ENSAM
ÉCOLE NATIONALE SUPÉRIEURE D'ARTS ET MÉTIERS
UNIVERSITÉ HASSAN II DE CASABLANCA

Royaume du Maroc
Ministère de l'Enseignement Supérieur,
de la Recherche Scientifique et de l'Innovation

DÉTÉCTION DES ÉMOTIONS

Réalisé par:

SALMA EZZAYDY

Sommaire

Introduction

L'objectif principal de ce projet est de développer un système permettant de détecter les émotions à partir de données visuelles en utilisant un modèle en Deep Learning. Le Deep Learning, une technique d'apprentissage automatique avancée, est utilisé pour entraîner un réseau de neurones à reconnaître et classifier les émotions humaines.

Le rapport est organisé en trois sections principales. Tout d'abord, il présente le contexte du projet ainsi que ses enjeux. Ensuite, il décrit en détail les différentes tâches réalisées, telles que la collecte de données, le prétraitement des données, la conception et l'entraînement du modèle en Deep Learning. Enfin, le rapport présente les résultats obtenus jusqu'à présent et offre des perspectives d'amélioration pour des travaux futurs.

Ce projet revêt une importance significative dans le domaine de la détection des émotions, car il ouvre la voie à des applications potentielles dans des domaines tels que la reconnaissance vocale, l'interaction homme-machine et l'analyse des sentiments. Les résultats et les conclusions présentés dans ce rapport serviront de base pour de futures recherches et développements dans ce domaine prometteur.

I. Analyse du projet

i) Choix du sujet

Une réflexion approfondie sur l'importance croissante de comprendre et d'interagir avec les émotions humaines a conduit au choix du sujet de détection des émotions par un modèle de Deep Learning.

La détection des émotions est bénéfique dans de nombreux domaines. Par exemple, la capacité d'identifier les émotions présentes dans la parole peut améliorer la compréhension des intentions et des états émotionnels des locuteurs dans le domaine de la reconnaissance vocale. Une détection précise des émotions dans le domaine de l'interaction homme-machine permet de personnaliser les réponses et les expériences en fonction de l'état émotionnel de l'utilisateur. De plus, la détection des émotions peut aider à comprendre les réactions des gens à des stimuli particuliers dans des domaines tels que la publicité, le marketing et la recherche en sciences sociales.

Le choix d'utiliser un modèle de Deep Learning pour la détection des émotions reposent sur sa capacité à apprendre des représentations complexes à partir de données brutes, permettant ainsi de saisir les caractéristiques subtiles et complexes liées aux émotions. Les capacités de calcul et d'apprentissage profondes offertes par les réseaux de neurones profonds peuvent être exploitées pour améliorer la précision et la fiabilité des modèles de détection des émotions.

ii) Cahier de charge

Voici le cahier des charges simplifié que nous avons créé pour le projet de détection des émotions en utilisant le Deep Learning :

- L'objectif :
  - Créez un modèle de Deep Learning capable de reconnaître et de classer les émotions dans des données visuelles.

- Spécifications techniques :
  - Utilisez des images comme données d'entrée.
  - Identifier et classer les émotions à détecter (joie, tristesse, colère, etc.).

- Méthodologie :
  - Recueillir et préparer un ensemble de données étiquetées contenant diverses illustrations d'émotions.
  - En utilisant les données préparées, concevoir et entraîner un modèle de Deep Learning comme un réseau de neurones convolutif ou récurrent.
  - Évaluer la performance du modèle en utilisant des métriques comme l'exactitude, la matrice de confusion, etc.

- Livrable :
  - Proposez un modèle de détection des émotions entraîné qui peut prédire les émotions à partir de données récemment acquises.
  - Rapportez les résultats et les performances de l'évaluation du modèle.

- Contraintes :
  - Respecter les exigences de ressources, telles que la puissance de calcul et la capacité de stockage nécessaires pour l'entraînement du modèle.
  - Pour faciliter le développement et l'expérimentation, utilisez des bibliothèques et des Framework de Deep Learning populaires.

