



# Intelligence Artificial Project Emotion and orientation detection

Leprêtre Romain & Ameeuw Basile  
195351 & 195371

12 janvier 2022



# Table des matières

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Prérequis</b>	<b>3</b>
<b>3</b>	<b>Structure</b>	<b>4</b>
<b>4</b>	<b>Fonctionnalité</b>	<b>5</b>
4.1	Entraînement . . . . .	5
4.1.1	Masque ou pas masque? . . . . .	5
4.1.2	Quel émotion? . . . . .	6
4.2	Exécution . . . . .	8
4.2.1	Reconnaissance faciale . . . . .	8
4.2.2	Détection du masque . . . . .	8
4.2.3	Détection de l'émotion . . . . .	8
4.2.4	Détection de l'orientation . . . . .	9
<b>5</b>	<b>Comment utilisé</b>	<b>9</b>
5.1	Entraînement . . . . .	9
5.2	Exécution . . . . .	9
<b>6</b>	<b>Dataset</b>	<b>9</b>
6.1	7 différentes émotions . . . . .	9
6.2	Avec ou sans masque . . . . .	10
<b>7</b>	<b>Conclusion</b>	<b>10</b>
7.1	Objectifs atteints / non atteints . . . . .	10
7.2	Pistes d'amélioration . . . . .	11
<b>8</b>	<b>Références</b>	<b>11</b>

Tout d'abord veuillez noter que ce rapport porte sur une application dont le code se trouve sur un repo github à l'adresse suivante  
⇒ <https://github.com/BasileAmeeuw/IA-project-orientation-and-mood-detection>

# 1 Introduction

L'objectif de ce projet est d'implémenter une application de reconnaissance faciale capable de distinguer l'émotion et l'orientation du visage. Ce projet étant une évaluation noté, certaines contraintes valent un certain nombre de points. Cette grille d'évaluation peut être retrouvé ci-dessous (total 20 points).

Reconnaissance faciale pour plusieurs visages	+ 3
Bonus si en temps réel	+ 1
Detection de l'émotion	+ 1
Detection de l'orientation	+ 2
Placement d'une emoji	+ 1
Detection d'une personne masqué avec emoji adapté	+ 2
Application mobile	+ 1
Entrainement de l'IA personel	+ 2
Rapport + documentation et présentation	+ 7

# 2 Prérequis

Ici nous allons expliquer comment etre dans les bonnes conditions pour pouvoir lancer l'application sur n'importe quel appareil Windows, Mac ou Linux.

Avant toute chose, l'application tourne en Python donc assurez vous d'avoir une version assez récente de Python 3 de préférence.

Ensuite, dans un premier temps, une des choses les plus compliqué est l'installation de Dlib. Pour pouvoir l'installer il va d'abord falloir installer cmake pour pouvoir utiliser ce framework et ensuite mettre le setup (je vous laisse parcourir internet si vous avez des doutes sur l'installation mais ce n'est pas très compliqué. Une fois cmake bien intégré il vous suffit d'installer dlib grâce à la commande `conda install -c conda-forge dlib` sur Anaconda ou `pip install dlib` autrement.

Ensuite il vous faudra installer quelques librairies Python qui se trouve dans le fichier requirements.txt et que vous pouvez retrouver ci-dessous :

```
tensorflow>=2.5.0*
keras==2.4.3
imutils==0.5.4
numpy==1.19.5
opencv-python>=4.2.0.32
matplotlib==3.4.1
argparse==1.4.0
scipy==1.6.2
scikit-learn==0.24.1
pillow>=8.3.2
streamlit==0.79.0
```

FIGURE 1 – Prérequis

Les autres sont des librairies déjà présente dans le Python de base.

**Optionel** : Si vous voulez effectuer l'entraînement vous même je vous conseille de faire tourner celui-ci sur un ordinateur avec GPU et pouvant supporter CUDA car en le forçant sur le CPU le travail pourrait durer des heures et voir même ne pas être supporté. (Il existe également des alternatives en ligne permettant de profiter de GPU partagé!)

