Ecole nationale de la statistique
et de l'analyse de l'information

Université Mohammed V de Rabat

Ecole Supérieure de Technologie de Salé

RAPPORT DE STAGE

Sujet du stage : Détection de masques

Etudiant 1A

Etudiant :
Walid SEFIANI

Encadrant :
Mr le professeur
Adil AMMAR

2019 - 2020

Remerciements :
Je tiens à remercier mon encadrant M. Adil AMMAR, ainsi que toutes le bureau de stages pour toute l'aide apportée.

Table des matières

Chapitre 1

Note de choix de stage :

Ce stage de première année étant un stage opérateur et dans une période assez exceptionnelle, la recherche de stage a d'abord commencé par une recherche dans le carrer center de l'Ensai, mais sans résultat puisque cette plateforme est surtout réservée aux étudiants de 2 et 3e année, les stages proposés durent en effet 3 à 6mois. En plus du carrer center, j'ai cherché dans différents sites, déposé cv et lettre de motivation pour des jobs étudiants pendant la période du confinement mais c'était difficile de trouver une entreprise à proximité. C'est alors une fois que l'Ensai nous a permis de faire un stage en télétravail, je me suis focalisé sur ce type de stage et j'ai alors agrandi mon champ de recherche partout dans le monde en me focalisant sur le domaine de la statistique et de la data science. Toujours en envoyant cv et lettre de motivation et parfois recours à des appels téléphoniques, j'ai réussi à accéder à une offre grâce à un contact familial et accepté grâce à un entretien téléphonique.

Ce qui a motivé mon choix pour ce stage c'est d'abord un stage en télétravail, question de sécurité, ensuite le sujet du stage qui m'a vraiment intéressé puisque c'est une solution innovante qui permet notamment de réduire le non-respect des gestes barrières et ce en utilisant ce que j'ai appris à l'Ensai en programmation mais bien sûr en utilisant une nouvelle bibliothèque.

Ce que j'attends de cette expérience c'est une maîtrise de cette bibliothèque de détection d'objets (opencv), des compétences en termes de présentation de travail et de vente de produit.

Chapitre 2

Observation et description du travail :

2.1 Introduction et présentation du projet

La crise du covid-19 ayant fait des dégâts assez déplorables dans le monde entier, nous avons trouvé intéressant de faire un projet qui consiste à faire face à cette pandémie par de l'intelligence artificielle et plus précisément en utilisant les réseaux de neurones. Ces réseaux de neurones sont des cellules se trouvant dans notre cerveaux et qui sont connectées les unes aux autres liées par des axones, par lesquelles elles peuvent envoyer ou recevoir des signaux aux autres neurones. Les réseaux de neurones artificiels étant loin d'être aussi efficaces que les cellules humaines, ces neurones sont modélisés par des fonctions mathématiques et des poids propres à chaque entrés afin de prédire une sortie différente une fois qu'on change le poids. Plus on fait des opérations de prédictions et plus il y'a d' « apprentissage », c'est-à-dire plus les liaisons entre les réseaux de neurones sont renforcés et c'est ce qu'on appelle l'efficacité synaptique.
Ce projet consiste à créer un détecteur de masques facials dans une image ou en direct en utilisant la webcam intégrée à l'ordinateur, ou en se connectant à une caméra extérieure, et ce en utilisant la bibliothèque Open cv qui est une bibliothèque graphique spécialisée dans le traitement d'image, et l'outil open source d'apprentissage automatique « Tensorflow ». L'explication de ces deux outils a été développée dans la section ressource et documentation automatique.

2.2 Architecture de l'application et explication du code

Le programme se compose de trois scripts python nommés respectivement train_mask_detector, detect_mask_image et enfin detect_mask_video, le premier script permet de construire notre modèle de détecteur de masques mask_detector.model qu'on va utiliser dans les deux autres scripts, le deuxième effectue la détection du masque facial dans les images statiques alors que le troisième script applique la détection du masque facial à chaque image du flux émis par la webcam.

