

ISEN

ALL IS DIGITAL!

OUEST



Projet M1

Année scolaire 2020/2021

Institut Supérieur de l'Électronique et du Numérique

Tél. : +33 (0)2.98.03.84.00

Fax : +33 (0)2.98.03.84.10

20, rue Cuirassé Bretagne

CS 42807 - 29228 BREST Cedex 2 - FRANCE

Détection et vérification du port du masque

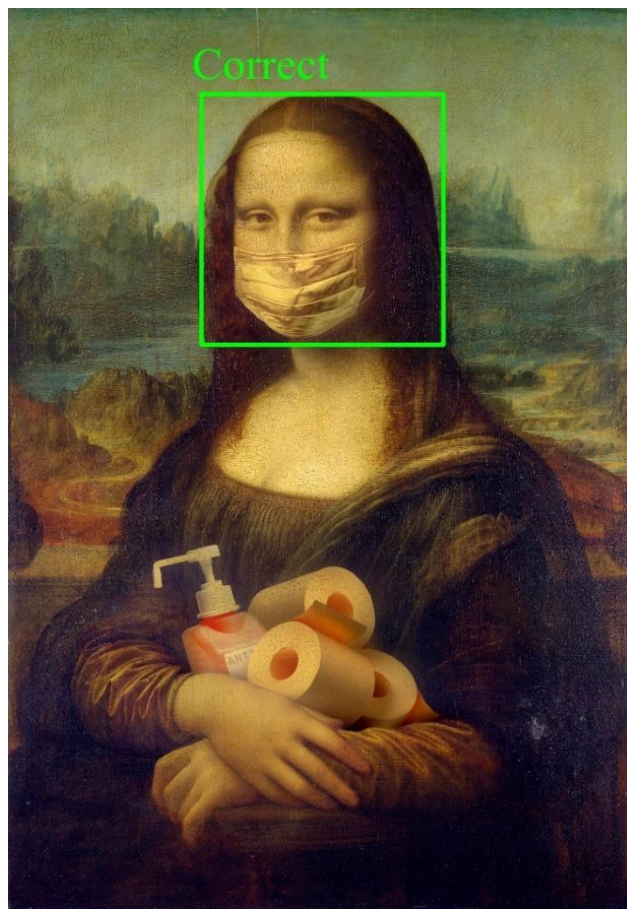


Image originale de Yaroslav Danylchenko postée sur Pexels.com

Proposé par : HADJADJI Bilal

Thématique : Informatique et traitement d'images

FOUQUEAU Maël

Domaine professionnel : Ingénieur de Projets et d'Affaires

ROCHARD Augustin

Domaine professionnel : Ingénieur de Projets et d'Affaires

Table des matières

1. Glossaire -----	p.1
2. Table des figures -----	p.5
3. Table des tableaux -----	p.7
4. Introduction -----	p.8
5. Cahier des charges -----	p.9
6. Gestion de projet :	
6.1 Organisation temporelle -----	p.14
6.2 La méthodologie de travail -----	p.17
6.3 La Base De Données -----	p.18
6.4 Organisation de la partie détection de visage -----	p.20
6.5 Organisation de la partie détection du port du masque -----	p.21
6.6 Organisation de la partie IHM -----	p.23
7. Développement technique :	
7.1 Spécifications :	
7.1.1 Langage de programmation -----	p.28
7.1.2 Système d'exploitation -----	p.28
7.1.3 IDE -----	p.29
7.1.4 Framework -----	p.29
7.1.5 Résumé -----	p.30
7.2 Schémas, organigrammes -----	p.30
7.3 Contributions techniques :	
7.3.1 Constitution du dataset -----	p.31
7.3.2 Pre-processing -----	p.32
7.3.3 Entraînement des CNNs -----	p.32
7.3.4 Détection des visages -----	p.36
7.3.5 Classification des visages par les CNNs -----	p.39
7.3.6 Combinaisons des classifieurs -----	p.41
7.3.7 Interface graphique -----	p.44
7.3.8 Conversion en exécutable -----	p.45
8. Conclusion -----	p.46
9. Bibliographie/Webographie -----	p.47
10. Annexes -----	p.49

1– Glossaire

Back-end : C'est la partie invisible pour l'utilisateur, mais qui va permettre le bon fonctionnement d'un site internet.

Batch size : Le batch size définit le nombre d'échantillons qui seront propagés dans le réseau.

BDD : Abréviation de Base De Données. Au sein du projet, c'est l'ensemble des images de visages sans masques, avec masques bien portés ainsi qu'avec masques mal portés, qui sert à entraîner les modèles de réseaux de neurones.

CNN : Les Convolutional Neural Networks sont une forme de réseaux de neurones artificiels qui opèrent des convolutions sur les données qui lui sont données en entrée.

Computer vision : D'après la définition de Wikipédia, le "computer vision est une branche de l'intelligence artificielle dont le principal but est de permettre à une machine d'analyser, traiter et comprendre une ou plusieurs images prises par un système d'acquisition".

Dataset : C'est une collection d'ensembles d'informations liées entre elles qui est composée d'éléments distincts mais qui peut être manipulée comme une unité par un ordinateur. Se rapproche ici d'une base de données.

Epoch : En termes de réseaux neuronaux artificiels, une epoch correspond à un cycle de l'ensemble complet de données de formation. En général, la formation d'un réseau neuronal prend plus de quelques époques. En d'autres termes, si nous alimentons un réseau neuronal en données de formation pendant plus d'une époque selon différents schémas, nous espérons obtenir une meilleure généralisation lorsqu'une nouvelle entrée "non vue" (données de test) lui sera donnée.

Faux Négatif : Un Faux Négatif est le résultat d'une prise de décision dans un choix à deux possibilités (positif et négatif), déclaré négatif, là où il est en réalité positif. Dans le projet actuel, un Faux Négatif serait la non-détection d'un visage par le système, alors qu'en réalité il y en a un.

Faux Positif : Un Faux Positif est le résultat d'une prise de décision dans un choix à deux possibilités (positif et négatif), déclaré positif, là où il est en réalité négatif. Dans le projet actuel, un Faux Positif serait la détection d'un visage par le système, alors qu'en réalité ce n'en n'est pas un.

Features extracting : Le features extracting est un processus de réduction de la dimensionnalité par lequel un ensemble initial de données brutes est réduit à des groupes plus faciles à gérer pour le traitement.

FFHQ : FFHQ, initiales de Flickr-Faces-HQ, qui est un dataset d'images de visages.

Frame : Ici, une frame signifie une image.

Framework : D'après la définition que donne Wikipédia, en programmation informatique, un framework (appelé aussi infrastructure logicielle ou infrastructure de développement) désigne un ensemble cohérent de composants logiciels structurels, qui sert à créer les fondations ainsi que les grandes lignes de tout ou d'une partie d'un logiciel (architecture). Un framework se distingue d'une simple bibliothèque logicielle.

Front-end : C'est la partie d'une page Internet ou d'une interface qu'un utilisateur peut voir et avec lesquelles il peut interagir directement.

Google Colab : Google Colab est un produit de Google Research. Il permet à n'importe qui d'écrire et d'exécuter le code Python de son choix par le biais du navigateur. C'est un environnement particulièrement adapté au machine learning, à l'analyse de données et à l'éducation.

Haar Cascade Classifier : Haar Cascade Classifier est une approche basée sur l'apprentissage automatique dans laquelle une fonction en cascade est formée à partir d'un grand nombre d'images positives et négatives. Elle est ensuite utilisée pour détecter des objets dans d'autres images.

IA : abréviation de "Intelligence Artificielle". C'est l'ensemble des théories et des techniques mises en œuvre en vue de réaliser des machines capables de simuler l'intelligence humaine.

IDE : D'après la définition de Wikipédia, En programmation informatique, un IDE (environnement de développement) est un ensemble d'outils qui permet d'augmenter la productivité des programmeurs qui conçoivent des logiciels.

IHM : Abréviation d'Interface Homme-Machine, c'est l'interface qui permet à l'utilisateur d'utiliser les algorithmes développés.

Learning rate : En apprentissage automatique et en statistique, le learning rate, dit aussi « taux d'apprentissage », est un paramètre d'ajustement dans un algorithme d'optimisation qui détermine la taille de l'étape à chaque itération tout en se déplaçant vers un minimum d'une fonction de perte.

Microsoft Visual Studio : Microsoft Visual Studio est une suite de logiciels de développement pour Windows et mac OS conçue par Microsoft.

Mode offline : Terme employé lorsqu'il est question de tests d'inférence sur des images du dataset.

Mode online : Terme employé lorsque qu'il est question de la partie applicative du projet, par exemple, si on effectue un test online, le test est effectué sur la webcam.

OpenCV : OpenCV est une bibliothèque graphique libre, initialement développée par Intel, spécialisée dans le traitement d'images en temps réel.

Phase d'apprentissage / de validation : L'ajustement d'un modèle se fait en deux étapes.

En premier lieu, il y a la phase d'apprentissage : on fait apprendre au modèle ce qu'on veut qu'il prédise en parcourant la BDD puis en comparant ses prédictions par rapport aux résultats attendus. Au fur et à mesure que l'on parcourt la BDD on corrige les poids attribués à chaque neurone en fonction des bonnes et mauvaises prédictions.

Ensuite, on vérifie que le modèle ne rentre pas dans le surentraînement en lui faisant faire des prédictions sur des images qui ne servent pas à ajuster les poids. Cela permet de tester la capacité du modèle à généraliser ce qu'il apprend, c'est la phase de validation.

Ces phases permettent à un modèle de comprendre la logique qu'elle doit intégrer pour pouvoir détecter correctement ce qu'on veut de lui.

Pre-processing : Une fois la BDD constituée, il y a une étape dite de “pre-processing” qui permet de convertir les images ou les données de manière générale dans une forme plus facile à manipuler par les réseaux de neurones.

PyQt : PyQt est un module libre qui permet de lier le langage Python avec la bibliothèque Qt. Il permet ainsi de créer des interfaces graphiques en Python.

PySide : D’après la définition de Wikipédia, PySide est un module libre permettant de créer des interfaces graphiques en Python. PySide se distingue de PyQt par le fait qu’il est disponible sous licence LGPL (Licence Publique Générale Limitée), c’est d’ailleurs ce qui a déclenché son développement.

PyTorch : C’est un framework d’apprentissage automatique open source qui accélère le passage du prototypage de la recherche au déploiement de la production.

Qt Designer : Qt Designer est un outil permettant de construire rapidement des interfaces graphiques avec des widgets issus du framework Qt GUI.

Réseaux de neurones : D’après la définition que donne Wikipédia, un réseau de neurones artificiels, ou réseau neuronal artificiel, est un système dont la conception est à l’origine schématiquement inspirée du fonctionnement des neurones biologiques, et qui par la suite s’est rapproché des méthodes statistiques.

Cette structure de réseaux de neurones artificiels s’est montrée particulièrement efficace en Computer Vision, c’est-à-dire dans l’analyse d’image par ordinateur.

En effet, les opérations de convolution permettent d’extraire des caractéristiques propres aux images comme les contours, les lignes verticales, les lignes horizontales, etc...

Sur-apprentissage : On parle de surapprentissage quand un modèle a trop appris les particularités de chacun des exemples fournis en exemple. Il présente alors un taux de succès très important sur les données d’entraînement (pouvant atteindre jusqu’à 100%), au détriment de ses performances générales réelles

Transfer Learning : C’est un ensemble de méthodes de Machine Learning où un modèle est développé pour exécuter une tâche et ce même modèle est réutilisé par la suite pour répondre à un autre besoin.

Dans cette section du Machine Learning se distinguent deux principaux types de Transfer Learning:

- Fine-tuning : on utilise un modèle pré-entraîné sur un dataset A comme base, puis on va l’entraîner à répondre à un nouveau besoin en l’entraînant sur un dataset B. On va mettre à jour tous les paramètres du modèle contrairement au features-extracting.
- Features-extracting : on reprend le principe du Transfer Learning, on réutilise un modèle pré-entraîné sur un dataset A, adapté à un nouveau besoin, c’est-à-dire réentraîné sur un dataset B. Contrairement un fine-tuning, on ne va pas mettre à jour tous les paramètres du modèle mais seulement ceux des dernières couches. Bien que cette méthode soit par conséquent moins précise, elle permet de gagner beaucoup de temps car elle nécessite moins de calculs. Cependant, les résultats de cette méthode se montrent souvent plus que satisfaisants.

Vrai Négatif : Un Vrai Négatif est le résultat d'une prise de décision dans un choix à deux possibilités (positif et négatif), déclaré négatif, là où il est en réalité bien négatif. Dans le projet actuel, un Vrai Négatif serait la non-détection d'un visage par le système, alors qu'en réalité il y a bien aucun visage.

Vrai Positif : Un Vrai Positif est le résultat d'une prise de décision dans un choix à deux possibilités (positif et négatif), déclaré positif, là où il est en réalité bien positif. Dans le projet actuel, un Vrai Positif serait la détection d'un visage par le système, là où il y a bien un visages.

2 – Table des figures

Figure 1 : Diagramme bête à corne du projet -----	p.9
Figure 2 : Diagramme pieuvre du projet -----	p.11
Figure 3 : Interface attendue -----	p.12
Figure 4 : Différentes salles de conversation Discord -----	p.17
Figure 5 : Dossiers communs Google Drive -----	p.17
Figure 6 : Exemples d'images composant la classe « With Mask », après le pre-processing -----	p.19
Figure 7 : Les 3 classes constituant la BDD -----	p.20
Figure 8 : Exemples de détection de visage avec le CNN ResNet-10 -----	p.20
Figure 9 : Valeur moyenne de précision de détection après entraînement du modèle -----	p.21
Figure 10 : Exemples de détection du CNN créé de A à Z par l'équipe -----	p.21
Figure 11 : Exemples de détection de port du masque -----	p.22
Figure 12 : Exemple de détection de plusieurs visages en même temps -----	p.23
Figure 13 : Mockup de la première interface réalisée -----	p.24
Figure 14 : Première interface réalisée -----	p.24
Figure 15 : Mockup de l'interface finale -----	p.25
Figure 16 : Interface finale -----	p.26
Figure 17 : Exemples de l'IHM secondaire selon différentes dimensions -----	p.27
Figure 18 : Librairie Microsoft C++ nécessaire pour OpenCV -----	p.29
Figure 19 : Schéma de la V1 du projet -----	p.30
Figure 20 : Schéma de la version définitive du projet -----	p.31
Figure 21 : Précision des 5 meilleurs CNNs avec le dataset de 1500 images -----	p.34
Figure 22 : Précisions des 5 meilleurs CNNs avec le dataset de 4500 images -----	p.34
Figure 23 : Précision de VGG16 en fonction de la taille de la base d'apprentissage -----	p.35
Figure 24 : Précision de Resnext50_32x4d en fonction de la taille de la base d'apprentissage -----	p.35
Figure 25 : Précision de Densenet en fonction de la taille de la base d'apprentissage -----	p.36
Figure 26 : Faux positif d'une détection de visage avec Haar Cascade Classifier -----	p.37
Figure 27 : Détection d'un visage masqué et sans masque avec Haar Cascade Classifier -----	p.37
Figure 28 : Détection d'un visage masqué de face et de profil avec le CNN pré-entraîné -----	p.38

Figure 29 : Comparaison des différents algorithmes de détection du visage -----	p.38
Figure 30 : Résultat du script permettant de construire la matrice de confusion pour le CNN -----	p.39
Figure 31 : Matrice de confusion du CNN de détection de visage / dataset de 900 images -----	p.39
Figure 32 : Processus suivi par le programme -----	p.39
Figure 33 : Coquille dans le processus suivie par le programme -----	p.40
Figure 34 : Test online des CNN -----	p.40
Figure 35 : Comparaison des modèles et des méthodes de combinaison / dataset de 4500 images -----	p.42
Figure 36 : Matrice de confusion des CNNs VGG-16, resnext50_32x4d et densenet -----	p.43
Figure 37 : Matrices de confusion de la combinaison par moyenne et par pondération des 3 CNNs ---	p.43
Figure 38 : Exemples de mauvaise prédiction par la combinaison par pondération -----	p.43
Figure 39 : Création de la page « Confidentialité » avec Qt Designer -----	p.44
Figure 40 : Interface exécutable parmi les fichiers nécessaires à son exécution -----	p.45
Figure 41 : Exemple de compte rendu de réunion -----	p.49
Figure 42 : Interface finale en mode jour -----	p.50
Figure 43 : Interface finale en mode sombre -----	p.50
Figure 44 : Partie « Instructions » de l'interface finale, en mode sombre -----	p.51
Figure 45 : Partie « A propos » de l'interface finale, en mode sombre -----	p.51
Figure 46 : Partie « Confidentialité » de l'interface finale, en mode sombre -----	p.52
Figure 47 : Partie « Confidentialité » de l'interface redimensionnable, en mode sombre -----	p.52

3 – Table des tableaux

Tableau 1 : QQQQCPC -----	p.9
Tableau 2 : Tableau de valorisation des fonctions -----	p.9
Tableau 3 : Test des fonctionnalités -----	p.12
Tableau 4 : Solutions en cas de tests ratés -----	p.12
Tableau 5 : Planning prévisionnel -----	p.13
Tableau 6 : Planning réel -----	p.14

4 – Introduction

L'année 2020 a été marquée par l'apparition de la pandémie mondiale du SARS-CoV2 dit COVID-19. Cette épidémie est sans précédent de par son ampleur ainsi que de par les changements qu'elle a contraints au sein de nos sociétés.

Le port du masque a ainsi été généralisé de telle sorte qu'il fait maintenant partie intégrante de notre quotidien. C'est un moyen très efficace de prévenir la contamination de nouveaux individus dans l'espace public car, dès juillet 2020, l'OMS confirmait que la propagation du virus se faisait principalement par l'air. [1]

C'est pour cette raison qu'en dehors d'un confinement strict, ces mesures de port du masque se montrent efficaces. Le masque ne permet pas une protection contre le virus, il permet de protéger les gens se trouvant autour, en ne transmettant pas les microbes à travers l'air.

Cette stratégie repose donc sur l'effort de chaque individu de bien vouloir porter le masque.

De nombreuses applications ont fait surface suite à ce besoin mais ces dernières ne permettent que de détecter si l'individu porte un masque et non pas s'il est bien ou mal porté. [2] [3]

Or ce détail a de l'importance car si le masque ne couvre pas le nez et la bouche, l'efficacité du masque est fortement diminuée pour ne pas dire quasi-inexistante. [4]

L'objectif de ce projet est, à partir d'un visage, de déterminer à quelle classe appartient l'individu parmi les suivantes : l'individu porte correctement son masque, l'individu ne porte pas correctement son masque, l'individu ne porte pas de masque.

Le tout devra être simple d'utilisation, ce qui induit que les algorithmes qui permettent cette classification seront employés au sein d'une IHM qui fera des détections à partir d'une webcam interne ou branchée à l'ordinateur.

Ce système pourra être déployé pour vérifier que les gens portent correctement le masque à l'entrée d'un magasin, par exemple. Il permet également d'améliorer la traçabilité des cas contacts dans l'espace public, la France ayant déjà commencé à utiliser l'IA pour vérifier si les gens portent le masque dans les transports en commun, à l'issue du premier confinement. [5]

Ce rapport décrit les différentes étapes nécessaires à la conception d'une telle application. Il revient sur les choix techniques adoptés tout au long du processus.

5 – Cahier des charges

Contexte :

Afin de prévenir efficacement la propagation du virus COVID-19, presque tout le monde porte un masque.

Cependant, certaines personnes n'obéissent pas aux règles et ne portent pas de masque ou ne le portent pas correctement.

Ainsi, l'objectif principal de ce projet est de réaliser un système basé sur le traitement d'image et l'intelligence artificielle capable de vérifier si un individu porte correctement son masque ou non. De plus, le système vérifie en outre si le masque est correctement porté.

Ainsi, il est important de définir les besoins d'un tel système :

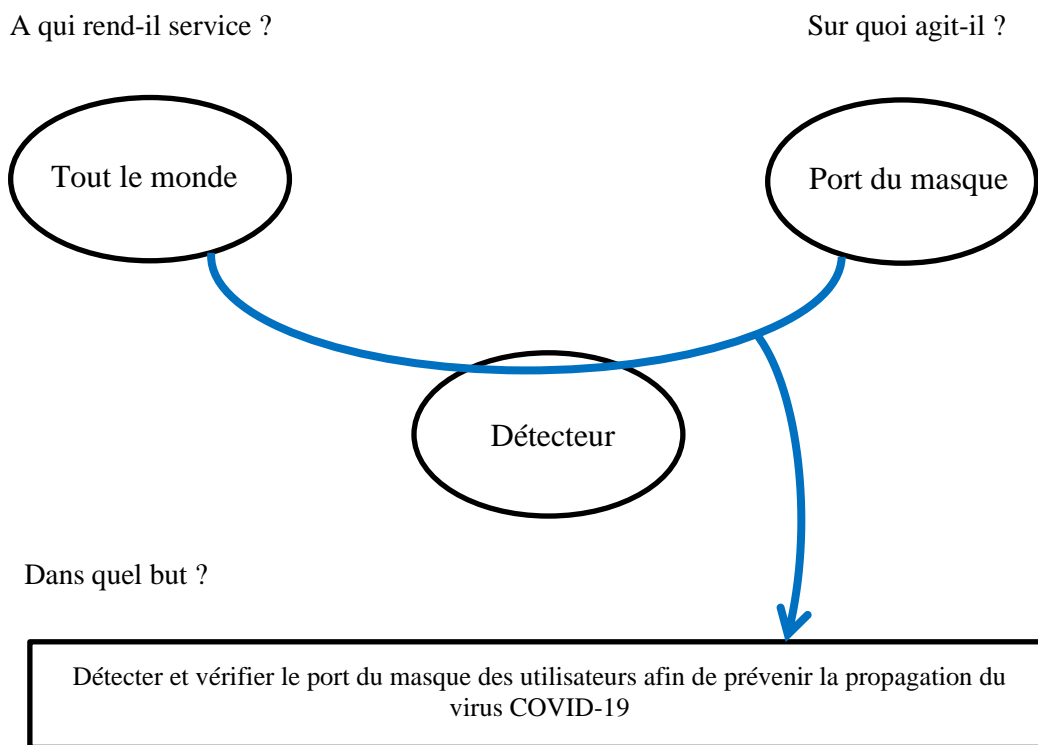


Figure 1 : Diagramme bête à corne du projet

Le produit final a donc pour but de détecter les gens qui ne portent pas bien leur masque, afin de prévenir la propagation et de protéger les autres du virus à partir de la détection qui a été réalisée par le détecteur sur la bonne tenue, ou non, du masque.

A partir d'une interface simple et épurée, l'utilisateur pourra donc procéder à cette détection.

Pour récapituler le projet, voici un QQQQCPC qui englobe les informations importantes autour de la réalisation du détecteur de port du masque, au sein de l'ISEN Nantes

Quoi	Un détecteur de port du masque
Qui	Origine : ISEN Nantes Clients : Tout le monde
Où	Création : ISEN Nantes
Quand	Interface disponible fin avril 2021
Comment	Informatique et traitement d'images avec de l'Intelligence Artificielle
Pourquoi	Pour vérifier le bon port du masque lié à la crise sanitaire du COVID-19 Pour le projet M1ISEN Nantes
Combien	Gratuit

Tableau 1 : QQQQCPC

L'objectif est donc de réaliser l'algorithme de détection du port du masque, ainsi que l'interface qui va avec, en 2 mois.

Pour cela, il est important de respecter plusieurs fonctions, 2 fonctions principales et 9 fonctions contraintes, qui permettront à l'utilisateur une détection optimale et une navigation facile sur l'interface.

