages et des repères faciaux appliqués pour trouver les caractéristiques faciales de la personne dans l'image.

Les principaux repères faciaux comprennent les yeux, le nez, le menton, les lèvres et les sourcils. Ces repères permettent de créer intelligemment un ensemble de données en formant un masque sur une image d'origine sans masque.

Enfin, l'ensemble de données a été divisé en deux classes avec deux différentes étiquettes : 4130 images « Avec masque » et 2100 images « Sans masque », les images ont été organisées, pour un total d'environ 6000 images.



Figure 22 : Echantillons d'images de la base de données.

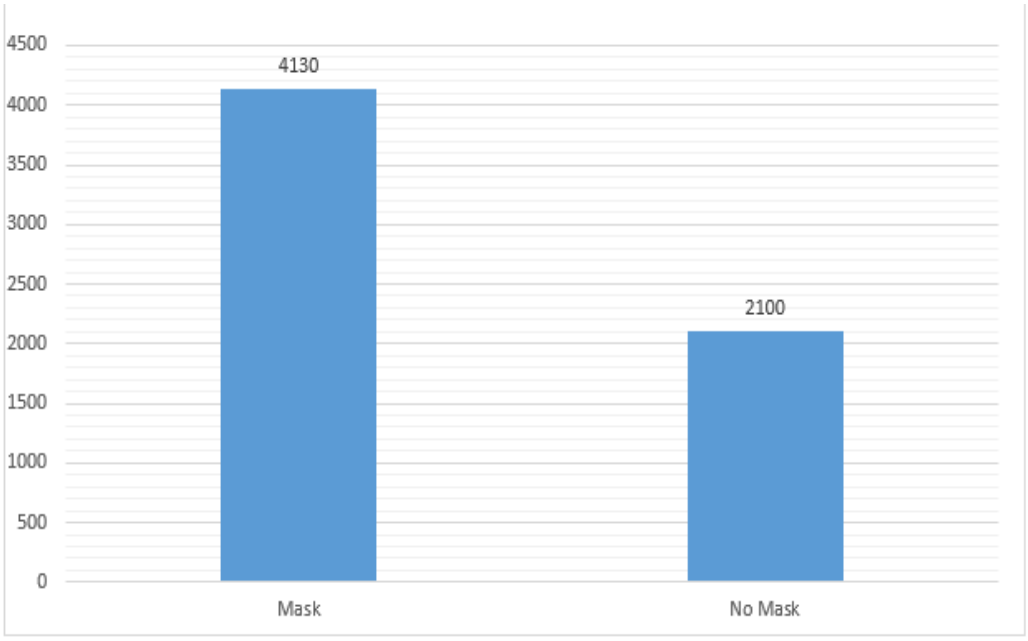


Figure 23 : Histogramme de la distribution de la base de données selon le type d'image.

III.2.2 Préparation et prétraitement des données

D'abord, on doit collecter le chemin de toutes les images et leurs étiquettes (Mask, No Mask) et les mettre dans une liste, après on charge le modèle de détection de visage préformé nommé « Caffemodel » en utilisant le module de OpenCV nommé « DNN.Caffemodel », qui est créé et entretenu par Berkeley AI Research (BAIR) et d'autres partenaires pour la communauté comme une alternative plus puissante et plus efficace aux techniques de détection d'objets existantes.

Des fichiers de modèle Caffe sont nécessaires, ils peuvent être téléchargés à partir du Dépôt GitHub OpenCV.

Le fichier deploy.prototxt décrit l'architecture du réseau.

Le fichier res_10 _300x300_ssd _iter_140000.caffemodel est le modèle qui contient les poids des couches (37).

On reçoit le nombre de visages identifiés après avoir utilisé le modèle de détection, qui est ensuite fourni comme entrée au modèle de détection de masque facial.

Le module DNN signifie Deep Neural Network module. C'est un inclus dans OpenCV qui est responsable de tous ce qui est liées à l'apprentissage profond. Il a été introduit dans OpenCV version 3 et maintenant dans la version 4.3. Ce module nous permet d'utiliser des réseaux de neurones préformés à partir de cadres populaires comme TensorFlow, PyTorch, Caffe, et YOLO, et d'utiliser ces modèles directement dans OpenCV.

Cela signifie que nous pouvons former des modèles en utilisant un Framework populaire comme TensorFlow et ensuite faire inférence prédiction avec OpenCV (38)**.**

La fonction cv2.dnn.readNetfromcaffe prend deux parameters :

« path/to/prototxtfile » et « path/to/Caffemodel weights »

En utilisant la fonction blobfromImage de Opencv pour nous donner l'entrée du model pour mettre en évidence la région d'intérêt (visage) dans l'image par un rectangle, mais l'algorithme du modèle Caffe produit environ 200 rectangles, dont la plupart sont non pertinents/redondants.

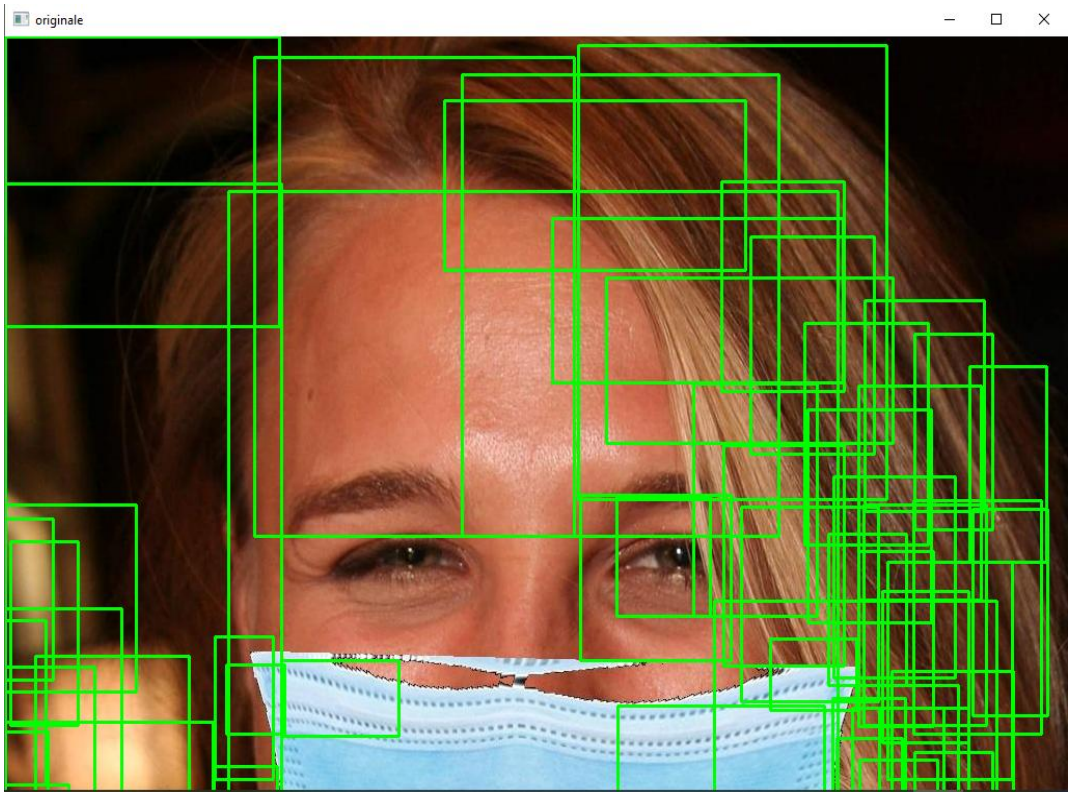


Figure 24 : La détection du visage avant l'application du seuil de confiance.

Par conséquent, on doit filtrer et enlever les rectangles inutiles. On se débarrasse de tous les rectangles qui ont une faible probabilité de détection d'objets.

Cela se fait par le seuil de confiance, et en gardant seulement les cases qui ont une probabilité de plus du seuil de confiance. Cette étape élimine les détections anormales d'objets.

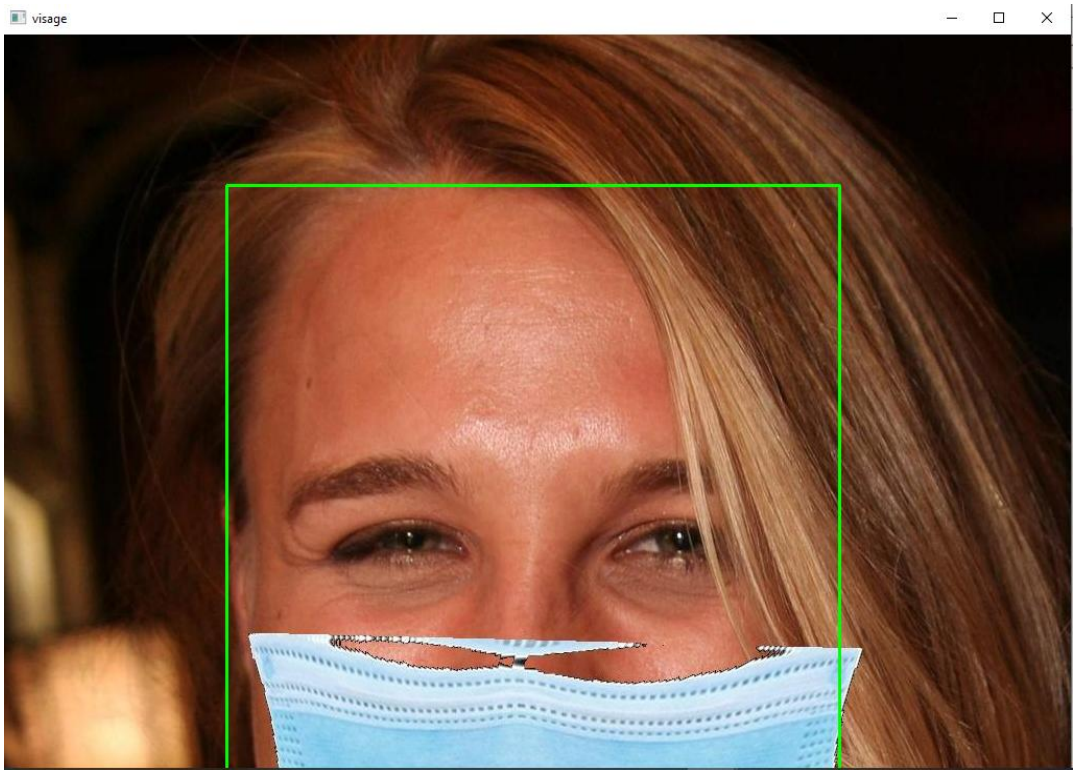


Figure 25 : La détection du visage après l'application du seuil de confiance.

Ensuite on rogne l'image selon le cadre du visage comme montré dans la figure 26, et on fait de même pour toutes les images de notre base de données. Apres rognage on redimensionne nos images pour 100*100 pixels, ce qui permet de réduire la charge sur la machine durant l'entrainement et de fournir des résultats optimaux. Le choix de la dimension est arbitraire selon les capacités du matériel utilisé.

Le redimensionnement se fait à l'aide du module blob from Image de Opencv qui fait extraire le blob, qui enveloppe les données réelles traitées et transmises par Caffe, et fournit également une capacité de synchronisation entre le CPU et le GPU.

Mathématiquement, un blob est un objet de type : vecteur, à 4 dimensions (images, canaux-RGB, largeur, hauteur). C'est grâce aux blobs que Caffe peut stocker les données.

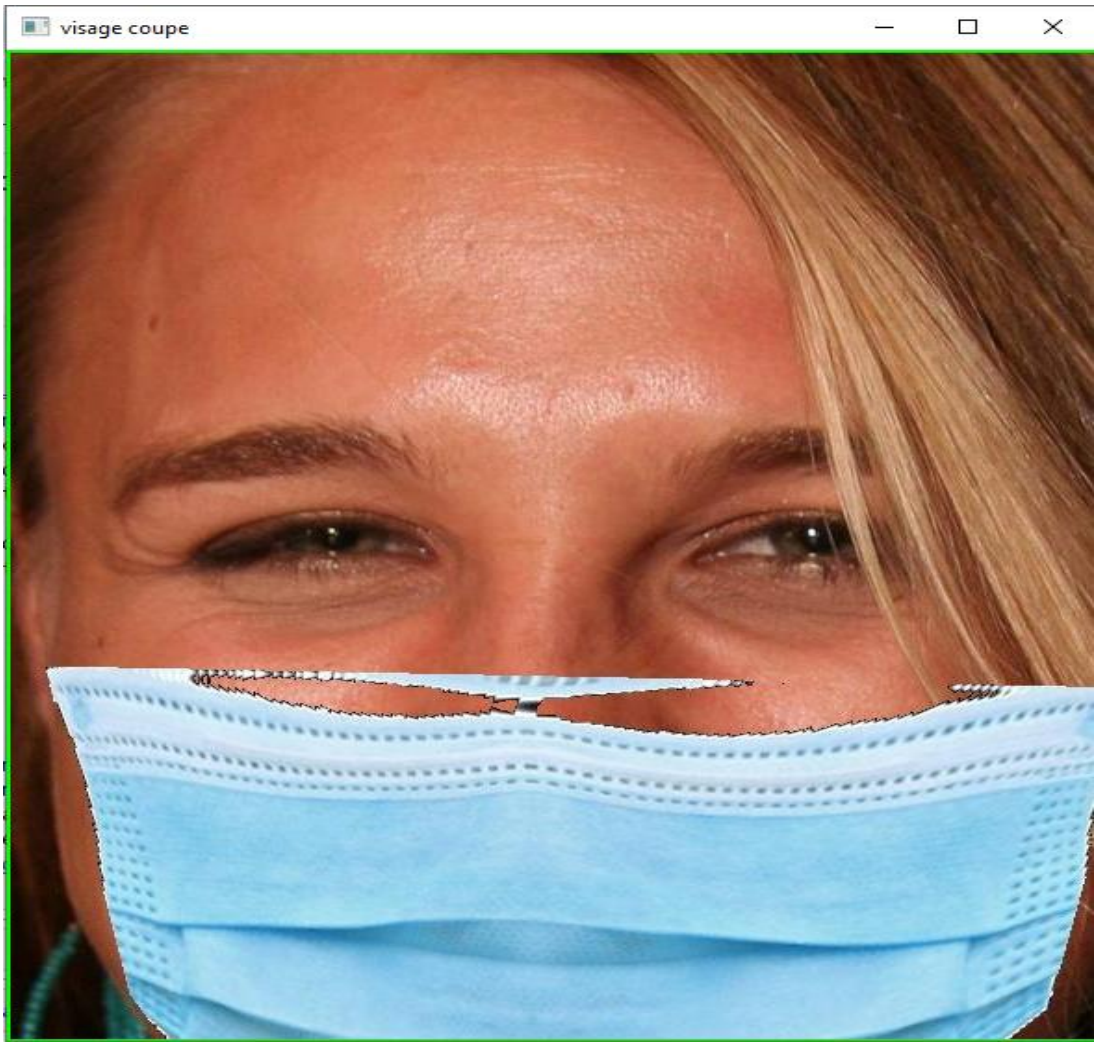


Figure 26 : Rognage de l'image selon le cadre du visage.

Pour afficher le blob on doit réduire la dimension en 3D, Puis on fait pivoter l'image pour avoir la même forme que l'image originale.

On fait extraire le blob de toutes les images avec leurs étiquettes, et on les stocke dans un fichier vecteur (.npz).

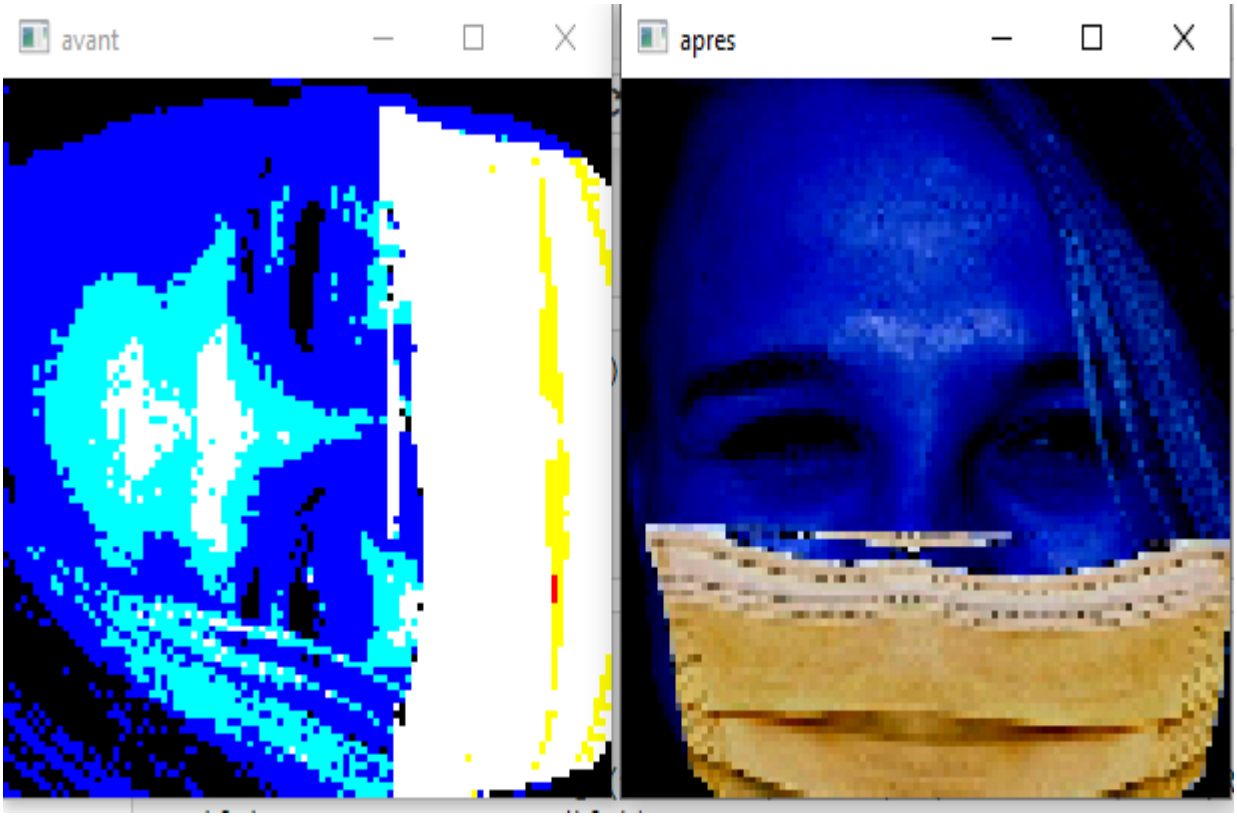


Figure 27 : Illustration d'un blob 3D.