### 3 Structure

La structure des différents éléments de l'application est assez clair. On peut retrouver 3 files importants à savoir :

- `main.py`
- `emotion_training.py`
- `mask_training.py`

C'est trois premiers fichiers Python sont respectivement le fichier pour lancer l'application (pour image et vidéo) et les deux suivants sont les fichiers pour entrainer le modèle.

Nous avons également fourni nos detection séparé à savoir `mask_detection.py` et `emotion_detection.py`.

Ensuite nous avons également quelques répertoires :

- Les répertoires "dataset"
  - Un répertoire `dataset_emotion`
  - Un répertoire `dataset_mask`
- Deux répertoires d' "images"
  - Un répertoire `image_saved`
  - Un répertoire `imageSmiley`
- un répertoire `model`

Les répertoires *dataset* contiennent toutes les images d'entraînements et de tests séparés en deux pour les images pour les émotions et les images pour le port du masque.

Ensuite les deux répertoires d'images sont pour l'un un répertoire contenant les images qui s'affichent pour remplacer la tête des acteurs de l'application et l'autre est un répertoire pouvant contenir les enregistrement effectué par l'application.

Enfin le dernier répertoire contient tous les différents modèles utilisés dans l'application.






 <code>deploy.prototxt</code>	23/11/2021 13:19	PROTOTXT File	30 KB
 <code>mask_detector.model</code>	31/12/2021 01:10	MODEL File	11.222 KB
 <code>model.json</code>	25/11/2021 22:43	JSON Source File	10 KB
 <code>model_weights.h5</code>	25/11/2021 22:43	H5 File	17.556 KB
 <code>res10_300x300_ssd_iter_140000.caffemod...</code>	23/11/2021 13:19	CAFFEMODEL File	10.417 KB

FIGURE 2 – Modèles

Rentrons maintenant dans ce dernier. Il y a 5 modèles différents pour trois fonctionnalités en effet `deploy.prototxt` et `res###.caffemodel` vont de paire ainsi que le modèle `model.json` et le `model_weights.h5`. L'explication des ces derniers sera expliqué dans le point fonctionnalité de ce rapport.

Enfin, il reste dans ce dossier un autre dossier `dlib-19.9` contenant les informations nécessaires au bon fonctionnement du Framework ainsi qu'un fichier `shape_predictor_68_face_landmarks.dat` afin de déterminer le type de déterminator pour le Framework.

## 4 Fonctionnalité

### 4.1 Entraînement

Pour l'entraînement les deux entraînements utilisent une réseau neuronal convolutif. Pour l'un il utilise une architecture assez connus (surtout dans les applications mobiles) à savoir *MobileNetV2* et dans l'autre on utilise une architecture et un modèle séquentielle plus complexe.

#### 4.1.1 Masque ou pas masque ?

Dans un premier temps il faut savoir que pour cet entraînement on utilise un dataset qui comporte deux répertoire à savoir un répertoire `with_mask` contenant toutes les images de personnes masquées et `without_mask` contenant toutes les images de personnes sans masques.

La première étape est donc de prendre ces deux répertoires et dans deux nouveaux dossiers virtuels `train` et `test`. Pour ce faire on va prendre 80% des photos de `with_mask` et faire un nouveau dossier virtuel `with_mask` dans `train` puis ensuite faire la même opératio avec l'autre répertoire. Ensuite on prend les 20% restant pour faire deux nouveaux dossiers dans `test`.

Cela a pour avantage d'avoir des échantillons différents pour chaque entraînement. Une fois ce réarrangement effectué on peut rentrer dans le vif du sujet.

L'entraînement utilise d'abord un modèle déjà implémenté ayant une architecture comme on peut le voir sur la figure suivante.