Les fonctions peuvent être plus moins flexibles mais doivent permettre de répondre au maximum au problème de port du masque qui, s'il est respecté, peut permettre de prévenir efficacement la propagation du virus et de sauver des vies.

La liste des fonctions est donc :

Type	Caractéristiques	Valeur/flexibilité
Fonction Fp1	Détecter sur un utilisateur son bon port du masque, son non port du masque ou son port incorrect du masque	Continuellement
Fonction Fp2	Afficher le résultat de cette détection dans une interface	Continuellement lors du lancement de la détection
Fonction Fc1	Respecter les consignes	Aucune
Fonction Fc2	Détecter de manière optimale pour chaque individu	Adaptation en fonction du look des individus
Fonction Fc3	Utiliser le langage python	Aucune
Fonction Fc4	Détecter simultanément plusieurs individus	Donne un conseil pour bien porter son masque
Fonction Fc5	Réutilisabilité du détecteur pour un autre environnement	Aucune
Fonction Fc6	Équipe, délais	2 personnes, 2 mois

Fonction Fc7	Respecter les règles et lois en vigueur	Adaptation en fonction des changements de lois
Fonction Fc8	Grande autonomie du détecteur	Capable de détecter continuellement pendant plusieurs heures
Fonction Fc9	Utilisation et compréhension facile de l'interface	Aucune

Tableau 2 : Tableau de valorisation des fonctions

Afin de mieux présenter le lien entre produit/service et son environnement, de voir les fonctions essentielles du produit et comment elles réagissent avec le monde extérieur, voici un diagramme pieuvre :

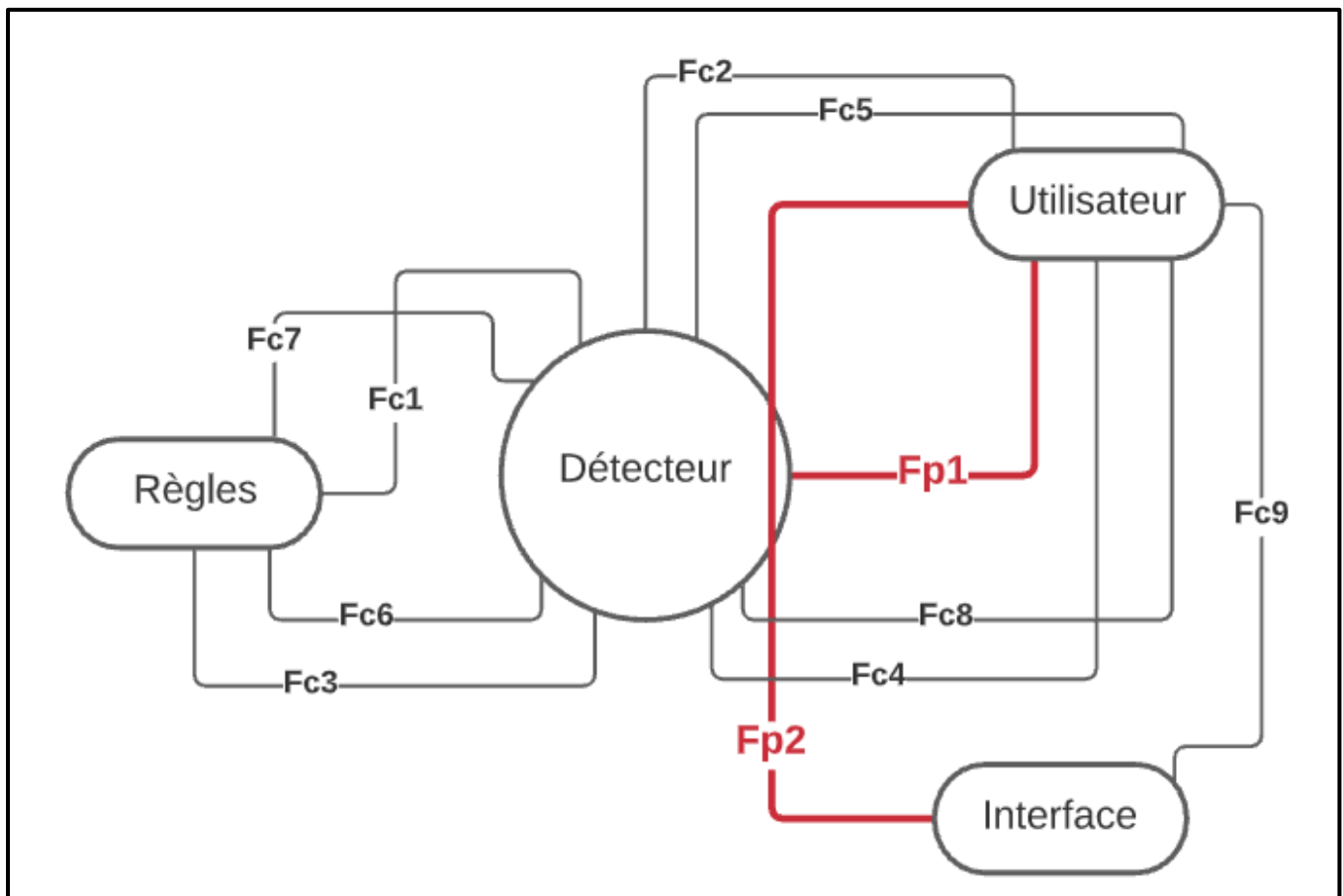


Figure 2 : Diagramme pieuvre du projet

Ce diagramme pieuvre, réalisé sur www.ludichart.com, représente bien la complexité du projet et montre parfaitement la multiplicité des différentes fonctions qui rendent ce projet minutieux et complet.

Les résultats attendus au début du projet étaient donc une interface simple d'utilisation permettant à un individu d'utiliser un algorithme de détection de port du masque. Voici le rendu :

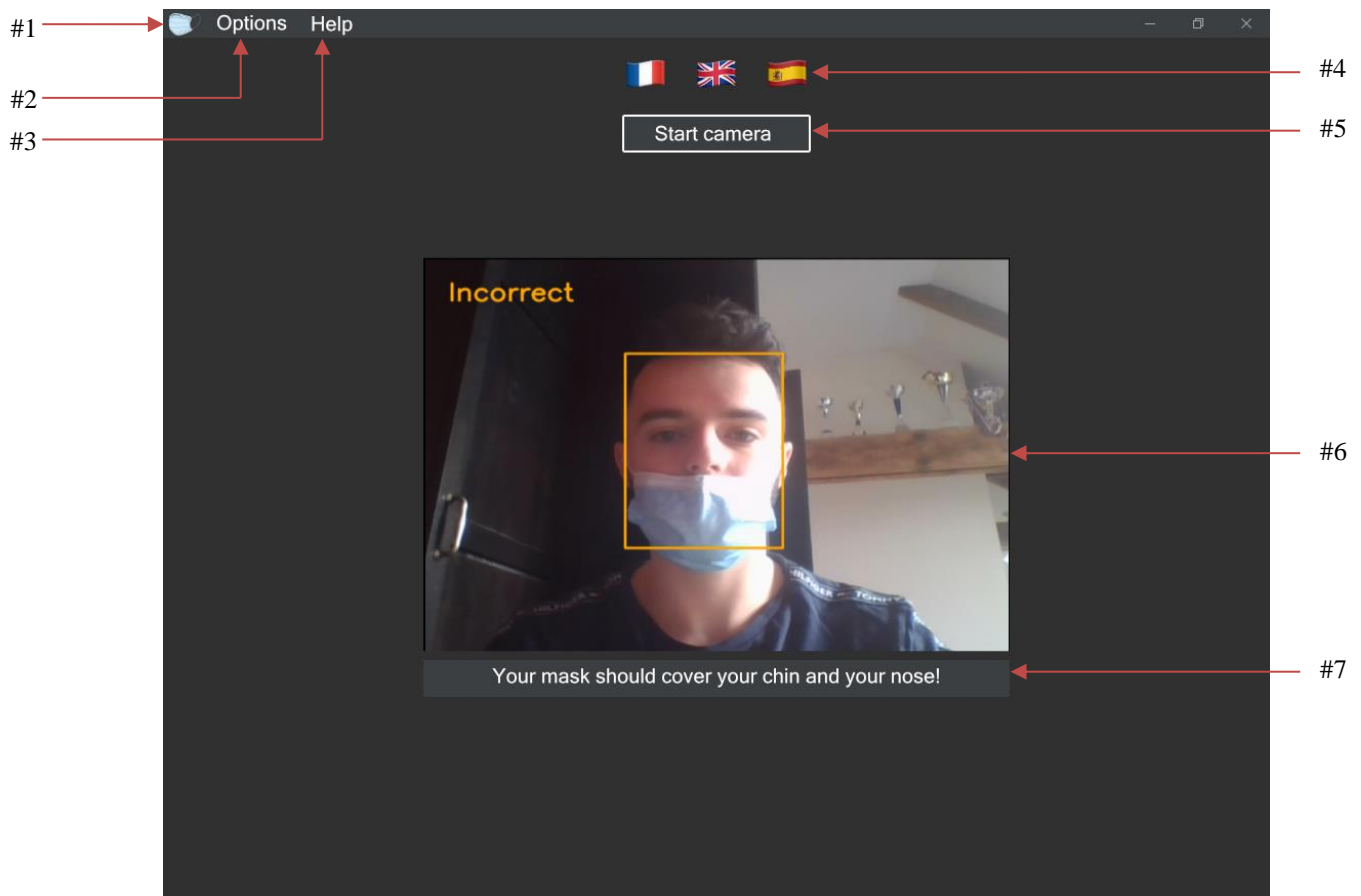


Figure 3 : Interface attendue

Cette interface est composée de :

- #1 : Icône libre de droit représentant un masque
- #2 : Menu « Options » dans lequel il sera possible de sélectionner la webcam que l'utilisateur souhaite utiliser parmi celles détectées, ainsi que de changer le thème de l'interface (sombre ou clair)
- #3 : Menu « Help » dans lequel il sera possible de voir les instructions pour bien utiliser l'interface, la politique de confidentialité ainsi que le résumé du projet
- #4 : Trois boutons représentant des drapeaux permettant de changer la langue de l'interface
- #5 : Bouton permettant de lancer la détection de port du masque sur l'utilisateur en fonction de la webcam choisie (webcam de l'ordinateur par défaut)
- #6 : Rendu de la détection du port du masque sur l'utilisateur après appui sur le bouton de lancement de la détection, avec la prédiction
- #7 : Conseil à l'utilisateur en fonction de comment il porte son masque

Pour garantir à l'utilisateur une expérience totale de l'interface et du système, voici des tests simples à réaliser pour s'assurer que les éléments indispensables à la bonne détection et à la bonne utilisation de l'interface sont opérationnels :

Test détection	Test conseil statique	Test bouton détection	Test webcams	Test thèmes
Vérifier que la prédiction affichée par le système corresponde bien à la manière dont l'utilisateur porte son masque	Vérifier que le conseil en dessous du rendu de la webcam soit en adéquation avec la prédiction du système et votre port (ou non) du masque	Vérifier que l'appui sur le bouton de lancement de détection lance la détection et affiche le visuel du résultat	Vérifier que lorsque l'utilisateur connecte une ou plusieurs webcams le système le détecte et l'affiche dans le sous menu « webcam » et qu'elle(s) soi(en)t utilisable(s)	Vérifier que lors du choix du thème, l'interface ajuste bien sa couleur de fond

Tableau 3 : Test des fonctionnalités

Si un de ces tests venait à être défaillant, voici un tableau récapitulant les possibles raisons de ce(s) dysfonctionnement(s), et la solution adaptée :

Tests ratés	Pourquoi ?	Solutions
Test détection	<ul style="list-style-type: none"> - Mauvaise lumière - Mauvaise qualité de webcam - Visage pas / mal détectable - Visage trop éloigné de la webcam 	<ul style="list-style-type: none"> - Faire en sorte que le visage soit bien en face de la caméra, bien dissimulable avec une bonne luminosité
Test conseil statique	<ul style="list-style-type: none"> - Le système détecte plus d'un visage et affiche donc toujours le même message 	<ul style="list-style-type: none"> - Faire en sorte qu'il n'y ait aucun visage derrière soi détectable par le système
Test bouton détection	<ul style="list-style-type: none"> - Aucune webcam n'est détectable - Webcam connectée est déjà utilisée par un autre programme 	<ul style="list-style-type: none"> - Vérifier de bien avoir au moins une webcam fonctionnelle et libre d'utilisation
Test webcams	<ul style="list-style-type: none"> - Le port de l'ordinateur n'est pas fonctionnel - Plus de quatre webcams connectées - Aucune webcam connectée 	<ul style="list-style-type: none"> - Vérifier la bonne fonctionnalité des ports USB - Ajouter au programme la possibilité d'utiliser plus de quatre webcams
Test thèmes	<ul style="list-style-type: none"> - Problème au niveau de la carte graphique 	<ul style="list-style-type: none"> - Changer de carte graphique

Tableau 4 : Solutions en cas de tests ratés

6 - Gestion de projet

6.1 - Organisation temporelle

Un projet qui s'étend sur plusieurs mois, qui demande autant de recherche et d'investissement n'est réalisable que si le groupe de travail est organisé du début à la fin. En effet, afin de pouvoir répondre au besoin initial, la plus grande constance dans l'organisation est plus que nécessaire et cela commence par un planning prévisionnel le plus précis possible.

Ainsi, il a été décidé de rédiger un tableau Excel sur Google Drive, commun au binôme afin qu'il soit rapidement consultable par tous les membres du groupe, dans lequel a été inscrit les objectifs de chaque jour des deux mois prévus. Les délais pour chaque tâche ont été le plus possible allongés afin de prévoir des possibles problèmes et de pouvoir les pallier sans soucis.

Etant donné que le sujet est nouveau pour les membres du groupe, les objectifs inscrits sont assez globaux mais entourent bien les réelles attentes nécessaires à se fixer afin de constituer le détecteur et l'interface dans les temps. Cela permet de se donner une idée générale avant de commencer à se pencher sur le sujet plus en profondeur. Ce planning prévisionnel était donc :

	Lundi	Mardi	Mercredi	Jeudi	Vendredi	Week-end
Semaine 0 25/01 – 31/01	Documentation sur les différents algorithmes de détection de visage	Documentation sur les différents algorithmes de détection de visage	Documentation sur les différents algorithmes de détection de visage	Documentation sur les différents algorithmes de détection de visage	Codage de l'algorithme de détection de visage	Codage de l'algorithme de détection de visage
Semaine 1 08/03 – 14/03	Documentation sur les algorithmes de reconnaissance d'objets utiles pour notre algorithme de détection de port du masque	Documentation sur les algorithmes de reconnaissance d'objets utiles pour notre algorithme de détection de port du masque	Documentation sur les algorithmes de reconnaissance d'objets utiles pour notre algorithme de détection de port du masque	Documentation sur les algorithmes de reconnaissance d'objets utiles pour notre algorithme de détection de port du masque	Codage de l'algorithme de détection de port du masque	Codage de l'algorithme de détection de port du masque
Semaine 2 15/03 – 21/03	Codage de l'algorithme de détection de port du masque	Codage de l'algorithme de détection de port du masque	Codage de l'algorithme de détection de port du masque	Codage de l'algorithme de détection de port du masque	Codage de l'algorithme de détection de port du masque	Codage de l'algorithme de détection de port du masque
Semaine 3 22/03 – 28/03	Codage de l'algorithme de détection de port du masque	Codage de l'algorithme de détection de port du masque	Codage de l'algorithme de détection de port du masque	Codage de l'algorithme de détection de port du masque	Codage de l'algorithme de détection de port du masque	Se refamiliariser avec le PyQt
Semaine 4 29/03 – 04/04	Coder l'IHM	Coder l'IHM	Coder l'IHM	Coder l'IHM	Coder l'IHM	Coder l'IHM
Semaine 5 05/04 – 11/04	Coder l'IHM	Coder l'IHM	Coder l'IHM	Coder l'IHM	Coder l'IHM	Coder l'IHM
Semaine 6 12/04 – 18/04	Rédaction compte rendu	Rédaction compte rendu	Rédaction compte rendu	Rédaction compte rendu	Rédaction compte rendu	Rédaction compte rendu
Semaine 7 19/04 – 25/04	Préparation soutenance orale	Préparation soutenance orale	Préparation soutenance orale	Préparation soutenance orale	Préparation soutenance orale	Préparation soutenance orale

Tableau 5 : Planning prévisionnel

Cependant, un planning prévisionnel est très souvent difficile à suivre à la lettre car on ne connaît pas encore toutes les contraintes mais également, il est compliqué de s'imaginer quelle tâche va prendre plus de temps qu'une autre tant qu'on n'est pas réellement rentré dedans.

C'est pour cela qu'après avoir pris connaissance de tous les enjeux liés à ce détecteur, il a été convenu avec le professeur référent de suivre un rythme précis jusqu'à la fin des projets : 1 semaine pour créer un algorithme de détection des visages, 2 semaines pour créer l'algorithme de détection du port du masque, 2 semaines pour créer l'interface et enfin 3 semaines de rédaction et de préparation de l'oral.

Ce projet a été rempli d'imprévus, de changements d'organisation et de remises en question. Mais cela n'a pas eu de réel impact sur le travail fourni car le planning était toujours là pour rassurer et pour rappeler que ces éléments de doutes et d'imprévus font partie intégrante du travail et sont prises en compte dans l'organisation initiale. Ainsi, voici le planning réel qui ressort à la fin du projet :

	Lundi	Mardi	Mercredi	Jeudi	Vendredi	Week-end
Semaine 0 25/01 – 31/01	<ul style="list-style-type: none"> - Lire les documents envoyés par M.HADJADJI + se documenter sur internet - Bilan de ce qu'on a compris jusqu'ici 	<ul style="list-style-type: none"> - Se documenter sur open CV + yolov3 - Réalisation mockups + réunion avec M.HADJADJI 	<ul style="list-style-type: none"> - Commencer partie 1 (détection etc.) + lire nouvel article + se renseigner sur les algos (SSD, YOLO...) - Choix algo + trouver BDD + commencer les tests d'algos 	<ul style="list-style-type: none"> - Vidéos YouTube pour comprendre open CV, tensorflow etc + commencer à coder détection visages 	<ul style="list-style-type: none"> - Envoi erreur code à M.HADJADJI + continuer à coder 	On continue à se documenter et coder
Semaine 1 08/03 – 14/03	<ul style="list-style-type: none"> - Réunion avec M.HADJADJI pour emploi du temps + montrer quels programmes tout faits on a trouvé + commencer détection visage (utiliser Pytorch) puis faire un point avec lui. + réussir à faire programme face detection. + difficile que ce que l'on pensait problèmes d'import , on va tester sur google collab 	<ul style="list-style-type: none"> - Import yolov5, pip, face_recognition --> journée settings - Succès pour installer dlib, cmake, face_recognition et nous regardons haarcascade 	<ul style="list-style-type: none"> - Chacun de notre côté on a avancé pour trouver un code autre que tensorflow pour une détection optimale et rapide du visage, on est tombé sur cafee model et open cv. CA MARCHE !! dnn / open CV/cafee model 	<ul style="list-style-type: none"> - Le prof a vu notre détecteur de visage et a validé. On garde nos anciens détecteurs pour comparer efficacité. - Tester la détection du visage sur BDD 100img avec masques (bien et mal portés) > Tester efficacité algorithme en % - Commencer à constituer BDD pour apprentissage 	<ul style="list-style-type: none"> - Test détecteur visage sur dataset 100 images (Mafa incorrectly wear) --> 100% détection - Test sur dataset 100 images (incorrectly wear et good wear) --> 95% - Début script pour cropper les visages dans les images de la BDD -> réduction du temps de calcul deep learning Réunion avec M.HADJADJI 	Algo isolement de visage dans BDD terminé. Début de documentation de CNN (vidéos)
Semaine 2 15/03 – 21/03	<ul style="list-style-type: none"> - Dataset (bien porté/mal porté/pas porté) terminé + on se documente sur CNN et réseaux de neurones/YOLO/détection 	<ul style="list-style-type: none"> on continue à se documenter sur CNN et réseaux de neurones/YOLO/détection 	<ul style="list-style-type: none"> - Appliquer le pre-processing sur le dataset (CROP + RESIZE 300) - Script pour ajouter les labels au dataset - Script pour trouver le meilleur paramètre "confidence" -> pas forcément utile au final 	<ul style="list-style-type: none"> - Entraîner un 2D-CNN sur le dataset constitué - Comparaison avec un modèle YOLOv5 custom 	<ul style="list-style-type: none"> -Entraîner un 2d-cnn sur dataset constitué --> échec car précision de 33%) 	Pistes à explorer : Prendre des modèles pré-entraînés et les entraîner à détecter les masques et les nez.
Semaine 3 22/03 – 28/03	<ul style="list-style-type: none"> - Reprise du modèle MobileNetV3 pré-entraîné puis entraînement sur notre dataset (précision max de 98%) - Tests unitaires sur nos visages (à faire) 	<ul style="list-style-type: none"> - Script comparaison entre les différents modèles de classification de torchvision - Script pour convertir les modèles ".pth" en ONNX puis Caffè2 pour être lus sur Open CV - PyQt commencé et terminé à 80% (il reste à régler la taille de l'interface, le choix de la webcam et à intégrer le code de détection du masque dans l'interface) 	<ul style="list-style-type: none"> - Sauvegarder les résultats du training et validation sur tous les modèles dans un fichier 'training-time.txt' - Test unitaire du modèle sur une image - Conversion en ONNX puis lecture avec Open CV -> Prédiction sur visage 	<ul style="list-style-type: none"> -Test des modèles intégrés à l'application (vérification des résultats de la validation) - Prédiction du port du masque basée sur les prédictions des 5 meilleurs modèles - Data augmentation : mettre des images des masques en noir & blanc pour détecter d'autres masques que 	<ul style="list-style-type: none"> - Test prédiction marche très bien sur Augustin mais pas bien sur Maël avec VGG16 - Ajout du calcul des moyennes de prédiction dans le code - Ré-implémentation du code dans interface (en cours) - Recherche pour intégrer YOLOv5 - Abandon de la partie choix webcam sur l'interface 	<ul style="list-style-type: none"> - Faire comparaison % validation combinaison 3 modèles // modèle tout seul - Faire la moyenne des 30 images pour l'envoyer au modèle - Changer le message en couleur en fonction de la

				les chirurgicaux - Étude learning-rate, optimizer, batch_size, epoch - Calcul des modèles sur 1500 img/classe : 12h		langue - Évaluer l'influence de BS pour un modèle => trouver BS critique - Faire varier le % de validation de 20% à 33% - Augmenter le nombre d'epoch à 50/100
Semaine 4 29/03 – 04/04	- Début Compte Rendu - Actualisation du planning - Ajouts de la réunion du vendredi	- Correction d'une coquille dans le code (envoi de webcam entière au lieu du visage seul) : très grosse amélioration de la prédiction - Début rédaction des comptes rendus de réunions	- Script pour comparer prédictions modèles seuls // combinaisons des modèles - Réunion avec M. HADJADJI - Élaboration d'un nouveau mockup pour améliorer l'interface - Toolbar : Options > Mode, Webcam Help : About, Instructions, Confidentiality - Codage de la nouvelle interface	- Finalisation de l'interface - Adaptation : resize de la nouvelle interface	- Prédiction sur plus d'un visage - Démonstration de l'interface pendant la réunion - Trouver un moyen pour actualiser chaque visage moins souvent - Remplir section About, Confidentiality et Instructions de l'IHM	- Ajouter mode sombre dans fenêtres secondaires - Avancer CR, CdC - Script comparaison moyenne / pondération - Script comparaison détection visage Caffe / Haarcascade - Resize fenêtre principale
Semaine 5 05/04 – 11/04	JOUR FÉRIÉ	- Test sur l'algo qui tourne pendant 3h : succès - Scripts de comparaison : moyenne ~ pondération DNN >> Haarcascade	- Début du Compte Rendu - Interface resizable simple	- Rédaction Compte Rendu	- Rédaction Compte Rendu	- Rédaction Compte Rendu
Semaine 6 12/04 – 18/04	- Rédaction Compte Rendu	- Rédaction Compte Rendu	- Rédaction Compte Rendu	- Rédaction Compte Rendu	- Rédaction Compte Rendu	- Rédaction Compte Rendu
Semaine 7 19/04 – 25/04	Préparation soutenance orale	Préparation soutenance orale	Préparation soutenance orale	Préparation soutenance orale	Préparation soutenance orale + script matrices de confusions	Préparation soutenance orale

Tableau 6 : Planning réel

Ainsi, ce qui peut ressortir de ce planning réel est que les prévisions de temps passées pour chaque partie du projet ont été respectées. Le fait d'avoir établi un planning prévisionnel assez large a permis à l'équipe de prendre le temps de réaliser chaque partie et même de recommencer entièrement l'IHM en cours de projet.

