**Détecteur des masques**

L’objectif de notre étude consiste à construire un outil qui permet à partir d’un jeu de données d’images de détecter si la personne possède un masque ou non.

Dans un second temps, nous élargirons notre modèle pour un détecteur en temps réel pour détecter si la personne sur la webcam porte un masque ou non.

On va apprendre à entraîner le détecteur de masque facial COVID-19 avec OpenCV, Keras / TensorFlow et Deep Learning.

* **ETUDE DE NOTRE JEU DE DONNEES :**

Notre dataset se compose de **1376 images** appartenant à deux classes:

* with\_mask : 690 images
* without\_mask : 686 images

L’ objectif est de former un modèle d'apprentissage en profondeur personnalisé pour détecter si une personne *porte* ou *non* un masque.

**STRUCTURE ET ARCHITECTURE DE NOS DONNEES:**

* On a créé un répertoire **dataset** où se trouve nos données réparties en deux sous répertoires **with\_mask** pour les images avec masques et **without\_mask** pour les images sans masques.
* Dans le répertoire examples se trouvent des images fournies pour tester le détecteur de masque facial à image statique.

Puis on a trois scripts Python l:

* **train\_mask\_detector.py**: Accepte notre jeu de données d'entrée et ajuste MobileNetV2 dessus pour créer notre **mask\_detector.model**
* **plot.png**: contenant des courbes de précision / perte.
* **detect\_mask\_image.py**: Effectue la détection du masque facial dans les images statiques
* **detect\_mask\_video.py**: À l'aide d’une webcam, ce script applique la détection du masque facial à chaque image du flux

On va maintenant entraîner notre détecteur de masque facial.

* **Implémentation de notre script de formation au détecteur de masque facial COVID-19 avec Keras et TensorFlow.**
* **dans train\_mask\_detector.py:**

Après l’importation des bibliothèques nécessaires à notre étude (on utilisera notamment pour répertorier les images dans notre ensemble de données. Et nous utiliserons matplotlib pour tracer nos courbes), on définit nos arguments de ligne de commande.

--dataset: Chemin vers le jeu de données d'entrée des visages avec ou sans masque.

--plot: Le chemin d'accès à votre tracé d'historique d'entraînement en sortie, qui sera généré à l'aide de matplotlib

--modèle: Chemin d'accès au modèle de classification de masque facial sérialisé obtenu.

On initie ensuite des constantes d'hyperparamètres, comme mon taux d'apprentissage initial, le nombre d'époques d'entraînement et la taille du lot (batch).

.

On va pouvoir pré-traiter nos données d'entraînement, les étapes sont les suivantes:

* Saisir toutes les **imagePaths** dans le jeu de données.
* Initialisation des données et des labels
* Boucle sur le imagePaths et chargement + prétraitement des images. Les étapes de prétraitement comprennent le redimensionnement à *224 × 224* pixels, la conversion au format de matrice et la mise à l'échelle des intensités de pixels dans l'image d'entrée à la plage *[-1, 1]* (via le preprocess\_input fonction de commodité)
* Mise à jour des listes de données et des labels.
* On S’assure que nos données d'entraînement sont au format de tableau NumPy
* on encode les labels.

On procède au fractionnement de nos données avec une répartition de 80% pour les données d’apprentissage et 20% pour le jeu de test.

On construit ensuite le générateur d’images d’entraînement pour l’augmentation des données.

On doit tout d’abord préparer MobileNetV2 pour le réglage fin de la configuration qui se fait en trois étapes:

* Chargez MobileNet avec des poids Image Net pré-entraînés.
* Construisez une nouvelle tête FC et ajoutez-la à la base à la place de l'ancienne tête
* Gelez les couches de base du réseau . Les poids de ces couches de base ne seront pas mis à jour pendant le processus de rétropropagation, tandis que les poids de couche de tête *seront* ajustés.

Cette méthode permet d’avoir un modèle de référence.

Avec nos données préparées et l'architecture du modèle en place pour un réglage précis, nous sommes maintenant prêts à compiler et à former notre réseau de détecteurs de masques faciaux:

Oncompile notre modèle avec le **Adam optimiseur**, unprogramme de décroissance du taux d'apprentissage.

Une fois la formation terminée, on évalue le modèle sur l'ensemble de test.

On effectue nos prédictions sur l’ensemble de test et on affiche les courbes de précision et de perte.

**Entraînement du détecteur de masque facial COVID-19 avec Keras / TensorFlow**

On peut entraîner notre détecteur de masque facial à l'aide de Keras, TensorFlow et Deep Learning.

une fois positionné sur le bon répertoire , on execute en ligne de commande l’instruction suivante: **python train\_mask\_detector.py -d dataset -m model -p myplot**





**La modélisation nous permet d’obtenir une précision d'environ 99% sur notre ensemble de test.**

Au vue des résultats fournis sur le modèle on remarque qu'il y a peu de signes de surajustement.

**Implémentation de notre détecteur de masque facial COVID-19 pour les images avec OpenCV**

Maintenant que notre détecteur de masque facial est formé, apprenons comment nous pouvons:

Charger une image d'entrée à partir du disque, détecter les visages dans l'image, appliquer notre détecteur de masque facial pour classer le visage comme with\_mask, ou sans\_masque.

Dans **detect\_mask\_image.py,** nous allons faire le travail suivant:

Après importation des bibliothèques nécessaires dons OpenCV pour la manipulation d'mages , on analyse les arguments de lignes de commandes:

--image: Le chemin vers l'image d'entrée contenant les visages pour l'inférence

--face: Le chemin vers le répertoire du modèle du détecteur de visage (nous devons localiser les visages avant de les classer)

--model: Le chemin vers le modèle de détecteur de masque facial que nous avons formé plus tôt dans ce tutoriel

--confidence: Un seuil de probabilité facultatif peut être défini pour remplacer 50% afin de filtrer les détections de visage faibles.