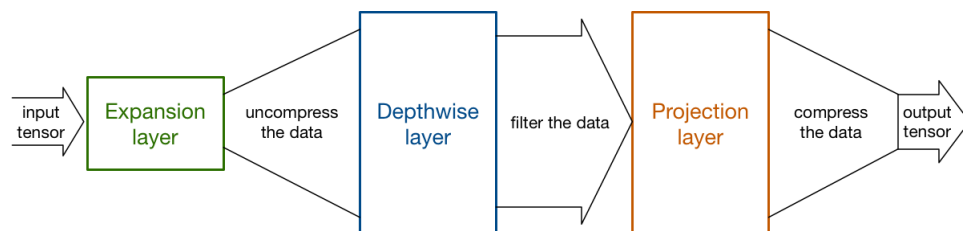


FIGURE 3 – MobileNetV2 architecture

On va donc utiliser ce modèle que l'ont va venir placer **sur** un modèle plus simple et avec des connections convolutionnelles.

```

headModel = baseModel.output
headModel = AveragePooling2D(pool_size=(7, 7))(headModel)
headModel = Flatten(name="flatten")(headModel)
headModel = Dense(128, activation="relu")(headModel)
headModel = Dropout(0.5)(headModel)
headModel = Dense(2, activation="softmax")(headModel)

```

FIGURE 4 – Modèle convolutionnelle

Comme optimisateur on utilise un optimisateur qui utilise l'*algorithme du gradient stochastique* et on utilise pas de callback étant donné qu'on a un classement sur deux possibilités et que 20 epochs suffisent à avoir un bon résultat.

**Remarque :** Nous avons utilisé cette entrainement qui est repris du repo suivant : <https://github.com/chandrikadeb7/Face-Mask-Detection> car il fonctionne très bien et donne de très bon résultat comme on peut le constater sur la figure ci-après.  
(LearningRate=1e-4, Epoch=20, BatchSize=32)

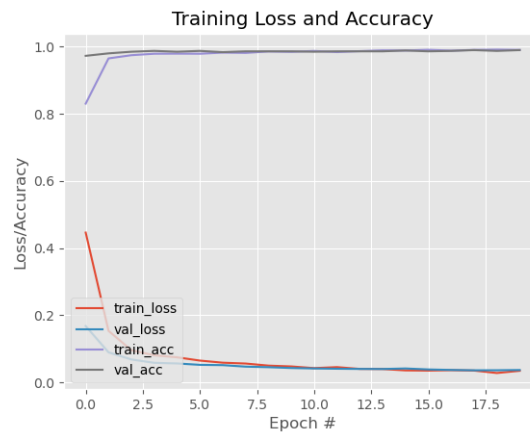


FIGURE 5 – Perte et précision

Enfin une fois l'entrainement terminé on sérialize et on enregistre le modèle. Pour ce faire on le sauve dans un format h5 qui offre la possibilité d'enregistrer un grand nombre de données structurées

#### 4.1.2 Quel émotion ?

Pour cet entrainement-ci contrairement au précédent on opte pour un dataset déjà "trié" avec 20% de tests et 80% de train. Ceci principalement dans une optique de gain de temps et car l'échantillonnage a été logiquement effectué dans de bonnes conditions au préalable. On va donc directement commencer l'entrainement grâce au réseau convolutif de neurones.



FIGURE 6 – Architecture modèle émotion

Le modèle pour lequel nous avons opté après quelques recherches sur internet est le suivant.

On a 2 systèmes de callbacks. Un qui réduit le `learning_rate` si le modèle stagne et qui arrête l'entraînement à partir d'un certain `learning_rate` et un qui enregistre le modèle si c'est le meilleur depuis le début et écrase l'ancien meilleur.

Ensuite on plot le résultat et on enregistre le modèle un peut différemment comme ici nous avons plus de possibilité. On enregistre le modèle dans un fichier `.json` et les poids dans un fichier `.h5`

## 4.2 Exécution

Pour l'exécution nous avons divisé celle-ci pour le rapport en 4 parties à savoir

1. La reconnaissance faciale
2. La détection des masques
3. La détection des émotions
4. La détection de l'orientation

A savoir que dans l'application elle se déroule également dans cet ordre.