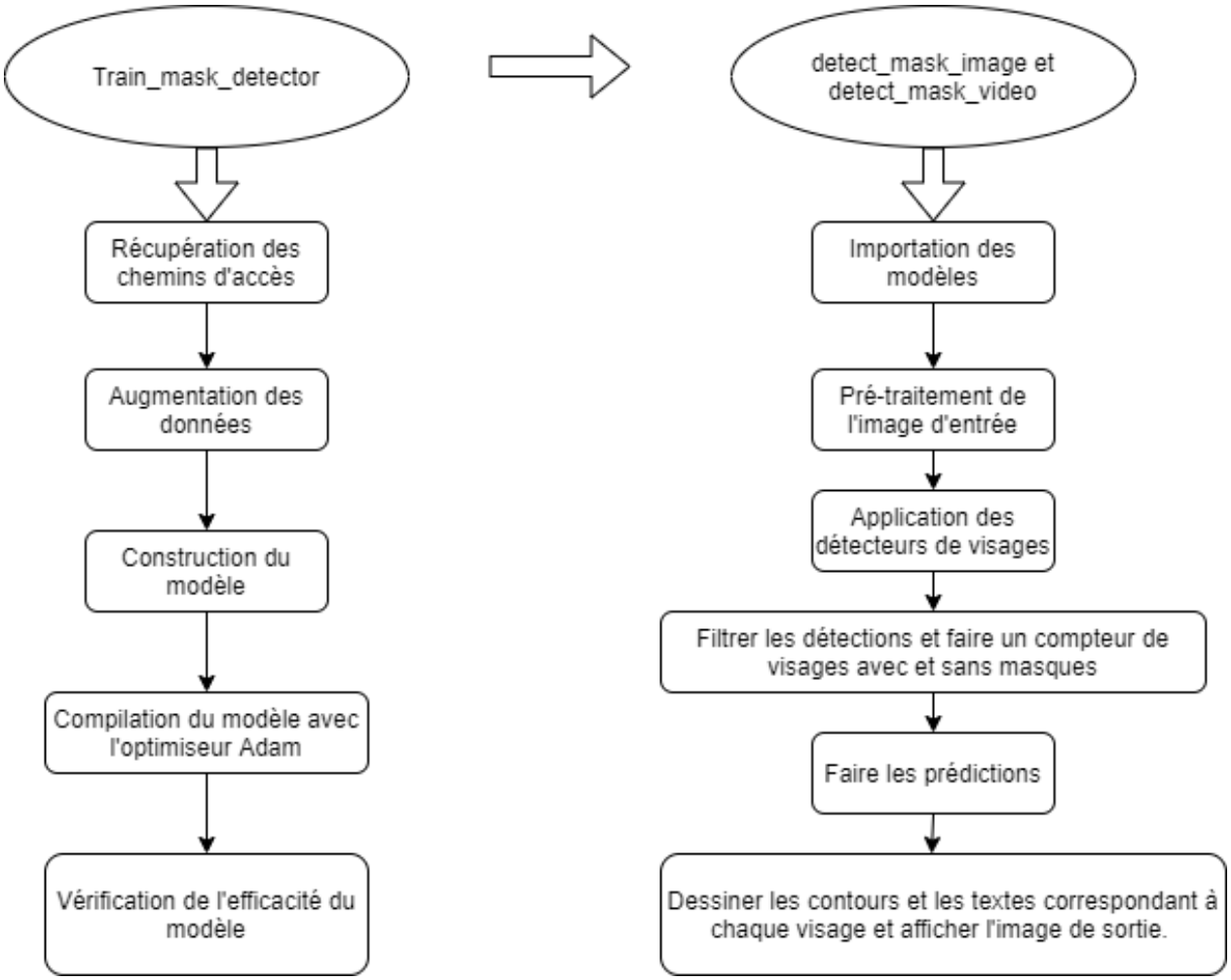

FIGURE 2.1 – Organigramme explicatif de l'application

2.2.1 Entrainement du modèle

Dans ce premier script on a d'abord importé les différentes Bibliothèques dont on aura besoin, comme Keras et TensorFlow pour ainsi former un classificateur qui permet de détecter automatiquement si une personne porte ou non un masque, nous allons affiner l'architecture MobileNet V2, une architecture très efficace qui peut être appliquée aux appareils embarqués avec une capacité de calcul limitée (ex., Raspberry Pi, Google Coral, NVIDIA Jetson Nano, etc.). Le déploiement de notre détecteur de masque facial sur des appareils embarqués pourrait réduire le coût de fabrication de tels systèmes de détection de masque facial, c'est pourquoi nous choisissons d'utiliser cette architecture. MobileNet-v2 est un réseau neuronal convolutif d'une profondeur de 53 couches. Alors qu'ImageNet est une base de données pré-entraînée du réseau formée sur plus d'un million d'images. Le réseau pré-entraîné peut classer les images en 1000 catégories d'objets, telles que le clavier, la souris, le crayon et de nombreux animaux. En conséquence, le réseau a appris de riches représentations de caractéristiques pour une large gamme d'images. Le réseau a une taille d'entrée d'image de 224 x 224 [1]. . De plus, il est important de supprimer les non-linéarités dans les couches étroites afin de maintenir le pouvoir de représentation comme on le verra un peu plus loin grâce à la fonctoin "Dropout". Dans ce type d'architecture, une connexion entre deux nœuds n'est autorisée que depuis les nœuds de la couche i vers les nœuds de la couche i + 1.

De plus, les nœuds de la couche i sont entièrement connectés aux nœuds de la couche i + 1. Cela implique que chaque nœud de la couche i se connecte à chaque nœud de la couche i + 1, c'est de là que vient le terme «entièrement connecté» ou «FC» en abrégé[2].

Pour cela on récupère d'abord les chemins de nos images dont on aura besoins, disponibles dans notre jeu de données dataset qui consiste au redimensionnement des images à 224x224 qui est la norme à utiliser pour ce type de traitement, la conversion au format numpy (matrice), et enfin la normalisation des valeurs des pixels qui vont être dorénavant comprises entre -1 et 1 avec la fonction preprocess_input. La variable label va nous servir à l'affichage selon la nature de l'image si la personne se trouvant dans l'image porte ou non un masque.

On segmente nos données en 80% de formation et les 20% restantes pour les tests.

Après ça, on utilise la technique d'augmentation de données qui consiste à augmenter la quantité de données à entrainer en générant des images dans notre cas grâce à la fonction ImageDataGenerator pour en extraire d'avantages de caractéristiques. Le fonctionnement de cette fonction est le suivant, ImageDataGenerator accepte un lot d'images d'entrée, transforme le lot de manière aléatoire, puis renvoie uniquement les données transformées de manière aléatoire. Le lot d'images utilisé est contenu dans le dossier dataset qui contient une quantité importante de visages avec et sans masques, notre modèle va donc être construit à partir de ces images grâce à l'apprentissage en profondeur[3]. On fait ensuite l'ajustement de la configuration en chargeant MobileNet avec des poids ImageNet pré-entraînés, sans tête de réseau, on construit une nouvelle tête FC et on l'ajoute au réseau. Cette nouvelle tête servira à résumer la présence de caractéristiques dans nos images d'entrée. Un problème avec les images de sortie est qu'elles sont sensibles à l'emplacement des des visages détectés dans l'entrée. Une approche pour traiter cette sensibilité consiste à sous-échantillonner ces détections. Cela a pour effet de rendre les cartes d'entités échantillonnées résultantes, qui sont vers le bas, plus résistantes aux changements de position de l'entité dans l'image. Dans notre cas, nous avons utilisé la mise en commun moyenne(Average Pooling). Cette méthode implique le calcul de la moyenne des valeurs pour chaque carré 7X7 de notre Matrice qui contient les valeurs des pixels. Ensuite avec la fonction "Dense", les 128 fonctions créées grâce à « l'apprentissage », génèrent à partir des pixels d'entrées deux valeurs supérieures à 0 qui sont la probabilité que le visage détecté ait un masque, et la probabilité que le visage n'ait pas de masque. Enfin la fonction d'activation "softmax" revoie la plus grande probabilité à la valeur 1 et la plus petite à la valeur 0. Après avoir eu le résultat qu'on voulait, on gèle toute les couches pour qu'il n'y ait plus de mise à jour des couches inférieures du modèle.