Ensuite, on charge à la fois nos modèles de détecteur de visage et de classificateur de masque facial.

Avec nos modèles d'apprentissage en profondeur désormais en mémoire, la prochaine étape consiste à charger et pré-traiter une image d'entrée.

Lors du chargement de notre --image on fait une copie et on saisit les dimensions du cadre pour une mise à l'échelle et un affichage.

Le prétraitement est géré par la fonction blobFromImage d' OpenCV. Comme indiqué dans les paramètres, on redimensionne à 300 × 300 pixels et on effectue une soustraction moyenne.

On effectue ensuite une détection de visage pour localiser où se trouvent tous les visages dans l'image.

Une fois que nous savons où chaque visage est prévu, nous nous assurerons qu'ils répondent aux seuil de confiance avant d'extraire les facesROIs.

On fait ensuite une boucle sur notre détection et on extrait la confiance et on mesure par rapport au seuil de confiance.

On calcule ensuite la délimitation de la boîte d'affichage dans laquelle figure l'image du visage.

Ensuite, nous allons exécuter le retour sur investissement du visage via notre modèle MaskNet.

Puis, on extrait le visage ROI via le découpage NumPy, on pré-traite le retour sur investissement et on effectue une détection de masque pour faire les prédictions avec ou sans masque.

On détermine la classe étiquette en fonction des probabilités renvoyées par le modèle de détecteur de masque et on attribue une couleur qui sera «vert» pour la présence de masque et «rouge» pour sans\_masque.

On dessine ensuite l'étiquette texte (y compris la classe et la probabilité), ainsi qu'un cadre de sélection rectangle pour le visage, en utilisant les fonctions de dessin OpenCV.

Une fois toutes les détections traitées, on affiche la sortie image.

**Détection du masque facial COVID-19 dans les images avec OpenCV.**

On va tester le détecteur de masques COVID en allant en ligne de commandes et pour celà on tape l'instruction:

**python detect\_mask\_image.py --image examples/example\_01.png**

On effectue le test à partir d'une image se trouvant dans le répertoire examples.

si un visage n'est pas trouvé, le détecteur de masque ne peut pas être appliqué. Ceci est dû au fait que l'image peut être obscurci par le masque.

**Implémentation de notre détecteur de masque facial COVID-19 dans des flux vidéo en temps réel avec OpenCV.**

Le détecteur fonctionne avec des images statiques et nous voir si il fonctionne en temps réel.

On utilise **detect\_mask\_video.py** pour réaliser ce travail:

Le fichier a la même structure que pour les images statiques.

On importe un VideoStream classe et temps. Ces deux éléments nous aideront à travailler avec le flux. Nous profiterons également de imutils pour sa méthode de redimensionnement sensible à l'aspect.

On utilise la fonction **detect\_and\_predict\_mask** pour faire le travail:

Elle accepte trois paramètres

frame: Une image de notre flux

faceNet : Le modèle utilisé pour détecter où sont les visages dans l'image

maskNet: Notre modèle de classificateur de masque facial COVID-19

Dans le modèle, on va détecter les visages et initialiser les listes, dont deux sont définies pour renvoyer la fonction. Ces listes incluent nos visages (c.-à-d., ROI), locs (les emplacements du visage), et préds (la liste des prédictions de masque / pas de masque).

On fait une boucle, pour filtrer les détections faibles et on extrait les boîtes englobantes tout en veillant à ce que les coordonnées de la boîte englobante ne tombent pas en dehors des limites de l'image.

Ensuite, on ajoute des ROI faciaux à deux de nos listes correspondantes.

Après avoir extrait les retours sur investissement des faces et le prétraitement nous ajoutons le visage ROI et cadres de délimitation à leurs listes respectives.

On peut exécuter notre visage grâce à notre prédicteur de masque.

La logique de l'algorithme ici est construite pour la vitesse. Tout d'abord, on s'assure qu'au moins un visage a été détecté.

On définit les arguments en ligne de commande qui incluent:

--visage : Le chemin vers le répertoire du détecteur de visage

--model: Le chemin vers notre classificateur de masque facial qualifié

--confidence: Le seuil de probabilité minimum pour filtrer les détections de visage faibles

Ici, on a initialisé notre:

Détecteur de visage

Détecteur de masque facial COVID-19

Flux vidéo de la webcam

On passe en boucle sur les images du flux:

Dans notre boucle sur les résultats de prédiction, on va créer un cadre de délimitation de visage et masquer / ne pas masquer la prédiction.

On détermine le cadre et la couleur

Enfin, on affiche les résultats et on effectue le nettoyage:

On a introduit également deux images de smiley dans une fenêtre à part signalant si on porte un masque ou non.

**Détection des masques faciaux COVID-19 avec OpenCV en temps réel**

Pour lancer le dérecteur de masques en temps réél, on lance en ligne de commande l'instruction suivante:

**python detect\_mask\_video.py**

**Conclusion:**

On constate que le détecteur de masques fonctionnent bien en temps réel et donne de bonnes prédictions.

Le détecteur fournit de bons résultats malgré des données d'entraînement limitées.

Pour améliorer le modèle, il faudrait rassembler des images de visages qui peuvent «confondre» notre classificateur en lui faisant croire que la personne porte un masque alors qu'en fait ce n'est pas le cas (comme des foulard sur le visage par exemple.)