III.2.3 Formation du modèle de détection de masque

On charge le fichier (.npz) contenant les données, ensuite, le processus de transformation de données des catégories (Mask, No Mask), appelé « one hot encoding » on l'applique sur un ensemble de données pour les transformé en données numériques.

Il est important d'effectuer l'encodage des fonctionnalités, car les algorithmes du deep learning ne traitent que des données numériques et non pas les données sous forme de texte.

Chaque catégorie est représentée par un type binaire contenant soit 0 ou 1. Le zéro représente la non-existence d'une catégorie, et le un représente sa présence (39) .

L'ensemble de données est divisé en deux parties : l'ensemble de données d'entrainement et l'ensemble de données de test, cette distribution est bien illustrée dans la figure 28.

L'ensemble de données de formation doit être composé de 80% d'images pour entraîner l'algorithme efficacement pour avoir une bonne précision de prédiction.

L'ensemble de données de test doit être composé de 20% d'images pour tester la précision de prédiction de l'algorithme.

Cette division donne 4981 images pour l'entrainement et 1246 images pour le test.

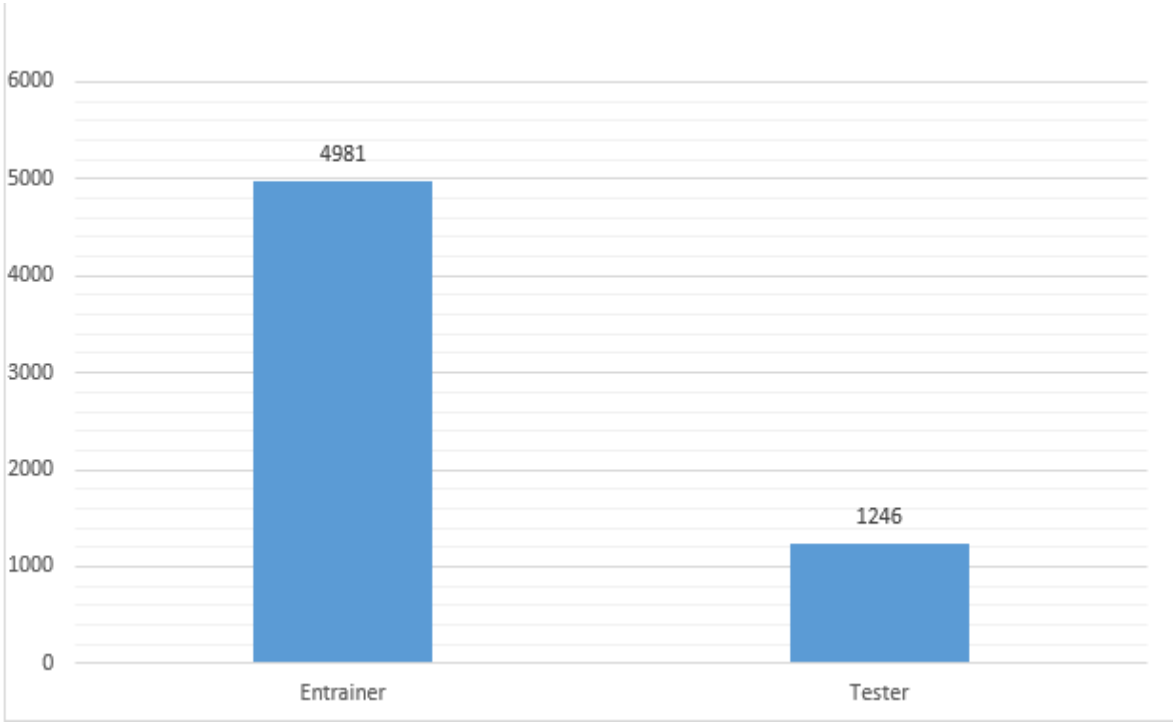


Figure 28 : Histogramme de la distribution de la base de données selon l'utilité.

III.2.3.1 Réseau de neurones convolutionnels

Dans ce projet, on a utilisé un CNN composé de trois couches de convolution, chaque d'entre elles est suivie par une couche de max-pooling, ensuite la sortie de ces couches est aplatie (flattened) et au final on ajoute deux couches entièrement connectées.

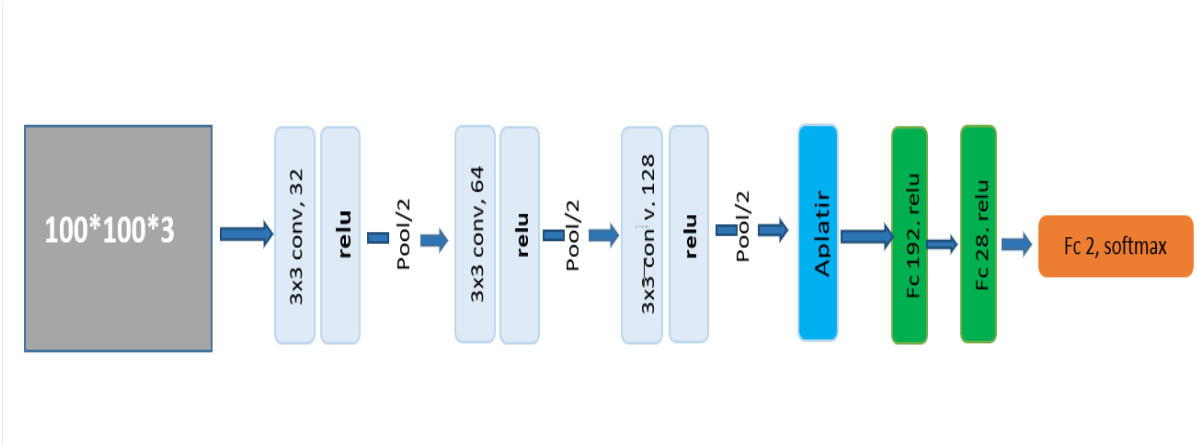


Figure 29 : L'ensemble des couches du réseau CNN proposé.

La première couche convolutif contient 32 filtres de dimension 3*3 pixels, la deuxième contient 64 filtres de même dimension et la troisième contient 128 filtres.

La première couche est responsable de capturer les caractéristiques de bas niveau, telles que la couleur, la coordination de gradient, les bords et les angles.

Lorsque la l'architecture passe par une autre couche convolutive, la sortie de la première couche devient l'entrée de la deuxième couche.

Avec plus de couches ajoutées, l'architecture s'adapte à des caractéristiques de haut niveau, comme une combinaison de courbes et de bords droits.

On associer chaque couche de convolution avec la fonction d'activation RELU qui rétablit toutes les valeurs négatives des pixels par zéro. Son but est de présenter la non-linéarité dans le modèle CNN puisque la plupart des données du monde réel seraient non linéaires.

Au fur et à mesure que le modèle se construit, en appliquant toutes ces opérations, et en ajoutant plus de couches convolutionnelles, l'architecture passe par des cartes d'activation qui produisent des caractéristiques plus convoluées pour mieux comprendre les caractéristiques des visages humains.

La couche de regroupement (max-pooling) sert à réduire la taille spatiale des caractéristiques convoluées. Dans ce fait, la puissance de calcul requise diminuera progressivement grâce à la réduction de la dimensionnalité.

La sortie de la couche aplatie est l'entrée pour les couches entièrement connectées, la première couche contient 192 neurones et la deuxième couche contient 28 neurones.

La fonction d'activation de Soft Max a été utilisée après les couches entièrement connectées. Elle fonctionne en transférant un vecteur de nombres dans un vecteur de probabilités.

Soft Max permet de normaliser les résultats et de les convertir en probabilités dont la somme doit être égale à 1.

L'objectif principal d'une couche entièrement connectée est l'utilisation des caractéristiques de haut niveau pour distinguer les personnes portant un masque des personnes non portantes du masque dans l'image d'entrée (40).

Pendant le processus d'entrainement, l'optimiseur Adam a été appliqué au modèle CNN. Adam est un algorithme d'optimisation du taux d'apprentissage adaptatif qui a été proposé spécialement pour la formation des réseaux de neurones profonds.

Cet optimiseur fonctionne en calculant chaque taux d'apprentissage en fonction des différents paramètres du modèle. Ainsi, l'utilisation de l'algorithme d'optimisation Adam, a été très bénéfique pour cette méthodologie, car il utilise les évaluations du premier et du second moment du gradient pour acclimater le taux d'apprentissage pour chacun des poids du réseau neuronal (41)**.** Il prend le taux d'apprentissage qui est fixé à 0,001, prend aussi la fonction de perte comme arguments.

La fonction de perte est une méthode d'évaluation de la mesure que notre algorithme utilise pour modéliser notre ensemble de données, elle évolue l'écart entre les prédictions faites par le réseau neuronal, et les valeurs réelles des observations utilisées pendant l'entrainement. Plus le résultat de cette fonction est minimisé, plus le réseau est performant

Lorsque on modifie l'algorithme pour améliorer le modelé, cette fonction indique si l'approche est bonne ou pas.

La fonction de pertes utilisée est appelé Cross-entropie catégorique, car on a une classification multi-classes (deux classes), où il doit y avoir le même nombre de nœuds de sortie que les classes.

La sortie de la couche finale doit passer par l'activation soft max afin que chaque nœud émet une valeur de probabilité entre 0 et 1 (42).

On aura le résumé du modèle tel que montré dans la figure 30.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 100, 100, 32)      896

 max_pooling2d (MaxPooling2D  (None, 50, 50, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 50, 50, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 25, 25, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 25, 25, 128)       73856

 max_pooling2d_2 (MaxPooling  (None, 12, 12, 128)      0
 2D)