On compile ainsi notre modèle avec l'optimiseur Adam. Ce compilateur prend deux valeurs l'optimiseur et l'erreur, si l'algorithme prédit une valeur alors la fonction d'erreur nous donne si nécessaire la médiocrité du modèle grâce à la fonction d'erreur, et alors l'algorithme utilisera la fonction d'optimisation pour donner une autre estimation, et ainsi de suite. Ce processus est répété 20 fois dans notre cas.

Notre dernière étape consiste à tracer nos courbes de précision et de perte et enregistrer la figure. D'après les courbes disponibles en annexe dans la section 0.3.2, nous obtenons une précision d'environ 99% sur notre ensemble de test.
D'après l'observation de la figure 10, nous pouvons voir qu'il y a peu de signes de sur-ajustement, avec la perte de validation inférieure à la perte d'entraînement. Compte tenu de ces résultats, nous espérons que notre modèle se généralisera bien aux images en dehors de notre ensemble de formation et de test.

2.2.2 Détection de masques dans une image

Dans ce deuxième script qui effectue la détection de masques dans une image d'entrée, on commence par importer quelques modules utiles de la bibliothèque Tensorflow, puis définir les chemins vers notre modèle de détection de visages construit précédemment et le poids correspondant, ensuite nous chargeons notre modèle de détection de masques, pour ensuite faire le prétraitement des images par la classification par apprentissage en profondeur (Deep Learning) grâce à la fonction Blobfromimage.

Cette opération consiste à ce qu'on appelle la soustraction moyenne qui est une technique utilisée pour aider nos réseaux de neurones convolutifs à lutter contre les changements d'éclairage dans les images d'entrée de notre jeu de données. Elle a comme paramètre notre image d'entrée, "scale factor" qui met à l'échelle notre image par un certain facteur et qui est par défaut 1, "size" la taille spatiale de l'image attendue par le réseaux de neuronnes convolutifs, les valeurs de pixels qu'on va soustraire à chaque pixel de l'image[6].

On introduit ensuite l'image prétraitée dans notre modèle, puis on détecte les visages dans l'image avec la fonction « forward » grâce à notre modèle de détection de visages. Après avoir franchi cette étape, on fixe un seuil à partir duquel on valide les détections de visages choisies qui est 0.4 dans notre cas, on détermine la région d'intérêt (ROI) c'est-à-dire la région ou se trouve le visage, on le stocke dans la variable face et on applique un processus de traitement qui consiste à la conversion au format BGR (Blue Green Red) et le redimensionnement de cette région à la taille 224x224 pixels.

On applique ensuite notre modèle de détection de masques sur notre région d'intérêt, on obtient 2 probabilités, la première correspond à l'absence de masques et la deuxième correspond à la présence de masques. On filtre ainsi la probabilité la plus grande et on affiche une image de sortie qui selon le résultat de la prédiction va entourer le visage d'un rectangle d'une couleur rouge et du texte « Mask », si le visage est sans masque, ou un rectangle d'une couleur verte ajoutée au texte « Mask » si le visage est porteur de masque. On a aussi introduit un compteur qui permet de donner le nombre de visages détectés avec et sans masques qu'on affiche dans l'image de sortie.

2.2.3 Détection de masques dans un flux d'images dans une vidéo

Dans ce troisième script, on répète le même processus que précédemment mais cette fois en ayant un flux d'images en entrée généré par la webcam. Le processus est le même que le script précédent, on commence par importer les bibliothèque nécessaires, on construit ensuite notre détecteur de masques avec la fonction detect_and_predict_mask qui a comme arguments "frame" qui est notre flux d'images, "faceNet" qui est notre détecteur de visages, et "maskNet" qui est notre détecteur de masques. Ensuite comme précédemment on fait le pré-traitement de notre image d'entrée avec la fonction "Blobfromimage", et on commence ensuite par détecter les visages qui se trouvent dans l'image grâce au modèle de détection de visages "facenet". La variable "face" coorespond à la région d'intérêt après la détection de chaque visage, qui est alors ajouté à la variable "faces" qui constitue la liste des visages détectés, mais comme mentionné dans le code, on ne garde que les objets détectés qui ont une probabilité supérieure à 0.5 d'être un visage, enfin vient l'étape de détection de masques dans les visages détectés grâce à la fonction "predict" et en se basant sur le modèle de détection de masques construit précédemment. Cette fonction retourne les coordonnées des visages détectés et la prédiction associée à chaque visage détecté. La fonction detect_and_predict_mask de détection de visages et de prédiction de masques étant construite, il est temps d'utiliser nos modèle de détection de visages et de masques construit précédemment dans le script train_mask_detector.
Après le chargement de ces deux modèles, on active la webcam ou une caméra extérieure pour construire notre variable "frame" qui sera notre "image d'entrée" ou plutôt notre "flux d'images d'entrée", on détecte les visages et prédit on s'ils contiennent bien des masques grâce à notre fonction detect_and_predict_mask. Si le visage contient bien un masque le programme continue à tourner en indiquant que le visage est bien masqué, mais dès qu'il détecte un visage sans masque le programme s'arrête pendant cinq seconde puis il reprend la détection normalement en l'ajoutant au compteur. A la fin de l'enregistrement, le programme affiche dans l'image le nombre de visages sans masques détectés.