- Potentiels d'amélioration :
    - Déterminer des avenues potentielles d'amélioration, telles que l'application de techniques de Deep Learning avancées, l'augmentation de la taille de l'ensemble de données, etc.

iii) Contraintes et méthodologie du travail

a. Contraintes

L'utilisation du jeu de données FER-2013 avec 7 types d'émotions pour la détection des émotions peuvent présenter certains obstacles. Parmi autres on cite :

- Distribution des classes :

Dans le jeu de données, les émotions peuvent être réparties de manière inégale, avec certaines émotions ayant plus d'exemples que d'autres. Cela peut entraîner un déséquilibre des classes, ce qui peut affecter les performances du modèle en termes de détection d'émotions moins fréquentes

-> Solution : Data augmentation par GANs (réseaux génératifs antagonistes) ou autres méthodes

- La variation intra-classe:

Les variations dans les expressions faciales et les caractéristiques au sein de chaque catégorie émotionnelle peuvent présenter un obstacle pour la prédiction pour différentes catégories en utilisant ce dataset

-> Solution : Eviter l'over-fitting pour rendre le modèle plus robuste

- Occlusion :

Lorsque des objets tels que les mains couvrent partiellement ou complètement le

visage d'une personne, elle peut avoir un problème d'occlusion. La détection précise des émotions peut être rendue difficile car une partie importante des caractéristiques faciales nécessaires à l'analyse est dissimulée.

-> Solution : Apprentissage robuste

b. Méthodologie du travail : Transfer Learning

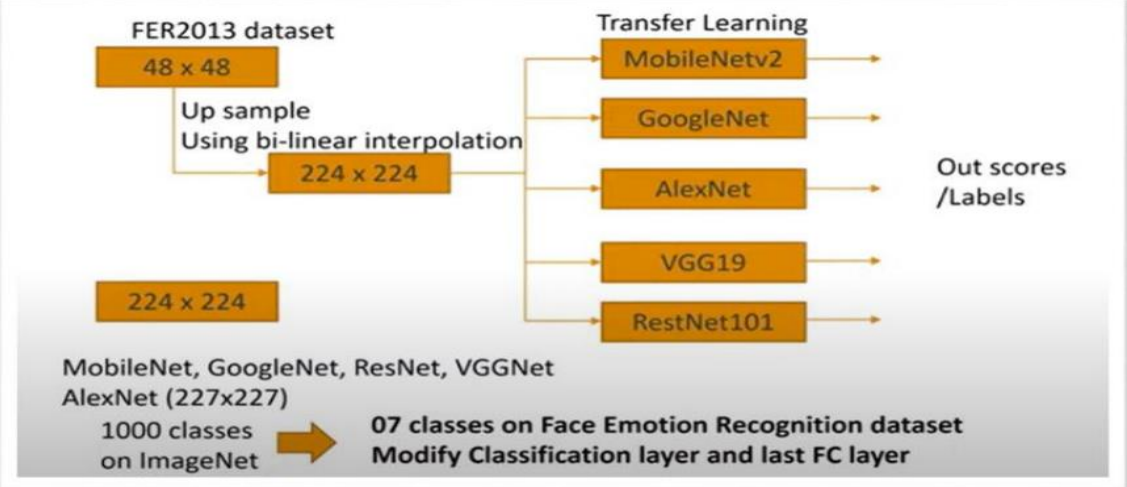

Figure 1 : Transfer Learning

Le transfert d'apprentissage permet de bénéficier des connaissances préalables du modèle pré-entraîné sur des tâches de vision par ordinateur, ce qui peut réduire le besoin de données d'apprentissage et améliorer les performances de la détection des émotions. De plus, cela évite de créer un modèle à partir de zéro, ce qui permet de gagner du temps et des ressources en utilisant les poids aboutis en d'autres architectures MobileNetv2,

AlexNet, VGG19...

Problème : L'utilisation directe d'architectures de modèles de Deep Learning telles que MobileNetv2, AlexNet et VGG-19 ..., nécessitent des images de résolution 224 x 224 pixels. Cependant, les images du dataset FER-2013 ont une résolution de 48 x 48 pixels.

Solution : Utilisation de l'interpolation bilinéaire ou d'autres méthodes d'ajustement de la taille des images en agrandissant ou en réduisant leur résolution.

-L'interpolation bilinéaire Elle fonctionne en tenant compte des pixels environnants pour déterminer les valeurs des pixels intermédiaires, ce qui permet de produire une nouvelle image avec la résolution souhaitée.

II. Réalisation du projet

i) Choix techniques

Python et ses librairies

Si le choix de Python pour l'apprentissage machine était évident, celui de la librairie à utiliser était plus difficile. De nombreux Framework d'apprentissage profond sont disponibles, et chacun présente ses propres avantages. Le leader **TensorFlow** est une bibliothèque open-source créée par Google pour l'apprentissage automatique. Elle est largement utilisée dans la construction et l'entraînement de modèles d'apprentissage automatique, en particulier des réseaux de neurones profonds. **TensorFlow** fournit une interface adaptable pour la création d'algorithmes d'apprentissage automatique et permet l'exécution de calculs sur des processeurs graphiques (GPU) pour accélérer le processus d'apprentissage.