6.2 - La méthodologie de travail

A la suite des décisions du gouvernement liées à la crise sanitaire, ce projet a dû être réalisé à 100% en distanciel, ce qui n'est pas un avantage. Ainsi, il a fallu s'adapter et trouver une méthodologie efficace afin de communiquer facilement.

Pour cela, il a été décidé, dans un premier temps, d'effectuer tous les jours les conversations sur l'outil Discord. Un serveur pour ce projet a donc été créé, dans lequel ont été organisé des salles de réunions et des salles de conversations faites exprès en fonction du sujet. Cela était nécessaire pour ne pas se perdre dans les avancées de chacun et dans tout ce qui était transmis.

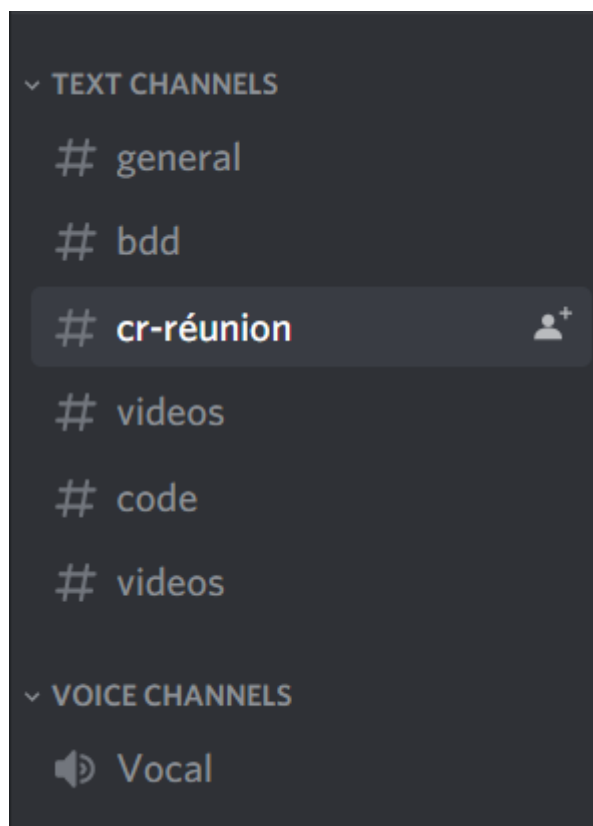


Figure 4 : Différentes salles de conversation Discord

Seulement, il a été rapidement décidé d'utiliser uniquement ce serveur pour les appels journaliers ainsi que pour l'envoi de vidéos ou d'informations rapides. Tout ce qui était des scripts de code Python ou des dossiers d'images ont été partagés sur un dossier commun Google Drive dans lequel il était plus facile de transmettre des fichiers lourds.

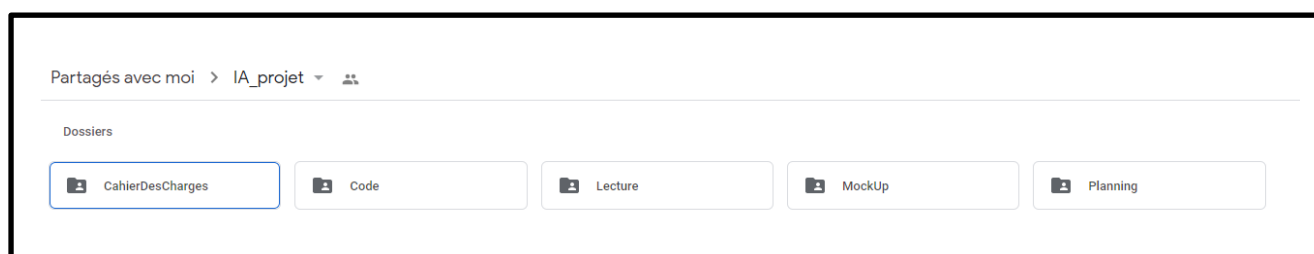


Figure 5 : Dossiers communs Google Drive

Afin d'avancer au mieux, un partage des tâches a donc été nécessaire. Pour cela, tous les matins à 8h30, les membres du groupe se réunissaient par appel Discord pour discuter du plan de la journée ainsi que de l'avancée de chacun. Il était décidé par la suite de qui allait faire quoi mais chacun

restait connecté sur le serveur afin de pouvoir échanger rapidement en cas de problème ou d'interrogation. Il était important de ne pas non plus être en appel toute la journée car il était facile de se distraire ou de partir dans tous les sens. A chaque fin de journée, chaque membre devait remplir son avancée sur le planning.

Chaque semaine, voire parfois plusieurs fois par semaine, il était convenu d'une réunion avec le professeur référent afin de parler des difficultés rencontrées, de montrer l'avancée du projet, de poser des questions ou de discuter des pistes possibles pour la suite. Pour permettre d'avoir un suivi précis régulier et de s'y retrouver facilement, un compte rendu de projet était rempli à la fin de chaque réunion (exemples de compte rendu : voir annexe 1).

6.3 - La Base De Données

Afin de réaliser le détecteur de visage et de port du masque en Intelligence Artificielle, la partie de constitution de la BDD est sans doute la plus importante.

Cependant, il est impossible de savoir à l'avance combien d'images seront nécessaires pour le dataset. On sait juste qu'avec les CNNs, plus la base est importante, plus la prédiction sera bonne.

Ainsi, afin d'avoir une BDD consistante, il a été décidé de récupérer 1500 images de chaque classe, c'est-à-dire 1500 images de personnes portant bien leur masque, 1500 images de personnes portant mal leur masque, et enfin 1500 images de personnes ne portant pas de masque.

Par ailleurs, cette problématique de détection de masque étant relativement nouvelle, il n'y a, à l'heure actuelle, pas de dataset faisant office de référence comme cela peut être le cas pour d'autres applications telles que la détection d'objets ou d'êtres vivants avec les datasets ImageNet ou COCO. [6] [7]

Les datasets qui existent s'avèrent incomplets sous plusieurs aspects.

Tout d'abord, il faut qu'un dataset ait une bonne représentativité ethnique pour ne pas induire de biais, ce qui est un sujet crucial à l'heure où même les GAFAM dont Google en particulier font face à des problèmes de nature éthique. [8]

L'IA doit réduire les inégalités, pas les accentuer.

Par ailleurs, un dataset donné doit inclure une variété de masques différents. En effet, bien qu'il y ait une prédominance des masques chirurgicaux, ces derniers se déclinent dorénavant en plusieurs coloris. De nombreuses personnes portent aussi des masques en tissus, bien que ces derniers soient un peu moins efficaces et tendent à être de moins en moins portés, l'application doit être en mesure de les reconnaître eux aussi.

Dans la plupart des cas, un dataset donné est complet sur un aspect mais pas l'autre, cela explique le fait qu'aucun dataset ne fasse office de référence.

Seul le dataset MaskedFace-Net [9], qui est un ensemble de données de visages humains avec un masque correctement ou incorrectement porté et sans masque (133783 images) basé sur l'ensemble de données Flickr-Faces-HQ (FFHQ), existe et pouvait être utile. Réutiliser ces images semblait être une réelle aubaine pour le projet mais en réalité cela était plus compliqué que prévu.

En effet, les images de la BDD doivent être diversifiées en termes de personnes (couleur de peau et origine) mais également en termes de couleur de masque, afin que le système arrive à détecter tout le monde sans problème. Ainsi, MaskedFace-Net a posé deux problèmes majeurs. Premièrement, malgré la bonne diversité des personnes présentes, il n'y avait que des masques chirurgicaux bleus, et deuxièmement, il y avait parfois plus d'une personne sur les images.

Il a donc fallu aller piocher dans d'autres datasets, [10] de tailles souvent bien moins importantes, des images de gens avec des masques bien portés et avec des masques mal portés, autres que les traditionnels bleus chirurgicaux. Étant donné qu'il fallait un total de 4500 images pour la BDD, ce ne fut pas une mince affaire, surtout qu'à part MaskedFace-Net, il n'existe aucun autre groupement d'images de personnes portant mal leurs masques. Grâce à quelques autres datasets existants présentant des personnes avec des masques pas uniquement chirurgicaux, il se pouvait que sur certaines photos le masque était mal porté. Cela s'explique par le fait que beaucoup de projets, concernant le port du masque, n'utilisent que deux classes (masque ou pas de masque) et classent les personnes portant un masque de manière incorrecte dans la catégorie des masques correctement portés. Ainsi, après de nombreuses heures de tri parmi les milliers d'images présentes, la BDD a finalement été constituée.

Cependant, il se posait toujours le problème énoncé précédemment concernant le nombre de personnes sur une image. De plus, les images rajoutées venant d'autres sites n'étaient pas de la même dimension. Il a donc fallu créer un algorithme de pre-processing afin d'envoyer au modèle une image contenant uniquement la tête de la personne répondant au critère de sa classe, et que toutes les images soient toutes dimensionnées pareil.

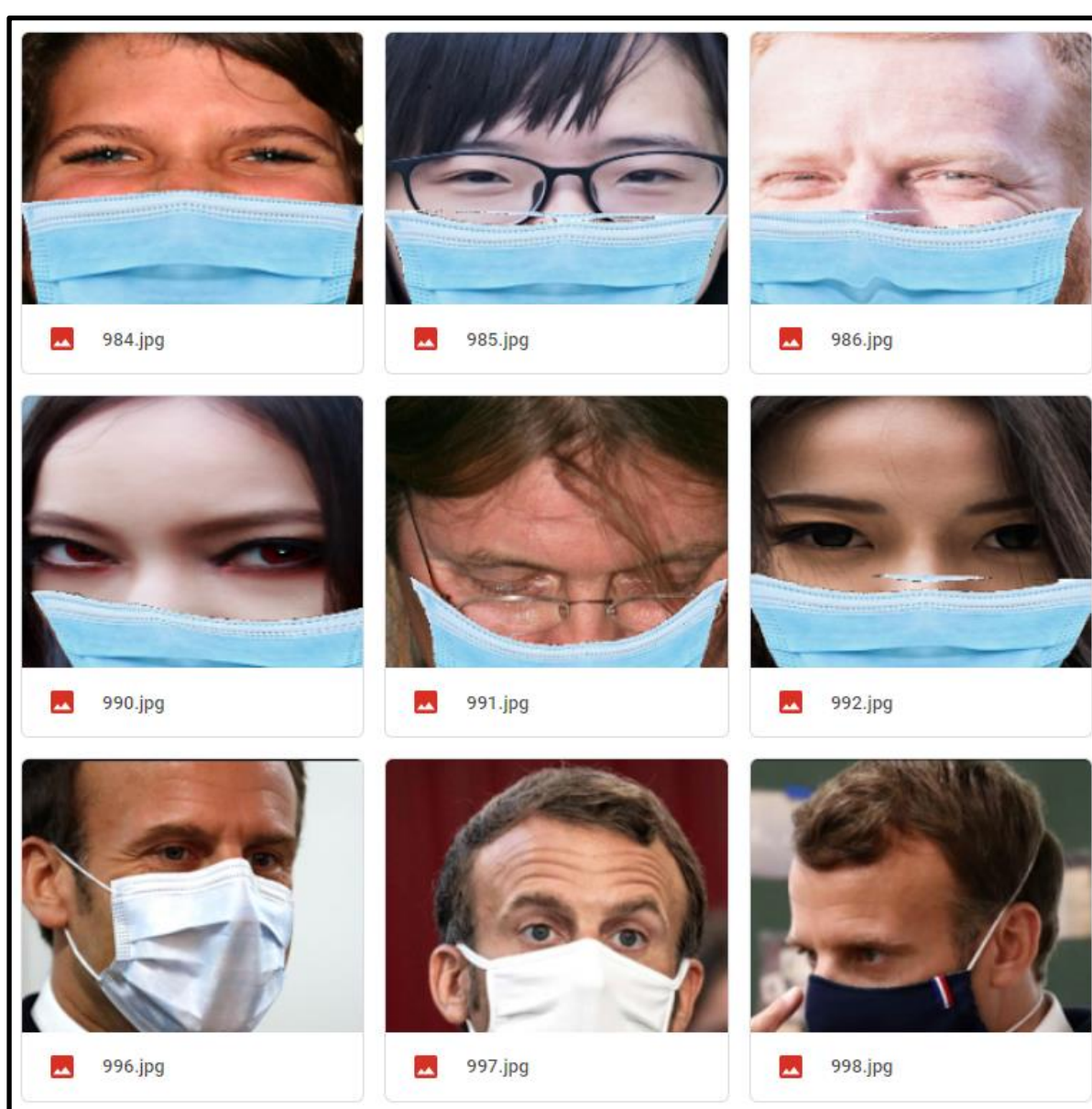


Figure 6 : Exemples d'images composant la classe « With Mask », après le pre-processing

Les 4500 images ont donc été regroupées par classe dans la BDD (With Mask, No Mask et Incorrect) après être passées par le pre-processing, afin d'être utilisées pour entraîner le modèle choisi.

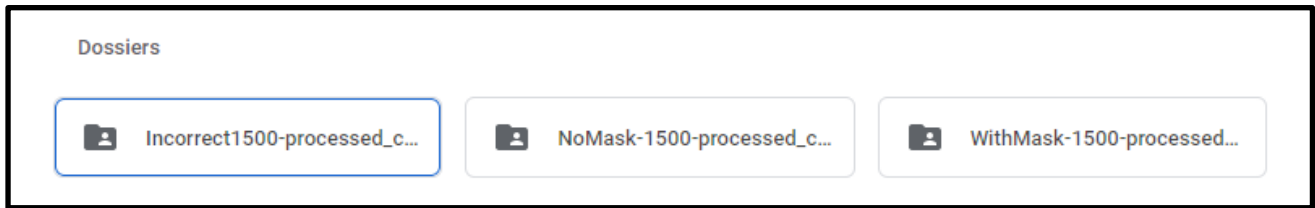


Figure 7 : Les 3 classes constituant la BDD

6.4 - Organisation de la partie détection de visage

Avant de pouvoir attaquer la partie de reconnaissance de port du masque, il fallait créer un algorithme capable de détecter plusieurs visages, masqués ou non. Pour cela, chaque membre du groupe devait essayer de concevoir le sien de son côté.

Après plusieurs recherches non concluantes sur des technologies avec des résultats de détection trop lentes ou encore une détection du visage partielle (non reconnaissance du visage de côté), le choix de partir sur un CNN pré-entraîné paraissait être la solution la plus adaptée.

Après s'être documenté sur les réseaux de neurones pré-entraînés à la reconnaissance de visage, il a été décidé de partir sur le CNN res10_300x300_ssd_iter_140000 utilisant la technologie Caffe Model. [11]

Cela a été de loin le résultat le plus concluant de toutes les technologies utilisées car cela permettait de reconnaître un visage de près ou de loin, de face ou de côté, avec ou sans masque, avec ou sans accessoire sur la tête, et avec une fluidité et une rapidité de détection optimale.

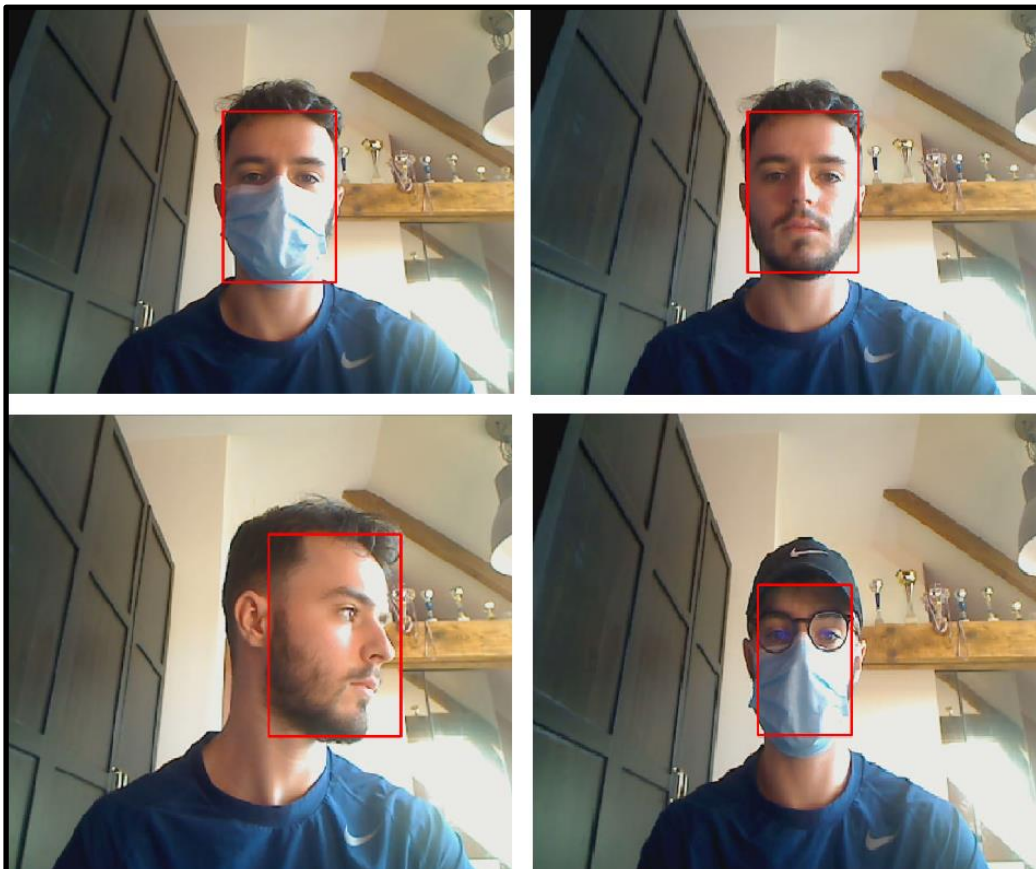


Figure 8 : Exemples de détection de visage avec le CNN ResNet-10

6.5 - Organisation de la partie détection du port du masque

Une fois l'algorithme de détection de visage créé, vient la partie de détection de port du masque. L'idée était également de partir sur CNN, mais cette fois-ci la mission était de tenter d'en créer un de A à Z.

Après s'être renseigné en profondeur sur ce qu'est ce réseau de neurones, comment il fonctionne et comment le paramétrer, chaque membre de l'équipe devait donc entraîner son propre CNN. Cependant, cette piste a été rapidement abandonnée, car après que chaque membre du groupe ait créé son propre réseau de neurone et l'ait entraîné sur le dataset, la précision sur la détection s'avérait bien trop faible. Elle ne dépassait jamais 33%, un résultat bien trop médiocre pour être gardé pour la suite du projet.

```
val accuracy: 0.3356
```

Figure 9 : Valeur moyenne de précision de détection après entraînement du modèle

Voici quelques exemples de détection sur le dataset du projet :

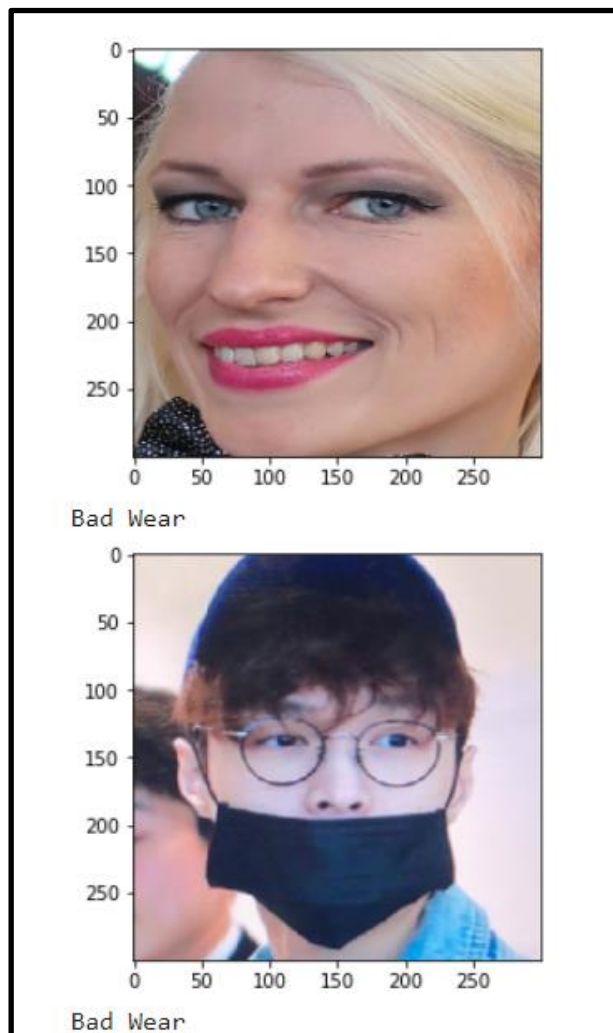


Figure 10 : Exemples de détection du CNN créé de A à Z par l'équipe

On se rend vite compte que le réseau de neurones a beaucoup de mal à prédire la bonne valeur (en moyenne 1 bonne prédiction toutes les 3 images) et qu'il conclut que le masque est incorrectement porté sur la première image alors que la personne n'en porte aucun.

La décision de repartir sur un modèle de CNN pré-entraîné a donc semblé plus judicieuse et efficace. L'idée était de réutiliser le CNN de détection de visage et de lui rajouter la détection de port du masque.

Après avoir entraîné 9 modèles, de la librairie Torchvision du framework Pytorch sur la BDD de 4500 images, les résultats de détection offline pour chaque modèle pré-entraîné étaient très bons.

En effet, certains modèles sortaient du lot et affichaient 98% de précision offline ce qui est largement au-dessus des 33% de précision des CNN créés précédemment.

Il restait à voir si ces modèles affichaient les mêmes résultats sur une détection online et il ressortait que certains modèles qui étaient très précis sur la détection offline, l'étaient moins sur des personnes réelles en utilisant la webcam.

Ainsi, après avoir testé chaque modèle avec la webcam, 3 sont ressortis particulièrement plus performants que les autres : VGG 16 [27], Resnext50 [28] et Densenet [29]. Il a donc été décidé de réaliser une combinaison de leur prédiction afin d'obtenir la meilleure possible. Seulement, il restait un petit problème de stabilité, parfois le détecteur mettait un certain temps avant de s'actualiser lors d'un changement de port du masque et se trompait de temps en temps lorsque le masque était mal porté.

Après avoir recherché d'où venait ce problème, la solution a rapidement été trouvée. L'image de la webcam envoyée au modèle était l'entièreté de ce qu'elle captait et non uniquement le visage de la personne testée. Le pre-processing n'était pas activé sur le mode online et ralentissait les performances de détection du modèle qui devait donc effectuer un travail supplémentaire.