Chapitre 3

Ressenti de l'expérience :

Mon stage a été en télétravail, il n'y a pas eu donc d'intéraction directe avec l'entreprise, j'ai travaillé avec quelques outils qui me sont familiers comme le language de programmation python ou encore la rédaction du rapport sur Latex. Ce qui m'a été un peu difficile à gérer, ce sont les nouvelles notions du machine learning et les réseaux de neurones qui ne sont pas aussi facile à assimiler du premier coup, mais avec un peu de discussion, de recherche et de persévérance on arrive à comprendre le concept. En effet le stage consistait d'abord à comprendre certaines notions, faire une analyse du modèle utilisé où il fallait chercher des ressources dans des livres et sur internet, ensuite venait la tâche de la valeur ajouté où il fallait trouver les limites du modèle et essayer d'en trouver des solutions, mais aussi proposer d'autres idées qui bien sûr vont être utiles surtout dans cette période de crise.

Personnellement, le télétravail ayant ses avantages et ses inconvénients, il n'accomplit pas le but principal du stage ouvrier qui requiert une interaction directe avec l'entreprise et les autres employés. Pour valider la découverte du monde du travail, je trouve qu'il faut avoir des rapports directs aussi avec l'employeur ainsi que des éventuels partenaires du travail. Mais il ne faut quand même pas nier les avantages du télétravail, en effet le télétravail permet de gagner beaucoup de temps surtout si le lieu de l'entreprise d'accueil se trouve un peu loin du lieu de la résidence principale, deuxièmement en tant que stagiaire on est plus concentré chez soi que quand on est sous la surveillance des employeurs, on est donc plus efficace dans notre travail mais on est bien sûr moins encadré que dans l'entreprise d'accueil et dans ce cas il faut trouver d'autres ressources pour essayer d'avancer en prenant des alternatives ou parfois en contournant quelques problèmes, comme prendre des cours en lignes pour comprendre les modèles disponibles adaptés à mon projet qui était largement plus efficace que d'essayer de construire son propre modèle qui était estimé selon moi à beaucoup plus qu'un ou deux mois de travail.

Étant étranger et dans cette période de crise, le télétravail n'était bien sûr pas aussi évident, puisqu'on ne pouvait pas retourner chez soi et il fallait donc s'adapter aux conditions. Mais ce qui était avantageux c'est qu'on pouvait travailler ailleurs que chez soi pour changer la routine du travail mais aussi voyager, ce qui n'empêchait pas du tout le télétravail puisque tout ce qu'on avait besoin c'est un ordinateur portable et un peu de concentration mais surtout un changement d'air.

Chapitre 4

Prise de recul et analyse de l'expérience :

On va dire que ce stage a répondu à quelques attentes mais pas tous. Pour ce type de projets qui consiste à du traitement d'images par apprentissage en utilisant les réseaux de neurones, j'estime qu'il faut beaucoup plus de temps pour en comprendre le concept et ce n'est pas très optimal pour un étudiant de première année. C'est vraiment intéressant, mais il vaut mieux avoir étudié quelques notions à l'école avant d'entamer ce type de projets qui exige énormément de temps et de patience, qu'on peut améliorer au fur et à mesures qu'ils y'a de nouvelles idées. Ce qui excitant alors dans ce projet c'est que tu peux toujours l'améliorer au fur et à mesures qu'il y'a de nouvelles exigences face à cette pandémie. Mais sincèrement, je ne pense pas avoir appris ce qui est nécessaire sur le monde du travail, parce que ce stage s'est limité à un rapport avec l'encadrant seul, je n'étais en contact avec aucun autre membre de l'entreprise.

J'ai eu cependant la possibilités de passer plusieurs entretiens avec plusieurs entreprises, qui m'a servi à identifier ce qui n'allait pas dans mon profil, à mettre en avant mes avantages en essayant de mener la conversation vers mes points forts pour ainsi dissimuler mes faiblesses. Même si ces entretiens étaient aussi en visio-conférence, j'ai quand même apprécié tout ce que j'ai appris en terme d'expérience de gestion de stress et d'estime de soi, parce qu'il y'avait parfois des stages qui étaient vraiment intéressent, donc il y'avait plusieurs candidats et il fallait donc se démarquer d'une certaine manière. Le problème que j'avais est surtout que j'avais l'intention de faire un stage au Maroc et donc je n'avais pas envoyé de candidatures assez tôt aux entreprises ou au job étudiant en France, ce qui n'était pas bien réfléchi puisque il y'avait beaucoup de candidatures et peu de postes disponibles avec cette période de crise.

Ce stage est cependant une expérience inédite pour moi, venant d'un cursus préparatoire, je n'ai donc jamais fait de stage ici ou à l'étranger, mais je trouve que c'est une expérience à vivre même si dans cette période ce n'était pas aussi amusant que ça avec le respect des gestes barrières et toutes les contraintes qui vont avec. J'ai donc su développer ma patience surtout dans la période de recherche de stage, comment est-ce qu'il faut cibler les entreprises, la façon dont il faut expliquer sont projet et ses réalisations sans être ambigu mais aussi la qualité de la rédaction qui va sans doute jouer sur la compréhension de l'employeur ou de n'importe quel lecteur.