Stockage des images

L'apprentissage automatique utilise fréquemment la sauvegarde d'un modèle entraîné en format H5 (HDF5). Ce format offre une portabilité indépendante du langage ou de la

bibliothèque utilisée, une structure hiérarchique pour gérer les modèles complexes, la possibilité de compression pour réduire la taille du fichier et la compatibilité avec de nombreux outils et bibliothèques d'apprentissage automatique.

Google Colaboratory

Tout entraînement de réseau de neurones nécessite une forte puissance de calcul. En particulier, notre système est basé sur un réseau convolutionnel profond, ce qui implique un grand nombre de poids à entraîner. C'est pourquoi nous avons décidé d'utiliser la plateforme Google Colaboratory. Il s'agit d'un environnement de notebook Jupyter qui s'exécute directement dans le Cloud et a l'avantage de mettre des GPU à disposition des utilisateurs. La plupart des librairies y sont préinstallées, et il est possible d'en ajouter simplement. Ainsi, nous avons pu utiliser la plateforme pour entraîner aussi bien des architectures Keras.

La plateforme est liée à Google Drive, il nous a donc suffit d'uploader nos jeux de données pour pouvoir les utiliser, et d'exporter nos réseaux entraînés en fin de session. Cette technologie nous a permis de gagner énormément de temps.

ii) Apprentissage machine

a. Pré-traitement des données

Le prétraitement des données dans ce code consiste à charger les images à partir d'un répertoire spécifié, les redimensionner à une taille donnée (224x224), et les stocker dans une liste appelée training_data avec leur classe correspondante.

-Les images sont importées du drive

```
directory = "/content/drive/MyDrive/data/train"
```

```
Classes = ["surprised", "sad", "neutral", "happy", "fearful", "angry", "disgusted"]
```

- Les images sont lues en utilisant OpenCV (**`cv2.imread`**)

```
for category in Classes:
    path = os.path.join(directory, category)
    for img in os.listdir(path):
        img_array = cv2.imread(os.path.join(path, img))
        plt.imshow(cv2.cvtColor(img_array, cv2.COLOR_BGR2RGB))
        plt.show()
        break
    break
```

- Puis redimensionnées à l'aide de la fonction **`cv2.resize`**

```
img_size = 224
new_array = cv2.resize(img_array, (img_size, img_size))
plt.imshow(cv2.cvtColor(new_array, cv2.COLOR_BGR2RGB))
plt.show()
```

- Ces dernières sont ajoutées à **`training_data`** sous la forme d'une paire contenant l'image et son étiquette de classe.

```
training_data = []

def create_training_data():
    for category in Classes:
        path = os.path.join(directory, category)
        class_num = Classes.index(category)
        for img in os.listdir(path):
            try:
                img_array = cv2.imread(os.path.join(path, img))
                new_array = cv2.resize(img_array, (img_size, img_size))
                training_data.append([new_array, class_num])
            except Exception as e:
                pass
```

- Après avoir parcouru toutes les images dans chaque classe, les données sont mélangées aléatoirement à l'aide de **`random.shuffle`**

```
import random

random.shuffle(training_data)
```

- Ensuite, les images sont transformées en un tableau numpy (X) et les valeurs sont normalisées en divisant les valeurs par 255 afin de les ramener dans la plage [0, 1]. De plus, les étiquettes de classe sont converties en un tableau numpy (Y).

```
X = []
y = []

for features,label in training_data:
    X.append(features)
    y.append(label)

X = np.array(X).reshape(-1, img_size, img_size, 3)
```

```
X = X / 255
```

```
Y = np.array(y)
```

b. Architecture de réseau de neurones

L'architecture de ce réseau de neurones combine les avantages du modèle MobileNetV2 pré-entraîné avec des couches supplémentaires destinées à la détection des émotions.