La correction de cette coquille a énormément amélioré la détection de port du masque online qui était déjà assez concluante. Peu importe comment le masque est porté, le détecteur prédit quasiment instantanément correctement.



Figure 11 : Exemples de détection de port du masque

Une fois le détecteur de port masque fini, le professeur référent a exigé la possibilité que le système puisse détecter le port du masque sur plusieurs visages en même temps. Ce qui n'était pas le cas jusqu'à présent car si la détection de visage fonctionnait sur plusieurs personnes, il n'en n'était pas de même pour la détection des masques qui se concentrait uniquement sur un seul visage.

Ainsi, le programme a dû être adapté pour s'adapter à la nouvelle consigne donnée.

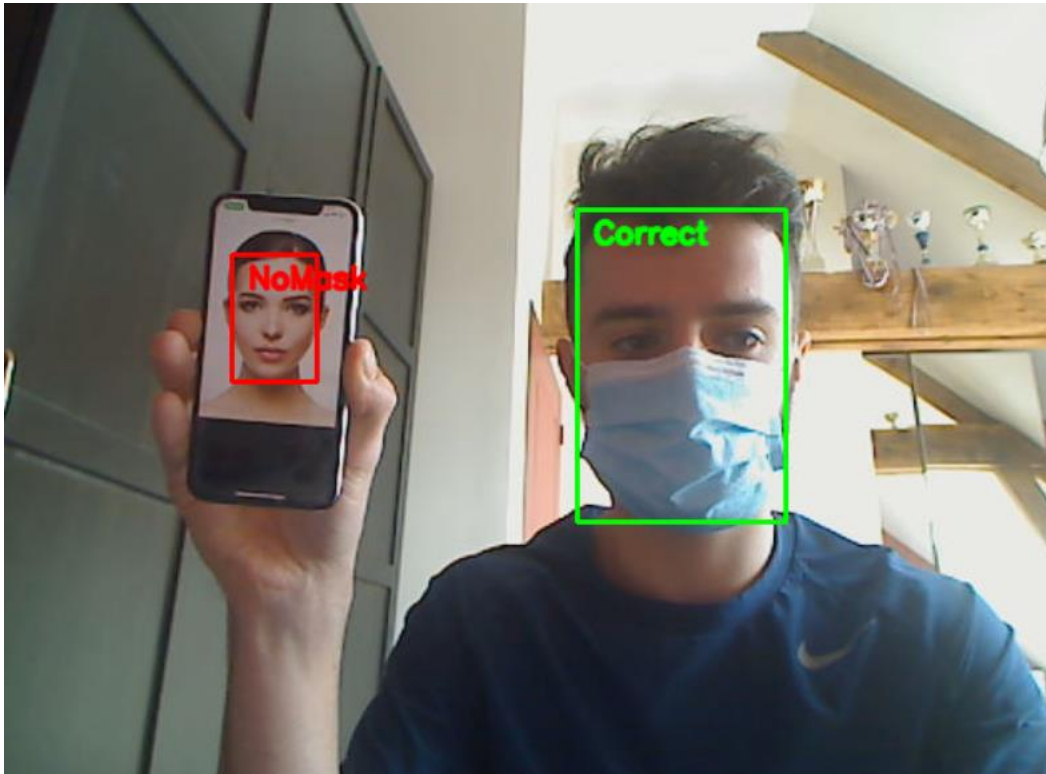


Figure 12 : Exemple de détection de plusieurs visages en même temps

Cependant, lors de l'apparition d'une nouvelle personne à détecter, l'algorithme inverse la première prédiction entre les visages, mais se stabilise moins d'une seconde après. Il y a donc une stabilisation rapide nécessaire au système pour détecter de manière optimale, à l'apparition de nouvelles têtes.

6.6 - Organisation de la partie IHM

La constitution d'une Interface Homme-Machine est une étape très importante dans le projet car elle lie l'algorithme et l'utilisateur. En effet, elle doit pouvoir permettre à un utilisateur lambda de pouvoir utiliser le détecteur de port du masque de manière simple, et d'avoir un retour rapide et visuel sur comment son masque est porté et ce qu'il doit faire si jamais il le porte mal.

Ainsi, les membres du groupe sont partis dans un premier temps sur une IHM très simple qui devait pouvoir afficher toutes les informations importantes à l'utilisateur dès son arrivée sur l'interface.

En effet, l'idée était que tout soit accessible, sans avoir à cliquer sur des boutons, à la manière d'un détecteur existant sur le web à cette adresse : <https://facemask-detection.com/>.

Ainsi, il devait figurer sur la gauche, un petit onglet présentant les instructions à respecter pour obtenir la meilleure détection possible. En-dessous de cet onglet, il devait y avoir un petit menu déroulant permettant de choisir sa webcam, enfin encore en-dessous, il devait se trouver un bouton menant vers une page présentant la politique de confidentialité.

Au milieu de l'IHM, il devait y avoir un grand rectangle contenant le retour de la webcam sur l'utilisateur avec la détection sur son visage du port du masque ainsi que la prédiction faite, et en-dessous de l'image, le conseil à suivre pour se protéger efficacement du COVID-19 en fonction du port du masque de l'utilisateur.

Enfin, sur la droite de l'interface devait se trouver 3 drapeaux (Espagnol/Français et Anglais) permettant de changer la langue.

Voici le mockup réalisé en début de projet :

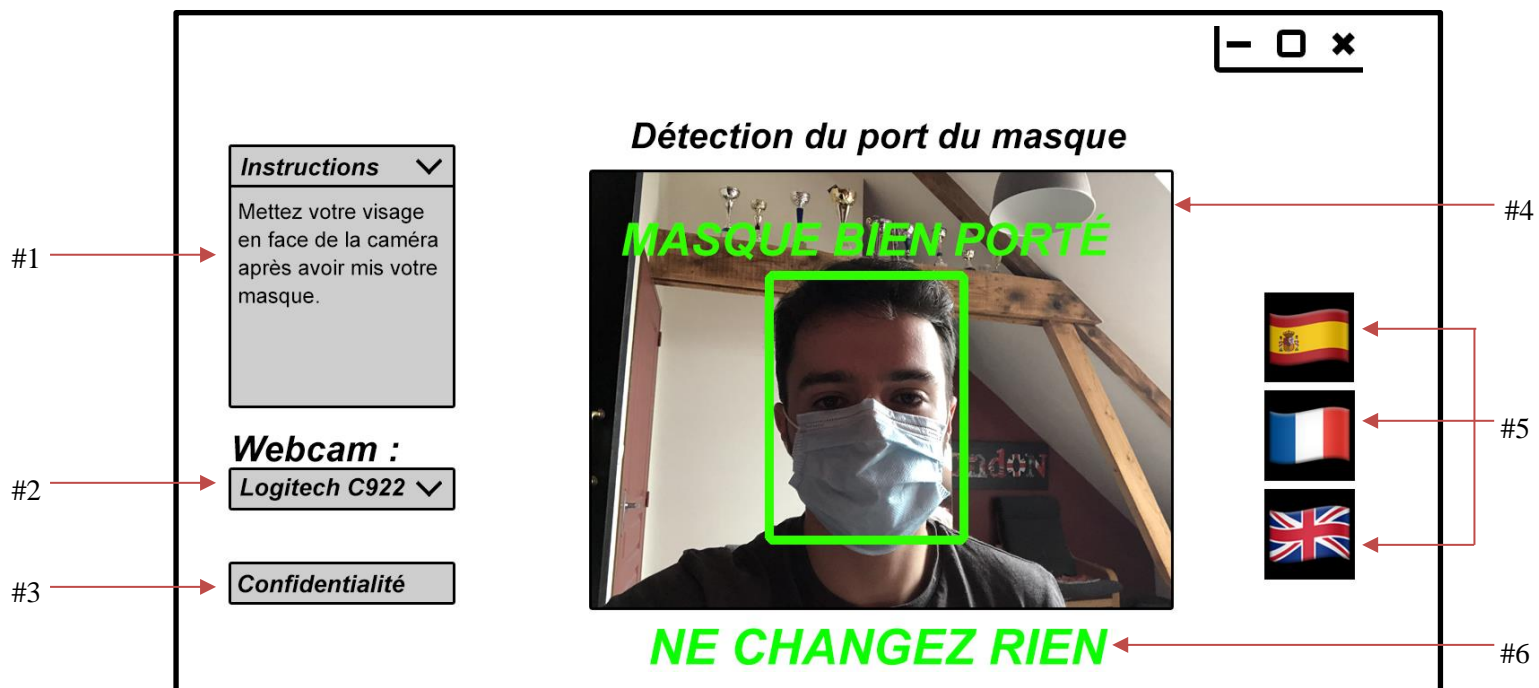


Figure 13 : Mockup de la première interface réalisée

Dans ce mockup, on retrouvait ce qui était attendu, c'est à dire :

#1 : Les instructions

#2 : Le choix de la webcam

#3 : La politique de confidentialité

#4 : Le visuel de détection de port du masque

#5 : Le choix de la langue

#6 : Le conseil de protection

Voici le rendu, codé en PyQt, de l'IHM correspondant au mockup précédent :

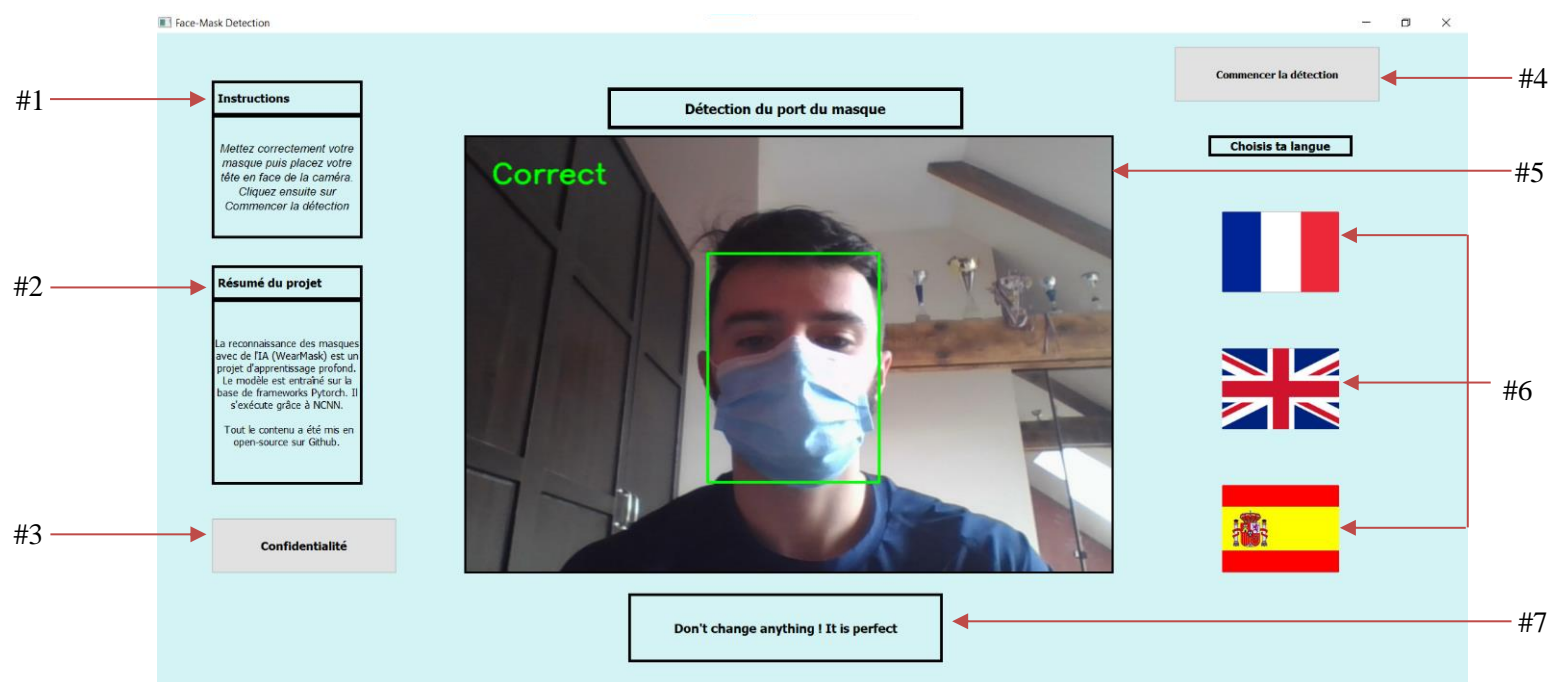


Figure 14 : Première interface réalisée

Dans cette première version de l'interface, se trouvaient :

#1 : Les instructions

#2 : Le résumé du projet

#3 : La politique de confidentialité

#4 : Bouton pour commencer la détection

#5 : Le visuel de détection de port du masque

#6 : Le choix de la langue

#7 : Le conseil de protection

On y retrouve presque tout ce qui était évoqué dans le mockup sauf 3 paramètres : le choix de la webcam, le bouton pour commencer la détection et le résumé du projet.

En effet, cette idée de pouvoir choisir la webcam a été abandonnée par les membres du groupe, ne jugeant pas cette option nécessaire, préférant la remplacer par le résumé du projet.

Ensuite, le bouton pour commencer la détection est nécessaire afin de lancer la webcam. La lancer à l'infini, sans que l'utilisateur veuille forcément que la détection commence, semblait inapproprié.

Cependant, il y avait un petit problème avec cette interface. Si elle était déployée sur un ordinateur d'une résolution d'écran différente du PC ayant servi à sa création, l'interface ne s'ajuste pas bien. En effet, les composants sur la gauche et la droite du visuel de la webcam peuvent se masquer si l'utilisateur veut changer les dimensions de la page, ce qui pose un réel problème.

Ainsi, il a été décidé en concertation avec les membres du groupe et le professeur référent de recréer une nouvelle interface plus épurée, en disposant les éléments dans une barre de menu.

Cette méthode permet de ne pas afficher des composants sur les côtés de l'interface, et donc enlève la possibilité qu'ils deviennent masqués en cas de changement de dimension.

Voici le nouveau mockup de l'interface finale :

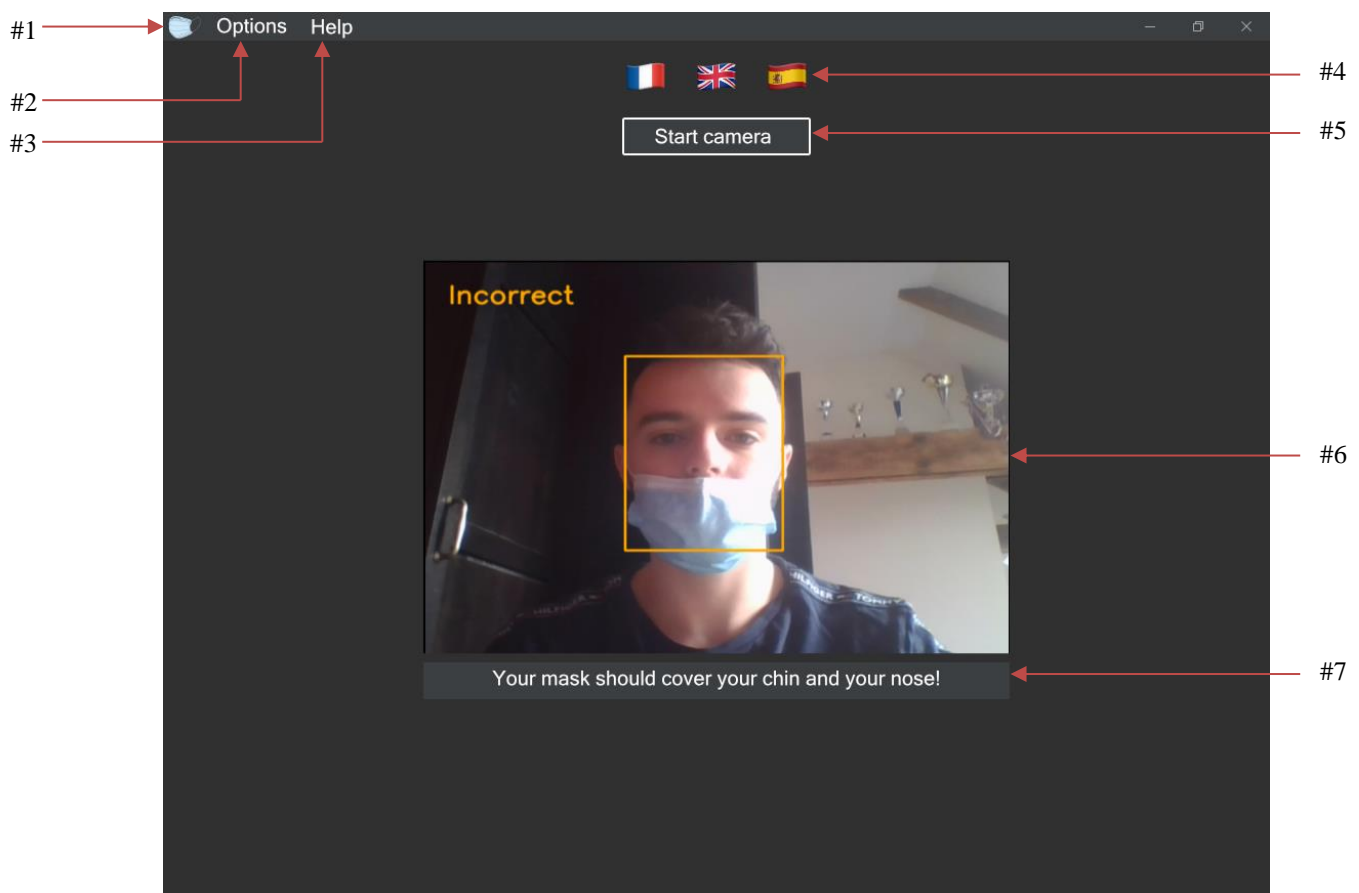


Figure 15 : Mockup de l'interface finale

Dans cette nouvelle interface, se trouvent :

- #1 : Icône libre de droit représentant un masque
- #2 : Menu « Options » dans lequel il sera possible de sélectionner la webcam que l'utilisateur souhaite utiliser parmi celles détectées, ainsi que de changer le thème de l'interface (sombre ou clair)
- #3 : Menu « Help » dans lequel il sera possible de voir les instructions pour bien utiliser l'interface, la politique de confidentialité ainsi que le résumé du projet
- #4 : Trois boutons représentant des drapeaux permettant de changer la langue de l'interface
- #5 : Bouton permettant de lancer la détection de port du masque sur l'utilisateur en fonction de la webcam choisie (webcam de l'ordinateur par défaut)
- #6 : Rendu de la détection du port du masque sur l'utilisateur après appui sur le bouton de lancement de la détection, avec la prédiction
- #7 : Conseil à l'utilisateur en fonction de comment il porte son masque

On retrouve dans cette interface la possibilité de choisir sa webcam, option qui figurait dans le premier mockup, mais qui avait été finalement abandonnée. Ce choix de webcam se révèle finalement nécessaire car ce système peut être réutilisé par un utilisateur qui possède une webcam d'ordinateur défectueuse et qui doit, pour faire fonctionner le détecteur, utiliser une webcam externe.

De plus, l'option de choix du thème est une option qui a été rajoutée pour permettre à l'utilisateur d'avoir une luminosité optimale en fonction de ses préférences.

Voici l'interface finale codée en PyQt :

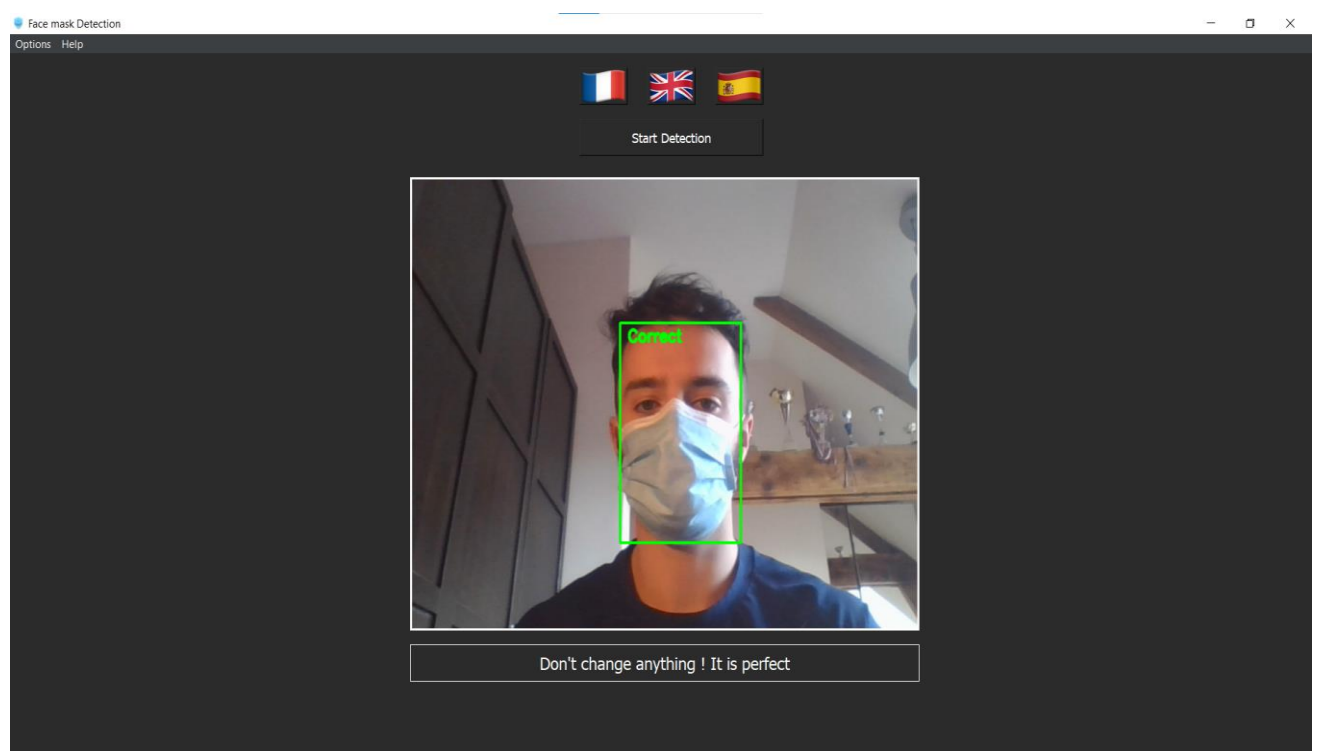


Figure 16 : Interface finale

Cette interface finale est en adéquation avec le mockup réalisé et contient toutes les options communiquées précédemment. Les images de la comparaison de l'interface en mode jour et sombre est accessible en annexes 2 et 3, ainsi que de les parties instruction, confidentialité et résumé du projet en annexes 4,5 et 6.

Ainsi, cette IHM se révèle plus épurée et plus professionnelle que la première. De plus, elle permet de réduire considérablement les problèmes liés au changement de résolution des écrans pouvant accueillir l'interface.

Étant codée depuis Qt Designer, cette interface se révèle assez esthétique, mais les positions des boutons du milieu ainsi que du retour webcam sont codées en dur. Ainsi, même si les problèmes de changement de dimension ont été largement réduits, il a été décidé de réaliser une IHM secondaire totalement redimensionnable sur n'importe quel ordinateur.

Cette nouvelle interface n'est pas faite pour être esthétique, mais s'adresse uniquement à des personnes souhaitant utiliser le détecteur avec une résolution différente de 1920*1080p.