Le conseil que je donnerai aux futurs stagiaires c'est de commencer très tôt la recherche de stages et de ne pas attendre mars ou avril parce que personnellement, j'ai eu du mal à trouver ce stage et de ne surtout rien lâcher, toujours chercher de nouvelles opportunités, les saisir et de choisir ce qui est intéressant pour le stagiaire, ne pas s'engager avec une entreprise alors qu'en même temps il y 'a une opportunité bien meilleure qui se présente à toi. Mais il faut aussi taper toutes les portes, passer plusieurs entretiens s'il le faut, bien réorganiser son profil, cv et lettre de motivation et enfin s'appliquer bien dans son stage afin laisser une bonne impression chez les recruteurs et peut-être, être recommandé ailleurs pour ses futurs stages ou emplois.

Annexe :

4.1 Démonstration théorique des fonctionnalités

4.1.1 L'optimiseur Adam

Un optimiseur est un algorithme utilisé en intelligence artificielle, utilisant généralement une fonction d'erreur ou aussi appelée fonction de coût qui calcule l'efficacité du modèle, et le réajuste ensuite pour gagner plus en précision. L'optimiseur Adam est l'optimiseur le plus utilisé (surtout pour les réseaux de neurones) en raison de son efficacité et de sa stabilité. C'est une méthode de taux d'apprentissage adaptatif, ce qui signifie qu'elle calcule les taux d'apprentissage individuels pour différents paramètres. Son nom est dérivé de l'estimation adaptative des moments, et la raison pour laquelle il est appelé ainsi est parce qu'Adam utilise des estimations des premier et deuxième moments de gradient pour adapter le taux d'apprentissage pour chaque poids du réseau neuronal[4]. Pour estimer les moments, Adam utilise des moyennes mobiles exponentiellement, calculées sur le gradient évalué sur un mini-lot courant :

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$

Où m et v sont des moyennes mobiles, g est un gradient sur le mini-lot actuel, et bêtas - nouveaux hyper-paramètres introduits de l'algorithme. Ils ont de très bonnes valeurs par défaut de 0,9 et 0,999 respectivement. Les vecteurs des moyennes mobiles sont initialisés avec des zéros à la première itération. Puisque m et v sont des estimations des premier et deuxième moments, nous voulons avoir la propriété suivante :

$$m_t = (1-\beta_1) \sum_{i=0}^{t} \beta_1^{t-i} g_i$$

Le but ici est d'avoir des estimateurs égales au paramètre que nous essayons d'estimer. Comme nous initialisons les moyennes avec des zéros, les estimateurs sont biaisés vers zéro. Nous devons maintenant corriger l'estimateur, de sorte que la valeur attendue soit celle que nous voulons. Cette étape est généralement appelée correction de biais. Les formules finales de notre estimateur seront les suivantes :

$$E[m_t] = E[(1-\beta_1) \sum_{i=1}^{t} \beta_1^{t-i} g_i]$$
$$= E[g_i](1-\beta_1) \sum_{i=1}^{t} \beta_1^{t-i} + \zeta$$
$$= E[g_i](1-\beta_1^t) + \zeta$$

La taille de pas réelle prise par l'Adam à chaque itération est approximativement liée à l'hyper-paramètre de taille de pas. Cette propriété ajoute une compréhension intuitive à l'hyper-paramètre de vitesse d'apprentissage non intuitif précédent. La taille du pas de la règle de mise à jour d'Adam est invariante par rapport à l'amplitude du gradient, ce qui aide beaucoup lorsque vous traversez des zones avec de minuscules gradients (tels que des points de selle). La seule chose à faire est d'utiliser ces moyennes mobiles pour mettre à l'échelle le taux d'apprentissage individuellement pour chaque paramètre. La façon dont se fait la mise à jour du poids avec Adam est la suivante :

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

Où w est les poids du modèle, eta est la taille du pas (cela peut dépendre de l'itération).