 flatten (Flatten)           (None, 18432)             0

 dense (Dense)               (None, 192)               3539136

 dense_1 (Dense)             (None, 28)                5404

 dense_2 (Dense)             (None, 2)                 58

=================================================================
Total params: 3,637,846
Trainable params: 3,637,846
Non-trainable params: 0
_________________________________________________________________
```

Figure 30 : Résumé du modèle.

Ensuite, on commence l'entrainement avec 20 itérations (epochs) (Le nombre d'itérationsest un hyper-paramètre qui définit le nombre de fois que l'algorithme d'apprentissage va travailler sur l'ensemble des données d'apprentissage) pour obtenir les meilleurs résultats de précision.

Les figures ci-dessous illustrent des graphiques de comparaison de la perte et la précision entre l'entrainement et le test du modèle.

Ces graphes montrent qu'en augmentant le nombre d'itérations, la perte diminue, tandis que la précision augmente.

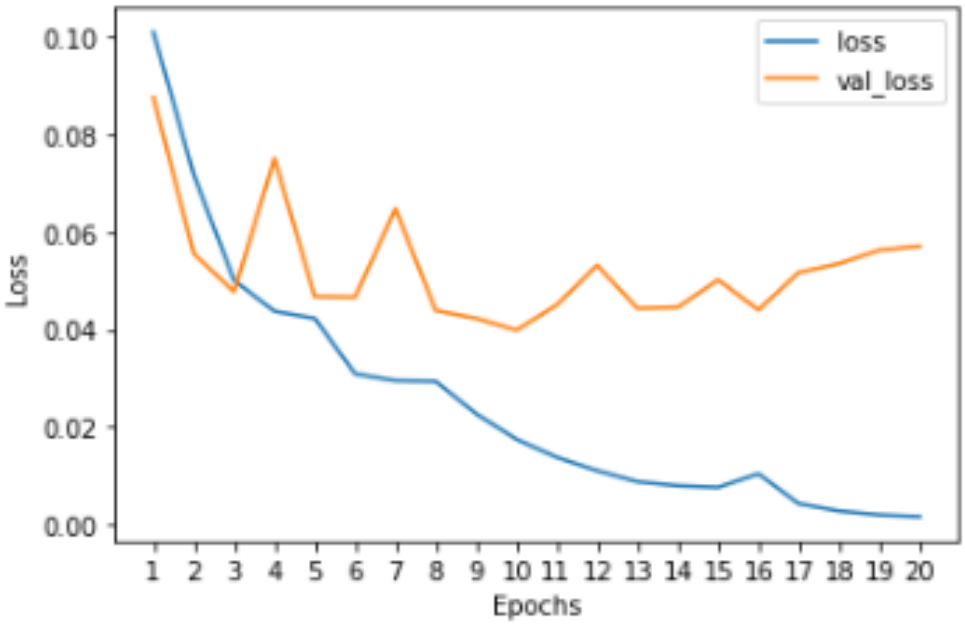


Figure 31 : Comparaison entre les courbes de perte dans l'entrainement et le test.

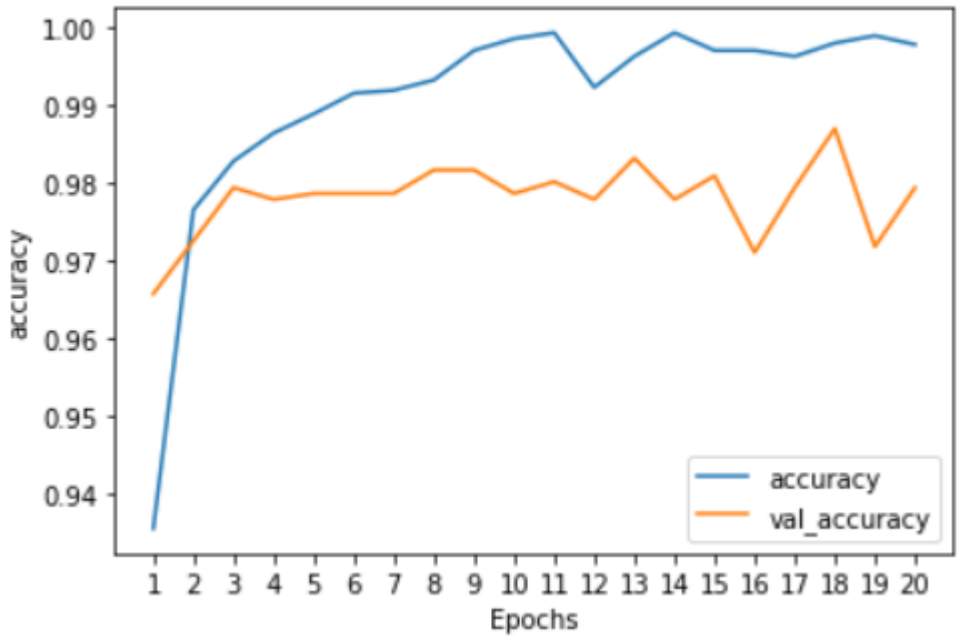


Figure 32 : Comparaison entre les courbes de précision dans l'entrainement et le test.

III.2.4 Implémentation du détecteur de masque

Pour implémenter le détecteur on utilise OpenCV, on charge le modèle de détection du visage ainsi que le modèle de détection du masque, ensuite on mit en place la fonction qui sert à :

- Extraire le blob et le faireentrerdans le détecteur de visage, afin de les détecter dans l'image d'entrée.

- Boucler à travers les visages détectés et les redimensionner à 100*100 pixels, les transformer en vecteur et les stocker dans une liste.

- Extraire les coordonnées des boites de délimitations et les stocker dans une liste.

- Faire entrer la liste des visages dans le détecteur de masque,afin de détecter l'existence ou la non existence du masque.

- Dessiner les boites de délimitation avec leurs étiquettes sur l'image, ainsi que la probabilité de détection à l'aide de OpenCV, s'il la personne porte un masque, la couleur de la boite est verte, sinon la couleur devient rouge.

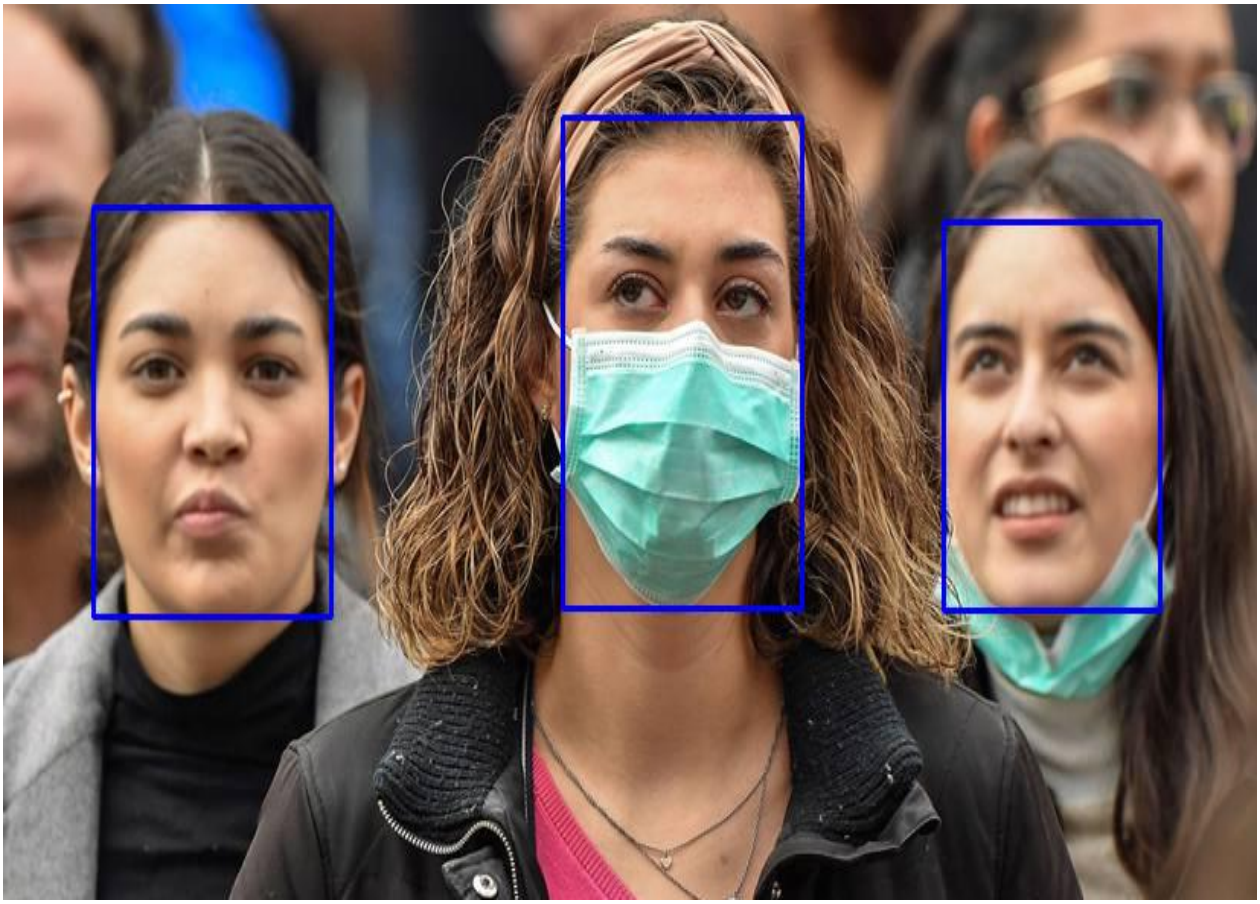


Figure 33 : Application de détection de visage

III.3 Détection de la distance sociale et le comptage des personnes avecYOLOv3 :

Dans la réalisation de notre projet, on a utilisé le model préformé nommé YOLOv3 pour la détection de personnes.

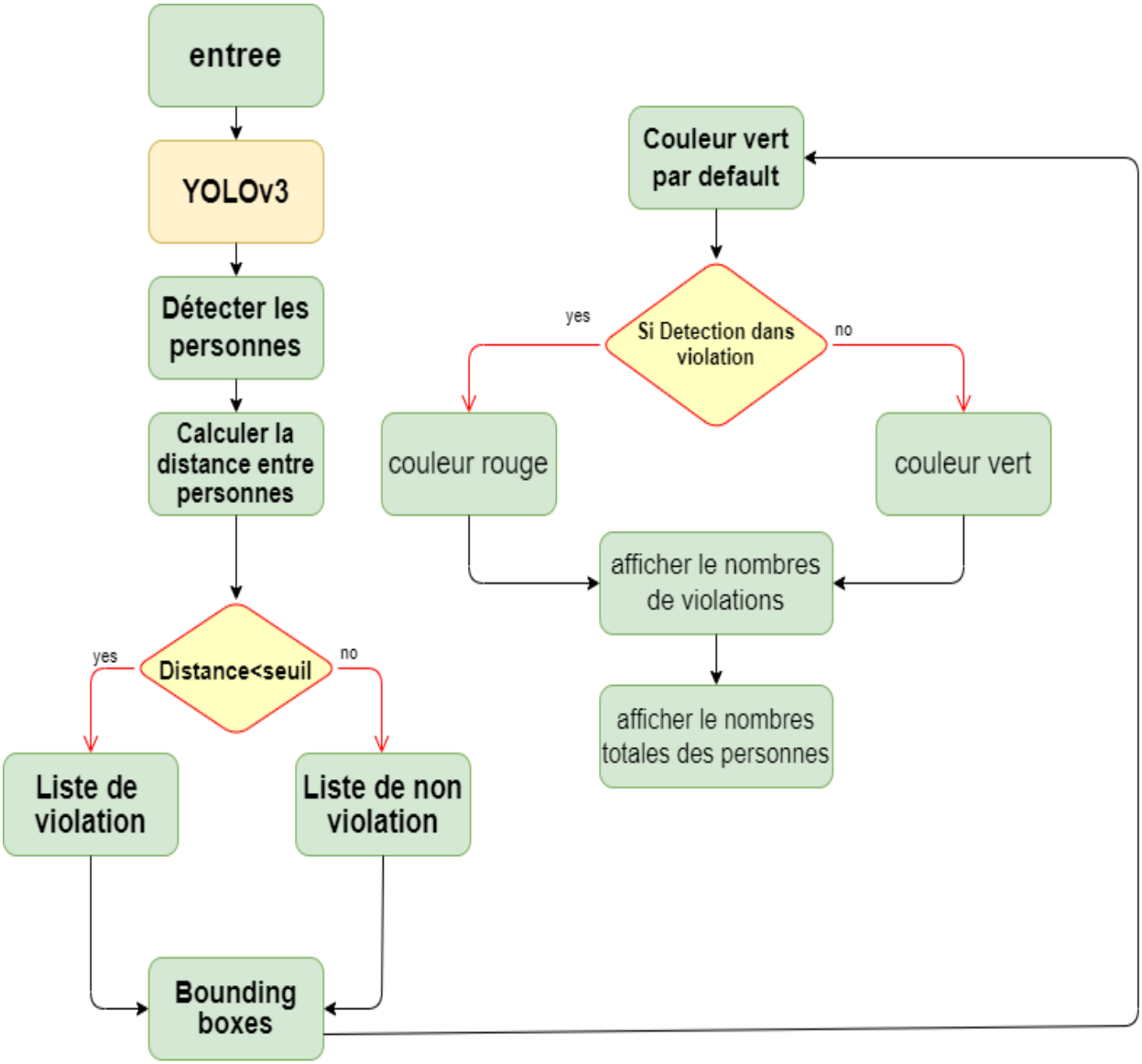


Figure 34 : Illustration de l'architecture du projet

YOLOv3 est un algorithme de détection d'objets en tempsréel qui identifie des objets spécifiques dans des images, vidéos et même dans des flux en direct.

Il utilise des caractéristiques apprises par un réseau neuronal convolutif profond, composé de 106 couches, dans le but de détecter un objet. Il a été créé par Joseph Redmon et Ali Farhadi.

YOLO a été formé sur labase de données coco, qui est composé de 80classes dans lesquelles Dans notre projet on s'intéresse seulement à la classe qui est dédiée aux personnes.

III.3.1 Détection de personnes :

En premier lieux, on charge le model de YOLO en utilisant le module DNN de OpenCV qui prend deuxparamètres « le chemin des poids et chemin de la configuration ».

Comme deuxième étape, on définit la classe qu'on veut détecter,dans notre cas c'est la classe des personnes.

Ensuite on construit un blob à partir de l'image d'entrée, puis on effectue un passage au détecteur d'objets YOLO. L'algorithmedu modèle YOLO produitenviron 10647détections, dont la plupart sont non pertinentes/redondantes.



Figure 35 : Les détections du model YOLO appliqué sur une image

Maintenant onélimine toutes les boîtes ayant une faible probabilité de détection d'objets. Cela est fait par le seuil de confiance. On gardeseulement les boîtes qui ont une probabilité deplus du seuil.Cette étape permet de se débarrasser des détections anormales des objets.

