Ecole **S**upérieure d'**I**ngénieurs
Numérique et **M**atériaux

Université de **B**ourgogne
Franche-Comté

Rapport PROJET : SYTEMES INTELLIGENTS AVANCES

Thème : Détection de port de masque (Machine learning)

Etudiants:

MEKONGO ABANDA Yannick

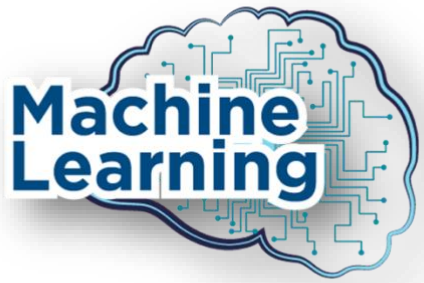
IT5 ILC

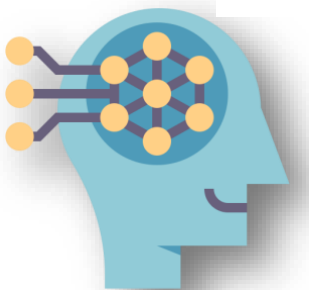
TONFE TCHATAT DAPHNIE

IT5 ILC

Professeur/Examinateur :

M. Brousse Olivier





PRESENTATION DE NOTRE PROJET :

Ce projet avait pour objectif principal de nous familiariser avec les concepts d'apprentissage automatique, dans le domaine de l'intelligence artificiel. Il était question pour nous de concevoir un algorithme de détection du port de masque. Cet algorithme devait donc être capable de détecter si un individu porte bien son masque, porte mal ou ne porte pas de masque. Avant de présenter notre travail, il est important de présenter les outils/Framework que l'on a utilisé pour arriver à la solution finale.

Framework utilisés :

- Pour la préparation et l'entraînement de notre modèle nous avons utilisé KERAS + TensorFlow
- Pour la détection des visages nous avons utilisés CaffeModell via openCV

Afin de réaliser notre modèle, nous avons eu recours aux étapes ci-dessous :

- La récupération automatique des visages contenus sur chaque image de la base générale des données via un script python que nous avons écrit.
- La préparation de notre ensemble de données.
- La création de notre jeu de données.
- La visualisation des données de notre ensemble de données.
- La configuration de l'ensemble des données pour les performances.
- La création du modèle.
- La compilation du modèle.
- Un bref résumé du modèle.
- La formation de notre modèle.
- La visualisation de la précision d'entrainement et de validation ainsi que des valeurs d'erreurs.
- Et enfin la prédiction du modèle, soit sur Flux vidéo, soit sur une image statique.

La récupération automatique des visages pour une classification des données

Nous avons opté pour une classification de nos données. Ainsi nous avons écrit un script python qui parcours les annotations (.XML), récupère les données de visages associées (et leur label) à l'image correspondante à celle-ci, puis sauvegarde chaque visage en tant qu'image avec pour

nom son label + un numéro d'itération. Ainsi on a des données moins volumineuses dans notre espace de stockage. A la fin nous avons donc 03 sous-dossiers (mask_weared_incorrect, with_mask et without_mask) contenant les visages correspondants.

La préparation de notre ensemble de données

Dans un dossier nommeé « dataset » nous avons 03 classes donc 03 dossiers :

- Mask_weared_incorrect
- With_mask
- Without_mask

La création de notre jeu de donnée :

Nous avons défini quelques paramètres pour le chargeur :

- Batch_size = 32
- Dimensions de l'image = 224x224
- Nous avons utilisé une séparation de validation lors du développement de notre modèle.
- Nous utilisons 80% des images pour la formation et 20% pour la validation.

VISUALISATION DE NOS DONNEES :

Nous générons un petit échantillon de nos données d'entraînement pour vérifier que chaque image est bien identifiée par son label :

CREATION DU MODEL, ENTRAINEMENT ET HISTORIQUE

Après avoir eu plusieurs tentatives d'entrainement nous avons opté pour l'augmentation des données et le dropOut :

- **Statistiques sans augmentation des données et sans DropOut :** Regardez l'image ci-dessous, on remarque que notre modèle n'est pas tout à fait au point car la précision de l'entrainement et de la validation sont largement faussées.


```
Epoch 1/10
191/191 [==============================] - 207s 1s/step - loss: 0.5150 - accuracy: 0.8088 - val_loss: 0.3799 - val_accuracy: 0.8630
Epoch 2/10
191/191 [==============================] - 183s 956ms/step - loss: 0.3752 - accuracy: 0.8771 - val_loss: 0.3700 - val_accuracy: 0.8689
Epoch 3/10
191/191 [==============================] - 178s 935ms/step - loss: 0.3409 - accuracy: 0.8855 - val_loss: 0.3196 - val_accuracy: 0.8919
Epoch 4/10
191/191 [==============================] - 177s 929ms/step - loss: 0.3061 - accuracy: 0.8967 - val_loss: 0.3300 - val_accuracy: 0.8919
Epoch 5/10
191/191 [==============================] - 178s 931ms/step - loss: 0.2719 - accuracy: 0.9025 - val_loss: 0.3573 - val_accuracy: 0.8860
Epoch 6/10
191/191 [==============================] - 178s 931ms/step - loss: 0.2204 - accuracy: 0.9249 - val_loss: 0.3437 - val_accuracy: 0.8893
Epoch 7/10
191/191 [==============================] - 178s 930ms/step - loss: 0.1979 - accuracy: 0.9295 - val_loss: 0.4190 - val_accuracy: 0.8683
Epoch 8/10
191/191 [==============================] - 186s 974ms/step - loss: 0.1688 - accuracy: 0.9353 - val_loss: 0.3904 - val_accuracy: 0.8814
Epoch 9/10
191/191 [==============================] - 178s 929ms/step - loss: 0.1098 - accuracy: 0.9651 - val_loss: 0.4613 - val_accuracy: 0.8879
Epoch 10/10
191/191 [==============================] - 194s 1s/step - loss: 0.0958 - accuracy: 0.9666 - val_loss: 0.5423 - val_accuracy: 0.8539
```

- **Statistiques après augmentation des données et sans DropOut** Après avoir fait du surraprentissage à savoir l'augmentation des données et le dropOut on remarque que notre entraînement produit de bons résultats :


```
Layer (type)                 Output Shape              Param #
=================================================================
rescaling_1 (Rescaling)      (None, 224, 224, 3)       0
_________________________________________________________________
conv2d (Conv2D)              (None, 224, 224, 16)      448
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 112, 112, 16)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 112, 112, 32)      4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 56, 56, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 56, 56, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 28, 28, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 50176)             0
_________________________________________________________________
dense (Dense)                (None, 128)               6422656
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 387
=================================================================
Total params: 6,446,627
Trainable params: 6,446,627
Non-trainable params: 0
```

Prédiction de notre modèle :

Nous avons pu tester la prédiction de notre modèle en temps réel puis avec une image statique :

Voici quelques exemples de résultats :

Prédiction par image statique :

Prédiction en temps réel :