4.1.2 Résultats de la construction de notre modèle de détecteurs de masques

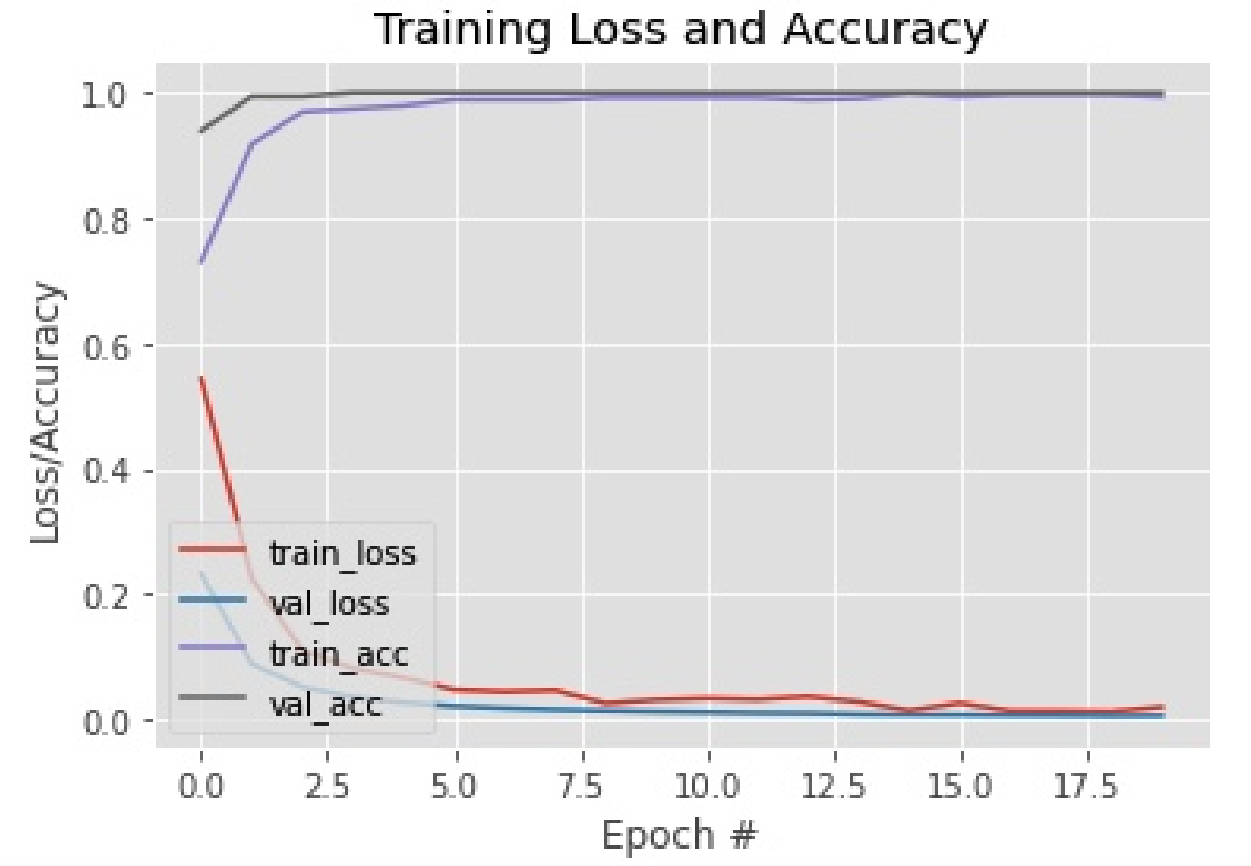

FIGURE 4.1 – Courbes d'erreurs et de précisions

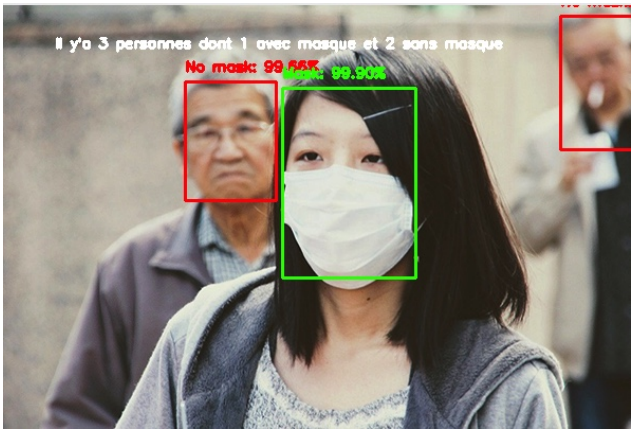

FIGURE 4.2 – Résultat de l'exécution du code sur un exemple d'une image

FIGURE 4.3 – Résultat de l'exécution du code en utilisant la webcam, avec le troisième script

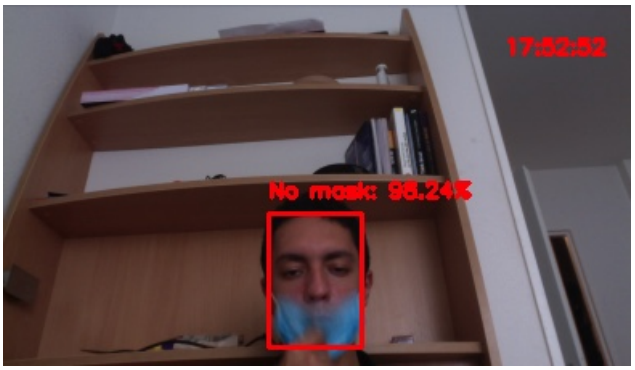

FIGURE 4.4 – Résultat de l'exécution du code en utilisant la webcam, avec le troisième script dès la détection d'un visage sans masque

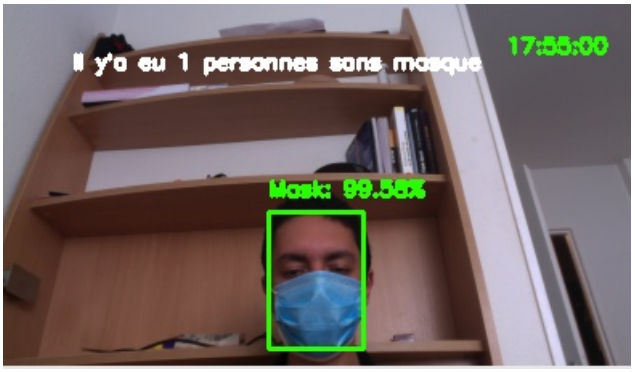

FIGURE 4.5 – Résultat de l'exécution du code en utilisant la webcam, avec le troisième script après la détection d'un visage sans masque

4.2 Ressource et documentation automatique

On présente ici la documentation automatique des principales bibliothèques utilisées pour l'élaboration de ce script python notamment openc, sklearn et Tensorflow.

cv2.cv2 (version 4.2.0)

index
c:\users\dell\anaconda3\lib\site-packages\cv2\cv2.cp37-win_amd64.pyd

Python wrapper for OpenCV.

Package Contents

cv2 — **data** (package)

Classes

builtins.Exception(builtins.BaseException)