Après l'application du seuil on aura un nombre de boîtes considérablement réduit, commele montre la figure 37, mais on a besoin d'une boite de d'élimination pour chaque objet détecté, cette boîte de délimitation est calculée en utilisant la suppressionnon-max.

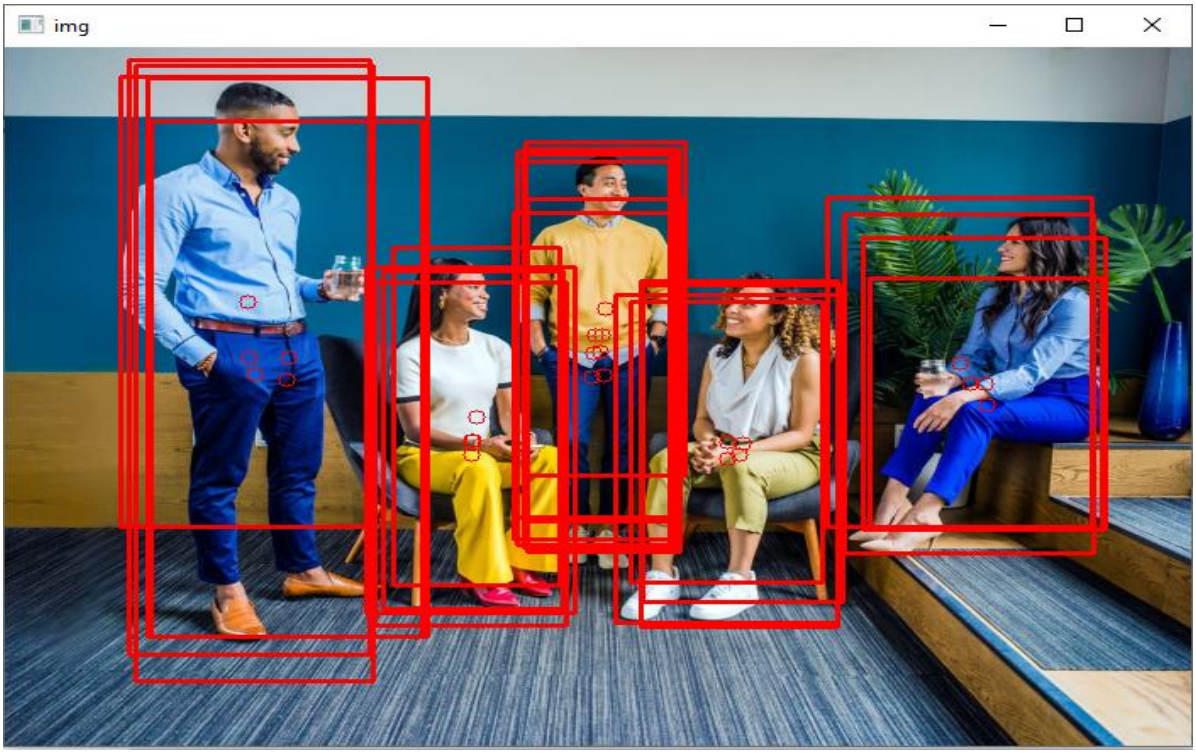


Figure 36 : L'image après application du seuil de confiance

La suppression non maximale fait appel à un concept appelé intersection sur union « IoU ». Elle prend en entrée deux boîtes et, comme son nom l'indique, calcule le rapport entre l'intersection et l'union des deux boîtes.

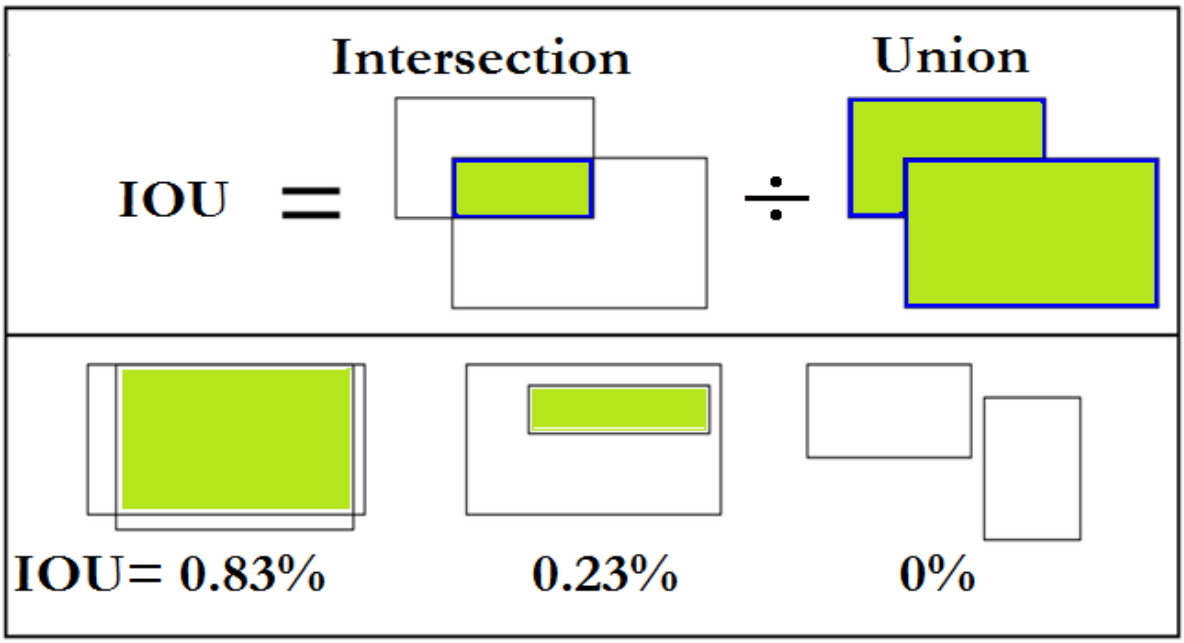


Figure 37 : Principe du fonctionnement de l'IoU

Après avoir défini l'IoU, la suppression non maximale fonctionne comme suit :

Elle sélectionne la case ayant la plus grande probabilité de détection et retire toutes les autres qui ont une valeur d'IoUélevée, après cochez la case sélectionnée comme étant « traitée », répéter jusqu'à ce qu'on se retrouve avec une boîte pour chaque objet détecté.

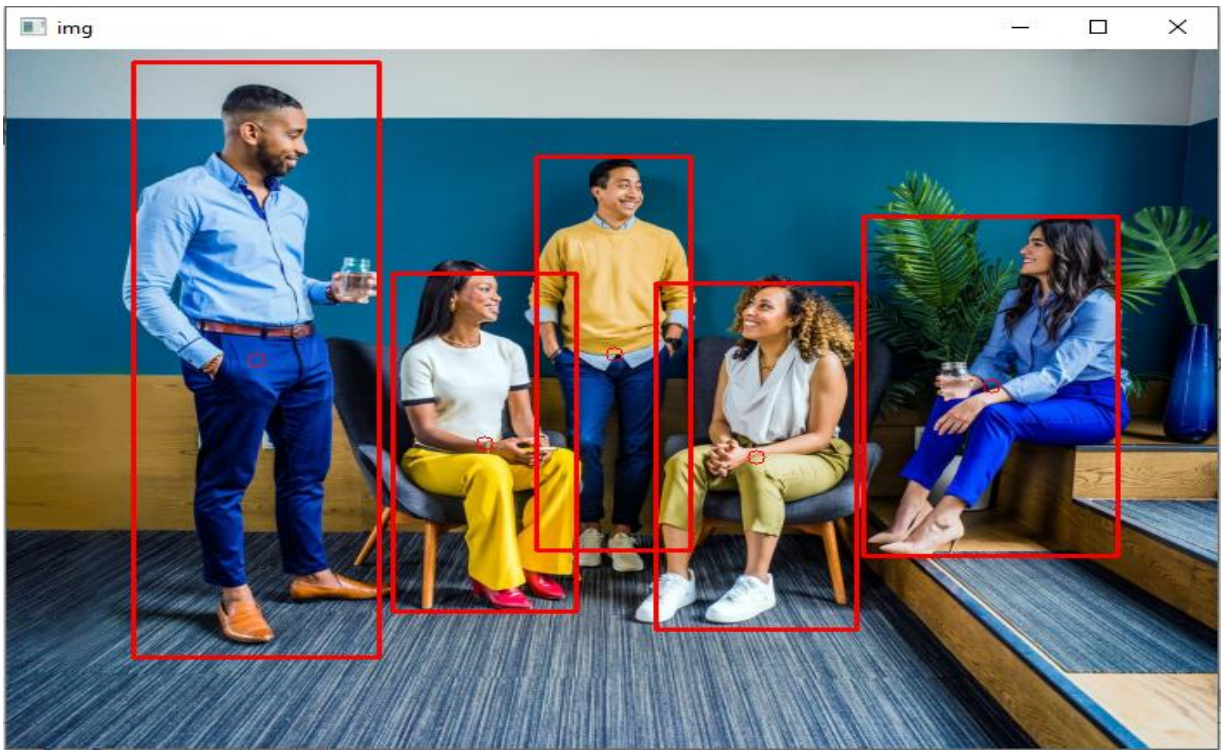


Figure 38 : L'image après application de la suppression non-max

III.3.2 Comptage des personnes et Distance sociale

Apres avoir détectél'existencede personnes dans l'image d'entrée, on extrait les centres de gravité de chaque personne détectée, puis on calcule la distance euclidienne entre ces centres.

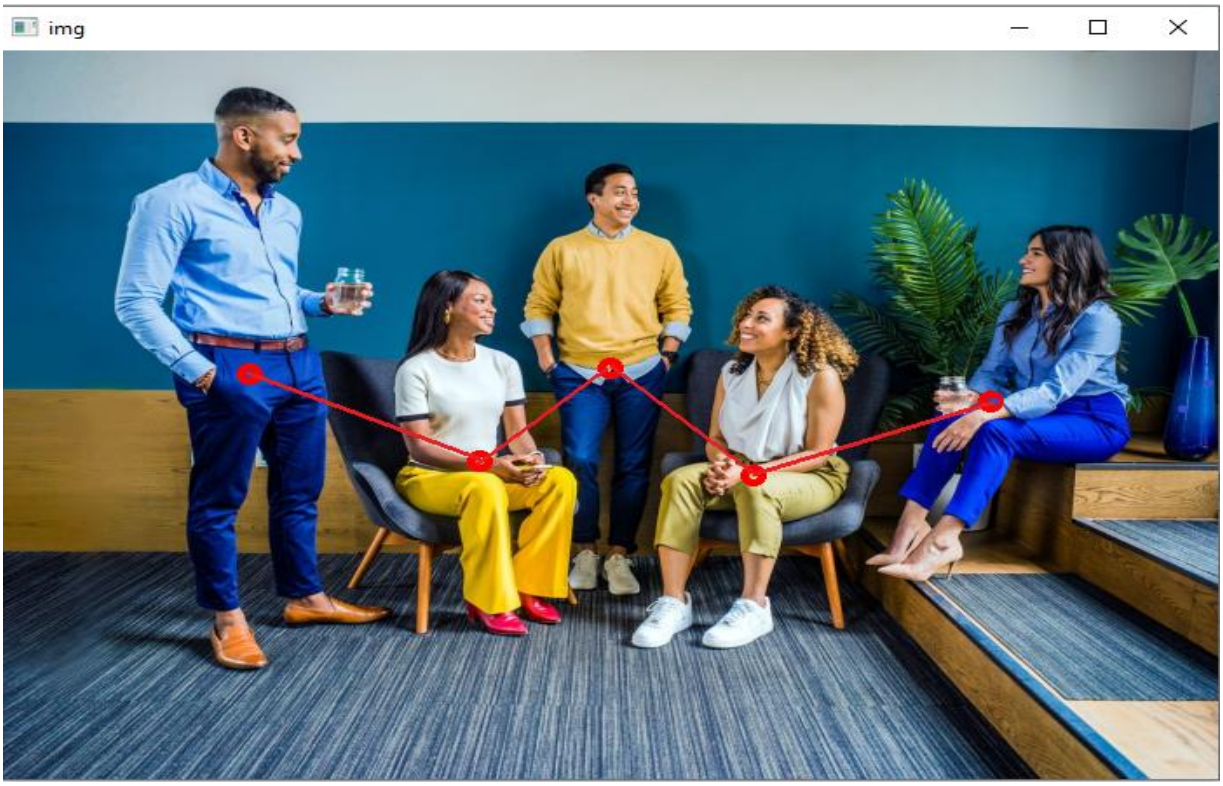


Figure 39 : Les centres de gravité de chaque personne détectée

Ensuite, on pose une condition sur la distance, si elle n'est pas satisfaite, on transfère lesdétections vers une liste de violation.

En fin, on extrait les case de délimitation et on fixe leurs couleurs par default en vert, si le case se trouve dans la liste de violation donc la couleur devient rouge.

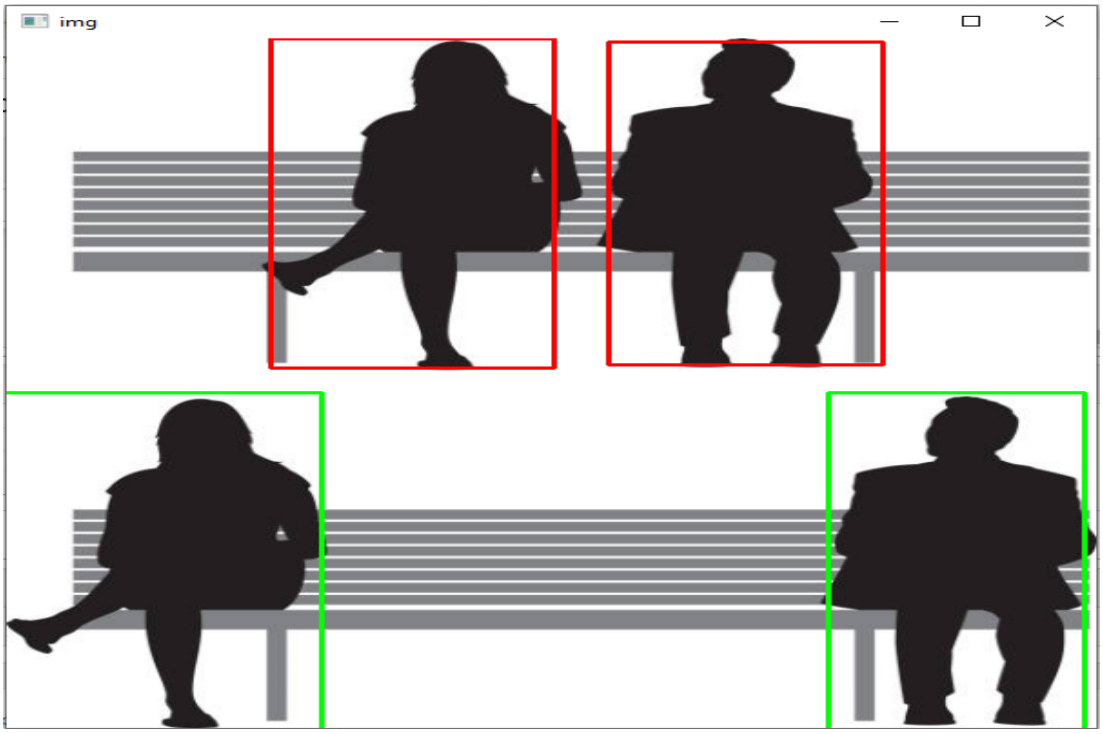


Figure 40 : Application du seuil de distance sur les personnes détéctées

III.4 Interface graphique

Pour une utilisation optimale etfacile des fonctions de détectionde violations, concernant le port du masque, la non-application de la distanciation social,le non-respect de la limite autoriséde nombre de personnes,nous avons décidé de concevoir une interface graphique, en utilisant les bibliothèquesTkinter et PyInstaller de Python.

La première permet de crée l'interface, et à la l'aide de la seconde on peut regrouperl'application et toutes ses dépendances en un seul paquet,dans le but de faciliter la tache pour l'utilisateurà fin qu'il puisse l'exécuter sans avoir besoin d'un environnement de programmation ou d'un interpréteurinterpréteur Python.

PyInstaller parcourt le script Python,il analyse le code pour découvrir tous les autres modules et bibliothèques dont le script a besoin pour s'exécuter. Il collecte ensuite des copies de tous ces fichiers y compris l'interpréteur Python actif et les place avec notre script dans un seul dossier, avec fichier exécutable.

Pour résumer le fonctionnement de notre applications de gestion de personnes nous illustrons ci-dessous un diagramme explicatif

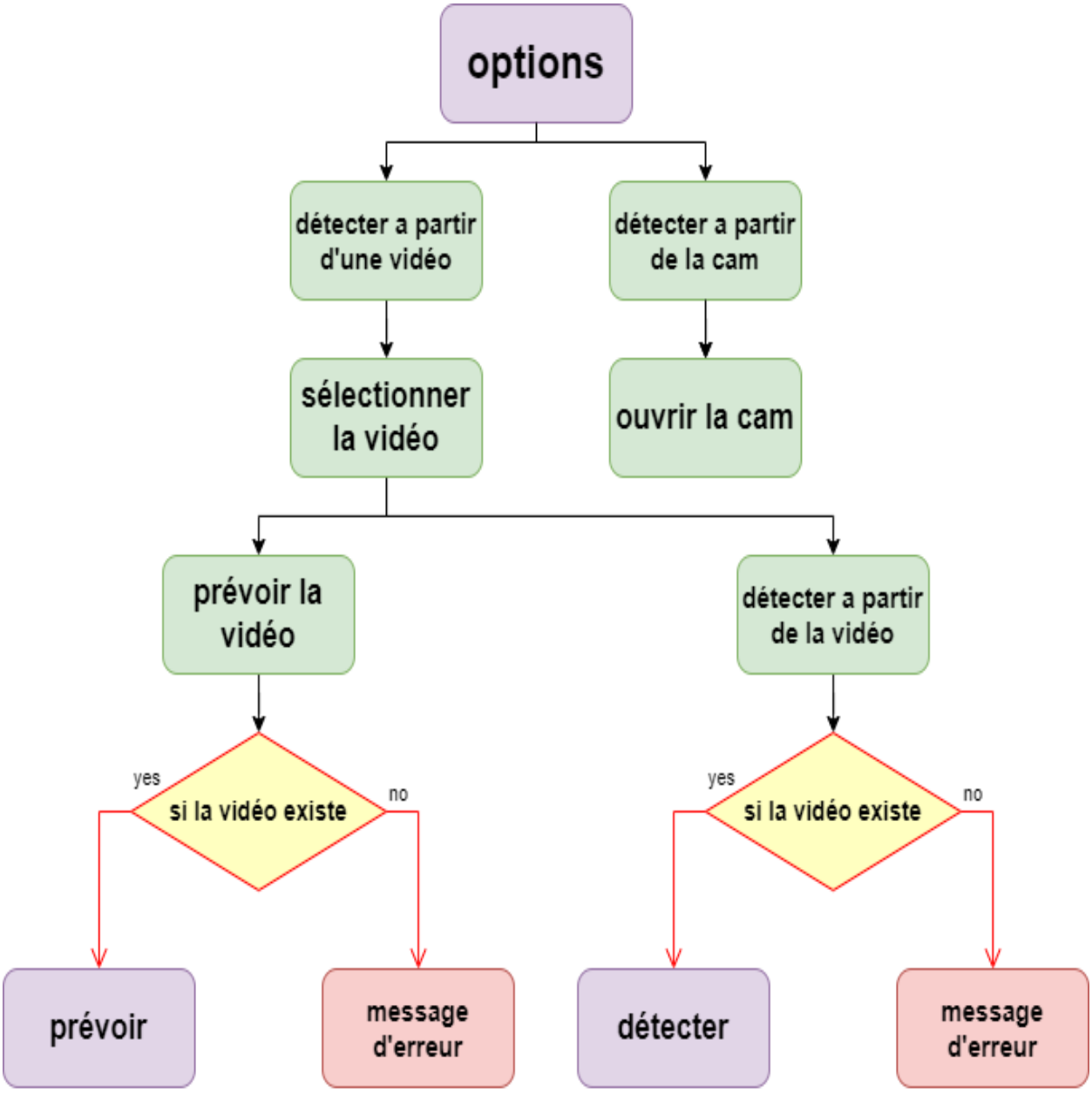


Figure 41 : Diagramme de fonctionnement de l'interface graphique

III.4.1 Fonctionnement et design de l'interface

III.4.1.1 Lancement de l'interface

Dés qu'on clique sur l'exécutable, le système lance notre application, et une fenêtre d'accueil s'affiche.

A travers cette fenêtre l'utilisateur peut choisir entre deux options, détecter à partir d'une vidéo ou bien par camera.



Figure 42 : Fenêtre d'accueil

III.4.1.2 Détecter à partir d'une vidéo

En choisissantl'option de détection à partir d'une vidéo, le système nous amène à la fenêtre vidéo qui contient un champ, dont lequel on colle le chemin de la vidéo souhaitée, ou bien tout simplement en cliquant sur le bouton sélectionner et la faire parcourir.



Figure 43 : Fenêtre pour l'option de détection à partir d'une vidéo

Au cas où le chemin de la vidéo est vide ou non valide, on recevra des messages d'erreur. Ces messages sont affichés à l'aide du module message-box de Tkinter.

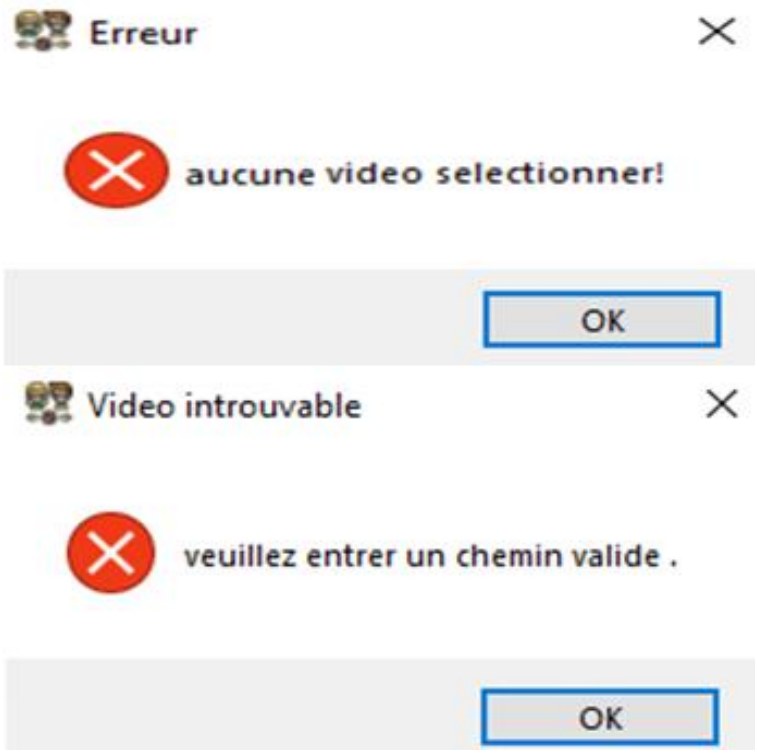


Figure 44 : Les messages d'erreur

Sur la même fenêtre de détection à partir d'une video, ona placé aussi quatre boutons :