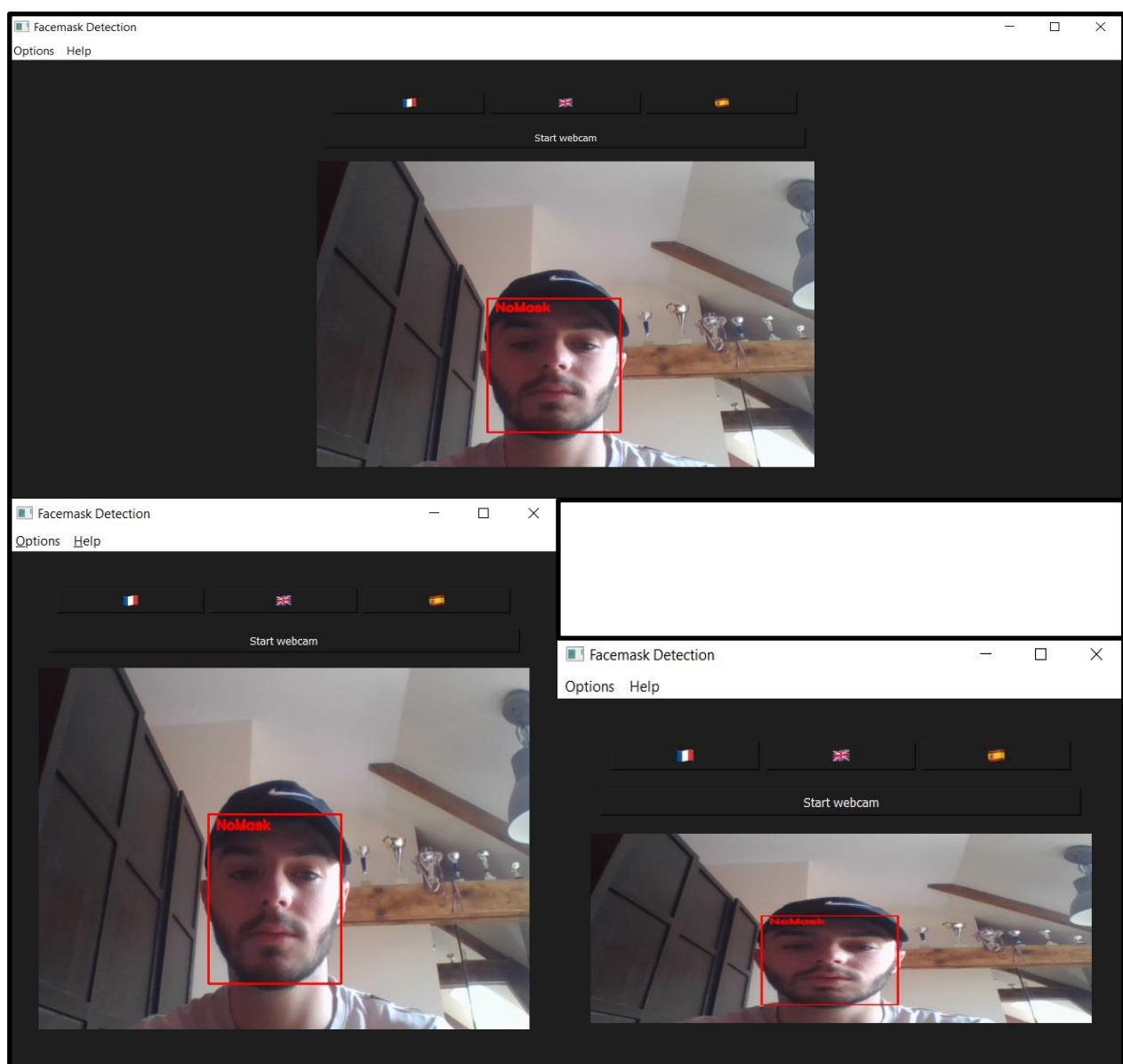


Figure 17 : Exemples de l'IHM secondaire selon différentes dimensions

Cette deuxième interface n'affiche donc pas de conseil afin de porter son masque de manière optimale, elle a été créée principalement pour permettre à tout le monde de bénéficier du détecteur, peu importe la dimension d'écran de son ordinateur.

En codant une IHM directement en Qt Designer, il est plus difficile d'obtenir une interface qui se veut redimensionnable et très esthétique car les positions des objets sont codées en dur. En utilisant cette application, l'aspect esthétique est privilégié à l'aspect pratique.

Désormais, n'importe quel utilisateur peut bénéficier du détecteur de port du masque.

7 – Développement technique

7.1 – Spécifications

7.1.1 - Langage de programmation

En premier lieu, le choix du langage de programmation était le premier choix qui devait être opéré.

Ce projet étant un projet de computer vision, le choix s'est axé sur la compatibilité d'OpenCV avec les différents langages de programmation que chaque membre de l'équipe a utilisé au préalable.

OpenCV est la librairie qui fait office de référence dans le milieu de la computer vision. La librairie est officiellement disponible pour deux langages : le C++, librairie native du framework, puis en Python où la librairie est également très répandue. [12] [13]

Bien que la librairie ne soit pas développée en Python, son implémentation est très complète de par le besoin grandissant d'avoir une librairie de computer vision solide et stable sur le langage haut-niveau.

Ce dernier étant le plus complet quand il s'agit d'IA. C'est le langage de programmation pour lequel le plus de librairies relatives au domaine sont disponibles. [14]

Par conséquent, c'est aussi le langage pour lequel le plus de ressources d'aides et de tutoriels sont disponibles.

Ainsi, la dernière version de Python à ce jour, la version 3.9 s'est avérée comme le choix le plus évident quand il a fallu choisir un langage de programmation.

7.1.2 - Système d'exploitation

Les ordinateurs sur lesquels ont été développés l'application fonctionnent sous Windows 10, c'est donc pour ce système d'exploitation que vont être détaillés les prérequis pour faire fonctionner l'application.

Les spécifications techniques de l'ordinateur ayant été utilisé pour calculer les modèles sont les suivantes :

- CPU : i5-4590 CPU @ 3.30GHz
- RAM : 8 GB
- GPU : GTX 970 4 GB GDDR5

En premier lieu, pour utiliser OpenCV sur Windows, il est nécessaire d'installer des composantes de Microsoft Visual Studio qui permettent l'exécution du code C++ dans lequel est codé la librairie.

[15]

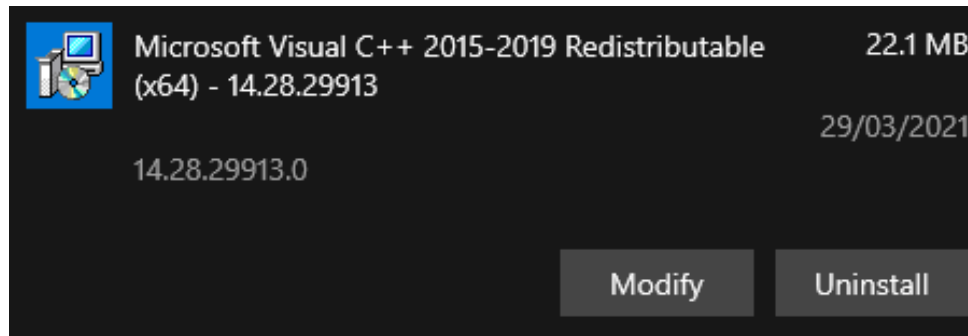


Figure 18 : Librairie Microsoft C++ nécessaire pour OpenCV

7.1.3 - IDE

Tout au long du processus de création, les algorithmes ont été développés à l'aide de l'IDE Pycharm, en utilisant la technologie d'environnement virtuel venv qui permet d'installer, spécifiquement pour un projet donné, les librairies utiles.

Cette technologie crée une copie de la version de Python spécifiée et installe sur cette dernière les librairies utilisées pour le projet.

Cela s'avère pratique lorsque l'on travaille sur plusieurs projets de manière simultanée et que ces derniers fonctionnent avec des versions différentes d'une même librairie, par exemple.

7.1.4 - Framework

Dans le cadre de ce projet, 3 frameworks ont été utilisés :

- OpenCV : pour manipuler et lire des images, dont celles provenant de la webcam
- PyTorch : pour créer et manipuler des réseaux de neurones [16]
- PyQt5 : une librairie qui permet de créer des applications graphiques en Python [17]

Plus haut dans cette section a déjà été abordé le choix d'OpenCV en tant que librairie de computer vision.

Concernant le choix du framework de Deep Learning, PyTorch a été recommandé par le professeur référent pour sa flexibilité et sa manipulation plus simple que son principal concurrent : Tensorflow, développé par les équipes de Google.

Par ailleurs, l'installation de Tensorflow pour une utilisation locale est un peu plus ténue que celle de PyTorch du fait que le framework est pensé pour fonctionner en synergie avec Google Colab, l'éditeur de code Python en ligne développé par la firme.

Comme l'objectif est de proposer une application exécutable en local, il était préférable de partir dès le départ sur un environnement local afin d'éviter des problèmes lors de la transposition du code en ligne vers une exécution locale.

Le choix du framework d'interface graphique utilisateur s'est appuyé sur les précédentes expériences de l'équipe qui a travaillé sur l'application.

PyQt est un framework qui se base sur Qt, qui permet la création d'interface graphique en utilisant le C++. PyQt permet donc de réutiliser ces fonctions développées en C++ au sein du langage Python.

PyQt a été préféré à PySide car bien que les deux frameworks proposent des fonctionnalités similaires, l'implémentation plus complètes des fonctions de PyQt a fait peser la balance en sa faveur.

7.1.5 - Résumé

Pour faire simple, en dehors de l'OS, de Python et des composantes de Microsoft Visual Studio qui permettent l'exécution du code C++, nous pouvons résumer les installations à réaliser dans l'environnement virtuel à ces 3 lignes de commandes dans le terminal de l'IDE :

- `pip install opencv-python`
- `pip install PyQt5`
- `pip install torch==1.8.1+cu111 torchvision==0.9.1+cu111 torchaudio==0.8.1 -f https://download.pytorch.org/whl/torch_stable.html`

7.2 – Schémas, organigrammes

Lors de l'élaboration du planning prédictif, comme décrit dans la partie organisation de projet, le travail à réaliser se divisait en plusieurs blocs :

- Pre-processing : redimensionnement des images, application de filtres
- Dataset : constitution d'une base de données équilibrée comportant plusieurs types de masques et représente l'ensemble de la population mondiale
- Entraînement de plusieurs CNNs, puis sélection de celui qui donne les meilleurs résultats sur la base de validation
- Détection des visages dans les images capturées par la webcam
- Envoi des visages au CNN qui détermine si le masque est bien ou mal porté, voire pas porté du tout
- Affichage du tout dans une interface graphique simple d'utilisation pour les usagers

Voici une représentation schématique des liens entre chaque bloc du projet énoncés ci-dessus :

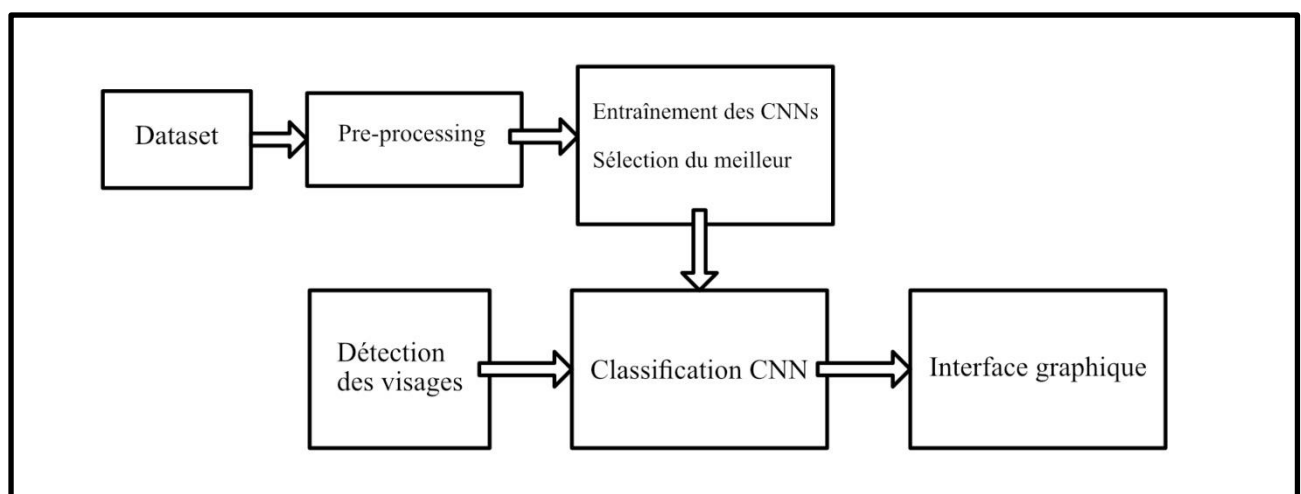


Figure 19 : Schéma de la V1 du projet

Plus tard, lors de la finalisation du bloc de classification, plutôt que de faire reposer la prédiction de la classe sur la décision d'un seul modèle de CNN, il a été décidé de combiner les décisions des 3 meilleurs modèles pour assurer une meilleure stabilité dans les prédictions.

Cela a donc modifié la représentation schématique du projet de la manière suivante :

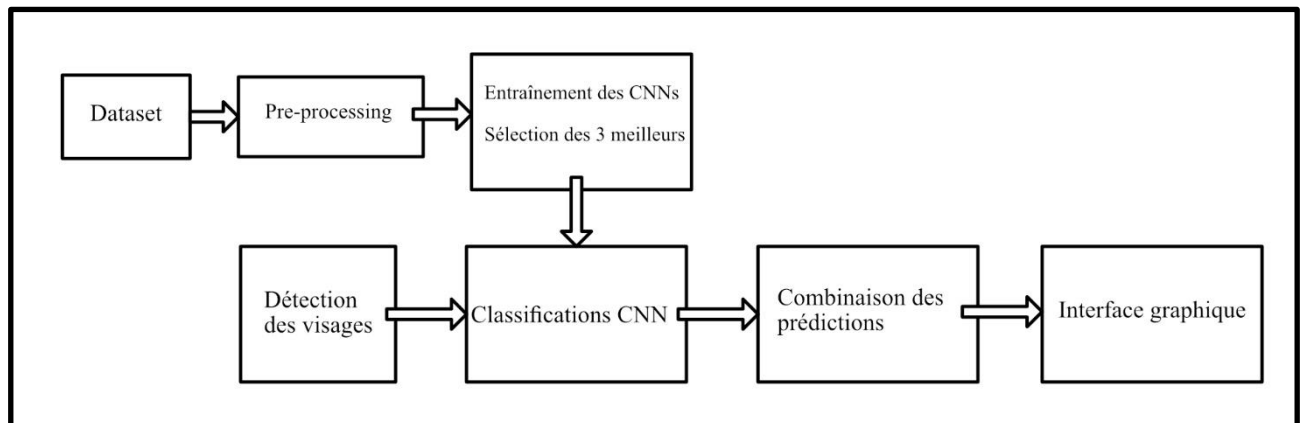


Figure 20 : Schéma de la version définitive du projet

7.3 – Contributions techniques

7.3.1 - Constitution du dataset

Comme mentionné dans la partie “Organisation de projet”, le choix d'un dataset n'était pas évident du fait d'un manque de dataset de référence pour les exercices de reconnaissance du port du masque.

Si un dataset en particulier venait à être utilisé, il induirait un biais lors de l'apprentissage des modèles de CNN, ce biais serait un biais de sur-apprentissage.

Cela signifie concrètement que le modèle aurait eu du mal à généraliser la tâche qui lui était impartie.

Dans ce cas précis, si le dataset utilisé avait été MaskedFace-Net [18], les CNNs entraînés auraient seulement prédit correctement le port du masque chirurgical de couleur bleue car le dataset ne comprend que des masques de cette couleur. [19]

Si aucun autre dataset comportant des images de personnes masquées n'existait, il aurait été possible de se contenter de ce dataset et de générer des images avec des masques de couleurs différentes en modifiant la teinte de l'image, c'est une méthode que l'on appelle “data augmentation”.

Cependant, il y a des inconvénients à cette méthode.

En premier lieu, modifier la teinte du masque, en modifiant celle de l'image, modifie également la teinte de la peau de l'individu qui porte le masque.

Il n'y a donc pas de moyen sûr de vérifier que cela n'induit pas un nouveau biais avant de procéder à l'apprentissage des modèles sur le dataset augmenté.

La méthode privilégiée a donc été de combiner plusieurs datasets différents pour contrebalancer les biais induits par les autres.

Pour rappel, les deux principaux biais que l'on peut attribuer à la plupart des datasets qui peuvent être trouvés en ligne sont les suivants : biais de représentation ethnique, biais de couleur du masque.

Ces combinaisons de datasets ont été réalisées par sélection manuelle de sorte à équilibrer le dataset final.

Lors de la première phase d'apprentissage des modèles, le dataset comportait environ 1500 images, soit 500 par classe.

Au sein de ces 500 images, 200 provenaient du dataset Face Mask Detection [20], tandis que les 300 restantes provenaient de MaskedFace-Net.

Puis, afin d'observer l'impact de la taille de la base d'apprentissage, la taille du dataset a été augmentée à 1500 images par classe, portant le total du dataset à environ 4500 images.

Par simplicité, il a été décidé de rajouter uniquement des images provenant de MaskedFace-Net car ces dernières n'avaient pas besoin de tri manuel, ce qui est une activité relativement chronophage.

7.3.2 - Pre-processing

Une fois le dataset constitué, il était nécessaire de produire un script en mesure de traiter les images avant de les envoyer aux CNNs dans le but d'entraîner ces derniers.

Le pre-processing est l'étape qui met en forme les images et harmonise leur forme pour que leurs caractéristiques en dehors de ce qui est pertinent (c'est-à-dire si le masque est bien/mal/pas porté) soient communes.

Cette étape est très importante, elle permet d'orienter les réseaux de neurones vers ce qui est attendu d'eux.

En premier lieu, grâce à l'algorithme de détection de visage qui sera détaillé plus bas au cours de ce rapport, il y a détection des coordonnées du visage principal dans l'image et extraction.

Le reste de l'image est ainsi supprimé, l'information principale est extraite, il ne reste plus qu'à appliquer des transformations standardisées.

En second lieu, le standard de pre-processing sur le dataset ImageNet, dataset sur lequel les modèles sont pré-entraînés, est le suivant [21]:

- Les images sont de la taille 224*224 pixels, elles comportent chacune 3 canaux RGB (Red Green Blue)
- Il faut normaliser les valeurs des canaux RGB de chaque pixel. Les moyennes des canaux RGB sont définies respectivement comme cela : (0.485, 0.456, 0.406)
Les écarts-types sont définis respectivement comme cela : (0.229, 0.224, 0.225)

Une fois ces transformations appliquées, le dataset est prêt à l'emploi.

7.3.3 - Entraînement des CNNs

Au terme de multiples tentatives de programmation de CNNs à la main "from scratch". Il a été décidé, après plus de recherches sur les pratiques courantes en computer vision, d'employer une méthode de Transfer Learning : le Features Extracting.

Concrètement, des modèles entraînés à détecter 1000 objets différents [22], dont des animaux ou bien des voitures sont ici repris pour détecter les 3 “objets” de l’application : visage sans masque, visage avec masque mal porté et visage avec masque bien porté.

L’entraînement consiste à reprendre les poids calculés au préalable pour ces modèles qui s’avèrent complexes de par leur nombreuses couches et opérations effectuées, puis à réajuster les poids des dernières couches.

Par ailleurs, dans cette application, du fait qu’on ne cherche pas à reconnaître 1000 instances différentes mais seulement 3, il convient de redimensionner la taille de la couche de sortie.

L’apprentissage par les modèles est standardisé de la manière suivante :

- Le nombre d’epoch est de 10.
C’est le nombre de fois que le dataset sera parcouru dans son entièreté pour affiner le modèle
- La batch size est de 10.
C’est le nombre d’éléments du datasets qui seront envoyés de manière simultanée au modèle
- Le learning rate est de 0.001. C’est la valeur par laquelle sont ajustés les poids.
- L’optimizer utilisé est l’algorithme du gradient stochastique, abrégé SGD en anglais, ce dernier coordonne l’ajustement des poids.

Pour mener à bien cette étape, il y a eu conception d’un script qui automatise l’entraînement d’une liste de modèles donnée, et sauvegarde dans un fichier les poids de ces derniers pour leur meilleure valeur de précision sur la base de validation au cours des 10 epochs.

Avant de procéder à l’entraînement, le script divise en base d’apprentissage et de validation le dataset qui lui est transmis.

Afin de pouvoir retenir les modèles ayant réalisés les meilleures précisions, le script sauvegarde non seulement les poids de chaque modèle mais aussi les précisions de chaque modèle à chaque epoch en les enregistrant dans un fichier texte.

La liste des modèles entraînés a été constituée à partir de ceux mis à disposition par PyTorch pour la classification d’images.

Influence de la variation de la taille de la base de données

Dans un premier temps, les modèles ont été entraînés sur le dataset initial qui comportait seulement 1500 images au total.

Au sein de chaque classe, 100 images étaient affectées à la base de validation et le reste à la base d’entraînement.

Les résultats se sont montrés très rapidement plus concluants que ceux des CNN “from scratch” qui avaient une précision s’élevant à 33%.

En effet, avec cette petite base de données, la précision s’élève tout de même à 99% pour le meilleur modèle.

```
The 5 most accurate models are:  
VGG : 0.99  
MobileNetV3 : 0.98  
DenseNet : 0.97  
ResNet : 0.94  
ResNet : 0.94
```

Figure 21 : Précision des 5 meilleurs CNNs avec le dataset de 1500 images

Par la suite, et sur les conseils du référent, la base de données utilisée pour entraîner les réseaux de neurones a été élargie à plus de 4500 images au total.

La routine d'apprentissage a donc été exécutée une nouvelle fois.

Bien que les résultats soient un peu plus faibles, les modèles n'en sont pas moins précis pour autant, ils témoignent simplement du fait que la base de validation a elle aussi triplé de volume et par conséquent, statistiquement, plus d'erreurs surviennent même si les différences de précisions restent minimales et cohérentes par rapport aux premiers calculs.

Certains modèles s'en sortent mieux avec cette base de données plus grande, et les modèles qui avaient déjà des bons résultats avec le plus petit dataset s'en sortent toujours bien.

```
The 5 most accurate models are:  
DenseNet : 0.9866  
VGG : 0.9766  
MobileNetV3 : 0.9733  
ResNet : 0.9733  
MNASNet : 0.9633
```

Figure 22 : Précision des 5 meilleurs CNNs avec le dataset de 4500 images

Ces résultats offline confirment que l'approche du Features extracting est plus que satisfaisante pour la problématique de détection du port du masque.

Influence de la variation de la taille de la base d'apprentissage

Dans un second temps, la nécessité d'étudier l'influence de la taille de la base d'apprentissage sur la précision des modèles s'est imposée.

Pour cette étude, la même taille a été conservée pour la base de validation, cette dernière était donc fixée à 900 images au total, soit 300 pour chaque classe.

Les modèles pour lesquelles les précisions ont été recalculées suivant ces nouveaux paramètres sont ceux qui figurent parmi les 5 plus précis à l'issue des apprentissages précédents.