- cv2.error

builtins.object

- cv2.Algorithm
  - cv2.AlignExposures
    - cv2.AlignMTB
  - cv2.BackgroundSubtractor
    - cv2.BackgroundSubtractorKNN
    - cv2.BackgroundSubtractorMOG2
  - cv2.BaseCascadeClassifier
  - cv2.CLAHE
  - cv2.CalibrateCRF
    - cv2.CalibrateDebevec
    - cv2.CalibrateRobertson
  - cv2.DenseOpticalFlow
    - cv2.DISOpticalFlow
    - cv2.FarnebackOpticalFlow
    - cv2.VariationalRefinement
  - cv2.DescriptorMatcher
    - cv2.BFMatcher
    - cv2.FlannBasedMatcher
  - cv2.GeneralizedHough
    - cv2.GeneralizedHoughBallard
    - cv2.GeneralizedHoughGuil
  - cv2.LineSegmentDetector
  - cv2.MergeExposures
    - cv2.MergeDebevec
    - cv2.MergeMertens
    - cv2.MergeRobertson
  - cv2.SparseOpticalFlow
    - cv2.SparsePyrLKOpticalFlow
  - cv2.StereoMatcher
    - cv2.StereoBM
    - cv2.StereoSGBM
  - cv2.Tonemap
    - cv2.TonemapDrago
    - cv2.TonemapMantiuk
    - cv2.TonemapReinhard
  - cv2.dnn_Layer
  - cv2.ml_StatModel
    - cv2.ml_ANN_MLP
    - cv2.ml_DTrees
      - cv2.ml_Boost
      - cv2.ml_RTrees
    - cv2.ml_EM
    - cv2.ml_KNearest
    - cv2.ml_LogisticRegression
    - cv2.ml_NormalBayesClassifier
    - cv2.ml_SVM
    - cv2.ml_SVMSGD
- cv2.AsyncArray
- cv2.BOWImgDescriptorExtractor
- cv2.BOWTrainer
  - cv2.BOWKMeansTrainer
- cv2.CascadeClassifier
- cv2.CirclesGridFinderParameters
- cv2.DMatch
- cv2.Feature2D
  - cv2.AKAZE
  - cv2.AgastFeatureDetector
  - cv2.BRISK
  - cv2.FastFeatureDetector
  - cv2.GFTTDetector
  - cv2.KAZE
  - cv2.MSER
  - cv2.ORB
  - cv2.SimpleBlobDetector
- cv2.FileNode
- cv2.FileStorage
- cv2.HOGDescriptor
- cv2.KalmanFilter
- cv2.KeyPoint
- cv2.PyRotationWarper
- cv2.QRCodeDetector
- cv2.SimpleBlobDetector_Params
- cv2.Stitcher
- cv2.Subdiv2D
- cv2.TickMeter
- cv2.UMat

sklearn (version 0.22.1)

index
c:\users\dell\anaconda3\lib\site-packages\sklearn__init__.py

```
Machine learning module for Python
==================================

sklearn is a Python module integrating classical machine
learning algorithms in the tightly-knit world of scientific Python
packages (numpy, scipy, matplotlib).

It aims to provide simple and efficient solutions to learning problems
that are accessible to everybody and reusable in various contexts:
machine-learning as a versatile tool for science and engineering.

See http://scikit-learn.org for complete documentation.
```

Package Contents

__check_build (package)	**datasets** (package)	**inspection** (package)	**neighbors** (package)
_build_utils (package)	**decomposition** (package)	isotonic	**neural_network** (package)
_config	discriminant_analysis	kernel_approximation	pipeline
_distributor_init	dummy	kernel_ridge	**preprocessing** (package)
_isotonic	**ensemble** (package)	**linear_model** (package)	random_projection
base	exceptions	**manifold** (package)	**semi_supervised** (package)
calibration	**experimental** (package)	**metrics** (package)	setup
cluster (package)	**externals** (package)	**mixture** (package)	**svm** (package)
compose (package)	**feature_extraction** (package)	**model_selection** (package)	**tests** (package)
conftest	**feature_selection** (package)	multiclass	**tree** (package)
covariance (package)	**gaussian_process** (package)	multioutput	**utils** (package)
cross_decomposition (package)	**impute** (package)	naive_bayes	

Functions

clone(estimator, safe=True)

```
Constructs a new estimator with the same parameters.

Clone does a deep copy of the model in an estimator
without actually copying attached data. It yields a new estimator
with the same parameters that has not been fit on any data.

Parameters
----------
estimator : estimator object, or list, tuple or set of objects
    The estimator or group of estimators to be cloned

safe : boolean, optional
    If safe is false, clone will fall back to a deep copy on objects
    that are not estimators.
```

config_context(**new_config)

```
Context manager for global scikit-learn configuration

Parameters
----------
assume_finite : bool, optional
    If True, validation for finiteness will be skipped,
    saving time, but leading to potential crashes. If
    False, validation for finiteness will be performed,
    avoiding error.  Global default: False.

working_memory : int, optional
    If set, scikit-learn will attempt to limit the size of temporary arrays
    to this number of MiB (per job when parallelised), often saving both
    computation time and memory on expensive operations that can be
    performed in chunks. Global default: 1024.

print_changed_only : bool, optional
    If True, only the parameters that were set to non-default
    values will be printed when printing an estimator. For example,
    ``print(SVC())`` while True will only print 'SVC()' while the default
    behaviour would be to print 'SVC(C=1.0, cache_size=200, ...)' with
    all the non-changed parameters.

Notes
-----
All settings, not just those presently modified, will be returned to
their previous values when the context manager is exited. This is not
```

tensorflow (version 2.2.0)

index
c:\users\dell\anaconda3\lib\site-packages\tensorflow__init__.py

```
Top-level module of TensorFlow. By convention, we refer to this module as
`tf` instead of `tensorflow`, following the common practice of importing
TensorFlow via the command `import tensorflow as tf`.

The primary function of this module is to import all of the public TensorFlow
interfaces into a single place. The interfaces themselves are located in
sub-modules, as described below.

Note that the file `__init__.py` in the TensorFlow source code tree is actually
only a placeholder to enable test cases to run. The TensorFlow build replaces
this file with a file generated from [`api_template.__init__.py`](https://www.github.com/tensorflow/tensorflow/blob/master/tensorflow/api_template.__init__.py)
```

Package Contents

_api (package)
audio (package)
autodiff (package)
autograph (package)
bitwise (package)
compat (package)
compiler (package)
config (package)
core (package)
data (package)
debugging (package)
distribute (package)
dtypes (package)
errors (package)
estimator (package)
examples (package)
experimental (package)
feature_column (package)
graph_util (package)
image (package)
io (package)
keras (package)
linalg (package)
lite (package)
lookup (package)
math (package)
mixed_precision (package)
mlir (package)
nest (package)
nn (package)
profiler (package)
python (package)
quantization (package)
queue (package)
ragged (package)
random (package)
raw_ops (package)
saved_model (package)
sets (package)
signal (package)
sparse (package)
strings (package)
summary (package)
sysconfig (package)
test (package)
tools (package)
tpu (package)
train (package)
v2
version (package)
xla (package)