- Le bouton **Sélectionné ;** qui nous donne la possibilité de parcourir une vidéo à partir du stockage interne.

- Le boutons **Visualiser ;** pour lire la vidéo choisie avant de lui appliquéle programme, pour s'assurer que nous avons bien choisis la bonne vidéo. si le chemin est vide ou non valide on recevra un des messagesd'erreurs cités la figure ci-dessus.

- Le bouton **Détecter ;** nous permet de lire la vidéo, et d'appliquer à chaque frame notre programme de détection et de comptage de personnes.

- Le bouton **Retour ;** pour revenir vers la fenêtre principale.

La fenêtre qui s'affiche lorsqu'on clique sur le bouton sélectionner est la suivante :

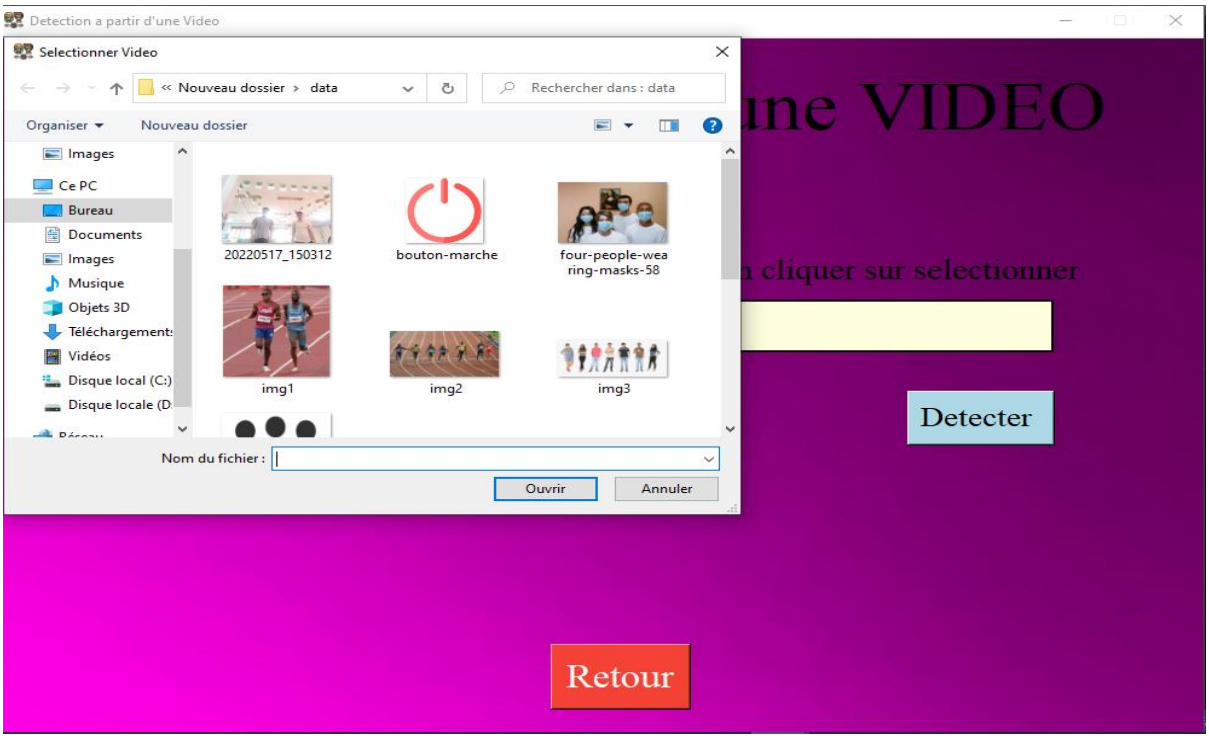


Figure 45 : L'affichage après avoir cliquer sur Sélectionner

La fenêtre qui s'affiche lorsqu'on clique sur le bouton Visualiser est la suivante :

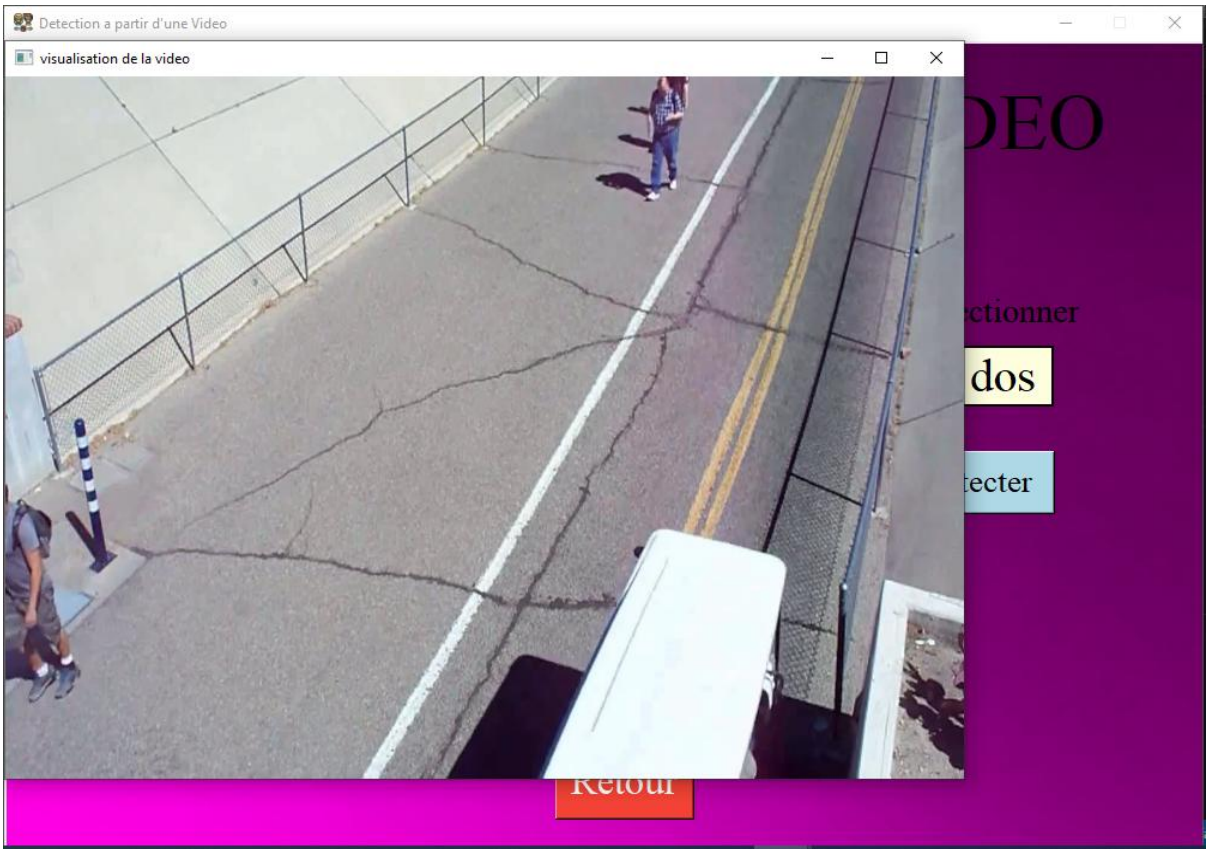


Figure 46 : L'affichage après avoir cliqué sur Visualiser

La prochaine étape à faire, est de lancé notre programme, il suffit simplement de cliquer sur le bouton **détecter.** Une petite fenêtre apparaitra indiquant le début de la détection après quelque secondes de patience, le temps du chargement, comme l'indique la figure ci-dessous :

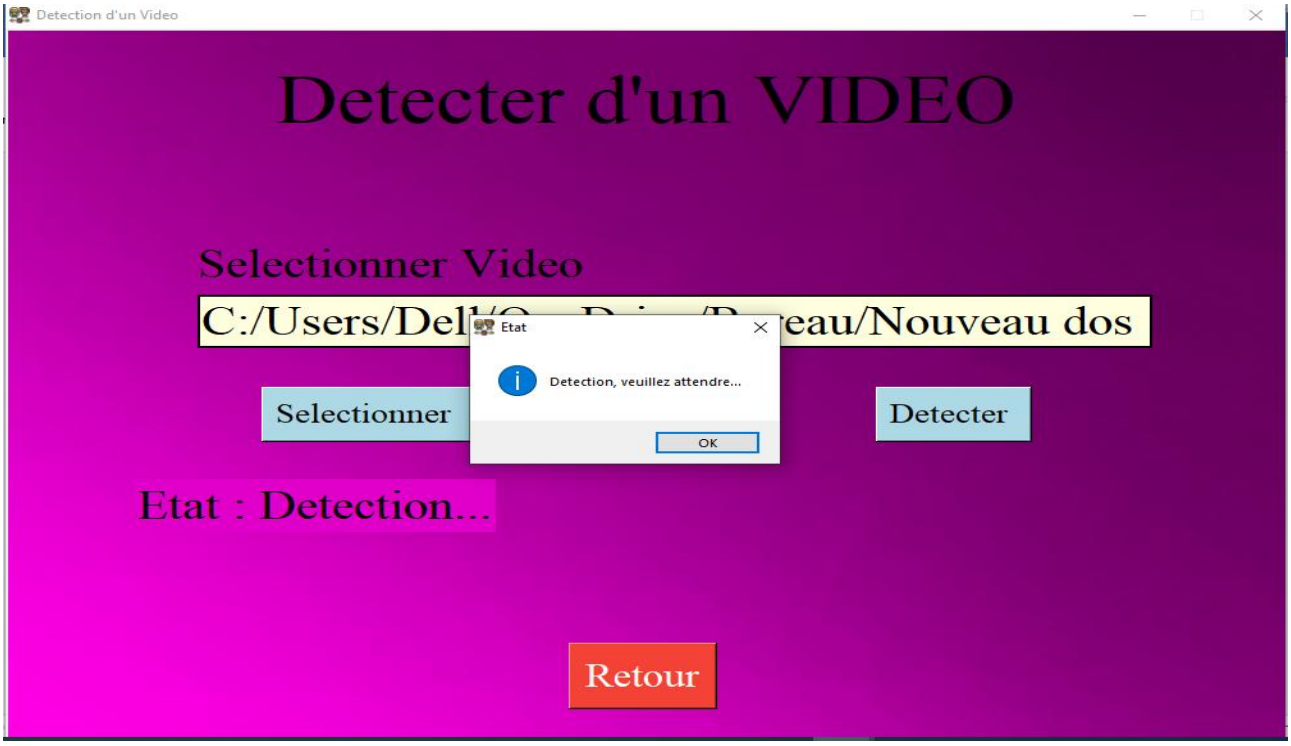


Figure 47 : L'affichage après avoir cliqué sur Détecter

Quand la détection s'achève, on reçoit une information que c'est fait, en même temps on peut toujours refaire les mêmes étapes avec d'autres video :



Figure 48 : L'affichage après la fin de la détection et du comptage par vidéo

Lorsqu'on clique sur le bouton Retour une petite fenêtre s'affiche, elle demande à l'utilisateur de bien vouloir confirmer son choix pour quitter :

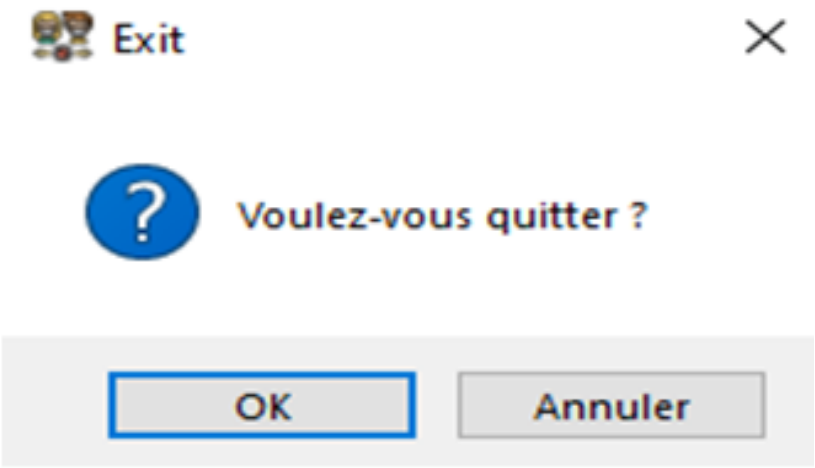


Figure 49 : La fenêtre de confirmation pour quitter

III.4.1.3 Détecter par camera

En choisissantl'option de détection par camera, le système nous amène à la fenêtre camera,on y trouve deux boutons.

Un bouton de retour vers la fenêtre d'accueil, et le deuxième permet ouvrir la webcam, pour commencer la détection en utilisant le module VideoCapture d'OpenCV qui ouvre la caméra et applique les détecteurs du masque et de la distance en temps réel.



Figure 50 : Fenêtre pour l'option de détection par caméra

Dès qu'onclique sur le bouton **ouvrirlacam**, une petite fenêtreapparaitra contenant un message, qui nous demande de patientez le temps de lancer la webcam, pourque la détection débute.

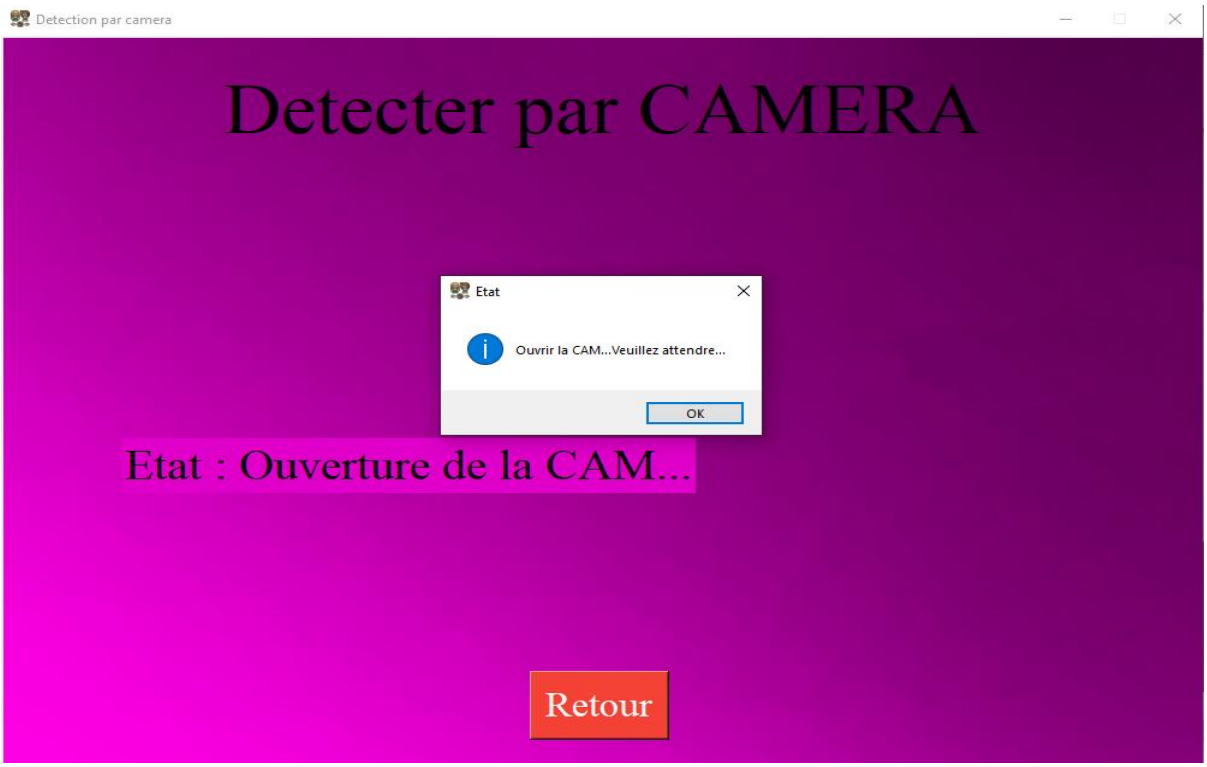


Figure 51 : L'affichage après avoir cliqué sur ouvrir la cam

Quand la détection s'achève, on reçoit une information que c'est fait,en même temps on peut toujours rouvrir la cam pour relancer la détection; comme le montre la figure suivante :



Figure 52 : L'affichage après la fin de la détection et du comptage par cam

III.4.2 Regrouper l'application

En utilisant Pyinstaller et le logiciel NSIS, on regroupe tous les fichiers, les scripts, les interpréteurs et les bibliothèques utilisées durant la création de notre application dans un seul fichier exécutable.

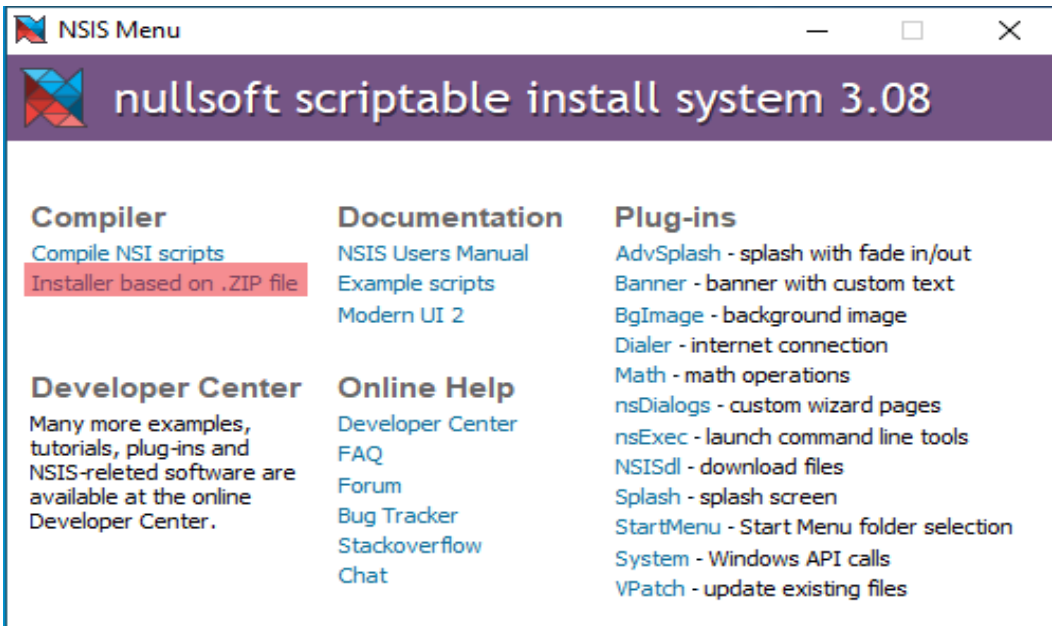


Figure 53 : La fenêtre du logiciel NSIS

A l'aide de la fonction d'installateur basé sur le fichier .zip , on sélectionne le fichier contenant l'exécutable et les autresdépendances et on les compresse dans un fichier avec extension .zip et on clique « Generate » pour générer le fichier d'installation.

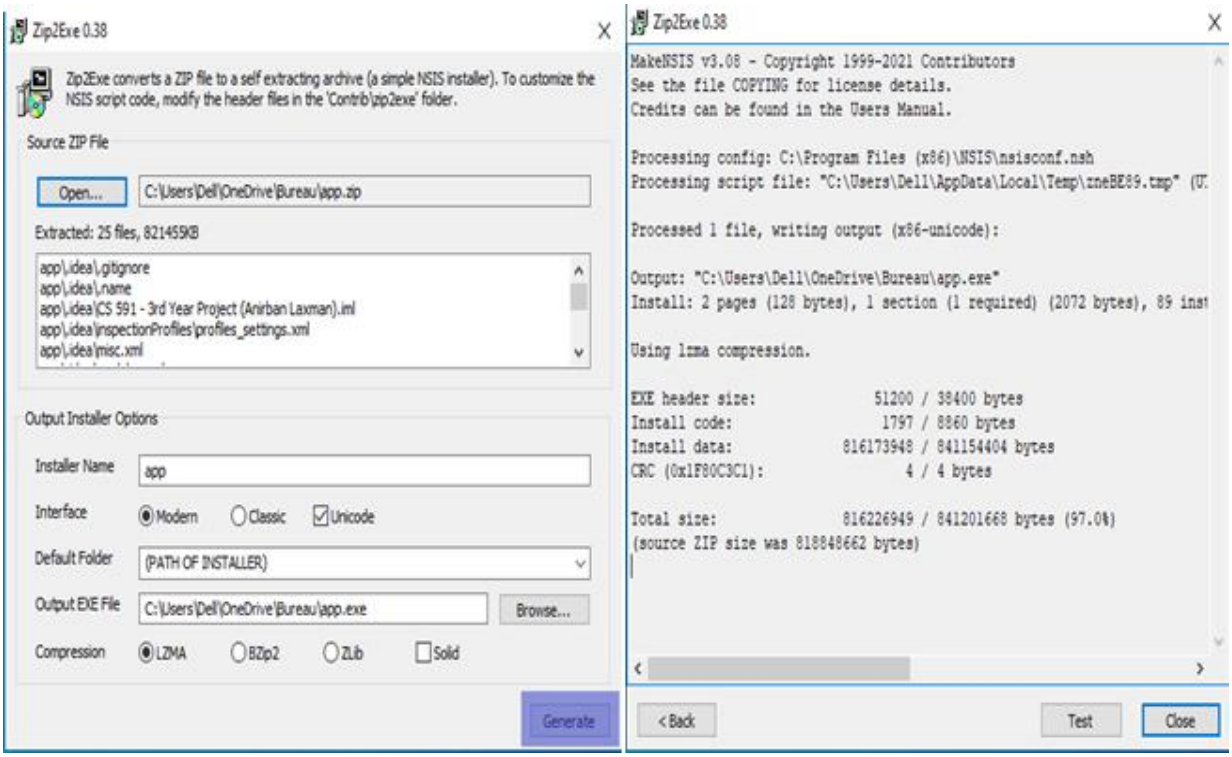


Figure 54 : Génération du fichier d'installation

Apres avoir générer le fichier d'installation, il reste que de choisirl'emplacement et de l'installer dans le périphérique de l'utilisateur.

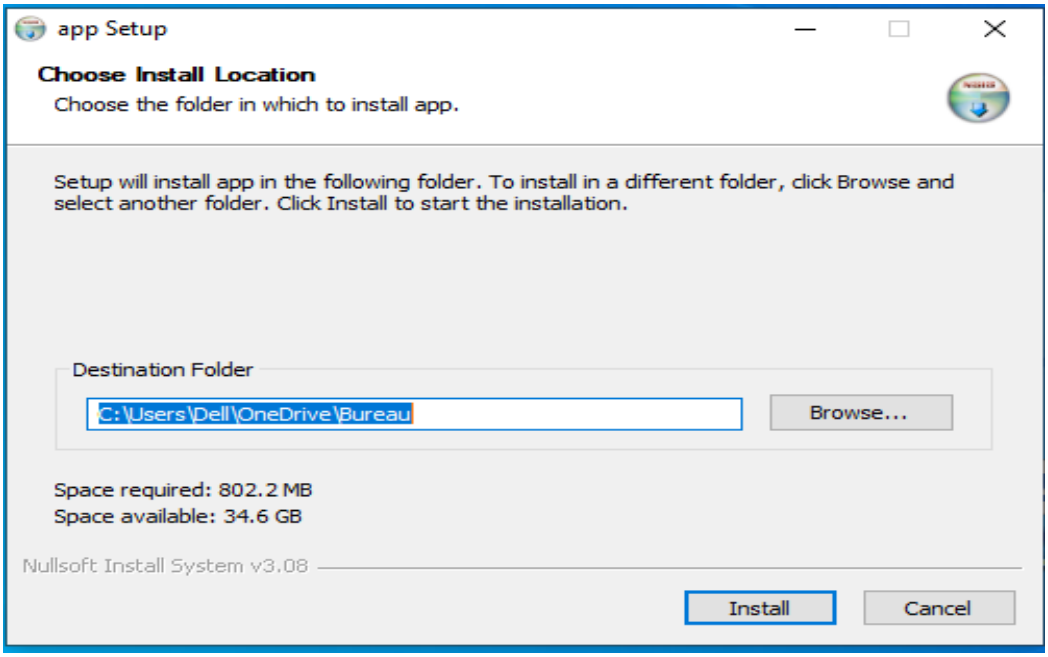


Figure 55 : Installation de l'interface graphique

III.5 Conclusion

Ce chapitre est le noyau de notre projet, dont lequel on a expliqué la procédure de la conception et réalisation de l'application de prévention contre le COVID-19, en détectant les violations concernant le port du masque et la non application de la distanciation social