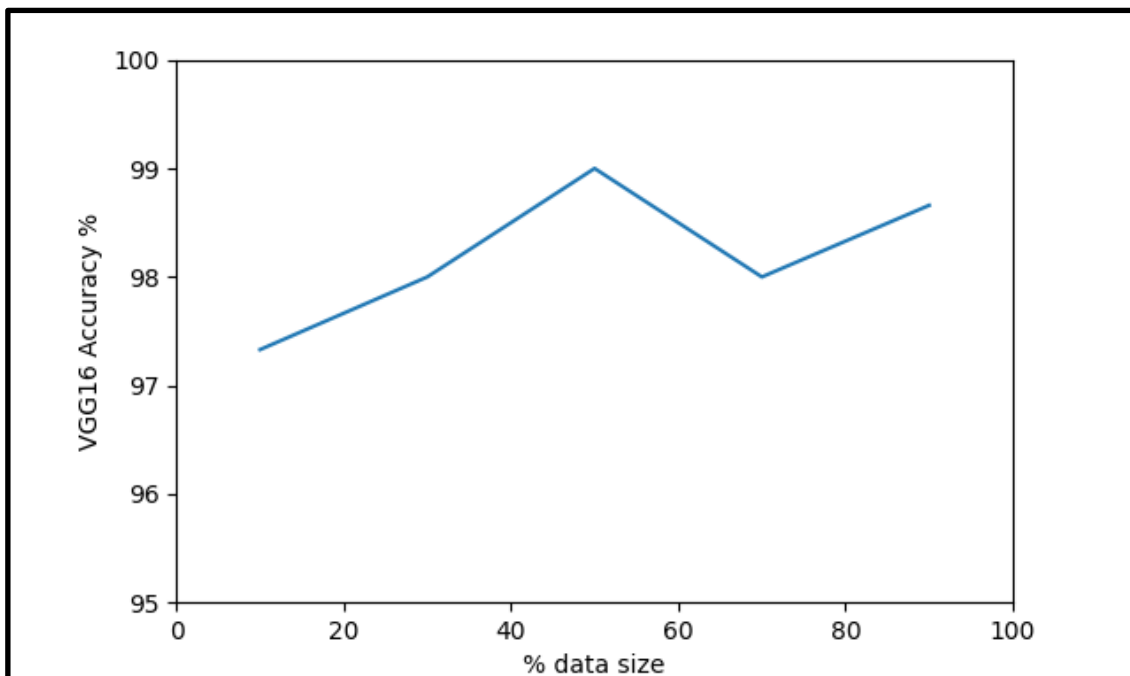


Figure 23 : Précision de VGG16 en fonction de la taille de la base d'apprentissage

Les résultats décrivent une précision relativement importante du modèle VGG16, et ce malgré une taille de dataset 10 fois moins importante pour apprendre. Sa meilleure précision est atteinte lorsque la taille de la base d'apprentissage est égale à 50% de la taille originelle.

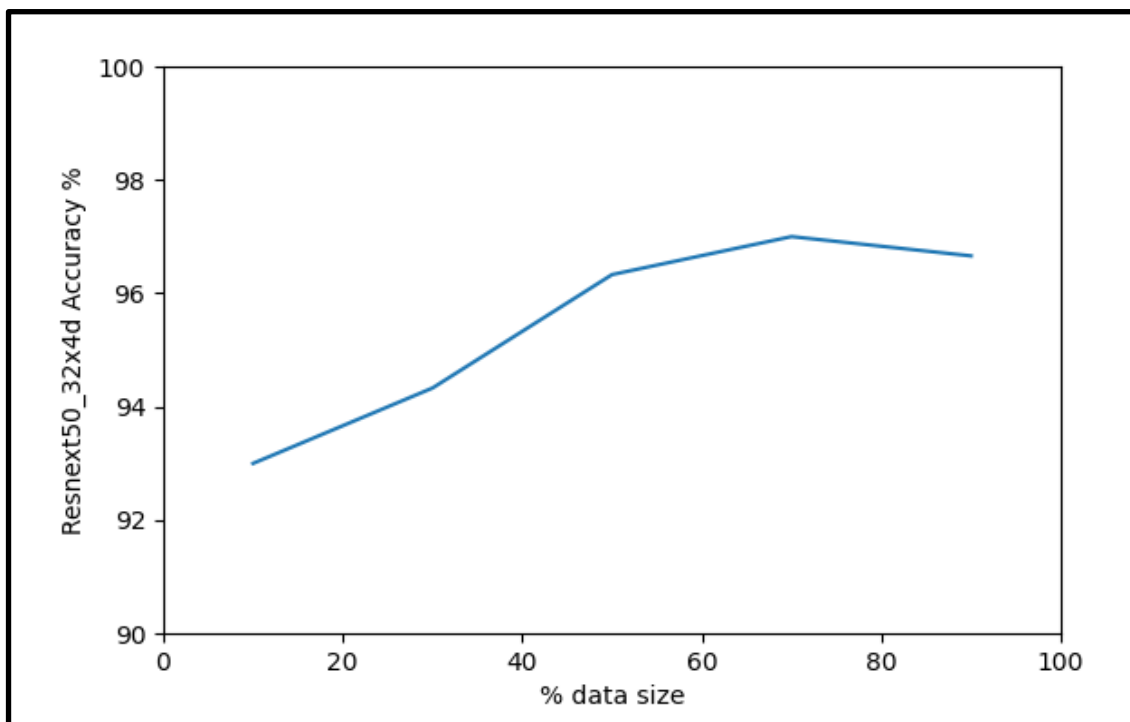


Figure 24 : Précision de Resnext50_32x4d en fonction de la taille de la base d'apprentissage

Dans le cas du réseau Resnext50_32x4d, une amélioration non négligeable est observée lorsque la taille de la base d'apprentissage est augmentée jusqu'à 70% de la taille de la base originelle.

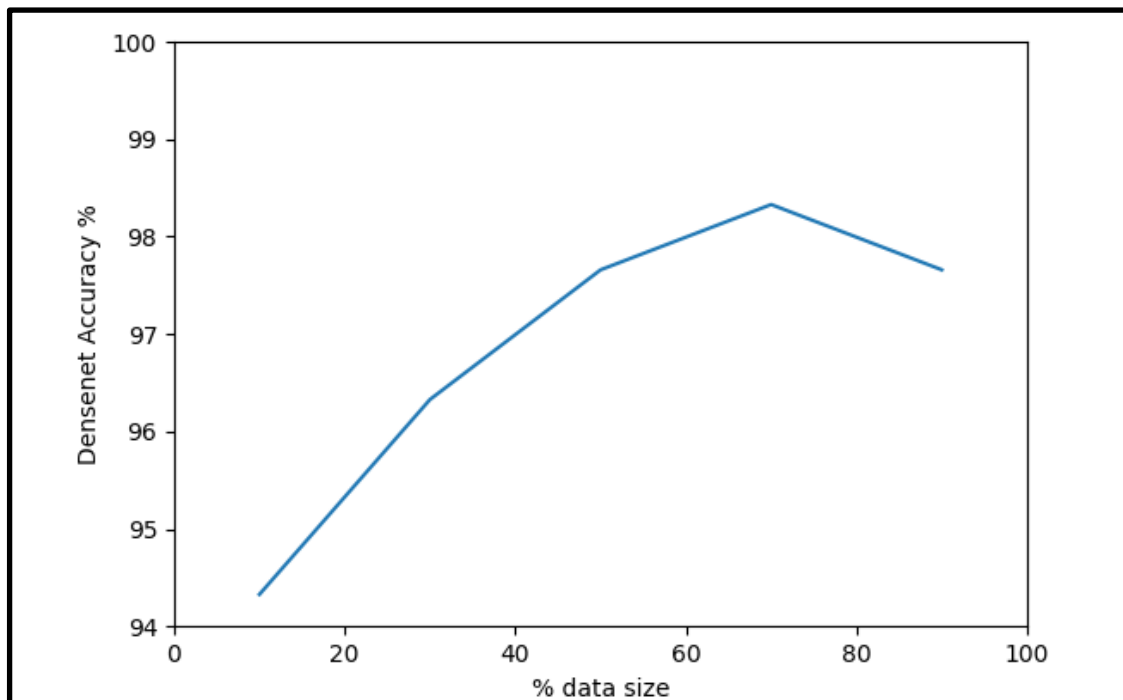


Figure 25 : Précision de Densenet en fonction de la taille de la base d'apprentissage

Le CNN Densenet augmente également en précision de manière significative lorsque la taille de la base d'apprentissage atteint les 70%.

Ces graphes montrent que la taille de la base d'apprentissage est suffisante pour cette application car un stade de plateau est atteint pour les réseaux de neurones aux alentours des 70%, voire 50% pour VGG-16.

Une optimisation de la précision de ces modèles serait ainsi possible si, pour chaque modèle entraîné, la taille de la base d'apprentissage variait en fonction des besoins de la structure du réseau.

7.3.4 - Détection des visages

Avec la montée en puissance du deep learning au cours de la dernière décennie, de multiples approches permettant de détecter un ou plusieurs visages au sein d'une image ont émergées.

Ainsi, pour cette partie du programme, il a été décidé de s'appuyer sur des travaux préexistants.

La première solution qui a été testée est la détection de visage en utilisant l'algorithme de détection d'objets nommé "Haar cascade classifier" [23]. C'est une méthode de détection d'objets qui s'appuie sur le machine learning, mise au point par Paul Viola et Michael Jones dans leur article, "Rapid Object Detection using a Boosted Cascade of Simple Features" en 2001.

Il s'agit d'une approche basée sur l'apprentissage automatique où une fonction en cascade est formée à partir d'un grand nombre d'images positives et négatives.

Cette façon d'opérer la détection de visage est très répandue et populaire de par sa simplicité. Elle est relativement facile à mettre en place.

Cependant, elle a ses limites. Tout d'abord elle a du mal à détecter un visage en mouvement, elle perd les coordonnées du visage de profil.

Elle peut également totalement se tromper dans la détection et détecter des faux positifs, comme par exemple confondre une oreille avec un visage :

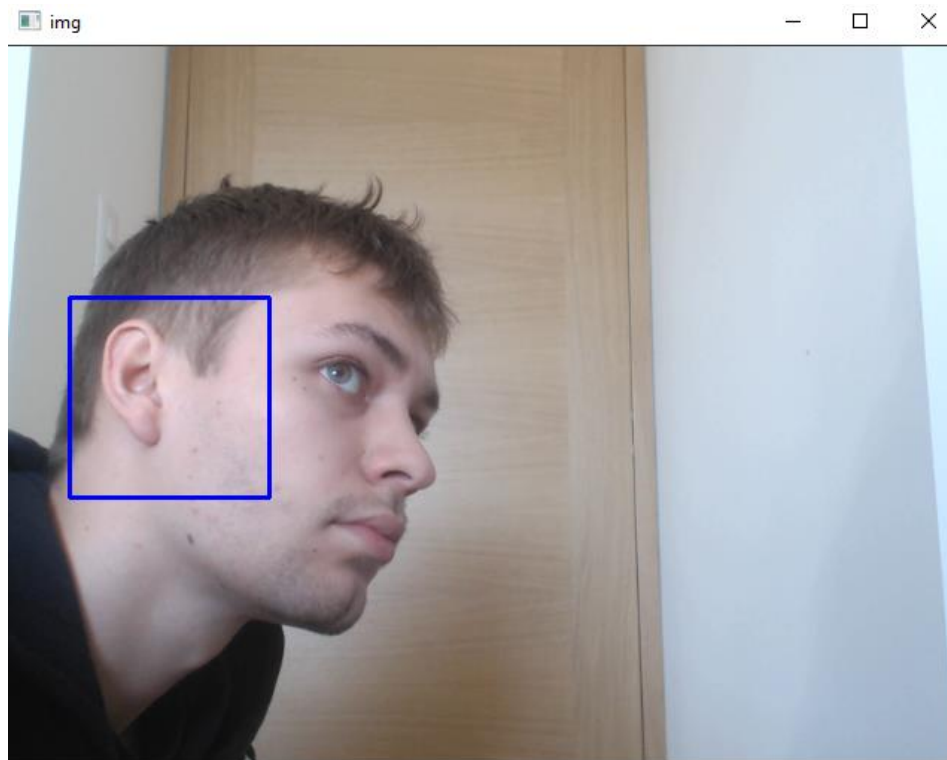


Figure 26 : Faux positif d'une détection de visage avec Haar Cascade Classifier

Il y a néanmoins un point fort à retenir de cette technologie dans le cas de l'application : elle permet de détecter un visage même si ce dernier est partiellement obstrué, avec un masque par exemple.

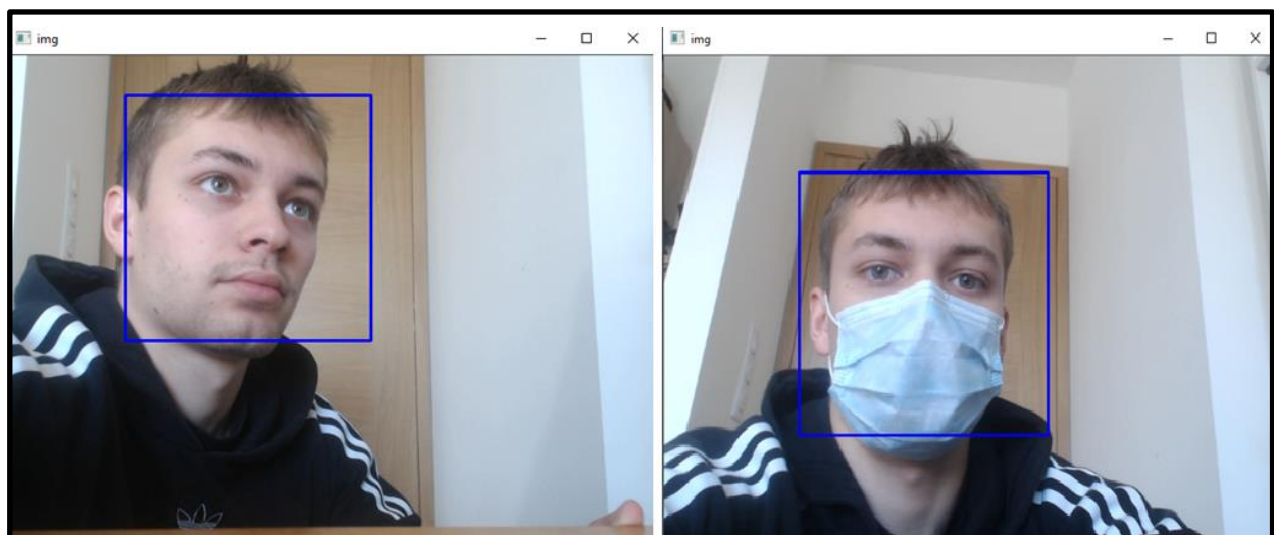


Figure 27 : Détection d'un visage masqué et sans masque avec Haar Cascade Classifier

Par la suite, d'autres technologies de détection de visage ont été testées, la plupart ne donnaient pas de résultats satisfaisants : soit elles demandaient trop de calculs et rendaient l'application en temps réelle impossible, soit elles ne permettaient pas de détecter un visage portant un masque.

Une seule technologie se démarquait du lot : la lecture et l'emploi directement via OpenCV d'un CNN pré-entraîné à détecter les visages. [24]

L'architecture employée par ce CNN est ResNet-10 [25], et le format de sauvegarde du fichier utilisé est Caffé Model. [26]

Cette technologie comporte de nombreux avantages : elle est, à l'instar de Haar Cascade Classifier, assez simple à mettre en place, relativement peu gourmande en ressources, et elle assure une stabilité remarquable dans la détection des coordonnées des visages.

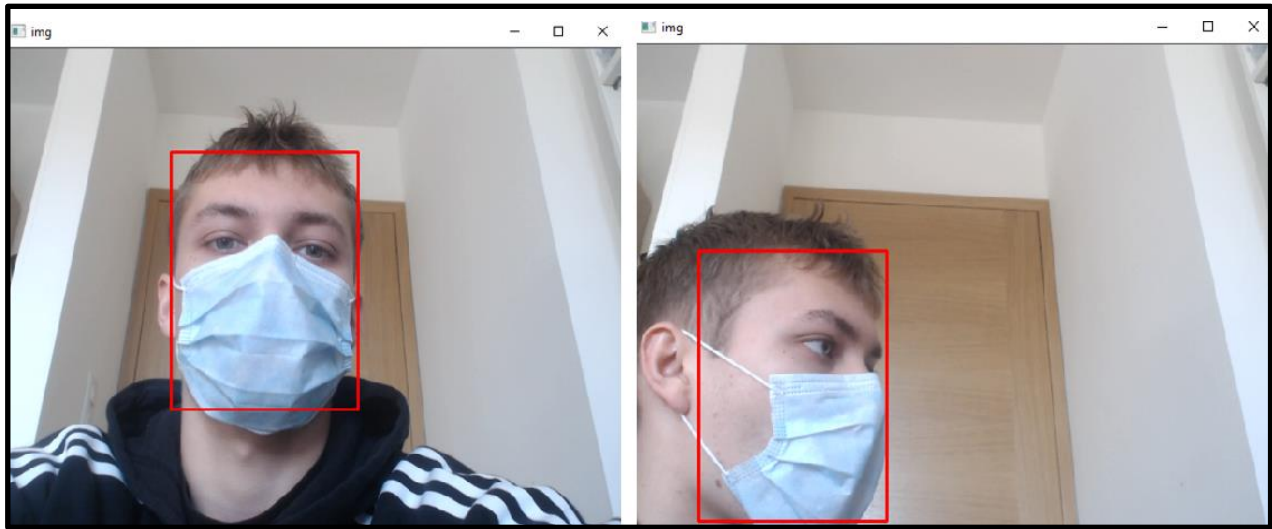


Figure 28 : Détection d'un visage masqué de face et de profil avec le CNN pré-entraîné

Ainsi, c'est cette solution qui a été retenue pour la suite du projet.

Afin de comparer les deux technologies présentées qui répondent au besoin de détection de visage même masqué, un script permettant de comparer leur efficacité en détection a été mis au point.

Ce dernier donne 900 images issues de notre dataset, qui ne contiennent qu'un seul individu par photo, à chacun des algorithmes et compte le nombre de visages détectés par chaque algorithme.

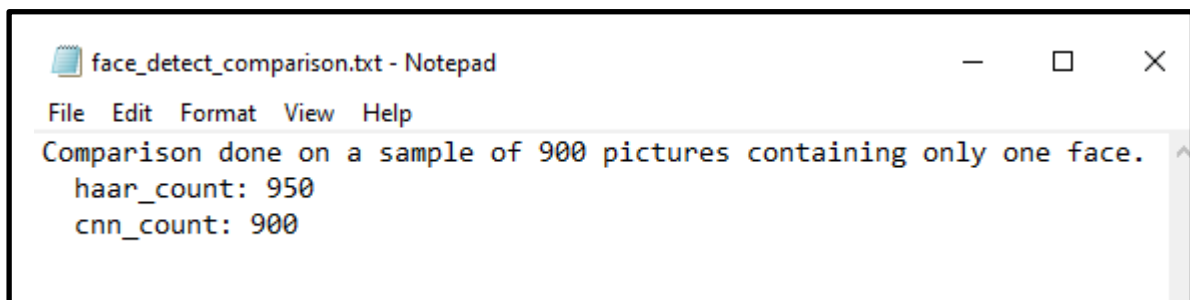


Figure 29 : Comparaison des différents algorithmes de détection du visage

Les résultats sont sans appel : l'algorithme Haar Cascade Classifier n'est pas très performant, il fait 50 faux-positifs dans la détection de visage, c'est à dire qu'il détecte plus de visages qu'il n'y en a réellement.

Pour plus de rigueur, un script permettant de construire la matrice de confusion pour le CNN ResNet-10 a été élaboré et présente les résultats suivants :


```

face_detect_confusion_matrix.txt - Notepad
File Edit Format View Help
Inference done on a sample of 900 pictures containing only one face.
TP: 900
TN: 0
FP: 0
FN: 0

```

Figure 30 : Résultat du script permettant de construire la matrice de confusion pour le CNN

La matrice de confusion est donc la suivante :

		Vraie Classe	
		Positif	Negatif
Classe prédite	Positif	TP = 900	FP = 0
	Negatif	FN = 0	TN = 0

Figure 31 : Matrice de confusion du CNN de détection de visage / dataset de 900 images

On en déduit ce tableau que le CNN fonctionne parfaitement pour la détection de visage car il obtient 100% de Vrais positifs (900/900) et 0% de Faux Positifs/Négatifs. Il ne pouvait pas détecter de Vrais Négatifs car le dataset ne comportait pas d'images sans visage.

7.3.5 - Classification des visages par les CNNs

Une fois les algorithmes de détection de visages et les modèles entraînés prêts à l'emploi, l'étape à venir est le passage des prédictions du mode offline vers le mode online.

Concrètement, cela revient à envoyer une image de taille 224*224 pixels des visages capturés à la webcam aux CNN sélectionnés. Évidemment, il faut au préalable appliquer les mêmes transformations qui ont été appliquées sur les images du dataset pour entraîner les modèles.

Pour connaître le détail de ces transformations, se référer à la section "Entraînement des CNNs".

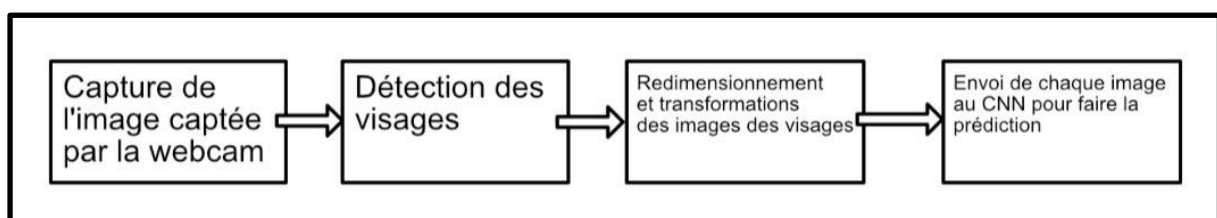


Figure 32 : Processus suivi par le programme

Les prédictions de classe sur les visages par les CNNs se sont montrés satisfaisants lors des essais online, c'est donc cette approche qui a été privilégiée et améliorée par la suite du projet.

Voici donc le rendu des tests online des CNNs :

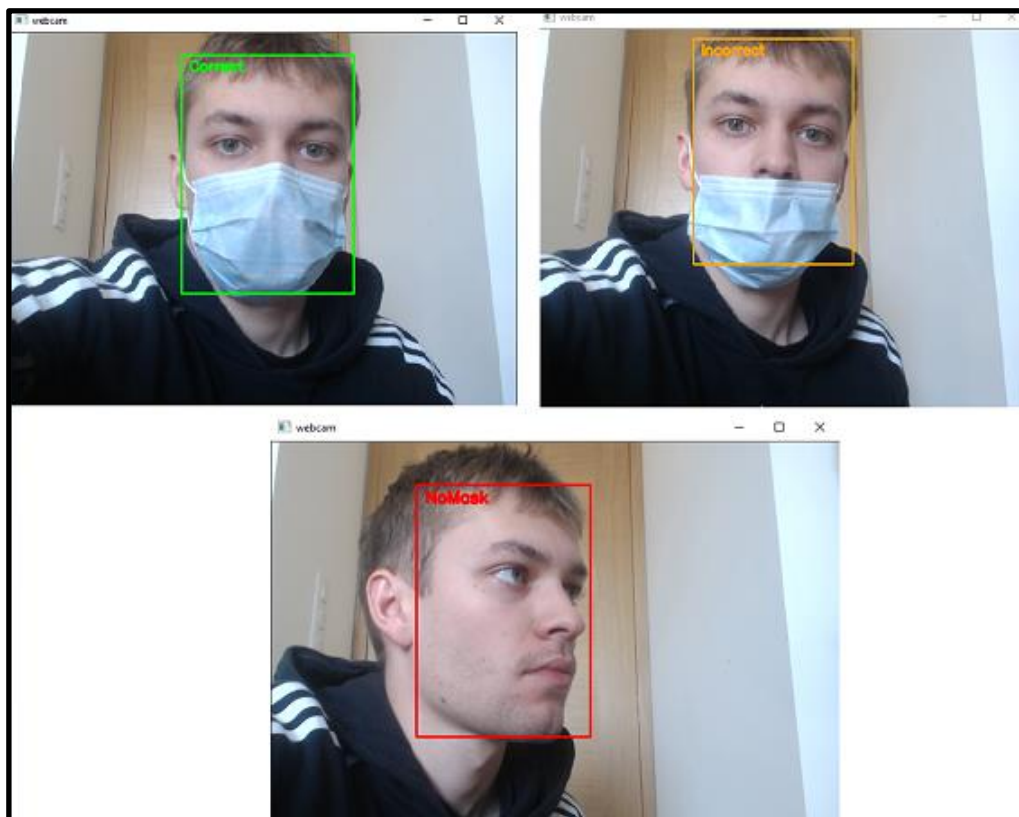


Figure 34 : Test online des CNN

Chaque CNN a été testé un par un dans le but de constater les résultats de la validation sur la partie online.