Classes

builtins.object

    tensorflow.python.eager.backprop.GradientTape
    tensorflow.python.framework.device_spec.DeviceSpecV2
    tensorflow.python.framework.ops.Graph
    tensorflow.python.framework.ops.Operation
    tensorflow.python.framework.ops.RegisterGradient
    tensorflow.python.framework.ops.name_scope_v2
    tensorflow.python.framework.tensor_shape.TensorShape
    tensorflow.python.framework.type_spec.TypeSpec

        tensorflow.python.data.ops.optional_ops.OptionalSpec
        tensorflow.python.framework.indexed_slices.IndexedSlicesSpec
        tensorflow.python.ops.tensor_array_ops.TensorArraySpec

    tensorflow.python.ops.critical_section_ops.CriticalSection
    tensorflow.python.ops.gradients_util.AggregationMethod
    tensorflow.python.ops.tensor_array_ops.TensorArray

enum.Enum(builtins.object)

    tensorflow.python.ops.unconnected_gradients.UnconnectedGradients
    tensorflow.python.ops.variables.VariableAggregationV2
    tensorflow.python.ops.variables.VariableSynchronization

tensorflow.python._dtypes.DType(pybind11_builtins.pybind11_object)

    tensorflow.python.framework.dtypes.DType

tensorflow.python.framework.composite_tensor.CompositeTensor(builtins.object)

    tensorflow.python.framework.indexed_slices.IndexedSlices(tensorflow.python.framework.tensor_like._TensorLike, tensorflow.python.framework.composite_tensor.CompositeTensor)
    tensorflow.python.framework.sparse_tensor.SparseTensor(tensorflow.python.framework.tensor_like._TensorLike, tensorflow.python.framework.composite_tensor.CompositeTensor)
    tensorflow.python.ops.ragged.ragged_tensor.RaggedTensor

tensorflow.python.framework.tensor_like._TensorLike(builtins.object)

    tensorflow.python.framework.indexed_slices.IndexedSlices(tensorflow.python.framework.tensor_like._TensorLike, tensorflow.python.framework.composite_tensor.CompositeTensor)
    tensorflow.python.framework.ops.Tensor
    tensorflow.python.framework.sparse_tensor.SparseTensor(tensorflow.python.framework.tensor_like._TensorLike, tensorflow.python.framework.composite_tensor.CompositeTensor)

tensorflow.python.framework.tensor_spec.DenseSpec(tensorflow.python.framework.type_spec.TypeSpec)

    tensorflow.python.framework.tensor_spec.TensorSpec(tensorflow.python.framework.tensor_spec.DenseSpec, tensorflow.python.framework.type_spec.BatchableTypeSpec)

tensorflow.python.framework.type_spec.BatchableTypeSpec(tensorflow.python.framework.type_spec.TypeSpec)

    tensorflow.python.framework.sparse_tensor.SparseTensorSpec
    tensorflow.python.framework.tensor_spec.TensorSpec(tensorflow.python.framework.tensor_spec.DenseSpec, tensorflow.python.framework.type_spec.BatchableTypeSpec)
    tensorflow.python.ops.ragged.ragged_tensor.RaggedTensorSpec

tensorflow.python.ops.init_ops_v2.Initializer(builtins.object)

    tensorflow.python.ops.init_ops_v2.Constant
    tensorflow.python.ops.init_ops_v2.Ones
    tensorflow.python.ops.init_ops_v2.RandomNormal
    tensorflow.python.ops.init_ops_v2.RandomUniform
    tensorflow.python.ops.init_ops_v2.Zeros

tensorflow.python.training.tracking.base.Trackable(builtins.object)

    tensorflow.python.ops.variables.Variable

tensorflow.python.training.tracking.tracking.AutoTrackable(tensorflow.python.training.tracking.base.Trackable)

    tensorflow.python.module.module.Module

class **AggregationMethod**(builtins.object)

```
    A class listing aggregation methods used to combine gradients.

    Computing partial derivatives can require aggregating gradient
    contributions. This class lists the various methods that can
    be used to combine gradients in the graph.

    The following aggregation methods are part of the stable API for
    aggregating gradients:

    *  `ADD_N`: All of the gradient terms are summed as part of one
       operation using the "AddN" op (see `tf.add_n`). This
       method has the property that all gradients must be ready and
       buffered separately in memory before any aggregation is performed.
    *  `DEFAULT`: The system-chosen default aggregation method.

    The following aggregation methods are experimental and may not
    be supported in future releases:

    * `EXPERIMENTAL_TREE`: Gradient terms are summed in pairs using
      using the "AddN" op. This method of summing gradients may reduce
      performance, but it can improve memory utilization because the
```

Bibliographie

[1] E.T **Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov Liang-Chieh Chen** *MobileNetV2 : Inverted Residuals and Linear Bottlenecks*

[2] J. **Jason Brownlee** , , *A Gentle Introduction to Pooling Layers for Convolutional Neural Networks*

[3] F. **https ://imgaug.readthedocs.io/en/latest/**

[4] N. **Lambert R.** *Le meilleur optimizer en IA ? Ranger = RAdam + LookAhead, n1 sur FastAI !*

[5] M. **Diederik P. Kingma and Jimmy Lei Ba** *ADAM : A METHOD FOR STOCHASTIC OPTIMIZATION*

[6] W **Adrian Rosebrock**, **Deep learning : How OpenCV's blobFromImage works**