### 4.2.1 Reconnaissance faciale

La reconnaissance faciale se passe grâce à un modèle fourni par `caffe`. En effet on réalise un modèle grâce au deux fichiers `prototxt` et `res10_300x300_ssd_iter_140000.caffemodel`. En effet on va détecter grâce à ce modèle un certain nombre de carré avec une certaine confiance. Grâce à un `threshold` qu'on va imposer on peut récupérer les visages plutôt évidents ou non selon celui-ci. Enfin on va parcourir chaque visage un par un pour passer à l'étape suivante.

**Remarque :** Nous n'avons pas utilisé `haarcascade` qui est plus simple car les résultats étaient bien en dessous en terme de reconnaissance faciale.

### 4.2.2 Détection du masque

Pour cette étape nous chargeons le modèle que nous avons obtenu grâce à l'entraînement et après avoir `resize` et `changer` les couleurs de la photos on rentre juste l'image du visage dans le modèle et il nous renvoie une probabilité de masque (et de non masqué) On définit alors un `threshold` (de 0.5) pour nous et s'il est supérieur on définit le visage comme masqué. Si celui-ci est inférieur alors on passe à l'étape suivante pour détecter l'émotion

### 4.2.3 Détection de l'émotion

Pour détecter l'émotion sur le visage concerné si celui-ci n'est pas masqué, nous chargeons dans un premier temps le modèle obtenu grâce à l'entraînement. Vu que l'entraînement est effectué en `grayscale` nous devons dans un premier temps prendre le visage et le `resize` pour avoir la même taille que le modèle ensuite on introduit l'image dans le modèle qui nous renvoie une probabilité pour chaque émotion (un vecteur de taille 7). Ensuite il suffit de choisir l'index dans le vecteur de la probabilité la plus grande. Et grâce à cet index on obtient l'émotion la plus probable. Une fois l'émotion trouvé ou si la personne est masqué nous passons ensuite à l'étape suivante.



#### 4.2.4 Détection de l'orientation

Une fois l'émotion ou le masque détecté on va passer à la détection de l'orientation de chaque visage. Pour se faire on va utiliser la librairie dlib. Dans un premier temps, on va définir un détecteur, ensuite on va prédire une région d'un visage et on va regarder grâce à différents points sur le visage la position des yeux, de la bouche, du nez etc. Une fois ceux-ci obtenus on va définir l'angle du visage en comparant ceux-ci avec une référence de base. Après quoi nous allons placer le smiley avec l'orientation détecté.

## 5 Comment utilisé

### 5.1 Entraînement

Pour effectuer ces entraînements, si vous avez simplement téléchargé le repo Github il vous suffit d'exécuter le fichier Python grâce à `python mask_training.py` ou bien `python emotion_training.py` pour entraîner respectivement le modèle pour détecter le masque ou celui pour détecter l'émotion.

Ensuite peut mettre 3 arguments différents :

- `--dataset` suivi d'un "path" pour définir le path de l'endroit du dossier avec le dataset
- `--plot` suivi d'un nom d'image pour l'enregistrement du plot de la précision et la perte de l'entraînement
- `--model` suivi d'un nom de modèle pour enregistrer le modèle avec ce nom

### 5.2 Exécution

Pour l'exécution de l'application, une fois toutes les différentes dépendances installées et configurées il vous suffit d'exécuter `python main.py` dans le dossier le contenant. Ceci exécutera l'application avec votre caméra en temps réel. Si vous voulez exécuter l'application sur une image ou une vidéo il vous faudra utiliser les arguments suivant :

- `--video` suivi du path de la vidéo que vous voulez analyser
- `--image` suivi du path de l'image que vous voulez analyser

## 6 Dataset

Il y a deux datasets pour cette application étant donné qu'il y a deux entraînements différents comme expliqué ci-avant. Ces deux datasets sont présentés ci-dessous.