Bien que les résultats se montraient satisfaisants, il arrivait que même les CNNs comptant parmi les plus précis se trompent de manière passagère.

À l'origine, les prédictions du CNN sélectionné se faisaient à chaque nouvelle frame capturée par la webcam. Ce mécanisme étant trop gourmand en ressources et compromettait la fluidité de la capture vidéo, le flux vidéo se mettant en pause le temps des calculs, un mécanisme de temporisation a été mis en place.

Cela permet, suivant les ressources de la machine sur laquelle le programme est exécuté, d'effectuer ces calculs toutes les 15, 30 ou 60 nouvelles images.

Cette fonctionnalité a été mise en place de la manière suivante, un compteur d'image est incrémenté à chaque nouvelle image capturée, lorsque ce compteur atteint la valeur de fréquence de rafraîchissement souhaitée, ce compteur est remis à 0 afin d'éviter les problèmes de mémoire.

La condition pour que les CNNs fassent leurs prédictions est que ce compteur soit égal à 0.

Le gain en fluidité est non-négligeable et s'est instantanément fait ressentir.

De surcroît, lorsque plus d'un visage est détecté par la capture, il faut multiplier par ce nombre de visages le temps de calcul nécessaire à chaque prédiction.

Le calibrage se fait donc suivant les capacités de la machine considérée ainsi que suivant l'application pour laquelle le programme est utilisé (détection dans un endroit avec beaucoup ou peu de passage).

L'optimisation de la fonctionnalité de détection de plusieurs visages est apparue à la fin du processus de classification des CNNs.

L'implémentation de cette fonctionnalité a requis l'ajout d'une fonction qui compte les visages au préalable avant même de faire les prédictions. Si le nombre de visages détectés est différent de celui de la frame précédente, alors le compteur est remis à 0 grâce à un système de drapeau.

Cela permet de passer outre le temps éventuel à attendre avant la prochaine actualisation des prédictions : à chaque changement du nombre de visages détectés, les calculs des réseaux de neurones sont à nouveau effectués.

7.3.6 - Combinaisons des classifieurs

Pour améliorer la stabilité de ces prédictions, l'approche qui a été privilégiée est la combinaison des prédictions de plusieurs modèles.

Trois modèles de réseaux de neurones entraînés ont été retenus à la suite de tests online : VGG 16 [27], Resnext50_32x4d [28], densenet [29].

Ces derniers figurent parmi les 5 modèles les plus précis mais mobilenet_v3_large a été écarté car sa stabilité était en deçà des autres modèles lors des tests online.

Cela peut être expliqué de par le fait que ce CNN a été conçu pour être déployé sur des terminaux mobiles, qui disposent par conséquent d'une puissance de calcul inférieure à celle d'un ordinateur.

Par la suite, deux types de combinaison ont été testés :

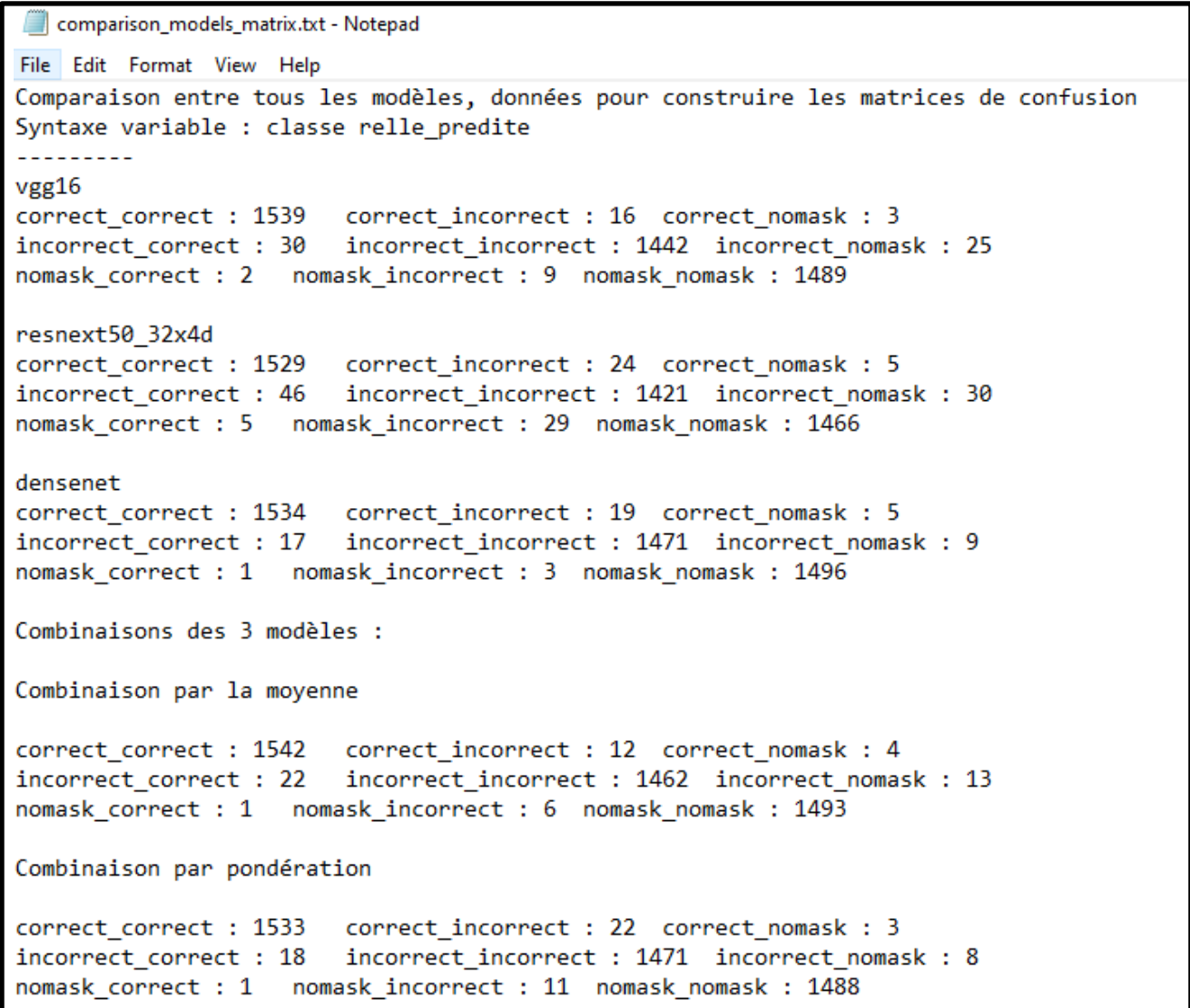
- Combinaison par pondération : un poids différent dans la décision finale est attribué à chaque modèle
- Combinaison par moyenne : chaque modèle dispose d'un poids égal aux autres dans la décision finale

Une combinaison par pondération semblait plus appropriée que les autres et a été retenue : si le CNN VGG-16 détecte un port incorrect du masque, alors c'est cette prédiction qui est retenue, peu importe celle des 2 autres CNNs.

Cette combinaison se démarquait dans les tests online car, suivant la façon dont le masque était incorrectement porté, seul ce réseau de neurone était en capacité de reconnaître le port incorrect du masque.

Afin de comparer l'efficacité de ces deux mesures destinées à améliorer les prédictions, un script qui compte les erreurs de prédictions de chaque méthode sur la base de données a été programmé. Le script compte également le nombre d'erreurs faite par chacun de ces 3 modèles de manière isolée.

Ce dernier, comme la majeure partie des scripts qui vise à comparer différentes technologies au sein du projet, écrit les résultats de la comparaison dans un fichier texte. Ce dernier permet par la suite de construire les matrices de confusion pour chaque méthode de prédiction.



```
comparaison_models_matrix.txt - Notepad
File Edit Format View Help
Comparaison entre tous les modèles, données pour construire les matrices de confusion
Syntaxe variable : classe relle_predite
-----
vgg16
correct_correct : 1539   correct_incorrect : 16   correct_nomask : 3
incorrect_correct : 30   incorrect_incorrect : 1442   incorrect_nomask : 25
nomask_correct : 2   nomask_incorrect : 9   nomask_nomask : 1489

resnext50_32x4d
correct_correct : 1529   correct_incorrect : 24   correct_nomask : 5
incorrect_correct : 46   incorrect_incorrect : 1421   incorrect_nomask : 30
nomask_correct : 5   nomask_incorrect : 29   nomask_nomask : 1466

densenet
correct_correct : 1534   correct_incorrect : 19   correct_nomask : 5
incorrect_correct : 17   incorrect_incorrect : 1471   incorrect_nomask : 9
nomask_correct : 1   nomask_incorrect : 3   nomask_nomask : 1496

Combinaisons des 3 modèles :

Combinaison par la moyenne

correct_correct : 1542   correct_incorrect : 12   correct_nomask : 4
incorrect_correct : 22   incorrect_incorrect : 1462   incorrect_nomask : 13
nomask_correct : 1   nomask_incorrect : 6   nomask_nomask : 1493

Combinaison par pondération

correct_correct : 1533   correct_incorrect : 22   correct_nomask : 3
incorrect_correct : 18   incorrect_incorrect : 1471   incorrect_nomask : 8
nomask_correct : 1   nomask_incorrect : 11   nomask_nomask : 1488
```

Figure 35 : Comparaison des modèles et des méthodes de combinaison / dataset de 4500 images

VGG16

Classe réelle

Classe prédite

	Correct	Incorrect	NoMask
Correct	1539	30	2
Incorrect	16	1442	9
NoMask	3	25	1489

resnext50_32x4d

Classe réelle

Classe prédite

	Correct	Incorrect	NoMask
Correct	1529	46	5
Incorrect	24	1421	29
NoMask	5	30	1466

densenet

Classe réelle

Classe prédite

	Correct	Incorrect	NoMask
Correct	1534	17	1
Incorrect	19	1471	3
NoMask	5	9	1496

Figure 36 : Matrice de confusion des CNNs VGG-16, resnext50_32x4d et densenet

Combinaison par moyenne

Classe réelle

Classe prédite

	Correct	Incorrect	NoMask
Correct	1542	22	1
Incorrect	12	1462	6
NoMask	4	13	1493

Combinaison par pondération

Classe réelle

Classe prédite

	Correct	Incorrect	NoMask
Correct	1533	18	1
Incorrect	22	1471	11
NoMask	3	8	1488

Figure 37 : Matrices de confusion de la combinaison par moyenne et par pondération des 3 CNNs

Cette comparaison met en relief une nette diminution du taux d'erreur par l'emploi des deux algorithmes de combinaison par rapport aux modèles seuls.

Les résultats des deux types de combinaison sont similaires avec un léger avantage pour la combinaison par moyenne.

Les erreurs qui reviennent le plus souvent sont des confusions entre les classes du masque correctement porté et mal porté.



Figure 38 : Exemples de mauvaise prédiction par la combinaison par pondération

Ces exemples d'erreurs montrent que les prédictions ont plus de chances d'être fausses lorsque le masque porté est atypique, d'une couleur proche de celle de la peau de l'individu, ou bien que ce dernier est trop obstrué par une main.

De plus, ces images sont de mauvaise qualité (d'une résolution inférieure à 224*224 avant le pre-processing) et représentent des conditions difficiles pour détecter le port du masque.

Néanmoins, sur les tests online, la combinaison par pondération s'est montrée un peu plus précise, c'est donc celle qui a été retenue.

7.3.7 - Interface graphique

Pour réaliser notre interface graphique en PyQt, il faut dans un premier temps disposer les composants dans une fenêtre principale. Afin d'être le plus proche possible des mockups réalisés, il a été décidé d'utiliser l'application Qt Designer [30], qui est un outil permettant de construire rapidement des interfaces graphiques avec des widgets issus du framework Qt GUI.

Cet outil a été particulièrement utile car il permet un gain de temps sur la partie disposition d'éléments. On visualise directement quel composant se place où dans l'interface, sans y passer des heures de code.

Cela permet de passer rapidement cette étape front-end qui peut rapidement se voir être fastidieuse et longue, à l'étape back-end qui n'est pas possible sur l'application.

En effet, relier les boutons à leur action ou coder la barre des menus n'est possible que sur un fichier Python.

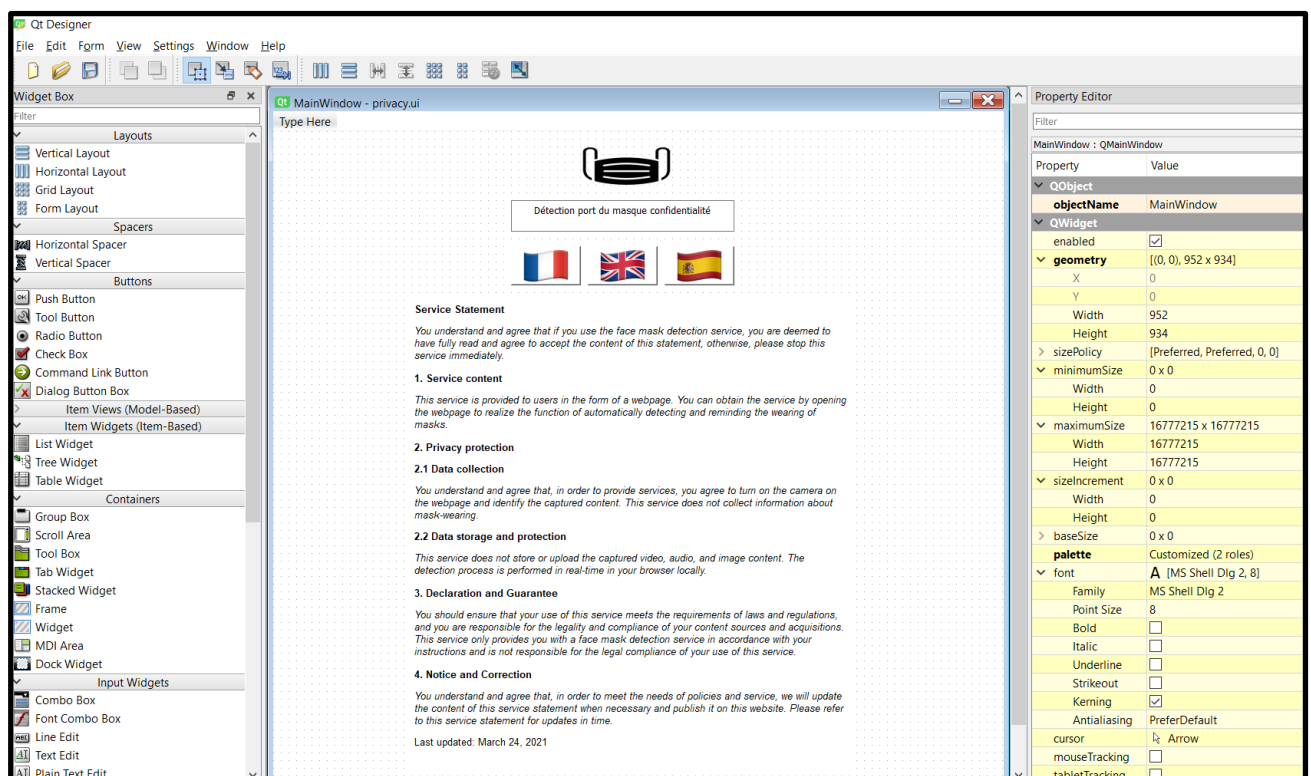


Figure 39 : Création de la page « Confidentialité » avec Qt Designer

Après avoir converti le fichier Qt Designer (dont l'extension est un .ui) en fichier de code Python (dont l'extension est un .py) à l'aide de l'invite de commande Windows, quelques légers problèmes ont alors été détectés.

Tout d'abord, le code créé par Qt Designer n'était pas optimisé et ne permettait pas de pouvoir avoir des passerelles entre plusieurs interfaces. Ainsi, lorsque l'utilisateur voulait changer le thème de l'interface principale, il n'était pas possible de pouvoir faire en sorte que ce thème s'actualise également sur les pages secondaires « Confidentialité », « Instructions » etc.

Ainsi, il a fallu modifier à la main, en redéfinissant la fenêtre principale afin qu'elle puisse actualiser certains paramètres sur les autres pages. Le code de l'interface a été réalisé entièrement avec PyQt5.

De plus, comme mentionné dans la partie « Organisation de la partie IHM », une interface redimensionnable pour différentes résolutions d'écran a été créée. Cette IHM a été codée en PyQt, sans passer par Qt Designer, à l'aide d'un système de grille que PyQt 5 nous permet d'utiliser, cela facilite le redimensionnement sans créer un algorithme à la main qui adapte la taille de chaque bloc.

L'interface est au final moins esthétique car la disposition des éléments est plus compliquée sans utiliser Qt Designer. L'IHM a été conçue pour pouvoir s'adapter à n'importe quelle résolution d'écran et permettre à n'importe qui de bénéficier du détecteur du port du masque.

De plus, la partie « Conseil » pour l'utilisateur a été enlevée pour cette IHM afin d'alléger le code. Sachant que la prédiction est disponible avec le détecteur, l'utilisateur sait déjà à partir de cette prédiction et en lisant la partie « Instructions », quoi faire afin de protéger au mieux les autres du virus. (annexe n°7)

7.3.8 - Conversion en exécutable

Pour que l'interface, et donc le détecteur de port du masque, soit utilisable par un utilisateur lambda qui ne possède aucune extension ni installation Python, il est important de transformer l'interface en exécutable.

Pour cela, il suffit d'installer PyInstaller sur l'invite de commande Windows en tapant : `pip install pyinstaller`. [31] Une fois l'installation terminée, il ne reste plus qu'à transformer l'interface principal (extension .py) en fichier exécutable (extension .exe) à l'aide de PyInstaller.

La version exécutable du projet est maintenant créée et placée dans un nouvel emplacement. Afin que celle-ci puisse fonctionner, le transfert des fichiers ayant servi à la conception du projet (les images, les modèles, etc) à l'emplacement de l'exécutable est obligatoire. Il faut respecter l'arborescence établie au préalable.

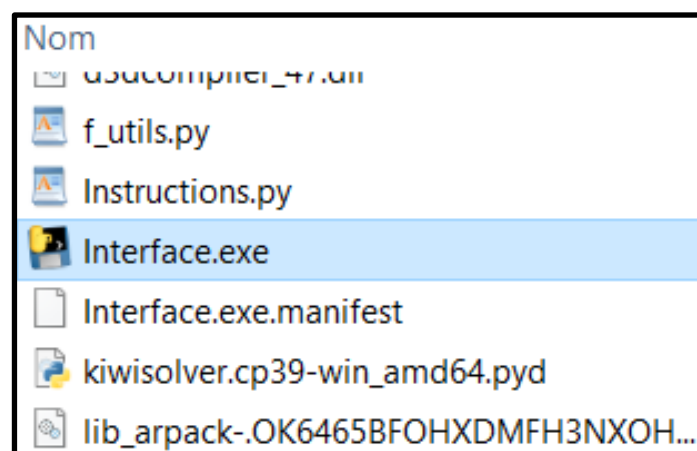


Figure 40 : Interface exécutable parmi les fichiers nécessaires à son exécution

L'utilisateur n'a plus qu'à cliquer sur « Interface.exe » et le tour est joué, l'interface se lancera sur son ordinateur.

8 – Conclusion

Le monde entier vit actuellement une crise sanitaire inédite, liée à l'épidémie du COVID-19. En attendant que la vaccination soit généralisée, pour tempérer l'épidémie, plusieurs dispositifs sont mis en œuvre : confinements localisés, couvre-feu ou encore port du masque rendu obligatoire.

D'après les études statistiques faites par de nombreux pays sur la propagation du coronavirus, il en ressort que la transmission est plus fréquente dans des endroits dans lesquels le public se rencontre et où il y a de l'interaction sociale.

Ainsi, il est difficile de contrôler que chaque personne respecte les dispositifs permettant d'endiguer cette épidémie car il reste impossible d'analyser un par un les comportements de chacun.

C'est pourquoi, ce projet répond à un besoin de démocratiser les bonnes pratiques concernant le port du masque car la propagation par l'air expiré est le vecteur principal de transmission du virus, il répond aussi à un besoin de distinguer le port correct et incorrect du masque, jusqu'alors confondus par la majorité des applications créées. Il permet ainsi d'améliorer la traçabilité des contaminations dans l'espace public.

Le projet propose une approche par le Transfer Learning pour résoudre ce problème. Il permet de détecter, sur une ou plusieurs personnes, si le masque est bien, mal ou pas porté.

Ce système est simple d'utilisation grâce à une interface graphique. Il nécessite, pour l'utilisateur, uniquement l'utilisation d'une webcam directement intégrée à l'ordinateur, ou externe.

Un "manuel d'utilisation" ainsi que la politique de confidentialité et un résumé du projet sont également disponibles directement sur l'interface afin de rendre l'expérience utilisateur la plus agréable et complète possible.

Plusieurs axes d'améliorations ont été relevés.

Premièrement, bien que le système fonctionne avec la plupart des masques, il n'a pas été autant entraîné sur les masques "atypiques" que pour reconnaître les masques chirurgicaux. Pour parfaire sa précision, il conviendrait d'agrémenter le dataset d'images correspondantes.