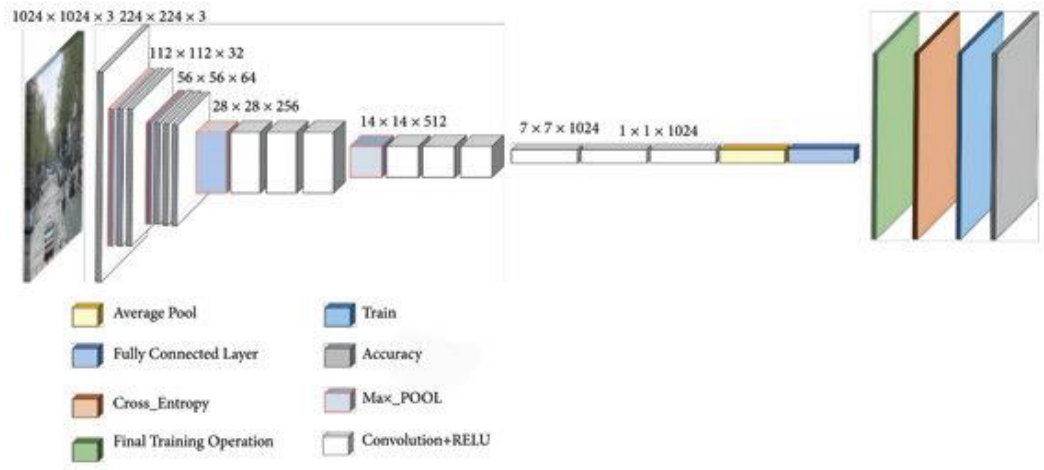

Figure 2 : The architecture of MobileNetV1

-Les couches de MobileNetV2 sont utilisées dans cette architecture comme un extracteur de caractéristiques efficace. La première couche du modèle reçoit l'input initial et l'output est récupéré à partir de la dernière couche. Cela permet de capturer les informations cruciales des images d'entrée et de les présenter dans un espace plus abstrait.

Chargement du modèle MobileNetV2 :

```
model=tf.keras.applications. MobileNetV2()
model.summary()
```

```
base_input =model.layers[0].input ## input
base_output= model.layers[-2].output
base_output
```

-Des couches supplémentaires sont ensuite ajoutées pour ajuster le modèle pour détecter les émotions :

=>Des couches denses (Fully Connected) avec des fonctions d'activation ReLu permettent d'apprendre des représentations non linéaires et de capturer des motifs complexes émotionnels.

Ces couches supplémentaires comprennent une couche Dense de 128 neurones suivie d'une activation ReLU, puis une autre couche Dense de 64 neurones avec une activation ReLU

```
final_output=layers.Dense (128) (base_output) ## adding new Layer, after the output of global pooling Layer
final_ouput=layers.Activation ('relu') (final_output) ## activation function
final_output=layers.Dense (64) (final_ouput)
final_ouput=layers.Activation( 'relu') (final_output)
```

=>La dernière couche dense de 7 neurones est configurée avec une activation Softmax pour classer les émotions selon sept classes distinctes.

```
final_output=layers.Dense (7, activation='softmax') (final_ouput) ## my classes are 07

final_output ## ouput
```

-Et finalement on obtient notre modèle combiné :

```
new_model =keras.Model(inputs=base_input, outputs=final_output)

new_model.summary()
```

Ce modèle est compilé avec une fonction de perte de sparse_categorical_crossentropy", un optimiseur "adam" et des métriques d'évaluation de la précision. Il est ensuite entraîné sur un ensemble de données (X, Y) pendant 25 époques.

iii) Résultats Obtenus

Le modèle est sauvegardé au format "Final_model_95p07.h5" après l'entraînement. Ensuite, il est chargé du fichier sauvegardé. Une image (cadre) est chargée d'un fichier et transformée en niveaux de gris. L'algorithme de détection de visage (faceCascade) est utilisé pour effectuer une détection de visage à partir d'un fichier XML fourni. Les visages détectés sont encadrés sur l'image, et un sous-ensemble de l'image contenant le visage est extrait.

La classification se fait en se basant sur les conditions suivantes :

```
if np.argmax(Predictions) == 0:
    status = "Surprised"
elif np.argmax(Predictions) == 1:
    status = "Sad"
elif np.argmax(Predictions) == 2:
    status = "Neutral"
elif np.argmax(Predictions) == 3:
    status = "Happy"
elif np.argmax(Predictions) == 4:
    status = "Fearful"
elif np.argmax(Predictions) == 5:
    status = "Angry"
elif np.argmax(Predictions) == 6:
    status = "Disgusted"

print(status)
```