### 6.1 7 différentes émotions

Pour la partie emotion nous avons utilisé le célèbre Dataset FER contenant un certains nombres d'images pour chaque catégorie comme on peut le voir sur la figure ci-dessous. Ces catégories sont au nombre de 7 à savoir "Angry", "Disgust", "Fear", "Happy", "Neutral", "Sad" et "Surprise". on peut également apercevoir sur les deux dernières lignes que le dataset est divisé en deux parties (une partie entraînement et une partie test) on peut également apercevoir qu'il s'agit de 80%

( $0,8 \times 35\,887 = 28\,709,6$ ) pour l'entraînement et 20% pour les tests. ( $0,2 \times 35\,887 = 7\,177,4$ )

```
3995 angry images
436 disgust images
4097 fear images
7215 happy images
4965 neutral images
4830 sad images
3171 surprise images
Found 28709 images belonging to 7 classes.
Found 7178 images belonging to 7 classes.
```

FIGURE 7 – Initialisation dataset pour emotions

**Remarque :** Il est tout de même important de noter que ce dataset n'est pas énorme et ne contient surtout que des hommes et femmes de type européen ce qui influence assez fort les résultats obtenus.

## 6.2 Avec ou sans masque

Le Dataset utilisé ici est beaucoup moins populaire et est un mélange de plusieurs en effet, il s'agit d'un Dataset en partie récolté sur le repo suivant : <https://github.com/chandrikadeb7/Face-Mask-Detection> mais c'est surtout un mélange d'image venant de Kaggle et RMFD (deux sites très populaires de Dataset) contenant uniquement des images de la réalité.

**Remarque :** Pour ce dataset contrairement à l'autre il y a surtout des personnes asiatiques néanmoins dans ce cas de figure-ci cela influence les résultats avec moins d'importance.

# 7 Conclusion

## 7.1 Objectifs atteints / non atteints

Objectifs	Priorité	Lien	Réalisé ?
Reconnaissance faciale pour plusieurs visages	1	main.py	100%
En temps réel	2		
Détection de l'émotion	1	detect_emotion_video.py	100%
Détection de l'orientation	1	orientation_detection.py	100%
Placement d'une emoji	2	main.py	100%
Détection d'une personne masquée avec emoji adapté	3	mask_detection.py	100%
Application mobile	4	Uniquement des recherches effectuées sans implémentation	20%
Entraînement de l'IA personnel	2	emotion_training.py mask_training.py	100%
Rapport + documentation et présentation	1	README + rapport	100%

## 7.2 Pistes d'amélioration

- Se pencher sur l'application mobile
- Emettre plus de possibilité d'argument pour rendre l'application plus scalable
- Etirer le dataset pour s'étendre à d'autres continents
- Améliorer la qualité de la reconnaissance de l'orientation
- Structure et commentaire du code
- Détecter l'orientation avant le masque et l'émotion

## 8 Références

[https://github.com/ai-coordinator/Face\\_orientation](https://github.com/ai-coordinator/Face_orientation)  
<https://github.com/dhruvpandey662/Emotion-detection>  
<https://analyticsindiamag.com/top-8-datasets-available-for-emotion-detection/>  
<https://github.com/atulapra/Emotion-detection>  
<https://github.com/iamcal/emoji-data>  
<http://dlib.net/>  
<https://keras.io/api/>  
<https://github.com/chandrikadeb7/Face-Mask-Detection>  
<https://github.com/opencv/opencv/tree/master>  
[https://on-demand.gputechconf.com/gtc/2015/webinar/deep-learning-course/  
getting-started-with-caffe.pdf](https://on-demand.gputechconf.com/gtc/2015/webinar/deep-learning-course/getting-started-with-caffe.pdf)  
<https://caffe2.ai/docs/tutorial-models-and-datasets.html>