Dans le but de faciliter à l'utilisateur l'exploitation de notre application, on lui a associé une interface graphique simple et commode, pour l'aider à s'assurer que les gens respectent les règles de sécurité liées à la COVID-19.

Chapitre 4

Tests et discussion

IV. Tests et discussions

IV.1 Introduction

Ce chapitre traite les résultats des tests effectués, lors de la conception de cette application, ces tests permettent d'évaluer son fonctionnement et sa stabilité, et de détecter ses lacunes et ses potentiels problèmes qu'on doit régler pour améliorer sa performance.

IV.2 Les tests de performances

IV.2.1 Tests de détection à partir des images

Pour tester notre modèle, nous l'appliquons d'abord surplusieurs images de personnes portant ou non portant un masque facial, pour expérimenter le taux de réussitedenotre modèle, est-ce qu'ildétecte correctement le masque facial et les personnes, ainsi que calculer leurs nombres et vérifier l'application de la distanciation sociale.

IV.2.1.1 Test 1

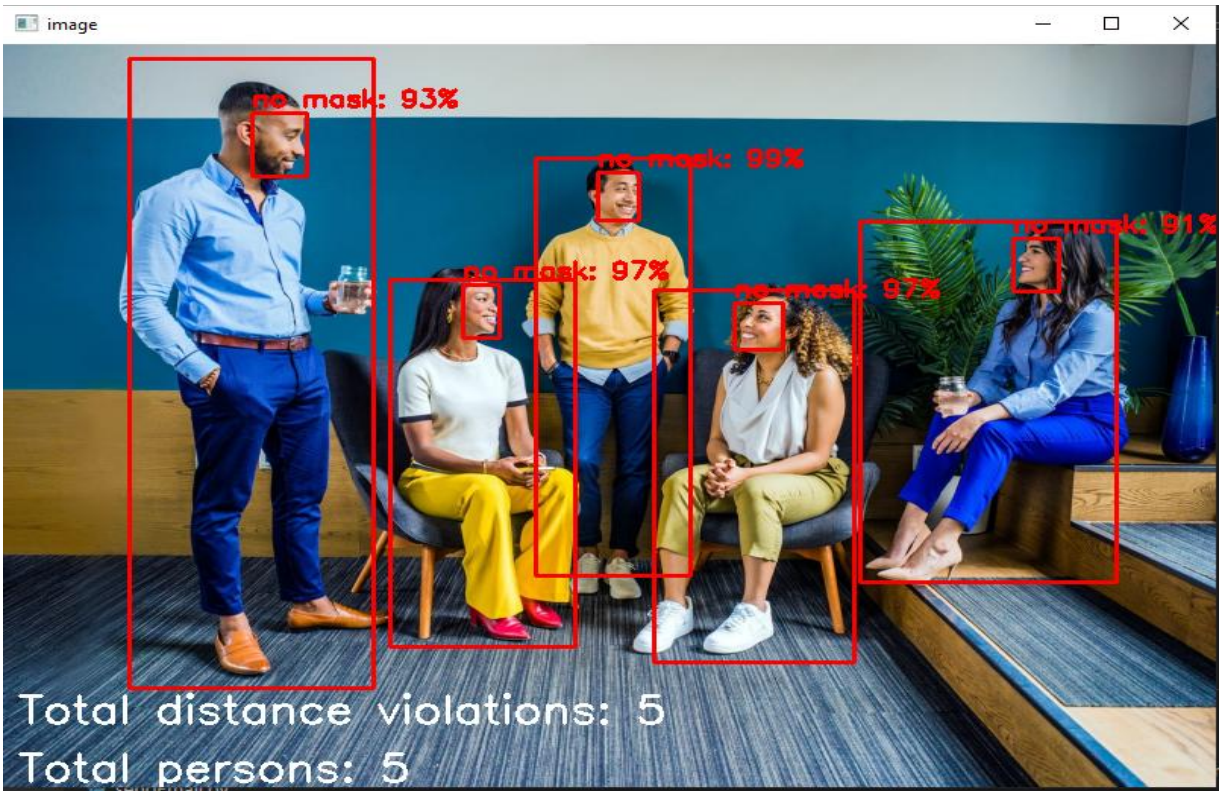


Figure 56 : Test 1 de détection à partir d'une image

Dans ce premier test on peu remarquer que le modèle fonctionne correctement en détectant 5 personnes « Total persons : 5 », comme indiqué dans la figure 58, ces personnes non pas respecter la distanciation social car un message s'affiche « Total distance violations : 5 », aucun d'entre eux porte un masque non plus, en remarquant les cases dans chaque visage qui affiche « no mask ».

IV.2.1.2 Test 2

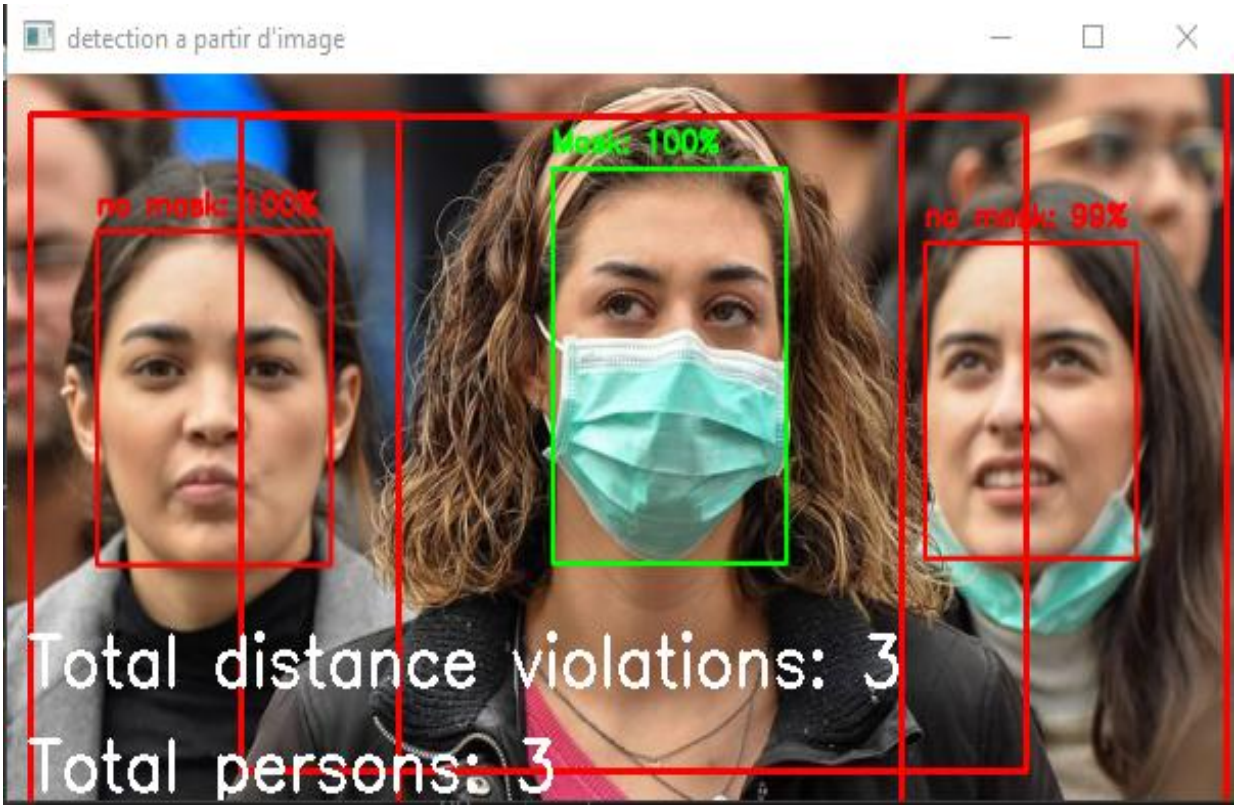


Figure 57 : Test 2 de détection à partir d'une image

Comme nous pouvons le voir dans l'image ci-dessus, le modèle a été capable de détecter 3 personnes correctement, etque celle du milieu porte un masque, son visage est encadré avec une case verte, contrairement aux deux autres entourés par des cases rouges indiquant qu'elles ne portent pas de masque.

Le nombre de violations de la distanciation sociale a été affiché dans l'image de sortie,« Total distance violations : 3 ».

IV.2.2 Tests de détection à partir d'une vidéo:

Afin de tester notre modèle sur des vidéos et c'est ce qui est le plus important, nous utilisons la détection à partir des vidéos de notre application.

Après avoir choisis sélectionné les vidéo et effectué la détection, nous visualisons les résultats des deux tests dans les images ci-dessous.

IV.2.2.1 Test 1

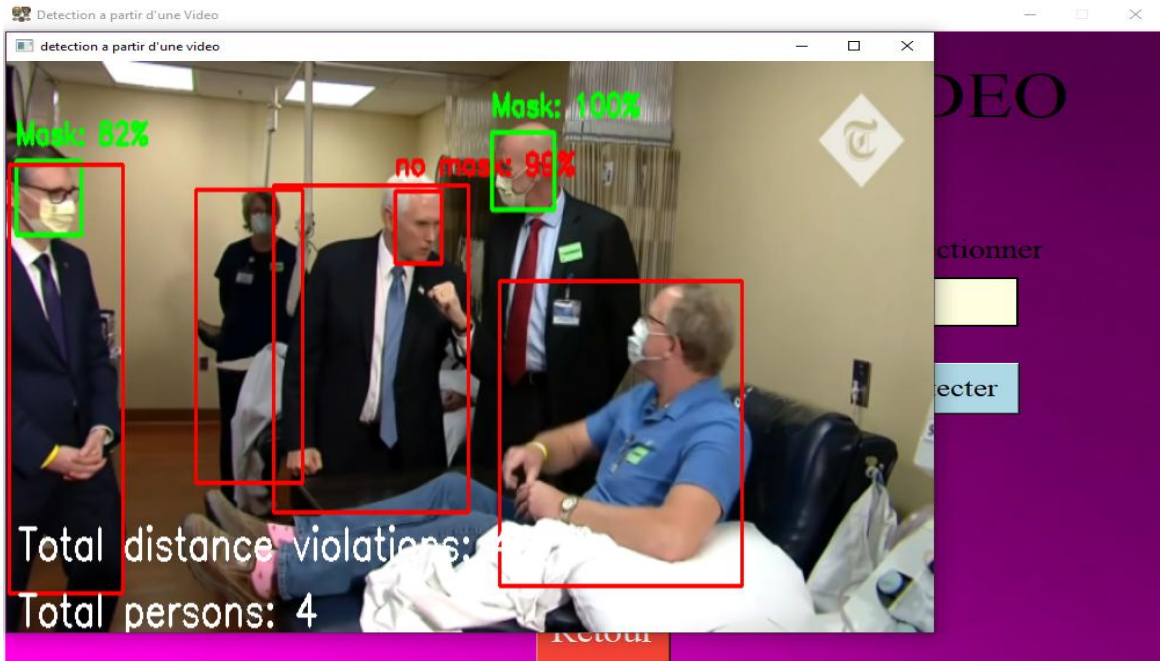


Figure 58 : Test 1 de détection à partir d'une vidéo

Notre application a rencontré deux problèmes pour deux personnes, le premier est que ces deux personnes malgré qu'ils ont bien été détectées, mais leurs masques ne l'était pas, pour la femme de derrière, la cause majeure est son éloignement de la caméra et ainsi que la mauvaise qualité de la vidéo qui a influencé sur le bon fonctionnement du modèle ; pour la personne assise,le masque n'a pas été détecté mais on peut rien reprocher au programme, car son visage était orienté vers l'autre direction. Le deuxième problème, est que le programme n'a pas détecté la personne avec cravate rouge, malgré qu'il ait détecté la présence du masque,.

IV.2.2.2 Test 2

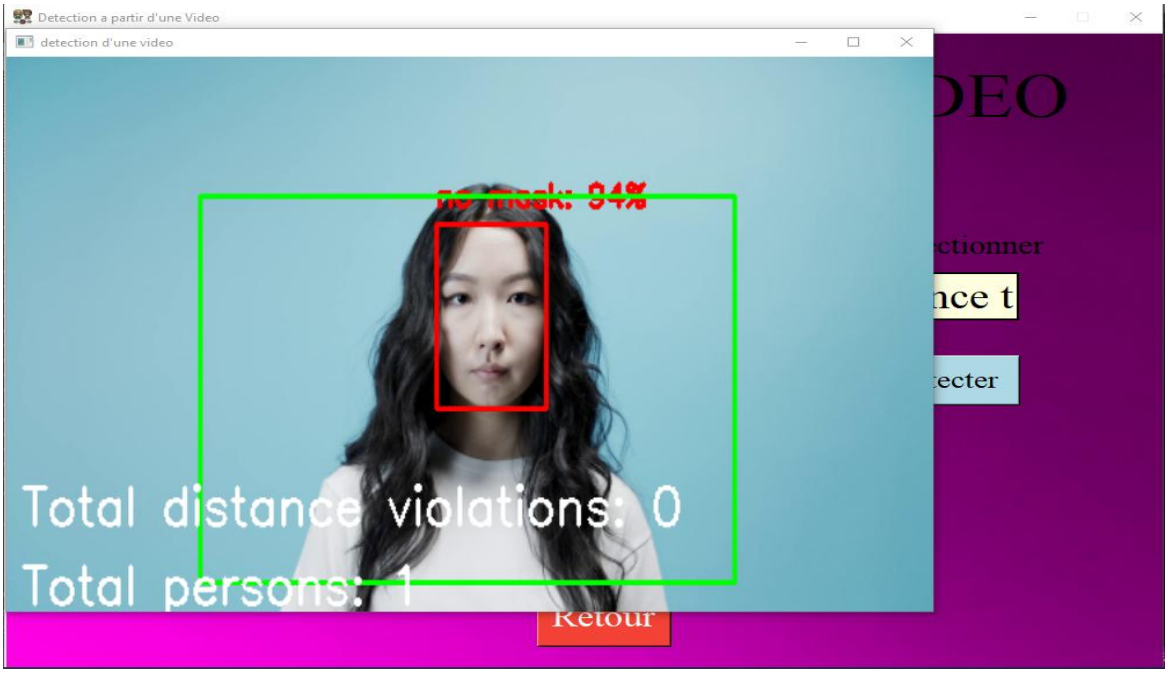


Figure 59 : Test 2 de détection à partir d'une vidéo_1

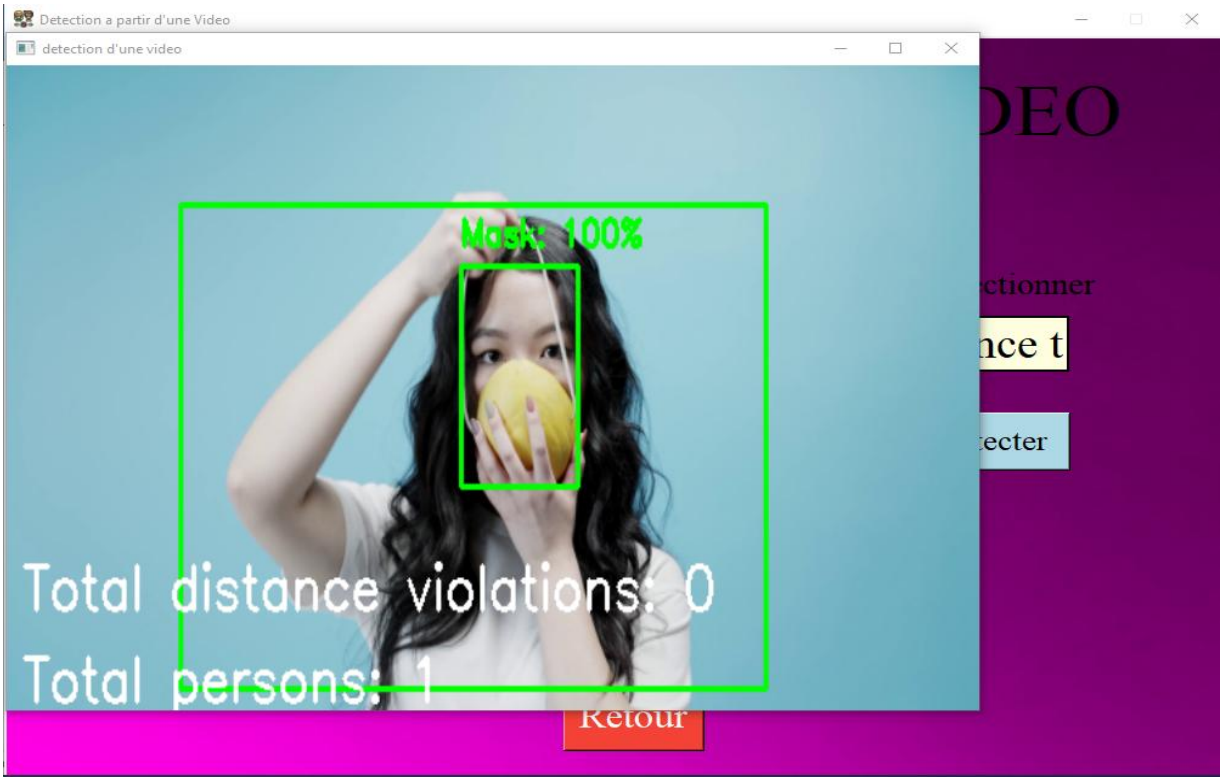


Figure 60 : Test 2 de détection à partir d'une vidéo_2

Pour ce test le modèle a fonctionné parfaitement.

IV.2.3 Tests de détection en temps réel

En utilisant l'option de détection via la caméra de l'application en temps réel, nous avons pu détecter le port des masques faciaux,calculer le nombre total de personnes et dedétecter les violations concernantla distanciation sociale.



Figure 61 : Test de détection en temps réel – non-distanciation et masque

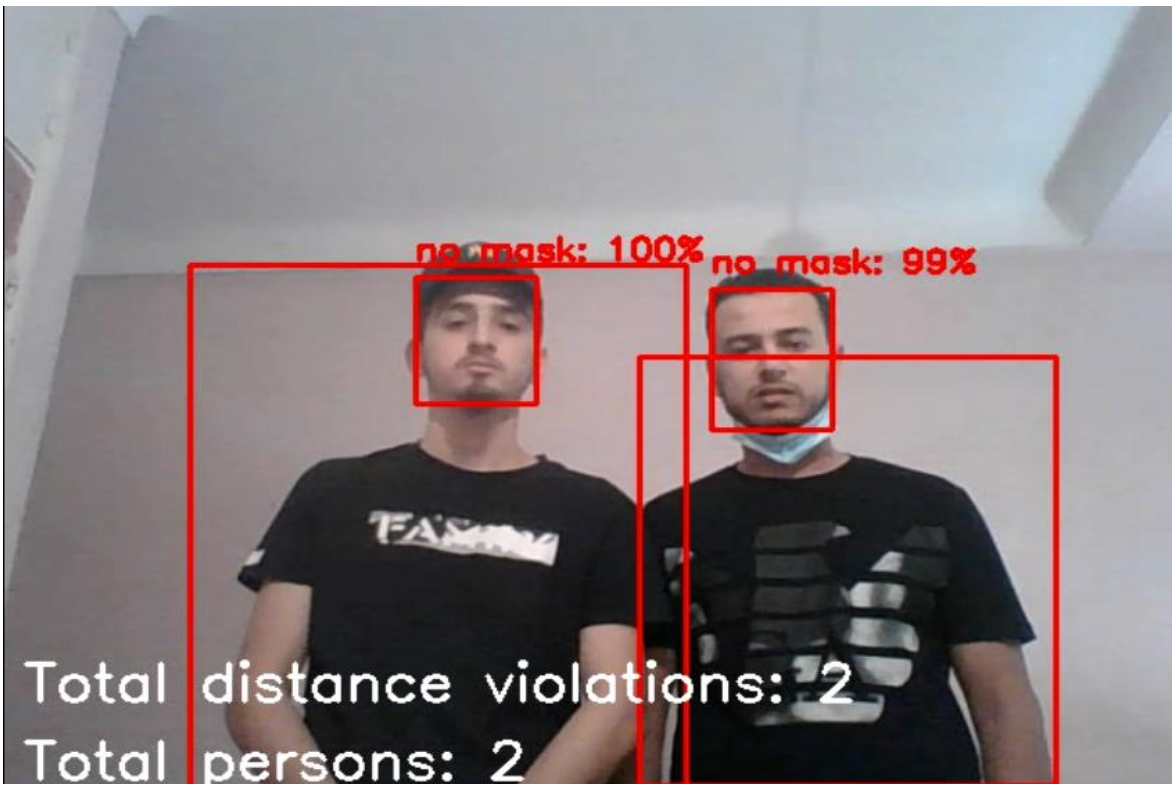


Figure 62 : Test de détection en temps réel – non-distanciation et pas de masque



Figure 63 : Test de détection en temps réel – distanciation et non-masque

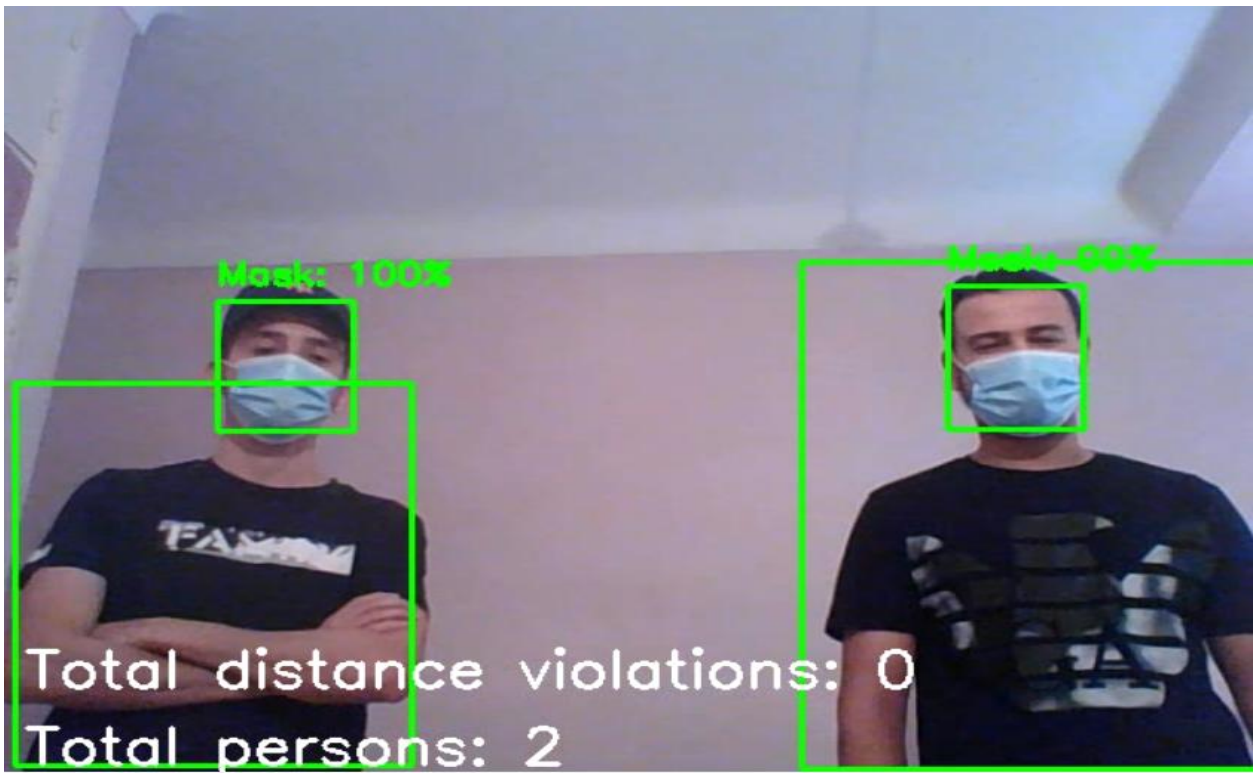


Figure 64 : Test de détection en temps réel – distanciation et masque

IV.3 Problèmes et limites

Pendant la réalisation de ce projet nous avons rencontré quelques difficultés et limites concernant :