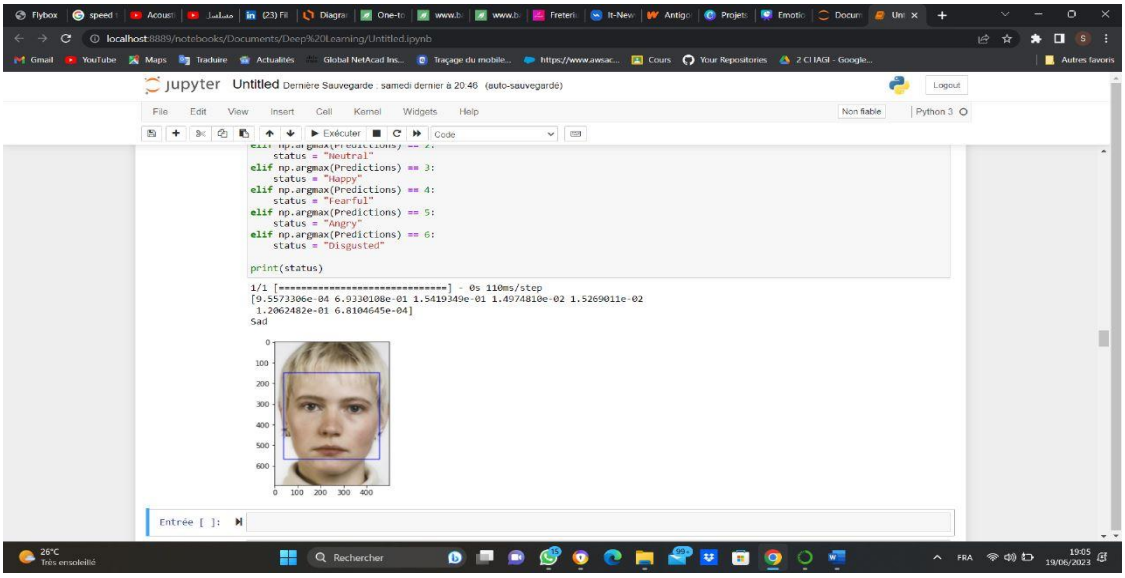

Figure 3 : Prédiction n°1 obtenue

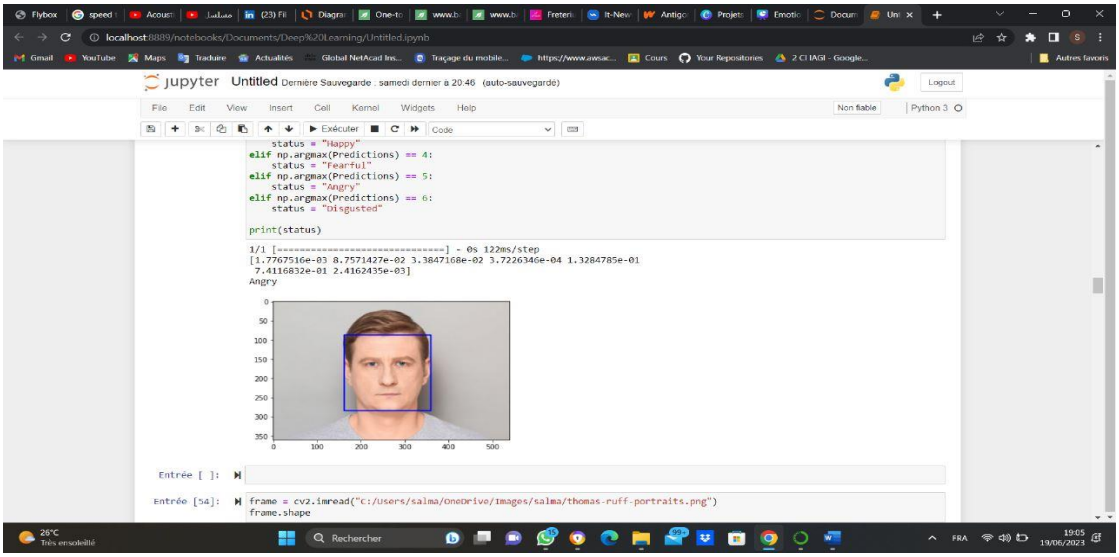

Figure 4 : Prédiction n°2 obtenue

Conclusion

Enfin, ce travail montre l'utilisation d'un modèle de détection des émotions basé sur MobileNetV2. Le modèle est entraîné avec succès sur la base de données FER-2013 et peut prédire avec précision les émotions des visages. Les performances du modèle ont été améliorées grâce à l'utilisation de techniques de prétraitement des données, de transfert d'apprentissage et d'interpolation bilinéaire

.

L'exploration de différentes architectures de réseaux neuronaux pour la détection des émotions, l'utilisation de techniques d'augmentation de données pour améliorer la généralisation du modèle, l'application de l'apprentissage actif pour choisir les échantillons les plus informatifs et l'expérimentation avec d'autres ensembles de données pour évaluer la performance du modèle dans des contextes plus diversifiés sont quelques extensions possibles de ce sujet. De plus, l'utilisation de techniques d'interprétabilité pour comprendre les facteurs et les caractéristiques les plus influents dans la prédiction des émotions pourrait être une direction de recherche intéressante.