En second lieu, les algorithmes de combinaisons des prédictions qui sont employés restent relativement basiques, il est probable que des algorithmes plus poussés comme celui de l'intégrale floue fassent gagner l'application en précision. [32]

9 – Bibliographie/Webographie

[1] : <https://www.who.int/news-room/q-a-detail/coronavirus-disease-covid-19-how-is-it-transmitted> - dernière consultation le 25/04/2021

[2] : Zekun Wang, Pengwei Wang, Peter C. Louis, Lee E. Wheless, and Yuankai Huo, Member, IEEE, WearMask: Fast In-browser Face Mask Detection with Serverless Edge Computing for COVID-19, publié le 04/01/2021, dernière consultation le 25/04/2021

[3] : G. Jignesh Chowdary, Narinder Singh Punn, Sanjay Kumar Sonbhadra, and Sonali Agarwal, Face Mask Detection using Transfer Learning of InceptionV3, publié le 20/09/2020, dernière consultation le 25/04/2021

[4] : Jeremy Howard, Austin Huang, Zhiyuan Li, Zeynep Tufekci, Vladimir Zdimal, Helene-Mari van der Westhuizen, An evidence review of face masks against COVID-19, publié le 26/01/2021, dernière consultation le 25/04/2021

[5] : James Vincent, France is using AI to check whether people are wearing masks on public transport, publié le 07/05/2020, dernière consultation le 25/04/2021

[6] : <http://www.image-net.org/> - dernière consultation le 25/04/2021

[7] : <https://cocodataset.org/#home> - dernière consultation le 25/04/2021

[8] : Margaret Mitchell and Timnit Gebru, Google fires second AI ethics researcher following internal investigation, publié le 19/02/2021, dernière consultation le 25/04/2021

[9] : <https://github.com/cabani/MaskedFace-Net> - dernière consultation le 25/04/2021

[10] : <https://www.kaggle.com/andrewmvd/face-mask-detection> - dernière consultation le 25/04/2021

[11] : Shuo Yang, Ping Luo, Chen Change Loy, Senior Member, IEEE and Xiaoou Tang, Fellow, IEEE, Faceness-Net: Face Detection through Deep Facial Part Responses, publié le 25/08/2017, dernière consultation le 25/04/2021

[12] : <https://opencv.org/> - dernière consultation le 25/04/2021

[13] : <https://www.python.org/> - dernière consultation le 25/04/2021

[14] : Claire D. Costa, Top Programming Languages for AI Engineers in 2021, publié le 17/03/2020, dernière consultation le 25/04/2021

[15] : <https://www.microsoft.com/en-us/download/details.aspx?id=48145> - dernière consultation le 25/04/2021

[16] : <https://pytorch.org/get-started/locally/> - dernière consultation le 25/04/2021

[17] : <https://pypi.org/project/PyQt5/> - dernière consultation le 25/04/2021

[18] : <https://github.com/cabani/MaskedFace-Net> - dernière consultation le 25/04/2021

[19] : Adnane Cabani, Karim Hammoudi, Halim Benhabiles, Mahmoud Melkemi, MASKEDFACE-NET - A DATASET OF CORRECTLY/INCORRECTLY MASKED FACE IMAGES IN THE CONTEXT OF COVID-19, 19/08/2020, dernière consultation le 25/04/2021

[20] : <https://www.kaggle.com/andrewmvd/face-mask-detection> - dernière publication le 25/04/2021

[21] : <https://pytorch.org/vision/stable/models.html> - dernière consultation le 25/04/2021

[22] : <https://gist.github.com/yrevar/942d3a0ac09ec9e5eb3a> - dernière consultation le 25/04/2021

[23] : https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html - dernière consultation le 25/04/2021

[24] : https://github.com/gopinath-balu/computer_vision/blob/master/CAFFE_DNN/res10_300x300_ssd_iter_140000.caffemodel - dernière consultation le 25/04/2021

[25] : Shuo Yang, Ping Luo, Chen Change Loy, Senior Member, IEEE and Xiaoou Tang, Fellow, IEEE, Faceness-Net: Face Detection through Deep Facial Part Responses, publié le 25/08/2017, dernière consultation le 25/04/2021

[26] : <https://caffe.berkeleyvision.org/> - dernière consultation le 25/04/2021

[27] : Karen Simonyan & Andrew Zisserman, VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION, publié le 10/04/2015, dernière consultation le 25/04/2021

[28] : Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun, Deep Residual Learning for Image Recognition, publié le 10/12/2015, dernière consultation le 25/04/2021

[29] : Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger, Densely Connected Convolutional Networks, publié le 28/01/2001, dernière consultation le 25/04/2021

[30] : <https://doc.qt.io/qt-5/qt designer-manual.html> - dernière consultation le 25/04/2021

[31] : <https://www.pyinstaller.org/> - dernière consultation le 25/04/2021

[32] : Belahcene Mebarka & Abdelmalik Ouamane, l'Intégrale Floue dans la Fusion d'un Système Multiclassifieurs pour la Reconnaissance de Visages, publié en 01/2013, dernière consultation le 25/04/2021

10 – Annexes

Annexe n°1 :

PROJET : Détection de port du masque

Réunion 5
Date : 24/03/2021

Présents :

- HADJADJI Bilal
- ROCHARD Augustin
- FOUQUEAU Maël

Objet : Présentation de notre programme de détection, effectuer nouvelle BDD, présenter notre interface commencée

I) Présentation de notre programme de détection

Nous devons donc nous documenter sur les réseaux convolutifs de neurones et les logiciels compatibles afin de créer notre détecteur. Nous avons donc voulu créer nous-mêmes notre CNN entier de A à Z en développant nous-mêmes toutes les couches, mais ce fut un échec car nous n'obtenions jamais mieux que 33% de bonnes détections après entraînement (très long) de nos modèles. Après cet échec, nous avons convenu d'utiliser des modèles pré-entraînés de Pytorch et de les entraîner à détecter les masques en changeant uniquement la dernière couche de ces réseaux.

Ainsi, nous avons décidé de créer un script comparant les résultats de comparaison entre les différents modèles de classification de Torchvision et nous avons obtenu une précision max de 98% ce qui est un excellent résultat. Nous avons donc présenté la détection des masques sur nous avec la webcam mais elle n'est pas très stable car nous observons une bonne détection des visages sans masque mais pas une bonne détection des masques mal portés et bien portés en utilisant le classifieur ayant la meilleure précision.

M.HADJADJI nous a donc conseillé de n'envoyer que 1 à 2 images maximum par seconde au système pour régler la stabilité.

II) Créer nouvelle BDD

Nous devons augmenter notre dataset à 1500 images par classes plutôt que 500 pour augmenter la qualité de détection. Il est également important d'inclure des personnes portant des masques de couleurs différentes et pas uniquement des masques chirurgicaux.

III) Présenter notre interface commencée

Nous avons décidé de commencer notre interface en PyQt et donc de se diviser le travail pour prendre un peu d'avance sur notre programme défini quelques jours avant. Nous avons réussi à créer une interface très ressemblante aux mockups. Seul problème, nous n'arrivons pas à créer un bouton permettant à l'utilisateur de changer et choisir sa webcam. A creuser. Sinon, interface validée par M.HADJADJI qui est satisfait.

IV) Objectifs pour la prochaine réunion

- ➔ Suivre les consignes données pour améliorer notre détecteur
- ➔ Créer nouvelle BDD
- ➔ Terminer l'interface

Date de la prochaine réunion : Vendredi 26 mars 2021

Figure 41 : Exemple de compte rendu de réunion

Annexe n°2 :

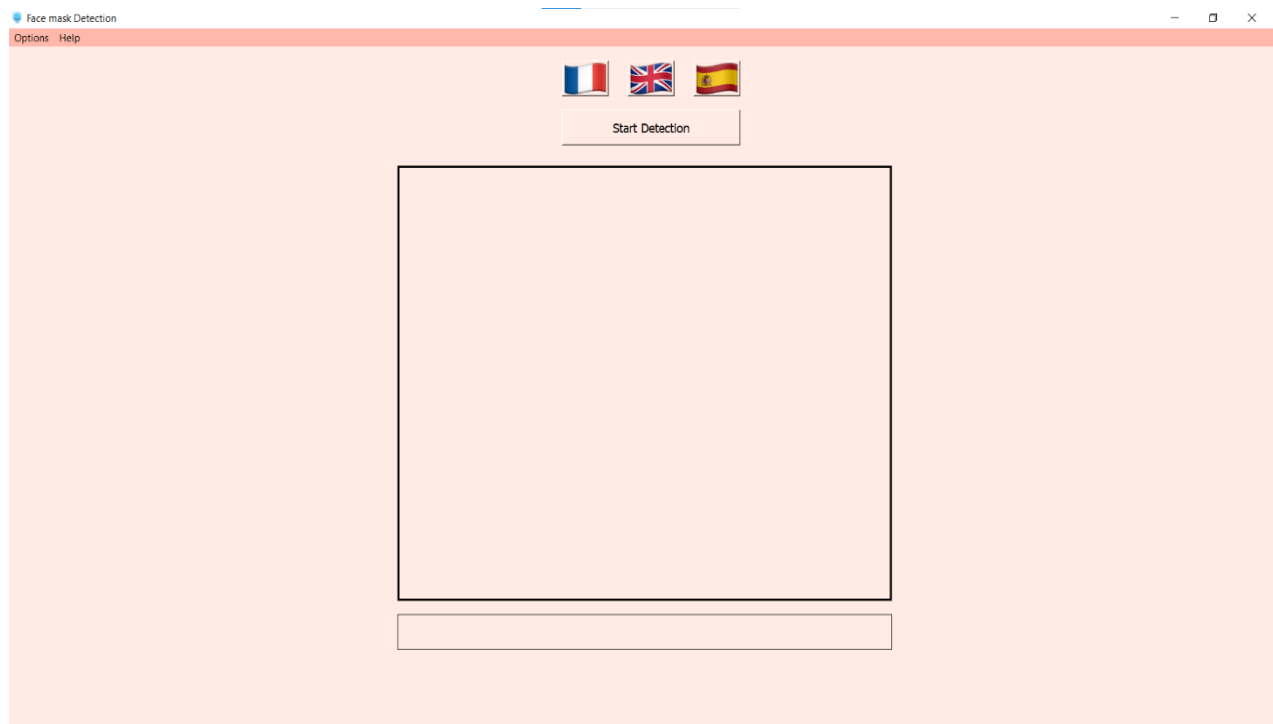


Figure 42 : Interface finale en mode jour

Annexe n°3 :

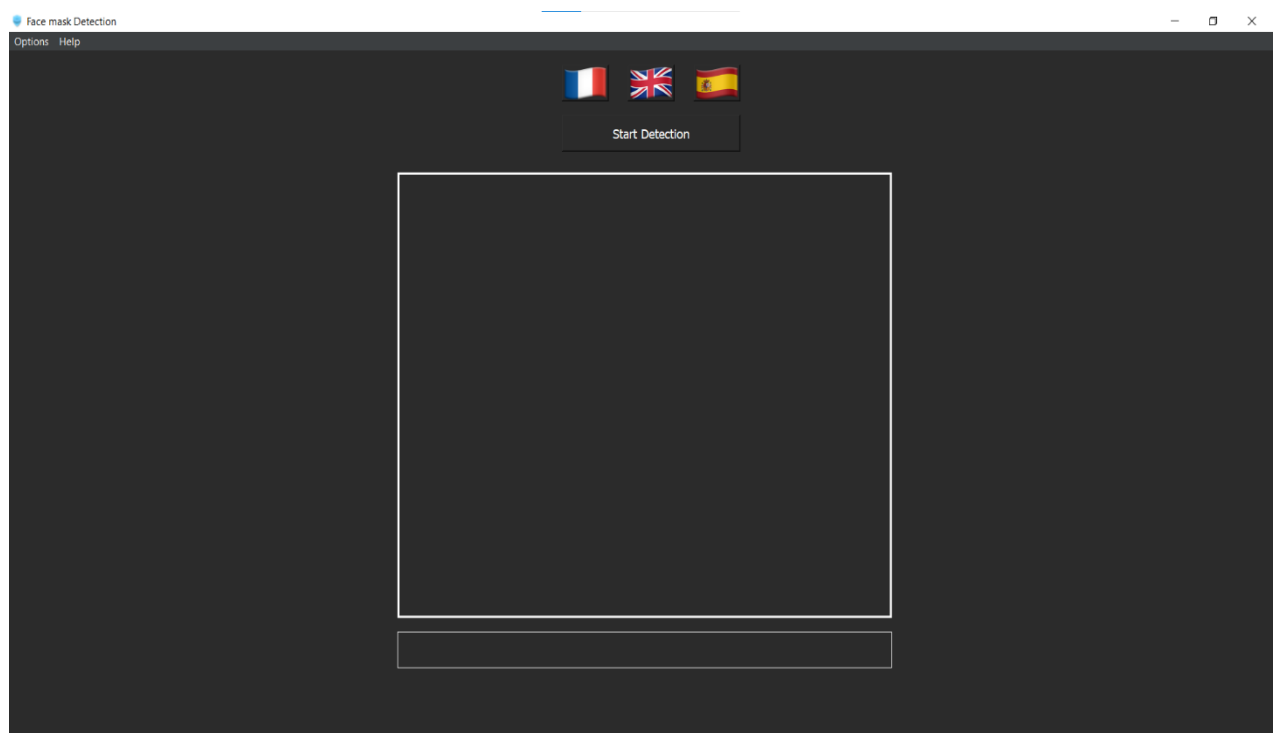


Figure 43 : Interface finale en mode sombre

Annexe n°4 :

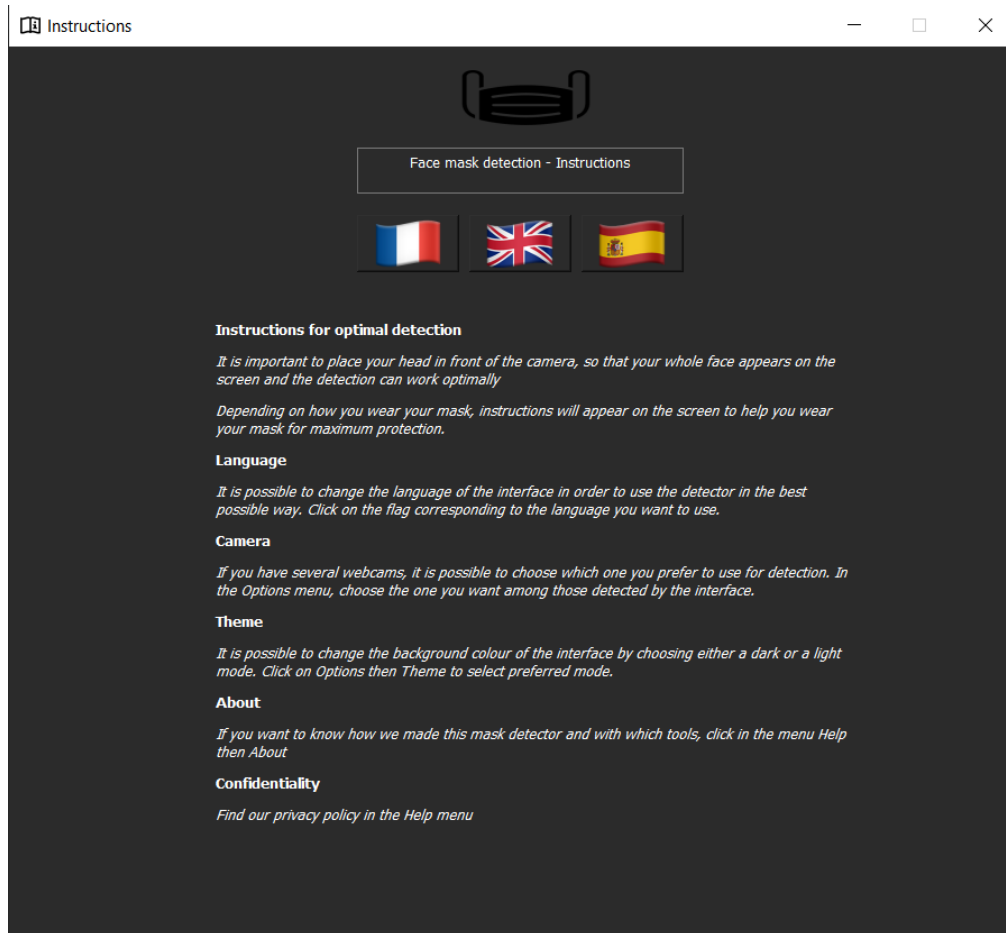


Figure 44 : Partie « Instructions » de l'interface finale, en mode sombre

Annexe n°5 :

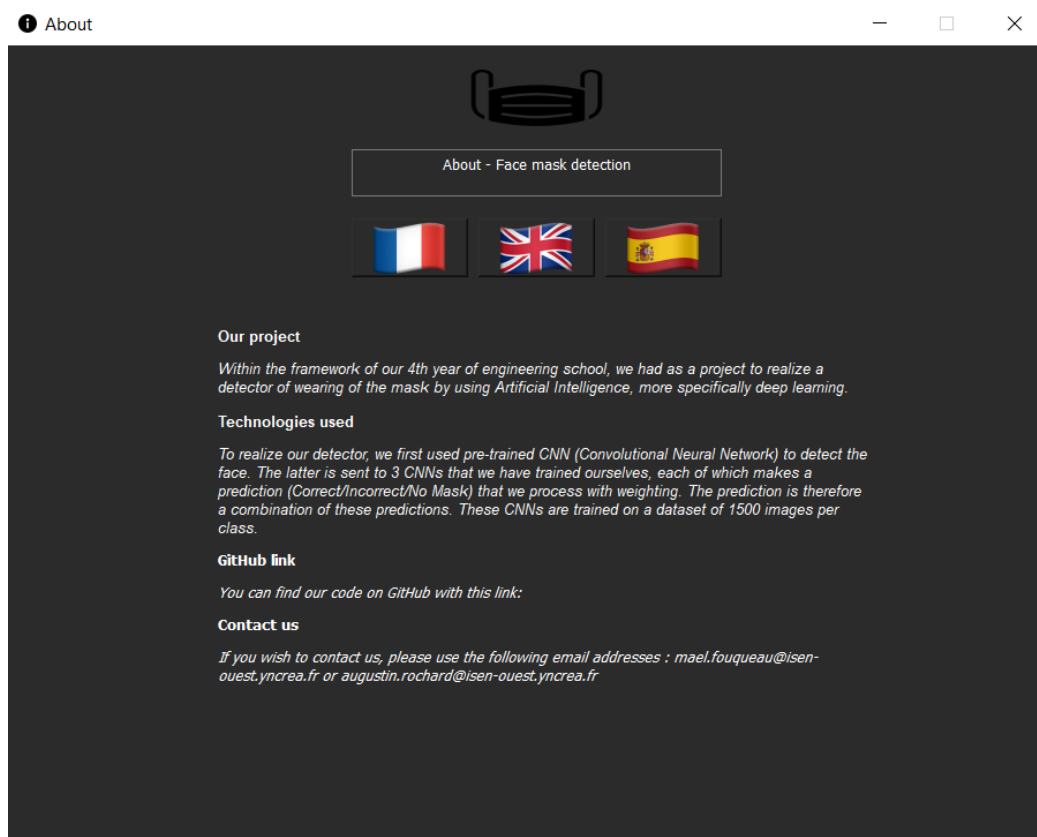


Figure 45 : Partie « A propos » de l'interface finale, en mode sombre

Annexe n°6 :

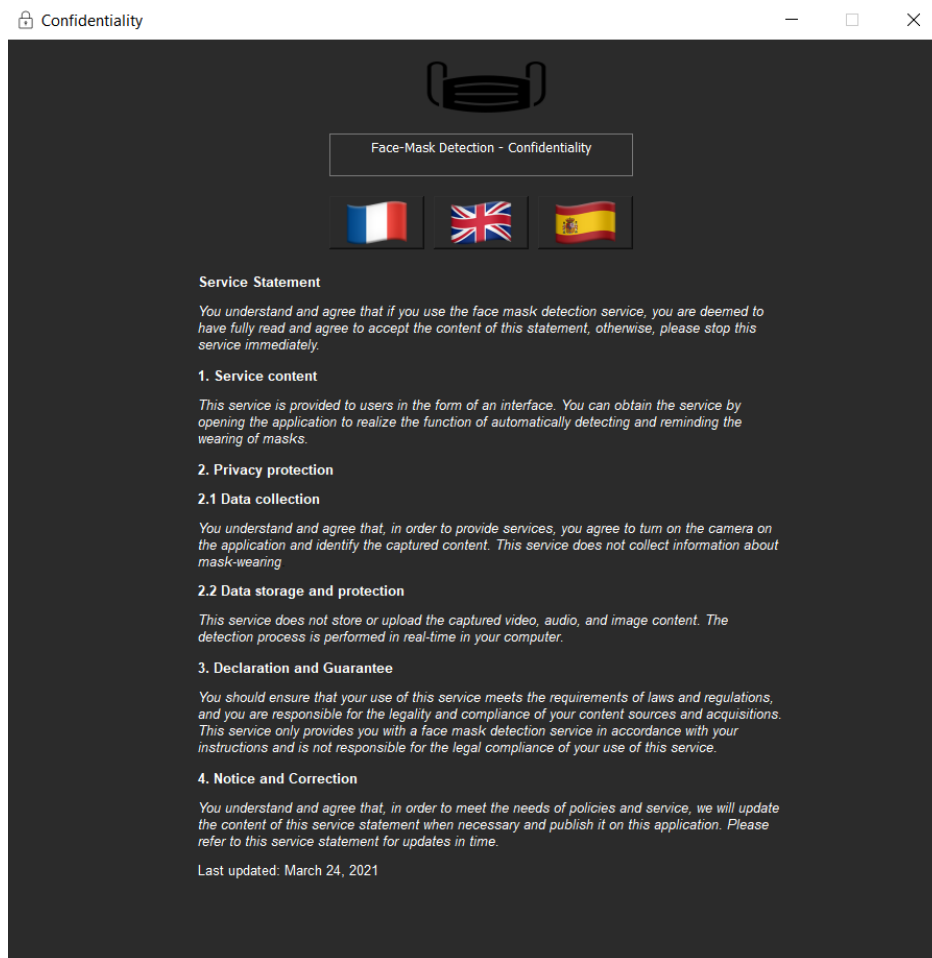


Figure 46 : Partie « Confidentialité » de l'interface finale, en mode sombre

Annexe n°7 :

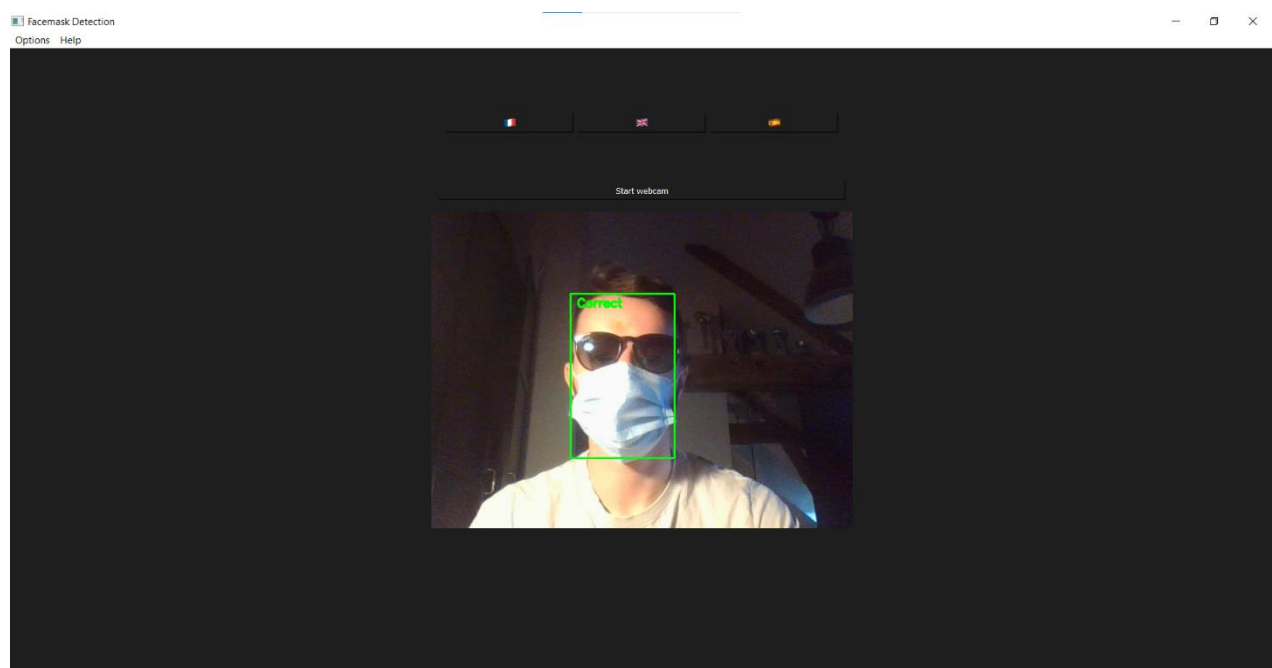


Figure 47 : Partie « Confidentialité » de l'interface redimensionnable, en mode sombre