- l'absence de la carte graphique dans le matériel utilisée, car elle est nécessaire pour le bon fonctionnement du modèle de détection, afin de bien traiter les images et de détecter les masques et les personnes, l'absence de la carte graphique signifie que le programme utilisera les ressources de calcul du processeur pour faire les détections, ce qui le rend beaucoup plus lent et moins efficace, ce qui parfois conduit à des fausses détections.
- Les vidéos à utiliser doivent avoir une qualité et une résolution élevées afin que le modèle puisse détecter correctement les visages et les masques.
- Le modèle n'a pas pu détecter les visages éloignés de la caméra et les visages latéraux avec des masques.
- La distance euclidienne a été calculée en prenant compte seulement la largeur et la hauteur du l'image sans prendre en considération la profondeur.
- La distance se calcule en pixel, donc cette dernière dépend de la résolution de l'image.

IV.4 Conclusion

Dans ce chapitre on a testé notre application, qui nous ont permis d'évaluer son fonctionnement et sa stabilité, afin de détecter ses lacunes et ses problèmes qu'on fera de notre mieux pour les minimiser au maximum et pour quoi pas les éliminer ce qui améliora le modèle de détection en générale.

Conclusion générale

Ce mémoire a été mené dans le but de concevoir une application d'aide à lutte contre la propagation de la pandémie du Covid-19. Bien qu'ils existent plusieurs travaux abordant la même thématique et cela sous différentes approches, la principale contribution de notre travail réside dans le fait d'avoir intégré plusieurs fonctionnalités à différentes utilités dans un même programme. En effet nous avons regroupés, la détection automatique du masque Covid, le calcul de distanciation entre différentes personnes et le comptage du nombre de ces derniers dans une seule application pour une flexibilité optimale.

Les travaux futurs de cette étude seront basés sur l'utilisation des modèles plus efficaces qui peuvent mieux détecter les personnes, calculer la distance sociale en unités de mesures réelles et prédire les masques mal portés, les masques sur les visages en vue de profil et les visages éloignés. L'une des applications que nous pouvons intégrer à notre projet est le dépistage thermique corporel automatisé à l'aide du deep learning en utilisant une caméra infrarouge, puisque l'élévation de la température est un symptôme clé de l'infection par le COVID-19.

Le dépistage thermique est l'une des procédures qui ont été utilisée pour réduire la propagation du corona virus, elle est effectuée à l'aide des thermomètres infrarouges portables sans contact où le travailleur de la santé doit être à proximité de la personne à examiner, ce qui rend les travailleurs de la santé vulnérables à l'infection, de plus, il est pratiquement impossible de détecter la température de chaque personne dans les lieux publics. La construction de tel détecteur nécessitera un grand ensemble de données d'images thermiques de personnes avec et sans covid 19 pour identifier correctement les personnes infectées

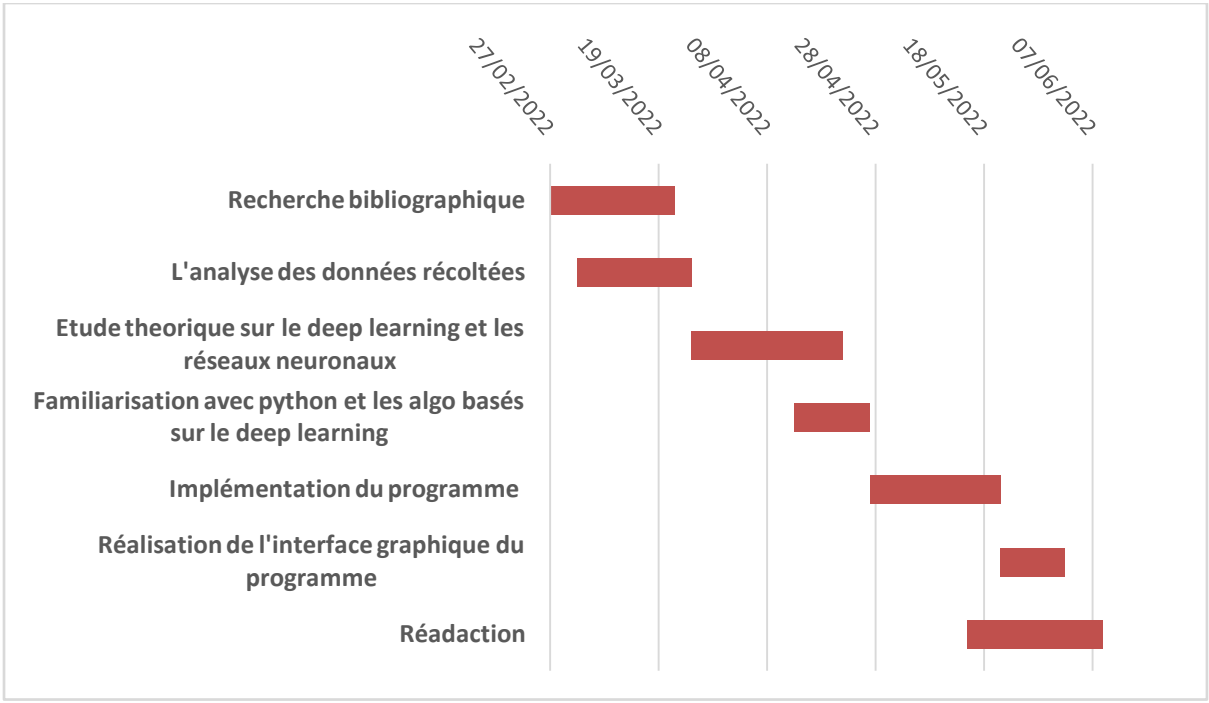
Annexe

Organisation et planning du projet:

Etapes	Date de début	Durée	Date de fin
Recherche bibliographique	17-février	23	22-mars
L'analyse des données récoltées	04-mars	21	25-mars
Etude théorique sur le deep learning et les réseaux neuronaux	25-mars	28	22- avril
Familiarisation avec python et les algorithmes basés sur le deep learning	13-avril	14	27- avril
Implémentation du programme	27- avril	24	21-mai
Réalisation de l'interface graphique du programme	21-mai	12	02-juin
Rédaction	15-mai	25	09-juin

Diagramme de Gantt

Les diagrammes de Gantt est un outil permettant de mieux localiser, identifier et planifier différentes tâches de projet, sur une période de temps, dans le but de respecter les délais des tâches à faire. Ainsi, pour notre projet de gestion de personnes, nous avons mis en place ce graphique pour la planification. Ci-dessous le diagramme de Gantt, qui regroupe les différentes étapes que nous avons accomplies.



```python
import os
from glob import glob
dirs = os.listdir('data')
images_path=[]
labels=[]
for folder in dirs:
    path=glob('./data/{}/*.*'.format(folder))
    label=['{}'.format(folder)]*len(path)
    images_path.extend(path)
    labels.extend(label)
```

Figure 65 : collection des images du Mask et sans Mask avec leurs étiquettes.

```python
# convolution neural network
import tensorflow as tf
from tensorflow.keras import layers
from tensorflow.keras import Sequential
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x,y_array,test_size=0.2,random_state=42)

model = Sequential([
    layers.Conv2D(32,3,padding='same',input_shape=(100,100,3),activation='relu'),
    layers.MaxPool2D(),
    layers.Conv2D(64,3,padding='same',activation='relu'),
    layers.MaxPool2D(),
    layers.Conv2D(128,3,padding='same',activation='relu'),
    layers.MaxPool2D(),
    layers.Flatten(),
    layers.Dense(192,activation='relu'),
    layers.Dense(28,activation='relu'),
    layers.Dense(2,activation='softmax')
    ])

# compiling CNN
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
              loss=tf.keras.losses.categorical_crossentropy,metrics=['accuracy'])
```

Figure 66 : réseau de neurones convolutionelle pour la détection du masque.

```
import gc
from tqdm import tqdm
import numpy as np
import cv2

def datapreprocess(image):
h,w=image.shape[:2]
blob=cv2.dnn.blobFromImage(image,1,(300,300),(104,177,123),swapRB=True)
face_detection_model.setInput(blob)
detections = face_detection_model.forward()
for i in range(0,detections.shape[2]):
    confidence = detections[0,0,i,2]
    if confidence>0.5:
        box=detections[0,0,i,3:7]*np.array([w,h,w,h])
        box=box.astype(int)
        pt1=(box[0],box[1])
        pt2=(box[2],box[3])
        cv2.rectangle(image,pt1,pt2,(0,255,0),2)
        roi=image[box[1]:box[3],box[0]:box[2]]

if roi is not None:
    blob=cv2.dnn.blobFromImage(face, 1, (100,100), (104,117,123), swapRB=True)
    blob_squeeze=np.squeeze(blob).T
    blob_rotate=cv2.rotate(blob_squeeze,cv2.ROTATE_90_CLOCKWISE)
    blob_flip=cv2.flip(blob_rotate,1)


data_img=[]
label_img=[]
i=0
for path, label in tqdm(zip(images_path,labels),desc='preprocessing'):
    img=cv2.imread(path)
    process_img = datapreprocess(img)
    if process_img is not None:
        data_img.append(process_img)
        label_img.append(label)

    i+=1
    if i%100==0:
        gc.collect()
x=np.array(data_img)
y=np.array(label_img)
```

```
preprocessing: 6244it [05:34, 19.85it/s]
```

Figure 67 : prétraitement des données (masque, sans masque).

```python
def monitori(image):

    # load the COCO class labels our YOLO model was trained on
    labelsPath = os.path.sep.join([MODEL_PATH, "coco.names"])
    LABELS = open(labelsPath).read().strip().split("\n")
    personIdx=LABELS.index("person")

    # provide the paths to the YOLO weights and model configuration
    weightsPath = os.path.sep.join([MODEL_PATH, "yolov3.weights"])
    configPath = os.path.sep.join([MODEL_PATH, "yolov3.cfg"])

    # load our YOLO object detector trained on COCO dataset (80 classes)
    model = cv2.dnn.readNetFromDarknet(configPath, weightsPath)
    # get the result of the output layers(detection layers)
    net = model.getLayerNames()
    net = [net[i - 1] for i in model.getUnconnectedOutLayers()]

    frame = detect_mask(image)#detect mask on the entry image
    frame = imutils.resize(frame, width=800)
    (H, W) = frame.shape[:2]
    results = []

    blob = cv2.dnn.blobFromImage(frame, 1 / 255.0, (416, 416),
        swapRB=True, crop=False)
    model.setInput(blob)
    layerOutputs = model.forward(net)

    boxes = []
    centroids = []
    confidences = []
    idxs = cv2.dnn.NMSBoxes(boxes, confidences, MIN_CONF,NMS_THRESH) # non max suppression
    if len(idxs) > 0:
        for i in idxs.flatten():
            # extract bounding box cordinates
            (x, y) = (boxes[i][0], boxes[i][1])
            (w, h) = (boxes[i][2], boxes[i][3])
            r = (confidences[i], (x, y, x + w, y + h), centroids[i])
            results.append(r)


    violate=set()
    # extract centroids and compute euclidien distence betweene thiese centroids
    if len(results)>=2:
        centroids=np.array([r[2] for r in results])
        D=dist.cdist(centroids, centroids, metric="euclidean")

        for i in range (0, D.shape[0]):
            for j in range (i+1, D.shape[1]):
                if D[i, j] <MIN_DISTANCE:
                    violate.add(i) #update violation list in hight
                    violate.add(j) #update violation list in width

    for (i, (prob, bbox, centroid)) in enumerate(results):
        (startX, startY, endX, endY) = bbox
        (cX, cY)=centroid
        color=(0,255,0) #set bounding box color to green as default

        if i in violate:
            color = (0,0,255) # change color to red if detection in violation list
        # draw rectangles around persons
        cv2.rectangle(frame, (startX,startY), (endX,endY), color, 2)

        text1='Total distance violations: {}'.format(len(violate))
        text2='Total persons: {}'.format(len(results))
        # display tne number of violations plus number of detected persons
        cv2.putText(frame,text1, (10, frame.shape[0]-50),cv2.FONT_HERSHEY_SIMPLEX, 1, (255,255,255),2)
        cv2.putText(frame,text2, (10, frame.shape[0]-5),cv2.FONT_HERSHEY_SIMPLEX,1, (255,255,255),2)
    return frame
```

Figure 68 : la fonction de la détection du masques, personnes et la distance sociale.

```python
def det_vid():
        global filename2

        video_path = filename2
        if (video_path == ""):
            mbox.showerror("Erreur", "aucune video selectionner!", parent = windowv)
            return


        video = cv2.VideoCapture(video_path)

        check, frame = video.read()
        if check == False:
            mbox.showerror('Video introuvable','veuillez entrer un chemin valide .', parent = windowv)
            #print('Video introuvable. veuillez entrer un chemin valide .')
            return

        info1.config(text="Etat : Detection...")
        mbox.showinfo("Etat", "Detection, veuillez attendre...", parent=windowv)

        while video.isOpened():
            # check is True if reading was successful
            check, frame = video.read()
            if(check==True):
                frame = monitorv(frame)
                img = cv2.resize(frame, (800, 600))

                cv2.imshow("detection d'une video", img)
                key = cv2.waitKey(1)
                if key & 0xFF == 27:
                    break
            else:
                break

        video.release()
        info1.config(text="Etat : Détection & comptage terminés")
        cv2.destroyAllWindows()
```

Figure 69 : détection à partir d'une vidéo.

```python
def det_cam():

        info1.config(text="Etat : Ouverture de la CAM...")
        mbox.showinfo("Etat", "Ouvrir la CAM...Veuillez attendre...", parent=windowc)

        video = cv2.VideoCapture(0)
        writer = None

        while video.isOpened():

            check, frame = video.read()
            if(check==True):
                frame = monitori(frame)
                img = cv2.resize(frame, (800, 600))

                cv2.imshow("detection par CAMERA", img)
                key = cv2.waitKey(1)
                if key == 27:
                    break
            else:
                break
        video.release()
        info1.config(text="Etat : Détection & comptage terminés")
        cv2.destroyAllWindows()

        vs = cv2.VideoCapture( 0)

        while True:
            (grabbed, frame) = vs.read()
            frame = cv2.resize(frame, (800, 600))
            frame = monitori(frame)
            cv2.imshow("Frame", frame)
            if cv2.waitKey(1) ==27:
                break
```

Figure 70 : détection a partir de la cam.

Bibliographie

1. **AI vs. machine learning vs. deep learning: Key differences.** [Online] https://www.techtarget.com/searchenterpriseai/tip/AI-vs-machine-learning-vs-deep-learning-Key-differences.

2. **What is neural network - application.** [Online] https://deepai.org/machine-learning-glossary-and-terms/neural-network.

3. **An Introduction To Recurrent Neural Networks And The Math That Powers Them.** [Online] https://machinelearningmastery.com/an-introduction-to-recurrent-neural-networks-and-the-math-that-powers-them/.

4. **Niklas Donges, A Guide to RNN: Understanding Recurrent Neural Networks and LSTM Networks.** [Online] https://builtin.com/data-science/recurrent-neural-networks-and-lstm.

5. **Techtarget, Search entreprise AI, convolutional neural network.** [Online] https://www.techtarget.com/searchenterpriseai/definition/convolutional-neural-network.

6. **Tom keldenich, CNN et Couche de Convolution, qu'est-ce que c'est ?, 2021.** [Online] https://www.inside-machinelearning.com/cnn-couche-de-convolution/.

7. **Rectified Linear Units (ReLU) in Deep Learning.** [Online] https://www.kaggle.com/code/dansbecker/rectified-linear-units-relu-in-deep-learning/notebook.

8. **Seb, What is Pooling in a Convolutional Neural Network (CNN): Pooling Layers Explained .** [Online] https://programmathically.com/what-is-pooling-in-a-convolutional-neural-network-cnn-pooling-layers-explained/.

9. **Panadda Kongsilp, CNN: Step 3— Flattening.** [Online] https://medium.com/@PK_KwanG/cnn-step-2-flattening-50ee0af42e3e.

10. **Diego unzueta, DEEP LEARNING FUNDAMENTALS : Convolutional Layers and Fully Connected Layers.** [Online] https://towardsdatascience.com/convolutional-layers-vs-fully-connected-layers-364f05ab460b.

11. **ZHANG, Y., ZHOU, D., CHEN, S., GAO, S., AND MA, Y. Single-image crowd counting via multi-column convolutional neural network.(2016).**

12. **SINDAGI, V. A., AND PATEL, V. M. A survey of recent advances in nn-based single image crowd counting and density estimation. 2018.**

13. **HU, Y., CHANG, H., NIAN, F., WANG, Y., AND LI, T. Dense crowd counting from still images with convolutional neural networks.**

14. **SINDAGI, V. A., AND PATEL, V. M. A survey of recent advances in nn-based single image crowd counting and density estimation.**

15. **PENG, D., SUN, Z., CHEN, Z., CAI, Z., XIE, L., AND JIN, L. Detecting heads using feature refine net and cascaded multi-scale architecture.**

16. **SAM, D. B., SURYA, S., AND BABU, R. V. Switching convolutional neural network for crowd counting.**

17. **ZHANG, X., AND CHEN, D. Csrnet: Dilated convolutional neural networks for understanding the highly congested scenes.**

18. **WANG, Q., GAO, J., LIN, W., AND YUAN, Y. Learning from synthetic data for crowd counting in the wild 2019.**

19. **M. Inamdar, N. Mehendale, Real-Time Face Mask Identification Using Facemasknet Deep Learning Network 2020.** [Online] https://doi.org/10.2139/ssrn.3663305. .

20. **S. Qiao, C. Liu, W. Shen, A. Yuille, Few-Shot Image Recognition by Predicting Parameters from Activations 2018.** [Online] https://doi.org/10.1109/CVPR.2018.00755..

21. **P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. Lecun, OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks,2014.** [Online]

22. **Erhan, C. Szegedy, A. Toshev, D. Anguelov, Scalable Object Detection using Deep Neural 2014.** [Online] https://doi.org/10.1109/CVPR.2014.276..

23. **Redmon, S. Divvala, R. Girshick, A. Farhadi, You onlylook once: Unified, realtime object detection.**

24. **A. Kumar, Z.J. Zhang, H. Lyu, Object detection in real time based on improved single shot multi-box detector algorithm, 2020.** [Online] https://doi.org/10.1186/s13638-020-01826-x.

25. **Jiang, X. Fan, and H. Yan, RetinaMask: A Face Mask detector,2020 .** [Online] http://arxiv.org/abs/2005.03950.

26. **R. Girshick, J. Donahue, T. Darrell, J. Malik, Region-based Convolutional Networks for Accurate Object Detection and Segmentation, .** [Online] https://doi.org/10.1109/TPAMI.2015.2437384..

27. **K. He, X. Zhang, S. Ren, J. Sun, Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition 2015.** [Online] https://doi.org/10.1109/TPAMI.2015.2389824.

28. **N.D. Nguyen, T. Do, T.D. Ngo, D.D. Le, An Evaluation of Deep Learning Methods for Small Object Detection, 2020 .** [Online] https://doi.org/10.1155/2020/3189691..

29. **Z. Cai, Q. Fan, R.S. Feris, N. Vasconcelos, A unified multi-scale deep convolutional neural network for fast object detection, 2016.** [Online] https://doi.org/10.1007/978-3-319-46493-0_22. .

30. **A. Shrivastava, R. Sukthankar, J. Malik, A. Gupta, Beyond Skip Connections: TopDown Modulation for Object Detection, 2016.**

31. **Z. Liang, J. Shao, D. Zhang, L. Gao, Small object detection using deep feature pyramid networks, in: Lecture Notes in Computer Science.**

32. **K. He, G. Gkioxari, P. Dollar, R. Girshick, Mask R-CNN, 2017.**

33. **A. Kumar, Z.J. Zhang, H. Lyu, Object detection in real time based on improved single shot multi-box detector algorithm 2020.** [Online] https://doi.org/10.1186/s13638-020-01826-x.

34. **Shashi yadav, Deep Learning based Safe Social Distancing and Face Mask Detection in Public Areas for COVID-19 Safety Guidelines Adherence 2020.**

35. **FaceMask Dataset .** [Online] https://www.kaggle.com/datasets/sumansid/facemask-dataset?select=Mask.

36. **Maskedface-net.** [Online] https://github.com/cabani/MaskedFace-Net.

37. **Caffe - Deep learning framework by BAIR.** [Online] https://caffe.berkeleyvision.org/]. .

38. **Taha anwar, Deep Learning with OpenCV DNN Module, A Comprehensive Guide, 2020.** [Online] https://bleedai.com/deep-learning-with-opencv-dnn-module-a-comprehensive-guide/.

39. **Preprocessing data in scikit.** [Online] https://scikit-learn.org/stable/modules/preprocessing.html#preprocessing-categorical-features.

40. **Mira Boulos, Facial Recognition and Face Mask Detection Using Machine Learning Techniques.** [Online] https://digitalcommons.montclair.edu/etd/728.

41. **Intuition of Adam Optimizer, 2020.** [Online] https://www.geeksforgeeks.org/intuition-of-adam-optimizer.

42. **a Verma, ShivUnderstanding different Loss Functions for Neural Networks, 2019.** [Online] https://shiva-verma.medium.com/understanding-different-loss-functions-for-neural-networks-dd1ed0274718.

Résumé

L'épidémie actuelle du coronavirus a semé la peur partout dans le monde, ce qui a incité les professionnels de la santé à se mobiliser pour en faire face.

L'intelligence artificielle était le moyen le plus efficient, grâce à son aptitude à reproduire l'intelligence humaine dans ses capacités d'apprentissage, en utilisant des algorithmes d'apprentissage profond qui marquent l'aube de la recherche sur cette dernière, ce qui aide à analyser les images médicales, prévoir la propagation et détecter la mésinformation ainsi que d'autres bienfaits.

Cependant, ses avantages évidents s'accompagnent non seulement de limitations générales de l'application des systèmes d'IA, mais aussi de défis qui caractérisent spécifiquement l'opérationnalisation des concepts de maladie et de santé.

Mots clés : Covid, intelligence artificielle, apprentissage profond, réseau neuronal.

Abstract

The current coronavirus outbreak has sown fear all over the world, promoting medical professionals to step up to deal with it.

Artificial intelligence was the most effective way, thanks to its ability to replicate human intelligence in its learning capabilities, using deep learning algorithms that mark the dawn of AI research. Which helps to analyze medical images, predict the spread and detect misinformation along with other benefits.

However, its obvious advantages come not only with general limitations of the application of AI systems, but also with challenges that characterize the operationalization of the concepts of disease and health.

Key words: Covid, artificial intelligence, deep learning, neural network.

ملخص

أثار وباء فيروس كورونا الحالي الخوف في جميع أنحاء العالم،مما شجع المهنيين الصحيين على التصعيد لمواجهته.

كان الذكاء الاصطناعي الوسيلة الأكثر كفاءة، بفضل قدرته على تقليد الذكاء البشري في قدراته التعلمية، باستخدام خوارزميات التعلم العميق التي تميز فجر البحث عن هذا الأخير. مما يساعد في تحليل الصور الطبية، التنبؤ بالانتشار ، اكتشاف المعلومات الخاطئة، بالإضافةللفوائد الأخرى.

ومع ذلك، فإن فوائده الجليّ لا تقترن فقط بالقيود العامة في تطبيق أنظمة الذكاء الاصطناعي، ولكن أيضًا بالتحديات التي تميز تفعيل مفاهيم المرض والصحة.

الكلمات المفتاحية:كوفيد، ذكاء اصطناعي، تعلم عميق، شبكة عصبية.